UNIVERSIDAD
DE MÁLAGA

Programa de Doctorado
Tecnologías Informáticas
Departamento de Lenguajes y Ciencias de la Computación

Tesis Doctoral:
Algoritmos Meméticos para la Resolución de Problemas Combinatorios de Satisfacción con Restricciones y con Simetrías

Autor: Rodríguez Rueda, David.

Director: Dr. Antonio J. Fernández Leiva
Co-Director: Dr. Carlos Cotta Porras

Málaga, Mayo 2020

AUTOR: David Rodríguez Rueda

http://orcid.org/0000-0002-4478-4949

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD DE LA TESIS PRESENTADA PARA OBTENER EL TÍTULO DE DOCTOR:

D. **David Rodríguez Rueda**
Estudiante del programa de doctorado **Tecnologías Informáticas** de la Universidad de Málaga, autor de la tesis, presentada para la obtención del título de doctor por la Universidad de Málaga, titulada:

Algoritmos Meméticos para la Resolución de Problemas Combinatorios de Satisfacción con Restricciones y con Simetrías

Realizada bajo la tutorización del Dr. Carlos Cotta y dirección del Dr. Antonio J. Fernández Leiva.

DECLARO QUE:

La tesis presentada es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, conforme al ordenamiento jurídico vigente (Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), modificado por la Ley 2/2019, de 1 de marzo.

Igualmente asumo, ante a la Universidad de Málaga y ante cualquier otra instancia, la responsabilidad que pudiera derivarse en caso de plagio de contenidos en la tesis presentada, conforme al ordenamiento jurídico vigente.

En Málaga, 15 de Mayo de 2020.

Fdo.: David Rodríguez Rueda

Dr. D. Antonio J. Fernández Leiva y Dr. D. Carlos Cotta Porras, Profesores Titulares del Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga

Certifican:

Que D. David Rodríguez Rueda, Licenciado en Informática por la Universidad Católica del Táchira de Venezuela, ha realizado en el Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga el trabajo correspondiente a su Tesis Doctoral titulada:

Algoritmos Meméticos para la Resolución de Problemas Combinatorios de Satisfacción con Restricciones y con Simetrías

Revisado el presente trabajo, estimamos que puede ser presentado al tribunal que ha de juzgarlo, y autorizamos la presentación de esta Tesis Doctoral en la Universidad de Málaga.

En Málaga, 15 de Mayo de 2020

Fdo.: Dr. Antonio J. Fernández Leiva
Director de la tesis
Titular de Universidad
Dpto. de Lenguajes y Ciencias de la Computación
Universidad de Málaga

Fdo.: Dr. Carlos Cotta Porras
Director de la tesis
Catedrático de Universidad
Dpto. de Lenguajes y Ciencias de la Computación
Universidad de Málaga

Algoritmos Meméticos para la Resolución de Problemas Combinatorios de Satisfacción con Restricciones y con Simetrías

David Rodríguez Rueda

Mayo, 2020

Resumen

Este trabajo investigativo se enfoca en la resolución de problemas complejos de optimización, principalmente con el objetivo de prestar atención al modelado y ajuste de diversas técnicas metaheurísticas con el fin de conseguir resultados de alta calidad en la resolución de problemas de optimización con simetrías. La principal motivación para el desarrollo de esta investigación ha sido presentar una metodología que reúna las líneas principales que se deben seguir al momento de abordar este tipo de problemas. Es por ello que hemos utilizado un enfoque incremental de corte integrativo que involucre aspectos relacionados con la construcción o aplicación de modelos adecuados para la representación de los problemas objeto de estudio, considerando diferentes formas de representación enmarcados en la teoría de la dualidad, e intentando emplear algún mecanismo que permita reducir el paisaje de búsqueda (esto es, ruptura de simetrías). Se ha empleado un esquema de colaboración utilizando diferentes modelos de arquitectura, así como algoritmos híbridos evolutivos con diferentes métodos de búsqueda local. Además, consideraremos la utilización de un enfoque colaborativo entre las metaheurísticas propuestas a través de la definición de topologías de comunicación entre los diferentes componentes que participan en dicho esquema. Este enfoque propuesto se engloba dentro del paradigma de los algoritmos meméticos y ha sido validado empíricamente por medio dos problemas de optimización combinatoria que presentan un alto grado de complejidad, cuyos espacios de búsqueda son ricos en lo que se refiere a presencia de estados simétricos, y que han sido tradicionalmente formulados y resueltos por medio de técnicas de programación lineal entera (ILP) y programación con restricciones (CP). A tal fin, se presenta un extenso análisis de los resultados obtenidos con el fin de validar la adecuación y la eficacia de las técnicas metaheurísticas propuestas. Dicho análisis incluye un estudio del empleo de diferentes arquitecturas cooperativas que utilizan un variado número de algoritmos metaheurísticos e híbridos, apoyándonos en métodos estadísticos propuestos para la evaluación de este tipo de algoritmos.

Hemos podido constatar que los algoritmos cooperativos, especialmente los que han sido basados en el modelo multi-agente, muestran un rendimiento muy prometedor, ubicado por encima de su

respectivo modelo individual. Además, queda claro que el modelo presentado nos proporciona información muy valiosa, ya que cada una de estas técnicas tiene una visión diferente del espacio de búsqueda, lo que permite a cada uno de los agentes aplicar una combinación de diferentes parámetros de exploración, lo que les permite escapar de los peligrosos óptimos locales. Estos resultados reafirman la validez de estos esquemas meméticos para abordar problemas de satisfacción con restricciones con simetrías, y allanan el camino al desarrollo de esquemas más sofisticados para la resolución de problemas en este dominio.

Palabras Clave: Metaheurísticas, hibridación, modelos cooperativos, sistemas multiagente, algoritmos meméticos, búsquedas locales, representación alternativa, ruptura de simetrías, espacio de búsqueda

Abstract

This research work focuses on solving complex optimization problems, paying particular attention to the modeling and adjustment of diverse metaheuristic techniques in order to achieve high quality results in solving such problems. The main motivation for the development of this research has been to present a methodology covering the most important characteristics that should be featured when addressing these problems. To this end, we have used an incremental integrative approach that involves aspects related to the construction and application of suitable models for the representation of the problems under scrutiny, considering different representations based on the theory of duality, and trying to employ mechanisms for simplifying the search landscape (i.e., symmetry breaking). We have considered the use of a collaboration scheme relying on different architecture models as well as on evolutionary hybrid algorithms with different local search methods. Furthermore, we have considered the use of a collaborative approach among the metaheuristics proposed through specific communication topologies among algorithms. This proposal thus falls within the paradigm of memetic algorithms and has been empirically validated by means of two combinatorial optimization problems that present a high degree of complexity, whose search spaces are rich in symmetric states, and that have been traditionally solved via integer linear programming (ILP) and constrain programming(CP). We present an extensive analysis of the results obtained in order to measure the performance of the proposals studied incorporating different cooperative architectures with different metaheuristics and hybrid algorithms, relying on statistical methods proposed for the evaluation of this type of algorithms.

We have been able to verify that cooperative algorithms, especially those that have been based in the multi-agent model, show a very promising performance, well above the corresponding individual models. In addition, it is clear that the model presented provides us very valuable information, since each of these techniques has a different view of the space of search, this allows each of the agents to apply a combination of different parameters of exploration, hence allowing them to escape from local optima. These results support the feasibility of these memetic approaches for tackling satisfaction problems with constraints with symmetries, and pave the way for developing more sophisticated methods to solve problems in this realm.

Keywords: Metaheuristics, hybridization, cooperative models, multi-agent systems, memetic algorithms, local search, alternative representation, symmetry breaking, search space

Agradecimientos

A la inteligencia suprema que mantiene el orden del universo y a la cual comúnmente llamamos DIOS.
A Alejandro y Gabriela, fuente inagotable de amor y de inspiración.
A Antonio y Carlos, por su incansable colaboración y apoyo.

Publicaciones

A continuación se mencionan las diferentes publicaciones que han sido producto del trabajo de investigación que estamos presentando y que avalan esta tesis doctoral.

- **Revistas en el Journal Citation Reports (JCR)**:
  1. David Rodríguez Rueda, Carlos Cotta, Antonio J. Fernández-Leiva, Metaheuristics for the Template Design Problem: Encoding, Symmetry and Hybridisation, Journal of Intelligent Manufacturing (Mayo 2020). Aceptado para su publicación.
     - Impact Factor: 4.311 (JCR 2019)
     - COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE (32/136, Q1),
     - ENGINEERING MANUFACTURING (11/50, Q1)
     - TERCIL: T1
     - CUARTIL: Q1
     - DOI: 10.1007/s10845-020-01587-w
     - First Published online: 20 June 2020 - https://link.springer.com/article/10.1007/s10845-020-01587-w
  2. David Rodríguez Rueda, Carlos Cotta, Antonio J. Fernández-Leiva, Memetic collaborative approaches for finding balanced incomplete block designs, *Comput. Oper. Res.*, 114 (2020)
     - Impact Factor: 3.424 (JCR 2019)
     - COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS (34/109, Q2)
     - OPERATIONS RESEARCH & MANAGEMENT SCIENCE (21/83, Q2)
     - ENGINEERING, INDUSTRIAL (14/48, Q2)
     - TERCIL: T1
     - CUARTIL: Q2
     - DOI: 10.1016/j.cor.2019.104804

- **Otras revistas**:
  1. David Rodríguez Rueda, Carlos Cotta, and Fernández-Leiva Antonio J. A memetic algorithm for designing balanced incomplete blocks. *International Journal of Combinatorial Optimization Problems and Informatics*, 2(1):14-22, 2011.
  2. David Rodríguez Rueda, Enrique Darghan, and Julio Monroy. A multi-agent proposal in the resolution of instances of BIBD. *Revista Colombiana de Estadística*, 39(2):267-280, July 2016.

- **Actas de Congresos**:
  1. David Rodríguez Rueda, Carlos Cotta, and Fernández-Leiva Antonio J. Finding balanced incomplete block designs with metaheuristics. In Evolutionary Computation in Combinatorial Optimization, 9th European Conference, EvoCOP 2009, Tübingen, Germany, April 15-17, 2009. Proceedings, volume 5482 of Lecture Notes in Computer Science, pages 156-167. Springer, 2009.
  2. David Rodríguez Rueda, Carlos Cotta, and Fernández-Leiva Antonio J. Una comparativa de metaheurísticas para el problema del diseño de bloques incompletos equilibrados. In VI Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB09), pages 191-197, 2009.
  3. David Rodríguez Rueda, Carlos Cotta, and Fernández-Leiva Antonio J. Un problema de diseño de plantillas: Un enfoque metaheurístico basado en búsqueda local. In VI Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y bioinspirados (MAEB2010), pages 743-750, Garceta, Valencia, 2010.
  4. David Rodríguez Rueda, Carlos Cotta, and Fernández-Leiva Antonio J. The template design problem: A perspective with metaheuristics. In Karl K. Sabelfeld and Ivan Dimov (eds.), Eighth IMACS Seminar on Monte Carlo Methods, pages 181-192, Borovets, Bulgaria, 2011, 2012. De Gryuter, Berlin/Boston.

Además de las publicaciones anteriores, los siguientes trabajos están en proceso de revisión o de preparación en el momento de escribir esta tesis doctoral:

1. David Rodríguez Rueda, Carlos Cotta, and Fernández-Leiva Antonio J. Data for the search of 2-designs: solutions and metaheuritics code. *Data in Brief*, 2020. En preparación.

Índice de Contenidos

Lista de Figuras

Lista de Tablas

Acrónimos

ACO Optimización basada en Colonia de Hormigas (*Ant Colony Optimization*)
AI Inteligencia Artificial (*Artificial Intelligence*).
BF Cambio de bit (*Bit Flip*).
BIBD Diseño de Bloques Incompletos Balanceados (*Balanced Incomplete Block Designs*).
C&P Corte y empaquetado (*Cutting and Packing*)
CM Modelos Cooperativos (*Cooperative Models*).
COP Problema de Optimización Combinatoria (*Combinatorial Optimization Problem).*
CP Programación con Restricciones (*Constraint Programming*)
CSP Problemas de satisfacción con restricciones (*Constraint Satisfaction Problem*)
DM Modelo Dual (*Dual Model*).
EA Algoritmo Evolutivo (*Evolutionary Algorithm*).
GA Algoritmo Genético (*Genetic Algorithm*).
HC Hill Climbing.
ILP Programación Lineal Entera (*Integer Linear Programming*)
IP Programación Entera (*Integer Programming*)
LP Programación Lineal (*Linear Programming*)
LS Búsqueda Local (*Local Search*).
MA Algoritmo Memético (*Memetic Algorithm*).
NFL No Free Lunch.
OR Investigación de Operaciones (*Operations Research*).
PM Modelo Primal (*Primal Model*).
SB Ruptura de Simetrias (*Symmetry Breaking*).
SW Intercambio (*Swap*).
TDP Problemas del Diseño de Plantillas (*Template Design Problems*).
TS Búsqueda Tabú (*Tabu Search*).

Prefacio

Podemos considerar que los avances de la humanidad en la mayoría de las ciencias (si no es que en todas) se han logrado gracias a un factor común: la búsqueda. Por ejemplo, un grupo de científicos en un laboratorio emprenden una ardua labor para encontrar el mejor medicamento para curar (o frenar) una enfermedad. El proceso que estos científicos emplean pudiéramos enmarcarlo (en forma muy genérica) en aspectos como: conocimiento de la enfermedad que desean curar, conocimiento de los fármacos que pueden resultar exitosos, mucha experimentación (que puede ser exhaustiva, aproximada, estocástica, etc.), observación y obviamente pruebas en seres vivos (y probablemente algún otro proceso que se me olvida mencionar). Estos aspectos los podemos considerar como: búsqueda de información, búsqueda de conocimiento o búsqueda de sabiduría. Sin embargo, estos tres conceptos son complementarios y aumentativos, es decir, no puede haber sabiduría si no hay conocimiento y no habrá conocimiento si no hay información. Luego, consideremos que lo que busca el científico es *conocimiento*. Si además suponemos que el grupo de científicos emplea un conjunto de herramientas diversas (seguramente entre ellas: recursos computacionales), y que el principal objetivo es encontrar el mejor medicamento en el menor tiempo posible, estaríamos reafirmando la hipótesis de que todo está enmarcado en un proceso de búsqueda. Por supuesto que cada proceso de búsqueda posee un objetivo particular dependiendo del problema que deseamos resolver. En el caso que nos ocupa en la presente tesis doctoral estamos apuntando hacia la resolución de problemas de búsqueda en el ámbito de la optimización combinatoria, cuya principal herramienta son los recursos computacionales.

Como se sabe, el empleo de técnicas para realizar procesos de búsqueda no es nada nuevo y se remonta a los inicios de la humanidad con los primeros pobladores del planeta, pero para no ir muy lejos, pensemos en las cuatro últimas décadas, cuando Fred Glover introdujo el término *Metaheurísticas* por el año 1986. Desde entonces este tipo de técnicas ha sido objeto de estudio de un amplio número de investigadores y ha sido aplicada con éxito en la resolución de múltiples problemas. El objetivo de esta investigación se centra en el estudio de algunas de estas técnicas y la propuesta de una metodología que pueda ser de utilidad para futuros investigadores que deseen hacer uso de este tipo de algoritmos. En este sentido, en este trabajo se presenta una serie de conceptos y revisiones del estado de la cuestión sobre las técnicas aproximadas que incluyen algoritmos metaheurísticos, meméticos y colaborativos, así como también, se aborda el tema de la optimización combinatoria en forma general, los conceptos básicos de simetrías, los tipos de simetrías más comunes y una revisión de algunas de las técnicas utilizadas para la ruptura de las mismas. Asimismo, se introducen algunos conceptos sobre la teoría de la dualidad y las representaciones alternativas en la resolución de problemas computacionales.

Los conceptos anteriores se ponen a prueba sobre problemas de optimización considerados difíciles de resolver. Así, se muestran resultados experimentales sobre los mismos aplicando técnicas de

búsqueda local basadas en trayectoria, así como métodos poblacionales e hibridaciones. Para imprimir mayor eficiencia y efectividad de los algoritmos empleados se ha propuesto un mecanismo colaborativo entre los diferentes actores que intervienen en los procesos de búsqueda. También se hace uso de representaciones alternativas para el modelado de los problemas que serán atacados, agregando, además, mecanismos que permiten frenar las exploraciones de zonas con posibles soluciones que presenten simetrías. Para probar el rendimiento de las propuestas se han utilizado dos problemas de la literatura científica, que han resultado difíciles de resolver y que además presentan un alto grado de simetrías. Tras la realización de una extensa experimentación, se realiza un análisis estadístico que permite dirigir el trabajo para la incorporación de ideas que coadyuven a lograr una amplia gama de soluciones de alta calidad en tiempos computacionales altamente aceptables.

Todo este trabajo nos permitirá proponer una metodología robusta que puede ser aplicada, en forma general, para abordar diversos problemas de optimización combinatoria con satisfacción, que contienen restricciones y que presentan simetrías. Nos enfocamos especialmente en problemas cuya formulación se ajusta, de forma natural, a un modelo de programación lineal entera (ILP, Integer Linear Programming) y que han sido tradicionalmente resueltos por métodos de programación con restricciones (CP, Constraint Programming). Al verificar la validez de las metaheurísticas para resolver u optimizar el tipo de problemas, mencionados anteriormente, este trabajo arroja luz sobre técnicas alternativas a los métodos de ILP y CP en este dominio y allana el camino para el desarrollo de modelos más sofisticados para dicho objetivo.

CAPÍTULO 1

INTRODUCCIÓN

En el lenguaje cotidiano, optimizar lo podemos considerar como el tratar de mejorar un determinado proceso; sin embargo, en el contexto más especializado (el ámbito científico), la optimización es el proceso de tratar de obtener la mejor solución posible para un determinado problema, minimizando el empleo de recursos y principalmente mejorar el tiempo para alcanzar el objetivo propuesto. En un problema de optimización podemos encontrar un conjunto de posibles soluciones y es necesario poder determinar cuál de ellas resulta ser la más efectiva, es decir, el objetivo es poder encontrar la mejor. De forma más precisa, estos problemas se pueden expresar como encontrar el valor de las variables de decisión, que involucra el problema a resolver, para los que una determinada función objetivo alcanza su valor máximo (maximización) o mínimo (minimización). Los valores de las variables en muchas ocasiones están sujetos a una serie de restricciones. La existencia de un amplio conjunto de problemas difíciles de ser optimizados que aparecen en la práctica y que requieren ser resueltos de manera más eficiente, ha evidenciado el desarrollo de técnicas eficientes para tratar de encontrar soluciones de alta calidad que utilicen los recursos computacionales de forma óptima. Este tipo de métodos, en los que el tiempo de ejecución de los procesos es tan importante como el grado de optimalidad de la solución obtenida, se les suele encontrar con el nombre de heurísticos, metaheurísticos, meméticos o simplemente aproximados y se han utilizado a lo largo de los últimos años para resolver complejos problemas de optimización combinatoria que aparecen en la empresa, las ciencias económicas, las industrias, la ingeniería y muchas otras áreas. Se han desarrollado muy rápidamente desde principios de los años ochenta para resolver un extenso y variado grupo de problemas.

Este capítulo se estructura de la siguiente forma: En la Sección 1.1 se describe la esencia del problema que deseamos abordar en este trabajo; en la Sección 1.2 se presentan los objetivos del trabajo; en la Sección 1.3 se describe la metodologia utilizada; en la Sección 1.4 se presentan las contribuciones, finalmente se presenta la distribución de la tesis.

1.1 El Problema

El ser humano se enfrenta, muy a menudo, a resolver situaciones que lo obligan a recurrir a procesos de toma de decisiones ante una gran gama de problemas, en los que no es suficiente el uso del sentido común y por ende hay que acudir a la ciencia, para poder enfrentarlos e intentar darles solución o al

menos llegar lo más cerca posible a esta. En todo tipo de problemas, para lograr su resolución, existen valores ideales o al menos umbrales de valores que pueden ser cuantificables [317]. Una categoría importante y compleja de estos problemas son aquellos en los que se desea maximizar o minimizar, dicho de otra forma: *optimizar*, una determinada operación o proceso, el cual depende de un número finito de datos de entrada (variables de entrada), y donde dichas variables pueden estar relacionadas entre sí por medio de parámetros condicionados (por ejemplo con una o más restricciones), o por el contrario, ser independientes unas de otras [41, 74, 261]. Este tipo de problemas requieren de un gran esfuerzo para abordarlos y así poder determinar la mejor forma de resolverlos, es decir, se deberá estudiar muy a fondo las particularidades y características que posee el problema para lograr diseñar un modelo que permita optimizar el objetivo planteado, generalmente por medio de la identificación de las variables específicas de éste, las restricciones existentes, la relación costo/beneficio en la toma de decisiones admisibles para cada variable y la elaboración de una función objetivo acorde a la naturaleza de la situación planteada. El conjunto de todos estos elementos definen un problema de *optimización combinatoria* [49, 56, 236], es decir, un problema de optimización combinatoria es aquel en el cual el espacio de búsqueda es discreto.

En matemática aplicada podemos decir que la optimización combinatoria se asocia con la rama de la optimización, y tiene una estrecha relación con la teoría de los algoritmos, la investigación de operaciones y la teoría en complejidad computacional (en las ciencias de la computación). Además, se relaciona con la inteligencia artificial e ingeniería de software, entre otras áreas que podríamos mencionar. Los algoritmos de optimización combinatoria son capaces de resolver instancias de aquellos problemas que han resultado, a lo largo del tiempo, difíciles de abordar utilizando una variedad de herramientas pertenecientes a alguna ciencia (o varias) en particular; normalmente se requiere la exploración del paisaje de soluciones (que suele ser muy grande) para dichas instancias. Apoyándonos en la teoría de la complejidad computacional se abre una brecha para entender que los procesos de optimización combinatoria resultan de gran importancia. Estas técnicas aplicadas a la optimización combinatoria están relacionados frecuentemente con los denominados problemas de corte NP-duros. Debido a la gran dificultad para el abordaje de este tipo de problemas, en términos generales, resulta muy complejo lograr resolverlos de forma eficiente, pero la literatura presenta trabajos que muestran que es posible encontrar soluciones de muy buena calidad denominadas aproximadas, en las cuales la teoría de la complejidad sugiere que ciertas instancias (derivadas, de menor complejidad) de estos problemas pueden ser resueltas eficientemente. Estas instancias derivadas (y que tiene menor complejidad) a menudo muestran ramificaciones prácticas que resultan muy interesantes para la comunidad científica.

Los *Problemas de Optimización Combinatoria* (COP's) ocurren en diversas áreas tales como la programación lineal, teoría de números, teoría de grafos, inteligencia artificial, electrónica, economía, medicina y muchas otras áreas. Todos estos problemas, suelen ser formulados matemáticamente como la minimización o maximización de cierta función objetivo definida en el mismo dominio [85]. Históricamente la *optimización combinatoria* comienza con la *programación lineal* que tiene como base el aporte en un gran número de aplicaciones incluyendo, planificación y distribución de la producción, finanzas, asignación de recursos económicos, simulación de circuitos y sistemas de control. Un punto crucial para el avance de esta disciplina fue el desarrollo de software en programación entera y la disposición de la computación paralela [9]. Junto con la necesidad de resolver problemas de naturaleza real que contribuían a solucionar casos particulares en la industria y/o los negocios, también surgieron problemas de corte académico, que se emplean para realizar estudios más teóricos con el fin de proporcionar documentación sobre las diversas bondades de las técnicas computacionales que se pueden emplear para resolver este tipo de planteamientos, así podemos mencionar algunos casos

emblemáticos como el Problema de las N-reinas [4, 194], el Viajante de Comercio [138, 262, 324], Asignación Cuadrática [191, 205, 257] y Asignación de Frecuencias [61, 190, 224], entre otros.

Muchas han sido las propuestas para enfrentar y tratar de dar solución de una forma eficiente a los problemas de optimización combinatoria (COP's). Así, podemos encontrar técnicas que van desde los métodos exactos, que intentan simplificar el problema mediante el uso de modelos lineales, pasando por algoritmos voraces que actúan tratando de explorar la totalidad del espacio de búsqueda, como otros que utilizan las estrategia "divide y vencerás", entre los que podemos mencionar: Programación Lineal, técnicas de Ramificación y Poda (*Branch & Bound*), o Programación Dinámica, entre otras. No obstante, estas técnicas, en la mayoría de los casos, tienen un desempeño muy pobre cuando las instancias del problema considerado poseen un espacio de búsqueda de grandes proporciones, quedándose atrapadas en la exploración debido a que los recursos computacionales les resultan deficientes, sin hablar del tiempo de respuesta. Así, una de las grandes debilidades de estas técnicas las podemos encontrar en los elevados costes computacionales, los cuales aumentan a medida que los tamaños de los paisajes de búsqueda se hacen exponencialmente grandes, por lo que para ciertas configuraciones de los problemas combinatorios se hace improbable el uso de este tipo de técnicas de resolución. Debido a estas limitaciones fueron apareciendo otro tipo de estrategias que emplean el conocimiento del problema para abordarlo a las cuales se les suele llamar *heurísticas*. El objetivo de estas propuestas es el de guiar la búsqueda en las regiones más prometedoras del espacio de soluciones candidatas de un determinado problema.

Sin embargo, los métodos heurísticos dependen en gran medida del conocimiento específico del problema a resolver coartando la posibilidad de generalizar para poder aplicar el mismo procedimiento a otro tipo de problema de naturaleza similar, es decir que son métodos de resolución de propósito específico. Además, por ser métodos aproximados no garantizan que la posible solución encontrada sea la óptima. Existe diversidad de métodos heurísticos pero podemos mencionar los métodos constructivos suelen ser métodos muy rápidos, puesto que crean una solución partiendo de una configuración *vacía* a la cual se le va agregando nuevos componentes hasta lograr la configuración de una solución completa, que es el resultado de la aplicación del algoritmo. Podemos decir, que encontrar un heurístico constructivo, en muchos casos, es relativamente fácil, sin embargo, las soluciones arrojadas por este tipo de técnicas son de calidad muy pobre. En los últimos años se han utilizado otro tipo de técnicas con el objetivo de tratar de alcanzar resultados de mayor grado de calidad que los mostrados por los algoritmos heurísticos tradicionales y han aparecido bajo el nombre de *Metaheurísticos*. El término fue introducido por Fred Glover en 1986 [119], y es considerado como una estrategia de muy alto nivel, ya que emplean diversos métodos para evaluar los espacios de búsqueda. Podemos decir que las metaheurísticas son un esquema general *no determinista* que requiere ser configurado con los datos específicos del problema (representación de los individuos a candidatos de soluciones, los operadores empleados para su manipulación, etc.) y que permiten abordar aquellos problemas con paisajes de búsqueda que poseen tamaños considerables. En este tipo de técnicas es muy importante centrar el interés en el equilibrio correcto (normalmente dinámico) que existe entre el proceso de diversificación (o exploración) e intensificación (o explotación). Dentro de las técnicas metaheurísticas se distinguen dos tipos de algoritmos de búsqueda. Por una parte tenemos, las configuraciones "inteligentes"de los algoritmos de búsqueda local. La finalidad de estas estrategias es tratar de evitar de alguna forma los óptimos locales y enfocarse en otras regiones prometedoras del espacio de búsqueda. Algunos de los métodos que utiliza este tipo de estrategia es el que emplea la *búsqueda tabú* [120, 121, 125], la *búsqueda local iterada* [171, 184], la *búsqueda con entorno variable* [234, 242], y el *enfriamiento simulado* [158, 312]. Estas metaheurísticas (*basadas en trayectoria*), parten de una solución ya completa y, usando la definición de *vecindario*, exploran parte del espacio de búsqueda hasta lograr en-

contrar un óptimo. Se les denomina así puesto que el proceso empleado para encontrar una posible solución describe una trayectoria desde el punto de partida hasta la consecución de la solución final. Una de las desventajas de este tipo de técnicas es la inminente posibilidad de quedar atrapados en un óptimo local que no representa una solución de alta calidad; un mecanismo que se emplea para lograr salir de esta situación es reiniciar la búsqueda desde otro punto del espacio de soluciones seleccionado al azar. Por otro lado, están aquellos algoritmos metaheurísticos que hacen uso de conjuntos de soluciones (llamados poblacionales) que posteriormente son optimizadas de forma simultánea durante la búsqueda, aquí podemos mencionar los *algoritmos genéticos* [151]. Las técnicas basadas en población han sido consideradas como esquemas que conllevan al establecimiento de un procedimiento auto-adaptativo, donde el uso de múltiples soluciones permite mantener la diversidad del proceso de búsqueda y así disminuir el riesgo de converger hacia un óptimo local. Otras técnicas que utilizan la estrategia poblacional son: Optimización por Enjambre de Partículas [166], Optimización por Colonia de Hormigas [80] y Algoritmos de Estimación de Distribuciones [240], por mencionar algunos ejemplos. Para una clasificación más precisa podemos mencionar el trabajo realizado por Blum y Roli [27].

Las técnicas Evolutivas, en general, tienen su fundamentación en la *teoría de la evolución de las especies* de *Charles Darwin* [73], donde dada una población de individuos la presión ambiental causa una selección natural. El proceso de evolución es el resultado de la consideración de dos procesos primarios: la selección natural y la reproducción. La selección determina cuáles de los integrantes de la población deben sobrevivir hasta reproducirse. La reproducción garantiza la combinación (o mezcla) y recombinación de sus genes entre los descendientes. Si deseamos aplicar los algoritmos genéticos a un determinado problema de interés, se hace necesario diseñar una representación de las soluciones potenciales de éste como una serie de parámetros o componentes (a los que se les llama *genes*). Estos parámetros o elementos se deben reunir para lograr formar una secuencia (cadena) que viene a representar una posible solución (individuo); a la cual, también, se le denomina *cromosoma* o *genotipo*. Esto se logra mediante la representación del problema a través de un modelo matemático [3, 283]. Básicamente podemos mencionar que los parámetros más relevantes están referidos a la población inicial, que puede ser tomada configurando al azar distintas secciones del espacio de búsqueda, o guiada tomando ciertas características del problema; inclusive podría tomarse como población inicial individuos resultantes de la aplicación previa de algún otro algoritmo heurístico. Luego, la evolución se realiza por medio de un ciclo que genera nuevos individuos mediante el empleo de operadores genéticos como el *cruce* y la *mutación*. Cada individuo debe ser evaluado para determinar si es apto para poder permanecer en la población y para ello utiliza *la calidad* como elemento de medida de la bondad según el valor de la función objetivo [147]. El reemplazo de individuos puede ser *generacional*, en donde se reemplaza toda la población en cada generación; o mediante *estado-estacionario*, sólo pocos individuos deberán ser sustituidos en cada generación. Algunas de las bondades de este tipo de técnica es su capacidad de evitar los óptimos locales, además de que no se hace necesario conocimientos específicos del problema que se desea resolver. Se realizan modificaciones aleatorias en sus soluciones candidatas, para luego, por medio de la función de aptitud, se determina si las modificaciones introducidas producen algún tipo de mejora o no.

Los Algoritmos Genéticos (AGs), han resultado muy utilizados ya que poseen robustez. Sin embargo, poseen la particularidad de que pueden converger hacia zonas del espacio de búsqueda poco prometedoras, además que los tiempos de convergencia suelen ser muy cortos, así como el decaimiento de la diversidad de la población, dependiendo de los parámetros utilizados. Por esta razón surgió la motivación de realizar combinaciones de diferentes conceptos de las técnicas conocidas para obtener propuestas con mejor rendimiento que exploten las ventajas de las técnicas puras al operar en forma

individual [29, 256]. La combinación de estrategias que permitan la reducción de la complejidad del problema, y el mejoramiento de las soluciones son los enfoques más usados en las propuestas encontradas en la literatura actual sobre este tema y podemos encontrarlas bajo el nombre de *algoritmos híbridos*. Sobre las *técnicas híbridas* podemos encontrar autores que han realizado algún tipo de clasificación [62, 252]. Básicamente podemos encontrar tres formas de combinar las metaheurísticas para lograr la hibridación: combinar componentes de una metaheurística en otra, las búsquedas cooperativas, que consiste en que varios algoritmos intercambian información para alcanzar un mismo objetivo y la combinación de métodos aproximados con sistemáticos (completos) [27].

En la misma línea de las técnicas híbridas se han utilizado ampliamente los *Algoritmos Meméticos*, que se manifiestan de manera más común por medio de la combinación de las llamadas búsquedas trayectoriales (basadas en trayectoria) con las búsquedas poblacionales (basadas en población). Este término fue introducido por Moscato, refiriéndose a la evolución cultural como metáfora de la evolución de ideas y apoyándose en el concepto de "meme" que corresponde a una unidad de evolución debida al aprendizaje individual [217]. Podemos encontrar un nutrido número de propuestas que permiten sustentar las bondades de estas técnicas como se puede revisar en [64, 144, 218, 220, 226, 227]. En el mismo campo de ésta técnicas se ha utilizado también un tipo de propuesta que está enmarcada en la hibridación como lo son los Algoritmos *Multi-Meméticos* que se inspiran en la capacidad de automejora de los *memes*, logrando así un enfoque de búsquedas auto-adaptativas [173, 174, 230, 285].

En los últimos años se han empleado las técnicas híbridas llamadas *búsquedas cooperativas* (también denominados *modelos cooperativos*). Los algoritmos de búsqueda cooperativa consisten en la ejecución en paralelo de algoritmos de búsqueda con un nivel variable de comunicación entre los diferentes actores del proceso (generalmente se les llama *agentes*). Estos *agentes* pueden ser diferentes algoritmos de búsqueda o pueden ser instancias del mismo algoritmo que trabajan en los diferentes modelos utilizados o ejecutándose ajustando los diferentes parámetros involucrados en el problema a resolver. El conjunto de técnicas que conforman los procesos de búsqueda en la cooperación pueden ser aproximados, completos o una combinación de enfoques aproximados y completos [10, 11, 12, 17, 209, 296, 297]. Estos agentes exploran de una forma particular el espacio de búsqueda por medio de un mecanismo de intensificación/diversificación propio de las metaheurísticas empleadas para tal fin, logrando de esta forma escapar de los óptimos locales debido al constante intercambio de información entre los diferentes actores que conforman dicho proceso. Bajo este esquema existen propuestas basadas en dos tipos básicos de agentes que pueden realizar procesos de intensificación o diversificación de acuerdo a la información que pueda suministrar el agente de control [300], o arquitecturas basadas en un esquema de múltiples niveles donde la comunicación entre las diferentes etapas de la arquitectura está permitida y por tanto, un algoritmo puede ser descrito como el resultado de la interacción de algunos agentes, cada uno especializado en una tarea específica [210].

Las técnicas metaheurísticas y sus posibles combinaciones nos permiten obtener algoritmos híbridos que contribuyen a enfrentar con éxito muchos problemas combinatorios, pero se hace necesario la esquematización de algún mecanismo que nos sirva de guía, de tal forma que podemos tener una visión precisa de los pasos a seguir y que nos conduzcan por la línea correcta para obtener éxito al enfrentar este tipo de problemas. Estos mecanismos poseen la bondad de minimizar esfuerzos y ahorrar tiempos de diseño de las acciones a seguir, además de que son viables debido a que suelen ser procesos altamente probados y que demuestran éxito en su aplicación. Es por ello que basados en esta idea, en el presente trabajo de investigación se realizará un estudio de diferentes propuestas para la combinación/hibridación de algunas técnicas; para ello podemos utilizar algoritmos de búsqueda poblacionales y trayectoriales que permitan abordar problemas que posean un alto grado de complejidad con la finalidad de lograr encontrar soluciones óptimas y de alta calidad que las que

podemos encontrar al aplicar los mismos esquemas trabajando en forma aislada (*stand-alone*). También, consideraremos la idea de incorporar diferentes representaciones alternativas para el modelado del problema que será objeto de estudio de tal forma que podamos analizar la viabilidad de cada esquema al enfrentar este tipo de problemas cuando operan con las técnicas seleccionadas. Asimismo, se revisarán los mecanismos existentes para la reducción de las simetrías presentes en las soluciones candidatas del problema; tomando algunas de estas ideas se desea introducir pequeñas variantes que permitan la eliminación progresiva de los espacios que muestren alguna similitud de las soluciones candidatas evaluadas; el objetivo de esta estrategia es poder reducir los tiempos de búsqueda y por consiguiente tratar de optimizar los recursos computacionales disponibles. Otro aspecto que consideramos relevante es la revisión del proceso de comunicación, es decir, como se realizá el intercambio de datos/información entre los diferentes componentes que intervienen en el proceso (políticas para el intercambio de soluciones), de tal forma que podamos incorporar nuevas ideas para guiar, de forma efectiva, hacia las posibles zonas del paisaje de búsqueda donde se encuentran los candidatos con mayor potencial para ser óptimos globales. Se debe estudiar la integración de diferentes arquitecturas (i.e. topologías) de cooperación que emplean diversos mecanismos de comunicación entre los componentes de los modelos, así como el comportamiento de las diferentes configuraciones al realizar variaciones progresivas en los parámetros de relevancia que se emplean en los distintos esquemas utilizados en esta investigación. La idea fundamental es proponer un esquema metodológico viable y robusto para la resolución de problemas de optimización combinatoria en general.

Hemos pensado en aplicar este estudio sobre dos problemas académicos que encontramos en la literatura científica, que corresponden a problemas de optimización/satisfacción combinatoria, cuya formulación ha sido utilizada especialmente para ser abordados por medio de técnicas de ILP o CP; que además se emplean en la soluciones de problemas reales, y de ser problemas con una gran riqueza en lo que a simetrías se refiere. El primero de ellos corresponde al *diseño de bloques equilibrados* en la vertiente de bloques *incompletos*, que es una variante ampliamente utilizado en el diseño de experimentos [178, 289]. Este problema es utilizado en el campo de la agricultura y biología, aunque hoy día su uso abarca muchas otras áreas. Los Diseños de Bloques Incompletos nos presentan una solución la cual nos permite disminuir la varianza del error y nos ofrece comparaciones mucho más precisas en los tratamientos de lo que puede darnos un diseño de bloques completos. El segundo problema considerado es el *diseño de plantillas* altamente empleado en la industria con el objetivo de disminuir la pérdida de materia prima y optimizar los procesos de fabricación del producto [250]. Nuestro objetivo es realizar una extensa experimentación involucrando diferentes actores y afinado parámetros que nos permitan realizar una validación de nuestra propuesta metodológica.

1.2 Objetivos

El diseño de técnicas y algoritmos eficientes que resuelvan adecuadamente problemas complejos de optimización es uno de los campos dentro de la investigación en *informática* con mayor repercusión en la actualidad. De hecho, debido a que la sociedad actual demanda continuamente mejor calidad en el diseño y elaboración de servicios, dar una solución adecuada y eficiente a este tipo de problemas es un objetivo que tiene gran importancia tanto para el mundo de la industria como para la comunidad científica. De acuerdo a estas premisas, el objetivo principal de esta tesis doctoral es el modelado y ajuste de diversas técnicas metaheurísticas con el fin de conseguir resultados de alta calidad en la resolución de problemas de optimización, principalmente problemas combinatorios que han sido formulados especialmente para ser resueltos mediante técnicas de programación con restricciones y programación entera y que tienen una naturaleza simétrica en cuanto existen gran cantidad de solu-

ciones distintas que pueden ser catalogadas como equivalentes. Para lograr este objetivo es necesario abordar ciertos tópicos que básicamente corresponden a los siguientes pasos: en primer lugar se debe realizar una revisión del estado del arte de las diferentes propuestas relacionadas con la resolución de problemas de optimización combinatoria que han mostrado avances significativos. Principalmente nos concentraremos en las tendencias de la computación emergente que involucra la rama de los algoritmos genéticos, las búsquedas locales y los procedimientos híbridos que han sido utilizados en años recientes. En segundo lugar, será necesario la construcción o aplicación de modelos adecuados para la representación de los problemas objeto de estudio, considerando diferentes formas de representación enmarcados en la teoría de la dualidad [162, 236, 332], e intentando emplear algún mecanismo que permita reducir el paisaje de búsqueda por medio de la supresión de estos estados simétricos del problema [266]. En tercer lugar se debe diseñar algún esquema de colaboración, utilizando diferentes modelos de arquitectura, así como algoritmos híbridos evolutivos con diferentes métodos de búsqueda local. En particular nos centraremos en la hibridación desde el punto de vista de la "simbiosis" de algoritmos basados en población y en técnicas de búsqueda trayectoriales, es decir un enfoque memético [270]. Además, consideraremos la utilización de un enfoque de cooperación entre las metaheurísticas propuestas a través de la definición de topologías de comunicación entre los diferentes componentes que participan en el esquema colaborativo. Cada uno de estos enfoques propuestos es validado empíricamente por medio del abordaje de dos problemas de optimización combinatoria como lo son: el diseño de bloques incompletos balanceados y el diseño de plantillas.

Por último, se deben realizar pruebas con diferentes arquitecturas cooperativas de diferentes algoritmos metaheurísticos e híbridos con el fin de determinar las bondades y debilidades de las propuestas estudiadas, para ello se realizará un amplio análisis estadístico de los resultados obtenidos por cada enfoque abordado apoyándonos en los métodos estadíiscos propuestos para la evaluación de este tipo de algoritmos [106].

1.3 Metodología

Hemos considerado tres tópicos que nos han parecido de gran relevancia y basándonos en ellos nuestra metodología incluye: primero, una revisión de la literatura actual sobre las diversas técnicas utilizadas para la resolución de problemas de optimización, enmarcados en los métodos aproximados en general, haciendo especial énfasis en los algoritmos metaheurísticos, la hibridación, así como los esquemas colaborativos. Esta revisión consistirá en la búsqueda del estado del arte de las diferentes técnicas metaheurísticas, la determinación de sus características, las ventajas y desventajas, así como sus potencialidades para la hibridación. El segundo aspecto se refiere a la forma en la cual se abordan los problemas que se utilizarán para las pruebas, considerando los modelos a emplear, la codificación respectiva relacionada con la generación del vecindario (para los métodos trayectoriales), la función de evaluación (función objetivo), la exploración del vecindario respectivo y la codificación de las técnicas seleccionadas. Para esto utilizaremos la metodología incremental propuesta por Mills en 1980 [212], este esquema combina elementos del Modelo Lineal Secuencial con la filosofía interactiva de Construcción de Prototipos. El Modelo Incremental se caracteriza porque es de naturaleza interactiva, lo que nos brinda al concluir cada incremento la obtención de un producto totalmente operacional. En este punto de la investigación consideraremos los diferentes esquemas de técnicas a emplear, iniciando con las técnicas metaheurísticas trabajando en forma individual para cada uno de los modelos considerados (representaciones, primales/duales y los esquemas sin-ruptura/con-ruptura de simetrías), pasando por la hibridación de los diferentes métodos y finalizando con la puesta en funcionamiento de los esquemas cooperativos, donde se consideren todos los actores que se involucran en el proceso de

resolución de los problemas objeto de estudio. Llegados a este punto, se abordarán los procesos de diseño y definición del mecanismo de hibridación y cooperación entre los algoritmos utilizados. Como tercer y último aspecto, se considerará incluir análisis estadísticos sobre los resultados obtenidos en las diferentes pruebas a realizar que nos permitan divisar las ventajas y desventajas de los esquemas empleados.

1.4 Contribuciones

En este apartado se listan las contribuciones que abordaremos en la investigación que vamos a realizar. De forma esquemática, estas contribuciones se pueden resumir como sigue:

- Definición de dos modelos de representación del problema del diseño de bloques incompletos y el diseño de plantillas, basándonos en la teoría de la dualidad y representaciones alternativas.
- Definición de estrategias para la supresión de simetrías en los problemas objeto de estudio de este trabajo de investigación.
- Aplicación de metaheurísticas básicas a la resolución de los problemas objeto de estudio.
- Desarrollo de un modelo de hibridación basándonos en los algoritmos metaheurísticos abordados.
- Definición de un modelo colaborativo entre las distintas metaheurísticas, basados en el esquema de topologías de intercambio de información y extensión de este modelo para lograr escalar hacia modelos meta-cooperativos de heurísticas.
- Aplicación y validación empírica del rendimiento de los diferentes escenarios atacados en los problemas considerados, como el diseño de plantillas y el diseño de bloques incompletos balanceados.
- Estudio del comportamiento de los diferentes parámetros considerados en los modelos meta-cooperativos, por medio de un análisis de sensibilidad de los mismos.
- Demostrar la validez de las técnicas metaheurísticas para abordar problemas combinatorios tradicionalmente abordados por técnicas de satisfacción con restricciones o de programación entera y que presentan un alto grado de simetrías.
- Diseño y desarrollado de técnicas metaheurísticas que pueden considerarse el estado-del-arte de las metaheurísticas al resolver los problemas objeto de estudio de este trabajo de investigación.

Además, se ha planeado un importante proceso de divulgación científica mediante la publicación en revistas y congresos especializados a nivel internacional.

1.5 Organización de la Tesis

Este trabajo de tesis está conformado por cuatro capítulos adicionales, por medio de los cuales se presenta el marco teórico relacionado con el campo de las metaheurísticas, las técnicas híbridas, los modelos de cooperación, las simetrías y los diferentes conceptos necesarios para introducir al lector en la comprensión del trabajo presentado. A continuación mostramos de forma más detallada el contenido de los capítulos:

- **Cap. 2:** Se presenta algunos conceptos básicos de las diferentes temas abordados en este trabajo, como es el caso de las simetrías, la teoría de la dualidad, las técnicas metaheruísticas, las técnicas híbridas y modelos cooperativos.

- **Cap. 3:** Aquí se presenta la formulación de los problemas que han sido utilizados, prestando especial atención a problemas combinatorios tradicionalmente abordados por técnicas de satisfacción con restricciones o de programación entera y que presentan un alto grado de simetrías. Además se da una visión general del enfoque metaheurístico que se ha propuesto para abordar los diferentes problemas que han sido objeto de estudio en este trabajo, así como los modelos empleados, y se presenta la experimentación realizada, junto el análisis estadístico respectivo.
- **Cap. 4:** Se presenta el enfoque que ha sido empleado al aplicar las diferentes técnicas híbridas sobre los problemas abordados, prestando atención a diferentes modelos colaborativos y utilizando diversas políticas para la migración y recepción en los diferentes proceso de comunicación de las topologías usadas, para el intercambio de soluciones. También, se muestra la experimentación y análisis respectivo.
- **Cap. 5:** Finalmente presentamos las conclusiones sobre todo el trabajo expuesto en el resto del documento, dando especial prioridad a las principales contribuciones que hemos conseguido en esta investigación, además de presentar trabajo futuro a ser desarrollado.

CAPÍTULO 2

ANTECEDENTES

En este capítulo vamos a presentar un conjunto de información que permite describir e identificar la naturaleza de los problemas que deseamos abordar en este proceso de investigación, en lo referente a la literatura científica existente en el área de interés. Para ello introduciremos algunos conceptos, aspectos teóricos y/o prácticos sobre los cuales las investigaciones existentes han enfocado estos problemas.

Este capítulo se estructura de la siguiente forma: En la Sección 2.1 y 2.2, se describen los conceptos básicos de Optimización y Complejidad computacional. Luego en las Secciones 2.3 y 2.4 se da una noción de lo que representan las simetrías y la dualidad en el ámbito de los aspectos relevantes de este trabajo. Luego en las secciones 2.5 y 2.6 se presenta una visión general de las técnicas metaheurísticas y las técnicas híbridas que son relevantes para nuestro estudio. Finalmente en la Sección 2.7, presentamos las contribuciones de esta tesis.

2.1 Optimización

El término optimización puede significar muchas cosas, pero podemos decir, según [260], que puede ser considerado como el proceso de determinar las condiciones que permiten encontrar el *mínimo* o *máximo* valor de una función dada, por lo tanto, sin perder generalidad, la optimización puede tomarse como minimización ya que el máximo de una función $(f(x))$ se puede encontrar buscando el mínimo del negativo de la misma función $(-f(x))$. La optimización combinatoria es uno de los campos de estudio más activos en el ámbito de la investigación de operaciones, la computación y matemática aplicada. Un gran número de este tipo de problemas combinatorios los podemos encontrar en el diseño de redes de comunicaciones (las telecomunicaciones), diseño de circuitos integrados (electrónica), visión artificial (computación), control aéreo (aeronáutica), manufactura (industria) y un vigoroso número de otras ciencias [74]. La mayoría de estos problemas pueden ser formulados matemáticamente como la minimización o maximización de una cierta función definida en un determinado dominio, comúnmente asociado al dominio discreto. Históricamente la *optimización combinatoria* comienza con la programación lineal entera, que ha dado a conocer un gran número de importantes aplicaciones en el campo de la planificación de la producción, asignación de tareas, economía, simulación de circuitos, entre otros.

Por otra parte, la incorporación de mecanismo de relajación de las restricciones en la programación lineal definió las bases para la posterior aparición de muchos de los algoritmos aproximados que podemos encontrar en la actualidad, para resolver los llamados problemas NP-Duros. El objetivo que se persigue al tratar de resolver un problema de optimización es lograr encontrar una solución que resulte óptima con un costo computacional y de uso de los recursos razonable. En este aspecto, la optimización numérica ha adquirido mucha atención entre la comunidad científica durante las últimas décadas, y quizás lo más confuso para el diseñador consiste en decidir qué técnicas de optimización son las que más se ajustan a las características que posee el problema bajo análisis.

Un problema de optimización (digamos *discreto*), según [180], se concentra en la minimización (o maximización) de una *función objetivo* f con un conjunto de soluciones factibles $\mathcal{S}$. Normalmente del conjunto $\mathcal{S}$ surge un subconjunto de 2^E (el conjunto de todos los subconjuntos de E), para algún $E \subset \mathbb{N}$, en cuyo caso se tiene un problema de optimización combinatoria. Por supuesto que, en este caso, no hay mucha dificultad, ya que podemos enumerar todas las soluciones factibles (aunque, obviamente, esto puede requerir mucho tiempo), pero ¿qué ocurriría si lo que queremos es la mejor solución? Usualmente, las soluciones factibles son descritas de alguna forma concisa, en lugar de una lista explícita. Pero, el desafío es poder desarrollar algoritmos que sean probablemente o prácticamente mejor que la simple enumeración de todas las soluciones factibles del problema. En la mayoría de los casos (si no es que en todos), el conjunto de soluciones factibles $\mathcal{S}$ se puede obtener en forma descriptiva en vez de una lista explícita. Por ejemplo, $\mathcal{S}$ puede ser el conjunto de rutas posibles para un "agente de ventas" sobre n puntos en algún espacio geográfico. Existen $(n-1)!/2$ (clases equivalentes) de rutas. Este tipo de problema, donde se requiere encontrar una buena ruta para el agente de ventas, es notablemente complicado (difícil). Así, el objetivo para un determinado algoritmo será proporcionar buenas soluciones sobre instancias del problema significativamente grandes, que las que pudiera arrojar un algoritmo exhaustivo. Podemos decir, que un algoritmo es *teóricamente eficiente* para una determinada clase de problemas si el número de instrucciones (pasos) que se requieren para resolver instancias del problema está delimitado por un polinomio, en lo que respecta al número de bits que se requieren para codificar el problema. Todo esto será posible, siempre que se pueda definir cuidadosamente un modelo computacional factible (i.e. máquina de Turing), el uso de nociones de problemas equivalentes (i.e. tiempo polinómico reducible); para poder determinar la ubicación del algoritmo en alguna de las clases de complejidad existente [180]. Nuestro interés se enfoca en la optimización de problemas que pueden representarse con modelos lineales y que presentan restricciones de igualdad y/o desigualdades, específicamente aquellos donde intervienen más de dos variables.

Un problema de optimización intenta dar respuesta a un tipo general de problemas de la forma:

$$\begin{aligned} &\min f(x) \\ &\text{s.t } x \in X \end{aligned} \tag{2.1}$$

Donde $x \in \mathbb{R}^n$ representa variables de decisión, $f(x)$ se denomina la *función objetivo* y es la encargada de medir la calidad de las decisiones (normalmente números enteros o reales) y $X \subset \mathbb{R}^n$ el conjunto de restricciones de la región factible. En particular, si el subconjunto $X = \mathbb{R}^n$, tendremos un *problema de optimización sin restricciones*:

$$\min_{x \in \mathbb{R}^n} f(x) \tag{2.2}$$

Un *problema de optimización con restricciones* [236], puede escribirse como:

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f(x) \\ &\text{s.t } c_i(x) = 0, i \in E, \\ &\quad c_i(x) \geq 0, i \in I \end{aligned} \tag{2.3}$$

Donde E, I representan, respectivamente, el conjunto de índices de las restricciones de igualdad y desigualdades, $c_i(x)$, $(i = 1, ..., m \in E \cup I)$ son las funciones que describen las restricciones. Si la función objetivo y las funciones de restricciones son lineales, este tipo de problemas es llamado *problema de programación lineal* [292].

Lo descrito anteriormente también aplica para los casos de maximización. Para la determinación de soluciones factibles debe cumplirse:

$$f(\bar{x}) \leq f(x) \; \forall \, x \in \mathbb{R}^n \tag{2.4}$$

2.2 Complejidad Computacional

Existen algunos problemas de *optimización combinatoria* que disponen de algoritmos eficientes que los resuelven. En cambio en otros el abordaje de estos no resulta tan evidente; esto es debido a que son de diferente complejidad. Consideraremos el tema de la complejidad computacional enfocada desde el punto de vista de las definiciones de eficiencia y efectividad de un algoritmo, según la forma como actúa sobre el problema que intenta resolver. La literatura científica nos muestra que uno de los aportes más influyentes, para medir la complejidad, fue la definición de las *Máquinas de Turing* en 1936 [14, 319], que resultaron dar una noción de computadora muy robusta, pero flexible. Con el avance de las computadoras que se desarrollaban en la década de los 40's y los 50's, la Máquina de Turing se mostró como un modelo teórico correcto en el aspecto computacional. En términos de la complejidad computacional, existe la idea de que un problema no está "resuelto correctamente" hasta que se encuentre un algoritmo de tiempo polinómico que pueda resuelverlo. Es por ello que nos estamos refiriendo a un problema como intratable, si es tan complejo que no existe algoritmo de tiempo polinómico capaz de darle solución. Para poder realizar una clasificación sobre un determinado problema o algoritmo, es necesario realizar una serie de análisis que involucran elementos como [9]:

- Un modelo computacional. Generalmente una Máquina de Turing.
- La modalidad de computación. Que puede ser determinista o no determinista.
- Los recursos computacionales. Como el tiempo, la cantidad de memoria, etc.
- Cota impuesta al recurso. Representa una función f asociada a los límites de las complejidad.

Para la determinación del modelo a utilizar para la evaluación de la complejidad se utiliza la definición de *Máquina de Turing*. Una máquina de Turing es una representación (o modelo) computacional que realiza un proceso de lectura/escritura de forma automática sobre una entrada llamada cinta, produciendo una salida en la misma. Su funcionamiento está basado en una función de transición, que toma dos elementos: (un estado inicial, una cadena de caracteres), pertenecientes a un alfabeto de entrada [288]. Los datos de una máquina de Turing están determinados por el estado actual y el símbolo que ha sido leído, un par (estado, símbolo), siendo la modificación del estado, la escritura de un símbolo nuevo y el movimiento del elemento llamado cabezal. Es probable que ocurra que para cada dupla (estado, símbolo) exista a lo sumo una posibilidad de ejecución, por lo tanto se considera como una *máquina de Turing determinista*, pero si existen al menos un par (estado, símbolo) con varias combinaciones de actuaciones se considera que se trata de una *máquina de Turing no determinista* [136, 215]. Además podemos mencionar las *Máquinas de Turing cuánticas* [235]. Podemos definir una máquina de Turing determinista como una 6-tupla [84]:

$$M_D = (\Sigma, \#, Q, q_0, \delta, F) \tag{2.5}$$

Donde:

- Σ es un conjunto finito de símbolos, llamado alfabeto[1].
- # es el símbolo en blanco (# $\in \Sigma$).
- Q es un conjunto finito de estados.
- q_0 es un elemento especial que representa el estado inicial ($q_0 \in Q$).
- δ es una función de transición, tal que: $\delta : Q \times \Sigma \longrightarrow Q \times \Sigma \times \{\rightarrow, \leftarrow, -\}$
- F es el conjunto de estados aceptados ($F \subseteq Q$).

Por otra parte, una máquina de Turing No-determinista se define de manera análoga, pero considerando que la función de transición δ es $\delta : Q \times \Sigma \longrightarrow 2^{Q \times \Sigma \times \{\rightarrow, \leftarrow, -\}}$. Esto es, en una máquina no determinista es posible que existan varias transiciones en un momento concreto, suponiéndose que todas ellas se exploran en paralelo.

2.2.1 Clases de Complejidad

La utilización de las máquinas de Turing para el análisis de la complejidad de algoritmos se remonta a los años 60. Entre los trabajos propuestos inicialmente para modelizar el fenómeno de la complejidad computacional cabe destacar a M. Blum [30]. Hartmanis y Stearns inician la ideología de las funciones de tiempo y espacio como un elemento adecuado para la medición [145] y con la aparición de la Tesis de Cobham Edmonds sobre los *problemas tratables* informáticamente se sientan las bases para la creación de una clasificación para este tipo de algoritmos [292]. Dejando de lado la potencial aparición de la computación cuántica, el modelo de máquina de Turing aún permanece como modelo de complejidad razonable, al momento de escribir este trabajo de investigación. Debido a la vasta gama de posibilidades y de teorías que existen sobre el tema nos concentraremos en algunos conceptos básicos relevantes para luego enfocarnos en dar una descripción de las diferentes clases de complejidad comúnmente utilizadas en la literatura científica, respecto a este tema.

Los problemas a clasificar son esencialmente los Problemas Decisionales. El estudio de los problemas de decisión proporcionan una forma conveniente de analizar los problemas de búsqueda, por ejemplo, el estudio de la complejidad de decidir la satisfacción de las fórmulas empleadas, proporciona una forma conveniente de estudiar la complejidad de encontrar tareas satisfactorias para tales fórmulas [131]. Un Problema decisional está determinado por un lenguaje $L \subset \sum^*$ sobre un alfabeto finito. Se trata de evaluar la función característica definida por L, es decir, la función $\chi L : \sum^* \longrightarrow \{0,1\}$ dada mediante:

$$\chi L(x) := \begin{cases} 1 & \text{si x} \in \text{L} \\ 0 & \text{en caso contrario} \end{cases} \tag{2.6}$$

Podemos expresar las siguientes definiciones con base en el recurso utilizado (Espacio/Tiempo) [9, 13]:

Definición 2.2.1 (Clase Determinista) *Se definen las siguientes clases de complejidad para los lenguajes recursivamente enumerables $L \subseteq \{0,1\}^*$. Sea $f : \mathbb{N} \longrightarrow \mathbb{R}_+$ una función monótona creciente. De lo cual se define:*

- ***L(M)** es el lenguaje aceptado por M.*

[1] También llamado alfabeto de la cinta (*tape alphabet*).

- ***T_M/S_M*** *es el tiempo/espacio de ejecución de M*
- ***DTIME***(f)*:=* $\{L \subseteq \{0,1\}^* : \exists$ *una máquina de Turing determinista M tal que:* $L = L(M)$ *y* $T_M \in O(f)\}$
- ***DSPACE***(f)*:=* $\{L \subseteq \{0,1\}^* : \exists$ *una máquina de Turing determinista M tal que:* $L = L(M)$ *y* $S_M \in O(f)\}$

Definición 2.2.2 (Clase No-Determinista) *Se definen las siguientes clases de complejidad para los lenguajes recursivamente enumerables* $L \subseteq \{0,1\}^*$. *Sea* $f : \mathbb{N} \longrightarrow \mathbb{R}_+$ *una función monótona creciente. De lo cual se define:*

- ***NTIME***(f)*:=* $\{L \subseteq \{0,1\}^* : \exists$ *una máquina de Turing no-determinista M tal que:* $L = L(M)$ *y* $T_M \in O(f)\}$
- ***NSPACE***(f)*:=* $\{L \subseteq \{0,1\}^* : \exists$ *una máquina de Turing, no-determinista M tal que:* $L = L(M)$ *y* $S_M \in O(f)\}$

El uso del "comportamiento asintótico" $O(f)$ fue justificado por los resultados obtenidos en [145], de donde se desprende las siguientes relaciones:

$$\begin{aligned} \textbf{DTIME}(f) &\subseteq \textbf{NTIME}(f), \\ \textbf{DTIME}(f) &\subseteq \textbf{DSPACE}(f) \subseteq \textbf{NSPACE}(f) \end{aligned}$$

Además, si $f \in O(g)$, tendríamos:

$$\begin{aligned} \textbf{DTIME}(f) &\subseteq \textbf{DTIME}(g), \\ \textbf{NTIME}(f) &\subseteq \textbf{NTIME}(g), \\ \textbf{DSPACE}(f) &\subseteq \textbf{DSPACE}(g), \\ \textbf{NSPACE}(f) &\subseteq \textbf{NSPACE}(g) \end{aligned}$$

Los Teoremas de Jerarquía de Tiempo y Espacio, debidos también a Stearns y Hartmanis, justifican el uso del término "clasificación". El crecimiento polinómico del tiempo parece una apuesta razonable[2] como definición. De esta forma podemos definir, en forma general, la clase P de problemas como [9, 330]:

Definición 2.2.3 (La clase *P*) *Se define la clase de los lenguajes tratables como la clase:*

$$P := \bigcup_{k \in \mathbb{N}} \textbf{DTIME}(n^k)$$

Podemos mencionar, además, que existen una serie de consideraciones relevantes, respecto al recurso que se desea evaluar, así, encontramos: *Clases determinadas por el tiempo*, *Clases de Funciones/Correspondencias*, *Clases determinadas por el espacio*, entre otras.

[2] Frente a complejidades de orden exponencial o doblemente exponencial

Corolario 2.2.4 *Sea $L \in \boldsymbol{DTIME}(s)$ un lenguaje aceptable en tiempo No Determinista acotado por una función construible en tiempo s. Entonces, existe una máquina de Turing determinista N tal que:*

$$L := \{x \in \Sigma^* : \exists y \in \Sigma^*, |y| \leq cs(|x|), x \cdot y \in L(N)\}$$

de tal modo que $T_N \in O(s)$.

Los lenguajes en **NTIME**(s) son los prefijos de palabras x de algún lenguaje de **DTIME**(s) de tal modo que hay sufijos que las continúan y que tienen talla acotada por s.

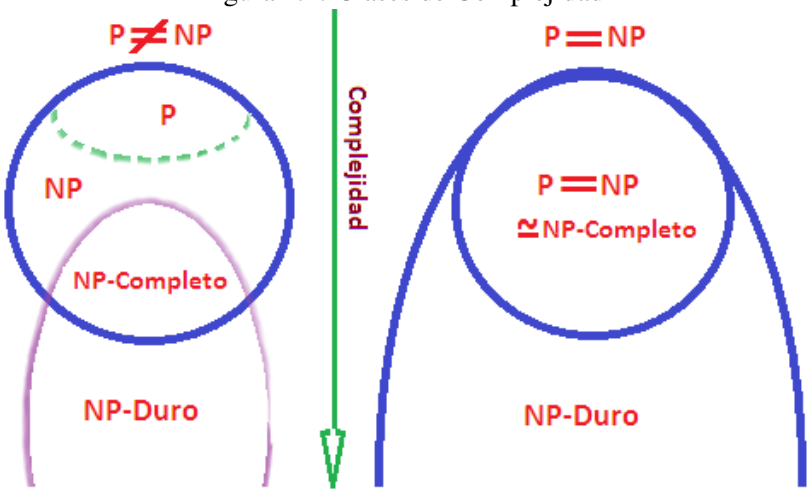
Definición 2.2.5 (La clase *NP*) *Se define la clase NP del modo siguiente:*

$$NP := \bigcup_{k \in \mathbb{N}} \boldsymbol{NTIME}(n^k)$$

Es decir, que se puede verificar en tiempo polinómico [228].

En la literatura actual podemos encontrar un gran número de problemas de optimización combinatoria que han sido analizados y clasificados dentro de las categorías mencionadas anteriormente, sobre todo los de la clases derivadas *NP*-duro y *NP*-completo. Podemos definir un problema como *NP*-duro si cualquier otro problema en *NP* se puede reducir a él. Si el problema en cuestión pertenece además a la *NP*, se denomina *NP*-completo. Si hubiera algún problema *NP*-completo resoluble de manera polinómica, entones $P = NP$. Esta cuestión está abierta, pero sabemos que o bien $P = NP$ o bien hay problemas en *NP* que no son ni polinómicos ni *NP*-completos. En la figura 2.1, podemos revisar en forma gráfica la clasificación de la complejidad descrita anteriormente.

Figura 2.1: Clases de Complejidad



2.2.2 Soluciones Aproximadas

Podemos decir que hasta los momentos no existe una demostración que pruebe que los problemas $P = NP$ o en su defecto $P \neq NP$, sin embargo, se sabe que las soluciones candidatas de los problemas de optimización combinatoria, pueden ser verificados en tiempo polinómico. A lo largo de la historia los investigadores han utilizado una gran variedad de algoritmos aproximados para resolver los tipos

de problemas que se consideran en la clases derivadas de *NP*, basándose en la premisa de que para todo problema (*NP*-duro y *NP*-completo) existe una transformación polinómica que permite llevar instancias de un problema no polinomial a polinomiales. *La aproximación* se puede considerar como un parámetro de precisión, que representa una *cota superior* respecto a la diferencia entre el valor objetivo de una solución óptima, para una determinada instancia del problema, y el valor devuelto por un algoritmo aproximado.

Decimos que un algoritmo mantiene una cierta relación de aproximación si produce soluciones de esta relación de aproximación en todas las instancias del problema que se está abordando. Por ello, podemos decir que una definición de la calidad de las aproximaciones se basa en un algoritmo que logre una cierta relación de aproximación dentro del tiempo polinómico. De está forma podemos definir la proximidad como [228]:

Definición 2.2.6 (La proximidad) *Dado un algoritmo A que permite encontrar una solución a un problema de optimización combinatoria* (S, f, ω)*, en la cual* $s_A \in S$ *denota una solución obtenida por A y* $f_A := f(s_A)$ *su* $f-valor$*. Dado* f_{opt}*, el* $f-valor$ *de una solución óptima. La proporción de proximidad de* s_A *está definida por* f_A/f_{opt}*, en el caso de problemas de minimización, y por* f_{opt}/f_A*, en el caso de problemas de maximización,*

Los métodos aproximados han mostrado un gran éxito en la resolución de problemas de corte académico como para problemas reales. Desde este punto de vista podemos mencionar la técnica de *Simulated annealing* (SA)[3], que fue utilizada por David T. Connolly (1990), para resolver el *quadratic assignment problem*. La propuesta muestra que es un esquema apropiado para resolver este problema, logrando encontrar mejores soluciones con menos cantidad de esfuerzo computacional [55]. Laarhoven et al. (1992) propusieron el *SA* para resolver el problema del *job shop scheduling problem*. El enfoque propuesto aplica la aceptación de las transiciones de costos utilizando una probabilidad diferente de cero para evitar quedar atrapados en los mínimos locales [313]. Hao (2010) propuso un algoritmo heurístico para resolver problema del *traveling salesman problem*. El enfoque aplica un operador de cruce y mutación especializado para lograr un equilibrio entre la velocidad y la precisión [142]. Kirkpatrick et al., 1983, aplicaron con éxito esta técnica sobre un problema real en el campo del diseño de circuitos [169]. Laguna et al. (1991) proponen métodos de búsqueda tabú (*Tabu Search*) para resolver el problema *single machine scheduling problem*. El enfoque consiste en el uso de tres métodos de búsqueda local dentro de una búsqueda tabú [175]. Battiti et al. (1994) introdujeron la "búsqueda reactiva" que utiliza el historial de la búsqueda para ajustar dinámicamente los parámetros relevantes [21]. Scrich et al. (2004) utilizaron esta técnica para resolver el *scheduling job problem* con el objetivo de minimizar la tardanza total [280]. Kinney et al. (2007) propusieron un grupo algoritmos basados en la búsqueda tabú, para resolver el *unicost set covering problem* dividiendo el espacio de soluciones en órbitas, e incorporando un esquema de relajación [167]. Ulrike (2011) apoyándose en las estrategias de intensificación y diversificación, con la ayuda la búsqueda tabú, demuestra que se obtienen resultado de gran calidad para resolver un *real-world scheduling problem* [279]. Atkinson et al. (1998) aplicaron la técnica (*Greedy randomized adaptive search procedure* GRASP) al problema del *time constrained vehicle scheduling*, aquí se utilizaron dos formas de búsqueda adaptativa llamados adaptación local y global, demostrando que el resultado alcanzado era bastante mejor con respecto a las propuestas del momento [15]. Yannis (2012) propuso una versión mejorada del GRASP llamada *Multiple Phase Neighborhood Search GRASP* (MPN-GRASP) para la resolución de problemas de rutas para vehículos. En este método, se aplica un criterio de parada basado en la relajación lagrangiana

[3]Esta y otras técnicas mencionadas en esta subsección serán descritas en la sección 2.5

y optimización del gradiente [192]. Cotta y Fernádez-Leiva integraron GRASP dentro de un algoritmo evolutivo para resolver el problema de las reglas de Golomb [59]. M. Dorigo et al (1994) propusieron una nueva heurística llamada *ant system* para la resolución de problemas de programación de talleres de trabajo. El esquema se basa en el uso de múltiples agentes (hormigas) que interactúan e intercambian información, logrando avanzar hacia zonas prometedoras del espacio de búsqueda [53]. Talbi et al. (2001) propusieron un modelo paralelo ACO (*Ant colony optimization*) para resolver el problema de asignación cuadrática. El algoritmo de optimización basada en colonias de hormigas en paralelo se combina con la búsqueda tabú [301]. Hlaing y Khine, propusieron un ACO para resolver problema del agente viajero en el que, se emplea una estrategia de distribución y la entropía de información. Además, se combina una heurística optimización local junto con el ACO básica [149].

2.3 Simetrías

La *simetría* es una antigua concepción que originalmente establece una relación de conmensurabilidad, y que implica la posibilidad de comparación, así como, la existencia de un factor común fácilmente representable. El término proviene de las traducciones de los **Elementos de Euclides** en las que dos segmentos, **a** y **b**, son llamados medibles (conmensurables) precisamente si hay otro segmento (un tercero) **c**, que puede ser usado una determinada cantidad de veces (en el dominio entero) para generar un segmento congruente a **a**, y otro número de veces también entero para reproducir un segmento congruente a **b** [134] . Así, la *simetría* estaba, al principio, estrechamente relacionada con la armonía, la belleza, y la unidad; aspecto determinante para su papel en las teorías naturales. En el lenguaje de la ciencia moderna, la *simetría* de las figuras geométricas (como los polígonos y poliedros regulares) se define en términos de su invariabilidad, bajo grupos específicos de rotaciones y reflexiones [35]. La naturaleza ofrece un sinnúmero de ejemplos de (aproximadas) formas simétricas: la simetría bilateral de los cuerpos humanos (al igual que los animales), la simetría pentagonal que se encuentra con frecuencia en las flores, la simetría hexagonal de los panales de las abejas, y muchas más. En el campo de la optimización combinatoria las simetrías son un fenómeno que está presente en muchos escenarios, incluyendo los naturales [186] y los artificiales [91], esto hace que los algoritmos de búsqueda empleados para encontrar una solución o todas las soluciones posibles, en problemas de esta naturaleza, son difíciles de abordar, ya sea porque algunos fueron desarrollados para problemas específicos o porque se requiere dedicar mucho tiempo a la transformación del problema original. Según Fahle, Schamberger y Sellman [92] una manera de reducir las simetrías es convertir el problema a resolver (P1) en otro problema (P2) con las mismas características del original pero eliminando todos o la mayoría de los estados simétricos. La principal dificultad que se encuentra está dada en la extensión del espacio de búsqueda por la presencia de muchos estados simétricos. Normalmente los problemas de este tipo requieren de una formulación adecuada para poder aplicar una estrategia de búsqueda que resulte en tiempos de respuesta favorables en la consecución de una posible solución en un espacio de búsqueda de gran tamaño.

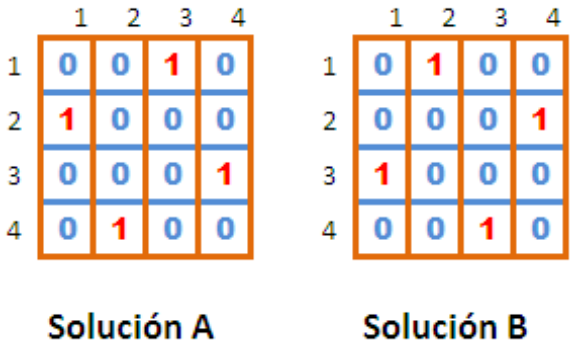
Es muy común que la palabra simetría se utilice en dos sentidos un poco diferentes: para indicar que algo está bien armónico y muy proporcionado, o en un sentido de mayor tecnicismo, para decir que un determinado objeto posee ciertas regularidades geométricas, o tal vez para resaltar la atención de un cierto conjunto de repetición. También es indudable que este tipo de repetición conduce a un importante factor estético, por lo que esta relación directa entre simetría y belleza ha sido un importante principio estético tanto en decoración como en arquitectura [160]. En la actualidad, el tema de las simetrías está siendo revisado ampliamente por la física fundamental moderna, tanto en la teoría cuántica, como la de la relatividad. Todas estas características se relacionan directamente con los pro-

blemas tradicionales de la ciencia filosófica, que incluyen el estado de las leyes de la naturaleza, las relaciones entre la teoría física, las matemáticas, y el mundo en general. Por mencionar un ejemplo de simetría dada por la impresión de repetición regular, es el caso de una mariposa o los rostros humanos, ver figura 2.2. Muchas simetrías geométricas nos la suministra la propia Naturaleza a través de toda su belleza y esplendor de una gran multitud de seres vivos como las flores, los corales, medusas, etc. Nuestro interés, se centra en aquellos tipos de simetrías que podemos encontrar en problemas

Figura 2.2: Simetría Natural

de optimización combinatoria, expresados mediante algún tipo de modelo matemático, que presentan características de similitud entre las distintas soluciones candidatas (o no soluciones) del problema. Como ya se ha indicado antes este tipo de simetrías están enmarcadas más en aquellos aspectos de repitencia o similitud dada, ya sea, por un valor numérico, por una forma geométrica o por algún tipo de permutación. Consideremos un ejemplo ilustrativo, si recurrimos al problema de las *N-Reinas* [71, 189, 287], formulado de la siguiente manera: Una matriz binaria (M) de orden $n \times n$, que representa todo el tablero, de tal forma que M_{ij} es 1 si hay una reina posicionada en la casilla (i, j) y es 0 en caso contrario. Asumiendo que, $n = 4$, una representación, considerada como una posible solución, del problema podría ser igual a la mostrada en la figura 2.3. Podemos encontrar que hay soluciones

Figura 2.3: Solución candidata (tablero 4×4).



(y también no soluciones) que de alguna forma son equivalentes en el espacio de búsqueda, así por ejemplo, para el tablero dado anteriormente podemos decir que la solución mostrada en la figura 2.3 (solución B), es una representación equivalente, simétrica. En este caso lo único que se ha hecho es invertir el tablero como si se hubiese colocado un espejo a la derecha del mismo. Podemos observar que una vez que se encuentra una solución es muy probable que existan otras soluciónes por inver-

siones de filas y columnas (para este problema en particular), claro que no sólo se estudia las simetrías que representan soluciones, sino además aquellas representaciones que no lo son. En los últimos años se ha trabajado en la ruptura de este tipo de simetrías utilizando varias estrategias, como lo es la de añadir nuevas restricciones al modelo de tal forma que se rompan el mayor número de simetrías posibles, reduciendo así el espacio de búsqueda, lo que permitiría un mejor manejo de éste. La literatura científica está llena de artículos enfocados a la ruptura de simetrías en los cuales lo primero que se aborda es la detección de las simetrías del problema, luego hay que imponer las restricciones que las rompan y posteriormente se implementa un algoritmo que sirva para resolverlas (e.g., evolutionary algorithms, constraint programming, local search entre otros). Desde el punto de vista de los problemas de satisfacción de restricciones o en problemas de optimización combinatoria las simetrías pueden causar grandes dificultades en la consecución de soluciones exactas y hay varias formas de tratar estas dificultades, las cuales van de desde la creación de modelos muy sofisticados para la ruptura de las simetrías, hasta la adición de restricciones durante la búsqueda. Tradicionalmente, los método de ruptura de simetrías añadiendo restricciones al problema suele ser aplicados en las técnicas de CP (Constraint Programming), pero en la comunidad de las metaheurísticas suelen ser de bajo perfil. Por otra parte, desde el punto de vista de un algoritmo evolutivo, las simetrías pueden abordarse a través de uso de operadores genéticos como el cruce, mutación y selección de individuos, de tal forma que se eviten las convergencias prematuras hacia una región de búsqueda poco prometedora. El empleo de técnicas híbridas y modelos de colaboración pueden ser alternativas de gran ayuda para enfrentar este tema como ha sido presentado en [11, 72, 195]. Adicionalmente el uso de representaciones alternativas ARFs (Asymmetric Representative Formulations), muestran ser prometedoras para enfrentar este tipo de problemas [163]

Para definir las *simterías*, a la luz de la programación con restricciones y considerando la definición de un problema de satisfacción de restricciones (CSP: Constraint Satisfaction Problem), podemos mencionar que, según Brailsford et al. [37], es una terna (X,D,C) donde:

- **X** es un conjunto de **n** variables $\{x_1, x_2, ..., x_n\}$.
- **D** $=< D_1, D_2, ..., D_n >$ es un vector de dominios. Cada variable x_i toma valores en el dominio D_i.
- **C** es un conjunto finito de restricciones que la solución debe satisfacer. Cada restricción c_i está definida sobre un conjunto de variables $\{x_1, x_2, ..., x_n\}$ que toman valores de sus respectivos dominios $D_1, D_2, ...D_n$.

Además, Meseguer y Torras [206] agregan lo siguiente:

- Una restricción c_i sobre el conjunto de variables ordenadas $var(c_i) = \{x_{i_1}, ..., x_{i_{r(i)}}\}$ especifica la relación $rel(c_i)$ de las combinaciones de valores permitidos por las variables $var(c_i)$.
- Un elemento de $rel(c_i)$ es una tupla $(v_{i_1}, ..., v_{i_{r(i)}}), v_i \in D(x_i)$.

Una solución de un CSP es un conjunto de valores asignados a las variables del conjunto **X**, que satisfacen todas las restricciones establecidas en el conjunto **C**. Típicamente los CSP son resueltos utilizando algoritmos de backtracking basados en la técnica de primero en profundidad.

Si pensamos en el problema de las N-Reinas [71] planteado como un problema de satisfacción de restricciones con un tablero 4x4, los elementos a considerar serían:

- **X**=$\{x_1, x_2, x_3, x_4\}$, las variables que tomarán los valores necesarios del dominio, para obtener una solución.
- El dominio **D** estaría representado por los posibles valores que las variables pueden tomar, en este ejemplo sería: $\{1, 2, 3, 4\}$.

- Las restricciones, e.g., **C**={ $\sum_{i=1,j=1}^{4} (diag_{neg}(i,j)) <= 1$,
  $\sum_{i=1,j=1}^{4} (diag_{pos}(i,j)) <= 1$, $\sum_{i=1,j=1}^{4} (Fila_i(i,j)) = 1\}$.

Según lo expuesto, las *simetrías* se pueden definir como una colección de $n+1$ mapeos $\theta, \theta_1, ..., \theta_n$ definidos como sigue:

- θ es un mapeo de variables, $\theta : \chi \to \chi$
- $\{\theta_1, \theta_2, ..., \theta_n\}$ es el dominio de mapeo, $\theta_i : D(x_i) \to D(\theta(x_i))$
- Las restricciones son transformadas por la adecuada combinación de variables del dominio de mapeo. Una restricción c_i es transformada en c_i^θ, con:
  $$var(c_i^\theta) = (\theta(x_{i_1}), ..., \theta(x_{i_{r(i)}})) \text{ y}$$
  $$rel(c_i^\theta) = \{\theta_{i_1}(v_{i_1}), ..., \theta_{i_{r(i)}}(v_{i_{r(i)}})\}.$$

2.3.1 Algunos tipos de Simetrías

Realizar una extensa clasificación de las simetrías sería un trabajo bastante arduo debido a la interpretación que cada una de las ramas de la ciencia le puede conferir a este tema, así por ejemplo podemos mencionar que existen muchas áreas que han dado, y siguen dando, importancia al tema de las simetrías, por mencionar algunas como:

- El dibujo (invarianza traslacional, de rotación, etc).
- La física (discretas, continuas).
- Las matemáticas (de funciones, reflexión)
- La química (molecular)
- La biología (radial, bilateral)
- La música (traslaciones, giro media vuelta)
- La computación (permutaciones, bilaterales)

En este ámbito, algunos autores han expresado su punto de vista respecto al tema, dando algún tipo de luz respecto a una posible forma de clasificación, así, según Backofen y Will [19], se consideran dos tipos básicos de simetrías como lo son:

- Simetrías Geométricas
- Simetrías por permutaciones.

Por otra parte, Benhamou [23] sugiere dos tipos de simetrías, las cuales denomina:

- Simetrías Semánticas.
- Simetrías Sintácticas.

McDonald y Smith [199] consideran dos clases de simetrías:

- Simetrías polinomiales.
- Simetrías exponenciales.

Gent, Kelsey, Linton, McDonald, Miguel y Smith [114], por su parte agregan las simetrías condicionales.

A continuación describiremos cada uno de los tipos de simetrías mencionados anteriormente.

2.3.1.1 Simetrías Geométricas

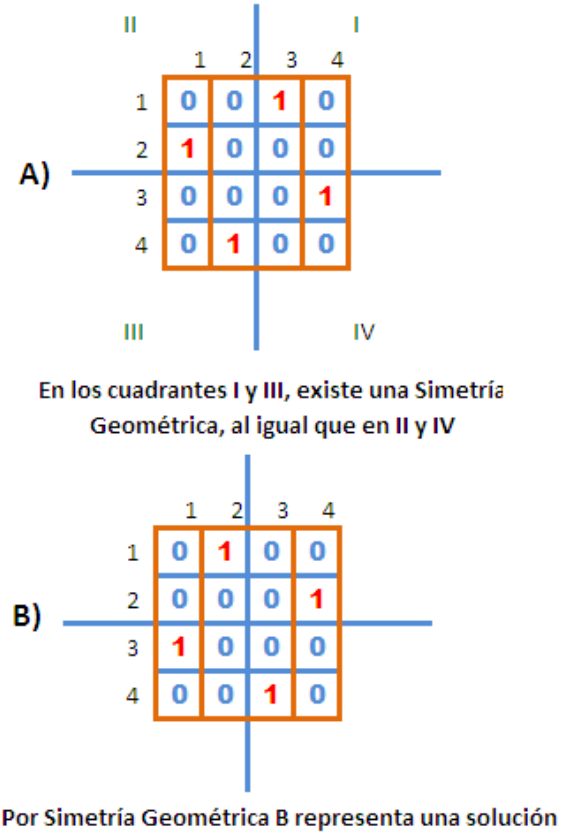
Considerando el caso de las simetrías en un espacio vectorial de **2** dimensiones ($\mathbb{Z}^2$) y su equivalencia en el espacio vectorial de **d** dimensiones ($\mathbb{Z}^d$). Se denotará como [19]:

- **S** conjunto de simetrías aplicables al problema.
- $\vec{\mathbf{x}}$, conjunto de vectores a transformar (generar su correspondencia simétrica).
- A_S, matrices de transformación, esto según la dimensión del espacio vectorial sobre el cual se esté trabajando.
- $\vec{\mathbf{v}}_S$, es el vector a obtener con los puntos de correspondencia simétrica.

Las simetrías para $\mathbb{Z}^2$ tienen exactamente la misma estructura que una representación en $\mathbb{Z}^d$. Esto se define por una transformación $S : \mathbb{Z}^d \to \mathbb{Z}^d$ con $S(\vec{x}) = A_S\vec{x} + \vec{v}_S$ que transforma $\mathbb{Z}^d$ en $\mathbb{Z}^d$. Esto es, la matriz A_S es una matriz ortogonal con la propiedad que el conjunto de columnas $\{\vec{v_1}, ..., \vec{v_d}\}$ de A_S es igual a $\{\pm\vec{e}\,|\,\vec{e}$ es el vector unidad de $\mathbb{Z}^d\}$.

Como ejemplo, consideremos el problema de las N-Reinas mencionado en la sección anterior; un tablero del problema es geométricamente simétrico respecto a los cuadrantes del mismo, esto lo podemos observar en la figura 2.4. Allí se puede apreciar que en la parte **A** los cuadrantes I y III son simétricos, así como los cuadrantes II y IV, de allí se puede encontrar una nueva solución aprovechando estas simetrías, como se indica en la parte **B**.

Figura 2.4: Ejemplo de simetría geométrica.



2.3.1.2 Simetrías por Permutaciones

Ahora consideremos problemas en el dominio entero finito, donde las variables tienen un dominio $D \subseteq \mathbb{N}$ asociado. Se denota con $Perm(D)$ el conjunto de todas las permutaciones de D, y con $Trans(D)$ el conjunto de todas las transposiciones de D (el intercambio de dos elementos de D). Con $S^{\chi}_{Perm(D)}$, se denota el conjunto de todas las simetrías por permutaciones de valores de las variables en χ.

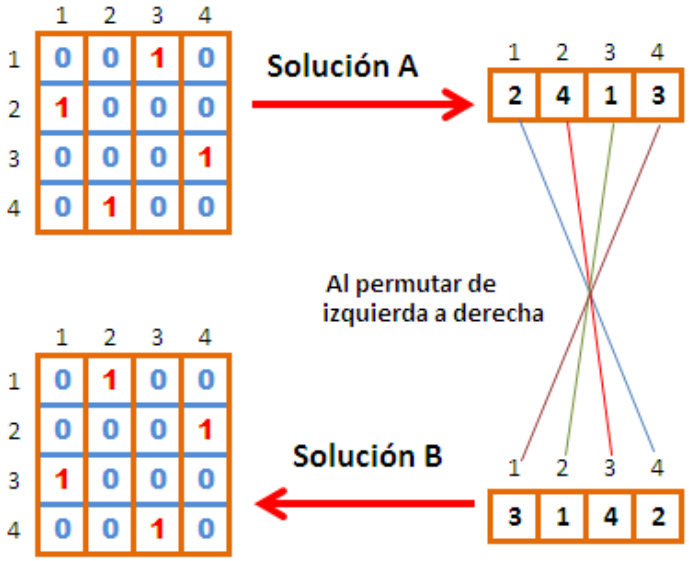
Así, dado χ como el conjunto de las soluciones variables y S al conjunto de las posibles simetrías, el conjunto de todas los valores permutables de χ se define como el conjunto de simetrías, en general se puede denotar como [19]:

$$S^{\chi}_{Perm(D)} = \{S | \exists \pi \in Perm(D) \forall \alpha : S(\alpha)(X) = \pi(\alpha(X))\}$$

El subconjunto de todas las transposiciones simétricas $S \in S^{\chi}_{Trans(D)}$ se define análogamente. Además, para todas las simetrías $S \in S^{\chi}_{Perm(D)}$ se dice que π son las permutaciones subyacentes de S si $S(\alpha)(X) = \pi(\alpha(X))$, con α representando el ángulo de rotación.

Continuando con el ejemplo de las N-Reinas, planteado como un vector [71] (no como tablero), una representación de simetría por permutaciones la podemos apreciar en la figura 2.5. Allí se permutan los valores del vector de izquierda a derecha, generando un nuevo vector, que contiene una nueva solución.

Figura 2.5: Ejemplo de simetría por permutaciones.

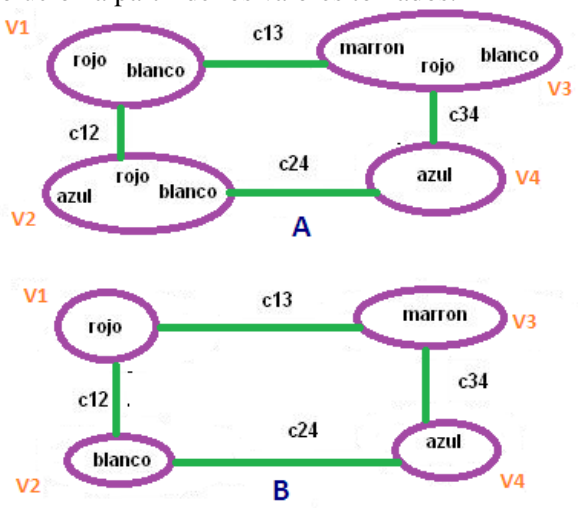


2.3.1.3 Simetrías Semánticas

Las simetrías semánticas tienen relación con la forma en que se colocan los elementos al momento de realizar las transformaciones correspondientes que conllevan a una representación semejante [23]. Se considera el abordaje de dos niveles de simetrías semánticas, las cuales son:

- Simetrías por satisfactibilidad, encontrar una solución.
- Simetrías para todas las soluciones, encontrar todas las soluciones.

Figura 2.6: Ejemplo de simetría semántica. En el aparte **A** se aprecian los dominios para cada una de las variables y en **B** una solución a partir de los valores tomados.



Simetrías por satisfactibilidad: Dos valores del dominio, b_i y c_i, para una variable $v_i \in V$ en un problema de satisfacción de restricciones (CSP: Constraint Satisfaction Problem) son simétricos, si las siguientes afirmaciones son equivalentes:

- Hay una solución para el CSP la cual contiene el valor de b_i.
- Hay una solución para el CSP la cual contiene el valor de c_i.

Simetrías para todas las soluciones: Dos valores del dominio, b_i y c_i, para una variable $v_i \in V$ en un CSP, son simétricos para una solución $sol(P)$, si y sólo si, cada solución del CSP que contiene el valor b_i puede ser mapeado en una solución que contenga el valor c_i y viceversa.

Consideremos ahora el problema de colorear un grafo (sólo 4 vértices) tal como se representa en la figura 2.6, [38]. En cada vértice asumiremos un dominio como el que se indica a continuación:

- V1=[rojo,blanco].
- V2=[azul,rojo,blanco].
- V3=[marron,rojo,blanco].
- V4=[azul].

Los colores rojo y blanco, representan valores simétricos del dominio; esto se puede ver en los vértices V1, V2 y V3. Así, una solución en donde uno de ellos aparezca puede ser intercambiada para generar otra solución. En la figura 2.6 se puede apreciar una solución (V1(rojo), V2(blanco), V3(marrón), V4(azul)), a través de una simetría semántica se lograría una solución intercambiando los valores de los vértices V1 y V2 (que serían los valores representativos de la simetría).

2.3.1.4 Simetrías Sintácticas

Se define una simetría sintáctica como una permutación σ de CSP $P = (V,D,C,R)$, si $[\forall R_{ij} \in R, \ll d_i, d_j \gg \in tuples(R_{ij}) \Rightarrow \ll \sigma(d_i), \sigma(d_j) \gg \in tuples(R_{ij})]$. Así, una simetría sintáctica en un CSP, es un dominio de variables intercambiables σ tal que $\sigma_R(R_i) = R_i, \forall R_i \in R$ [23].

Las simetrías sintácticas vienen definidas por una permutación de valores que hace que el pro-

blema de satisfacción con restricciones permanezca invariable. Por ejemplo, pensando en el problema de las N-Reinas (tablero 4x4) donde tenemos cuatro variables (representando las columnas del tablero) y cada una de ella puede tomar cuatro valores diferentes (por ejemplo, si X_1 vale 2 quiere decir que en la primera columna la reina se posiciona en la fila 2, si X_1 vale tres pues quiere decir que en la primera columna la reina se posiciona en la fila 3, etc.), entonces una simetría sintáctica viene determinada por la siguiente permutación **d**:

$$\begin{aligned} &\text{d(2)}=x_2 \\ &\text{d(3)}=x_3 \\ &\text{d(1)}=x_1 \\ &\text{d(4)}=x_4 \end{aligned}$$

Esto nos indica que los valores de 2 y 3 son intercambiables al igual que los valores de 1 y 4. Esto quiere decir que si aplicamos una simetría sintáctica a cualquier configuración del tablero la configuración resultante guarda las mismas propiedades que la original, es decir, si era una solución continuará siéndolo, y si no lo era no representará solución alguna. Así, para un caso particular donde una solución sea $s = \{3,1,4,2\}$, tendría su correspondencia simétrica si aplicamos el concepto de "simetría sintáctica", obteniendo lo siguiente:

$$\begin{aligned} &\text{d(2)}=x_2 \leftrightarrow \text{d(2)=1} \leftrightarrow \text{d(2)=4} \\ &\text{d(3)}=x_3 \leftrightarrow \text{d(3)=4} \leftrightarrow \text{d(3)=1} \\ &\text{d(1)}=x_1 \leftrightarrow \text{d(1)=3} \leftrightarrow \text{d(1)=2} \\ &\text{d(4)}=x_4 \leftrightarrow \text{d(4)=2} \leftrightarrow \text{d(4)=3} \end{aligned}$$

$$\text{Así, } s' = \{2,4,1,3\}$$

2.3.1.5 Simetrías Polinomiales y Exponenciales

Si se define un CSP L, como un conjunto finito de variables X, donde cada variable X_i pertenece al dominio finito $D(X_i)$. Una posible solución para L es una asignación de todas las variables: $\{\forall i \exists j | X_i = D(X_i)_j\}$, tal que, un conjunto de restricciones finito C sean satisfechas. Un CSP es un CPS *simétrico*, si hay simetrías que actúan sobre él. En este caso, se define una simetría como [199]:

Definición 2.3.1 *Dado un CPS L, con un conjunto de restricciones C, una simetría de L es una función biyectiva $f : A \longrightarrow A$, donde A representa algún estado de la búsqueda*[4]*, i.e. una lista de variables asignadas, un conjunto del dominio correspondiente etc., de tal manera que se cumple lo siguiente:*

- *Dado A, una asignación parcial o completa de L, si A satisface las restricciones C, también lo hace $f(A)$.*
- *De forma similar, si A no es una solución factible, tampoco lo será $f(A)$.*

De esta forma, podemos definir las simetrías *polinomiales* y *exponenciales*, como sigue:

Simetrías Polinomiales: Cuando el número de simetrías se incrementa polinomialmente con respecto al tamaño del conjunto de variables X y su dominio $D(X)$, se dice que son simetrías polinomiales. Es decir, que la complejidad de la evaluación de las posibles soluciones y no soluciones tiene un desempeño computacional del orden de $O(n^2)$.

[4]Las simetrías pueden aparecer en los CSP independientemente de la notación utilizada. Así, para generalizar, el método de representación de una asignación y la aplicación de una simetría se deja al lector

Simetrías Exponenciales: Cuando el número de simetrías se incrementa exponencialmente con respecto al tamaño del conjunto de variables X y su dominio $D(X)$, se dice que son simetrías exponenciales y el número de soluciones (y no soluciones) crece de igual forma. En estos casos las técnicas para su resolución suelen tener un desempeño computacional del orden de $O(n^2)$.

2.3.1.6 Simetrías Condicionales

En la mayoría de los casos, cuando se habla de simetrías en un CSP, se piensa en una función biyectiva que se conoce totalmente antes de comenzar la búsqueda, pero esto no siempre es así, algunas simetrías surgen espontáneamente, dependido del problema tratado y de las decisiones que se tomen en el momento de la búsqueda, a este tipo de simetrías se les llama *condicionales*. Ejemplo de este tipo de fenómeno se suele encontrar en los problemas de planificación [103, 113]. Podemos definir, en forma general, una simetría condicional como sigue:

Definición 2.3.2 *Consideramos a* θ *como un conjunto de restricciones, y a g, una función biyectiva, tal que,* $g(\theta) : A \longrightarrow A$*, donde A es una asignación (parcial). Si* $g(\theta)$ *requiere* Θ *para ser verdadera, antes de que se convierta en una simetría, entonces decimos que* $g(\theta)$ *es una simetría condicional.*

Una simetría condicional en un CSP P_c genera sólo un sub-problema P'_c de P_c. Las condiciones de la simetría son las restricciones necesarias para generar P'_c desde P_c. Una simetría condicional es una generalización de simetrías incondicionales. Las simetrías incondicionales pueden ser vistas como una simetría condicional con un conjunto vació de condiciones.

2.3.2 Rompiendo Simetrías

Una simetría, según las ciencias físicas, puede ser *exacta*, *aproximada* ó *rota*. *Exacta* significa que es incondicionalmente válida; *aproximada*, significa válida bajo ciertas condiciones; *rota* puede significar la ausencia de la simetría, dependiendo de los objetos considerados en este contexto [42]. El estudio de la ruptura de simetría también se remonta a Pierre Curie [43]. De acuerdo a Curie, la *Ruptura de Simetrías* cumple el siguiente rol: para la ocurrencia de un determinado fenómeno en algún medio, el grupo original de simetrías debe ser de naturaleza baja (roto, en terminología actual) para el grupo de simetrías del fenómeno (o para un subgrupo de las simetrías del fenómeno) por la acción de alguna causa. En este sentido la *Ruptura de Simetrías* sería lo que *crea el fenómeno*. Generalmente, la ruptura de una cierta simetría, no implica la desaparición total de las simetrías, ya que una situación donde se aplica ruptura de la simetría se caracteriza porque se reduce el nivel de la simetría del fenómeno original. De allí que es posible describir la ruptura de las simetrías, según la física, en términos de la relación existente entre la trasformación de los grupos, en particular entre un grupo sin ruptura y los subgrupos donde se aplica la ruptura. Así, la física identifica dos tipos diferentes de ruptura de la simetría de las leyes físicas : **explícita** y **espontánea** [35].

En los problemas de optimización combinatoria, al abrigo de la computación, la ruptura de las simetrías está estrechamente vinculada con la posibilidad de reducir la exploración del espacio de búsqueda, tratando de evitar aquellas soluciones (y no soluciones) que hacen invariable un determinado estado del paisaje de búsqueda. Muchas aplicaciones, basadas en técnicas de búsqueda en general, exhiben formas naturales de simetrías que pueden aumentar significativamente la dificultad para resolver los problemas de optimización. Esto ha hecho que en las últimas décadas se esté prestando mucha atención al tema de la ruptura de las simetrías. En los últimos años han aparecido un

gran número de propuestas focalizadas en la ruptura de simetrías, incluyendo esquemas generales de ruptura. Por mencionar algunos trabajos como el de Backofen y Sebastian [19], que proponen un método general para la eliminación de simetrías arbitrarias, considerando una extensión de las nociones de simetrías utilizadas en años anteriores (simetrías geométricas, por permutaciones, árboles de búsqueda). Gent y Smith [111], introducen un método para la ruptura de las simetrías durante la búsqueda (*symmetry breaking during search* SBDS), donde resaltan la importancia que su propuesta retorna sólo una posible solución desde cada conjunto simétrico. Fahle et al. [92] proponen un método que permite detectar simetrías basados en *puntos de detección de simetrías* durante el proceso de búsqueda (*Symmetry Breaking via Dominance Detection*, SBDD). El algoritmo genera, en todo momento, nuevos puntos de detección, para suprimir aquellos nodos que resulten ser equivalentes a otros que ya fueron expandidos. Benhamou y Mohamedda [24], definen una estrategia para detectar y eliminar las simetrías locales, mediante el uso de las definiciones de simetrías semánticas y sintácticas previamente abordadas en [23]. En este ámbito, Meseguer y Torras [206], consideran la ruptura de simetrías como la *poda* de soluciones "no buenas" (emplean el término *nogoods*), de esta forma definen las simetrías *nogoods* de una determinada rama como: Dado un estado s definido por la asignación de variables pasadas $\{(x_i, v_i)\} i \in P_a$, donde P_a es el conjunto de asignaciones o variables pasadas, θ es un estado simétrico a s, y x_k la variable correspondiente. Si después de la asignación de x_k se encuentra una solución *nogoods* p, la cual se denota por:

$$p = \bigwedge_{j \in P'_a, P'_a \subseteq P_a} (x_j, v_j) \Rightarrow (x_k \neq v_k)$$

Entonces, es fácil deducir que $\theta(p)$ tampoco lo es. Si p no es una solución, esto indica que viola la restricción c. Por la definición de simetría, $\theta(p)$ viola las restricciones simétricas c^{θ}, con lo cual el estado s' evaluado en $\theta(p)$ será descartado, ocasionando una ruptura de la simetría θ.

2.3.2.1 Técnicas para la Ruptura de Simetrías

Muchos investigadores han tratado el tema de la ruptura de simetrías tanto en problemas de optimización combinatoria como en problemas de satisfacción de restricciones utilizando planteamientos clásicos, a continuación se menciona algunos de estos problemas.

- Golfer problem [310].
- Cutting stock problems [139, 328].
- Ramsey problem [90].
- Balanced incomplete block designs [246, 268].
- n-Queens problem [71, 287].
- Graph coloring problem [22, 46].
- Scheduling problem [75, 133].
- Traveling salesman problem [150, 179].
- Max-cut problem [170, 282].
- Minimun tardy task problem [6, 7].

La importancia económica de los problemas reales que se plantean como problemas de optimización combinatoria es grande y el número de aplicaciones potenciales crece día a día y a través del abordaje de estos problemas se han utilizado los resultados obtenidos para resolver situaciones de la vida cotidiana, como lo son:

- Secuencias de ADN [118].
- Distribución de materiales [340].

- Enrutamiento de vehículos [48].
- Diseño de redes de comunicaciones [278].
- Asignación de frecuencias a teléfonos celulares [190].
- VLSI (Very Large Scale Integration, Integración en Escala muy Grande) [155]
- Problemas financieros [52].
- Planificación de trayectorias de Robots, consultar [305].

Muchas de estas situaciones deben ser planteadas como problemas de satisfacciones de restricciones. Según Barber y Salido [20] la resolución de problemas de satisfacción de restricciones (CSP, Constraint Satisfaction Problem) consta de dos fases diferentes:

- Modelar el problema como un problema de satisfacción de restricciones, es decir, mediante un conjunto de variables, dominios y restricciones.
- Procesar el problema de satisfacción de restricciones resultante. Esto mediante:
  - Técnicas de consistencia: que se basan en la eliminación de valores inconsistentes de los dominios de las variables.
  - Algoritmos de búsqueda: Que se basan en la exploración sistemática del espacio de soluciones hasta encontrar una solución o demostrar que no existe ninguna.

Los problemas de diseño combinatorio se consideran, según Smith [284], que pueden modelarse como un problema de satisfacción de restricciones (CSP), de tal forma que, es posible la reducción de las simetrías considerando tres aspectos fundamentales:

- Remodelar el problema, de tal forma que se descarten las posibles simetrías existentes.
- Agregar restricciones al modelo de tal forma que sólo sean satisfechas por una función en cada clase equivalente.
- Agregar restricciones al modelo, durante la búsqueda.

Por otra parte, Mcdonald y Smith [198, 199], determinaron dos aspectos fundamentales que pueden mejorar los métodos la ruptura de simetrías en CSPs, en términos generales, si se presta atención a:

- La técnica empleada para la ruptura de simetrías
- Los esquemas de representación de las simetrías.

La técnica es el cómo nosotros aplicamos la ruptura de simetrías y en este proceso ellos consideran algunos métodos que es posible utilizar como lo son:

- Introduciendo nuevas restricciones al CSP, que frenan las simetrías.
- Emplear heurísticas para la ruptura de simetrías.
- Suspender la búsqueda en los subárboles que son simétricamente equivalentes a los previamente expandidos.
- Verificando que los nuevos nodos a evaluar no son simétricamente similares a los previamente procesados.

La representación de simetrías se refiere a como se implementan las simetrías descritas en el problema y de qué manera se utiliza dicha implementación para aplicar una técnica que permita la ruptura de las mismas. Los problemas con simetrías, en satisfacción de restricciones o en problemas de optimización combinatoria, presentan grandes dificultades en encontrar soluciones exactas y para ello hay que emplear sofisticados modelos que impongan algún tipo de restricción para tratar de reducir la cantidad de simetrías. No obstante, algunos problemas contienen simetrías inherentes que no pueden ser eliminadas por remodelamiento. Así mismo, es posible utilizar algoritmos genéticos para la ruptura de simetrías y reducir el espacio de búsqueda, esto se logra a través de la incorporación de operadores genéticos adecuados, como por ejemplo el operador *cruce* que puede ser modificado y adaptado a las

necesidades del problema a tratar [239].

2.3.2.2 Propuestas Existentes para Romper las Simetrías en CP y CSP

El uso de algoritmos exactos y heurísticas para resolver problemas de optimización combinatoria es un dominio de trabajo típico en informática. Pero para problemas no triviales estos algoritmos requieren un elevado consumo de memoria o tiempo de ejecución. Esto ha hecho que se esté estudiando y desarrollando diversas técnicas que conlleven a reducir dichos inconvenientes. Los primeros trabajos realizados sobre la ruptura de simetrías se aplicaron a problemas con satisfacción de restricciones [104, 146, 281], desde entonces se ha ampliado el campo de estudio sobre el aprovechamiento de las simetrías en procesos de búsquedas [23, 70, 197, 253]. Sin embargo, en los últimos años ha habido un marcado incremento en el número de propuestas que abordan el estudio de simetrías en CSP [19, 101, 111], todos ellos han introducido nuevas técnicas para tratar simetrías en CSP. A continuación se presentan algunos de los trabajos que se han desarrollado en el ámbito de la ruptura de simetrías, tanto en el área de CSP como en los AG (Algoritmos Genéticos).

Según Walsh [322], podemos encontrar varias formas de romper las simetrías, las cuales pueden ser:

- Ruptura de simetrías de valores.
- Ruptura de simetrías de variables.
- Ruptura de simetrías de valores y variables.
- Ruptura de simetrías por variables/valores.

Para un ejemplo de la ruptura de **simetrías de valores**, podemos continuar con el problema de las N-Reinas, visto como un vector. Si se agrega una restricción al problema como la siguiente: $x_1 \in \{1,2\}$, se está obligando que los valores de la variable x_1 sólo pueden asumir el valor 1 y 2 del dominio, y en este caso la ruptura se logra por la incorporación de restricciones de valores. Si por el contrario, se agrega una restricción al problema como: $x_1 < x_4$, se lograría romper ciertas **simetrías de variables**. En general, al considerar las posibles simetrías que se presentan en el problema a tratar, las soluciones simétricas no son estudiadas; por ejemplo: Sol_2={3,1,4,2} no sería estudiada si previamente se ha analizado la solución Sol_1={2,4,1,3}, pues en el caso de Sol_2, $x_1 = 3 < x_4 = 2$ no se cumple la condición y por lo que se descarta ésta. Los demás tipos de ruptura de simetrías son una combinación del frenado de éstas por variables y valores.

SBDD: Este método ha sido propuesto por Fahle et al. [92] y lo llamaron *Symmetry Breaking via Dominance Detection (SBDD)* debido a que se basa en la detección de las relaciones de dominio entre los subárboles de búsqueda. Consiste en la detección de puntos de selección en el proceso de búsqueda. Basados en la generación de puntos de selección, durante la búsqueda, se chequean dichos puntos para determinar si son equivalentes a nodos que ya han sido expandidos anteriormente. Si es así, el punto de selección correspondiente será podado, de lo contrario, se procesa normalmente. En general, la propuesta se basa en los siguientes principios, para los efectos de reducir las simetrías sobre el espacio de búsqueda:

- Una base de datos τ que almacena la información del espacio de búsqueda ya procesado.
- Una función específica del problema $\Phi : (P^{\triangle}, P^{\square}) \rightarrow \{false, true\}$ que arroja *true* si el patrón $P^{\triangle}$ es dominado por $P^{\square}$ bajo la misma función simétrica φ.
- Si las simetrías son usadas por propagación, es necesario que exista una función que, para todas las variables x, remueva todos los valores de b, desde el dominio de x para los cuales

$$\Phi(P^{\triangle}[x = b], P^{\square}) = true.$$

SBDS: Gent y Smith [111] proponen un método para la ruptura de simetrías durante la búsqueda (Symmetry Breaking During Search), el cual se basa en tres objetivos básicos:

- Garantizar que nunca se realizarán búsquedas para encontrar soluciones simétricas.
- Utilización de estrategias heurísticas como sea posible.
- Detección de cuales simetrías se mantienen invariantes durante la búsqueda antes de pasar a una nueva rama.

Esta técnica implica el mantenimiento de una lista de simetrías de un problema dado y la utilización de ésta para la generación de restricciones que frenen el avance hacia soluciones simétricas.

SES: Backofen y Will [19] introducen un nuevo método para la exclusión de simetrías llamada *symmetry excluding search (SES)*, y está basado en la noción: "restricción de simetrías", la cual se aplica en la modificación de un algoritmo general de búsqueda por restricciones. Este método no influye sobre la estrategia de búsqueda a aplicar. En la propuesta se explica cómo funciona la modificación del algoritmo de búsqueda a través del uso de dos algoritmos llamados *naive algorithm (naive_ses)* e *improved algorithm (ses)*:

procedure naive_ses(C_p, C_{store})	procedure ses($\overrightarrow{Cp}, C_{store}$)
if C_{store} determina una solución α then	if C_{store} determina una solución α then
write α	write α
elsif $\perp \notin C_{store}$ then	elsif $\perp \notin C_{store}$ then
seleccione una restricción c	seleccione una restricción c
$C^l_{store} := C_{store} \bigwedge c$	$C^l_{store} := C_{store} \bigwedge c$
$C^l_p := C_p \bigwedge c$	$\bigwedge \forall s \in S : \overrightarrow{Cp}^l_s \Leftrightarrow \overrightarrow{Cp}_s \bigvee s_{con}(c)$
naive_ses(C^l_p, C^l_{store})	ses($\overrightarrow{Cp}^l, C^l_{store}$)
$C^l_{store} := C_{store} \bigwedge \neg c$	$C^l_{store} := C_{store} \bigwedge \neg c$
$\bigwedge \forall s \in S : s_{con}(C_p) \rightarrow s_{con}(\neg c)$	$\bigwedge \forall s \in S : \overrightarrow{Cp}_s \rightarrow s_{con}(\neg c)$
naive_ses(C_p, C^r_{store})	ses($\overrightarrow{Cp}, C^r_{store}$)
endif	endif

Donde se pueden distinguir las siguientes variables y elementos:

- α corresponde a una solución.
- c corresponde a una restricción en particular.
- C_{store} representa una estructura de almacenamiento donde se registrarán, lo que los autores han denominado, "las pistas" del conjunto de restricciones.
- Cp corresponde al conjunto de restricciones.
- $\overrightarrow{Cp}$ representa un conjunto de variables booleanas.
- s_{con} representa el conjunto de antecedentes.

El algoritmo *naive_ses* funciona en combinación con Cp, agregando las implicaciones sobre la rama derecha en C_{store}. Por otra parte, en *ses*, por la rama derecha sólo se insertan las implicaciones modificadas.

ECL^{i}PS^e: *ECL^{i}PS^e* es una plataforma desarrollada por Wallace et al. [320] para la Programación Lógica con Restricciones (PLC), permitiendo además, técnicas de programación matemática y es-

tocástica. ECL^iPS^e fue diseñado para resolver problemas combinatorios industriales en las áreas de planificación, programación de tareas y asignación de recursos; contiene una librería para la ruptura de simetrías durante la búsqueda (SBDS) y actualmente corresponde con el sistema la ruptura de simetrías distribuida con un solucionador de restricciones. Esta plataforma introduce dos tipos de modelos:

- Un modelo conceptual, que captura las especificaciones del problema.
- Un modelo de diseño, que está bien afinado para ofrecer soluciones a través del uso de un computador.

GAP-ECL^{i}PS^e: Gent et al. [112] presentan una interface entre ECL^iPS^e y GAP (Groups, Algorithms and Programming). Esta interface proporciona un eficiente método para tratar problemas de satisfacción de restricciones con un gran número de simetrías, sin utilizar gran esfuerzo en la programación. La idea principal es utilizar GAP como una caja negra, mientras que la implementación ECL^iPS^e actúa sobre el árbol de búsqueda para resolver las posibles simetrías, así, ECL^iPS^e proporciona un conjunto de resultados teóricos sobre los grupos de simetrías. Esta implementación utiliza esos resultados para frenar alguna simetría que ocurra durante la búsqueda. Está propuesta, en general, utiliza una implementación de *SBDS* para permitir que los equivalentes simétricas de las asignaciones se determinen por cálculo dentro de GAP. Apoyándose en ECL^iPS^e y formulando el problema como un CSP, de la forma:

$$C \wedge x_1 \in D(x_1) \wedge \ldots \wedge x_n \in D(x_n) \tag{2.7}$$

Donde C representa las restricciones del problema y $D(x_i)$ es el dominio de la i-ésima variable. Obteniendo, primero, información sobre algún tipo de simetría global, y activando un procedimiento de búsqueda binaria *backtrack* por medio de SBDS. En el contexto de asignaciones parciales de A, después de que la asignación $Var = Val$ falle, SBDS emite un mensaje del tipo $g(A) \Longrightarrow g(Var \neq Val)$ a cada elemento g en el grupo de simetrías. Esto asegurará que si alguna vez se visita un equivalente simétrico de A nunca se probará el equivalente de $Var = Val$.

NuSBDS: McDonald [198] introduce un sistema para el frenado de las simetrías a través del uso de ILOG, la cual es una herramienta para resolver problemas con restricciones. Lo novedoso de la propuesta NuSBDS, se basa en dos aspectos:

- La teoría de grupos utilizada por NuSBDS es transparente al programador de restricciones, por lo que no es necesaria la aplicación de conocimientos profundos en matemática.
- Un conjunto de *macros* han sido escritas para permitir al programador de restricciones definir sus propios grupos de manera fácil y rápida.

Las *macros* son, en esencia, funciones que representan diversos tipos de simetrías. De esta forma, diferentes *macros* pueden ser llamadas en forma repetida para la combinación de distintos tipos de simetrías y crear así nuevos grupos de ellas. Esto permitirá que el programador de restricciones pueda crear grupos complicados de simetrías utilizando muy pocos comandos de forma sencilla.

ARF: Este tipo de técnica fue introducida por [40], en la cual se emplea una representación alternativa que permite reducir las simetrías. Posteriormente [162, 318] utilizaron esta idea incorporando una mejora a la formulación de las representaciones alternativas. La principal idea es introducir nuevas variables de decisión asimétricas que indican si un objeto pertenece a un clúster específico, pero el

clúster se identifica con el objeto indexado más bajo. La ventaja de esta formulación es que elimina la simetría entre los grupos, que existe en la formulación clásica.

Heurística para la Ruptura de Simetrías: La idea propuesta por Messeguer y Torras [206] es utilizar las simetrías para guiar la búsqueda. A través del uso de las simetrías se dirige la búsqueda hacia subespacios con una alta densidad de estados no simétricos, para frenar tantas simetrías como sea posible. Se emplea una heurística por selección de las variables, la cual puede ser teóricamente combinada con *mínimo-dominio* heurístico. Al asignar una variable se rompen las simetrías. Esta heurística busca a la variable que más simetrías puede frenar al ser asignada, orientando de esta forma a subespacios con menor número de estados simétricos.

Uso de Predicados para la Ruptura de Simetrías: Crawford et al. [70] presentan un esquema general donde las simetrías son aprovechadas a través de la adición de predicados para la ruptura de las mismas. El grupo introduce un orden sobre el conjunto de variables, y utiliza éste para construir un orden lexicográfico para el conjunto de asignaciones. En general la implementación consiste en los siguientes pasos:

- Se toma una cadena como entrada (teoría) y se convierte en un grafo, tal que el automorfismo del grafo representa exactamente las simetrías dadas en la teoría.
- Se utiliza la herramienta desarrollada por McKay [200] (véase [213] como complemento) para la generación del grupo de automorfismos.
- Tras la generación del grupo de automorfismos se construye el árbol de simetrías y se crea el predicado para la ruptura de las mismas.

2.3.2.3 Ruptura de Simetrías más allá de CSP

Como lo indica la literatura, las técnicas metaheurísticas tiene la particularidad de explorar porciones del espacio de búsqueda lo que permite, en cierta forma, el uso eficiente de los recursos (memoria, tiempo, etc.). La presencia de estados simétricos en algunos problemas de optimización hacen que los procesos de búsqueda exhaustiva sean impensables. Tradicionalmente se ha empleado programación con restricciones y con satisfacción de restricciones para abordar este tipo de problemas. La idea fundamental es evitar explorar aquellas zonas del espacio de búsqueda que resultan, en algunos casos, con un alto grado de similitud, es decir que conducen a los algoritmos utilizados a evaluar posibles soluciones candidatas que presentan características similares a las que se han revisado previamente en alguna de las etapas del proceso de búsqueda y que no aportan ninguna clase de beneficio a este proceso. La literatura científica presenta un gran número de trabajos que relacionan las metaheurísticas y la ruptura de simetrías. Podemos mencionar el trabajo realizado por Hoyweghen et al. [152], que utilizan un algoritmo evolutivo para tratar el tema de las simetrías. Básicamente aplican cambios en la función de fitness, adaptación de operadores genéticos y prestan gran atención a la redundancia en el genotipo/fenotipo. Se ha tratado la ruptura de simetrías aplicando algoritmos genéticos por medio del agrupamiento de la población en una determinada región simétrica del espacio de búsqueda, como es el caso de la propuesta presentada por Pelikan y Goldberg [238]. Así mismo, Rogers et al. [265], utiliza un algoritmo genético haciendo énfasis en los operadores de cruce y mutación, los cuales afectan la ruptura de las simetrías del problema. Adak y Demiriz [2], presenta una propuesta para tratar los problemas simétricos a través del empleo de Colonia de Hormigas (Ant Colony Optimization, ACO), aplicados a instancias del problema del viajante (Travelling Salesman Problem, TSP) y de

la asignación cuadrática (Quadratic Assignment Problem, QAP). En la propuesta de Santana et al. [277], se emplean algoritmos evolutivos (EA) y de estimación de distribución (EDA) aplicados al 'power domains'. Finalmente, mencionaremos el trabajo realizado por Prestwich y Roli [248], que combinan un algoritmo genético con la búsqueda tabú para mejorar el SBNO (Symmetry Breaking by Nonstationary Optimisation), aplicado al problema del diseño de bloques incompletos (BIBD).

2.4 Dualidad

El tema de la *Dualidad* ha sido acuñado por la Programación Lineal (*Linear programming*, LP) para indicar que todo problema formulado como un *LP*, tiene asociada una segunda representación del mismo, conocido como su problema equivalente (Dual). Ambos se relacionan muy estrechamente, hasta tal punto que el modelo usado para uno puede ser generado a partir del modelado del otro, por tanto la solución óptima del modelo utilizado en el primero, nos proporciona información muy amplia acerca de la generación de la solución óptima del segundo [137]. La complementariedad y la dualidad son dos conceptos que están estrechamente relacionados, y juegan un papel fundamental en muchos campos de las ciencias matemáticas, la ingeniería y la optimización. Su estudio ha sido abordado a lo largo de la historia de la ciencia, iniciando con la conocida *transformación de Legendre* que fue introducida formalmente en 1787. Los llamados *puntos de silla lagrangianos* juegan un papel importante en la teoría de la dualidad y la optimización con restricciones. En forma similar, se desarrolló la *teoría de la dualidad* en matemáticas basada en los principios variacionales convexos y la optimización con la propuesta de la *Transformada de Fenchel* en 1949. En la rama de las ciencias de la computación los llamados teoremas de *punto interior* se basan, de igual forma, en la teoría de la dualidad *Lagrangiana*, que ha surgido como una técnica revolucionaria durante largos años. La optimización con restricciones, la variabilidad de las inecuaciones y los teoremas de punto fijo están estrechamente ligados a la programación matemática. El esquema primal-dual para la resolución de problemas de optimización ha sido extensamente utilizado en apoyo a la ya conocida teoría de la dualidad Lagrangiana por diversos investigadores como Aubin at el. [16], Thach et al. [304], Mehrotra [202], entre otros.

La teoría de la dualidad se ha empleado, en programación matemática, en una gran variedad de aplicaciones y está estrechamente relacionada con la programación paramétrica y análisis de sensibilidad. Aparte del interés conceptual, que proporciona interpretaciones en el aspecto económico de los problemas a resolver; también podemos aprovechar la información que proporciona la dualidad para mejorar el desempeño de algún tipo de algoritmo. Pero, además, se han mostrado muy buenos resultados sobre la aplicación de está teoría en el ámbito de la *LP* [108], la Programación Entera (*Integer Programming*, IP) [332]. Mientras que los algoritmos para LP producen programas duales únicos (que son relativamente fáciles de obtener), los algoritmos para *IP* generan una función de dualidad, cuya caracterización depende del método usado para resolver el problema IP primal.

2.4.1 Conceptos Básicos

Para dar una visión generalizada de la teoría de la dualidad, debemos introducir un par de conceptos básicos. Consideremos un problema de optimización expresándolo como programación lineal entera [154].

Definición 2.4.1 (Primal y Dual en LP) *Dados* $f(x) = cx$ *y* $g(x) = Ax$, *con* $x \in \mathbb{R}^n$, *y* $A \in \mathbb{R}^{m \times n}$, *la*

representación primal viene dada por:

$$\begin{aligned} \max \quad & cx \\ \textit{Sujeto a:} \quad & Ax \leq b \\ & x \geq 0 \end{aligned} \tag{2.8}$$

Y su correspondiente representación Dual, estaría dada por:

$$\begin{aligned} \min \quad & yb \\ \textit{Sujeto a:} \quad & yA \geq c \\ & y \geq 0 \end{aligned} \tag{2.9}$$

Las restricciones mostradas en 2.8 son todas las posibles combinaciones no-negativas de las restricciones de 2.9. De allí que cualquier solución factible en 2.8, también lo será en 2.9. Por otra parte, al poner los pesos de x igual a los vectores unitarios, las restricciones de 2.8 están incluidos en 2.9. Por lo tanto los conjuntos de soluciones factibles son los mismos para ambos problemas; mostrando de esta forma su equivalencia.

Definición 2.4.2 (Primal y Dual en IP) *Consideremos, $c = (c_1, ..., c_n)$ y $A = (a_1, ..., a_n)$, de tal manera que a_j denota la j-ésima columna de la matriz A de dimensiones $n \times m$. Asumiendo además, que todos los valores son enteros. Dado también, $x = (x_1, ..., x_n)$. Podemos formular la representación primal como:*

$$\begin{aligned} \max \quad & \textstyle\sum_{j=1}^{n} c_j x_j \\ \textit{Sujeto a:} \quad & \textstyle\sum_{j=1}^{n} a_j x_j \leq b \\ & x_j \geq 0, \forall j \in \mathbb{Z}^n, \textit{ con } j = 1,...,n \end{aligned} \tag{2.10}$$

Igualmente, su correspondiente representación Dual, sería:

$$\begin{aligned} \min \quad & \textstyle\sum_{j=1}^{n} b_j y_j \\ \textit{Sujeto a:} \quad & \textstyle\sum_{j=1}^{n} a_j y_j \geq c \\ & y_j \geq 0, \forall j \in \mathbb{Z}^n, \textit{ con } j = 1,...,n \end{aligned} \tag{2.11}$$

2.5 Heurísticas y Metaheurísticas

Muchos de los problemas de optimización resultan tener un elevado grado de dificultad en el momento de abordar su solución, en este ámbito un gran número de algoritmos han sido propuestos para atacar este tipo de problemas, en forma general. En los últimos años los investigadores han podido darse cuenta que los algoritmos exactos han resultado poco factibles para ser empleados en la solución de estos problemas. Como se sabe, el esfuerzo computacional y el uso de recursos es insostenible para la aplicación de estos, lo que ha permitido el surgimiento de nuevas técnicas para enfrentar los nuevos retos, dando lugar a los llamados *métodos aproximados*, que intentan obtener soluciones factibles de alta calidad. Así, podemos mencionar los métodos *Heurísticos*. Se usa el término heurística para hacer referencia a un procedimiento que permite aportar soluciones sobre un problema, ofreciendo un excelente rendimiento, en lo que a la calidad de las soluciones se refiere, y a los recursos computacionales empleados. No es necesario realizar un riguroso análisis formal sobre el problema, sino más bien, el

conocimiento experto sobre la tarea a realizar. Para hacer frente a la resolución de problemas específicos, se han empleado técnicas heurísticas exitosos, de las cuales, se ha tratado de tomar lo que ha sido crucial en su éxito para poder utilizarlos a otros problemas o llevarlos a contextos mucho más extensos. Sin embargo, los heurísticos se han quedado cortos en un gran número de problemas y han dado paso a otras técnicas aproximadas como lo son las *metaheurísticas*, que son técnicas inteligentes que nos permiten diseñar o tratar de mejorar procedimientos heurísticos de talla muy general con un grado de rendimiento muy elevado. La relevancia de las metaheurísticas se refleja en la publicación de libros sobre este campo en los últimos años – véanse [83, 109, 135, 263] entre otros. Las metaheurísticas, enmarcadas en una visión global, están estrechamente relacionadas con técnicas aplicadas a procesos de búsqueda, donde todas aquellas posibles soluciones que resultan factibles en el proceso de dar solución al problema se consideran como elementos del espacio de búsqueda, que se van variando a medida que se emplean las distintas operaciones que han sido diseñadas para llegar a encontrar la solución definitiva (normalmente basadas en vecindarios). Por ello, y además, porque los procedimientos de búsqueda heurística constituyen el eje central de las metaheurísticas, con frecuencia se suele interpretar el término metaheurísticas como una técnica aplicable, en esencia, a los procedimientos de búsqueda sobre aquellos espacios de soluciones alternativas.

El término *heurística*, proviene del vocablo griego *heuriskein*, que podría describirse como *encontrar*, *descubrir* o *hallar*. Según, los registros históricos, este término fue utilizado por primera vez por el matemático George Polya en su libro *How to solve it* [243]. El autor quería expresar las reglas con las cuales las personas gestionan el *conocimiento común*, que pueden simplificarse como:

(i) Buscar un problema parecido que ya haya sido resuelto.
(ii) Determinar la técnica empleada para su resolución, así como las soluciones obtenidas.
(iii) Si es posible, utilizar la técnica y solución descrita en el punto (ii) para resolver el nuevo problema planteado.

Las técnicas heurísticas que se emplean en la resolución de un problema de optimización pueden ser de propósito general o de propósito específico. Los métodos heurísticos específicos deben ser diseñados utilizando el máximo conocimiento del problema a resolver y el análisis del modelo. Los procesos específicos que se han sido bien diseñados suelen tener un rendimiento significativamente más alto que los mostrados por las heurísticas generales. Las heurísticas de propósito más general, por el contrario, muestran otro tipo de bondades, como la sencillez, la robustez y la adaptabilidad de los procedimientos. Podemos considerar que existen dos tipos básicos de estos métodos: Constructivos y de Búsqueda. El primero de ellos puede desarrollarse por medio de la aplicación de técnicas como: Voraz, Descomposición, Reducción, Manipulación del Modelo, entre otras. El segundo, puede utilizar estrategias de búsqueda local como: *first improvement*, *best improvement* o Aleatorizada. Una de las grandes desventajas de los algoritmos heurísticos, consiste en la carencia de mecanismos que les permitan escapar de los óptimos locales. Debido a las limitaciones de los algoritmos heurísticos se fueron desarrollando nuevos algoritmos que permitieran superar las dificultades de las heurísticas, de esta forma Glover, en el año 1986, [119] introduce el término *Metaheurística*. Etimológicamente, este término, deriva de dos palabras de origen griego, *meta* y *heurística*. El segundo término ya fue descrito anteriormente. El prefijo *meta*, que podría traducirse como *más allá de*, *en un nivel superior*. Este tipo de algoritmo, de propósito general, consiste en procedimientos iterativos que guían a los métodos heurísticos básicos para explorar de forma inteligente el espacio de búsqueda de un problema. Según Osman y Kelly [234], las metaheurísticas se pueden definir como:

> Las metaheurísticas son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria en los que los heurísticos clási-

> cos no son efectivos. Las metaheurísticas proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica, sistemas neuronales y los mecanismos estadísticos.

Esa familia de propuestas incluyen a (pero no está restringida a) colonias de hormigas (Ant Colony Optimization, ACO), algoritmos evolutivos (Evolutionary Algorithms, EA), búsqueda local iterada (Iterated Local Search, ILS), enfriamiento simulado (Simulated Annealing, SA), y búsqueda tabú (Tabu Search, TS). Se pueden encontrar revisiones de metaheurísticas en [5, 27, 124]. De las diferentes descripciones de metaheurísticas encontradas en la literatura se han mencionado ciertas propiedades fundamentales que permiten caracterizar a este tipo de métodos:

- Las metaheurísticas son estrategias o plantillas generales que *guían* el proceso de búsqueda.
- El objetivo es una exploración del espacio de búsqueda eficiente para encontrar soluciones (casi) óptimas.
- Las metaheurísticas son algoritmos no exactos y generalmente son no deterministas.
- Pueden incorporar mecanismos para evitar las áreas del espacio de búsqueda no óptimas.
- El esquema básico de cualquier metaheurísticas es general y no depende del problema a resolver.
- Las metaheurísticas hacen uso del conocimiento del problema que se trata de resolver en forma de heurísticos específicos que son controlados por una estrategia de más alto nivel.

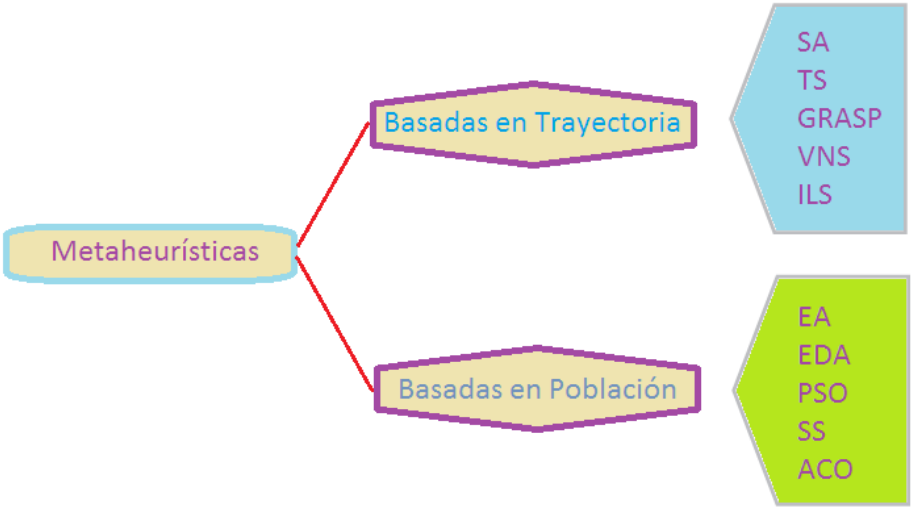
Podemos decir que las metaheurísticas son una especie de *plantilla general*, de enfoque no determinista, donde deben incorporarse todos aquellos datos específicos del problema (representación de las soluciones, vecindarios, operadores para manipularlas, función de evaluación, etc.) y que permiten atacar problemas con espacios de búsqueda de gran tamaño. Por lo tanto es de especial interés el correcto equilibrio (generalmente dinámico) que haya entre diversificación (o exploración) e intensificación (o explotación). El término *diversificación* se refiere a la exploración del espacio de búsqueda, mientras que *intensificación* se refiere a la explotación de alguna región concreta de dicho espacio. Establecer un equilibrio entre estos dos aspectos contrapuestos es muy importante, puesto que por un lado deben identificarse rápidamente las regiones prometedoras del paisaje de búsqueda global y por el otro lado no se debe dedicar mucho tiempo sobre aquellas regiones que ya han sido visitadas o que no contienen soluciones de alta calidad. A continuación ofrecemos una definición formal, según Luque [185], de los elementos incluidos en una metaheurística y que dependiendo de como la instanciemos produce un mecanismo u otro:

Definición 2.5.1 *Una metaheurística $\mathcal{M}$ está caracterizada por una tupla $\mathcal{M} = \langle \Theta, \Phi, \sigma, \mu, \lambda, \Xi, \tau \rangle$. Donde:*

- $\Theta = \{\theta_1, ..., \theta_\mu\}$ *es el conjunto de estructuras con las que trabaja $\mathcal{M}$. Cada $\theta_i \in \chi$, y χ es el conjunto de todas las posibles soluciones al problema.*
- $\Phi : \chi^\mu \times \Xi \longrightarrow \chi^\lambda$ *es el operador que modifica las estructuras de Θ.*
- $\sigma : \chi^{\mu+\lambda} \times \Xi \longrightarrow \chi^\mu$ *es una función que permite seleccionar las nuevas estructuras a ser tratadas en la siguiente iteración de $\mathcal{M}$.*
- μ *es el número de estructuras con las que trabaja $\mathcal{M}$.*
- λ *es el número de las nuevas estructuras generadas de la aplicación de Φ a Θ en cada iteración de $\mathcal{M}$.*
- $\Xi = \{\xi_1, \xi_2, ...\}$ *es un conjunto de variables de estado que contienen información relativa a la evolución de $\mathcal{M}$. Cualquier $\mathcal{M}$ contendrá, al menos, un elemento $\xi_1 = \theta^*$ que representa la mejor estructura encontrada.*
- $\tau : \Theta \times \Xi \longrightarrow \{true, false\}$ *es una función que decide la terminación del algoritmo.*

En la literatura existen diversas propuestas de clasificación de las metaheurísticas, según la perspectiva desde donde se les vea, por ejemplo, a través de criterios tales como, inspirados en la naturaleza o no, empleo de memoria, estructuras de vecindad, si se basan o no en poblaciones (conjunto de candidatos), en base a la naturaleza de la función objetivo, y otros criterios. Blum y Roli [27], propusieron una clasificación basada en las técnicas utilizadas por las metaheurísticas para encontrar posibles soluciones candidatas, en general, se establecen dos grandes grupos: *basadas en trayectoria* y *basadas en población*. Una visión general de algunas de las metaheurísticas existentes según los dos grandes grupos señalados se puede ver en la figura 2.7. La principal característica de los métodos trayectoriales es que parten de un punto y mediante exploración del vecindario van actualizando la solución candidata actual, formando una trayectoria. Los métodos basados en población, por su parte, realizan el proceso de búsqueda por medio de la evolución de un conjunto de puntos en el espacio de soluciones. No hay una técnica metaheurísticas lo suficientemente buena para todos los problemas de

Figura 2.7: Clasificación de las Metaheurísticas: Dos grandes Grupos



optimización combinatoria, de acuerdo con uno de los teoremas del *No Free Lunch* (NFL), todas estas técnicas se comportarían de manera similar en el promedio de "todos los problemas de optimización combinatoria". Sin embargo, un algoritmo podría ser superior a otro, si esté puede incorporar información específica del dominio del problema, esto según, el mismo teorema NFL [176, 331]. Una de las grandes desventajas de las metaheurísticas, son las evaluaciones innecesarias de un gran número de soluciones candidatas en el proceso de búsqueda de la solución óptima.

Muchas de las dificultades que pueden encontrar las técnicas metaheurísticas han tratado de ser superadas por medio de la incorporación de mecanismo de *Hibridación* entre estas. Las *metaheurísticas híbridas*, pueden proporcionar una conducta mucho más eficiente incorporando flexibilidad cuando se les utiliza para tratar de resolver problemas del mundo real o problemas de gran escala. La idea fundamental es lograr una sinergia mediante la combinación de las características fuertes de cada una de las metaheurísticas de una forma complementaria. Las metaheurísticas híbridas se pueden abordar desde dos enfoques básicos: **modo de colaboración** y **modo de integración** [256]. En el modo de colaboración la principal característica está dada por el intercambio de información entre las técnicas metaheurísticas involucradas, ya sea que se emplee un mecanismo secuencial o paralelo. Para el modo de integración es necesario que se haga uso de alguna otra metaheurísticas que esté subordinada a la metaheurística dada, el caso más emblemático, de este tipo de técnica, son los **Algoritmos Meméticos**

[285].

Otro aspecto de gran relevancia, ha sido la aparición, en el campo de las metaheurísticas, de un nuevo enfoque como lo es la integración de varias metaheurísticas en procesos de colaboración, donde se emplean diversas configuraciones (llamadas *topologías*) para emprender mecanismo de **cooperación** entre ellas. A esto se le suele llamar "búsquedas cooperativas" o "modelos cooperativos", incluyendo la definición de mecanismos para la selección de las técnicas metaheurísticas que van a intervenir en el proceso (llamadas en la literatura como *hiperheurísticas*). Los modelos cooperativos se caracterizan porque pueden proporcionar información sobre el dominio del problema a través de la utilización de diferentes metaheurísticas, esto permitirá lograr una ventaja de rendimiento en el algoritmo, lo cual es consistente con uno de los teoremas del NFL. Se han propuesto diferentes arquitecturas de moldeos de cooperación entre los que podemos mencionar **MAGMA** [210] y **COSEARCH** [300], dependiendo de la estrategia de comunicación y la composición de los elementos involucrados en el proceso (comúnmente llamados en la literatura científica como *agentes*).

2.5.1 Conceptos Comunes de las Metaheurísticas

Como se sabe, las metaheurísticas son estrategias inteligentes que se pueden utilizar para diseñar o mejorar procedimientos heurísticos muy generales, con la finalidad de elevar su rendimiento. La relevancia de las metaheurísticas se refleja en la publicación de un extenso número de artículos y libros que documentan el éxito de estas técnicas. Han sido empleadas para encontrar buenas soluciones, no necesariamente la solución óptima, a problemas de optimización combinatoria considerados computacionalmente difíciles [79, 83, 110]. Las metaheurísticas se basan en la utilización de diferentes paradigmas y proporcionan varias alternativas para evitar quedar atrapados en los llamados óptimos locales; que representa una de las dificultades que debe enfrentar este tipo de técnicas. Una de las características principales de las metaheurísticas es su particularidad de "personalizar" su funcionamiento, esto les imprime una caracterización de adaptabilidad, lo que les permite mejorar su desempeño de acuerdo a las particularidades del problema que se desea resolver. Existe una gran gama de aplicaciones de estas técnicas, incluyendo áreas como la programación con restricciones, inteligencia artificial, optimización combinatoria, logística, entre otras. Melían et al. [203], propusieron una serie de características deseables que cualquier metaheurísticas debería cumplir, y conforme a estas recomendaciones está técnica debería ser:

- **Simple**: Basadas en principios simples y claros (fácil de entender).
- **Precisa**: Las fases deben ser formuladas en términos concretos.
- **Consistente**: Los elementos deben ser deducidos desde sus principios.
- **Eficaz**: Debe proporcionar soluciones de alta calidad cercanas al óptimo u óptimos globales.
- **Eficiente**: Alta probabilidad de lograr las mejores soluciones y buen uso del tiempo de ejecución y espacio de memoria.
- **General**: Debe ser utilizable en una amplia variedad de problemas.
- **Adaptable**: Capaz de adaptarse a diferentes contextos de aplicación sin modificaciones importantes.
- **Robusta**: Debe ser muy sensible a pequeños cambios en el contexto del problema.
- **Interactiva**: Permitir un mejor desempeño a través del conocimiento de un usuario.
- **Múltiple**: Debe proporcionar soluciones alternativas de alta calidad.
- **Autónoma**: Su funcionamiento debe ser libre de parámetros o en su defecto los parámetros se puedan determinar automáticamente en la medida de lo posible.

En la literatura científica sobre metaheurísticas se utilizan los conceptos relacionados con su funcionamiento y estructuras, estos conceptos afectan directamente el rendimiento y la eficiencia en la resolución de problemas de optimización. Nos referimos a elementos estructurales como: representación de la solución, la definición de la función objetivo y la vecindad, así como estrategias funcionales como la intensificación/diversificación que determina el rendimiento de la metaheurística y la calidad de la solución.

2.5.1.1 Representación

Para poder realizar la implementación de una determinada metaheurística es necesario una codificación (representación), de las soluciones candidatas. La representación juega un papel importante, permitiendo la eficiencia y eficacia de las metaheurísticas, además de constituir un paso esencial para el diseño de estas técnicas. También resulta de gran importancia los operadores que se aplican a esta representación, como el vecindario, la recombinación, etc. De hecho, cuando se define una representación, hay que tener en cuenta cómo se evaluará la solución y cómo funcionarán los operadores de búsqueda. Según Talbi [299], una buena representación de la solución debe poseer las siguientes características:

- **Integridad**: Una de las principales características de la representación es la integridad, es decir, todas la soluciones asociadas al problema deben estar representadas en dicha codificación.
- **Conectividad**: La característica de conectividad es muy importante en el diseño de cualquier algoritmo de búsqueda. Una ruta de búsqueda debe existir entre dos soluciones de espacio de búsqueda. Cualquier solución del espacio de búsqueda, especialmente la solución óptima global, debe ser alcanzada.
- **Eficiencia**: La representación debe ser fácil de manipular por los operadores de búsqueda. Las complejidades de tiempo y el espacio de los operadores que se ocupan de la representación deben reducirse.

2.5.1.2 Función Objetivo

Para poder verificar si una determinada solución candidata ha logrado el objetivo de llegar a ser un óptimo es necesaria la utilización de una *Función Objetivo*. Está asociada con el valor real de la solución candidata que describe la calidad o el *fitness* de dicha solución. Es un elemento de gran importancia que debe ser considerado con mucho cuidado al momento del diseño de una metaheurística, ya que es quien aporta información importante que nos permite dirigir la búsqueda hacia aquellas zonas del paisaje de búsqueda donde potencialmente podemos alcanzar soluciones candidatas de alta calidad. La función objetivo puede definirse como $f : S \longrightarrow \mathbb{R}$, es por ello que representa un valor absoluto y permite un completo ordenamiento de todas las soluciones del espacio de búsqueda en términos de la calidad de cada una de ellas. $\mathbb{R}$ representa el espacio de soluciones. Así, existirá alguna función de decodificación d que puede ser aplicada, tal que: $d : R \longrightarrow S$, generará una solución que puede ser evaluada por la función f [86].

2.5.1.3 Vecindario

La noción de vecindario es esencial para comprender la búsqueda local. Intuitivamente, una solución s' se denomina un vecino de s si es posible llegar al primero desde el segundo en un solo paso (por la aplicación de un operador llamado *movimiento*). La vecindad $\mathcal{N}(s)$ de una solución s es el conjunto

de todos sus vecinos. Hay que acotar que las relaciones vecinales son a menudo –aunque no siempre– simétricas. Podría considerarse la vecindad como el subconjunto del espacio de búsqueda que contiene soluciones que están "**próximas**" respecto a una determinada solución que está siendo objeto de exploración y que se considera como solución inicial. La vecindad se genera al aplicar una función (llamada *función de vecindad*) sobre una determinada solución inicial y aplicando una pequeña perturbación sobre ésta. Las soluciones vecinas son generadas por medio de un mecanismo de *movimiento* y son seleccionadas y aceptadas de acuerdo a algún criterio predefinido con anterioridad. Podemos definir el vecindario como una función $\mathcal{N} : N \longrightarrow 2^N$ aplicada sobre el conjunto de individuos de N, la cual se asigna a cada solución en el espacio de búsqueda N (digamos $s \in N$), un conjunto de soluciones de $\mathcal{N}(N)$, de esta forma una solución s' se considerará como vecina de s si la solución s' es el resultado de aplicar una leve perturbación y se encuentra en la vecindad de s ($s' \in \mathcal{N}(s)$). De acuerdo al tipo de representación de problema que se esté abordado dependerá la estructura de vecindad [27].

2.5.1.4 Soluciones Iniciales

Las soluciones iniciales son los puntos de partida que permiten al proceso de búsqueda ir realizando mejoras a cada individuo tomado inicialmente como punto de partida. Dichas mejoras se logran por medio del concepto de *vecindad*, que ha sido introducido en el apartado anterior. Existen varios métodos para generar las soluciones iniciales tales como: enfoque al azar, voracidad o por medio de otras metaheurísticas. Generalmente es preferible implementar un procedimiento sencillo para la generación de las soluciones iniciales, lo que permitirá realizar un gran número de iteraciones globales. Los métodos más sofisticados consumen mucho más tiempo de cómputo en la generación de soluciones iniciales lo que podría conducir a la reducción del muestreo del espacio de soluciones.

2.5.1.5 Diversificación e Intensificación

La idea básica de este tipo de conceptos es tratar de explorar y explotar las porciones del espacio de búsqueda que resultan prometedoras como resultado del cumplimento de ciertos criterios prefijados, con el fin de asegurarse que la mejor solución puede ser encontrada en esa porción del espacio de búsqueda. Según Duarte [223] la **intensificación** es la forma de definir el proceso de búsqueda más exhaustivo que puede llevar a cabo una metaheurística en una vecindad dada. Mientras que la **diversificación** es la forma de definir el proceso mediante el cual la metaheurística es capaz de visitar diversas vecindades lejanas. Debe existir un marcado equilibrio entre la diversificación y la intensificación ya que son contrapuestas, es decir, cuanto más tiempo se dedique a intensificar la búsqueda en una región dada menos tiempo podrá dedicarse a otras regiones inexploradas (y viceversa).

2.5.2 Métodos Trayectoriales

En esta parte se describen las metaheurísticas identificadas como métodos de trayectoria. Estos métodos de trayectoria son los que trabajan en una única solución, se denominan de esta forma porque el mecanismo de búsqueda que emplean se caracteriza porque avanzan definiendo una trayectoria en el espacio de búsqueda. Las características de los métodos de trayectoria proporcionan conocimiento sobre el rendimiento del algoritmo y su potencia con respecto a la instancia que se aborda. Las siguientes subsecciones proporcionan una introducción y estudio de diversos tipos de métodos trayectoriales más comúnmente utilizados o que más han sido documentados en el ámbito científico de la optimización combinatoria.

2.5.2.1 Hill Climbing

Esta es la forma más sencilla para realizar la búsqueda local en un espacio de soluciones, y se obtiene al perturbar una solución tomada inicialmente, la cual será sustituida si la nueva solución, recién generada, supera, en términos de calidad, a la solución antes de la perturbación. *Hill Climbing* (HC), es una estrategia iterativa que incorpora mejoramiento repetitivo y ha sido utilizado para resolver problemas de optimización combinatoria [208]. Esta técnica se relaciona con el gradiente de ascenso, pero no es necesario conocer el valor del gradiente o incluso su dirección: sólo requiere probar iterativamente nuevas soluciones candidatas en la región del actual candidato, y adoptar las nuevas soluciones si se obtiene alguna mejora respecto al candidato inicial. Esto permitirá subir la colina hasta llegar a alcanzar óptimos locales. Existen algunas variantes de esta técnica como por ejemplo *Steepest Ascent Hill-Climbing* y *Steepest Ascent Hill-Climbing with Replacement*. Hill Climbing se considera un método no exhaustivo y sin memoria, ya que no explora todo el espacio de búsqueda ni mantiene un historial de las soluciones visitadas. Una forma de escapar del estancamiento se logra por medio de la aplicación de reinicios aleatorios.

Algoritmo 1: Pseudocódigo del Algoritmo Hill Climbing

```
   input : S_i
   output: Mejor Individuo Alcanzado.
 1 begin
 2     S_i ← GeneraCandidato();
 3     while Criterio de parada No encontrado do
 4         S_c ← Vecindario(Copy(S_i));
 5         if EvaluarCalidad(S_c)>EvaluarCalidad(S_i) then
 6             S_i ← S_c;
 7         end
 8     end
 9 end
10 return S_i;
```

El seudocódigo correspondiente para la técnica HC se puede ver en el Algoritmo 1. Esté método de búsqueda ha sido empleado con éxito en diferentes áreas como: Sistemas Biométricos [132], Mapas Topográficos [311] y Cifrado [33].

2.5.2.2 Simulated Annealing (SA)

El modelo de funcionamiento de este algoritmo proviene del proceso físico del templado de metales. La fundamentación de este control se basa en el trabajo de Metropolis et al. [207], en el campo de la termodinámica estadística. Para conseguir que la estructura de las moléculas del metal posea las propiedades deseadas de flexibilidad o resistencia, se hace necesario controlar la velocidad en el que se aplica el proceso de templado (enfriamiento). Si esto se aplica adecuadamente, el estado final del metal será un estado de mínima energía. El fin general de este tipo de técnicas es encontrar una muy buena aproximación al valor óptimo de una función (definida) en un espacio de búsqueda de tamaño considerable. En cada fase (iteración), el algoritmo de recocido simulado (*Simulated Annealing*, SA) explora algunos vecinos del estado actual s y aplicando una probabilidad decide entre efectuar un movimiento a un nuevo estado s' o permanecer en el estado s. El algoritmo SA fue una de las primeras

técnicas que implementó una estrategia explícita para escapar de los óptimos locales (por medio de la aceptación de una fracción de soluciones sin mejora). Surge de la mecánica estadística [168] y fue el primero que se presentó como un algoritmo de búsqueda para problemas de optimización combinatoria [44, 82, 89]. Su fácil implementación junto con sus propiedades de convergencia que combina la técnica del Hill-Climbing con dirigirse por un camino aleatorio de tal manera que se pueda lograr tanto completitud como eficiencia, esto ha permitido que ésta sea una técnica muy popular en las dos pasadas décadas. Es utilizado típicamente para resolver problemas discretos de optimización, y en menor medida, en problemas de dominio continuo.

En cada iteración del algoritmo SA, aplicado a un problema de optimización combinatoria discreto, los valores para dos soluciones candidatas se comparan entre sí (la solución actual y la nueva solución generada). Las soluciones que muestran alguna mejora son aceptadas, mientras que una fracción de las soluciones que no presentan mejoras son aceptadas con el fin de escapar de los óptimos locales. La probabilidad de aceptar soluciones sin mejora dependen de un parámetro denominado *temperatura*, el cual se calcula generalmente siguiendo la ley de distribución de Boltzmann [116]. Éste parámetro se decrementa durante el proceso de búsqueda, de tal forma que al inicio del proceso existe una alta probabilidad de aceptar movimientos *uphill* y gradualmente se decrementa, convergiendo a un algoritmo simple de mejora iterativa. Las soluciones iniciales son generadas por medio de mecanismo como aleatoriedad, heurísticos o por un proceso de construcción. El algoritmo utiliza dos mecanismo combinando *movimientos aleatorios* y *mejoras iterativas*.

El algoritmo SA comienza con una solución inicial $s \in S$, para genera una solución vecina $s' \in \mathcal{N}(S)$ (aleatoriamente o mediante alguna regla específica). Una solución candidata s', es aceptada basándose en una probabilidad de aceptación, tal que:

$$P\{\text{Acepta } s'\} = \begin{cases} exp[-(f(s') - f(s))/t_k] & \text{Si } f(s') - f(s) > 0 \\ 1 & \text{Si } f(s') - f(s) \leq 0 \end{cases} \tag{2.12}$$

Donde t_k representa el parámetro de la temperatura en la iteración k, tal que:

$$t_k > 0 \text{ para todo } k \text{ y } \lim_{k \longrightarrow \infty} T_k = 0. \tag{2.13}$$

Está probabilidad de aceptación representa un elemento básico para el mecanismo de búsqueda en el algoritmo de recocido simulado. Si la temperatura se reduce lentamente en forma suficiente, entonces el sistema puede llegar a un equilibrio (estado estable) en cada iteración k. Este equilibrio sigue la distribución de Boltzmann, la cual puede ser descrita como la probabilidad de que el sistema comience en el estado $s \in S$ con una energía $f(s)$ con temperatura T, tal que:

$$P\{\text{Estado } s \text{ con Temperatura } T\} = \frac{exp(-f(s)/t_k)}{\sum_{s'' \in S} exp(-f(s'')/t_k)} \tag{2.14}$$

La elección de un esquema de enfriamiento apropiado es crucial para el rendimiento del algoritmo. El esquema de enfriamiento define el valor de la temperatura en cada iteración k, a través de una función de temperatura. El pseudocódigo para esté tipo de técnica se puede ver en el algoritmo 2. Una forma de determinar la temperatura inicial es muestrear el espacio de búsqueda con una ruta aleatoria y evaluar de forma genérica el promedio y la varianza de la función objetivo, pero existen diversos esquemas más sofisticados como los mostrados en [159]. Existen variantes del algoritmo SA, cuyo objetivo es controlar diferentes formas de esquematización de enfriamiento, con el fin de mejorar el tiempo de convergencia a un óptimo global. Algunas de estas variantes son: *Fast Annealing*, *Very Fast Simulated Re-annealing* (VFSR) o *Adaptive Simulated Annealing* (ASA) [159]. También hay que mencionar su versión paralela como se muestra en [98].

Algoritmo 2: Pseudocódigo del Algoritmo Simulated Annealing

```
input  : t , S_i
output: Mejor Individuo Alcanzado.
1  begin
2      S_i ← GeneraCandidato();
3      t ← ObtenerTemperatura();
4      Best ← S_i;
5      while Criterio de parada No encontrado do
6          S_c ← Vecindario(Copy(S_i));
7          if EvaluarCalidad(S_c)>EvaluarCalidad(S_i) or Rand[0,1] <
             e^((EvaluarCalidad(S_c) − EvaluarCalidad(S_i)) / t) then
8              S_i ← S_c;
9          end
10         t ← ActaulizarTemperatura();
11         if EvaluarCalidad(S_i)>EvaluarCalidad(Best) then
12             Best ← S_i;
13         end
14     end
15 end
16 return Best;
```

Algunas de las áreas donde ha sido aplicado el algoritmo SA involucra la estructuración de rutas en circuitos impresos (VLSI[5]) [45], en software (para la predicción de la calidad) [34], problemas de planificación [1], entre otros.

2.5.2.3 Tabu Search (TS)

El término Búsqueda Tabú (**Tabu Search**, TS) fue introducido en 1986 por Fred Glover en el mismo artículo que introdujo el término metaheurística [119]. Los principios fundamentales de la búsqueda fueron elaborados en una serie de trabajos a finales de los 80's y principios de los años 90, que fueron luego unificados en el libro "Tabu Search" en 1998 [128]. El destacado éxito de la búsqueda tabú para resolver problemas de optimización difíciles, principalmente aquellos que han surgido en aplicaciones reales, ha permitido una explosión de nuevos usos durante los últimos años, que aparecen resumidas en [127]. La búsqueda tabú se basa en dos elementos básico: el uso de *memoria adaptativa* y la *exploración sensible*. La utilización de memoria adaptativa es una característica que permite que los procesos sean capaces de avanzar hacia zonas del paisaje de búsqueda, donde posiblemente se localizan las soluciones candidatas que potencialmente pueden llegar a ser óptimos globales, de forma eficaz y eficiente. Dado que los procesos locales son guiados por la obtención de información a lo largo de los procesos de búsqueda, la técnica tabú concuerda con diseños que en contraparte confían en procedimientos semi-aleatorios, que utilizan una forma de muestreo. El uso de memoria adaptativa contrasta, también, con los diseños de memoria regidos por estrategias de ramificación y acotación. El uso de exploración responsiva empleada en la búsqueda tabú se deriva del supuesto de que una elección errónea de la estrategia podría proporcionar mejor información que una elección mucho más buena, realizada al azar. Se piensa que una elección estratégica mala proporciona buenas pistas que permiten

[5] Acrónimo en inglés de Very Large Scale Integraton

guiar la búsqueda hacia zonas más prometedoras. Es por ello que la exploración responsiva unifica los principios de la búsqueda inteligente; explotando así las características de aquellas soluciones buenas, al tiempo que explora nuevas regiones del espacio de búsqueda más prometedoras.

La búsqueda tabú se caracteriza porque establece un mecanismo para evitar volver a revisar soluciones candidatas ya exploradas y por tanto evitar caer en ciclos, para ello se utiliza una estructura almacenada en memoria temporal a la que se le llama *lista tabú* y que permite llevar un registro de las soluciones ya evaluadas [47]. Esto permite aceptar sólo aquellas soluciones vecinas si el movimiento correspondiente no es tabú. El estado tabú de un determinado movimiento no es permanente ya que su duración se mantiene durante unos cuantos pasos de búsqueda y este valor se conoce como *tabu tenure*. El valor del *tabu tenure* puede ser fijo para todos los movimientos del proceso de búsqueda o puede variar para diferentes movimientos o etapas del proceso de búsqueda. La membresía a la lista tabú puede realizarse a través de diferentes criterios, como por ejemplo, orden de aparición, valor de la función objetivo, entre otros.

La búsqueda tabú procede de igual forma que lo hace cualquier método de búsqueda local o búsqueda por vecindario, realizando un proceso iterativo desde un solución candidata dada hacia otra que está ubicada en el mismo vecindario, hasta lograr una condición de terminación. Cada solución x estará asociada a un vecindario $N(x) \in X$, y será posible llegar a otra solución candidata $x' \in N(x)$ desde x a través de un proceso llamado *movimiento*. Es posible contrastar TS con un método simple de ascenso, que unicamente permite movimientos entre los candidatos vecinos que producen alguna mejora con respecto a la función objetivo y finalizan cuando no se detectan mejoras. El inconveniente evidente del método de descenso es que los óptimos obtenidos, en la mayoría de los casos, no será un óptimo global. En cambio, TS permite la realización de movimientos que pueden llegar a desmejorar la solución candidata inicial, estos movimientos se realizan desde una estructura del vecindario modificada $N^*(x)$. Este vecindario resulta de llevar un registro histórico de los movimientos realizados durante el proceso de búsqueda. Este registro puede ser implementado por medio del uso de memoria a corto y largo plazo.

Existen una serie de elementos fundamentales a considerar como lo son:

- Uso de Memoria
- Intensificación y Diversificación
- Período Tabú
- Criterio de aspiración

Uso de Memoria: En la búsqueda tabú la estructura de memoria funciona refiriéndose a cuatro dimensiones principales: reciente, frecuencia, calidad e influencia. Las memorias basadas en lo *reciente* y en *frecuencia* permiten el balance entre diversificación e intensificación que toda técnica de búsqueda heurística debe poseer. La dimensión de lo reciente es quizá la estructura de memoria mayormente utilizada en las implementaciones de búsqueda TS. La dimensión de *calidad* se refiere a la posibilidad de diferenciar la cualidad de las soluciones encontradas a lo largo de la búsqueda, permitiendo identificar características similares de las diversas soluciones que han sido consideradas como buenas o las rutas que conducen hasta ellas. Constituyen un mecanismo que permite premiar las acciones que conllevan a soluciones de calidad y penalizar a aquellas que nos llevan a soluciones pobres. La última dimensión referida a la *influencia* permite considerar el impacto de las acciones tomadas durante el proceso de búsqueda, no sólo en lo referente a las buenas soluciones, sino además, la estructura de las mismas. La memoria utilizada en la búsqueda tabú es tanto *explícita* como *implícita*. En la primera, se van almacenando en memoria aquellas soluciones completas, comúnmente soluciones élite visitadas durante el proceso de búsqueda, mientras que el segundo aspecto, hace referencia al almacenamiento

de información sobre determinados atributos de las soluciones que sufren alguna modificación al pasar de un estado a otro. En algunos casos, la denominada memoria explícita es usada para evitar la visita de soluciones, ya evaluadas, más de una vez.

Intensificación y Diversificación: Las estrategias de intensificación se basan en la modificación de reglas que permitan realizar combinaciones de movimientos y la utilización de características relevantes de las diversas soluciones encontradas a lo largo del proceso de búsqueda, esto permitirá la identificación de conjuntos de soluciones élite para la utilización de sus atributos más relevantes para incorporarlos en la creación de nuevas soluciones. La membresía en el conjunto élite a menudo se determina mediante el establecimiento de un umbral que está relacionado con el valor de la función objetivo de la solución encontrada durante la búsqueda. Las estrategias de diversificación, sin embargo, buscan incorporar nuevos atributos y combinaciones de atributos que no se incluyeron dentro de las soluciones generadas previamente. En una forma, estas estrategias se comprometen a conducir la búsqueda a regiones diferentes a las ya exploradas con anterioridad. Es importante tener en cuenta que la intensificación y diversificación no son mutuamente opuesta, en cambio se refuerzan mutuamente.

Período Tabú: TS selecciona atributos de las soluciones exploradas con anterioridad y los registra mediante la creación de una o varias listas tabú. Estas listas almacenan los atributos tabú-activos e identifican, explícita o implícitamente, estados tabú actuales. El período tabú (*tabu tenure*) es el número de iteraciones o el tiempo, que un determinado elemento (movimiento o atributo) se mantiene en la lista tabú. Si todos los elementos están identificados con la misma *tenure*, ésta estará identificada por el tamaño de la lista tabú, si todos los elementos de la lista tabú no poseen la misma *tenure*, un determinado elemento que se registró en la lista tabú antes que otro podrá salir mucho después. Estas variaciones del período tabú de los atributos hace posible crear diferentes formas de balance entre las estrategias de memoria a corto y a largo plazo.

Criterio de aspiración: Permite que un determinado movimiento sea admitido aunque esté clasificado como tabú, eliminando así la clasificación tabú aplicada con anterioridad. Aunque muchas de las aplicaciones que podemos encontrar en la literatura emplean únicamente un tipo de criterio simple de aspiración, el cual consiste en eliminar una clasificación tabú de un determinado movimiento de ensayo cuando dicho movimiento conduce a una solución de mayor calidad que la mejor obtenida hasta ahora. Podemos considerar dos tipos de aspiración: de movimientos y de atributos. Según Glover y Laguna [128], podemos encontrar diferentes variantes del criterio de aspiración, tales como:

- Aspiración por Default: Si todos los movimientos disponibles se clasifican como tabú, y no hay otro criterio de aspiración, entonces se seleccionará el movimiento "menos tabú".
- Aspiración por Objetivo: si el movimiento produce una solución mejor que la mejor de la región donde se encuentra la solución (local). Si el movimiento produce una solución mejor que la mejor obtenida hasta el momento (global).
- Aspiración por Dirección de Búsqueda: Si se obtiene una mejora en la nueva solución y la dirección de la búsqueda no ha cambiado.
- Aspiración por Influencia: Si una alta influencia de movimiento se ha realizado desde el establecimiento del estado tabú.

. Otros criterios de aspiración pueden ser revisados en [148]. El pseudocódigo para el método de Búsqueda Tabú se puede ver el Algoritmo 3. El método TS ha sido empleado en diferentes áreas, como por ejemplo, la alineación de múltiples secuencias [264], generación de mapas [327], simulaciones

Algoritmo 3: Pseudocódigo del Algoritmo Tabu Search

```
   input  : S_i
   output: Mejor Individuo Alcanzado.
 1 begin
 2     L ← InicializarTList();
 3     Mejor ← S_i;
 4     while Criterio de parada No encontrado do
 5         if Longitud(L)>MaxL then
 6             RemoverAntiguo(L);
 7         end
 8         S_c ← Vecindario(Copy(S_i));
 9         for n ← 1 to t do
10             S_w ← Vecindario(Copy(S_i));
11             if S_w ∉ L and (EvaluarCalidad(S_w)>EvaluarCalidad(S_c) or S_c ∈ L) then
12                 S_c ← S_w;
13             end
14         end
15         if S_c ∉ L and EvaluarCalidad(S_c)>EvaluarCalidad(S_i) then
16             S_i ← S_c;
17             AgregarToList(L,S_c);
18         end
19         if EvaluarCalidad(S_i)>EvaluarCalidad(Best) then
20             Mejor ← S_i;
21         end
22     end
23 end
24 return Mejor;
```

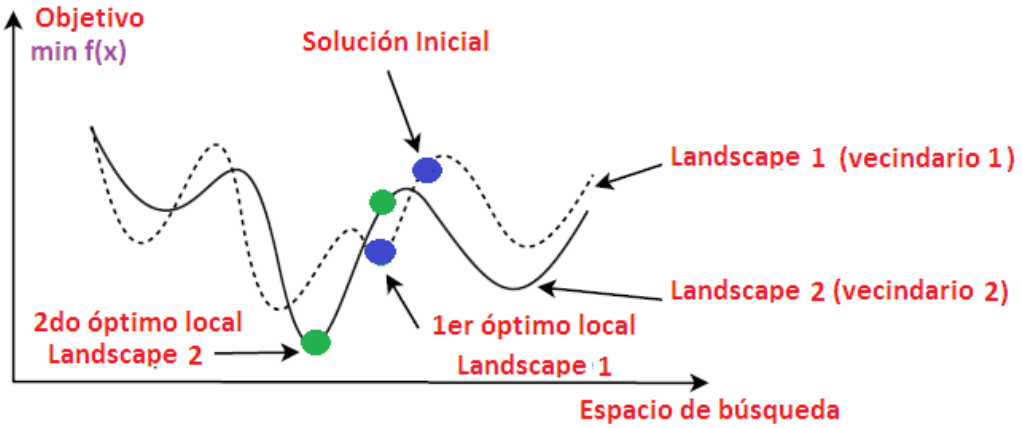
[78], planificación de transporte [338], juegos [293], entre otras.

2.5.2.4 Variable Neighborhood Search (VNS)

La Búsqueda por Entornos Variables (Variable Neighborhood Search, VNS) es una técnica reciente que se fundamenta en cambiar de forma sistemática la estructura del entorno. La idea original surgió bajo la consideración de varias estructuras de entornos y variarlas sistemáticamente para evitar los óptimos locales. El VNS básico genera una solución del entorno de la solución analizada, ejecuta una búsqueda (normalmente monótona) local desde ella hasta lograr alcanzar un óptimo local, que reemplaza a la solución actual si se ha detectado una mejora y modifica la estructura del entorno en caso contrario. Una versión de esta estrategia es la búsqueda descendente por entornos variables (VND), esta aplica una búsqueda monótona basada en entornos, cambiando de forma sistemática la estructura de entornos cuando se alcanza un mínimo local. Otra posibilidad es utilizar una selección de los entornos, a través de la elección de un conjunto de sus atributos (variables), de tal manera que se produzca una descomposición, y por lo tanto se definen determinados subproblemas, este modelo es denominado búsqueda por descomposición de entornos variables (VNDS). VNS trabaja bajo la premisa de que el uso de diversas vecindades de búsqueda local puede generar diferentes óptimos

locales y que los óptimos globales son un óptimo local para un vecindario determinado [299], véase la figura 2.8. Según Hassen at al. [141], está basada en tres hechos simples:

Figura 2.8: Uso de diferentes vecindades en VNS



- Un mínimo local con una estructura de entornos no lo es necesariamente con otra.
- Un mínimo global es mínimo local con todas las posibles estructuras de entornos.
- Para muchos problemas, los mínimos locales con la misma o distinta estructura de entornos están relativamente cerca, esto implica que los óptimos locales proporcionan información acerca del óptimo global.

El pseudocódigo básico para la Búsqueda de Entornos Variables se puede ver en el algoritmo 4. Está técnica ha sido aplicada con éxito en diversas áreas, como problemas de planificación [8], problema de asignación cuadrática [337], el problema del $p-centro$ [214], el problema de la $p-mediana$ [140], entre otros.

2.5.2.5 Iterated Local Search (ILS)

La búsqueda local iterada o *Iterated Local Search* (ILS) [124], establece como forma de funcionamiento la aplicación repetida de perturbaciones y búsquedas locales sobre una solución candidata dada. Utilizando un mecanismo de "*reconstrucción y mejora*" sobre una solución particular, se logra obtener soluciones cercanas al óptimo global. Esta técnica fue propuesta a finales de los 90's en la tesis doctoral de Stützle [290]. Este sencillo y rápido algoritmo basado en trayectoria tiene como dato de entrada una solución candidata generada previamente, a partir de la cual el proceso se divide en tres fases. En la primera, se introduce una perturbación o modificación en la solución inicial. Dicha variación (perturbación) no puede ser ni muy pequeña, lo que permitiría que el algoritmo llegue a alcanzar mínimos locales de los que no sea capaz de escapar, ni muy grande, en cuyo caso el algoritmo podría no seguir un camino definido, y por tanto se comportará como una técnica de búsqueda local que utiliza reinicios aleatorios. En la segunda fase el algoritmo utilizará una operación de búsqueda local (Local Search - LS) sobre la solución candidata actual. Finalmente se debe establecer una operación de actualización que determine, en base a un criterio de aceptación, si la solución generada por medio de la operación de LS debe sustituir a la solución previa. Normalmente esta operación consiste en comparar la calidad de ambas soluciones, quedándose con la que proporciona mejoría en la solución del problema. En algunos esquemas básicos la metaheurística se mejora incorporando características adicionales, como por ejemplo, la utilización de una estructura que guarde el histórico

Algoritmo 4: Pseudocódigo del Algoritmo VNS

```
   input  : S_i
   output: Mejor Individuo Alcanzado.
 1 begin
 2     S_i ← GeneraCandidato();
 3     while Criterio de parada No encontrado do
 4         l ← 1;
 5         while l ≤ l_max do
 6             S_c ← Vecindario(S_c,l);
 7             if EvaluarCalidad(S_c)>EvaluarCalidad(S_i) then
 8                 S_i ← S_c;
 9                 l ← 1;
10             else
11                 l ← l + 1;
12             end
13         end
14     end
15 end
16 return S_i;
```

para determinar el reinicio del algoritmo si no se consigue mejorar la solución candidata actual. Según Den et al. [76], la ILS posee cuatro elementos principales:

- Un procedimiento para generar una solución inicial.
- Un procedimiento de búsqueda local.
- Un esquema de perturbación de la solución.
- Un criterio de aceptación, que decide desde cuál solución se continua la búsqueda.

La definición de cada uno de los elementos influenciará la efectividad de la generación de una ruta aleatoria en el espacio de búsqueda por parte del algoritmo ILS.

En cualquier caso, se dará mayor relevancia en este algoritmo a la operación de búsqueda local. Esta operación debe ser muy efectiva y rápida en la medida de lo posible, ya que es la que permitirá la incorporación de mejoras sobre la solución candidata actual. La efectividad permitirá lograr mejores resultados al aplicar el algoritmo ILS. Desde un punto de vista de simplicidad, esta técnica de búsqueda puede ser considerada como una plantilla que será utilizada por el algoritmo en cada una de las iteraciones. Podemos optimizar la LS para realizar búsquedas más potentes, por ejemplo, utilizando pequeñas ejecuciones de alguna heurística o tal vez una metaheurísticas (enfriamiento simulado, hill climbing, búsquedas tabú, etc.). Prestar atención al diseño de un algoritmo de búsqueda local que pueda adaptarse de la mejor forma posible a las características del problema que se está tratando, es de suma importancia, y en muchas ocasiones no se trata de un proceso trivial, donde quizás se deba dar mayor importancia a la rapidez del algoritmo que a la eficiencia del mismo. En problemas que poseen grandes dimensiones, la operación de búsqueda puede resultar muy costosa, es por ello, que la elección de una excelente técnica de perturbación resulta crítica para que el algoritmo tenga un buen funcionamiento, puesto que debe ser una operación complementaria a la búsqueda local. Es por ello la importancia para encontrar soluciones óptimas la determinación de un buen balance entre la intensificación (proporcionado por la LS) y diversificación (que corresponde a las perturbaciones). El pseudocódigo básico para la Búsqueda Local Iterada se puede ver en el algoritmo 5. Está técnica

ha sido aplicada con éxito en diversas áreas, como problemas de planificación [54, 183], partición de grafos [193], problemas de máxima satisfactibilidad [286], asignación cuadrática [291], y el problema del coloreado de grafos [237], entre otros.

Algoritmo 5: Pseudocódigo del Algoritmo ILS

```
   input : S_i
   output: Mejor Individuo Alcanzado.
 1 begin
 2     S_i ← GeneraCandidato();
 3     S_i ← BLocal(S_i);
 4     S_c ← S_i;
 5     while Criterio de parada No encontrado do
 6         S_i ← Perturbación(S_i);
 7         S_i ← BLocal(S_i);
 8         if EvaluarCalidad(S_i)>EvaluarCalidad(S_c) then
 9             S_c ← S_i;
10         end
11     end
12 end
13 return S_c;
```

2.5.3 Métodos Poblacionales

También se les puede llamar búsqueda basadas en grupos, en la cual se sustituye la solución actual que recorre el espacio de soluciones, por un grupo de soluciones que realizan intercambio de características entre ellas al momento de la realización del recorrido. Además de los diferentes movimientos que pueden ser aplicados a las soluciones candidatas que integran este conjunto, al cual podemos llamarlo grupo (también, población de búsqueda), se requiere considerar otros operadores para obtener nuevas soluciones a partir de las generadas previamente. Las estrategias de búsquedas poblacionales tuvieron su inicio a raíz del famoso *Algoritmo Genético* propuesto en [151]. Estas técnicas operan en forma intrínseca por medio de un mecanismo de exploración paralelo del paisaje de búsqueda y su efectividad depende, en gran medida, de cómo se maneje dicha población, lo que se contrapone a las técnicas trayectoriales que sólo utilizan una única solución para manipularla, normalmente, por medio de un mecanismo de entornos. Las siguientes subsecciones proporcionan una introducción y estudio de diversos tipos de métodos poblacionales abarcando los más comúnmente utilizados o que más han sido documentados en el ámbito científico de la optimización combinatoria.

2.5.3.1 Genetic Algorithm (GA)

Este tipo de técnicas tiene un fuerte basamento biológico, es decir, se trata de imitar los procesos evolutivos por medio de la selección natural. En la naturaleza los individuos de una población están en constante competición por la obtención de los recursos, tales como, comida, agua, refugio, etc. En este tipo de competencia, aquellos individuos que poseen las características de fortaleza son los que tienen mayores posibilidades de éxito en la lucha por la obtención de los diversos recursos necesarios para satisfacer sus necesidades básicas, y por lo tanto serán los más aptos para la supervivencia, lo

que permitirá que su descendencia tenga la mejor información genética del grupo. La estructura de un Algoritmo Genético (Genetic Algorithm **GA**) simple se puede resumir de la siguiente manera: Durante la iteración *t*, un GA mantiene una población de soluciones potenciales (cromosomas), $P_{(t)} = \{x_1^t, ..., x_n^t\}$, cada candidato a solución x_i^t es evaluado para producir una medida de calidad o *fitness*. Por tanto, una nueva población (generación *t+1*) es formada por medio de la selección de los mejores individuos. Algunos de los miembros de esa nueva población han sufrido pequeñas alteraciones al ser sometidos a operadores de cruce y mutación, lo que permitirá formar nuevas posibles soluciones. El operador de cruce combina las características de los cromosomas de, al menos, dos padres para formar descendientes por medio del intercambio de segmentos de genes de sus padres. También, interviene el operador de *mutación* que altera en forma arbitraria uno o más genes de un cromosoma seleccionado por medio de cambios aleatorios con una probabilidad establecida. Los algoritmos genéticos, en general, deben poseer los siguientes 5 componentes [283]:

- una representación por medio de genes para los individuos de la población del problema,
- un mecanismo para crear la población inicial de posibles soluciones
- una función objetivo, que permita evaluar la calidad de cada una de las posibles soluciones en términos del *fitness*
- un operador genético que permita alterar la composición de sus descendientes
- establecer los valores adecuados de los diversos parámetros que utilizará el algoritmo (tamaño de la población, probabilidad de aplicación de los operadores genéticos, tipo de aridad, etc.)

El pseudocódigo para un algoritmo genético básico se muestra en el Algoritmo 6. Los aspectos más resaltantes de los GA se pueden resumir de las siguiente manera:

- Representación de individuos.
- Población inicial.
- Función de adaptación.
- Selección.
- Reproducción.
- Mutación.

Representación de individuos: El esquema de representación de los individuos de la población es el que define la forma en que se asocian los cromosomas con las posibles soluciones del problema. Para elaborar el esquema de representación, se utilizan los parámetros que identifican a las soluciones, y luego se utiliza una codificación de estos parámetros dentro del denominado cromosoma. El esquema de representación debería garantizar que exista al menos una posible codificación para cada una de las soluciones candidatas posibles. Las soluciones que no estén dentro de este esquema de representación de cromosomas no serán exploradas por el algoritmo genético.

Población inicial: La población inicial es un factor muy relevante y representa la principal fuente de material genético para un GA. La población inicial debería permitir una distribución de los diferentes cromosomas por el espacio de soluciones. La manera más simple de cumplir con este objetivo es elegir cromosomas al azar. El uso de esquemas híbridos [221, 222] puede ayudar a generar una población de partida (inicial) conformada por soluciones de mediana calidad, lo que permite el ahorro de tiempo en los procesos de evolución, siempre y cuando se garantice una dispersión suficiente para la búsqueda.

Función de adaptación: La función de adaptación, comúnmente denominada *fitness*, proporciona una medida de la calidad de cada cromosoma como una posible solución candidata del problema, y permite determinar su probabilidad de ser escogido para la fase de reproducción, y de esta forma,

Algoritmo 6: Pseudocódigo del Algoritmo GA Básico

```
   input :
   output: Mejor Individuo Alcanzado.
 1 begin
 2     P ← IPoblación();
 3     Evaluar(P) ;
 4     S_θ ← Mejor(P);
 5     while No Alcance el Criterio de Parada do
 6         while i ∈ N/2 do
 7             F ← DosPadres(P);
 8             if Aleatorio[0,1)>p_Cruce then
 9                 F ← Cruce(F);
10             end
11             for i ← 1 To Tamaño(F) do
12                 if Aleatorio[0,1) < p_mutacion then
13                     f[i] ← Mutación(F[i]);
14                 end
15             end
16             P' ← AgregueSolución(F);
17         end
18         P ← Remplace(P');
19         S_θ ← ObtengaMejor(P,S_θ);
20     end
21     return S_θ ;
22 end
```

poder pasar parte de su material genético a la siguiente generación. La función de adaptación provee la presión que hace evolucionar la población hacia cromosomas más aptos, por ello una buena definición de esta función es crucial para el correcto desenvolvimiento del algoritmo.

Selección: El mecanismo de selección de los individuos que serán utilizados para reproducirse se realiza mediante un operador genético. El operador hace la elección basándose en los valores de adaptación de los individuos. Podemos encontrar distintos operadores de selección como lo proponen en [129, 211], entre estos podemos destacar los siguientes:

- Por Ranking.
- Basada en torneo.
- Por ruleta.
- Selección Proporcional.

Reproducción: En la fase de reproducción se utiliza el operador genético de cruce. Este operador es el encargado de realizar la transferencia del material genético de una generación a la siguiente. Normalmente se utilizan parejas, seleccionadas de la población, para aplicar este tipo de operador. Existen diversas variantes de este tipo de operador como se describe en [187, 295], entre los más comunes podemos mencionar:

- Cruce monopunto.

- Cruce multipunto.
- Cruce uniforme.

Mutación: El operador de mutación trabaja a nivel de bloque dentro de los cromosomas, haciendo cambios aleatorios de acuerdo a una probabilidad $P_{mutation}$ (probabilidad de mutación). La naturaleza del cambio depende de la composición de los bloques de los cromosomas. Si cada bloque es un bit (en la codificación binaria), el único cambio posible es invertir su valor. Si los bloques son números reales, la modificación podría dar lugar a las múltiples alternativas que serán utilizadas de acuerdo a la naturaleza del problema.

Reemplazo: Cada nuevo individuo generado producto de los operadores genéticos debe ser evaluado y si es posible reincorporado a la población actual. Este mecanismo permitirá avanzar hacia generaciones más prometedoras y lograr converger hacia las soluciones óptimas del problema. El reemplazo puede ser ejecutado mediante procesos como: uniformidad, elitismo, por torneo, etc.

Este tipo de algoritmos ha sido empleado en un variado número de problemas, entre los cuales podemos mencionar: Damage detection in structural elements [276], Substitution box (S-box), [326] y The minimizing potential energy function [303].

2.5.3.2 Búsqueda Dispersa (Scatter Search)

Este tipo de estrategia emplea un conjunto de referencia basada en la recopilación de buenas soluciones dispersas que permiten la conducción de la búsqueda, así como, el mantenimiento de un grado satisfactorio de diversidad. La propuesta inicial se originó en estrategias para crear reglas de decisión compuestas [123]. En Glover [122], se introduce la combinación ponderada (*weighted combination*) como el principal mecanismo utilizado para generar soluciones nuevas. En esta versión se enfatizan las búsquedas de corte lineal entre dos soluciones y el empleo de pesos para obtener muestras en dicha línea. Ademas, se utiliza el concepto de combinar soluciones de buena calidad con soluciones diversas. El método incluye una componente de intensificación que consiste en tomar una porción mayor de la línea que ha producido mejores soluciones.

Esta técnica ha sido utilizada con éxito en problemas como: The vehicle routing problem (VRP) [302], The distributed permutation flowshop problem [225] y Golomb Rulers [63], entre otros.

2.5.3.3 El Re-encadenamiento de Caminos (Path Relinking)

Es una metaheurística asociada a la búsqueda dispersa que utiliza la información que es obtenida de las soluciones de mayor calidad. La información obtenida es utilizada para incorporar mejoras en soluciones que se encuentran posteriormente. Básicamente se trata de aplicar el proceso de generación de soluciones explorando las 'trayectorias' que establecen conexiones entre soluciones de alta calidad. Partiendo de una solución prometedora se explora una ruta de soluciones hacia la otra solución incorporando atributos de la segunda en la primera de ellas. Esta ruta se construye tomando atributos de la segunda solución candidata que lo acerca más a ella. A continuación se toman, como puntos de partida para nuevas fases de mejora, soluciones del recorrido anterior [126].

Algunos trabajos muestran el éxito de este tipo de técnica al ser aplicados a: inferencia filogenética [58], Binary quadratic programming [255], The max-min diversity problem [325] y the resource levelling problem [259].

2.6 Técnicas Híbridas

Las técnicas metaheurísticas, en general, ofrecen ventajas prácticas para la resolución de problemas de optimización con alto grado de dificultad. Estas ventajas incluyen múltiples bondades como la simplicidad de su diseño, la robustez para adaptarse a los cambios, su flexibilidad, generalidad, entre otras. Son aplicables a un gran número de problemas en los cuales otras técnicas no alcanzan a resolverlos o fracasan en su intento. Gran parte de las técnicas metaheurísticas han reportado éxito en su aplicación como es el caso de los algoritmos genéticos [102, 130, 323], el Hill Climbing [33, 132, 311], la búsqueda tabú [78, 264, 338], la búsqueda por entornos variables [8, 140, 214], búsqueda local iterada [54, 183, 291], y muchos otras.

Muy a pesar de los éxitos que se han reportado en la literatura científica respecto a las técnicas metaheurísticas estas no logran satisfacer todas las expectativas que se presentan en la resolución de problemas de optimización, es por ello que han surgido nuevas estrategias que hacen uso de la combinación de diferentes algoritmos para diseñar e implementar técnicas más robustas. La integración de diferentes técnicas de aprendizaje y adaptación para superar las limitaciones individuales y activar efectos sinérgicos se ha podido lograr por medio de la "Hibridación" o fusión entre estas técnicas [204, 219].

A la luz de uno de los teoremas del *No Free Lunch* se ha determinado que no existe ningún método particular que pueda garantizar el mejor rendimiento en todos los problemas de cualquier naturaleza y por lo tanto, una metaheurística puede mostrar un alto rendimiento en algunos problemas y desempeñarse pobremente en otros escenarios. La incorporación de algún tipo de mejora sobre una determinada técnica metaheurística, mediante la combinación de algunos componentes tomados de otros métodos o la amalgama de componentes de dos o más métodos diferentes es llamado *metaheurística híbrida* [335]. El desarrollo de este tipo de hibridación se basa en el principio de que la sinergia puede generar algoritmos de alto rendimiento que exploten y combinen las ventajas de las estrategias puras que operan en forma individual [29, 256]. Una forma de integración de las técnicas híbridas se basa en un proceso de integración con las técnicas metaheurísticas de AI/OR (Artificial Intelligence/Operations Research) y programación con restricciones, Branch & Bound, o cualquier otra técnica basada en árboles y la programación lineal entera [28, 29]. Un amplio estudio de sistemas de hibridación con métodos exactos se puede ver en [165].

2.6.1 Clasificación de las Metaheurísticas Híbridas

Existen unas cuantas propuestas para la taxonomía de las Metaheurísticas Híbridas [88, 165, 298]. En resumen podemos mencionar que los aspectos más resaltantes a considerar en el diseño de una algoritmo híbrido son los siguientes:

- Híbrido de bajo nivel con Relevo o LRH. La hibridación de bajo nivel aborda la composición funcional de un único método de optimización. En esta clase de hibridación, una función dada de una metaheurística es reemplazada por otra metaheurística.
- Híbrido de alto nivel con Relevo o HRH. Esta clase representa la hibridación de metaheurísticas donde cada una de ellas se ejecuta independientemente (autonomía) y en secuencia.
- Híbrido de bajo nivel con Trabajo en equipo o LTH. Son algoritmos híbridos no auto-contenidos que se ejecutan en una secuencia particular. Poseen dos componentes esenciales: Exploración y Explotación. El ejemplo sería un algoritmo Memético.
- Híbrido de alto nivel con Trabajo en equipo o HTH. Representan varios algoritmos auto-contenidos

que realizan una búsqueda en paralelo y cooperan para encontrar un óptimo.

En nuestra investigación estamos particularmente interesados en un enfoque híbrido denominado *algoritmo Memético* que fue planteado por Moscato [217] y que tiene como característica principal la utilización de un esquema de integración que incorpora un mecanismo de mejora local (LS), como parte integral de un algoritmo genético [285].

En las siguientes secciones describiremos brevemente dos de las técnicas híbridas que utilizaremos para abordar nuestra investigación, como los son: (a) los algoritmos meméticos y (b) los modelos de cooperación.

2.6.2 Algoritmos Meméticos

Los Algoritmos Meméticos (Memetic Algorithms, MAs) son una familia de metaheurísticas que intentan reunir ideas y características de diferentes técnicas de resolución, como por ejemplo GAs y LSs (Local Search). El adjetivo "memético" viene del término inglés "meme", acuñado por R. Dawkins para designar al análogo del gen en el contexto de la evolución cultural. Es importante mencionar, sin embargo, que el uso de esta terminología no representa el propósito de adherirse a una metáfora de funcionamiento concreta (la evolución cultural en este caso), sino más bien lo contrario: hacer explícito que se difumina la inspiración puramente biológica, y se opta por modelos más genéricos en los que se aprende, manipula y se transmite información. En relación con esto último y refiriéndose a la forma en la que más comúnmente un MA puede implementarse[6], se encuentran diversos trabajos que hacen uso de nombres similares para referirse a éstos (e.g., EAs híbridos o lamarckianos), o que usando el término MA, hacen una interpretación muy limitada del mismo. Aun así, puede decirse que un MA es una técnica de búsqueda en la que una población de algoritmos (llamados agentes) optimizadores compiten y cooperan de forma sinérgica [219]. Más aún, estos agentes hacen uso explícito de las características y conocimiento del problema que se pretende resolver, tal como sugiere tanto la teoría como la práctica [172]. En términos generales, un MA puede ser visto como uno (o varios) procesos de búsqueda local actuando sobre un conjunto de soluciones candidatas que se acoplan en episodios periódicos de cooperación por medio de procesos de recombinación. El pseudocódigo para un algoritmo Memético básico se puede ver en el algoritmo 7. El proceso de inicialización de la población es el responsable de la creación del conjunto inicial de soluciones candidatas, en contraposición con los algoritmos genéticos, los MAs, suelen utilizar procedimientos sistemáticos para obtener posibles soluciones de alto grado de calidad como punto de partida, esto se puede logra por medio de la utilización de heurísticas constructivas [60, 105] o por medio de búsquedas locales. Para el caso del criterio de finalización, típicamente se chequea el número total de iteraciones, al llegar a un número máximo de iteraciones sin mejora, o haber alcanzado cierto número de re-inicializaciones. El procedimiento de cooperación y mejora constituyen el centro de un MA. El proceso de cooperación comprende una serie de etapas cada una de las cuales corresponde a la aplicación iterativa de un determinado operador particular que toma soluciones de la etapa anterior, generando nuevas soluciones. En cuanto al procedimiento de mejora, que se aplica por medio de un procedimiento de búsqueda local sobre las soluciones de la población para lograr obtener individuos de mayor calidad. Este proceso juega un papel importante en el diseño de un MA. El procedimiento de competición es utilizado para reconstruir la población actual por medio de un población anterior generando una nueva población. Existen dos posibilidades principales para la realización de esta reconstrucción: the plus strategy y the comma strategy [26]. Este último suele ser considerado como menos propenso al estancamiento.

[6]Esta forma más común es la que se denomina *interpretación restrictiva* de los algoritmos meméticos, y es la que entenderemos implícitamente cuando nos refiramos a estas técnicas en este trabajo.

Algoritmo 7: Pseudocódigo de MA Báscio

```
   input  : P, par
   output: Mejor Individuo Alcanzado.
 1 begin
 2     pop ← Inicializar (par,P);
 3     repeat
 4         nuevapop1 ← Cooperar(pop, par, P);
 5         nuevapop2 ← Mejorar(nuevapop1, par, P);
 6         pop ← Competir(pop, nuevapop2);
 7         if Corverge(pop) then
 8             pop ← Reiniciar(pop, par);
 9         end
10     until CondiciónTerminación(par);
11 end
12 return ElMejor(pop);
```

Algoritmo 8: Mejora local para Población Inicial

```
 1 Función Inicializar(In par : Parameter,In P : Problem)
 2     begin
 3         pop ← 0;
 4         for j ← 1 to par.TamañoPob do
 5             i ← SolucionAleatoria(P);
 6             i ← BLocal(i,par,P);
 7             pop ← pop∪{i};
 8         end
 9     end
10 return pop;
```

2.6.3 Modelos de Cooperación

Como ya lo hemos comentado en las secciones anteriores de este trabajo, las técnicas de búsqueda se han utilizado ampliamente para resolver muchos problemas de optimización que han resultado difíciles de abordar por diversas ciencias, como lo es la computación. Estas técnicas pueden incluir tanto algoritmos completos (llamadas técnicas exactas), así como también métodos aproximados (heurísticos). En los algoritmos de búsqueda completos, incluida la programación dinámica, nos permiten garantizar que vamos a encontrar una solución óptima para un problema de tamaño finito en tiempo limitado. Sin embargo, cuando el espacio de búsqueda del problema aumenta de tamaño, el tiempo que necesitan los algoritmos completos puede aumentar exponencialmente, lo que hace que este tipo de técnicas resulte impensable al momento de ser utilizada para su resolución. Por otra parte, los algoritmos de búsqueda aproximada pueden encontrar una buena solución (no óptima) en menos tiempo. Las técnicas híbridas han surgido para dar respuesta a este tipo de inconvenientes, un algoritmo híbrido es una combinación de algoritmos (completos y/o aproximados) utilizados para resolver el problema en cuestión, como es el caso de los algoritmos Meméticos o las hiperheurísticas, entre otras, sin embargo aunque se ha podido vencer muchas dificultades con esta estrategia, aún quedan vacíos que deben ser resueltos. Como medida para solventar esta situación, en la ultima década, se

han estudiado nuevos mecanismo de hibridación en los procesos de búsqueda llamados "búsqueda cooperativa" (también *modelos cooperativos*) a la luz de la computación paralela [209], en la cual, por medio del empleo de diferentes elementos (a los que se les suele llamar agentes), en donde cada uno de ellos opera explorando, de una forma singular, el espacio de búsqueda a través de mecanismos de intensificación/diversificación consubstancial a las metaheurísticas utilizada, este proceso permitirá abordar con cierto éxito los estancamientos que producen los óptimos locales mediante el intercambio de información del paisaje de búsqueda explorado. La búsqueda cooperativa es una categoría de algoritmos paralelos, en la que varios de los algoritmos de búsqueda se ejecutan en paralelo para resolver el problema de optimización, estos algoritmos de búsqueda (ejecutados en paralelo) pueden ser diferentes, es por ello que podemos decir qué una técnica de búsqueda cooperativa también puede verse como un algoritmo híbrido. Las técnicas cooperativas son una de las muchas áreas que han sido extensivamente objeto de estudio en la última década para abordar y dar solución a muchos problemas de optimización en los que los paisajes de búsqueda se hacen muy extensos. Si vemos esta estrategia con un enfoque básico esto implica tener más de un módulo de búsqueda ejecutándose e intercambiando información entre ellos para explorar el espacio de búsqueda de manera mucho más eficientemente y llegar a mejores soluciones. En muchas ocasiones se considera desde la perspectiva de algoritmos de enfoque paralelo ya que estos módulos pueden ejecutarse en paralelo. Por otro lado, algunos investigadores los clasifican como un enfoque híbrido ya que estos procesos de búsqueda podrían emplear diferentes heurísticas [67, 298].

Existen varias definiciones de lo que se considera *búsqueda cooperativa* algunas de ellas las podemos encontrar en [298, 308, 309]. Para Toulouse et. al [306], los algoritmos cooperativos son *estrategias que ejecutan en paralelo varios programas de búsqueda en la misma instancia de problema de optimización*. En general podemos decir que la búsqueda cooperativa consiste en una búsqueda realizada por agentes que intercambian información sobre estados, modelos, subproblemas completos, soluciones u otros características del espacio de búsqueda sobre el cual se está aplicando el proceso. Las principales ideas que podemos considerar para definir un modelo de cooperación podemos resumirlas como [196]:

- Sistemas multi-agentes, donde un conjunto de entidades simples actúan de forma mancomunada para mejorar progresivamente una población inicial.
- Algoritmos Multi-meméticos, que corresponden a un tipo específico de algoritmo memético que emplea un conjunto de estrategias de búsqueda local para mejorar una población de soluciones en conjunto con un algoritmo genético.
- Hiperheurísticas, donde un conjunto de heurísticas de bajo nivel son manejadas por una heurística de control, la cual tiene la tarea de determinar cuál debe aplicarse en cada momento. Esta decisión se realiza utilizando información sobre el desempeño de la heurística de bajo nivel.
- Metaheurística paralela Multi-hilo, donde se ejecutan varias heurísticas en paralelo aplicando procesos de colaboracion en los cuales realizan interacmbio de información/rendimiento para alcanzar regiones prometedoras del espacio de búsqueda, evitando los mínimos/máximos locales.

Existen dos características fundamentales relativas a los modelos colaborativos [68, 209, 307]:

- Un conjunto de algoritmos autónomos (los agentes), cada uno de los cuales implementa un método de soluciones particulares.
- Un esquema de colaboración que combina sus componentes autónomos, en una estrategia única y unificada para resolver el problema planteado.

2.6.3.1 Sistema Cooperativo

Podemos considerar, de acuerdo al alcance de este trabajo, que un *sistema cooperativo* que se emplea para la optimización, consiste en una serie de agentes (algoritmos) que trabajan en conjunto de manera colaborativa para alcanzar una serie de objetivos propuestos. Estos '*agentes*' pueden corresponder a personas, neuronas, computadoras, empresas, aviones o cualquier combinación de estas [156].

Definición 2.6.1 (Sistema cooperativo) *Un sistema cooperativo para optimización consiste en un conjunto de agentes:* $A = \{a_1, a_2, ..., a_n\}$*; en el cual cada agente i*

- *contiene un conjunto de opciones:* $D_i = \{o_1, o_2, ..., o_{m(i)}\}$,
- *una función objetivo:* $E_i : D_1 \times D_2 \times ... \times D_n \rightarrow R$,
- *y un esquema cooperativo para hacer el cambio:* $S_i : D_1 \times D_2 \times ... \times D_n \rightarrow D_i$

La función objetivo del sistema es $\sum_i E_i(x)$*, denotada como* $E(x)$*, donde* $x \in D_1 \times D_2 \times ... \times D_n$.

El cambio de agente i es denotado como $\tilde{x}_i \in D_i$*. Todos juntos forman la elección del sistema, denotado como* $\tilde{x}$*,* $\tilde{x} = \{ \tilde{x}_i, \tilde{x}_2, ..., \tilde{x}_n \}$

Dado D, será el producto cartesiano de D_i*s*

Obviamente, $\tilde{x} \in D$*. A veces, la función objetivo de un sistema también es llamada como el objetivo global del sistema y las funciones objetivos de los agentes se denominan los objetivos secundarios del sistema.*

Una solución óptima del sistema se denota como $(x_1^*, ..., x_n^*)$ *o simplemente* x^*.

Así podemos considerar que todos los agentes que operan en forma conjunta permiten tomar las mejores decisiones que hacen que una elección para el sistema sea mucho mas efectiva que si operaran en un esquema simple (sin cooperación). Este es un proceso iterativo, en el cual la cooperación más importante es la opción de descarte. La opción de descarte es un proceso donde cada agente descarta ciertas opciones de su conjunto de opciones que es poco probable que se elijan en una solución para el sistema. A medida que avanza la iteración, podemos esperar las mejores decisiones de descarte para cada agente. Después de algunos pasos de iteración, si solo queda una opción para cada agente, entonces se encuentra una solución para el sistema.

La literatura muestra un serie de propuestas que permiten establecer diversas combinaciones para la colaboración entre las metaheurísticas [29]. Dichos enfoques permiten a las diferentes técnicas de búsqueda combinar estrategias independientes, por lo que, cada uno de los elementos de la colaboración es un método independiente que, en teoría, sería capaz de dar solución al problema que se desea resolver; es por ello que las estrategias de cooperación deben ser diseñados de manera explícita. Muchos investigadores han propuesto un conjunto de estrategias de cooperación, como lo ha indicado Toulouse et al. [307], el cual ha considerado múltiples instancias de la búsqueda tabú ejecutándose en paralelo, en conjunto con el intercambio continuo de algunos de los datos almacenados en la memoria tabú (o lista tabú). En esta misma línea, Toulouse et al. [309] han propuesto un enfoque de cooperación jerárquico basado en la descomposición del problema en particular, por medio de la aplicación de este paradigma al problema de partición de grafos. También, Crainic & Gendreau [66] sugieren un modelo colaborativo de búsqueda tabú en paralelo, para abordar el problema del diseño de redes, el cual ha presentado una notable mejora con respecto a las estrategias de búsqueda independiente. Posteriormente, Crainic et al. [69] también presentaron un método para la búsqueda múltiple cooperativa usando Búsqueda de Vecindad Variable aplicado al problema de p-mediana. Recientemente Vasile y

Ricciardi [315], presentan un esquema para la búsqueda colaborativa MACS (Multi-Agent Collaborative Search), que está basado en mover una serie de agentes a diferentes regiones del dominio del espacio de búsqueda, y el uso de pesos para cada función objetivo. Por su parte, Fernández y Gutiérrez [93], propone dos esquemas genéricos para distribuir una serie de UcMAs (user-centric memetic algorithm), que actúan como agentes independientes y, finalmente, se sincronizan para el intercambio de información.

Algunos trabajos donde se ha aplicado búsqueda cooperativa con éxito son: [153], el cual aplicó un esquema de búsqueda cooperativa para resolver el problema de 'Redes de Robots', el problema de la mochila [241], el problema de diseño de redes [66], el problema del 'unmanned aerial vehicles (UAVs)' [333], el 'Tool Switching Problem' [10, 11], entre otros.

2.7 Contribuciones

En este capítulo fueron presentados los conceptos básicos relacionados con las técnicas metaheurísticas en general, enfocándonos en los elementos que las caracterizan, así como una descripción básica de los pseudocódigos más relevantes. En particular se abordan los métodos basados en trayectorias y basados en poblaciones que son las principales técnicas utilizadas en los siguientes capítulos del presente trabajo de investigación. Además, se han presentado aspectos importantes sobre los estados simétricos, los esquemas duales y las técnicas hibrídas. La idea es introducir al lector en este tema de tal forma que le facilite la lectura y comprensión de las ideas que deseamos expresar en el presente trabajo. Se trata de una contribución a nivel de esta memoria de tesis, más que a nivel del estado del arte

CAPÍTULO 3

PROBLEMAS Y METAHEURÍSTICAS BÁSICAS

En esta sección describiremos los problemas que se emplearán para la aplicación de las diferentes técnicas que se van a utilizar para el abordaje de este trabajo de investigación. Hemos considerado dos problemas para enfocarnos en la correspondiente experimentación, que nos permitirá realizar un extenso análisis de los resultados obtenidos en lo referente al comportamiento del gran número de técnicas y algoritmos que utilizaremos a lo largo de este trabajo, estos corresponden a:

- Diseño de Bloques Incompletos Equilibrados
- Diseño de Plantillas

Este capítulo se estructura de la siguiente forma: En la Sección 3.1 se presenta una visión general y se describen los conceptos básicos para el problema del diseño de bloques incompletos equilibrados (BIBD), las formulaciones para su representación, tanto a nivel natural como para su representación alternativa, los enfoques metaheurísticos empleados para las dos representaciones considerando la ruptura de simetrías. La Sección 3.2 da una visión general del tema de las plantillas y se describen los conceptos básicos para el problema del diseño de plantillas (TDP), las formulaciones para su representación, tanto a nivel natural como para su representación alternativa, los enfoques metaheurísticos empleados para las dos representaciones considerando la ruptura de simetrías. Luego en las secciones 3.3 y 3.4 se describen las características de las técnicas metaheurísticas utilizadas (trayectoriales y poblacionales) sobre las representaciones de los problemas abordados; se presentan los resultados obtenidos y un análisis estadístico de estos, tanto para el BIBD como para el TDP. Finalmente en la Sección 3.5 y 3.6, presentamos una revisión de los resultados obtenidos para ambos problemas y las contribuciones de este capítulo a la tesis.

3.1 Diseño de Bloques Incompletos (BIBD)

Hace ya más de 90 años, los trabajos del pionero Ronald Fisher [95] mostraron como los métodos estadísticos y en particular el diseño de experimentos podrían ayudar a solventar el problema de como descubrir y entender la complejas relaciones que pueden existir entre las diversas variables, aún

cuando los datos presentaran algún tipo de contaminación producido por el error experimental. Desde su uso, inicialmente en la agricultura, el diseño de bloques ha sido una de las áreas de gran interés en el mundo de la investigación científica que incluye agricultura, Biología, Ingeniería, Medicina, Física, etc. En muchas situaciones prácticas, la utilización del diseño de bloques completos no es del todo apropiada por varias razones, entre las cuales podemos mencionar el factor tiempo, los recursos disponibles (unidades experimentales, tratamientos disponibles, etc.), los factores a manipular en la experimentación, entre otros. Este hecho impulsó el desarrollo de diversos tipos de diseños basados en bloques incompletos, que se utilizan en una gran variedad de campos del diseño de experimentos. La teoría del diseño de experimentos [65], es la encargada de dilucidar respecto al uso de este tipo de estrategias. La idea principal es tratar de organizar los elementos de un conjunto finito en subconjuntos de manera que se mantengan ciertas propiedades importantes entre los tratamientos empleados y el diseño experimental utilizado. Muchos de los tipos de diseño que se emplean incluyen el *diseño de bloques incompletos equilibrados* (*Balanced Incomplete Block Designs* - BIBD), *t-design*, diseños equilibrados por parejas, cuadrados latinos ortogonales, entre otros. Muchas de las preguntas fundamentales sobre este tipo de técnicas están relacionadas con la existencia o no de algún tipo específico de diseño.

3.1.1 Visión General

Los diseños de bloques son en general estructuras de datos (i.e., vectores o bloques de datos) que se utilizan con la finalidad de establecer cierto control de forma sistemática sobre la variabilidad de datos causadas por factores externos. Cuando se presentan este tipo de factores, es posible generar diseños por bloques capaces de separar y eliminar esta variación del resto de los efectos de los factores de interés. El tipo de diseño que emplea bloques incompletos equilibrados es una de las técnicas utilizadas con frecuencia para tratar la variabilidad de los datos. Los diseños de bloques incompletos fueron introducidos por Yates [334], para experimentos en los que el número de unidades experimentales por bloque es menor que el número de tratamientos.

El diseño de bloques incompletos equilibrados (BIBD) compara todos los tratamientos con igual precisión y es un arreglo tal que todos los tratamientos tienen igual número de réplicas y cada par de tratamientos se presenta en el mismo bloque un número igual de veces en algún lugar del diseño. El equilibrio obtenido con el mismo número de ocurrencias de todos los pares de tratamientos en el mismo bloque tiene como resultado una precisión igual en todas las comparaciones entre los pares de medidas de tratamientos. Según John [164], el diseño de bloques incompletos tiene r de t tratamientos en b bloques de k unidades experimentales con $k < t$ y el número total de unidades experimentales de $N = rt = bk$. El número de bloques donde ocurre cada par de tratamientos es $\lambda = r(k-1)/(t-1)$, donde $\lambda < r < b$. El valor entero λ se deriva del hecho de que cada tratamiento está apareado con los otros $t-1$ tratamientos en algún lugar del diseño, λ veces. Existen $\lambda(t-1)$ pares para un tratamiento específico en el experimento, el mismo tratamiento aparece r bloques con $k-1$ de los otros tratamientos, y cada tratamiento aparece en $r(k-1)$ pares. Por lo tanto:

$$\lambda(t-1) = r(k-1) \text{ o } \lambda = r(k-1)/(t-1).$$

Se puede construir un diseño de bloques incompletos equilibrados mediante la asignación de las combinaciones adecuadas de tratamientos a cada uno de los $b = \binom{t}{k}$ bloques, pero con frecuencia es posible obtener el equilibrio con menos bloques. Existen tablas con diseños útiles para muchas situaciones prácticas en Cochran y Cox [50] y en Fisher y Yates [96].

3.1.2 Conceptos Básicos

Gran parte de las siguientes definiciones han sido extraídas del libro "Handbook of Combinatorial Designs" de C. Colbourn [51].

Los BID (Diseños de Bloques Incompletos) ofrecen una gran ventaja al dar soluciones que nos permite disminuir la varianza del error y proporciona comparaciones más precisas entre tratamientos de lo que es posible con un diseño de bloques incompletos. A este tipo de diseños se les llama *t-design* (denotado como t-(v, k, λ) design), y puede definirse, en forma simple, como una dupla $(X, \mathcal{B})$ donde X corresponde a un conjunto de *v-set* de puntos y $\mathcal{B}$ una colección de *k-subsets* de X (blocks) que cumple con la propiedad que los *t-subset* de X están contenidos en exactamente λ bloques. El parámetro λ representa el índice del tipo de diseño considerado.

Podemos definir un diseño de bloques en general como:

Definición 3.1.1 (Diseño de Bloques) *Un diseño es una dupla $(X, \mathcal{A})$ tal que se cumplen y se satisfacen las siguientes propiedades:*

1. *X es un conjunto de elementos llamados puntos.*
2. *$\mathcal{B}$ es una colección (i.e. bloques múltiples) de subconjuntos no vacíos de X, llamados Bloques.*

Definición 3.1.2 (2-design) *Los 2-(v, k, λ) design se denominan Diseño de Bloques Incompletos equilibrados.*

1. *Se denomina Diseño Simple, siempre que no existan dos bloques idénticos.*

Definición 3.1.3 (2-(v, k, λ) de bajo orden) *Un Diseño de Bloques Incompletos equilibrados (BIBD) es una dupla $(V, \mathcal{B})$ donde V corresponde a un conjunto de v-set y $\mathcal{B}$ es una colección de b de los k-subsets de V (bloques) en la cual, cada elemento de V está contenido en exactamente r bloques y cada 2-subset de V está contenido en exactamente λ bloques. Las variables v, b, r, k, y λ son los parámetros de un BIBD.*

Definición 3.1.4 (BIBDs isomorfos) *Dos BIBDs (V_1,B_1), (V_2,B_2) son isomorfos si existe una relación biyectiva $\alpha : V_1 \longrightarrow V_2$, tal que, $B_1\alpha = B_2$. En la resolución de BIBDs, un isomorfismo, se define de manera similar. Un automorfismo es un isomorfismo de un diseño consigo mismo. El conjunto de todos los automorfismos de un diseño forma un grupo.*

Si dos bloques en un diseño son idénticos, se dice que ellos representan *Bloques Repetidos*. Es por ello que podemos referirnos a $\mathcal{B}$ como un conjunto múltiple de bloques en lugar de un conjunto.

Definición 3.1.5 (BIBD) *Dados los parámetros $v,k,\lambda \in \mathbb{N}^+$, tal que $v > k \leq 2$. Un Diseño de Bloques Incompletos equilibrados (v,k,λ) ((v,k,λ)-BIBD), es un diseño (X,$\mathcal{A}$) , tal que se cumplen y se satisfacen las siguientes propiedades*

1. *$|X| = v$,*
2. *Cada bloque contiene exactamente k puntos,*
3. *Todos los pares de puntos distintos, están contenidos en exactamente λ bloques.*

El numeral 3 de la definición 3.1.5 se denomina la propiedad *balance*. Un BIBD, se le llama *Diseño de Bloque Incompleto*, porque $k < v$, y por lo tanto, todos sus bloques son incompletos. Un BIBD posiblemente puede contener bloques repetidos si $\lambda > 1$. Además debemos introducir dos propiedades básicas para un BIBD:

Proposición 3.1.6 *En un (v,k,λ)-BIBD, todos los puntos ocurren en exactamente*

$$r = \frac{\lambda(v-1)}{k-1}$$

bloques.

El valor r es a menudo llamado el *número de replicación* del BIBD.

Proposición 3.1.7 *Un (v,k,λ)-BIBD, contiene exactamente*

$$b = \frac{vr}{k} = \frac{\lambda(v^2-v)}{k^2-k}$$

bloques.

Si queremos registrar los valores de los cinco parámetros, denotaremos un BIBD como (v,b,k,r,λ)-BIBD

Proposición 3.1.8 *Si (v,k,λ)-BIBD, existe, entonces $\lambda(v-1) \equiv 0 \pmod{k-1}$ y $\lambda v(v-1) \equiv 0 \pmod{k(k-1)}$.*

La palabra *incidencia* es utilizada para describir la relación entre bloques y variedades en el diseño. Así, podemos decir que un bloque B tiene una relación de incidencia sobre un tratamiento V (o viceversa), para describir que V es un miembro de B. De esta forma, si un diseño tiene b bloques $(B_1,...,B_b)$ y v variedades $(V_1,...,V_v)$, se define la matriz de *incidencia* $v \times b$ (A) con (i,j) entradas (a_{ij}) como sigue:

$$a_{ij} = \begin{cases} 1 & \text{si } V_i \in B_j \\ 0 & \text{en caso contrario} \end{cases} \tag{3.1}$$

En la matriz de incidencia cada bloque se ubica sobre las columnas de la matriz y cada variedad corresponde a las filas.

La proposición 3.1.8 permite determinar las condiciones necesarias para las familias de BIBDs con valores fijos de k y λ. Una de las principales metas de la teoría combinatoria del diseño es determinar las condiciones necesarias y suficientes para la existencia de un (v,b,k,r,λ)-BIBD. Podemos considerar, en forma general, la definición de un BIBD como: un arreglo de v objetos distintos y b bloques de tal forma que cada bloque contiene exactamente k objetos distintos, y en donde cada objeto ocurre en exactamente r bloques diferentes, y cada par de objetos diferentes (a_i,a_j) aparecen juntos en exactamente λ bloques. Cada uno de los b bloques $B_1,...,B_b$ contienen k objetos, pero para los diferentes bloques B_j (con $j \in 1,...,b$), pueden contener los mismos objetos. Así, un diseño en bloques, no es simplemente una colección de subconjuntos de un conjunto, sino que además, es un arreglo de objetos y bloques que determinan una relación de incidencia que permite conocer cuales objetos pertenecen a cuales bloques. Nótese que:

- Cada conjunto contiene exactamente k símbolos distintos.
- Cada símbolo se encuentra exactamente en r conjuntos.
- Cada par de símbolos están juntos en exactamente λ conjuntos.

Los parámetros de un BIBD son, por lo tanto (v,b,r,k,λ), donde:

- b es el número de bloques.

Figura 3.1: Ejemplo de solución para BIBD (7,7,3,3,1).

$$\begin{array}{ccccccc} 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{array}$$

- v es el número de variedades en el diseño.
- r es el número de réplicas en cada variedad.
- k es el número de elementos distintos en cada bloque.
- λ es el número de conjuntos en donde cada par de símbolos ocurre; el cual deberá de satisfacer que $\lambda(v-1) = r(k-1)$.

Un diseño es simétrico si $v = b$ lo que genera como consecuencia que $r = k$. Este tipo de diseños simétricos, son generalmente utilizados en un orden máximo de $v = b = 7$, restricción que puede ser eliminada a través de la construcción de diseños de mayor número de variedades y por supuesto de mayor número de bloques. Un ejemplo para la solución de un BIBD se puede apreciar en la figura 3.1, tomado de [244]. En este caso, los parámetros correspondientes son: $(v = 7, b = 7, r = 3, k = 3, \lambda = 1)$

3.1.3 Modelado de un BIBD

Para la representación de los individuos al problema se empleará un modelo matricial. Este tipo de representación es una forma muy natural de expresar las soluciones candidatas del problema BIBD, según la ecuación 3.2 que corresponde a la matriz de incidencia. Avanzando en esta línea, podemos mencionar esquemas propuestos por diferentes autores [206, 245, 246, 321], que han atacado este problema utilizando la programación con restricciones. La idea fundamental es la utilización de un conjunto de variables de decisión, que pueden estar en el dominio de los valores enteros o binarios. El modelo más directo de este tipo para BIBD está dado por una matriz binaria (de incidencia) $A = \{a_{ij}\}_{v \times b}$ donde cada variable $a_{ij} \in \{1,0\}$. Según este esquema existen tres tipos de restricciones, a saber:

- Para cada fila existe una restricción b-aria (ha de haber r unos por fila).
- Para cada columna restricción v-aria (ha de haber k unos por columna).
- Para cada par de filas existe una restricción $2b$-aria (ha de haber λ unos comunes).

Esto quiere decir que cada solución candidata está expresada como una matriz binaria A, tal que:

- Cada fila representa el número de objetos considerados en cada experimento.
- Cada columna representa el número de bloques considerados en total.

Así la matriz resultante tendrá $v \times b$ elementos, sujetos a las restricciones anteriormente indicadas. Según estos esquemas es necesaria la definición de las ecuaciones de desigualdad y/o igualdad, que expresan las restricciones que se deberán cumplir para poder obtener soluciones candidatas que podrían resultar en un óptimo global. Así por ejemplo podemos describir, de forma muy general, el es-

quema propuesto por [206], que utiliza una transformación del problema de restricciones no-binarias a binarias. Basándose en la premisa que existen $(v(v-1)/2)$ $2b$-aria restricciones que aseguran que el producto escalar de cada par de filas es exactamente λ, consideramos que dos filas i y j de un BIBD deben tener exactamente λ unos en la misma columna. Esto se representa mediante λ variables x_{ijp}, con $1 \leq p \leq \lambda$; donde x_{ijp} contiene la pth columna común para la fila i y j. Por lo que hay, $v(v-1)/2$ pares de filas, al igual que $\lambda v(v-1)/2$ variables, todas comunes al dominio $\{1,...,b\}$. Desde dichas variables, un BIBD $v \times b$, representan una matriz binaria T que se describe como sigue:

$$T_{[i,c]} = \begin{cases} 1 & \text{si } \exists j,p \text{ tal que } x_{ijp} = c \text{ o } x_{jip} = c \\ 0 & \text{en caso contrario} \end{cases} \tag{3.2}$$

Las restricciones respectivas se expresan como sigue:

$$x_{ijp} \neq x_{ijp'};\ \sum_{c=1}^{b} T[i,c] = r;\ \sum_{i=1}^{v} T[i,c] = k$$

Donde: $1 \leq p,\ p' \leq \lambda,\ 1 \leq i,\ j \leq v,\ 1 \leq c \leq b$. Según los autores, este tipo de formulación debería reducir notablemente el esfuerzo de poda.

Por su parte, Prestwich [246], propone, de forma similar a Meseguer y Torras, un modelo binario del problema utilizando restricciones, pero tratando de mantener la naturaleza del problema al no hacer la transformación del esquema no-binario. Para ello define variables que denotaban la posición de *unos* y *ceros* en cada fila y columna, y empleando cinco conjuntos de variables:

$$\begin{aligned} R1_{i,j} &\quad (1 \leq i \leq v,\ 1 \leq j \leq r) \\ R0_{i,j} &\quad (1 \leq i \leq v,\ 1 \leq j \leq b-r) \\ C1_{i,j} &\quad (1 \leq i \leq b,\ 1 \leq j \leq k) \\ C0_{i,j} &\quad (1 \leq i < b,\ 1 \leq j \leq v-k) \\ S1_{i,j,p} &\quad (1 \leq i \leq j \leq v,\ 1 \leq p \leq \lambda) \end{aligned}$$

Donde, $R1_{i,j}$ denota la posición en la fila de los r unos, y $R0_{i,j}$ los de $b-r$ ceros. De forma similar $C\{0,1\}_{i,j}$ denota la posición de los k unos y los $v-k$ ceros en la columna i. $S1_{i,j,p}$ denota las posiciones comunes de los λ unos en la fila i,j. $R\{0,1\}_{i,j}$ y $S1_{i,j,p}$ deben estar en el dominio $\{1...b\}$, además, $C\{0,1\}_{i,j}$ tienen dominio $\{1...v\}$.

De estos dos esquemas descritos anteriormente, podemos deducir un esquema que muestra gran similitud puesto que se mantiene la idea básica de la matriz de incidencia A en su representación binaria $a_{ij} \in \{0,1\}$ y las restricciones de los k unos por fila de tal forma que se cumpla $\sum_{i=1}^{v}(M_{ij}) = k$, además de conservar la cantidad de k unos por columna tal que $\sum_{j=1}^{b}(M_{ij}) = r$. De igual manera debe cumplirse la restricción respectiva para el producto entre cada par de filas de la matriz de incidencia tal que $\sum_{k=1}^{b} M_{ik}M_{jk} = \lambda$ con $i,j \in \{1,...,v\}$ [268, 270]. El BIBD es un problema en donde, comúnmente, se utiliza la satisfacción de restricciones para su resolución, pero que puede convertirse en un problema de optimización mediante la introducción de una función objetivo que medirá el número de restricciones violadas por una determinada solución candidata. Así, la función de evaluación nos dará el resultado de sumar todas las discrepancias con respecto al valor deseado en filas, columnas y en el producto escalar por cada par de filas. Sea $I = \langle v,b,r,k,\lambda \rangle$ la instancia del problema BIBD que se desea resolver. Entonces, la función objetivo viene dada por:

$$f^I(M) = \sum_{i=1}^{v} \phi_{ir}(M) + \sum_{j=1}^{b} \phi'_{jk}(M) + \sum_{i=1}^{v-1} \sum_{j=i+1}^{v} \phi''_{ij\lambda}(M)$$

Donde

$$\phi_{ir}(M) = \left| r - \sum_{j=1}^{b} m_{ij} \right|$$

$$\phi'_{jk}(M) = \left| k - \sum_{i=1}^{v} m_{ij} \right|$$

$$\phi''_{ij\lambda}(M) = \left| \lambda - \sum_{k=1}^{b} m_{ik} m_{jk} \right|$$

Obviamente, el objetivo es minimizar el número de restricciones violadas. El óptimo global de la función es por lo tanto $f^I(M^*) = 0$, donde M^* representaría una solución factible.

3.1.4 El Vecindario para el BIBD

En este aspecto podemos considerar dos posibles funciones de vecindad a saber:

- **Bit-Flit**: Que emplea una distancia de Hamming igual a 1 (es decir, un vecindario basado en una sola variación), en la que una solución vecina es considerada como el resultado de invertir exactamente una posición en la matriz de incidencia binaria. Sea C una solución candidata, es decir, una matriz binaria (A) de orden $v \times b$ (véase 3.2), y sea p_{ij} y c_{ij} (para $1 \leq i \leq v$ y $1 \leq j \leq b$), respectivamente, la celda y el valor de la celda en la fila i de la columna j en la matriz C. Entonces el conjunto de movimientos posibles es de $M(C) = \{(p_{ij}, \overline{c_{ij}} \mid 1 \leq i \leq v, 1 \leq j \leq b\}$, tal que $\overline{0} = 1$ y $\overline{1} = 0$. Así, la vecindad queda definida por:

$$\mathcal{N}_{bit-flip}(A) = \{A' \mid H(A,A') = 1\}$$

- **Swap**: Este tipo de vecindad es altamente utilizada en los problemas cuya principal operación para la generación de los vecinos son las permutaciones. Se denota por $\mathcal{N}_{swap}(.)$, y representa la operación en la que se utiliza el intercambio. Esto será posible, siempre que la permutación sean diferente en dos posiciones de la secuencia, es decir, para un vecino $\pi \in \mathbb{P}_n$.

$$\mathcal{N}_{swap}(\pi) = \{\pi' \in \mathbb{P}_n | H(\pi,\pi') = 2\}$$

  Donde $H(\pi,\pi') = n - \sum_{i=1}^{n} [\pi_i = \pi'_i]$ es la distancia de Hamming entre las secuencias π y π', y $[.]$ es el corchete de Iverson (es decir, $[P] = 1$, si P es cierta, y $[P] = 0$, en caso contrario).

3.1.5 Ejemplo Ilustrativo

Para ilustrar la definición formal del problema dado en las secciones previas, vamos a describir un pequeño ejemplo. Consideremos la instancia $I = \langle v, b, r, k, \lambda \rangle$, tal que, $I = \langle 8, 14, 7, 4, 3 \rangle$, Considerando los parámetros de esta instancia, deben existir para cada fila $r = 7$ unos, $k = 4$ unos por cada columna y cada 1 debe aparecer en exactamente (λ) bloques distintos. En la figura 3.2 se puede apreciar una posible solución candidata. Nótese que para esta posible solución, las discrepancias relativas a las filas y columnas corresponden a cero, debido a que se cumple con las restricciones relativas a filas y columnas, no obstante el *fitness* correspondiente para este individuo es de 4, debido a que no se han cumplido completamente las restricciones relativas a λ, puesto que hay 4 discrepancias respecto

Figura 3.2: BIBD. Posible solución candidata para la instancia $I = \langle 8, 14, 7, 4, 3 \rangle$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
2	1	1	1	0	0	0	0	1	1	1	1	0	0	0
3	1	0	0	0	0	1	1	1	1	0	0	0	1	1
4	1	0	0	0	1	0	1	0	0	1	1	1	0	1
5	0	0	1	1	1	0	0	0	1	0	1	0	1	1
6	0	0	1	0	1	1	0	1	0	1	0	1	1	0
7	0	1	0	1	0	1	0	0	0	1	0	1	1	1
8	0	1	0	1	0	0	1	1	1	0	1	1	0	0

Figura 3.3: Problema del BIBD: Cálculo del fitness para la instancia $I = \langle 8, 14, 7, 4, 3 \rangle$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
2	1	1	1	0	0	0	0	1	1	1	1	0	0	0
3	1	0	0	0	0	1	1	1	1	0	0	0	1	1
4	1	0	0	0	1	0	1	0	0	1	1	1	0	1
5	0	0	1	1	1	0	0	0	1	0	1	0	1	1
6	0	0	1	0	1	1	0	1	0	1	0	1	1	0
7	0	1	0	1	0	1	0	0	0	1	0	1	1	1
8	0	1	0	1	0	0	1	1	1	0	1	1	0	0

Discrepancias por filas : 0
Discrepancias para las Columnas: 0
Discrepancias para el producto escalar:

Filas: 2x7 = 2
Filas: 2x8 = 4
Filas: 6x7 = 4
Filas: 6x8 = 2

Total: 4 Discrepancias (Fitness)

a esta restricción, esto lo podemos ver entre las filas 2 y 7, donde el producto escalar de estas dos filas se computa en 2, debiendo ser 3, esto genera una discrepancia. En este ejemplo existen 3 productos más, que no cumplen las restricciones respectivas, lo que hace que el fitness respectivo sea el valor ya indicando anteriormente. En la figura 3.3, podemos ver un resumen del cálculo del fitness de la solución candidata utilizada en el ejemplo (figura 3.2).

3.1.6 Trabajos Relacionados para BIBD

La generación de diseños por bloques es un conocido problema de optimización combinatoria, el cual lo podemos enmarcado dentro de la categoría de difícil resolución (en relación a la falta de algoritmos polinómicos conocidos, ya que desde un punto de vista de decisión, dados (v, b, r, k, l) se puede saber, en muchos casos, si existe solución [57]), que ha surgido en la rama del diseño de experimentos considerado en complejidad de alta gama [51]. Se utilizó originalmente en el diseño estadístico de experimentos [50, 94, 97, 201, 314] pero la ciencia le ha encontrado un gran número de aplicaciones, como por ejemplo, la criptografía [39], el diseño de circuitos integrados [258], y la teoría de la codificación [177] entre otros. La generación de diseño de bloques incompletos proporciona una excelente referencia para abordar nuestro estudio, ya que es escalable y tiene una amplia variedad de instancias, que van desde casos sencillos a otros muy complejos de resolver. La escalabilidad del problema así

Tabla 3.1: Número (#) y porcentaje (%) de instancias del problema BIBD resueltas por las metaheurísticas básicas e integrativas (identificadas en la primera columna) trabajando en forma separada sobre el conjunto de las 86 instancias tomadas de [32, 246]. La tercera columna indica una referencia en la cual el método ha sido reportado.

algorithm	# (%)	Ref.
NN-SA	16 (18.60%)	[32]
CLS	56 (65.12%)	[246, 247]
TS_{sw}	57 (66.28%)	[268]
GA_{sw}	37 (43.02%)	[268]
TS_{sw}	59 (68.60%)	[270]
GA_{Gd}	48 (55.81%)	[270]
MA_{Gd}	**63** (73.26%)	[270]
TABU-BIBD(20)	78 (90.70%)	[336]
Multi-step	**79** (91.86%)	[188]
Ts+Hc	74 (86.05%)	[272]

como su dificultad ha permitido realizar un gran número de pruebas sobre diversas técnicas computacionales para medir su rendimiento, en este sentido, se han utilizado métodos exactos (que incluyen técnicas exhaustivas) [115], resultando poco convenientes incluso para los diseños de tamaño relativamente pequeño. Otra técnica empleada ha sido la programación con restricciones [206, 245], en cuanto a este enfoque vale la pena mencionar que uno de los puntos clave de interés del problema es su naturaleza simétrica. Otras propuestas han utilizado métodos híbridos que combinan técnicas heurísticas con redes de neuronas, como lo ha presentado Bofill *et al.* [31]. En los últimos años las técnicas metaheurísticas han sido empleadas sobre este problema, como lo ha mostrado Prestwich [246, 247]. En forma general, podemos resumir el rendimiento de las diferentes técnicas metaheurísticas (medido en número de casos de instancias resueltas) que se han mencionado a lo largo de esta investigación (véase la tabla 3.1). No proporcionamos tiempos de ejecución ya que esta información no ha sido suministrada para todos los métodos y además, depende en gran medida de muchos factores como CPU, memoria RAM (Random Access Memory) y otro tipo de recursos.

3.1.7 Enfoques Metaheurísticos para el BIBD

Los enfoques metaheurísticos que deseamos utilizar abarcan tanto técnicas de búsqueda local, como algoritmos genéticos. Antes de describir dichas técnicas en mayor detalle es conveniente detenerse en la definición de las vecindades básicas consideradas, ya que éstas tendrán luego traslación a las diferentes técnicas empleadas.

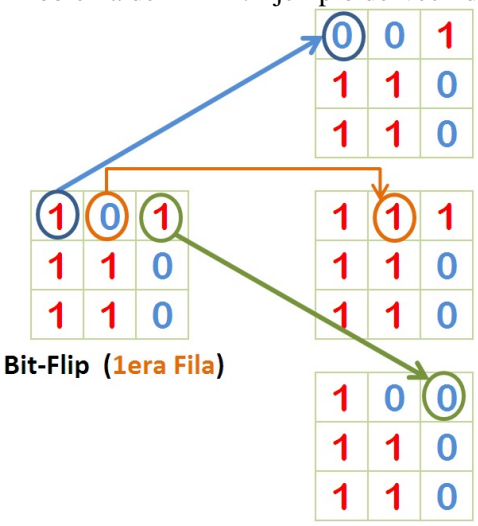
3.1.7.1 Vecindades Consideradas

La primera vecindad considerada se basa en la noción de distancia de Hamming, y se denotará por bit-flip. Sea $H(M_1, M_2)$ la distancia de Hamming entre dos matrices binarias M_1 y M_2. Entonces, la vecindad Bit-Flip queda definida por:

$$\mathcal{N}_{bit-flip}(M) = \{M' \mid H(M, M') = 1\}$$

Como ejemplo, si la instancia considerada fuera $I = \langle 3,3,2,2,1 \rangle$, la figura 3.4 mostraría una parte de los vecinos de un posible candidato a solución (sólo los resultantes de modificar un bit de la primera fila).

Figura 3.4: Problema del BIBD: Ejemplo de Vecindad Bit-Flip



Claramente, el tamaño completo de esta vecindad es

$$|\mathcal{N}_{bit-flip}(M)| = vb$$

y la evaluación de cada uno de los vecinos contenidas en la misma requiere el recómputo incremental de $v+1$ restricciones (1 fila + 1 columna + $v-1$ productos escalares). Si tenemos en cuenta que la evaluación completa de una solución requiere el cómputo de $v+b+v(v-1)/2$ restricciones, se tiene que la exploración completa del vecindario es equivalente a

$$n_{equiv} = \frac{vb(v+1)}{v+b+v(v-1)/2}$$

evaluaciones completas.

Por otra parte, se ha considerado una segunda vecindad, a la que denominamos swap. En esta vecindad se mantiene fijo el número de bits a uno en cada fila, ya que la operación básica es el intercambio de posiciones dentro de las mismas. Así, podemos definir esta vecindad como

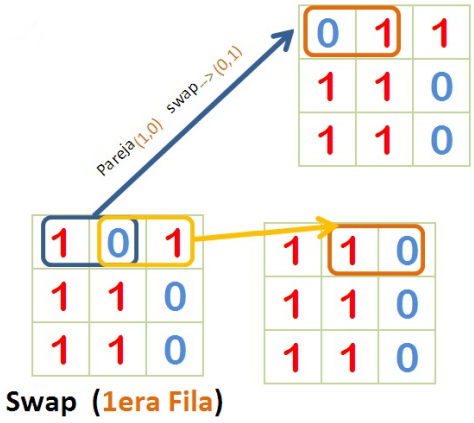
$$\mathcal{N}_{swap}(M) = \{M' \mid \exists! i,j,k : \; m_{ij} = m'_{ik} = 0, m'_{ij} = m_{ik} = 1\}$$

Si continuamos con el ejemplo anterior una parte de la vecindad swap sería la mostrada en la figura 3.5. Nótese que puesto que sólo tiene sentido intercambiar bits con diferentes valores, el tamaño de la vecindad es

$$|\mathcal{N}_{swap}(M)| = vr(b-r)$$

de donde puede deducirse el número equivalente de evaluaciones completas correspondiente a la exploración completa de la vecindad. Nótese que en principio no sería necesario reevaluar las restricciones correspondientes a las filas, ya que el número de unos por fila no varía. Sin embargo, el impacto de este factor es muy pequeño, ya que el esfuerzo de cómputo está dominado por el término cuadrático del denominador.

Figura 3.5: Problema del BIBD: Ejemplo de Vecindad Swap



3.1.7.2 Técnicas de Búsqueda Local

Vamos a considerar dos versiones para la técnica hill climbing (HC), basadas en cada una de las dos vecindades definidas. La única diferencia entre ambas, aparte del empleo de una vecindad diferente, es la inicialización:

- si se emplea la vecindad bit-flip, la inicialización es completamente aleatoria.
- si se emplea la vecindad swap, la inicialización es aleatoria, pero sujeta a la restricción de que cada fila tenga exactamente r unos.

Por lo demás, el procedimiento es el estándar: se explora completamente la vecindad, y se elige la mejor solución (*steepest ascent*), salvo que ésta sea peor que la actual, en cuyo caso se ha llegado a un óptimo local, y se reinicia la búsqueda.

Adicionalmente se ha considerado un algoritmo de búsqueda tabú (TS), también definido sobre la base de las dos vecindades anteriores. Como en el caso de HC, la inicialización se realiza atendiendo al tipo de vecindad que se empleará, y la exploración de la vecindad es completa. Los demás elementos del algoritmo son los que siguen:

- En el caso de emplear la vecindad bit-flip, se considerará tabú la modificación de una determinada celda de la matriz. En el caso de la vecindad swap, los movimientos tabú son intercambios entre posiciones de la misma fila.
- La persistencia tabú de los movimientos aceptados se elige aleatoriamente en cada paso dentro del rango $[\beta/2, 3\beta/2]$, donde $\beta = vb$ en el caso de bit-flip, y $\beta = vbr$ en el caso de swap.
- El criterio de aspiración es mejorar la mejor solución encontrada hasta el momento.
- Tras un número n_{intens} de evaluaciones sin mejora se intensifica la búsqueda, volviendo a la mejor solución conocida.

3.1.7.3 Algoritmo Genético

Al igual que con las técnicas de búsqueda local, en el caso del empleo de algoritmos genéticos se han considerado dos variantes, correspondientes a cada una de las vecindades. En el caso del empleo de vecindad bit-flip, el algoritmo genético es una versión estándar, en la que la inicialización de la población se realiza de manera aleatoria, y las operaciones de cruce y mutación pueden realizarse de

manera ciega. Concretamente, se ha considerado el operador de cruce uniforme (UX) y la mutación por inversión de bits.

En el caso del empleo de la vecindad swap, el GA ha de adaptarse para trabajar en todo momento manteniendo la restricción del número de unos por fila. Para ello:

- Cada individuo se genera inicialmente de forma aleatoria, aplicando la restricción de que para cada fila de la matriz de incidencia sólo pueden aparecer r unos.
- El cruce se realiza mediante una variante de UX a nivel de filas: se realizan torneos aleatorios para determinar si la fila i-ésima ha de copiarse completamente de un padre o del otro.
- La mutación se realiza mediante intercambio de posiciones, sujeta a la restricción de que dichas posiciones estén en la misma fila, y que los valores de las mismas sean diferentes (de lo contrario la mutación dejaría al individuo inalterado).

Por lo demás, se ha considerado un modelo de evolución de estado estacionario con reemplazo del peor, y se ha realizado la selección por torneo binario.

3.1.7.4 Función Objetivo

El BIBD es un problema de satisfacción de restricciones que convertiremos en un problema de optimización mediante la introducción de una función objetivo que medirá el número de restricciones violadas por una determinada solución cadidata. Así, la función de evaluación nos dará un valor numérico que resulta de sumar todas las discrepancias con respecto al valor deseado en filas, columnas y en el producto escalar por cada par de filas. Sea $I = \langle v, b, r, k, \lambda \rangle$ la instancia del problema BIBD que se desea resolver. Entonces, la función objetivo viene dada por:

$$f^I(M) = \sum_{i=1}^{v} \phi_{ir}(M) + \sum_{j=1}^{b} \phi'_{jk}(M) + \sum_{i=1}^{v-1} \sum_{j=i+1}^{v} \phi''_{ij\lambda}(M)$$

Donde

$$\phi_{ir}(M) = \left| r - \sum_{j=1}^{b} m_{ij} \right|$$

$$\phi'_{jk}(M) = \left| k - \sum_{i=1}^{v} m_{ij} \right|$$

$$\phi''_{ij\lambda}(M) = \left| \lambda - \sum_{k=1}^{b} m_{ik} m_{jk} \right|$$

Obviamente, el objetivo es minimizar el número de restricciones violadas. El óptimo global de la función es por lo tanto $f^I(M^*) = 0$, donde M^* representaría una solución factible.

3.1.8 Representación Dual: BIBD

El problema de BIBD puede expresare claramente como un problema de satisfacción con restricciones, pero se puede transformar en un problema de programación no lineal como se indica a continuación:

$$P\text{:} \quad \max \sum_{i=1}^{v}(\sum_{l=1}^{b} x_{il}) + \sum_{l=1}^{b}(\sum_{i=1}^{v} x_{il}) + \sum_{i=1}^{v-1}\sum_{j=i+1}^{v}(\sum_{l=1}^{b} x_{il}x_{jl}) \tag{3.3}$$

Sujeto a:

$$\sum_{l=1}^{b} x_{il} \leq r, \text{ i=1,...,v} \tag{3.4}$$

$$\sum_{i=1}^{v} x_{il} \leq k, \text{ i=1,...,b} \tag{3.5}$$

$$\sum_{l=1}^{b} x_{il}x_{jl} \leq \lambda, \text{ i=1,...,v, i<j} \tag{3.6}$$

$$x_{il} \in \{0,1\}, \text{ i=1,...,v, l=1,...,b} \tag{3.7}$$

Notese, que para una solución factible arbitraria $X = x_{ij} \in P$, se debe cumplir que:

$$z(X) \leq vr + bk + \frac{v(v-1)}{2}\lambda \tag{3.8}$$

Si se satisfacen las inecuaciones (3.4) − (3.6), además de (3.8), podemos decir que X contiene una solución para BIBD$\langle v,b,r,k,\lambda\rangle$. Este tipo de representación podríamos llamarla como **PRIMAL**.

Ahora bien, una posible representación alternativa (a la que llamaremos como **DUAL**) para el BIBD, corresponde a la minimización (contraria a la representación primal), de tal forma que podemos expresar este como sigue:

$$P\text{:} \quad \min\ f^I(M) = \sum_{i=1}^{v}\phi_{ir}(M) + \sum_{j=1}^{b}\phi'_{jk}(M) + \sum_{i=1}^{v-1}\sum_{j=i+1}^{v}\phi''_{ij\lambda}(M) \tag{3.9}$$

Sujeto a:

$$\sum_{j=1}^{b}(M) = r \tag{3.10}$$

$$\sum_{i=1}^{v}(M) \geq k \tag{3.11}$$

$$\sum_{k=1}^{b} m_{ik}m_{jk} \geq \lambda \text{ con i,j} \in \{1,...,v\} \tag{3.12}$$

$$\forall M \in \{0,1\}$$

Donde:

$$\phi_{ir}(M) = \left| r - \sum_{j=1}^{b} m_{ij} \right|$$

$$\phi'_{jk}(M) = \left| k - \sum_{i=1}^{v} m_{ij} \right|$$

$$\phi''_{ij\lambda}(M) = \left| \lambda - \sum_{k=1}^{b} m_{ik}m_{jk} \right|$$

El óptimo global de la función es por lo tanto $f^I(M^*) = 0$, donde M^* representaría una solución factible. Por otra parte, si consideramos el vecindario swap, el tamaño de la vecindad estaría dado por:

$$|\mathcal{N}_{swap}(M*)| = bk(v-k)$$

Nótese que sólo tiene sentido intercambiar bits con diferentes valores en la misma columna.

Sin embargo debemos aclarar que este modelo de representación nos limitaría la exploración de zonas prometedoras debido a que, en la mayoría de los casos (si no es que en todos), el valor de $k <= r$. Consideremos un ejemplo: para $I = \langle 8,14,7,4,3\rangle$, el espacio de búsqueda para este caso sería

del orden de los 224 vecinos, muy por debajo del modelo primal, en el que exploraríamos alrededor de 392 vecinos. Debido a esto, hemos divisado una representación alternativa basada en una matriz de incidencia **DECIMAL**, como se explica a continuación.

A este modelo lo vamos a identificar como **Modelo Decimal** (**Dec**). En este caso la matriz de incidencia resultante será una matriz decimal $M_{v\times r}$, donde cada celda de la matriz M_{ij} contendrá el número de la columna donde se debe aplicar un tratamiento t. Por ejemplo, consideremos la instancia BIBD-$\langle 7,7,3,3,1\rangle$, cuya representación primal por medio de la matriz de incidencia binaria, como se muestra en la figura 3.6. Para este modelo la matriz de incidencia sería $M_{7\times 7}$ osea $M_{v\times b}$. Este mismo

Figura 3.6: BIBD-$\langle 7,7,3,3,1\rangle$.

0	1	0	1	0	1	0
1	0	0	1	0	0	1
1	1	1	0	0	0	0
0	0	1	0	0	1	1
0	1	0	0	1	0	1
1	0	0	0	1	1	0
0	0	1	1	1	0	0

ejemplo estaría representado por una matriz $M_{7\times 3}$ ($M_{v\times r}$) en el modelo Dual, como se puede ver en la figura 3.7.

Figura 3.7: Representación PRIMAL (izquierda, matriz de incidencia Binaria) y DUAL (derecha, matriz de incidencia Decimal) de la instancia BIBD-$\langle 7,7,3,3,1\rangle$.

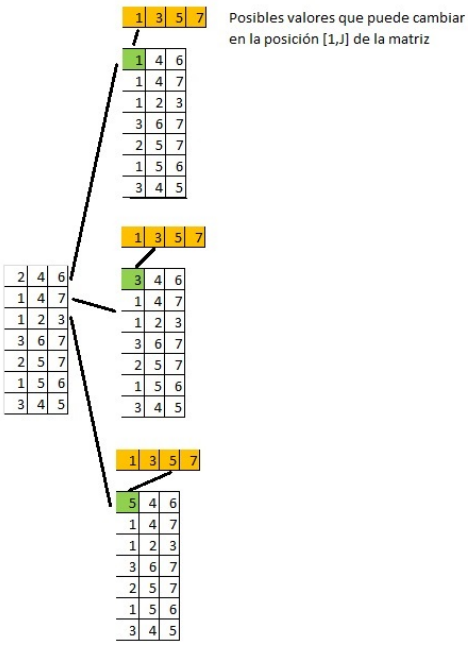
	1	2	3	4	5	6	7
1	0	1	0	1	0	1	0
2	1	0	0	1	0	0	1
3	1	1	1	0	0	0	0
4	0	0	1	0	0	1	1
5	0	1	0	0	1	0	1
6	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0

	1	2	3
1	2	4	6
2	1	4	7
3	1	2	3
4	3	6	7
5	2	5	7
6	1	5	6
7	3	4	5

3.1.8.1 Posible Vecindario para el Modelo Dual del BIBD

Un posible vecindario estaría dado por los valores $b' \in \{1..b\}$, como se muestra en la figura 3.8. Nótese que la restricción de los r unos por fila queda implícita. La formulación para esta representación, corresponde a una relajación del problema usado en programación con restricciones en la cual se define la función objetivo: como la minimización de la violación de restricciones (similar al modelo

Figura 3.8: Problema del BIBD: Posible vecindario para el Modelo Dual (BIBD-$\langle 7,7,3,3,1\rangle$).



primal), pero considerando sólo los parámetros k y λ, como se indica a continuación:

$$\min\ f_d^I(M^d) = \sum_{j=1}^{b} \psi_j(M^d,k) + \sum_{i=1}^{v-1}\sum_{j=i+1}^{v} \psi'_{ij}(M^d,\lambda) \tag{3.13}$$

tal que

$$\forall j,h \in \mathbb{N}_r^+ : j \neq h \Rightarrow m_{ij}^d \neq m_{ih}^d, \qquad \forall i \in [1,b] \tag{3.14}$$

donde

$$\psi_j(M^d,k) = \left| k - \sum_{i=1}^{v}\sum_{h=1}^{r} [m_{ih}^d = j] \right|, \qquad \forall j \in [1,b] \tag{3.15}$$

$$\psi'_{ij}(M^d,\lambda) = \left| \lambda - \sum_{h=1}^{r}\sum_{l=1}^{r} [m_{ih}^d = m_{jl}^d] \right|, \qquad \forall i,j \in [1,v] : i < j \tag{3.16}$$

En la ecuación (3.15) y (3.16) hemos empleado *los paréntesis de Iverson* [] (i.e., [P]=1 si P es verdadero, y 0 en caso contrario). Nótese que, agregando una restricción de valor completamente diferente asociada con cada fila – i.e., restricción (3.14)– cada fila i ahora contiene las r (requeridas) asignaciones de bloque de objeto i, y por lo tanto la restricción que requiere que cada objeto se coloque en r bloques, se mantiene implícitamente en el nuevo modelo dual. Por lo tanto, la función a ser minimizada – i.e., (3.13) – solo requiere de la atención de dos componentes. El primero, que se muestra en la ecuación (3.15), el cual calcula las discrepancias con respecto al valor k. Nótese que $\sum_{i=1}^{v}\sum_{h=1}^{r}[m_{ih}^d = j]$ calcula cuántas veces el j-ésimo bloque aparece en el BIBD, y por lo tanto la ecuación (3.15) equivale a calcular las discrepancias con respecto al valor requerido de k (i.e., cada

bloque $j \in [1,b]$ tiene que contener exactamente k objetos, algo que sucede cuando el primer componente $\psi_j(M^d,k)$ es igual a 0). El segundo componente se muestra en la ecuación (3.16), y calcula las discrepancias con respecto al valor λ. Observe que cada objeto i tiene que coincidir con cualquier otro objeto j en exactamente λ bloques, lo que significa que cada dos filas se tienen que compartir λ bloques (como en el modelo primario). Por lo tanto, medimos las discrepancias entre dos objetos (ecuación 3.16) con respecto al valor requerido λ (es decir, dos objetos distintos i y j tienen que ser parte de exactamente λ bloques, algo que sucede cuando el segundo componente $\psi'_{ij}(M^d,\lambda)$ es igual a 0).

Una solución al problema BIBD es, por lo tanto, una configuración M^{d*} tal que $f_d^I(M^{d*}) = 0$. En este modelo, la vecindad $\mathcal{N}_{sw}(M^d)$ es similar a la versión *swap* considerada para el modelo primal (veáse la sección 3.1.3), es decir, un vecino de una matriz dada M^d, es otra matriz de incidencia M'^d que es obtenida de este último, al reemplazar un elemento $\phi \in \mathbb{N}_b^+$ contenido en una celda m_{ij}^d por cualquier otra etiqueta en $\mathbb{N}_b^+ \setminus \{\phi\}$, siempre atentos a la restricción 3.14

3.1.8.2 Propuesta de Ruptura de Simetrías para BIBD

Nuestra propuesta para la ruptura de las simetrías sobre el problema BIBD, se basa en la utilización de *Value symmetry breaking* descrita por Walsh [322], en la cual se asigna un rango del dominio de valores a una ó varias variables para frenar algún tipo de simetría. Además, inspirados en la propuesta de Puget [254], utilizaremos la fijación de valores en las primeras filas y columnas en las soluciones candidatas del BIBD, imponiendo las restricciones respectivas, manteniendo a dicha solución como factible a lo largo del proceso de búsqueda. Veamos un ejemplo: Considerando el problema de los cuadros latinos [77], específicamente del orden 3×3. Asumiendo, además, supondremos que el dominio es un subconjunto de I^k de algún k, en el cual el orden de las variables se enumeran como: $v_1, v_2, ..., v_n$ y el orden de los valores es seleccionado en orden creciente. Entonces, tendremos 9 variables modeladas en una matriz cuadrada, como en la figura 3.9. Por lo tanto, existirán restricciones diferentes en cada

Figura 3.9: Cuadro Latino 3×3

v_{11}	v_{12}	v_{13}
v_{21}	v_{22}	v_{23}
v_{31}	v_{32}	v_{33}

fila y cada columna. Las simetrías de este problema (al igual que el BIBD) pueden representarse por permutaciones de filas, columnas, además, permutaciones de valores. Este tipo de ruptura es descrito por Flener et al. [100] como *Restricciones lexicográficas*. Aplicando esta idea, podemos fijar la primera fila y primera columna (según el orden lexicográfico) como se muestra en la figura 3.10. Los siguientes pasos para lograr obtener una solución dependerían de la técnica que se emplee para

Figura 3.10: Rompiendo las Simetrías

1	2	3
2	v_{22}	v_{23}
3	v_{32}	v_{33}

la exploración/explotación del vecindario respectivo y del algoritmo de búsqueda que se emplee (i.e GA, TS, etc.). Pero podemos observar, por ejemplo, que los valores fijados de las variables V_{21} y v_{31} no podrán intercambiarse con ninguno otro que se encuentre la misma fila, lo que permite reducir el espacio de búsqueda (inicialmente 3!, se reduce a 2!) en cada fila y/o columna.

Para el caso particular del problema del BIBD, la idea principal es fijar las primeras filas y/o columnas al momento de crear las soluciones candidatas, de tal forma que no sea necesario explorar la totalidad del espacio de búsqueda. Nuestra propuesta se basa en fijar las dos primeras filas de la matriz de incidencia de cada individuo, cumpliendo con las siguientes condiciones:

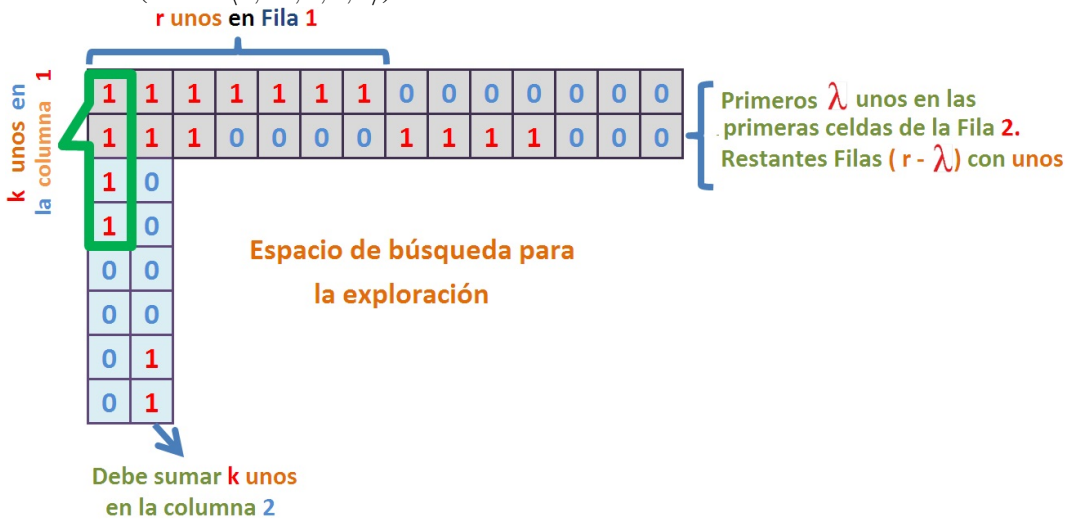
- Fijar los primeros r unos en las primeras celdas de la primera fila de la matriz de incidencia.
- En la segunda fila se debe fijar los λ unos en las primeras celdas de la matriz y los restantes $r-\lambda$ unos en las últimas celdas de ésta fila.

Además de esto es posible fijar las dos primeras columnas de la matriz de incidencia, de la manera siguiente:

- Fijar los primeros k unos en las primeras celdas de la primera columna de la matriz.
- En la segunda columna se debe determinar cuántos unos se colocaron en las primeras filas de esa columna ($\psi = \sum m_{i2}$, con $i = 1...F_i$ para la 2da columna, donde F_i=Número de filas a fijar), luego fijar los $k-\psi$ unos en las últimas celdas de esta columna.

Aquí hay que considerar que para el caso de la fijación de las columnas se debe tomar en cuenta la correspondiente condición de las filas, previamente fijadas. Un ejemplo para la instancia $\langle 8,14,7,4,3\rangle$ se puede ver en la figura 3.11.

Figura 3.11: Problema del BIBD, modelo Primal: Ejemplo de la estrategia utilizada para la reducción de las Simetrías (BIBD-$\langle 8,14,7,4,3\rangle$).



La exploración del espacio de búsqueda se hará por medio de la estrategia de intercambio (*swap*), pero sin incluir las filas y/o columnas que ya se han fijado. El pseudocódigo para la inicialización de los individuos, así como, la exploración del vecindario se puede ver en los algoritmos 9 y 10.

Por otras parte, para el caso de la representación DUAL, podemos utilizar un criterio similar. La ruptura de simetrías para este modelo viene dada por la fijación de las dos primeras filas de la matriz de incidencia, de tal forma que se mantenga la condición de conservar las r celdas a uno en la primera

Algoritmo 9: Pseudocódigo para Exploración de Vecindario Primal.

```
input  : ind, v, b, r, k, λ
output: Mejor individuo del vecindario.
1  begin
2      B_f ← BloqFilas();
       // Bf y Bc representan el número de filas/columnas que se bloquean
3      B_c ← BloqColum();
4      IndMejor ← ind;
5      for i ← B_f to v do
6          for j ← B_c to b do
7              for x ← j + 1 to b do
8                  if ind_ij + ind_ix == 1 then
9                      indVecino ← ind;
10                     Swap(indVecino, j, x);
11                     if Fitness(indVecino) < Fitness(ind) then
12                         IndMejor ← indVecino;
13                     end
14                 end
15             end
16         end
17     end
18     ind ← IndMejor;
19 end
```

de las filas y que no se viole la restricción para λ. Manteniendo el ejemplo utilizado en el apartado anterior, el esquema para la ruptura podría generalizarse como lo indica la figura 3.12.

3.2 Diseño de Plantillas

Las plantillas son esquemas generales que permiten agilizar el trabajo, normalmente de reproducción, para la generación de muchas copias idénticas o casi idénticas (el diseño de plantillas no tiene que ser muy elaborado, personalizado o de corte sofisticado). Si se quiere un trabajo más refinado, más creativo, podemos decir que la plantilla no es sino un punto de partida, una idea aproximada de lo que deseamos hacer. Las plantillas, por norma general, suelen ser utilizadas por sistemas automatizados, con el fin de lograr un objetivo de forma mucho más rápida. Las plantillas, se utilizan en todos las áreas de la industria y la tecnología. Una plantilla nos sirve como la base para agrupar una diversidad de elementos comunes (patrón) y que en sí es lo que constituirá la plantilla.

3.2.1 Visión General

Los problemas de corte y empaquetado (*Cutting and Packing Problems C&P*) son un conjunto de problemas, ampliamente estudiados, enmarcados dentro de la Optimización Combinatoria con una gran variedad de usos de tipo industrial. Partiendo de unos modelos básicos, se encuentra gran cantidad de variantes derivadas. Para la modelización de los problemas de Corte y Empaquetado, la ciencia de

Algoritmo 10: Pseudocódigo Inicialización de Candidatos.

input : $v, b, r, k, \lambda, B_f, B_c$

```
// Bf y Bc representan el número de filas/columnas que se bloquean
```

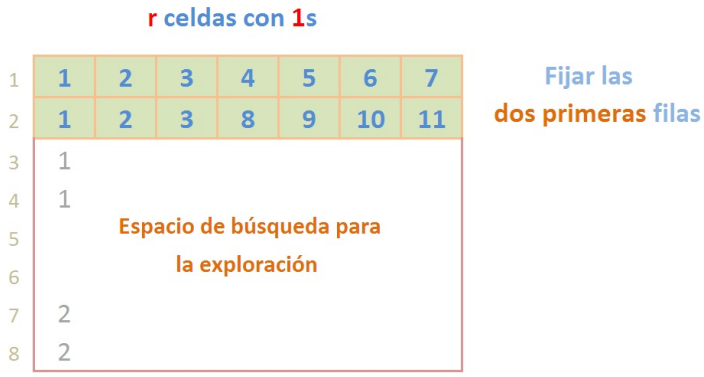
output: ind

```
1  begin
2      ind_ij ← {0,0};
3      ind_1r ← {1};
4      ind_2λ ← {1};
5      ind_2(r+(r−λ)) ← {1};
6      ind_k1 ← {1};
7      ind_(2+(k−2))2 ← {1};
8      for i ← B_f to v do
9          van ← 0;
10         for x ← 1 to B_c do
11             van ← van + ind_ix;
12         end
13         for x ← 1 to van do
14             j ← randEx(b − B_c) + B_c;
15             ind_ij ← 1;
16         end
17     end
18 end
```

programación matemática nos permite abordar este tema. Algunos de estos problemas *C&P* (digamos, Set Packing, Bin Packing Problem (BPP)) son del tipo *NP*-Completos, de acuerdo con la Teoría de la Complejidad Computacional [107]. Básicamente, consiste en colocar un conjunto de elementos, por lo general pequeños, en uno o más objetos, por lo general de grandes dimensiones, sin que se superpongan, de modo de minimizar/maximizar una función objetivo dada. Podemos encontrar muchas aplicaciones importantes en las industrias de la madera, del vidrio, del aluminio, del cuero; además en el diseño de circuitos integrados (Large Scale Integration−*LSI* y Very Large Scale Integration−*VLSI*), en el paginado de periódicos, elaboración de empaques en general (cajas, etiquetas, caratulas, etc.) y en la distribución de la carga de contenedores y camiones. Debido al papel dominante que juegan los patrones y su naturaleza como combinaciones geométricas, se puede decir que los problemas de *C&P* pertenecen al campo de combinatorias geométricas, porque, dentro de cada objeto grande se deben ordenar uno o más objetos pequeños de tal manera de evitar solapamientos y de encajar en los límites del objeto geométrico, es decir, cómo cortar los pequeños objetos. También se denominan combinatorios debido a que se deben tomar decisiones de cuáles serán los elementos a producir y de qué objeto se obtendrán. En otras palabras, a cada uno de los grandes objetos se le asigna una serie de pequeños elementos y además cada elemento se asigna como máximo a un único objeto grande.

La gran gama de aplicaciones mostradas en la literatura llevó a que Dyckhoff [87], a desarrollar una tipología que nos va a permitir clasificar los problemas de Corte y Empaquetado. Una tipología en este caso particular consiste en arreglar los problemas en categorías homogéneas, con base a un conjunto de criterios dados. Pero además, debe ser capaz de suministrar una base compacta para un análisis estructural de los tipos de problemas considerados, y permitir tanto la definición de problemas estándar como el desarrollo de modelos y técnicas algorítmicas e incluso generadores de instancias. En

Figura 3.12: Problema del BIBD: Reducción de simetrías para el modelo DUAL (BIBD-$\langle 8,14,7,4,3\rangle$).



la tabla 3.2, podemos observar la tipología aportada por Dyckhoff respecto a este tipo de problemas.

Tabla 3.2: Tipología para los Problemas de Corte y Empaquetado de Dyckhoff's [87].

1-. **Dimensionalidad**:
Número mínimo dimensiones necesarias
para describir el problema.
2-. **Tipo de asignación entre ítems y objetos**:
(B) Se asignan todos los objetos a una selección de ítems
(V) Se asignan todos los ítems a una selección de objetos
3-. **Surtido de objetos grandes**:
(O) Un único gran objeto
(I) Muchos objetos, todos ellos iguales
(D) Diferentes objetos
4-. **Surtido de ítems pequeños**:
(F) Pocos ítems de diferentes dimensiones
(M) Muchos ítems de muchas dimensiones
(R) Muchos ítems de relativamente pocas dimensiones
(C) Muchos ítems todos iguales

3.2.1.1 Estructura de los Problemas *C&P*

Debido a la similitud que existen entre la gran variedad de problemas de Corte y Empaquetado, se puede extrapolar una estructura común, de la manera siguiente [329]:

Dados dos subconjuntos de elementos, de tal manera que:

- Un conjunto de objetos grandes (entrada, suministro),
- Un conjunto de pequeños elementos (items) (salida, demanda).

Estos pueden estar definidos en una, dos, tres o un número aún mayor (N) de dimensiones geométricas. Se debe seleccionar alguno o todos los objetos pequeños, agrupándolos en uno o más subconjuntos,

con el fin de asignar cada uno de los subconjuntos resultantes a uno de los objetos de gran tamaño tales que la condición geométrica se mantenga, es decir, los pequeños artículos de cada subconjunto que se han colocado sobre el objeto grande correspondan tal que:

- Todos los pequeños objetos del subconjunto, se puedan colocar dentro del objeto de mayor tamaño,
- Los objetos pequeños no se superpongan (no se solapen).

Además, conocida la función de optimización (en una dimensión o varias), se puede notar que una posible solución del problema puede utilizar uno o todos los objetos grandes o uno o todos los objetos pequeños, respectivamente. Formalmente se pueden distinguir cinco subcategorías:

- La selección de los objetos de gran tamaño.
- La selección de los objetos pequeños.
- La agrupación los objetos pequeños.
- La asignación del grupo de objetos pequeños, sobre los objetos más grandes.
- El arreglo de los objetos más pequeños en cada uno de los objetos de gran tamaño con respecto a la geometría de estos.

Aquí, se establece una "jerarquía" en la categorización de problemas: puros, básicos, intermedios, refinados, estándar de primer nivel, estándar de segundo nivel, problemas especiales, problemas extensivos y variantes.

Basándonos en estas categorizaciones, vamos a utilizar un problema que consideramos que puede enmarcarse en la categoría de los problemas de Corte y Empaquetado, debido a que las características de este lo ubican dentro de la categoría de problemas especiales. Este problema es el llamado "Template Design Problem" (*Problema de Diseño de Plantillas* TPD). Si realizamos una revisión de las subcategorías mostradas podemos indicar que se cumplen:

- Tiene dimensionalidad: Los objetos involucrados poseen largo $\times$ ancho (dos dimensiones).
- Se requiere asignar a una plancha (la plantilla) un conjunto de objetos (las cajas).
- Se presentan un conjunto de objetos de pequeñas dimensiones.
- Objetos de gran tamaño (las plantillas).
- Se requiere ubicar los objetos más pequeños sobre los objetos más grandes.
- Los objetos, pequeños y grandes, poseen una forma geométrica particular, que debe ser considerada para lograr el objetivo.
- Además, se debe aclarar que la producción de este tipo de empaques posee una demanda inicial que debe ser cumplida.

Las características especiales que posee el problema a tratar (TPD), son las siguientes:

- Los objetos a acomodar son todos de la misma forma y tamaño.
- Los objetos de gran tamaño (llamados plantillas), tienen forma rectangular, y están formados por subespecies, llamados *slot* donde pueden colocarse varios de los objetos más pequeños.
- Debemos minimizar el número de plantillas a utilizar (objetos más grandes), así como, minimizar el desperdicios (los recortes).
- Se debe maximizar la producción de empaques de tal forma que se puede satisfacer la demanda utilizando el menor número de plantillas.
- La forma de los objetos es ideal pero puede presentar algún tipo de variación (como el color). Esto podemos enmarcarlo como diferentes objetos.

Según lo indicado, el TDP cabe perfectamente dentro de la categoría de los problemas de *C&P* que podrías llamar *especiales*.

Figura 3.13: Problema del TDP: Posible forma del Empaque solicitado[a].

[a]Fotografía de Rich Mitchell - https://www.flickr.com/photos/mtchlra/12199275603

3.2.2 Conceptos Básicos

Actualmente las empresas de fabricación se enfrentan a problemas relacionados con la reducción de los residuos de materia prima. Por lo general, para lograr este objetivo se requiere un esfuerzo importante en el análisis del problema con el fin de obtener un modelo adecuado que permita una mayor producción con un mínimo de residuos. El problema del diseño de plantillas que consideramos es un ejemplo de este tipo de desafíos. El TDP se origina en la industria, cuando se requiere de la elaboración e impresión de los embalajes de cierto tipo de productos (ver figura 3.13), en los cuales la forma es igual pero existen algún tipo de variación en su presentación. Este problema puede ser abordado utilizando técnicas de programación con restricciones y programación matemática. El TDP contemplado aquí se abordó por primera vez en [251] a petición de una empresa que requería producir, utilizando el mínimo número de plantillas, una variedad de embalajes de diversas presentaciones pero manteniendo la misma forma. Esta variedad de productos abarca el empaquetado de comida para el ser humano, animales domésticos y carátulas para revistas. Aquí tratamos variaciones pequeñas, por ejemplo, diferentes embalajes (i.e., cartones) para comidas de gatos sólo difieren en el color de fondo y en la etiqueta impresa, manteniendo su forma. El problema se describe como sigue: Dado un conjunto de variaciones de un determinado diseño, con igual forma y tamaño, de los cuales se conoce la demanda de cada variación, se requiere diseñar una o más plantillas (ver figura 3.14) de igual capacidad para satisfacer la demanda solicitada para cada variación. El diseño debe ser capaz de minimizar el número total de impresiones o troquelados de cada plantilla de tal forma que satisfaga la demanda requerida en cada variación [249]. Considérese el ejemplo mostrado para imprimir cartones (i.e., empaques) [251], para diferentes variedades de comida de gatos según la demanda mostrada en la tabla 3.3. Todos los diseños de cartones tienen el mismo tamaño y la misma forma, y sólo difieren en su impresión. Si consideramos en este ejemplo que cada plantilla consta de 9 espacios o *slots*, tenemos pues 9 slots y 7 variaciones de empaques para los productos. Una forma simple, aunque costosa, de resolver esta situación consiste en utilizar siete plantillas, una para cada variación, pero esto incrementaría los costos de producción y los desperdicios de material terminado. Otra posible

Figura 3.14: Problema del TDP: Posible forma de una Plantilla. Tomado de [251].

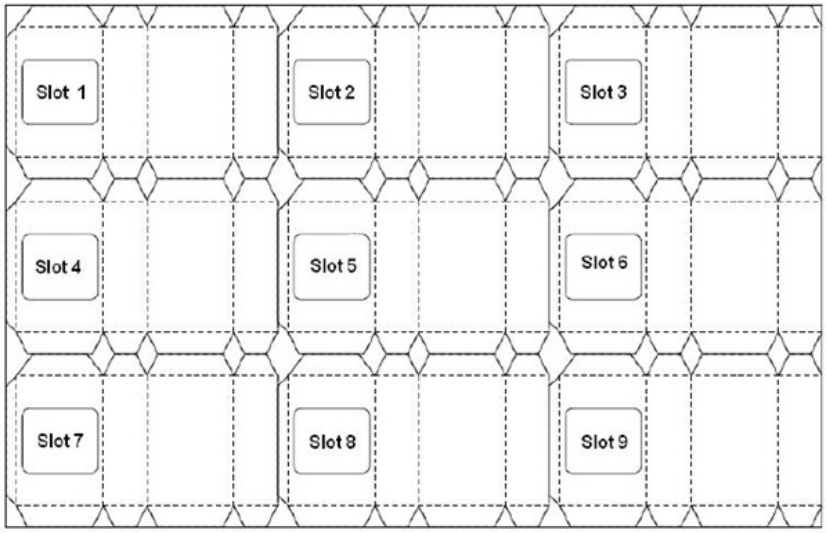


Tabla 3.3: Problema del TDP: Demanda por producto y variación.

Variación	Cantidad ($\times$1000)
Hígado	250
Conejo	255
Atún	260
Pollo Doble	500
Sardina Doble	500
Pollo	800
Sardina	1100
Total	3665

solución podría considerar usar una única plantilla con un slot para cada variación y los otros dos slots restantes dedicarlos a las variaciones con las demandas más altas (pollo y sardina), y realizar 550000 impresiones de la plantilla.

3.2.3 Modelado del TDP

Los problemas de *C&P* puden ser formulados, en forma general, utilizando un modelo matemático basado en programación lineal entera (ILP *Integer Linear Programming*) [117, 182]. Así, por ejemplo, un modelo para el problema de empaquetado de dos dimensiones podemos expresarlo de forma muy básica de la manera siguiente:

Definición 3.2.1 *Dado un vector columna A_j de naturaleza binaria, que contiene n elementos, entonces a_{ij} (donde $i = 1,...,n$) tomará el valor 1, si el ítem i pertenece al j_{vo} patrón, y el valor 0 en otro caso. El conjunto de todos los patrones factibles está representado por la matriz A, formada por todas las columnas A_j ($j = 1,...,M$), tal que:*

$$min \sum_{j=1}^{M} x_j \tag{3.17}$$

$$Sujeto\ a: \sum_{j=1}^{M} a_{ij}x_j = 1\ con\ (i = 1,...,n), x_j \in \{0,1\}\ con\ (j = 1,...,M).$$

Para el caso del Diseño de Plantillas, podemos considerar, de igual forma, un modelo basado en ILP. Así, un posible modelo estaría representado de la manera siguiente [250]: Intentar encontrar con la menor cantidad de plantillas posible un número mínimo de impresiones para satisfacer una demanda conocida. Asumimos un conjunto de T plantillas de tamaño fijo, con s slots en cada plantilla, y V variaciones en la cual cada variación i es descrita por una demanda Q_i. El problema lo planteamos como la minimización de la pérdida de material:

$$\varphi = \min \sum (U_i + O_i) \tag{3.18}$$

Sujeto a:

$$\sum (p_{ij}R_j + U_i - O_i) = Q_i$$
$$\sum p_{ij}R_j \geqslant (1 - l_i)Q_i$$
$$\sum p_{ij}R_j \leqslant (1 + u_i)Q_i$$
$$R_j, U_i, Oi \geqslant 0$$

donde:

- U_i: infraproducción de la variación i,
- O_i: sobreproducción de la variación i,
- R_j: impresiones de la plantilla j,
- Q_i: demanda de la variación i,
- l_i: cantidad de producción, tolerancia, inferior a la demanda de la variación i,
- u_i: cantidad de producción, tolerancia, superior a la demanda de la variación i,
- p_{ij}: slots en la plantilla j en la cual la variación i aparece.

Las tolerancias l_i, u_i permiten ajustar la producción ligeramente por encima o por debajo del objetivo marcado, lo que nos lleva a considerar como factibles aquellas posibles soluciones que se encuentren en el rango: $(1 - l_i)Q_i \leqslant \varphi \leq (1 + u_i)Q_i$.

Debemos mencionar, además, que las soluciones candidatas estarán representadas por un modelo matricial como el indicado por [99]. Cada individuo factible a solución estará representado por una matriz $M = \{p_{ij}\}_{V \times T}$ donde cada $p_{ij} \in \{1..s\}$ representa el número de slots en la plantilla j en la cual la variación i aparece. Se imponen además las siguientes restricciones:

$$\forall i \sum_j R_j p_{ij} \geqslant Q_i$$

$$\forall j \sum_i p_{ij} = s$$

Para determinar cuándo una determinada solución candidata es mejor que otra, se puede utilizar un resolutor de modelos de programación lineal entera, como es el caso de `lp_solve` [25]. Dado

$M = \{p_{ij}\}$, de tal forma que M es una posible configuración del diseño (solución candidata). Si transformamos a M en el modelo respectivo como un *ILP*, al cual llamaremos como τ, la función objetivo f se expresaría como:

$$\textbf{min } f(\tau) = \texttt{lp\_solve}(\tau)$$

Para ello debemos considerar 3 niveles de estratificación para dicha función, como se indica a continuación:

1. El modelo ILP es resuelto por `lp_solve`. Si una solución es encontrada, se utilizará el valor óptimo de ILP como el valor objetivo de la minimización.
2. `lp_solve` no regresa una solución. Se procede a relajar las restricciones de la *sobre-producción*, para volver a aplicar el resolutor. En este caso la solución devuelta es penalizada a agregando un valor numérico elevado, para convertir dicha solución candidata en un fitness de baja calidad.
3. Si aún cuando se aplique la primera relajación, `lp_solve` no logra una solución, se procede a realizar una segunda relajación sobre las restricciones de la *infra-producción*, para reevaluar, aplicando una nueva penalización (de mayor valor), sobre el resultado obtenido. Nótese que en este punto el resolutor de seguro encuentra una posible solución, debido a la máxima relajación del modelo.

3.2.4 El Vecindario para el TDP

Trabajamos con base a la posibilidad de modificar una determinada solución candidata en sólo dos posiciones de alguna plantilla. Esto implica que a la representación matricial se aplican cambios graduales de suma o resta sobre una determinada plantilla, siempre que esto sea posible. Lógicamente, hay que pensar que las soluciones candidatas entregadas a la función objetivo deben ser diseños válidos, para evitar errores inesperados. Por ejemplo, debemos asegurarnos de que en un determinado candidato no queden variaciones sin slots. Una forma de evitar esto es asegurar que cada plantilla contenga V variaciones y que cada variación tenga al menos un slot. En general el vecindario se basa en cambiar el número de slots utilizados por una determinada variación. Así, para una plantilla j se elimina un slot de la variación i y se reasigna a otra variación i', siempre que esto sea posible. Dado $P = \{p_{ij}\}$, podemos expresarlo como:

$$\mathcal{N}(P) = \{P' \mid \exists j,i,i' : p_{ij} > 0, p_{i'j} < s, p'_{ij} = p_{ij} - 1, p'_{i'j} = p_{i'j} + 1\}$$

$$\text{tal que} \quad \begin{array}{l} 1 < p_{ij} < s \\ \forall i \sum_{j=1}^{T} p_{ij} > 1 \\ p_{i'j} < s \end{array}$$

Nótese que: $|\mathcal{N}(P)| \in O(Tv^2)$. El tamaño de un determinado vecindario es aproximadamente $T \cdot V \cdot (V-1)$. Debido al gran tamaño del vecindario una exploración exhaustiva resultaría en un elevado esfuerzo computacional, por lo que, nuestro interés será la evaluación de un pequeño porcentaje de vecinos el cual no excede de 500 vecinos por exploración, esto para el escenario más complejo.

3.2.5 Generación de Soluciones Iniciales

Nos enfocaremos en considerar dos métodos para la generación de las soluciones candidatas iniciales: (1) asignando en forma aleatoria la configuración inicial de las plantillas, respetando siempre la política de no dejar ningún slot vacío, es decir debe cumplirse la restricción $\sum_{j=1}^{T} p_{ij} > 0$; (2) Asignación de la configuración de las plantillas utilizando una heurística que podría consistir en inicializar

Algoritmo 11: Pseudocódigo para inicialización bajo demanda

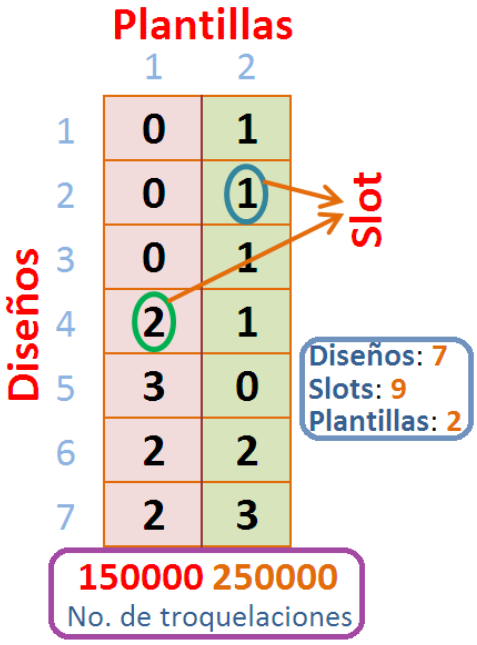
```
1  begin
2      p ← InicPlantilla(T,V);
       // Calcula la fracción de la demanda para cada variación
3      pq ← CalculaProbabilidad(Q);
4      for i ← 1 to T do
           // Rellena la plantilla i-th de acuerdo al porcentje de la demanda
5          Rellenar(p,pq,i,V);
           // Cacule el número de troquelaciones para la plantilla i-th
           // y actualice la probabilidad de acuerdo a la demanda restante
6          R ← min_{j=1...V} {(1−L_j)Q_j/p_ij | p_ij > 0};
7          for j ← 1 to V do
8              Q_j ← Q_j − R∗p_ij;
9          end
10         pq ← CalculaProbabilidad(Q);
11     end
12 end
```

según la demanda requerida por cada variación de acuerdo a la plantilla, es decir que la distribución de la demanda se hace con base al número de plantillas que se desea utilizar en cada candidato. Hay que señalar que existe la posibilidad que quede alguna variación sin asignación de slot, con lo cual es necesario hacer una reparación para evitar este inconveniente. El pseudocódigo correspondiente para el segundo tipo de inicialización podemos revisarlo en el algoritmo 11.

3.2.6 Ejemplo Ilustrativo

Para ilustrar la definición del diseño de plantillas, descrito en la sección 3.2, introduciremos un ejemplo. Consideremos el ejemplo ya descrito en la sección 3.2.2, donde se requiere una demanda como la indicada en la tabla 3.3 (sección 3.2.2), y se dispone de 9 espacios o *slots*, en cada una de las plantillas disponibles, y 7 diseños (variaciones) de empaques para los productos. Una configuración inicial podría ser la que se muestra en la figura 3.15. El número de troquelaciones, o (*pressings*), representa el número de veces que deberá imprimirse dicha plantilla para poder alcanzar la demanda de empaques para un determinado diseño. En la configuración mostrada se puede apreciar que la función fitness deberá devolver el número de pressing mínimo que sería necesario para lograr satisfacer la demanda, tratando de desperdiciar el menor número de empaques terminados (estos empaques no serán comprados por el cliente). Se puede observar que tendríamos 550.000 ($2 \times 150000 + 1 \times 250000$) unidades de empaques del diseño ó variedad **Pollo Doble** para cubrir una demanda solicitada de 500.000 unidades, con lo cual estamos superando las unidades solicitadas. Nótese, que para el caso de los empaques de comida tipo **Hígado** la demanda se cubre exactamente, sin desperdicio de material, no así la variedad **Conejo** donde la producción estaría por debajo de la demanda. Obviamente, esta es una posible solución candidata que no llega a ser un óptimo. Para este ejemplo tendríamos un total de material terminado de 3.600.000 unidades.

Figura 3.15: Problema del TDP: Configuración de posible solución candidata.



3.2.7 Trabajo Relacionado

Un gran número de propuestas han sido publicadas sobre la resolución de los bien conocidos problemas de Corte y empaquetado (*C&P*). Podemos mencionar las propuestas que incluyen algún tipo de técnica basada en metaheurísticas, como es el caso de Dowsland [81], que presenta una de las primeras propuestas utilizando metaheurísticas para resolver el problema del empaquetado de dos dimensiones (**2SP**, Two-Dimensional Bin Packing Problem). Utilizando el algoritmo de recocido simulado (SA) para explorar las soluciones factibles en la que algunos de los artículos se superponen. Durante la búsqueda, la función objetivo es, pues, el área de superposición, y el vecindario contiene todas las soluciones correspondientes a los elementos verticales u horizontales. Jakobs [161], propone un algoritmo genético (GA), para resolver el problema de empaquetado de objetos con forma de polígono. La propuesta se basa en el uso de patrones de empaquetado empleando permutaciones, además de utilizar la estrategia *Bottom-Left*, para determinar la posición de los ítems. Loid et al. [181], emplean la búsqueda tabú (TS) para el diseño de un framework que permite atacar el problema **2SP** (Two-Dimensional Strip Packing Problem). Por medio de un esquema de búsqueda y el empleo de vecindarios que son independientes del problema de empaquetado que se intenta resolver. El trabajo realizado por Shinji [157], que emplea la técnica de *iterated local search* (ILS) sobre una solución inicial generada por perturbaciones leves, se logran encontrar soluciones altamente factibles al problema de RPP (rectangle packing problem). Para resolver un problema de almacenaje de contenedores en una terminal, Rajeeva et al. en [216] emplean la técnica de recocido simulado con un proceso de transferencia e intercambio de vecindarios. Empleando el enfoque de secuencia de pares, los autores descomponen el problema en dos sub-problemas que pueden ser resueltos utilizando programación estocástica.

Respecto a la resolución del problema del diseño de plantillas, se encuentra muy pocas propuestas,

sin embargo, podemos mencionar el trabajo presentado por Proll y Smith [250], que hace uso de un modelo de programación lineal entera y programación con restricciones para resolver esté problema. Prestwich et al. [249], proponen un modelo ILP, en la misma línea de Proll y Smith, que además utiliza una técnica de búsqueda local basada en SAT (*Boolean Satisfiability Algorithm*), para atacar el problema del TDP bajo demanda incierta.

3.2.8 Enfoques Metaheurísticos para el TDP

El TDP muestra una clara estructura combinatoria y puede ser abordado como un problema de optimización combinatoria. Aquí describimos ahora dos técnicas de búsqueda local: *Hill Climbing* (HC), Búsqueda Tabú (TS) y un algoritmo Genético. Para este fin procederemos a describir el posible vecindario involucrado, la función de evaluación y las técnicas y procedimientos utilizados.

3.2.8.1 Vecindario Considerado

Trabajamos con base a la posibilidad de modificar una determinada solución candidata en sólo dos posiciones de alguna plantilla. Esto implica que a la representación matricial se aplican cambios graduales de suma o resta sobre una determinada plantilla, siempre que esto sea posible. Lógicamente, hay que pensar que las soluciones candidatas entregadas a LpSolve[1] deben ser diseños válidos, para evitar errores inesperados. Por ejemplo, debemos asegurarnos de que en un determinado candidato no queden variaciones sin slots. Una forma de evitar esto es asegurar que cada plantilla contenga V variaciones y que cada variación tenga al menos un slot. En la figura 3.16, podemos observar el esquema de vecindario aplicable a la representación primal. En general el vecindario se basa en

Figura 3.16: Problema del TDP: Ejemplo de vecindario Primal (basado en Diseños).

cambiar el número de slots utilizados por una determinada variación. Así, para una plantilla j se

[1] LpSolve es el conocido solucionador de programación lineal (entero) de uso libre. URL: http://lpsolve.sourceforge.net/5.5/

elimina un slot de la variación i y se reasigna a otra variación i', siempre que esto sea posible. Podemos expresarlo como:

$$\exists j : \exists i' : p'_{ij} = p_{ij} - 1, p'_{i'j} = p_{i'j} + 1 \tag{3.19}$$

$$\begin{aligned} \text{tal que} \quad & 1 < p_{ij} < s \\ & \forall i \sum_{j=1}^{T} p_{ij} > 1 \\ & p_{i'j} < s \end{aligned} \tag{3.20}$$

El tamaño de un determinado vecindario es aproximadamente $T \cdot V \cdot (V-1)$. Debido al gran tamaño del vecindario una exploración exhaustiva resultaría en un elevado esfuerzo computacional por lo que consideraremos la evaluación de un pequeño porcentaje de vecinos el cual no debería exceder de 500 vecinos por exploración, esto para el escenario más complejo.

3.2.8.2 Técnicas de Búsqueda Local

Consideramos dos técnicas de búsqueda local, Hill Climbing y Tabu Search. El procedimiento para el HC es estándar: se explora una porción de la vecindad, y se elige la mejor solución (*steepest ascent*), salvo que ésta sea peor que la actual, en cuyo caso se ha llegado a un óptimo local, y se reinicia la búsqueda. Igualmente, para el TS se explora una porción de la vecindad y se impone la siguiente política:

- Trasladarse al mejor vecino que no sea tabú, aún si es peor que la solución actual.
- Un movimiento es tabú si se intenta retroceder a un estado anterior $p' \rightarrow \{p_{ij^{++}}, p_{kj^{--}}\}$ almacenado en la lista tabú. La notación $++$ y $--$ trata de indicar que cada vecino está relacionado con incrementar en un slot una determinada variación y decrementar en un slot otra variación diferente en una misma plantilla tal y como se indicó anteriormente
- La persistencia tabú de los movimientos aceptados se elige aleatoriamente en cada paso dentro del rango $[\beta/2, 3\beta/2]$, donde $\beta = V^2T$.
- El criterio de aspiración es mejorar la mejor solución encontrada hasta el momento.
- Tras un número n_ι de evaluaciones sin mejora se intensifica la búsqueda, volviendo a la mejor solución conocida.

3.2.8.3 Algoritmo Genético

Con respecto al GA, se ha considerado un modelo de evolución de estado estacionario con reemplazo del peor [18]. Se maneja una población de soluciones candidatas y se aplica un mecanismo de selección iterativa, cruce y reemplazo. La selección se hace por medio de un torneo binario y el reemplazo siguiendo una política $(\mu, 1)$ (un nuevo individuo es generado e insertado en la población por medio del reemplazo del peor). Se ha utilizado una variante del operador de cruce uniforme UX [294]. Basándonos en la simetrías del problema, las plantillas son comparadas para encontrar las similitudes entre ellas. Posteriormente, se hace el intercambio de información a nivel de plantillas por una selección al azar del par de plantillas coincidentes (esta selección se hace tomando una variación de uno de los padres). La operación de mutación se realiza de la misma forma como se describió en el vecindario respectivo para las búsquedas locales (veáse la sección 3.2.4).

3.2.8.4 Función Objetivo

Nuestra propuesta se basa en el modelo mostrado por [251]. Según Proll y Smith, el TDP puede formularse como sigue: Intentar encontrar con la menor cantidad de plantillas posible un número mínimo de impresiones para satisfacer una demanda conocida. Asumimos un conjunto de T plantillas de tamaño fijo, con s slots en cada plantilla, y V variaciones en la cual cada variación i es descrita por una demanda Q_i. El problema lo planteamos como la minimización de la pérdida de material:

$$\varphi = \min \sum (U_i + O_i) \tag{3.21}$$

tal que

$$\sum (p_{ij} R_j + U_i - O_i) = Q_i \tag{3.22}$$

$$\sum p_{ij} R_j \geqslant (1 - l_i) Q_i \tag{3.23}$$

$$\sum p_{ij} R_j \leqslant (1 + u_i) Q_i \tag{3.24}$$

$$R_j, U_i, Oi \geqslant 0 \tag{3.25}$$

donde:

- U_i: infraproducción de la variación i,
- O_i: sobreproducción de la variación i,
- R_j: impresiones de la plantilla j,
- Q_i: demanda de la variación i,
- l_i: cantidad de producción, tolerancia, inferior a la demanda de la variación i,
- u_i: cantidad de producción, tolerancia, superior a la demanda de la variación i,
- p_{ij}: slots en la plantilla j en la cual la variación i aparece.

Las tolerancias l_i, u_i permiten ajustar la producción ligeramente por encima o por debajo del objetivo marcado, lo que nos lleva a considerar como factibles aquellas posibles soluciones que se encuentren en el rango: $(1 - l_i)Q_i \leqslant \varphi \leq (1 + u_i)Q_i$. La función de evaluación de un determinado candidato podrá ser obtenida por medio de la aplicación de la conocida biblioteca de resolución de problemas de programación lineal entera, *LpSolve*, la cual nos devuelve los valores R_i que expresan el número de troquelaciones que deben ser aplicadas a cada plantillas para alcanzar la demanda requerida. El objetivo es minimizar la pérdida de materia prima.

3.2.9 Representación Dual: TDP

La representación primal para este problema ya fue descrita en la sección 3.2.3. Existen cinco elementos necesarios a considerar en este problema, los cuales corresponden a: las variaciones de cada producto ($\boldsymbol{V}$), el número de plantillas que se desean utilizar ($\boldsymbol{T}$), el número de celdas (slot) disponibles por cada plantilla ($\boldsymbol{S}$), la demanda del producto, requerida por cada variación ($\boldsymbol{Q}$) y el número de troquelaciones a ser aplicadas a cada plantilla ($\boldsymbol{R}$). Con base a esto podemos formalizar el TDP de la siguiente forma: dado un escenario del problema, éste estará representado por $R = \langle M, Q \rangle$, donde M representa un matriz decimal que contiene la configuración de una solución candidata, Q es un arreglo que contiene la demanda de cada una de las variaciones existentes y R el número total troquelaciones por plantilla. Para el caso particular de representación Dual, cada plantilla se configurará con un conjunto de celdas $\langle s_{1_i} ... s_{d_i} \rangle$, donde i indica el número de celdas asignadas a un determinado diseño d. Para cada plantilla se debe cumplir que $\sum_{i=1}^{d} (s_i) = S$ y a su vez que $\sum_{j=1}^{T} (s_j) > 0$, en cualquier otro

Figura 3.17: Problema del TDP: Ejemplo de Modelo Dual Decimal por Slot (D_{SD}).

caso la configuración será errónea. En general, para encontrar una solución factible φ se debe aplicar la ecuación:

$$\varphi = Min(\sum(U_i + O_i)) \tag{3.26}$$

Incorporando algunas restricciones:

$$\sum(\sum(p_{p_{ij}j})R_j + U_i - O_i) = Q_i \tag{3.27}$$

$$\sum p_{p_{ij}j}R_j \geq (1 - l_i)Q_i \tag{3.28}$$

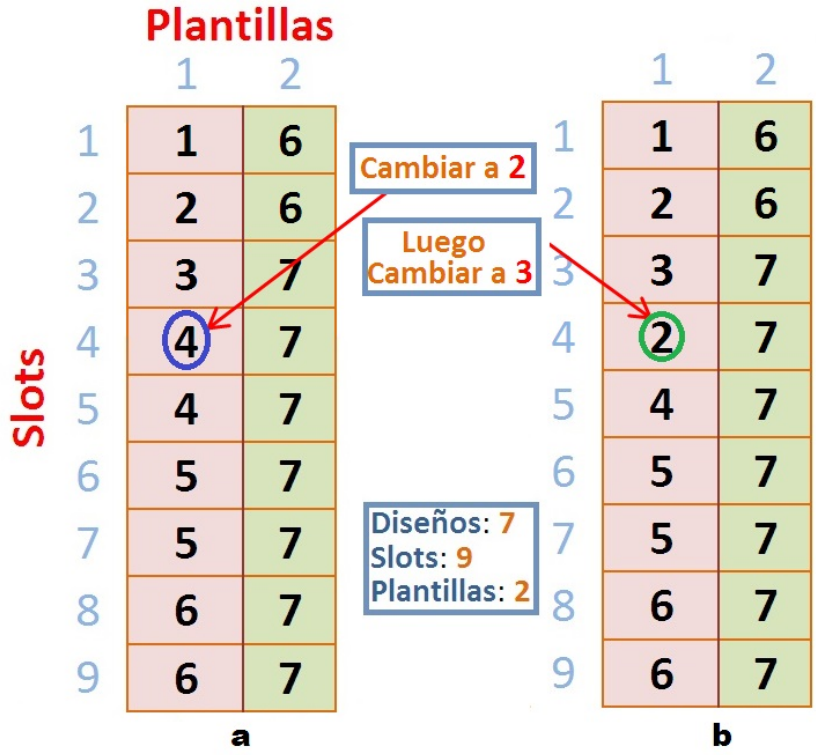
$$\sum p_{p_{ij}j}R_j \leq (1 + u_i)Q_i \tag{3.29}$$

Con $i \in [1, ..., s]$. Observemos que para la minimización del número de troquelaciones se debe agrupar el número de diseños que debe ser aplicado a cada plantilla j, esto se logra con $d_i = \sum_{i=1}^{v} p_{(p_{ij},j)}$. De igual forma se debe tomar en cuenta que cada uno de los diseños debe ocupar al menos un slot, es decir $d_i > 0$, para considerar este individuo como una solución candidata factible. Un ejemplo, de la representación *dual* propuesta, lo podemos ver en la figura 3.17.

3.2.9.1 Posible Vecindario para el Modelo Dual del TDP

La evaluación del vecindario para esta propuesta se seleccionó de acuerdo a la posibilidad de poder cambiar, en un determinado slot, un diseño por otro. Se recorre la matriz correspondiente por cada plantilla y se escoge de manera progresiva un slot cuyo valor será cambiado iterativamente por cada uno de los diseños disponibles (siempre que sea distinto al que ocupa actualmente el slot). Un ejemplo para este tipo de exploración se muestra en la figura 3.18. Podemos apreciar que en la figura 3.18 literal (a), el slot 4 puede ser evaluado con el diseño 2 (que es distinto de 4), luego con el diseño 3 (figura 3.18 literal (b)) y así sucesivamente esta evaluar la totalidad de los diseños disponibles para este escenario.

Figura 3.18: Problema del TDP: Ejemplo de vecindario Dual (basado en Slot).



3.2.9.2 Propuesta de Ruptura de Simetrías para TDP

Para aplicar la ruptura de simetrías en este problema debemos considerar cada uno de los modelos utilizados, en forma separada; además, consideraremos que los diseños se han ordenado, en forma ascendente, con base a la demanda de la producción. Consideraremos la utilización de los siguientes modelos alternativos de representación de las soluciones candidatas:

- Modelo basado en diseños (o variaciones): La representación PRIMAL que deseamos emplear está relacionada directamente con el número de diseños de cada escenario, así un eventual candidato será una matriz $M_{V\times T}$ que contendrá en sus celdas el número de slot que ocupará cada diseño en una determinada plantilla, con lo cual las filas de la matriz representan los diseños y las columnas, las plantillas (ver figura 3.19).
- Modelo basado en Slots: La representación DUAL que consideráremos está relacionada directamente con el número de slots de cada escenario, así un eventual candidato será una matriz $M_{S\times T}$ que contendrá en sus celdas el número del diseño que estará ocupando dicho slot en una determinada plantilla, con lo cual las filas de la matriz representan los slots y las columnas las plantillas (ver figura 3.17).

Claramente, el modelo Primal, está definido de tal forma que se está aplicando ruptura de las simetrías en forma explícita, ya que la forma de exploración del vecindario (principalmente para la propuesta de búsqueda local), no permiten la presencia de simetrías de filas. Sin embargo, pueden aparecer un gran número de simetrías relacionadas con la permutación de plantillas en las diferentes configuraciones de las soluciones candidatas. Consideremos la solución mostrada en la figura 3.20(a), si

Figura 3.19: Problema del TDP: Ejemplo de vecindario para el modelo **PRIMAL**

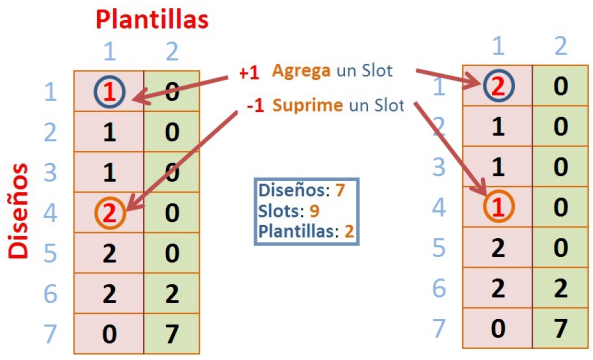
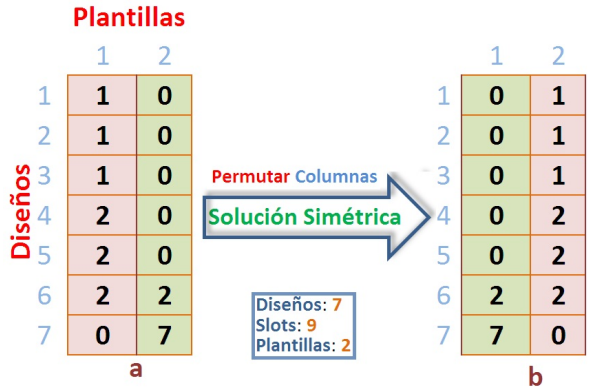


Figura 3.20: Problema del TDP: Reducción de simetrías, Modelo **PRIMAL**.



aplicamos un intercambio en las columnas para esta configuración, el resultado sería una solución candidata simétrica respecto a la original. La solución mostrada en la figura 3.20(b), representa una configuración simétrica respecto a la solución candidata de la figura 3.20(a), en este caso sólo se han permutado las dos columnas (observemos que para este ejemplo, sólo se están utilizando dos plantillas). Con el fin de minimizar la presencia de este tipo de simetrías hemos propuesto la utilización de un orden lexicográfico para la configuración de las soluciones candidatas. Es decir, las nuevas configuraciones de soluciones candidatas serán evaluadas imponiendo la siguiente restricción donde: la plantilla i es **mayor que** la plantilla $i+1$ (los resultados serán idénticos si aplicáramos el operador relacional 'menor que'):

$$\forall T_i \in T : \; T_i > T_{i+1} \tag{3.30}$$

Al aplicar la ordenación lexicográfica por columnas, la configuración del candidato mostrado en la figura 3.20(b), no podrá ser considerada debido a la restricción impuesta.

El modelo Dual, en cambio, presenta mayor grado de simetrías, ya que además de las simétricas de columnas (plantillas), se caracteriza porque presenta simetrías de filas. Si en la figura 3.17, intercambiamos la fila uno (S1) con la fila 9 (S9), el resultado sería un individuo simétrico respecto al original, por poner un ejemplo. Para romper las simetrías por filas, en este caso para las propuestas de búsqueda local, se implementará un mecanismo de ruptura parcial de las simetrías por medio de la

Figura 3.21: Problema del BIBD: Ejemplo de ruptura de simetrías, para el vecindario del modelo **DUAL**.

imposición de un orden incremental en la ubicación de las demandas por cada dieseño en la plantilla respectiva.

- Se asignaran a las primeras celdas de la matriz los $D/2$ diseños existentes, que tienen las demandas de menor valor numérico.
- Los restantes diseños serán asignados en los slot inferiores de la plantilla, que corresponderán a las demandas de mayor valor numérico.

Importante señalar que se alternará la asignación de los $D/2$ diseños de las celdas iniciales y luego finales, para realizar una ruptura por columnas (considerando sólo las $T-1$ plantillas iniciales). Para las propuestas Genéticas, se utilizará un tipo de cruce por columnas (plantillas) eliminando las plantillas repetidas. Así, si utilizamos el ejemplo de la figura 3.19, para la plantilla 1, ocuparíamos los primeros slots con los diseños de menor demanda, digamos D1-D4 para los slots S1-S5 (sólo a manera de ejemplo), y los restantes diseños (de mayor demanda) con los restantes slots. No podría ocurrir que se intercambiará D1 (que está en el slot S1) con D6 (que está en el slot S9). La figura mostrada en 3.21, no ocurriría nunca, ya que esto no sería permitido por la restricción impuesta. Claramente sería una ruptura parcial de las simetrías. Pero considerando que, al igual que el modelo primal, sólo se explora un pequeño porcentaje de vecinos, de cierta forma, esto también ayudará con la ruptura de las simetrías.

3.3 Técnicas Metaheurísticas para Resolver el BIBD

Para abordar el problema del Diseño de Bloques Incompletos hemos empleado tres técnicas metaheurísticas conocidas, dos de ellas son de corte trayectorial y una basada en población, como se describe a continuación:

- Dos técnicas trayectoriales:
  - Búsqueda local por acenso a la colina (Hill Climbing).
  - Búsqueda local Tabú (Tabu Search).
- Búsqueda basada en algoritmos genéticos (Genetic Algorithm).

3.3.1 Técnicas Trayectoriales

Tal como se mencionó en la sección 3.1.7, las técnicas metaheurísticas trayectoriales de búsquedas locales se basan en la exploración de un determinado vecindario proporcionado por una eventual solución candidata. Por tanto, es conveniente, tratar en primer lugar la representación de las soluciones y la estructura de esta vecindad, y posteriormente el espacio de búsqueda subyacente. Sin embargo, hay que considerar además, las dos representaciones alternativas que han sido definidas para atacar el problema, como se ha descrito en las secciones 3.1.3 y 3.1.8. Una vez considerado este aspecto, resultarían los entornos de vecindad descritos en la sección 3.1.4 y 3.1.8.1. Una configuración inicial, para una determinada instancia $I = \langle M, \varphi \rangle$ (con M como la matriz de incidencia y φ como el conjunto de parámetros v, b, r, k, y λ), estaría dada por el cumplimiento de las restricciones correspondientes en cada una de las filas de M, de esta forma una posible solución candidata (si consideramos el modelo primal del problema BIBD) sería una simple permutación $m_i = \langle m_1, m_2, ..., m_n \rangle \in \mathbb{P}_n$, con $m_i \in \mathbb{N}_n^+$ y $\mathbb{P}_n$ representa el total de permutaciones disponibles de los elementos en $\mathbb{N}_n^+$, con $\mathbb{N}^+$ los enteros positivos y n representaría el número de filas de la matriz M.

Estos vecindarios son explorados por medio de la utilización de técnicas de búsqueda local (LS). Una de las técnicas empleadas en esta investigación es el conocido algoritmo *steepest-ascent* (Hill Climbing, *HC*), en la cual, dada una solución candidata m, su vecindario $\mathcal{N}(m)$ es explorado y la mejor solución encontrada es tomada como el candidato actual. Al no encontrar una mejora el proceso se considera como un estancamiento y se procede a reiniciar en otro punto del espacio de búsqueda tomando una nueva solución candidata que puede ser generada al azar. La segunda técnica considerada es la búsqueda tabú, la cual ha sido descrita anteriormente en la sección 3.1.7. El mecanismo de funcionamiento, así como los parámetros involucrados en el desarrollo de las técnicas de búsqueda local, utilizadas en esta investigación, se aplican en forma idéntica para ambas representaciones alternativas del problema de diseño de bloques incompleto, por esta razón solo hemos descrito las técnicas que corresponden al modelo que hemos llamado *Primal*.

3.3.2 Técnica Poblacional

Hemos considerado utilizar un algoritmo genético de estado estacionario (**steady state GA**). Para el caso de la representación Primal, consideramos dos variantes de este. Ambos utilizan selección basada en torneo binario con reemplazo del peor individuo de la población. Pero debemos aclarar que difieren ligeramente en la etapa de reproducción. El primero de ellos, al cual denotaremos como GA_{bf}, hace uso de la vecindad basada en intercambio *bit-flip*, utiliza cruce uniforme y para la operación de mutación se emplea *bit-flip*, definida en la sección 3.1.7.1. A la segunda variante, la hemos llamado GA_{sw}, y se relaciona con el vecindario *swap*. Debemos aclarar que este último algoritmo utiliza cruce a nivel de filas (las cuales son seleccionadas en forma aleatoria de cada uno de los padres), pero empleando la operación *swap* (sección 3.1.7.1), para la aplicación de la etapa de mutación. Como se puede deducir, esto implica que la configuración de cada una de las filas nunca cambia y, por lo tanto, la inicialización de la población debe hacerse con soluciones que cumplan todas las restricciones correspondientes por fila, de la misma forma que fue empleada en los casos de las técnicas HC_{sw} y TS_{sw}. De igual forma, esta inicialización guiada puede ser opcionalmente aplicada para la técnica GA_{bf}, aunque no es obligatorio. Para mantener la diversidad en la población, las dos variantes, descartan los individuos duplicados (con igual carga genética). Además, se introduce un mecanismo de reinicio para reactivar la búsqueda cada vez que se produce un estancamiento. Esto se hace conservando una fracción de $f_{\%}$ de los mejores individuos en la población actual, y refrescando al resto de la población

con individuos aleatorios. Este procedimiento se desencadena después de varias evaluaciones de n_ι sin mejoría de la actual solución. Para el caso de la representación alternativa (*Dual*), el algoritmo genético funcionaria de forma similar, pero considerando que el vecindario tiene una variación (véase la sección 3.1.8.1, para mayores detalles), las operaciones de cruce y mutación también varían, y su funcionamiento quedaría de la forma que se describe a continuación:

- Cruce a nivel de filas. Para este caso no hemos considerado el cruce uniforme a nivel de celdas puesto que los resultados obtenidos en el modelo primal resultaron poco prometedores.
- Mutación basada en el vecindario (cambiar un valor de la celda por un valor decimal que no sea repetido, según las restricciones de las filas).

3.3.3 Resultados Experimentales para el BIBD

Las pruebas se han realizado sobre 86 instancias (tabla 3.4) del problema del Diseño de Bloques Incompletos equilibrados con $v \times b < 1000$.

Tabla 3.4: Conjunto de instancias utilizadas para realizar la experimentación sobre el problema del BIBD, para las técnicas que se han propuesto en este trabajo de investigación (i.e. LSs, GAs, MAs) y que han sido tomadas de [32, 246].

ID	v	b	r	k	λ	vb	ID	v	b	r	k	λ	vb
1	8	14	7	4	3	112	44	25	25	9	9	3	625
2	11	11	5	5	2	121	45	15	42	14	5	4	630
3	10	15	6	4	2	150	46	21	30	10	7	3	630
4	9	18	8	4	3	162	47	16	40	10	4	2	640
5	13	13	4	4	1	169	48	16	40	15	6	5	640
6	10	18	9	5	4	180	49	9	72	32	4	12	648
7	8	28	14	4	6	224	50	15	45	21	7	9	675
8	15	15	7	7	3	225	51	13	52	16	4	4	676
9	11	22	10	5	4	242	52	13	52	24	6	10	676
10	16	16	6	6	2	256	53	10	72	36	5	16	720
11	12	22	11	6	5	264	54	19	38	18	9	8	722
12	10	30	12	4	4	300	55	11	66	30	5	12	726
13	16	20	5	4	1	320	56	22	33	12	8	4	726
14	9	36	16	4	6	324	57	15	52	26	7	12	780
15	8	42	21	4	9	336	58	27	27	13	13	6	729
16	13	26	8	4	2	338	59	21	35	15	9	6	735
17	13	26	12	6	5	338	60	10	75	30	4	10	750
18	10	36	18	5	8	360	61	25	30	6	5	1	750
19	19	19	9	9	4	361	62	20	38	19	10	9	760
20	11	33	15	5	6	363	63	16	48	15	5	4	768
21	14	26	13	7	6	364	64	16	48	18	6	6	768
22	16	24	9	6	3	384	65	12	66	22	4	6	792
23	12	33	11	4	3	396	66	12	66	33	6	15	792
24	21	21	5	5	1	441	67	9	90	40	4	15	810
25	8	56	28	4	12	448	68	13	65	20	4	5	845
26	10	45	18	4	6	450	69	11	77	35	5	14	847

Tabla 3.4: Continuación de la tabla

ID	v	b	r	k	λ	vb	ID	v	b	r	k	λ	vb
27	15	30	14	7	6	450	70	21	42	10	5	2	882
28	16	30	15	8	7	480	71	21	42	12	6	3	882
29	11	44	20	5	8	484	72	21	42	20	10	9	882
30	9	54	24	4	9	486	73	16	56	21	6	7	896
31	13	39	12	4	3	507	74	10	90	36	4	12	900
32	13	39	15	5	5	507	75	15	60	28	7	12	900
33	16	32	12	6	4	512	76	18	51	17	6	5	918
34	15	35	14	6	5	525	77	22	42	21	11	10	924
35	12	44	22	6	10	528	78	15	63	21	5	6	945
36	23	23	11	11	5	529	79	16	60	15	4	3	960
37	10	54	27	5	12	540	80	16	60	30	8	14	960
38	8	70	35	4	15	560	81	31	31	6	6	1	961
39	17	34	16	8	7	578	82	31	31	10	10	3	961
40	10	60	24	4	8	600	83	31	31	15	15	7	961
41	11	55	20	4	6	605	84	11	88	40	5	16	968
42	11	55	25	5	10	605	85	22	44	14	7	4	968
43	18	34	17	9	8	612	86	25	40	16	10	6	1000

Este grupo de instancias muestran una diversidad de escenarios que van desde poco complejos (fácil resolución), así como otros con un alto grado de complejidad (de difícil resolución) [246]. Cada uno de los algoritmos de búsqueda (digamos HC y TS) se ha ejecutado 30 veces para cada instancia del problema. Cada lanzamiento se realizó considerando $n_{vec} = 2 \cdot 10^6$ vecinos en cada una de las instancias estudiadas; el valor de este parámetro no ha sido seleccionado en forma arbitraria, sino que hemos considerado realizar un número de iteraciones que concuerden con el número de nodos visitados en el trabajo realizado por Prestwich descrito en [246]. En el caso de TS, la intensificación se realiza tras $n_{intens} = n_{vec}/10$ vecinos sin mejora. Para el algoritmo genético se tomó una población de 100 individuos, con una probabilidad de cruce de $p_X = 0.9$, y una probabilidad de mutación $p_M = 1/vb$, además se han considerado las posibles combinaciones empleando los dos operadores de *cruce/mutación* que se han descrito previamente.

3.3.3.1 Convenciones de Notación para las Técnicas Metaheurísticas del BIBD

Cada una de las técnicas en particular se identificará a través de una secuencia de identificadores separados por un punto. Para el caso particular de las metaheurísticas básicas (como se describe en 3.3) las cuales corresponden a Hill Climbing (HC), búsqueda tabú (TS), algoritmo Genético (GA). Todos ellos se basan en el vecindario de intercambio. Además, para los métodos basados en población (es decir, GA), el procedimiento de recombinación se caracteriza por el operador particular utilizado –nos enfocamos aquí en el uso del operador de cruce voraz (Gd), para el caso de la representación alternativa Primal y Cruce estándar para la representación Dual – y por su aridad, es decir, el número m de padres utilizados (denotados como Am).

Además, utilizamos un símbolo de asterisco ($*$) para indicar el uso de las configuraciones para la ruptura de simetría, una letra B para indicar que los individuos están codificados en una representación

binaria (es decir, el modelo alternativo que hemos llamado primal) y una letra *D* para indicar que están codificados en una representación numérica *decimal* (es decir, la formulación dual).

Consideremos un ejemplo para estos algoritmos no-colaborativos: $Hc.B$ (respecto a $Hc.B*$) denota la técnica Hill Climbing, la cual ha sido implementada usando el modelo primal (en representación Binaria) y el esquema sin ruptura de simetrías (respeto a con ruptura); $Ts.D*$ (respecto de $Ts.D$), el cual denota la técnica basada en búsqueda tabú, implementada con el modelo Dual (representación Decimal) y el esquema con ruptura de simetrías (respecto a la configuración sin-ruptura); de igual forma, $GA.B*.A2.Gd$ denota un algoritmo genético con cruce de 2 padres y operador voraz (como ha sido explicado en la sección 4.1.3 – véase [270] para mayores detalles relacionados con este tipo de operador); esta propuesta hace uso de la codificación binaria (modelo primal) con una configuración que reduce las simetrías (con ruptura de simetrías); y para el caso de $GA.D.A4.Gd$ representa un algoritmo genético con cruce de 4 padres, operador Voraz, con formulación dual y sin ruptura de simetrías.

3.3.3.2 Resultados para el Modelo Primal del BIBD: Sin Ruptura de Simetrías

Los datos mostrados en las tablas 3.5 y 3.6, representan los resultados de las ejecuciones de los algoritmos de búsqueda local.

Estas estadísticas muestran que las versiones estudiadas para los vecindarios *bit-flip* de los algoritmos, no muestran un desempeño prometedor para la resolución del problema del diseño de bloques incompletos, pues obtienen un bajo porcentaje de soluciones para las instancias estudiadas, sólo el $TS_{\text{bit-flip}}$ es capaz de resolver 27 de ellas, lo que representa el 31.40%. Por otra parte, en la tabla 3.7 se muestran los datos de los resultados obtenidos al aplicar las dos versiones del algoritmo genético estudiado.

Dichas estadísticas muestran un pobre desempeño de estos algoritmos al enfrentar este problema, ya que el mayor porcentaje de soluciones se alcanza en la versión con vecindario swap, y este no supera el 23.26%. En general las estadísticas muestran una clara mejoría de los algoritmos de búsqueda local con respecto a las técnicas genéticas básicas en la obtención de soluciones para el problema analizado. Se puede observar que el TS_{swap} muestra el mejor desempeño ya que se comporta significativamente mejor que el resto de los algoritmos en más del 93% de las instancias (según un test Wilcoxon ranksum – tabla 3.8 –, a nivel estándar 0.05). TS_{swap} obtiene soluciones para 57 de las 86 instancias consideradas, lo que nos da un 66.28% de instancias resueltas, frente a un 23.26% mostrado por GA_{swap}. Nótese además cómo el número de ejecuciones con éxito es notablemente mayor.

Estas pruebas preliminares nos han permitido divisar un enfoque de trabajo concentrándonos en un aspecto que hasta este punto parece relevante, como resultado de esta experimentación: Emplear el vecindario basado en intercambio (***swap*** (Sw)) y dejar de lado el vecindario *bit-flip*, ya que la las versiones de los algoritmos con vecindario *Swap* superan en más del 64% de los casos a su contraparte *bit-flip* como lo podemos ver en la tabla 3.8. Otro aspecto relevante que hemos detectado está relacionado con el operador de cruce estándar (UX), en el cual encontramos dos problemas: el primero es su ceguera: selecciona aleatoriamente las filas de los padres, lo que puede ser una buena estrategia de diversificación. Sin embargo, las instancias del problema más difíciles pueden requerir un enfoque con una mayor grado de intensificación, por ejemplo, podemos basarnos en una elección más inteligente (*smart*) de las filas para construir la descendencia. El segundo problema es el hecho de que el operador UX no considera las simetrías: dos matrices diferentes pueden representar la misma solución a través de una permutación de las filas, pero UX las trataría como soluciones

Tabla 3.5: Resultados para el algoritmo Hill Climbing aplicado sobre el problema del BIBD, con 30 ejecuciones. Instancias tomadas de [32, 246]. $\bar{x}, \sigma, B, S$, representan la media, la desviación estándar y el mejor fitness encontrado en las ejecuciones, así como el número de ejecuciones con éxito. Para el Modelo Primal, sin ruptura de simetrías.

ID	Vecindario Bit-Flip			Vecindario Swap			ID	Vecindario Bit-Flip			Vecindario Swap		
	$\bar{x} \pm \sigma$	B	S	$\bar{x} \pm \sigma$	B	S		$\bar{x} \pm \sigma$	B	S	$\bar{x} \pm \sigma$	B	S
1	3.50 ± 1.12	0	2	0.00 ± 0.00	0	30	44	139.67 ± 3.80	130	0	100.40 ± 3.08	95	0
2	5.13 ± 4.62	0	13	0.00 ± 0.00	0	30	45	35.33 ± 3.17	29	0	15.00 ± 2.14	11	0
3	5.83 ± 1.24	4	0	0.00 ± 0.00	0	30	46	84.60 ± 3.53	78	0	50.63 ± 3.22	45	0
4	5.73 ± 1.44	0	1	0.00 ± 0.00	0	30	47	29.63 ± 2.74	23	0	13.10 ± 2.41	8	0
5	3.13 ± 4.15	0	19	0.00 ± 0.00	0	30	48	47.57 ± 2.63	43	0	20.47 ± 2.60	11	0
6	11.27 ± 1.63	7	0	1.47 ± 1.93	0	19	49	25.93 ± 3.80	13	0	2.00 ± 2.00	0	15
7	7.80 ± 1.19	5	0	0.00 ± 0.00	0	30	50	54.90 ± 4.43	41	0	17.60 ± 2.09	13	0
8	36.60 ± 2.68	30	0	4.33 ± 5.17	0	17	51	24.67 ± 2.44	16	0	7.63 ± 1.62	4	0
9	14.67 ± 1.70	10	0	3.97 ± 0.84	0	1	52	42.20 ± 3.00	36	0	10.03 ± 2.76	4	0
10	37.23 ± 1.67	33	0	4.47 ± 5.67	0	18	53	38.87 ± 4.54	29	0	3.17 ± 2.18	0	9
11	22.27 ± 2.14	16	0	6.13 ± 1.26	4	0	54	96.27 ± 4.14	88	0	42.00 ± 3.75	33	0
12	11.30 ± 1.35	9	0	2.53 ± 1.93	0	11	55	31.93 ± 3.45	21	0	5.87 ± 2.17	0	2
13	20.67 ± 2.07	16	0	8.40 ± 4.12	0	5	56	104.97 ± 5.44	94	0	61.43 ± 4.26	50	0
14	11.20 ± 1.66	7	0	0.27 ± 1.00	0	28	57	72.53 ± 4.81	59	0	44.57 ± 1.65	41	0
15	12.07 ± 2.03	8	0	0.00 ± 0.00	0	30	58	214.20 ± 4.93	200	0	137.03 ± 6.52	111	0
16	16.17 ± 2.08	12	0	6.13 ± 1.06	4	0	59	110.00 ± 4.82	102	0	58.27 ± 3.85	51	0
17	28.53 ± 2.20	22	0	9.63 ± 1.58	6	0	60	25.50 ± 3.51	17	0	3.43 ± 1.84	0	6
18	19.67 ± 1.96	14	0	2.67 ± 1.89	0	10	61	73.53 ± 4.81	63	0	49.53 ± 4.11	41	0
19	78.00 ± 2.49	73	0	45.50 ± 3.38	35	0	62	116.73 ± 5.07	102	0	48.60 ± 3.42	42	0
20	19.17 ± 2.18	15	0	4.10 ± 1.35	0	2	63	41.77 ± 3.29	35	0	18.47 ± 2.92	13	0
21	38.77 ± 3.09	33	0	13.37 ± 2.26	6	0	64	50.37 ± 4.20	40	0	20.63 ± 3.01	13	0
22	40.47 ± 2.05	35	0	19.80 ± 2.17	15	0	65	24.13 ± 2.46	20	0	6.97 ± 2.51	4	0
23	16.30 ± 1.62	13	0	5.00 ± 1.18	4	0	66	51.70 ± 4.18	43	0	7.40 ± 2.56	0	1
24	48.87 ± 4.57	34	0	23.30 ± 7.20	0	1	67	33.90 ± 6.15	15	0	3.33 ± 2.36	0	9
25	19.20 ± 3.52	12	0	0.00 ± 0.00	0	30	68	26.67 ± 3.62	17	0	9.73 ± 2.35	4	0
26	15.37 ± 1.80	9	0	2.00 ± 2.00	0	15	69	38.47 ± 3.39	31	0	5.43 ± 2.06	0	1
27	46.27 ± 3.00	41	0	17.00 ± 2.42	11	0	70	64.90 ± 3.83	56	0	36.90 ± 3.16	29	0
28	59.83 ± 3.88	52	0	22.70 ± 2.52	16	0	71	76.73 ± 3.92	69	0	44.70 ± 3.28	37	0
29	24.00 ± 2.52	19	0	3.87 ± 1.73	0	4	72	125.93 ± 6.71	106	0	59.17 ± 5.88	46	0
30	17.63 ± 2.51	13	0	0.67 ± 1.49	0	25	73	53.83 ± 3.85	43	0	22.50 ± 3.28	17	0
31	20.63 ± 2.37	16	0	6.37 ± 1.94	4	0	74	28.00 ± 6.10	14	0	5.70 ± 2.58	0	3
32	26.97 ± 2.47	20	0	8.63 ± 1.74	5	0	75	66.43 ± 4.98	53	0	19.10 ± 2.75	13	0
33	43.53 ± 2.28	37	0	20.67 ± 2.56	15	0	76	60.73 ± 4.05	48	0	30.60 ± 4.57	21	0
34	40.87 ± 2.60	35	0	16.43 ± 2.68	10	0	77	153.00 ± 6.62	131	0	67.27 ± 5.53	54	0
35	36.20 ± 3.91	28	0	6.13 ± 1.67	4	0	78	43.30 ± 3.57	35	0	16.37 ± 3.02	11	0
36	135.73 ± 3.86	128	0	84.43 ± 4.10	72	0	79	39.20 ± 4.46	27	0	14.60 ± 3.53	9	0
37	27.83 ± 3.61	20	0	3.53 ± 1.43	0	4	80	83.07 ± 6.50	70	0	24.83 ± 3.22	17	0
38	29.00 ± 4.93	16	0	0.13 ± 0.72	0	29	81	134.73 ± 5.88	122	0	100.77 ± 5.04	87	0
39	67.13 ± 4.35	57	0	28.17 ± 2.00	23	0	82	231.17 ± 5.85	219	0	175.60 ± 6.37	160	0
40	19.40 ± 3.02	11	0	2.67 ± 1.89	0	10	83	312.40 ± 6.15	300	0	206.10 ± 6.14	194	0
41	19.23 ± 2.43	14	0	4.10 ± 1.60	0	3	84	42.53 ± 5.08	34	0	7.70 ± 2.64	0	1
42	26.90 ± 3.22	20	0	4.53 ± 1.67	0	2	85	102.07 ± 4.68	94	0	57.73 ± 4.63	44	0
43	85.57 ± 3.60	79	0	34.93 ± 3.22	28	0	86	167.60 ± 6.52	150	0	98.57 ± 5.78	88	0

diferentes y mezclaría las filas ciegamente para producir una solución inviable (es decir, una solución con filas repetidas). Para abordar estos dos inconvenientes se ha ideado un operador de recombinación codicioso (GrX – también lo podemos denotar como Gd –). GrX comienza creando un conjunto con todas las filas disponibles de los padres, claro está que se descartan las filas repetidas (es decir con la misma carga genética) y luego (si hay suficientes filas diferentes, de lo contrario se invoca el UX estándar), seleccionando aleatoriamente una fila inicial, y luego probando todas las filas disponibles, para mayores detalles se puede revisar la sección 4.1.3.

En la tabla 3.9 se muestran los resultados obtenidos para los algoritmos genéticos con los operadores estándar y voraz. Los resultados muestran un mayor número de soluciones de instancias para el operador de cruce Ux, con 48.83%, frente al operador GrX, con un porcentaje de 29.07%.

3.3.3.3 Resultados para el Modelo Dual del BIBD: Sin Ruptura de Simetrías

Los datos mostrados en las tablas 3.10 y 3.11, representan los resultados de las ejecuciones de los algoritmos de búsqueda local, así como los algoritmos genéticos aplicados a la representación alter-

Tabla 3.6: Resultados experimentales del BIBD para el algoritmo Tabu Search con 30 ejecuciones. Instancias tomadas de [32, 246]. $\bar{x}, \sigma, B, S$, representan la media, la desviación estándar y el mejor fitness encontrado en las ejecuciones, así como el número de ejecuciones con éxito. Para el Modelo Primal, sin ruptura de simetrías.

ID	Vecindario Bit-Flip			Vecindario Swap			ID	Vecindario Bit-Flip			Vecindario Swap		
	$\bar{x} \pm \sigma$	B	S	$\bar{x} \pm \sigma$	B	S		$\bar{x} \pm \sigma$	B	S	$\bar{x} \pm \sigma$	B	S
1	3.30 ± 2.62	0	10	0.00 ± 0.00	0	30	44	108.80 ± 6.48	96	0	66.70 ± 9.48	22	0
2	6.93 ± 6.06	0	12	0.00 ± 0.00	0	30	45	17.53 ± 2.64	9	0	4.30 ± 1.44	0	2
3	6.17 ± 2.40	0	2	0.00 ± 0.00	0	30	46	58.37 ± 4.32	50	0	32.37 ± 1.74	29	0
4	4.30 ± 1.68	0	3	0.00 ± 0.00	0	30	47	16.87 ± 4.13	8	0	2.43 ± 1.99	0	12
5	3.67 ± 4.81	0	18	0.00 ± 0.00	0	30	48	24.23 ± 3.50	18	0	8.70 ± 1.72	4	0
6	7.33 ± 2.53	4	0	0.00 ± 0.00	0	30	49	1.50 ± 2.01	0	19	0.00 ± 0.00	0	30
7	1.67 ± 2.10	0	18	0.00 ± 0.00	0	30	50	23.43 ± 3.96	16	0	6.03 ± 1.35	4	0
8	28.60 ± 5.37	15	0	0.00 ± 0.00	0	30	51	8.70 ± 2.42	0	1	0.00 ± 0.00	0	30
9	9.27 ± 2.86	4	0	0.00 ± 0.00	0	30	52	14.37 ± 2.95	10	0	0.40 ± 1.20	0	27
10	27.13 ± 7.10	0	1	0.00 ± 0.00	0	30	53	6.20 ± 3.26	0	3	0.00 ± 0.00	0	30
11	13.50 ± 3.39	5	0	0.00 ± 0.00	0	30	54	50.77 ± 5.97	39	0	24.53 ± 2.05	20	0
12	3.93 ± 2.73	0	8	0.00 ± 0.00	0	30	55	6.83 ± 2.19	4	0	0.00 ± 0.00	0	30
13	16.60 ± 4.10	8	0	0.00 ± 0.00	0	30	56	68.33 ± 4.83	51	0	40.47 ± 2.68	35	0
14	3.13 ± 2.05	0	8	0.00 ± 0.00	0	30	57	43.17 ± 2.57	38	0	40.13 ± 0.43	40	0
15	1.60 ± 2.39	0	19	0.00 ± 0.00	0	30	58	160.80 ± 8.58	145	0	100.43 ± 3.85	91	0
16	10.60 ± 2.24	4	0	0.00 ± 0.00	0	30	59	66.53 ± 4.08	56	0	35.90 ± 2.53	30	0
17	16.33 ± 3.25	8	0	2.93 ± 1.77	0	8	60	2.47 ± 2.12	0	12	0.00 ± 0.00	0	30
18	7.27 ± 2.79	0	1	0.00 ± 0.00	0	30	61	54.60 ± 5.22	43	0	14.40 ± 11.64	0	10
19	59.27 ± 5.28	50	0	0.37 ± 1.97	0	29	62	65.37 ± 6.74	54	0	29.77 ± 1.80	26	0
20	8.37 ± 2.44	4	0	0.00 ± 0.00	0	30	63	20.73 ± 3.59	13	0	5.17 ± 1.44	0	1
21	22.70 ± 3.94	15	0	5.77 ± 0.88	4	0	64	23.53 ± 2.70	19	0	8.13 ± 1.69	4	0
22	27.43 ± 3.23	23	0	4.70 ± 4.41	0	12	65	5.50 ± 2.03	0	1	0.00 ± 0.00	0	30
23	8.33 ± 2.29	4	0	0.00 ± 0.00	0	30	66	12.57 ± 4.11	5	0	0.00 ± 0.00	0	30
24	33.33 ± 12.10	0	2	0.00 ± 0.00	0	30	67	2.23 ± 2.56	0	16	0.00 ± 0.00	0	30
25	1.43 ± 1.99	0	19	0.00 ± 0.00	0	30	68	7.37 ± 2.11	4	0	0.00 ± 0.00	0	30
26	3.77 ± 2.35	0	7	0.00 ± 0.00	0	30	69	7.27 ± 2.54	0	1	0.13 ± 0.72	0	29
27	26.10 ± 4.04	18	0	8.13 ± 1.82	4	0	70	40.53 ± 3.87	32	0	18.23 ± 1.84	14	0
28	34.70 ± 3.91	27	0	11.70 ± 1.51	9	0	71	47.37 ± 4.42	38	0	24.17 ± 2.60	17	0
29	8.57 ± 2.67	4	0	0.00 ± 0.00	0	30	72	70.63 ± 5.49	61	0	36.77 ± 2.60	32	0
30	2.67 ± 2.34	0	12	0.00 ± 0.00	0	30	73	21.90 ± 3.18	17	0	8.30 ± 1.73	4	0
31	8.47 ± 2.40	4	0	0.00 ± 0.00	0	30	74	2.20 ± 2.70	0	17	0.00 ± 0.00	0	30
32	12.93 ± 2.05	9	0	0.27 ± 1.00	0	28	75	24.90 ± 4.72	15	0	5.63 ± 2.33	0	2
33	25.63 ± 3.40	20	0	9.37 ± 1.87	4	0	76	32.00 ± 3.53	24	0	14.13 ± 2.53	9	0
34	22.67 ± 2.66	18	0	6.70 ± 1.39	4	0	77	84.23 ± 8.92	68	0	43.87 ± 2.20	41	0
35	12.70 ± 3.25	6	0	0.00 ± 0.00	0	30	78	16.47 ± 2.80	11	0	3.17 ± 2.07	0	8
36	100.07 ± 6.39	85	0	49.47 ± 18.59	0	3	79	15.00 ± 3.17	10	0	1.43 ± 2.06	0	20
37	6.13 ± 2.63	0	2	0.00 ± 0.00	0	30	80	32.00 ± 5.14	22	0	10.23 ± 2.01	6	0
38	2.33 ± 2.83	0	17	0.00 ± 0.00	0	30	81	109.23 ± 7.99	89	0	22.90 ± 18.70	0	12
39	36.27 ± 3.98	29	0	15.47 ± 1.78	12	0	82	184.40 ± 6.34	173	0	134.13 ± 5.08	124	0
40	3.30 ± 2.69	0	11	0.00 ± 0.00	0	30	83	230.37 ± 10.02	207	0	159.63 ± 5.92	148	0
41	5.47 ± 1.65	4	0	0.00 ± 0.00	0	30	84	7.60 ± 3.59	0	1	0.50 ± 1.28	0	26
42	7.47 ± 3.31	4	0	0.00 ± 0.00	0	30	85	61.90 ± 4.47	54	0	34.80 ± 2.56	31	0
43	48.80 ± 5.53	39	0	20.47 ± 1.87	16	0	86	107.70 ± 5.54	99	0	65.70 ± 3.64	56	0

nativa, que hemos llamado DUAL. Debemos aclarar que a partir de esta sección llamaremos en forma indistinta al operador **GrX** como **Gd**.

Podemos observar que la técnica de búsqueda local TS muestra el mejor rendimiento, si nos referimos al número de soluciones resueltas, puesto que logra resolver cerca del 50% de las instancias frente a un 6.97% del HC.

Sin embargo, para el modelo DUAL, quien muestra un mejor rendimiento es el algoritmo genético basado en cruce estándar, ya que logra resolver el 53.49% de las instancias, algo más del 3% de lo que logra la técnica TS de la misma representación.

3.3.3.4 Resultados para el Modelo Primal del BIBD: Con Ruptura de Simetrías

Al aplicar el modelo con ruptura de simetrías para la representación Primal, encontramos que los algoritmos con mejor desempeño (considerando el número de instancias resueltas por cada uno de ellos) lo tienen la versión Tabú con vecindario Swap con 59.30% y la versión GA que utiliza el operador de cruce voraz con cruce multi-padre (aridad 4), la cual resuelve un 50% de instancias.

Tabla 3.7: Resultados experimentales del BIBD para el Algoritmo Genético con 30 ejecuciones. Instancias tomadas de [32, 246]. $\bar{x}, \sigma, B, S$, representan la media, la desviación estándar y el mejor fitness encontrado en las ejecuciones, así como el número de ejecuciones con éxito. Para el Modelo Primal, sin ruptura de simetrías.

ID	Vecindario Bit-Flip			Vecindario Swap			ID	Vecindario Bit-Flip			Vecindario Swap		
	$\bar{x}\pm\sigma$	B	S	$\bar{x}\pm\sigma$	B	S		$\bar{x}\pm\sigma$	B	S	$\bar{x}\pm\sigma$	B	S
1	3.60 ± 2.15	0	7	0.43 ± 1.31	0	27	44	128.83 ± 6.00	115	0	107.20 ± 6.49	84	0
2	6.27 ± 6.17	0	14	1.17 ± 3.01	0	26	45	27.37 ± 5.49	16	0	19.47 ± 2.79	14	0
3	6.43 ± 2.28	4	0	3.03 ± 2.43	0	11	46	72.40 ± 4.66	59	0	57.57 ± 4.86	45	0
4	5.60 ± 1.69	4	0	2.30 ± 2.02	0	13	47	23.70 ± 3.42	16	0	18.60 ± 2.79	13	0
5	6.10 ± 4.78	0	10	2.10 ± 3.99	0	23	48	35.30 ± 4.53	26	0	25.40 ± 3.49	16	0
6	7.90 ± 2.37	4	0	4.27 ± 1.67	0	3	49	7.17 ± 2.00	3	0	4.87 ± 1.89	0	2
7	3.90 ± 1.96	0	5	0.80 ± 1.60	0	24	50	35.27 ± 4.26	27	0	23.17 ± 2.65	17	0
8	25.93 ± 8.03	0	1	18.17 ± 7.88	0	3	51	16.47 ± 2.92	11	0	11.73 ± 2.57	4	0
9	10.13 ± 2.90	4	0	6.80 ± 2.14	0	1	52	22.93 ± 3.83	17	0	14.73 ± 2.50	10	0
10	30.97 ± 8.21	0	1	21.33 ± 8.43	0	2	53	13.53 ± 2.80	9	0	7.10 ± 2.39	4	0
11	14.40 ± 2.27	9	0	9.47 ± 2.93	5	0	54	71.23 ± 5.64	63	0	48.23 ± 4.37	42	0
12	6.47 ± 2.12	4	0	4.93 ± 1.41	0	1	55	14.93 ± 3.17	9	0	9.10 ± 2.37	5	0
13	18.33 ± 3.50	8	0	15.10 ± 2.13	11	0	56	93.57 ± 7.07	80	0	70.83 ± 5.47	56	0
14	5.27 ± 1.65	0	1	2.53 ± 2.14	0	12	57	54.00 ± 4.31	48	0	45.73 ± 1.97	42	0
15	4.03 ± 1.58	0	3	1.40 ± 2.03	0	20	58	190.97 ± 8.00	175	0	149.83 ± 7.65	135	0
16	12.57 ± 3.37	7	0	9.83 ± 1.88	7	0	59	90.30 ± 6.20	79	0	66.97 ± 5.74	52	0
17	17.47 ± 3.38	11	0	12.83 ± 2.66	6	0	60	10.97 ± 2.82	4	0	6.73 ± 1.98	0	1
18	7.87 ± 2.83	0	1	4.97 ± 1.30	4	0	61	68.53 ± 6.38	54	0	59.63 ± 5.29	47	0
19	61.87 ± 4.12	53	0	51.10 ± 3.79	42	0	62	86.93 ± 6.75	76	0	58.47 ± 4.56	50	0
20	10.27 ± 2.46	6	0	7.40 ± 1.91	4	0	63	33.73 ± 4.56	27	0	24.63 ± 3.06	18	0
21	23.70 ± 3.84	17	0	18.27 ± 2.67	12	0	64	37.50 ± 4.64	28	0	26.83 ± 3.34	20	0
22	30.93 ± 3.14	24	0	24.70 ± 2.88	19	0	65	15.23 ± 2.77	11	0	10.33 ± 2.31	6	0
23	11.03 ± 2.15	7	0	7.93 ± 2.32	4	0	66	21.97 ± 3.75	14	0	13.27 ± 2.61	9	0
24	44.47 ± 7.58	23	0	30.73 ± 13.49	0	4	67	9.83 ± 2.53	5	0	6.73 ± 2.34	0	1
25	4.40 ± 2.08	0	4	1.87 ± 2.33	0	18	68	17.60 ± 3.79	9	0	12.27 ± 3.46	4	0
26	7.60 ± 2.32	0	1	5.70 ± 1.92	4	0	69	17.93 ± 3.32	12	0	10.83 ± 2.42	6	0
27	30.77 ± 4.05	23	0	22.67 ± 3.43	15	0	70	56.10 ± 5.87	46	0	44.67 ± 4.41	35	0
28	39.57 ± 4.23	30	0	27.80 ± 3.52	21	0	71	64.53 ± 4.79	55	0	53.10 ± 3.33	42	0
29	11.23 ± 2.38	7	0	7.90 ± 1.85	4	0	72	101.00 ± 7.10	87	0	68.20 ± 5.69	56	0
30	6.43 ± 2.59	0	2	4.43 ± 2.28	0	4	73	39.77 ± 4.91	29	0	29.57 ± 3.90	23	0
31	14.03 ± 3.09	8	0	10.83 ± 3.19	5	0	74	12.17 ± 3.42	5	0	8.60 ± 2.65	4	0
32	17.37 ± 2.93	9	0	12.23 ± 2.77	7	0	75	41.03 ± 5.43	31	0	25.53 ± 3.23	18	0
33	33.70 ± 3.63	25	0	25.73 ± 1.93	22	0	76	51.00 ± 4.64	40	0	37.10 ± 3.58	29	0
34	29.07 ± 3.72	22	0	21.33 ± 2.81	15	0	77	122.70 ± 9.99	92	0	81.00 ± 5.32	68	0
35	17.03 ± 2.46	12	0	11.47 ± 2.51	7	0	78	32.50 ± 4.06	25	0	22.00 ± 4.02	15	0
36	116.90 ± 5.03	106	0	91.93 ± 4.63	84	0	79	27.37 ± 3.64	20	0	21.50 ± 2.64	17	0
37	10.20 ± 2.29	5	0	5.87 ± 1.56	4	0	80	51.77 ± 5.75	40	0	32.27 ± 2.79	28	0
38	4.83 ± 2.21	0	3	3.70 ± 2.25	0	7	81	135.50 ± 8.48	116	0	115.10 ± 7.03	94	0
39	47.73 ± 3.71	40	0	33.00 ± 4.04	25	0	82	225.23 ± 7.18	211	0	195.50 ± 7.35	178	0
40	8.40 ± 2.27	5	0	6.00 ± 1.59	3	0	83	295.73 ± 11.02	279	0	227.70 ± 9.40	209	0
41	11.93 ± 2.95	6	0	7.83 ± 2.30	4	0	84	19.03 ± 3.41	12	0	12.37 ± 2.17	7	0
42	12.70 ± 3.59	6	0	8.53 ± 2.05	4	0	85	85.93 ± 5.84	75	0	68.17 ± 4.98	60	0
43	57.93 ± 5.43	47	0	41.07 ± 4.28	34	0	86	149.83 ± 8.03	135	0	115.77 ± 5.87	96	0

Tabla 3.8: Resultados para el BIBD al aplicar el Test estadístico Wilcoxon ranksum. Cada entrada en la tabla indica el porcentaje de casos en los que el algoritmo etiquetado en la fila supera al algoritmo etiquetado en la columna. De las técnicas metaheurísticas (LS y GA), para el Modelo Primal, sin ruptura de simetrías.

	HC_{bf}	HC_{sw}	TS_{bf}	TS_{sw}	GA_{bf}	GA_{sw}
HC_{bf}	–	0.00%	0.00%	0.00%	1.16%	0.00%
HC_{sw}	100%	–	76.64%	0.00%	100%	91.86%
TS_{bf}	95.35%	12.79%	–	0.00%	86.05%	34.88%
TS_{sw}	100%	97.67%	100.00%	–	100%	93.02%
GA_{bf}	93.02%	0.00%	0.00%	0.00%	–	0.00%
GA_{sw}	98.84%	6.98%	26.74%	5.81%	98.84%	–

3.3.3.5 Resultados para el Modelo Dual del BIBD: Con Ruptura de Simetrías

Al aplicar el modelo con ruptura de simetrías para la representación Dual, encontramos que los algoritmos con mejor desempeño lo tienen la versión Tabú con 53.49% y la versión GA que utiliza el operador de cruce Ux con cruce multi-padre (aridad 4), al cual obtiene un total de 51.16% de instan-

Tabla 3.9: Resultados experimentales del BIBD para los GAs (30 ejecuciones por instancia, tomadas de [32, 246]) utilizando los operadores UX (GA_{UX}) y GrX (GA_{GrX}). Para el Modelo Primal, sin ruptura de simetrías.

ID	GA_{UX}			GA_{GrX}			ID	GA_{UX}			GA_{GrX}		
	$\bar{x}\pm\sigma$	B	S	$\bar{x}\pm\sigma$	B	S		$\bar{x}\pm\sigma$	B	S	$\bar{x}\pm\sigma$	B	S
1	0.00 ± 0.00	0	30	0.00 ± 0.00	0	30	44	87.67 ± 6.52	70	0	114.03 ± 3.74	104	0
2	0.00 ± 0.00	0	30	0.00 ± 0.00	0	30	45	11.63 ± 1.72	8	0	20.07 ± 2.41	15	0
3	0.00 ± 0.00	0	30	0.00 ± 0.00	0	30	46	43.17 ± 3.31	36	0	63.17 ± 4.28	55	0
4	0.27 ± 1.00	0	28	0.13 ± 0.72	0	29	47	9.30 ± 1.57	6	0	19.00 ± 2.28	14	0
5	0.00 ± 0.00	0	30	0.00 ± 0.00	0	30	48	16.13 ± 2.32	11	0	28.37 ± 3.01	21	0
6	0.27 ± 1.00	0	28	0.93 ± 1.69	0	23	49	0.83 ± 1.53	0	23	4.03 ± 1.82	0	3
7	0.00 ± 0.00	0	30	0.00 ± 0.00	0	30	50	14.10 ± 1.92	11	0	27.10 ± 3.04	20	0
8	4.40 ± 6.81	0	21	8.17 ± 7.43	0	13	51	5.73 ± 1.71	0	1	11.30 ± 2.30	5	0
9	3.53 ± 1.67	0	5	3.50 ± 1.86	0	6	52	7.80 ± 1.78	4	0	16.43 ± 1.73	14	0
10	7.50 ± 6.64	0	13	6.10 ± 7.23	0	17	53	2.23 ± 1.96	0	13	6.87 ± 1.93	3	0
11	4.87 ± 1.41	0	1	6.43 ± 2.29	0	2	54	35.60 ± 3.06	30	0	58.37 ± 4.14	48	0
12	1.17 ± 1.79	0	21	2.13 ± 2.00	0	14	55	4.00 ± 1.79	0	4	10.13 ± 2.22	5	0
13	5.23 ± 5.65	0	16	11.23 ± 2.47	7	0	56	53.97 ± 4.70	44	0	79.70 ± 4.06	72	0
14	0.67 ± 1.49	0	25	0.67 ± 1.49	0	25	57	41.20 ± 1.14	40	0	47.90 ± 1.92	45	0
15	0.00 ± 0.00	0	30	0.10 ± 0.54	0	29	58	127.63 ± 5.00	118	0	164.00 ± 5.64	151	0
16	5.60 ± 1.23	4	0	6.23 ± 1.67	4	0	59	48.93 ± 3.38	43	0	75.10 ± 4.54	65	0
17	7.17 ± 1.83	4	0	10.67 ± 1.60	7	0	60	1.77 ± 1.94	0	16	6.77 ± 2.65	0	2
18	1.40 ± 1.85	0	19	2.90 ± 1.96	0	9	61	41.57 ± 4.28	29	0	65.03 ± 3.47	56	0
19	31.23 ± 8.50	0	1	48.83 ± 3.82	41	0	62	42.77 ± 2.59	37	0	69.23 ± 5.12	58	0
20	2.87 ± 1.89	0	9	4.80 ± 1.47	0	1	63	14.57 ± 2.30	10	0	27.93 ± 2.66	24	0
21	9.93 ± 1.59	6	0	15.33 ± 2.24	11	0	64	17.77 ± 2.58	10	0	32.63 ± 2.97	26	0
22	15.57 ± 2.39	10	0	22.83 ± 2.24	18	0	65	4.47 ± 1.89	0	3	10.63 ± 2.32	5	0
23	2.87 ± 1.93	0	9	6.27 ± 1.55	4	0	66	5.90 ± 2.04	0	1	14.37 ± 2.56	8	0
24	7.03 ± 9.66	0	19	32.83 ± 5.32	16	0	67	1.57 ± 1.71	0	16	6.47 ± 1.78	3	0
25	0.00 ± 0.00	0	30	0.37 ± 1.11	0	27	68	6.33 ± 1.72	3	0	13.77 ± 2.28	10	0
26	1.13 ± 1.75	0	21	2.20 ± 2.10	0	14	69	4.23 ± 2.06	0	4	10.97 ± 2.68	4	0
27	13.70 ± 1.59	10	0	20.97 ± 2.26	17	0	70	31.03 ± 3.33	26	0	53.00 ± 3.60	46	0
28	17.27 ± 2.62	12	0	28.10 ± 3.49	20	0	71	38.23 ± 3.63	29	0	64.03 ± 3.77	54	0
29	3.87 ± 1.89	0	5	6.40 ± 1.65	4	0	72	51.97 ± 3.77	45	0	84.70 ± 4.63	73	0
30	0.27 ± 1.00	0	28	1.77 ± 1.98	0	16	73	16.90 ± 2.23	13	0	35.37 ± 2.88	30	0
31	4.43 ± 2.12	0	4	10.23 ± 1.94	7	0	74	2.40 ± 2.18	0	13	8.73 ± 2.19	3	0
32	6.60 ± 1.80	4	0	11.90 ± 2.17	6	0	75	15.33 ± 2.94	11	0	31.83 ± 2.82	24	0
33	15.23 ± 2.29	11	0	25.63 ± 2.73	17	0	76	24.37 ± 2.63	19	0	45.63 ± 3.18	38	0
34	12.83 ± 2.48	6	0	21.13 ± 2.32	18	0	77	60.73 ± 3.44	54	0	99.43 ± 4.83	89	0
35	4.53 ± 1.33	0	1	10.27 ± 1.90	6	0	78	12.23 ± 2.38	9	0	27.20 ± 3.61	20	0
36	74.23 ± 4.76	63	0	96.13 ± 4.54	86	0	79	11.13 ± 2.78	7	0	25.37 ± 2.59	19	0
37	1.87 ± 2.05	0	16	4.70 ± 1.72	0	2	80	20.43 ± 2.17	15	0	39.93 ± 3.31	33	0
38	0.23 ± 0.88	0	28	1.07 ± 1.53	0	20	81	90.17 ± 7.62	71	0	128.07 ± 5.01	118	0
39	22.27 ± 2.35	17	0	36.93 ± 3.36	28	0	82	167.93 ± 7.60	151	0	216.63 ± 5.49	205	0
40	1.40 ± 1.91	0	19	4.93 ± 1.91	0	2	83	194.63 ± 6.68	181	0	254.77 ± 7.58	237	0
41	2.57 ± 2.36	0	13	6.87 ± 2.22	0	1	84	5.30 ± 1.83	0	1	14.43 ± 3.01	6	0
42	2.87 ± 1.93	0	9	7.97 ± 1.74	4	0	85	49.83 ± 3.59	42	0	83.10 ± 3.81	76	0
43	28.97 ± 3.03	24	0	45.90 ± 3.56	40	0	86	90.90 ± 5.02	81	0	138.37 ± 5.38	128	0

cias.

3.3.4 Análisis Estadísticos para las Técnicas Metaheurísticas del BIBD

Para revisar el comportamiento de las diferentes técnicas metaheurísticas al ejecutarse en forma separada, optamos por la utilización de un torneo de rangos (rank-based approach y otros tests como: Friedman e Iman-Davenport, Holm y Wilcoxon ranksum). Para ello, hemos obtenido el rango r_j^i de cada grupo de algoritmos j en cada instancia i (se le asigna el valor 1 a la mejor técnica, y el rango k a la peor de ellas, k es el número de algoritmos utilizados en el torneo); en caso de empate se asigna un rango promedio. Para poder realizar la clasificación correspondiente para todos los casos hemos utilizado la suma del número de soluciones encontradas en cada instancia de problema por cada uno de los algoritmos que pertenece al grupo.

El mejor grupo recibe el rango 1 y el peor recibe el rango k, donde $k = 24$ es el número de grupos involucrados en el ranking. Los resultados para este torneo los podemos ver en la figura 3.22, allí podemos notar que la técnica basada en búsqueda local Tabú que emplea la representación alternativa Binaria (Primal), obtiene la mejor posición en el rango.

Tabla 3.10: Resultados experimentales del BIBD para los Algoritmos de Búsqueda local HC y TS (para el Modelo DUAL) con 30 ejecuciones. Instancias tomadas de [32, 246]. $\bar{x}, \sigma, B, S$, representan la media, la desviación estándar y el mejor fitness encontrado en las ejecuciones, así como el número de ejecuciones con éxito. Para el Modelo Dual, sin ruptura de simetrías.

ID	Hc.D			Ts.D			ID	Hc.D			Ts.D		
	$\bar{x}\pm\sigma$	B	S	$\bar{x}\pm\sigma$	B	S		$\bar{x}\pm\sigma$	B	S	$\bar{x}\pm\sigma$	B	S
1	0.00 ± 0.00	0	30	0.00 ± 0.00	0	30	44	138.33 ± 4.41	130	0	80.90 ± 5.20	71	0
2	8.00 ± 1.79	0	1	0.00 ± 0.00	0	30	45	31.80 ± 2.21	28	0	5.93 ± 1.61	4	0
3	4.73 ± 1.41	0	1	0.13 ± 0.72	0	29	46	82.40 ± 3.74	74	0	38.00 ± 2.31	33	0
4	4.13 ± 0.50	4	0	0.13 ± 0.72	0	29	47	25.40 ± 3.31	18	0	5.57 ± 1.26	4	0
5	1.87 ± 3.38	0	23	0.00 ± 0.00	0	30	48	44.27 ± 3.57	38	0	11.37 ± 2.33	6	0
6	8.73 ± 1.50	6	0	0.67 ± 1.49	0	25	49	10.33 ± 2.74	4	0	0.00 ± 0.00	0	30
7	2.93 ± 1.77	0	8	0.00 ± 0.00	0	30	50	47.13 ± 4.22	40	0	9.43 ± 1.65	4	0
8	34.53 ± 1.63	32	0	6.87 ± 6.90	0	15	51	20.07 ± 2.71	14	0	0.13 ± 0.72	0	29
9	12.67 ± 1.49	10	0	2.70 ± 1.92	0	10	52	31.20 ± 3.41	22	0	2.73 ± 1.97	0	10
10	35.73 ± 1.91	32	0	3.33 ± 6.25	0	23	53	18.27 ± 2.29	14	0	19.80 ± 6.72	4	0
11	19.53 ± 1.33	18	0	5.60 ± 1.85	0	1	54	90.33 ± 4.56	82	0	29.37 ± 2.64	24	0
12	7.53 ± 1.84	4	0	0.00 ± 0.00	0	30	55	19.87 ± 3.14	12	0	0.53 ± 1.36	0	26
13	18.00 ± 2.42	8	0	5.57 ± 4.69	0	12	56	103.00 ± 6.23	86	0	45.93 ± 3.48	40	0
14	6.40 ± 1.82	4	0	0.00 ± 0.00	0	30	57	194.00 ± 0.00	194	0	9.77 ± 2.45	4	0
15	4.60 ± 1.72	0	2	0.00 ± 0.00	0	30	58	210.33 ± 5.44	200	0	116.50 ± 6.86	98	0
16	13.93 ± 1.90	10	0	2.27 ± 2.17	0	14	59	106.33 ± 5.64	94	0	41.57 ± 3.76	35	0
17	25.53 ± 2.40	20	0	6.47 ± 1.52	4	0	60	12.47 ± 1.91	8	0	0.00 ± 0.00	0	30
18	12.80 ± 2.10	8	0	0.13 ± 0.72	0	29	61	67.60 ± 4.91	56	0	39.17 ± 6.45	29	0
19	74.93 ± 2.91	70	0	37.20 ± 7.64	16	0	62	108.67 ± 4.14	100	0	38.47 ± 4.29	28	0
20	15.47 ± 1.71	12	0	0.80 ± 1.60	0	24	63	38.20 ± 4.08	30	0	7.47 ± 1.73	4	0
21	35.73 ± 2.11	30	0	10.67 ± 1.66	8	0	64	47.33 ± 4.66	38	0	9.90 ± 2.34	4	0
22	38.07 ± 2.28	34	0	15.60 ± 2.93	4	0	65	18.07 ± 2.56	12	0	0.93 ± 1.69	0	23
23	13.53 ± 1.69	10	0	0.67 ± 1.49	0	25	66	30.87 ± 3.49	22	0	72.00 ± 6.94	56	0
24	44.53 ± 2.78	38	0	2.63 ± 6.06	0	25	67	12.33 ± 2.37	8	0	2.73 ± 3.18	0	15
25	5.73 ± 1.61	4	0	0.00 ± 0.00	0	30	68	20.67 ± 4.27	14	0	0.80 ± 1.60	0	24
26	9.80 ± 2.09	6	0	0.00 ± 0.00	0	30	69	23.27 ± 3.67	16	0	3.93 ± 2.73	0	8
27	42.33 ± 2.83	36	0	12.27 ± 2.39	6	0	70	58.73 ± 4.05	52	0	21.93 ± 1.84	18	0
28	55.33 ± 3.07	46	0	16.73 ± 1.98	13	0	71	74.80 ± 5.26	64	0	29.13 ± 3.19	20	0
29	17.87 ± 2.00	14	0	0.00 ± 0.00	0	30	72	125.67 ± 6.99	112	0	43.20 ± 4.28	37	0
30	8.13 ± 1.93	4	0	0.00 ± 0.00	0	30	73	48.53 ± 4.06	40	0	12.17 ± 3.10	6	0
31	16.80 ± 2.34	12	0	0.67 ± 1.49	0	25	74	15.07 ± 3.30	8	0	10.10 ± 4.01	4	0
32	23.93 ± 2.66	18	0	3.13 ± 1.98	0	8	75	52.47 ± 4.94	44	0	12.37 ± 3.59	5	0
33	41.67 ± 3.10	36	0	13.63 ± 1.66	9	0	76	60.47 ± 5.21	50	0	20.67 ± 4.11	12	0
34	38.07 ± 3.29	32	0	9.43 ± 1.54	7	0	77	141.93 ± 6.21	130	0	146.83 ± 5.56	135	0
35	26.27 ± 2.72	20	0	1.33 ± 1.89	0	20	78	37.93 ± 4.49	26	0	23.10 ± 11.34	11	0
36	132.33 ± 4.79	122	0	73.27 ± 4.61	65	0	79	30.73 ± 3.67	24	0	5.77 ± 2.62	0	2
37	15.20 ± 2.04	10	0	0.00 ± 0.00	0	30	80	69.07 ± 5.97	56	0	295.50 ± 7.46	285	0
38	7.40 ± 2.26	4	0	0.00 ± 0.00	0	30	81	129.13 ± 4.25	120	0	51.87 ± 22.63	0	4
39	65.33 ± 4.24	56	0	19.43 ± 1.73	14	0	82	232.73 ± 6.25	220	0	144.50 ± 4.88	137	0
40	10.87 ± 2.11	6	0	0.00 ± 0.00	0	30	83	307.80 ± 8.20	290	0	178.50 ± 7.97	163	0
41	13.47 ± 2.31	10	0	0.00 ± 0.00	0	30	84	23.73 ± 4.64	16	0	45.07 ± 6.28	32	0
42	18.93 ± 2.82	12	0	0.00 ± 0.00	0	30	85	98.60 ± 5.27	86	0	38.43 ± 3.68	29	0
43	79.27 ± 4.83	66	0	25.20 ± 2.61	20	0	86	170.60 ± 7.64	154	0	78.50 ± 4.69	67	0

Además, como se muestra en tabla 3.12, existen diferencias significativas (estadísticamente hablando) con respecto a las otras propuestas. Esto lo podemos asegurar, debido a que los valores para los tests son muy superiores a los valores críticos, en cada uno de los casos.

Los resultados para el test de Holm la podemos visualizar en la tabla 3.13, en la cual se corrobora la tesis que hemos manejado sobre el rendimiento de estas técnicas, estadísticamente hablando. Aunque debemos aclarar que la técnica basada en ruptura de simetrías (Ts.B*), parece no pasar este tipo de test.

Tabla 3.11: Resultados experimentales del BIBD para los Algoritmos Genéticos (del Modelo DUAL) con 30 ejecuciones. Instancias tomadas de [32, 246]. $\bar{x}, \sigma, B, S$, representan la media, la desviación estándar y el mejor fitness encontrado en las ejecuciones, así como el número de ejecuciones con éxito. Para el Modelo Dual, sin ruptura de simetrías.

ID	GA.D.A2.Ux			GA.D.A2.Gd			ID	GA.D.A2.Ux			GA.D.A2.Gd		
	$\bar{x}\pm\sigma$	B	S	$\bar{x}\pm\sigma$	B	S		$\bar{x}\pm\sigma$	B	S	$\bar{x}\pm\sigma$	B	S
1	0.00 ± 0.00	0	30	0.00 ± 0.00	0	30	44	88.77 ± 5.91	77	0	103.47 ± 4.73	93	0
2	0.00 ± 0.00	0	30	0.00 ± 0.00	0	30	45	10.07 ± 2.64	6	0	14.70 ± 2.24	12	0
3	0.27 ± 1.44	0	29	0.00 ± 0.00	0	30	46	42.77 ± 2.96	37	0	53.50 ± 2.75	46	0
4	0.67 ± 1.49	0	25	0.27 ± 1.00	0	28	47	8.37 ± 2.04	6	0	13.93 ± 2.74	7	0
5	0.00 ± 0.00	0	30	0.00 ± 0.00	0	30	48	15.67 ± 1.72	12	0	22.97 ± 2.96	17	0
6	1.60 ± 1.96	0	18	0.73 ± 1.65	0	25	49	0.10 ± 0.54	0	29	0.83 ± 1.53	0	23
7	0.00 ± 0.00	0	30	0.00 ± 0.00	0	30	50	14.00 ± 2.35	10	0	19.53 ± 2.03	16	0
8	9.37 ± 6.80	0	10	6.13 ± 6.59	0	16	51	4.30 ± 2.45	0	5	7.43 ± 2.14	4	0
9	3.07 ± 2.08	0	9	2.37 ± 2.09	0	13	52	6.50 ± 2.13	0	1	11.33 ± 1.92	7	0
10	8.27 ± 7.46	0	13	4.87 ± 6.42	0	19	53	1.77 ± 1.93	0	16	3.70 ± 1.85	0	5
11	5.40 ± 2.17	0	2	5.17 ± 2.08	0	2	54	34.67 ± 2.97	27	0	47.93 ± 3.30	42	0
12	1.07 ± 1.77	0	22	0.67 ± 1.51	0	25	55	2.97 ± 2.02	0	9	6.17 ± 1.57	4	0
13	6.10 ± 5.45	0	13	6.13 ± 4.98	0	11	56	54.27 ± 3.53	45	0	69.67 ± 4.13	62	0
14	0.40 ± 1.20	0	27	0.00 ± 0.00	0	30	57	10.60 ± 2.06	4	0	16.17 ± 2.71	9	0
15	0.10 ± 0.54	0	29	0.00 ± 0.00	0	30	58	126.90 ± 7.70	107	0	152.23 ± 5.61	137	0
16	4.93 ± 1.39	0	1	5.30 ± 1.59	0	1	59	51.30 ± 3.74	44	0	65.43 ± 3.87	60	0
17	7.27 ± 1.84	4	0	8.13 ± 1.69	4	0	60	1.20 ± 1.72	0	20	3.00 ± 1.59	0	6
18	1.23 ± 1.89	0	21	1.63 ± 1.89	0	17	61	41.13 ± 3.96	35	0	54.67 ± 3.90	42	0
19	34.53 ± 8.91	0	1	41.20 ± 5.92	16	0	62	42.53 ± 2.38	36	0	58.90 ± 3.88	47	0
20	2.93 ± 1.97	0	9	3.57 ± 1.69	0	5	63	12.73 ± 2.35	8	0	21.07 ± 2.35	17	0
21	10.57 ± 1.98	6	0	12.47 ± 1.87	8	0	64	16.43 ± 2.35	12	0	25.37 ± 3.02	20	0
22	15.53 ± 3.05	6	0	18.60 ± 2.62	12	0	65	3.70 ± 1.81	0	5	6.07 ± 1.63	3	0
23	3.33 ± 1.96	0	7	4.40 ± 1.14	0	1	66	4.93 ± 2.34	0	4	9.30 ± 2.02	5	0
24	7.97 ± 10.11	0	18	21.07 ± 10.61	0	4	67	0.47 ± 1.20	0	26	1.47 ± 1.96	0	19
25	0.00 ± 0.00	0	30	0.00 ± 0.00	0	30	68	4.90 ± 2.36	0	3	8.53 ± 2.29	4	0
26	1.47 ± 1.93	0	19	1.57 ± 1.82	0	17	69	3.67 ± 2.13	0	7	6.27 ± 1.79	4	0
27	13.00 ± 1.84	9	0	17.03 ± 2.09	12	0	70	28.87 ± 3.56	22	0	44.77 ± 2.81	38	0
28	17.97 ± 2.23	14	0	23.03 ± 2.32	16	0	71	37.07 ± 3.87	29	0	53.43 ± 4.12	45	0
29	1.77 ± 2.03	0	17	4.23 ± 1.38	0	2	72	51.53 ± 3.51	45	0	71.20 ± 4.18	63	0
30	0.23 ± 0.88	0	28	0.10 ± 0.54	0	29	73	16.20 ± 3.31	10	0	27.37 ± 1.97	24	0
31	3.73 ± 2.53	0	8	5.83 ± 1.51	4	0	74	2.00 ± 1.91	0	14	3.50 ± 1.77	0	5
32	6.03 ± 2.07	0	1	8.17 ± 1.69	4	0	75	14.60 ± 1.94	11	0	22.73 ± 2.68	18	0
33	15.90 ± 1.97	13	0	20.53 ± 2.25	14	0	76	24.50 ± 3.63	18	0	37.10 ± 3.21	30	0
34	11.73 ± 2.11	7	0	15.90 ± 2.41	9	0	77	60.70 ± 4.17	54	0	84.07 ± 4.43	76	0
35	4.73 ± 1.91	0	3	6.53 ± 1.93	0	1	78	11.27 ± 2.26	6	0	19.77 ± 2.56	16	0
36	75.93 ± 5.54	61	0	86.53 ± 4.16	77	0	79	10.33 ± 2.26	4	0	17.60 ± 2.73	11	0
37	0.97 ± 1.62	0	22	2.17 ± 2.08	0	14	80	19.47 ± 2.54	13	0	31.07 ± 3.48	22	0
38	0.13 ± 0.72	0	29	0.10 ± 0.54	0	29	81	92.00 ± 5.81	81	0	115.60 ± 4.79	107	0
39	22.87 ± 2.72	17	0	31.07 ± 1.95	27	0	82	168.13 ± 5.67	159	0	200.47 ± 6.11	183	0
40	0.83 ± 1.53	0	23	1.13 ± 1.77	0	21	83	198.07 ± 7.04	186	0	235.00 ± 6.38	219	0
41	2.27 ± 2.25	0	14	3.53 ± 1.75	0	5	84	4.20 ± 2.44	0	6	7.63 ± 2.12	4	0
42	2.40 ± 2.20	0	13	4.37 ± 1.45	0	2	85	49.67 ± 3.26	42	0	70.73 ± 3.61	63	0
43	27.53 ± 2.17	23	0	37.77 ± 3.01	31	0	86	89.73 ± 4.53	81	0	118.90 ± 5.52	106	0

Tabla 3.12: Resultados para el test de Friedman e Iman-Davenport ($\alpha = 0.05$) para las metaheurísticas de búsquedas locales y algoritmo genético aplicadas sobre el problema del BIBD para los modelos Primal y Dual (Con/Sin Ruptura de simetrías), B identifica el modelo de representación Primal sin ruptura de simetrías, B* modelo Primal con ruptura de simetrías; D el modelo Dual sin ruptura de simetrías, y D* el modelo Dual con ruptura de simetrías.

	Friedman value	Critical χ^2 value	Iman-Davenport value	Critical F_F value
All 24	1380.373256	35.172462	196.329445	1.534780

Figura 3.22: Propuesta básica: Torneo para las diferentes técnicas metaheurísticas aplicadas sobre el problema del BIBD para los modelos Primal y Dual (Con/Sin Ruptura de simetrías), B identifica el modelo de representación Primal sin ruptura de simetrías, B* modelo Primal con ruptura de simetrías; D el modelo Dual sin ruptura de simetrías, y D* el modelo Dual con ruptura de simetrías. Cada caja comprende desde el primer al tercer cuartil de la distribución, la mediana (2do cuartil) es indicada con una línea vertical, la zona sin valores atípicos (whiskers span) es de 1.5 veces el rango intercuartil, además los valores atípicos están marcados con un signo '+'.

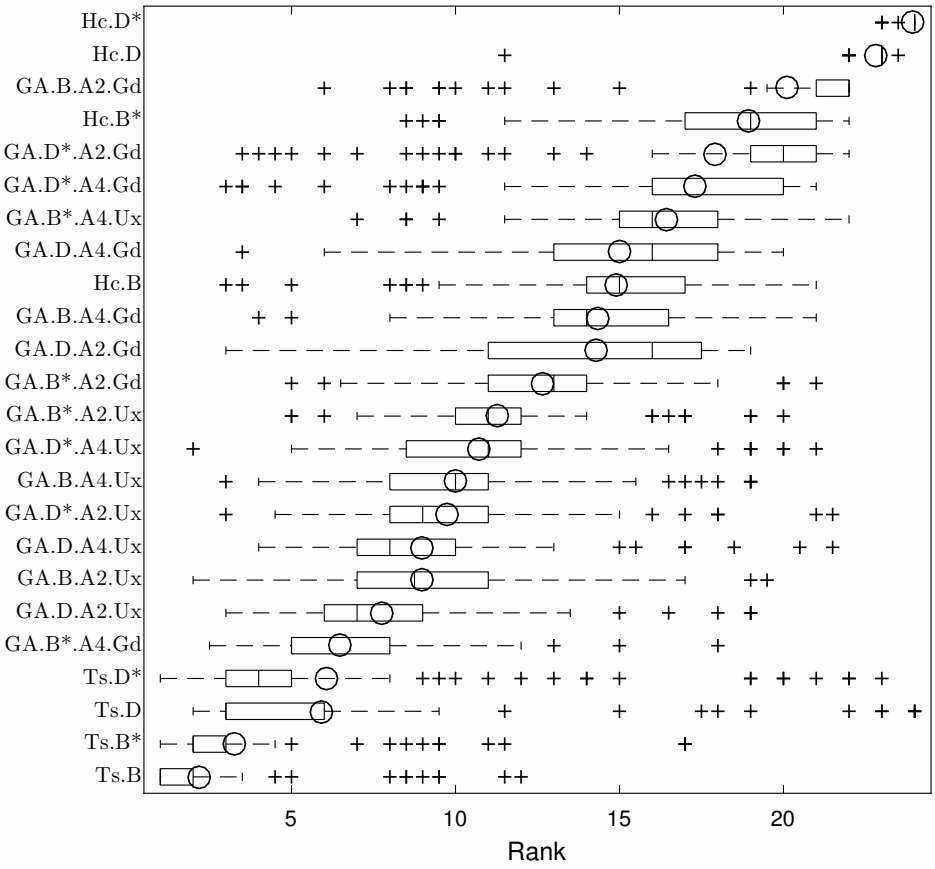


Tabla 3.13: Resultados para el test de Holm (con $\alpha = 0.05$) usando Ts.B como algoritmo de control, para las metaheurísticas de búsquedas locales y algoritmo genético aplicadas sobre el problema del BIBD para los modelos Primal y Dual (Con/Sin Ruptura de simetrías), B identifica el modelo de representación Primal sin ruptura de simetrías, B* modelo Primal con ruptura de simetrías; D el modelo Dual sin ruptura de simetrías, y D* el modelo Dual con ruptura de simetrías.

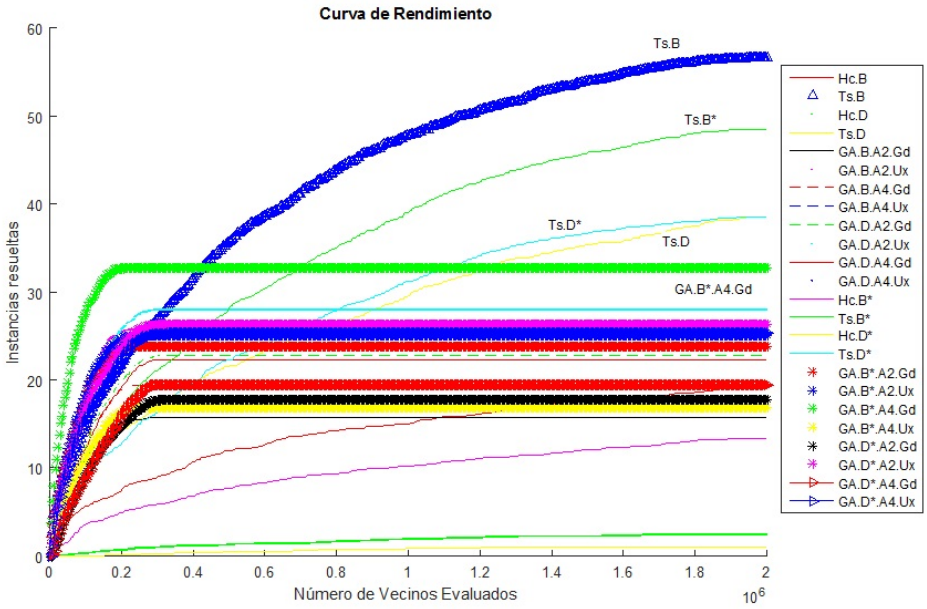
i	strategy	z-statistic	p-value	α/i
1	Ts.B*	9.867e-01	1.619e-01	5.000e-02
2	Ts.D	3.478e+00	2.530e-04	2.500e-02
3	Ts.D*	3.623e+00	1.455e-04	1.667e-02
4	GA.B*.A4.Gd	3.968e+00	3.620e-05	1.250e-02
5	GA.D.A2.Ux	5.187e+00	1.070e-07	1.000e-02
6	GA.B.A2.Ux	6.287e+00	1.622e-10	8.333e-03
7	GA.D.A4.Ux	6.314e+00	1.363e-10	7.143e-03
8	GA.D*.A2.Ux	6.998e+00	1.295e-12	6.250e-03
9	GA.B.A4.Ux	7.241e+00	2.227e-13	5.556e-03
10	GA.D*.A4.Ux	7.904e+00	1.349e-15	5.000e-03
11	GA.B*.A2.Ux	8.433e+00	1.691e-17	4.545e-03
12	GA.B*.A2.Gd	9.705e+00	1.436e-22	4.167e-03
13	GA.D.A2.Gd	1.125e+01	1.199e-29	3.846e-03
14	GA.B.A4.Gd	1.127e+01	9.384e-30	3.571e-03
15	Hc.B	1.182e+01	1.567e-32	3.333e-03
16	GA.D.A4.Gd	1.189e+01	6.362e-33	3.125e-03
17	GA.B*.A4.Ux	1.323e+01	3.118e-40	2.941e-03
18	GA.D*.A4.Gd	1.403e+01	4.798e-45	2.778e-03
19	GA.D*.A2.Gd	1.457e+01	2.237e-48	2.632e-03
20	Hc.B*	1.555e+01	8.019e-55	2.500e-03
21	GA.B.A2.Gd	1.662e+01	2.398e-62	2.381e-03
22	Hc.D	1.913e+01	6.463e-82	2.273e-03
23	Hc.D*	2.020e+01	5.193e-91	2.174e-03

Tabla 3.14: Resumen de resultados del test de significancia estadística para las técnicas metaheurísticas de búsquedas locales y algoritmo genético aplicadas sobre el problema del BIBD para los modelos Primal y Dual (Con/Sin Ruptura de simetrías), B Primal sin ruptura de simetrías, B* Primal con ruptura de simetrías; D Dual sin ruptura de simetrías, y D* Dual con ruptura de simetrías. Cada entrada en la tabla indica el porcentaje de instancias en la cual el algoritmo etiquetado en la fila supera el rendimiento del algoritmo etiquetado en la columna, con una diferencia estadísticamente significativa en el nivel 0.05, según el test de Wilcoxon ranksum.

	Hc.B	Tx.B	Hc.D	Tx.D	GA.B.A2.Gd	GA.B.A2.Ux	GA.B.A4.Gd	GA.B.A4.Ux	GA.D.A2.Gd	GA.D.A2.Ux	GA.D.A4.Gd	GA.D.A4.Ux	Hc.B*	Tx.B*	Hc.D*	Tx.D*	GA.B*.A2.Gd	GA.B*.A2.Ux	GA.B*.A4.Gd	GA.B*.A4.Ux	GA.D*.A2.Gd	GA.D*.A2.Ux	GA.D*.A4.Gd	GA.D*.A4.Ux
Hc.B	—	0.00%	97.67%	8.14%	76.74%	1.16%	5.81%	2.33%	34.88%	2.33%	37.21%	2.33%	74.42%	1.16%	98.84%	2.33%	5.81%	2.33%	2.33%	24.42%	62.79%	2.33%	62.79%	3.49%
Tx.B	88.37%	—	97.67%	61.63%	84.88%	73.26%	83.72%	77.91%	75.58%	72.09%	77.91%	75.58%	89.53%	44.19%	98.84%	60.47%	76.74%	74.42%	68.60%	84.88%	82.56%	73.26%	80.23%	75.58%
Hc.D	1.16%	1.16%	—	4.65%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	95.35%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Tx.D	73.26%	12.79%	93.02%	—	75.58%	54.65%	70.93%	54.65%	65.12%	53.49%	65.12%	56.98%	79.07%	1.16%	94.19%	30.23%	65.12%	61.63%	51.16%	74.42%	72.09%	60.47%	69.77%	63.95%
GA.B.A2.Gd	3.49%	5.81%	97.67%	9.30%	—	0.00%	1.16%	0.00%	0.00%	0.00%	0.00%	0.00%	12.79%	0.00%	98.84%	3.49%	2.33%	1.16%	0.00%	3.49%	1.16%	2.33%	1.16%	2.33%
GA.B.A2.Ux	77.91%	13.95%	98.84%	18.60%	82.56%	—	69.77%	4.65%	61.63%	3.49%	63.95%	4.65%	87.21%	2.33%	98.84%	10.47%	54.65%	36.05%	8.14%	75.58%	75.58%	15.12%	73.26%	29.07%
GA.B.A4.Gd	23.26%	4.65%	98.84%	9.30%	77.91%	5.81%	—	0.00%	40.70%	1.16%	45.35%	0.00%	72.09%	0.00%	98.84%	2.33%	6.98%	4.65%	2.33%	36.05%	66.28%	1.16%	65.12%	2.33%
GA.B.A4.Ux	76.74%	11.63%	98.84%	20.93%	83.72%	3.49%	69.77%	—	60.47%	0.00%	62.79%	1.16%	88.37%	0.00%	98.84%	6.98%	53.49%	25.58%	4.65%	75.58%	74.42%	11.63%	70.93%	19.77%
GA.D.A2.Gd	15.12%	11.63%	98.84%	12.79%	84.88%	6.98%	12.79%	8.14%	—	2.33%	8.14%	1.16%	55.81%	1.16%	98.84%	4.65%	5.81%	3.49%	4.65%	22.09%	73.26%	3.49%	66.28%	4.65%
GA.D.A2.Ux	80.23%	15.12%	98.84%	19.77%	83.72%	17.44%	76.74%	20.93%	72.09%	—	68.60%	8.14%	87.21%	2.33%	98.84%	5.81%	62.79%	46.51%	5.81%	79.07%	74.42%	25.58%	72.09%	34.88%
GA.D.A4.Gd	11.63%	10.47%	98.84%	15.12%	83.72%	3.49%	8.14%	5.81%	1.16%	3.49%	—	0.00%	48.84%	1.16%	98.84%	4.65%	4.65%	2.33%	2.33%	15.12%	63.95%	2.33%	53.49%	3.49%
GA.D.A4.Ux	75.58%	12.79%	98.84%	19.77%	83.72%	4.65%	68.60%	3.49%	63.95%	3.49%	68.60%	—	88.37%	1.16%	98.84%	6.98%	58.14%	33.72%	5.81%	75.58%	75.58%	11.63%	72.09%	22.09%
Hc.B*	6.98%	2.33%	98.84%	10.47%	61.63%	2.33%	2.33%	2.33%	17.44%	2.33%	18.60%	3.49%	—	0.00%	98.84%	0.00%	0.00%	0.00%	0.00%	0.00%	34.88%	0.00%	31.40%	0.00%
Tx.B*	86.05%	10.47%	98.84%	55.81%	88.37%	82.56%	86.05%	87.21%	83.72%	82.56%	86.05%	83.72%	91.86%	—	98.84%	51.16%	76.74%	73.26%	67.44%	84.88%	82.56%	73.26%	80.23%	75.58%
Hc.D*	1.16%	1.16%	2.33%	4.65%	1.16%	1.16%	1.16%	1.16%	1.16%	1.16%	1.16%	1.16%	1.16%	1.16%	—	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Tx.D*	84.88%	9.30%	97.67%	25.58%	87.21%	70.93%	79.07%	74.42%	80.23%	70.93%	83.72%	72.09%	90.70%	8.14%	98.84%	—	68.60%	67.44%	48.84%	76.74%	77.91%	61.63%	74.42%	62.79%
GA.B*.A2.Gd	45.35%	10.47%	98.84%	17.44%	80.23%	3.49%	39.53%	1.16%	50.00%	1.16%	53.49%	0.00%	91.86%	10.47%	100.00%	11.63%	—	0.00%	0.00%	62.79%	69.77%	0.00%	67.44%	1.16%
GA.B*.A2.Ux	68.60%	13.95%	98.84%	17.44%	83.72%	1.16%	56.98%	2.33%	60.47%	3.49%	63.95%	0.00%	91.86%	13.95%	100.00%	18.60%	20.93%	—	0.00%	75.58%	73.26%	1.16%	68.60%	3.49%
GA.B*.A4.Gd	81.40%	9.30%	98.84%	17.44%	84.88%	40.70%	73.26%	51.16%	70.93%	24.42%	75.58%	38.37%	91.86%	10.47%	100.00%	17.44%	75.58%	63.95%	—	77.91%	72.09%	41.86%	70.93%	48.84%
GA.B*.A4.Ux	5.81%	8.14%	98.84%	15.12%	73.26%	6.98%	1.16%	3.49%	36.05%	3.49%	37.21%	5.81%	65.12%	8.14%	100.00%	10.47%	3.49%	6.98%	9.30%	—	55.81%	0.00%	53.49%	0.00%
GA.D*.A2.Gd	6.98%	6.98%	97.67%	11.63%	81.40%	3.49%	1.16%	2.33%	0.00%	3.49%	1.16%	2.33%	32.56%	6.98%	100.00%	9.30%	5.81%	2.33%	5.81%	8.14%	—	1.16%	0.00%	1.16%
GA.D*.A2.Ux	80.23%	13.95%	98.84%	20.93%	84.88%	6.98%	4.65%	4.65%	63.95%	0.00%	68.60%	3.49%	91.86%	13.95%	100.00%	12.79%	45.35%	24.42%	10.47%	80.23%	75.58%	—	72.09%	8.14%
GA.D*.A4.Gd	3.81%	5.81%	97.67%	11.63%	81.40%	4.65%	4.65%	5.81%	0.00%	4.65%	1.16%	3.49%	34.88%	5.81%	100.00%	8.14%	6.98%	4.65%	5.81%	11.63%	3.49%	4.65%	—	1.16%
GA.D*.A4.Ux	79.07%	12.79%	98.84%	17.44%	82.56%	8.14%	63.95%	8.14%	61.63%	1.16%	66.28%	3.49%	89.53%	12.79%	100.00%	9.30%	37.21%	15.12%	12.79%	77.91%	76.74%	1.16%	75.58%	—

La tabla 3.14 presenta una prueba, basada en suma de rangos. En esta tabla se está mostrando el porcentaje de instancias en las que un determinado algoritmo tiene un mejor rendimiento (considerando una significación estadística en el nivel 0.05, según una prueba de Wilcoxon ranksum) que otro algoritmo, debemos tomar en cuenta que las entradas (i, j) y (j, i) en esta tabla no necesariamente suman 100%, ya que hay casos en los que no hay una diferencia significativa entre los algoritmos comparados. Nótese, por ejemplo, que la técnica *Ts.B*, es un 61% significativa para más del 95% de los casos, estadísticamente hablando.

Figura 3.23: Problema del BIBD: Curva de rendimiento de las diferentes técnicas metaheurísticas aplicadas (HC,TS y GA) para los modelos Primal y Dual (Con/Sin Ruptura de simetrías), B Primal sin ruptura de simetrías, B* Primal con ruptura de simetrías; D Dual sin ruptura de simetrías, y D* Dual con ruptura de simetrías.



La figura 3.23 representa la curva de rendimiento, considerando el número de instancias resueltas, con respecto al número de evaluaciones ejecutadas por cada algoritmo, en términos de la función objetivo. Como podemos apreciar, la técnica que parece permanecer mucho más activa (encuentra más soluciones en función del tiempo de ejecución) es TS.B. Lo que no sorprende, dado los análisis que ya se han aplicado anteriormente en esta misma sección.

3.4 Técnicas Metaheurísticas para Resolver el TDP

El conjunto de técnicas metaheurísticas que han sido aplicadas para atacar el problema del BIBD se han empleado de forma similar en el problema del TDP (véase la sección 3.2.8).

Tabla 3.15: Resolución del TDP por medio de un modelo ILP [251]: Resultados

Problema	T	Overall % Dev.	Max. % Dev.
Cat Food Cartoon	3	−2.59	−7.27
Herbs Cartoon	3	−2.91	−8.70
Magazine Inserts	4	−2.49	−10.00

3.4.1 Técnicas Trayectoriales

Consideramos dos técnicas de búsqueda local, Hill Climbing y Tabu Search, que funcionan con base a los procesos y parámetros previamente descritos en la sección 3.2.8.2. Debemos aclarar, que para el caso de la representación alternativa que hemos llamado DUAL, los algoritmos empleados en las diferentes metaheurísticas aplicadas funciona de forma idéntica que el descrito para el modelo que hemos llamado PRIMAL, con la salvedad, claro está, que la exploración de los vecindarios varía de acuerdo al modelo.

3.4.2 Técnica Poblacional

Hemos considerado utilizar, al igual que para el problema del BIBD (véase la sección anterior 3.2.8.3), un algoritmo genético de estado estacionario [18] (GA). El proceso consiste en generar una población de posibles soluciones candidatas, vía un algoritmo guiado, para realizar las etapas de seleccionan, mutación y reemplazan iterativamente. Para ser precisos, la selección se realiza mediante torneos binarios, y el reemplazo se realiza siguiendo la política–$(\mu, 1)$, es decir, se genera un nuevo individuo y se inserta en la población reemplazando al que contenga el menor fitness. En cuanto a la generación de los descendientes, se realiza por recombinación y mutación, como de costumbre. El operador de recombinación utilizado es una variante del cruce uniforme (UX) [295]: primero, las plantillas se comparan para encontrar la mejor coincidencia entre ellas (hay que tener presente la existencia de las simetrías del problema y el hecho de que las plantillas se pueden reordenar arbitrariamente) . Posteriormente, se realiza un intercambio de información a nivel de plantilla, eligiendo aleatoriamente una plantilla de cada par de plantillas combinadas (una de cada uno de los padres). Evidentemente esto equivale a un intercambio de columnas en la matriz $M = \{p_{ij}\}$. La mutación se maneja de la misma manera que en el vecindario que ya ha sido descrito anteriormente, para este problema.

3.4.3 Resultados Experimentales para el TDP

Para la realización de las pruebas se han empleado tres escenarios de complejidad creciente [251]. La tabla 3.15 muestra los resultados obtenidos por Proll y Smith considerando su modelo ILP (tabla 3 y ecuaciones 7-10 en [251]). *Max % Dev* Se refiere a la máxima desviación para un diseño determinado (un valor negativo indica la infraproducción y un valor positivo la sobreproducción), y *Overall Dev.* a la media de las desviaciones. La tabla 3.16 muestra los mejores resultados obtenidos por Proll y Smith al resolver el TDP como un problema de satisfacción con restricciones (CSP), tabla 7 en [251].

En las pruebas ejecutadas hemos considerado los escenarios y demandas empleadas por Proll y Smith, las cuales se muestran en la tabla 3.17.

Tabla 3.16: Resolución del TDP por medio de un modelo CSP [251]: Resultados

Problema	T	Overall % Dev.	Max. % Dev.
Cat Food Cartoon	3	-1.4	-1.9
Herbs Cartoon	2	-3.2	$+10/-6.7$
Magazine Inserts	3	-2.4	$+7.4/-8.3$

Tabla 3.17: Demanda por cada escenario, tomado de Proll y Smith [251].

Problema	Slots por Plantilla	Variaciones	Cantidades demandadas (x1000)
Cat Food Cartoon	9	7	250,255,260,500,500,800,1100
Herbs Cartoon	42	30	60,60,70,70,70,70,70,70,70,80, 80,80,80,90,90,90,90,90,90,100, 100,100,100,150,230,230,230,230,280,280
Magazine Inserts	40	50	50,53,55,60,85,90,100,100,105,110, 137,140,140,140,150,150,150,150,150,150, 150,150,168,170,170,195,195,200,200,200, 210,210,225,230,230,230,250,250,250,250, 250,250,250,250,265,270,270,375,375,405

3.4.3.1 Convenciones de Notación para las Técnicas Metaheurísticas del TDP

Cada una de las técnicas en particular se identificará a través de una secuencia de identificadores separados por un punto. Para el caso particular de las metaheurísticas básicas (como se describe en 3.4.1) las cuales corresponden a Hill Climbing (HC), búsqueda tabú (TS), algoritmo Genético (GA). Además, para los métodos basados en la población (es decir, GA), el procedimiento de recombinación se caracteriza por el operador particular utilizado –nos enfocamos aquí en el uso del operador de cruce voraz (Gd), para el caso de la representación alternativa Primal y Cruce estándar para la representación Dual – y por su aridad, es decir, el número m de padres utilizados (denotados como Am).

Además, utilizamos un símbolo de asterisco ($*$) para indicar el uso de las configuraciones para la ruptura de simetría, una letra P para indicar que los individuos están codificados en una representación *primal* (es decir, el modelo alternativo basado en *diseños*, véase la sección 3.2.9.2) y una letra D para indicar que están codificados en una representación *dual* (es decir, la formulación basada en *slots*).

Consideremos un ejemplo para estos algoritmos no-colaborativos: $Hc.P$ (respecto a $Hc.P*$) denota la técnica Hill Climbing, la cual ha sido implementada usando el modelo primal (en representación basada en Diseños) y el esquema sin ruptura de simetrías (respeto a con ruptura); $Ts.D*$ (respecto de $Ts.D$), el cual denota la técnica basada en búsqueda tabú, implementada con el modelo Dual (representación basada en Slots) y el esquema con ruptura de simetrías (respecto a la configuración sin-ruptura); de igual forma, $GA.P*.A2.Gd$ denota un algoritmo genético con cruce de 2 padres y operador voraz; esta propuesta hace uso de la codificación basada en Diseños (modelo primal) con una configuración que reduce las simetrías (con ruptura de simetrías); y para el caso de $GA.D.A4.Gd$ representa un algoritmo genético con cruce de 4 padres, operador Voraz, con formulación dual y sin ruptura de simetrías.

Tabla 3.18: Resultados obtenidos para el TDP de los escenarios tomados de Proll y Smith [251], aplicando Hill Climbing (20 ejecuciones por instancia). 10% de sobre/infra-producción y con inicialización aleatoria. Modelo Primal, sin ruptura de simetrías.

Problema	T	Mejor solución obtenida	Pressings	Overall Dev.	Max.Dev
Cat Food	2	[0,0,0,0,0,2,7]	157143		
		[1,1,1,2,2,2,0]	250000	0.80	-3.85/ 1.79
	3	[1,1,1,2,0,2,2]	250000		
		[0,0,0,0,3,2,4]	150000		
		[0,1,1,0,7,0,0]	7143	0.14	-1.10/ 0.84
Hb.Cartoon	2	[1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,1,1,1,2,3,2,2,3,4,3]	70000		
		[0,4,0,0,0,0,0,0,0,1,1,1,1,2,1,2,1,1,0,2,2,2,1,2,6,6,1,0,4]	15000	4.00	-5.56/ 40.00
	3	[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,3,2,2,2,4,4]	66000		
		[0,0,2,2,2,2,2,2,2,0,0,0,0,4,5,4,3,0,4,1,1,1,1,1,0,1,1,1,0,0]	2000		
		[0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,2,1,2,2,2,2,1,2,6,6,6,1,1]	16000	0.91	-2.22/ 10.00
Mz.Inserts	3	[1,1,0,1,1,0,1,1,0,0,0,0,0,0,0,0,0,2,0,0,1,0,1,2,0,0,1,1,2,1,0,2,1,1,3,0,2,2,1]	90000		
		[0,0,1,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,1,1,0,1,0,0,1,2,0,1,0,1,1,0,1,1,2,1,2,2,1,2,0,1,1,1,1,0,2,1,1,2]	118500		
		[0,0,0,0,0,3,0,0,3,0,1,1,1,1,1,1,1,1,0,1,1,2,1,2,0,2,0,3,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,1,1,2,0,1,3,2,2]	31500	7.17	-10.00/ 115.45
	4	[0,0,0,0,0,0,0,1,1,0,1,1,2,0,0,0,1,0,0,1,1,1,0,2,1,3,0,0,0,1,0,2,1,2,0,0,1,2,2,2,0,1,0,2,0,0,2,2,1,3]	70000		
		[0,0,0,0,1,1,1,0,0,0,1,1,0,1,1,1,0,1,1,1,0,0,1,0,1,0,1,1,1,1,1,0,1,0,2,2,2,1,0,0,1,1,2,0,3,2,0,1,2,2]	90000		
		[0,3,0,0,0,0,1,1,0,0,0,0,0,0,3,0,1,0,3,0,4,1,0,1,0,0,0,3,2,0,1,0,2,0,2,0,1,0,3,2,1,1,0,0,2,1,1,0,0]	20000		
		[1,0,1,1,0,0,0,0,1,2,0,0,0,1,0,1,1,1,0,0,0,1,2,0,0,0,2,2,1,0,2,1,1,1,1,0,0,0,2,1,2,1,1,2,0,1,2,2,2,0]	55000	3.76	-10.00/ 19.05

3.4.3.2 Resultados para el Modelo Primal del TDP: Sin Ruptura de Simetrías

Las tablas 3.18 y 3.19 muestran los resultados con inicialización aleatoria para las soluciones candidatas. Los resultados para los casos de inicialización heurística se presentan en las tablas 3.20 y 3.21. Los algoritmos han sido ejecutados utilizando los siguientes porcentajes de vecindarios: 50% para Cat Food, 5% y 10% para Herbs Cartoon, 1% y 5% para Magazine Inserts. Los resultados mostrados corresponden a los mejores resultados obtenidos en las diferentes pruebas realizadas. En general podemos observar que para el caso del primer escenario todas las versiones de los algoritmos muestran un rendimiento muy bueno para esta instancia ya que obtienen la mejor solución de forma consistente, superando los resultados mostrados por Proll y Smith; sin embargo, la versión HC con inicialización heurística obtiene con mayor frecuencia la mejor solución, con lo que podemos considerarla como la mejor opción.

Para el caso de Herbs, la mejor solución es obtenida por la versión TS con inicialización aleatoria y con una exploración del 5% del vecindario, aunque los resultados son parecidos a la versión de inicialización heurística; igualmente la mejor solución aquí encontrada supera a la mostrada por Proll y Smith. Para Magazine sólo se logró encontrar soluciones para el escenario con 4 plantillas con la

Tabla 3.19: Resultados obtenidos para el problema del TDP de los escenarios tomados de Proll y Smith [251], al aplicar Tabu Search (20 ejecuciones por instancia). 10% de sobre/infra-producción y con inicialización aleatoria. Modelo Primal, sin ruptura de simetrías.

Problema	T	Mejor solución obtenida	Pressings	Overall Dev.	Max.Dev
Cat Food	2	[1,1,1,2,2,2,0]	250000		
		[0,0,0,0,0,0,2,7]	157143	0.80	-3.85/ 1.79
	3	[0,0,0,0,0,0,2,7]	150000		
		[1,1,1,2,2,2,0]	250000		
		[0,1,1,0,0,0,7]	7143	0.14	-1.10/ 0.84
Hb.Cartoon	2	[1,1,1,1,1,1,0,1,1,1,1,1,1,1,1, 1,1,0,1,1,1,1,1,2,3,3,3,3,4,3]	65500		
		[0,0,0,0,0,0,4,0,0,1,1,1,1,1,2, 1,1,5,1,2,2,2,2,1,2,2,2,2,1,5]	16750	2.97	-8.61/ 10.00
	3	[1,1,1,1,1,1,0,1,1,0,1,0,0,1,0, 1,1,1,2,1,2,1,1,1,3,2,4,4,4,4]	49500		
		[1,1,2,0,2,0,0,0,2,1,1,1,1,2,2, 1,1,2,0,0,0,2,0,3,3,4,3,3,3,1]	10250		
		[0,0,0,1,0,1,3,1,0,3,1,3,3,1,3, 1,1,1,0,2,0,1,2,3,2,4,0,0,2,3]	23250	2.11	-7.78/ 10.00
Mz.Inserts	3	[0,0,0,0,0,0,0,1,1,1,0,0,0,1,0,0,1,1,0,0,0,0,1,0,2,1, 2,1,1,1,0,0,0,2,2,1,1,2,1,1,2,0,2,2,2,0,1,1,2,1,2]	100000		
		[1,1,1,0,2,2,0,0,0,2,1,1,1,1,1,1,1,1,3,1,1,1,0,0,0, 0,0,0,2,1,1,1,1,1,1,1,1,0,3,1,0,1,1,1,0,0,0,0,0,0]	50000		
		[0,0,0,1,0,0,0,0,0,0,1,1,0,1,1,0,0,1,0,1,1,0,2,0,1, 0,1,1,0,2,2,2,0,0,1,1,0,2,0,0,3,0,0,0,3,2,2,2,3,2]	85000	4.51	-10.00/ 41.67
	4	[0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,2,0,0,2,0,2,0,0,1, 1,2,0,1,0,0,1,1,2,0,3,2,2,1,1,1,1,1,1,2,0,0,4,3,1]	79750		
		[1,1,0,1,0,0,1,0,1,0,0,1,0,0,2,0,0,2,0,0,1,0,0,2,0, 0,0,0,2,3,2,0,1,0,0,0,0,0,0,3,1,1,1,3,1,1,5,1,0,0,2]	54000		
		[0,0,0,0,0,2,0,1,1,0,2,2,2,3,1,0,0,1,2,0,1,0,1,0,1, 0,1,2,0,1,1,0,1,2,0,0,2,2,0,0,0,0,0,0,1,0,1,0,3,3]	44000		
		[0,0,1,0,0,0,1,1,0,2,1,0,1,0,0,1,0,0,1,0,1,0,2,1,1, 2,0,2,0,0,1,2,1,0,4,0,0,0,0,2,2,2,0,2,0,0,3,1,0,2]	56000	3.38	-10.00/ 10.00

versión del algoritmo TS con inicialización heurística con una exploración del 5% del vecindario.

Refiriéndonos a las técnicas trayectoriales, podemos indicar, que los porcentajes de soluciones factibles para cada uno de los escenarios se pueden resumir como:

- Cat Food: Se obtienen más de 85% de soluciones factibles en todas las versiones de los algoritmos utilizados con el escenario de 2 plantillas y un 100% para 3 plantillas.
- Herbs Cartoon: Para 2 plantillas el mejor comportamiento lo tiene TS con inicialización aleatoria y con exploración del 10% del vecindario, alcanzando un 15% de soluciones factibles. Para el caso de 3 plantillas se alcanza un 95% de factibilidad con Tabu Search que usa inicialización heurística y una exploración del 5% del vecindario.
- Magazine Inserts: Ninguna de nuestras propuestas alcanza soluciones factibles para 3 plantillas, pero se logra un máximo de 15% de factibilidad para 4 plantillas con TS con inicialización heurística y un 5% de exploración del vecindario.

Tabla 3.20: Resultados obtenidos para el problema del TDP de los escenarios tomados de Proll y Smith [251], al aplicar Hill Climbing (20 ejecuciones por instancia). 10% de sobre/infra-producción y con inicialización heurística. Modelo Primal, sin ruptura de simetrías.

Problema	T	Mejor solución obtenida	Pressings	Overall Dev.	Max.Dev
Cat Food	2	[0,0,0,0,0,2,7]	157143		
		[1,1,1,2,2,2,0]	250000	0.80	-3.85/ 1.79
	3	[1,1,1,0,0,2,4]	233333		
		[0,0,0,3,3,2,1]	166667		
		[2,3,4,0,0,0,0]	7222	0.12	-0.89/ 0.85
Hb.Cartoon	2	[1,0,1,1,1,1,1,1,1,1,1,1,0,1,1, 1,1,1,1,2,1,1,1,2,2,3,2,3,4,4]	70000		
		[0,4,0,0,0,0,0,0,0,1,1,1,5,1,2, 1,1,1,2,0,2,2,2,0,6,2,6,1,1,0]	15000	4.29	-6.67/ 40.00
	3	[1,1,2,0,0,2,5,2,0,3,3,1,0,2,0, 0,2,2,2,0,0,0,0,0,2,3,1,4,1,3]	10167		
		[1,1,1,0,0,1,0,1,0,1,1,0,1,1,2, 2,1,1,1,2,1,2,2,2,4,3,3,1,3,3]	49500		
		[0,0,0,3,3,0,1,0,3,0,0,3,1,1,0, 0,1,1,1,0,2,0,0,2,0,2,3,6,5,4]	23334	2.41	-8.96/ 10.00
Mz.Inserts	3	[0,1,0,1,0,0,0,2,0,2,0,1,0,0,0,0,0,0,1,0,0,1,1,1,0, 3,0,1,0,1,1,2,1,0,1,0,2,0,0,3,1,2,1,2,0,0,0,1,3,4]	57044		
		[1,0,2,0,2,0,2,0,0,0,0,2,0,0,0,1,0,0,0,3,0,2,0,0,1, 1,1,0,1,0,1,2,1,2,1,2,3,0,0,2,1,0,1,0,0,0,3,1,1,0]	45304		
		[0,0,0,0,0,1,0,0,1,0,1,0,1,1,1,1,1,1,1,0,1,0,1,1,1, 0,1,1,1,1,1,0,1,1,1,1,0,2,2,0,1,1,1,1,2,2,1,2,1,1]	136324	7.11	-10.00/ 64.74
	4	[1,0,0,1,0,2,0,0,1,0,0,3,0,0,0,2,0,0,1,1,1,0,1,1,0, 2,1,0,2,1,3,0,0,1,2,0,1,0,0,2,0,0,1,0,0,3,2,1,2,1]	45455		
		[0,0,1,0,0,0,1,1,0,1,1,0,1,1,1,0,1,1,1,1,0,1,1,0,1, 1,0,2,1,1,0,0,2,1,0,0,1,1,1,1,2,2,1,1,2,0,0,1,2,2]	109091		
		[0,0,0,1,1,0,0,0,0,0,1,0,1,1,0,0,0,2,0,0,2,2,1,0,0, 0,2,0,0,2,0,1,0,1,1,2,2,1,1,2,1,1,2,1,2,1,0,2,2,1]	22727		
		[0,1,0,0,1,0,0,0,1,0,0,0,0,0,1,1,1,0,0,0,1,0,0,2,1, 0,2,0,0,0,1,3,0,1,2,3,1,2,2,0,0,0,1,2,0,2,3,3,0,2]	59091	3.77	-9.09/ 98.35

3.4.3.3 Resultados para el TDP: Primal, Dual, con/sin Ruptura de Simetrías

Realizamos 20 ejecuciones por cada instancia del problema. El número de evaluaciones para cada instancia está sujeto al número de variaciones y número de plantillas, según el escenario abordado: $n_{\mathrm{v}} = 1000 \cdot T \cdot V \cdot (V-1) \cdot \%_v$, donde $\%_v$ representa la muestra porcentual de vecinos a evaluar pues una evaluación completa es tremendamente costosa (e.g., para el escenario más complejo, *Magazine Inserts* con $V = 50$ y $T = 4$, evaluar el 5% de la vecindad significa evaluar $4,9 \cdot 10^5$ vecinos). Se puede observar que para los casos de los escenarios *Herbs Cartoon* y *Magazine Inserts* el vecindario es de tamaño considerable lo que supone un elevado esfuerzo computacional para una exploración exhaustiva, por esta razón hemos utilizado una porción del vecindario para su evaluación. Las pruebas realizadas apuntan hacia la consideración de un número de vecinos no mayor a 500, para el escenario más complejo. Para el caso de TS la intensificación se realiza tras $n_{\mathrm{t}} = n_{\mathrm{v}}/10$ número de evaluaciones sin mejora.

Otro aspecto a resaltar, para la realización de las pruebas experimentales aplicadas sobre el problema del TPD, ha sido que hemos considerado 24 algoritmos basados en técnicas metaheurísticas, de los cuales han resultado 8 algoritmos de búsquedas locales; resultantes de la combinación de: (a) 2

Tabla 3.21: Resultados obtenidos para el problema del TDP de los escenarios tomados de Proll y Smith [251], al aplicar Tabu Search (20 ejecuciones por instancia). 10% de sobre/infra-producción y con inicialización heurística. Modelo Primal, sin ruptura de simetrías.

Problema	T	Mejor solución obtenida	Pressings	Overall Dev.	Max.Dev
Cat Food	2	[0,0,0,0,0,2,7]	157143		
		[1,1,1,2,2,2,0]	250000	0.80	-3.85/ 1.79
	3	[0,1,1,0,0,7,0]	7143		
		[1,1,1,2,2,0,2]	250000		
		[0,0,0,0,0,5,4]	150000	0.14	-1.10/ 0.84
Hb.Cartoon	2	[1,1,1,1,1,1,1,1,1,1,1,1,1,1,0, 1,0,1,1,1,0,1,1,2,3,3,4,3,3,4]	63250		
		[0,0,0,0,0,0,0,0,0,1,1,1,1,1,5, 1,5,1,1,2,5,2,2,1,2,2,0,2,5,1]	18375	4.41	-9.64/ 10.00
	3	[1,1,2,0,0,2,0,2,2,3,0,3,0,1,0, 1,1,1,1,2,4,1,1,1,5,1,1,1,3,1]	13334		
		[2,0,0,1,1,0,1,0,0,0,0,0,3,1,0, 1,3,1,3,1,0,0,0,2,1,0,5,5,4,7]	26000		
		[0,1,1,1,1,1,1,1,1,1,2,1,0,1,2, 1,0,1,0,1,1,2,2,2,3,5,2,2,3,2]	43333	2.25	-8.15/ 8.89
Mz.Inserts	3	[0,0,0,0,0,1,1,1,0,0,0,1,0,0,1,0,0,1,1,0,0,1,0,1,0, 2,2,0,0,0,1,2,2,0,0,1,0,0,2,0,2,0,1,1,2,2,2,3,3,3]	97222		
		[0,0,0,0,1,0,0,0,0,0,1,0,1,1,0,1,2,0,0,2,2,0,2,0,2, 0,0,1,1,1,0,0,0,2,0,1,3,3,0,3,0,3,2,1,0,1,1,1,1,0]	83334		
		[1,1,1,1,0,0,0,0,2,2,1,1,1,1,1,1,0,1,1,0,0,1,0,2,0, 0,0,2,2,2,2,0,1,1,4,1,0,0,1,0,1,0,0,1,1,0,0,0,0,2]	56389	3.29	-7.41/ 23.53
	4	[1,1,0,0,0,0,0,2,0,0,0,2,1,1,1,1,3,0,1,2,0,0,0,0,0, 0,0,1,2,1,0,0,0,1,0,0,0,2,1,0,0,0,3,0,0,4,0,1,1,7]	53571		
		[0,0,1,1,1,0,0,0,0,2,0,0,1,1,0,0,0,1,0,1,0,0,3,2,2, 3,0,1,0,1,0,1,0,1,0,0,0,0,0,1,4,0,0,4,3,1,0,1,3,0]	55000		
		[0,0,0,0,0,1,1,0,1,0,1,0,0,0,1,1,0,1,1,0,1,1,0,0,0, 0,2,1,1,0,2,1,2,1,2,2,2,1,1,2,0,2,1,0,1,0,3,2,1,0]	96429		
		[0,0,0,0,1,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,2,2,0,2,2, 1,0,0,0,3,1,2,1,1,1,1,2,1,3,0,1,2,0,1,0,0,0,3,2,1]	30000	2.36	-8.33/ 8.09

representaciones alternativas (llamados modelo Primal –*P*– y Dual –*D*–), (b) 2 esquemas basados en la ruptura, no-ruptura de simetrías y (c) 2 métodos trayectoriales – HC y TS –. En otras palabras, tendríamos $2 \times 2 \times 2 = 8$; y un total de 16 versiones para los algoritmos poblacionales (GA), resultantes de las combinaciones basadas en los literales (a) y (b), junto con dos tipos de operadores de cruce –Ux/Gd–, así como la recombinación de 2 o 4 progenitores ($2 \times 2 \times 2 \times 2 = 16$). Los resultados de rendimiento obtenidos por todas estas metaheurísticas se muestran en la tabla 3.22.

Dicha tabla muestra, en las columnas desde la dos a la cuatro, y para cada uno de estos 24 algoritmos básicos (identificados en la primera columna), la cantidad de veces que se ha encontrado una solución factible para el escenario abordado del problema en sus 20 ejecuciones. Entre paréntesis, también se muestra el porcentaje de éxito correspondiente, respecto al número de ejecuciones. Hemos considerado que una solución es factible, de acuerdo con cada escenario, si cumple con las restricciones $\sum p_{p_{ij}j} R_j \geq (1 - l_i) Q_i$ and $\sum p_{p_{ij}j} R_j \leq (1 + u_i) Q_i$, de modo que la desviación total está definida por el umbral establecido. En general, según lo que podemos observar en la figura 3.24, las variantes TS, HC y GA funcionan muy bien en el caso de menor complejidad. Además, a primera vista, los algoritmos *Hc.D** funcionan mejor que sus contrapartes de búsqueda local (es decir, TS). Más específicamente, el algoritmo GA que trabaja en el modelo Primal con una recombinación de 4 padres (y

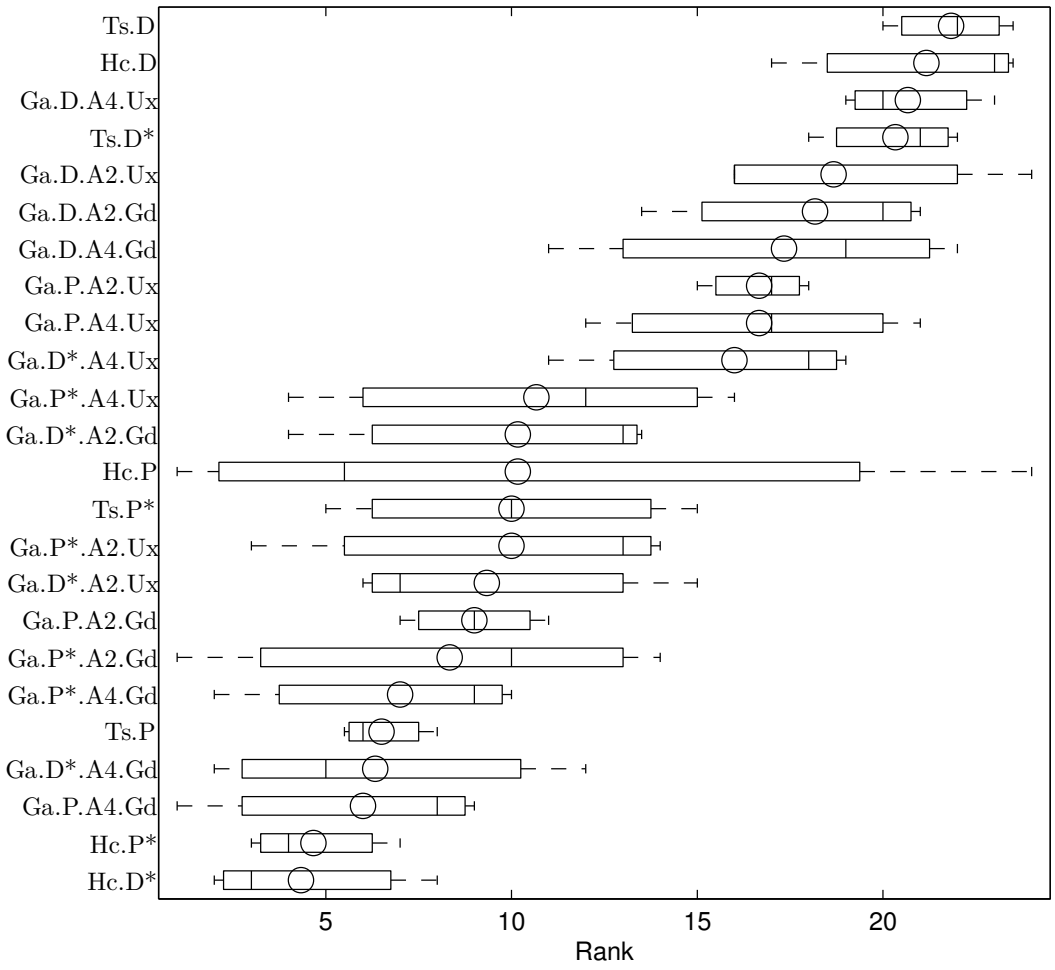
Tabla 3.22: Propuestas básicas para el TDP: Número (y porcentaje) de soluciones factibles encontradas por los algoritmos de corte básico actuando sobre el conjunto de instancias tomadas de [251]. Las filas están ordenadas de acuerdo al torneo asignado para cada algoritmo (se muestra en la penúltima columna). En la última columna el símbolo ✗ indica los algoritmos que no son dominados por otra técnica. Para los dos modelos Primal/Dual, con/sin ruptura de simetrías, P Primal sin ruptura de simetrías, P* Primal con ruptura de simetrías; D Dual sin ruptura de simetrías, y D* Dual con ruptura de simetrías.

Metaheuristics	Cat Food	Herbs C.	Magazine I.	Average ranking	N.D
Hc.D*	**20 (100.00 %)**	1 (5.00 %)	0 (0.00 %)	4.33333	✗
Hc.P*	**20 (100.00 %)**	1 (5.00 %)	0 (0.00 %)	4.66667	✗
Ga.P.A4.Gd	17 (85.00 %)	**18 (90.00 %)**	0 (0.00 %)	6.00000	✗
Ga.D*.A4.Gd	19 (95.00 %)	2 (10.00 %)	0 (0.00 %)	6.33333	✗
Ts.P	19 (95.00 %)	1 (5.00 %)	0 (0.00 %)	6.50000	
Ga.P*.A4.Gd	16 (80.00 %)	0 (0.00 %)	1 (5.00 %)	7.00000	✗
Ga.P*.A2.Gd	14 (70.00 %)	0 (0.00 %)	**3(15.00 %)**	8.33333	✗
Ga.P.A2.Gd	17 (85.00 %)	1 (5.00 %)	0 (0.00 %)	9.00000	
Ga.D*.A2.Ux	18 (90.00 %)	1 (5.00 %)	0 (0.00 %)	9.33333	
Ts.P*	14 (70.00 %)	1 (5.00 %)	0 (0.00 %)	10.00000	
Ga.P*.A2.Ux	14 (70.00 %)	0 (0.00 %)	1 (5.00 %)	10.00000	
Hc.P	**20 (100.00 %)**	0 (0.00 %)	0 (0.00 %)	10.16667	
Ga.D*.A2.Gd	19 (95.00 %)	0 (0.00 %)	0 (0.00 %)	10.16667	
Ga.P*.A4.Ux	13 (65.00 %)	0 (0.00 %)	1 (5.00 %)	10.66667	
Ga.D*.A4.Ux	15 (75.00 %)	0 (0.00 %)	0 (0.00 %)	16.00000	
Ga.P.A2.Ux	12 (60.00 %)	0 (0.00 %)	0 (0.00 %)	16.66667	
Ga.P.A4.Ux	14 (70.00 %)	0 (0.00 %)	0 (0.00 %)	16.66667	
Ga.D.A4.Gd	10 (50.00 %)	0 (0.00 %)	0 (0.00 %)	17.33333	
Ga.D.A2.Gd	8 (40.00 %)	0 (0.00 %)	0 (0.00 %)	18.16667	
Ga.D.A2.Ux	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	18.16667	
Ts.D*	5 (25.00 %)	0 (0.00 %)	0 (0.00 %)	20.33333	
Ga.D.A4.Ux	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	20.66667	
Hc.D	13 (65.00 %)	0 (0.00 %)	0 (0.00 %)	21.16667	
Ts.D	10 (50.00 %)	0 (0.00 %)	0 (0.00 %)	21.83333	

que usa el operador Gd –Greedy–) ha demostrado ser muy eficiente para los escenarios problemáticos de complejidad baja y media. Sin embargo, en términos generales, este algoritmo GA encuentra la mayor cantidad de soluciones, exactamente 35 (de 60 ejecuciones, 20 ejecuciones por cada uno de los escenarios del problema).

Debido al alto número de variantes de los algoritmos que hemos implementado, no es fácil particularizar el rendimiento de cada uno de ellos en comparación con el resto, simplemente inspeccionando las tablas obtenidas, por lo que hemos optado por un torneo basado en rangos. Más precisamente, hemos calculado el rango r_j^i de cada algoritmo j en cada instancia i. En todos los casos, hemos utilizado a efectos de clasificación la suma del número de soluciones factibles encontradas en cada instancia de problema (del conjunto de 20 ejecuciones). El mejor algoritmo recibe el rango 1 y el

Figura 3.24: Propuesta Básica: Torneo de las diferentes metaheurísticas básicas trabajando sobre el problema del TDP para los escenarios tomados de [251]. Cada caja comprende desde el primer al tercer cuartil de la distribución, la mediana (2do cuartil) es indicada con una línea vertical, la zona sin valores atípicos (whiskers span) es de 1.5 veces el rango intercuartil, además los valores atípicos están marcados con un signo '+'. Para los dos modelos Primal/Dual, con/sin ruptura de simetrías, P Primal sin ruptura de simetrías, P* Primal con ruptura de simetrías; D Dual sin ruptura de simetrías, y D* Dual con ruptura de simetrías.



peor recibe el rango k, donde $k = 24$ es el número de algoritmos involucrados en el ranking. Luego, en la quinta columna de la tabla 3.22, mostramos el valor de clasificación promedio obtenido de las distribuciones de los rangos de cada algoritmo. El torneo mostrado en la tabla 3.24 nos indica que la variante del algoritmo HC que utiliza la representación Dual con el esquema de ruptura de simetrías logra el mejor lugar en el Ranking.

3.4.4 Test Estadísticos para las Técnicas Metaheurísticas del TDP

Por otra parte, vamos a considerar la aplicación los tests de Friedman, así como el de Iman-Davenport.

Tabla 3.23: Propuesta Básica para el TDP: Resultados obtenidos al aplicar el test de Friedman and Iman-Davenport considerando los escenarios tomados de [251]. Para los dos modelos Primal/Dual, con/sin ruptura de simetrías, P Primal sin ruptura de simetrías, P* Primal con ruptura de simetrías; D Dual sin ruptura de simetrías, y D* Dual con ruptura de simetrías.

	Friedman value	Critical χ^2 value	Iman-Davenport value	Critical F_F value
All 24	46.303333	35.172462	4.080188	1.766805

Para afianzar las conclusiones sobre el comportamiento de las técnicas abordadas, aplicaremos además, el test de Holm. Los resultados para estos test se pueden apreciar en las tablas 3.23 – 3.24.

Tabla 3.24: Propuesta Básica para el TDP: Resultados obtenidos al aplicar el test de Holm, utilizando Hc.D* como algoritmo de control. Para los dos modelos Primal/Dual, con/sin ruptura de simetrías, P Primal sin ruptura de simetrías, P* Primal con ruptura de simetrías; D Dual sin ruptura de simetrías, y D* Dual con ruptura de simetrías.

i	algorithm	z-statistic	p-value	α/i
1	Hc.P*	5.774e-002	4.770e-001	5.000e-002
2	Ga.P.A4.Gd	2.887e-001	3.864e-001	2.500e-002
3	Ga.D*.A4.Gd	3.464e-001	3.645e-001	1.667e-002
4	Ts.P	3.753e-001	3.537e-001	1.250e-002
5	Ga.P*.A4.Gd	4.619e-001	3.221e-001	1.000e-002
6	Ga.P*.A2.Gd	6.928e-001	2.442e-001	8.333e-003
7	Ga.P.A2.Gd	8.083e-001	2.095e-001	7.143e-003
8	Ga.D*.A2.Ux	8.660e-001	1.932e-001	6.250e-003
9	Ts.P*	9.815e-001	1.632e-001	5.556e-003
10	Ga.P*.A2.Ux	9.815e-001	1.632e-001	5.000e-003
11	Hc.P	1.010e+000	1.562e-001	4.545e-003
12	Ga.D*.A2.Gd	1.010e+000	1.562e-001	4.167e-003
13	Ga.P*.A4.Ux	1.097e+000	1.363e-001	3.846e-003
14	Ga.D*.A4.Ux	2.021e+000	2.165e-002	3.571e-003
15	Ga.P.A2.Ux	2.136e+000	1.633e-002	3.333e-003
16	Ga.P.A4.Ux	2.136e+000	1.633e-002	3.125e-003
17	Ga.D.A4.Gd	2.252e+000	1.217e-002	2.941e-003
18	Ga.D.A2.Gd	2.396e+000	8.287e-003	2.778e-003
19	Ga.D.A2.Ux	2.483e+000	6.521e-003	2.632e-003
20	Ts.D*	2.771e+000	2.792e-003	2.500e-003
21	Ga.D.A4.Ux	2.829e+000	2.335e-003	2.381e-003
22	Hc.D	2.916e+000	1.775e-003	2.273e-003
23	Ts.D	3.031e+000	1.218e-003	2.174e-003

Según el test de Friedman e Iman-Davenport, los valores críticos son ligeramente menores que los arrojados por los tests respectivos, por lo que no está clara la existencia de diferencias significa-

tivas (véase la tabla 3.23). Al revisar el test de Holm, la mayoría de las técnicas no parecen mostrar diferencias significativas (cerca de 88%), (véase la tabla 3.24).

3.5 Revisión de Resultados: BIBD y TDP

En este capítulo se muestra una extensa experimentación aplicando las diferentes técnicas metaheurísticas que hemos seleccionado, al abordar los problemas de diseño de bloques incompletos y el de diseño plantillas. Los diferentes análisis realizados nos muestran que las técnicas empleadas operan de manera adecuada para resolver los problemas utilizados, ofreciendo resultados alentadores.

Como análisis importante podemos decir que el uso de diferentes formas de exploración del vecindario, por medio de la incorporación de representaciones alternativas, permiten a las diferentes técnicas explorar zonas del espacio de búsqueda complementarias, lo que conlleva a encontrar soluciones de alta calidad utilizando menos recursos

La técnica TS se ha comportado de manera adecuada al resolver los dos problemas atacados.

Respeto a los operadores de cruce utilizados en los GAs, nos permite verificar como proporcionando cierto nivel de inteligencia a este operador su capacidad de obtener mejores soluciones se puede apreciar con claridad.

Por otra parte, la incorporación de procesos de ruptura de simetrías nos permite abrigar la esperanza de que esta estrategia proporciona mayor potencia a las técnicas metaheurísticas empleadas.

Finalmente, podemos mencionar que este análisis de resultados nos permitirá determinar las acciones que debemos seguir para la incorporación de otros mecanismos que contribuyan a potenciar el uso de estas técnicas para resolver problemas de optimización.

3.6 Contribuciones

En este capítulo presentamos los diferentes problemas que hemos abordado para realizar la experimentación correspondiente, utilizando las técnicas básicas de búsquedas locales, tanto trayectoriales como poblacionales. También, se muestran los detalles de los tipos de representación alternativa que hemos empleado sobre los dos problemas utilizados a lo largo de la investigación, los esquemas de ruptura de simetrías y la experimentación que ha sido aplicada. Además, se realiza un análisis estadístico de los resultados obtenidos.

Las contribuciones que consideramos importantes mencionar al concluir esta parte del trabajo son:

- Proponemos una nueva formulación alternativa a los problemas abordados que sirven como otra opción a la formulación clásica de estos.
- Proponemos el empleo de las primeras aplicaciones de algunas metaheurísicas a problemas de optimización que tradicionalmente se han abordado mediante métodos LP y CPS.
- Se presentan las primeras aplicaciones de algunas metaheurísicas a los dos problemas indicados utilizando espacios de búsqueda complementarios, frente a la tradicional aplicación de la misma representación del problema.
- Se ha considerado la posibilidad de imponer procedimientos de ruptura de simetrías para reducir el espacio de búsqueda de los problemas (en las representaciones primal y alternativa).

Todo este trabajo nos ha permitido describir aportaciones publicadas en la literatura científica, y que además han sido incorporadas en este capítulo:

- David Rodríguez Rueda, Carlos Cotta y Fernández-Leiva Antonio J. [267, 268, 269, 271].
- David Rodríguez Rueda, Enrique Darghan y Julio Monroy [272].

CAPÍTULO 4

TÉCNICAS HÍBRIDAS

En este capítulo vamos a presentar las diferentes técnicas híbridas que hemos utilizado en la presente investigación, así como los resultados obtenidos, para los dos problemas que han sido abordados. Concretamente nos referiremos a las propuestas de algoritmos:

- Meméticos.
- Colaborativos.

Este capítulo se estructura de la siguiente forma: En la Secciones 4.1 y 4.2, se presentan las propuestas meméticas para resolver, tanto, el problema del BIBD como el TDP, describiendo los optimizadores locales y los esquemas de hibridación. En la Sección 4.3, se describe la propuesta colaborativa a utilizar, así como, la forma de interacción del modelo, presentando las políticas de migración/recepción y las topologías utilizadas. Luego en las secciones 4.4 y 4.5 se presentan los métodos híbridos a utilizar, así como los resultados experimentales par los problemas abordados, también, se realiza un análisis estadístico de los resultados. En la Sección 4.6 se presenta una breve discusión de los resultados del capítulo y finalmente, el la Sección 4.7 se presentan las contribuciones de este capítulo a la tesis

4.1 Propuestas Meméticas para el Problema del BIBD

Nuestra propuesta se basa en la combinación de un algoritmo genético de estado estacionario con dos optimizadores locales. Estos optimizadores corresponden con métodos de:

- Hill Climbing.
- Tabu Search.

4.1.1 Optimizadores Locales

En esta investigación estamos proponiendo la utilización de dos técnicas de búsqueda local: Hill Climbing y Tabu Search. En el primero de ellos se han considerado dos versiones para el HC, (*Hill*

Climbing – HC), basadas en cada una de las dos vecindades definidas en la sección 3.1.4. La única diferencia entre ambas, aparte del empleo de una vecindad diferente, es la inicialización:

- Si se emplea la vecindad bit-flip, la inicialización es completamente aleatoria.
- Si se emplea la vecindad swap, la inicialización es aleatoria, pero sujeta a la restricción de que cada fila tenga exactamente r unos.

Por lo demás, el procedimiento es el estándar: se explora completamente la vecindad, y se elige la mejor solución (*steepest ascent*), salvo que ésta sea peor que la actual, en cuyo caso se ha llegado a un óptimo local, y se reinicia la búsqueda.

La segunda técnica considerada ha sido un algoritmo TS (Tabu Search – TS), también definido sobre la base de las dos vecindades diseñadas. Para el caso de HC, la inicialización se realiza atendiendo al tipo de vecindad que se empleará, y la exploración de la vecindad es completa. Los demás elementos del algoritmo son los que siguen:

- En el caso de emplear la vecindad bit-flip, se considerará tabú la modificación de una determinada celda de la matriz. En el caso de la vecindad swap, los movimientos tabú son intercambios entre posiciones de la misma fila.
- La persistencia tabú de los movimientos aceptados se elige aleatoriamente en cada paso dentro del rango $[\beta/2, 3\beta/2]$, donde $\beta = vb$ en el caso de bit-flip, y $\beta = vbr$ en el caso de swap. Nótese la particularidad para cada instancia del problema.
- El criterio de aspiración es mejorar la mejor solución encontrada hasta el momento.
- Tras un número n_{intens} de evaluaciones sin mejora se intensifica la búsqueda, volviendo a la mejor solución conocida.

4.1.2 Algoritmo Genético

Al igual que con las técnicas de búsqueda local, en el caso del empleo de algoritmos genéticos se han considerado dos variantes, correspondientes a cada una de las vecindades. En el caso del empleo de vecindad bit-flip, el algoritmo genético es una versión estándar, en la que la inicialización de la población se realiza de manera aleatoria, y las operaciones de cruce y mutación pueden realizarse de manera ciega. Concretamente, se ha considerado el operador de cruce uniforme (UX) y la mutación por inversión de bits. En el caso del empleo de la vecindad swap, el GA ha de adaptarse para trabajar en todo momento manteniendo la restricción del número de unos por fila. Para ello:

- Cada individuo se genera inicialmente de forma aleatoria, aplicando la restricción de que para cada fila de la matriz de incidencia sólo pueden aparecer r unos.
- El cruce se realiza mediante una variante de UX a nivel de filas: se realizan torneos aleatorios para determinar si la fila i-ésima ha de copiarse completamente de un padre o del otro.
- La mutación se realiza mediante intercambio de posiciones, sujeta a la restricción de que dichas posiciones estén en la misma fila, y que los valores de las mismas sean diferentes (de lo contrario la mutación dejaría al individuo inalterado).

Por lo demás, se ha considerado un modelo de evolución de estado estacionario con reemplazo del peor, y se ha realizado la selección por torneo binario.

Algoritmo 12: Pseudocódigo para el Algoritmo Memético.

```
1  begin
2      for i ← to TamPoblación do
3          pop[i] ← GenerarMatriz(v,b,r);
4          Evaluar(pop[i]);
5      end
6      while numEvals < maxEvals do
7          if rand < p_X then
8              padre_1 ← SelecciónTorneo(pop);
9              padre_2 ← SelecciónTorneo(pop);
10             descendencia ← Recombinación(padre_1, padre_2);
11         else
12             descendencia ← SelecciónTorneo(pop)
13         end
14         descendencia ← Mutación(descendencia, p_M);
15         if rand < p_LS then
16             descendencia ← BLocal(descendencia, Evals_LS)
17         end
18         Evaluar(descendencia);
19         pop ← Reemplace(pop, descendencia);
20         if Estancamiento (n_ι) then
21             pop ← Reiniciar(pop, f%)
22         end
23         ActualizaEvaluaciones(numEvals);
24     end
25 end
```

4.1.3 Propuesta Memética para el $\langle v,b,r,k,\lambda\rangle$-BIBD Problem

Los algoritmos meméticos (MAs) representan una metaheurística de los paradigmas de la optimización basada en el aprovechamiento del conocimiento del problema para explotar sus principales características y realizar una combinación sinérgica de ideas tomadas desde otras metaheurísticas basadas en población en combinación con otras basadas en trayectoria. El MA considerado en esta investigación utiliza las características más relevantes de dos buscadores locales incrustados en la propuesta memética junto con un mecanismo de recombinación basado en técnicas heurísticas.

Nuestro MA utiliza un algoritmo genético (GA) como mecanismo subyacente de búsqueda basado en la población. El GA no es un simple mecanismo que mezcla procesos de diversificación con búsqueda locales muti-arranque, sino que por el contrario, incorpora componentes de intensificación (normalmente operadores de recombinación heurísticos) con la aplicación de un importante mecanismo de descubrimiento de nuevas soluciones de alta calidad. En el algoritmo 12 se puede apreciar el pseudocódigo del algoritmo memético utilizado. Como se puede observar el nivel superior del MA se asemeja a un GA de estado estacionario. El proceso de selección se realiza usando torneo binario para la generación de nuevos individuos, además de utilizar un mecanismo de migración del reemplazo del peor individuo de la población. Para mantener la diversidad en la población, los individuos duplicados son descartados y un mecanismo de reinicio se emplea para reactivar la búsqueda cuando se detecta un

Algoritmo 13: Pseudocódigo para el Algoritmo de Recombinación.

```
1 begin
2     filas ← ∅;
3     for i ← to numPadres do InserteFilas(padre_i, filas) ;
4     hijo.fila[1] ←ExtraeAletoria(filas);
5     for i ← 2 to v do
6         foreach fila ∈ filas do Verifica(hijo, fila, i) ;
7         hijo.fila[i] ← Mejor(filas);
8     end
9 end
```

estancamiento en la calidad de las soluciones. Esto permitirá mantener un fracción $f_{\%}$ de los mejores individuos en cada generación, e incorporando nuevos individuos por medio de la aplicación de un mecanismo de refrescamiento aleatorio. Este procedimiento se activa después de un número de n_{ι} evaluaciones sin mejora, respecto del mejor individuo encontrado hasta el momento. El operador de mutación se aplica por medio del intercambio de dos posiciones (con valores {0,1]) en la misma fila.

Para este trabajo hemos utilizados dos tipos de operadores de recombinación: un operador de cruce estándar y un operador definido con base al intercambio (swap), al cual hemos llamado "cruce voraz" (*GrX*, tambien *Gd*). En el algoritmo 13 se puede apreciar el pseudocódigo para el proceso de recombinación. Como podemos observar, GrX inicia con la creación de un *pool de filas* ($P_n(Rows)$) con todas las filas disponibles en los padres (suprimiendo las filas idénticas). Luego se aplicará uno de los operadores de cruce: (a) si existen suficientes filas (es decir n>v) se aplicará el operador GrX, (b) de lo contrario se aplica el operador estándar Ux. Para el caso particular del operador *Greedy*, se selecciona, en forma aleatoria, un total de **v** filas, para posteriormente iniciar con el chequeo de las demás filas disponibles. El objetivo es quedarnos con aquella combinación de filas que viole el menor número de restricciones. Por supuesto, esto nos eleva el tiempo computacional (esto será considerado para los efectos del tiempo de respuesta). Para hacer esto, observe que los productos escalares se pueden almacenar en un caché una vez que se han calculado, y por lo tanto, cada vez que se agrega una nueva fila, sólo se deben calcular los productos escalares con respecto a la fila seleccionada en el paso anterior. Si hay ρ filas en el conjunto, la cantidad de cálculos es $(\rho-1)+\cdots+(\rho-v+1) = (\rho-v/2)(v-1)$. Dado que la cantidad total de productos escalares en una evaluación completa es $v(v-1)/2$, el costo de GrX es en cada caso equivalente a $2\rho/v-1$ evaluaciones. Este aspecto es calculado en la función de costo computacional total del algoritmo para propósitos de comparación.

4.2 Propuestas Meméticas para el Problema del TDP

El esquema del MA utilizado consiste en dotar a un algoritmo genético (GA) con un mecanismo de mejora local. El esquema para el algoritmo MA se puede ver en el algoritmo 12. Particularmente hemos combinado el GA con HC y TS. En cuanto a la probabilidad de aplicación de la mejora local se ha determinado según lo descrito por Heart [143]. Diferentes versiones de este esquema de algoritmo memético han sido analizadas, por medio de la aplicación de las dos técnicas de búsqueda local descritas previamente. La estrategia de mejora local ha sido aplicada justo después de la etapa de mutación. La búsqueda local es aplicada en forma individual de acuerdo a una probabilidad p_{LS}; en caso de aplicación de esta mejora local, el algoritmo es lanzado un número de $Evals_{LS}$ evaluaciones.

Esta es una estrategia de mejora muy simple, y es necesario recordar que existen otras técnicas mucho más complejas en la literatura para determinar qué tipo de mejora puede aplicarse, así como a qué individuo de la población en particular se debe aplicar dicha mejora [232, 233].

4.3 Nuestra Propuesta Colaborativa

En este modelo se incluyen las representaciones que hemos denominado ***Primal*** y ***Dual***, estamos considerando la utilización de las tres metaheurísticas (HC,TS y GA) junto con los dos algoritmos meméticos (MA_{Hc} y MA_{Ts}) descritas en las secciones anteriores. La arquitectura de esta estrategia de cooperación se basa en los métodos de optimización utilizados tanto de los algoritmos de búsqueda locales como de los algoritmos híbridos empleados. Las técnicas de búsqueda local utilizan la exploración del vecindario de un determinado candidato a solución. Para el caso particular del problema del TDP: los candidatos a solución están representados por plantillas, cada plantilla está formada por el número variaciones demandadas y su respectiva cantidad de celdas asignadas. Para una determinada plantilla τ su eventual vecindario $\Phi(\tau)$ estaría representado en la posibilidad de asignar una nueva celda para un diseño ν_i (adicionar una celda) y la supresión de otra celda para un diseño distinto $\nu_{i'}$ (substraer una celda). A este vecindario se le aplica un mecanismo de explotación por medio de dos técnicas de búsqueda local.

La primera técnica es: *Hill Climbing* (HC), en la cual dada una solución candidata τ, su vecindario $\Phi(\tau)$ es explorado y la mejor solución es tomada como la solución actual, siempre que ésta sea mejor que la actual. Si no se alcanza una solución mejor a la actual, el proceso es reiniciado desde otro punto del espacio de búsqueda. La segunda técnica considerada es la búsqueda tabú (*Tabu Serach*, TS) la cual ha sido descrita en [271].

Con respecto al GA, se ha considerado un modelo de evolución de estado estacionario con reemplazo del peor [18]. Se maneja una población de soluciones candidatas y se aplica un mecanismo de selección iterativa, cruce y reemplazo. La selección se hace por medio de un torneo binario y el reemplazo siguiendo una política $(\mu, 1)$ (un nuevo individuo es generado e insertado en la población por medio del reemplazo del peor). Se ha utilizado una variante del operador de cruce uniforme UX [294]. Basándonos en la simetrías del problema, las plantillas son comparadas para encontrar las similitudes entre ellas. Posteriormente, se hace el intercambio de información a nivel de plantillas por una selección al azar del par de plantillas coincidentes (esta selección se hace tomando una variación de uno de los padres). La operación de mutación se realiza de la misma forma como ha sido descrito en el vecindario respectivo para las búsquedas locales. Por otra parte, el esquema del MA utilizado es muy simple, y consiste en dotar a un algoritmo genético (GA) con un mecanismo de mejora local.

El esquema para el algoritmo MA se puede ver en el algoritmo 12. Particularmente hemos combinado el GA con HC y TS. En cuanto a la probabilidad de aplicación de la mejora local se ha determinado en $1/16$, según lo descrito por Heart [143]. En todas las propuestas metaheurísticas (i.e. HC, TS, GA y MA) la función de evaluación de un determinado candidato es obtenida por medio de la aplicación de la conocida biblioteca de resolución de problemas de programación lineal entera, *LpSolve*, la cual nos devuelve los valores R_i que expresan el número de troquelaciones que deben ser aplicadas a cada plantillas para alcanzar la demanda requerida. El objetivo es minimizar la pérdida de materia prima. Diferentes versiones de este esquema de algoritmo memético han sido analizadas, por medio de la aplicación de las dos técnicas de búsqueda local descritas previamente. La estrategia de mejora local ha sido aplicada justo después de la etapa de mutación. La búsqueda local es aplicada en forma individual de acuerdo a una probabilidad p_{LS}; en caso de aplicación de esta mejora local,

el algoritmo es lanzado un número de $Evals_{LS}$ evaluaciones. Esta es una estrategia de mejora muy simple, y es necesario recordar que existen otras técnicas mucho más complejas en la literatura para determinar que tipo de mejora puede aplicarse, así como a que individuo de la población en particular se debe aplicar dicha mejora [232, 233].

Con respecto al problema del BIBD: todos los agentes serán inicializados con soluciones candidatas en forma aleatoria, siempre que cumplan con la restricción $\sum_{j=1}^{b}(M) = r$. Luego, cada algoritmo será ejecutado un número máximo de iteraciones donde, en cada ciclo, los agentes aplican una determinada técnica de búsqueda actualizando el conjunto de soluciones candidatas (nótese que si el agente aplica un método LS se toma un individuo del conjunto para intentar una mejora, pero si aplica un método basado en población se tomará la totalidad de las soluciones del conjunto y posiblemente se generen nuevas soluciones aleatorias para completar la población inicial). Los agentes transfieren soluciones entre ellos de acuerdo a la topología considerada. El reemplazo de las soluciones en el conjunto se realiza basándonos en una de las políticas de reemplazo estudiadas (además se descartan los individuos duplicados).

4.3.1 Esquemas Topológicos para la Cooperación

Para la aplicación de los esquemas topológicos hemos visualizado tres configuraciones que representan nuestro conjunto de alternativas combinables, lo que puede llevar a generar varios escenarios factibles de ser utilizados, como se muestra en la figura 4.1. Los mecanismos de colaboración, para el intercambio de información entre agentes, se describen a continuación:

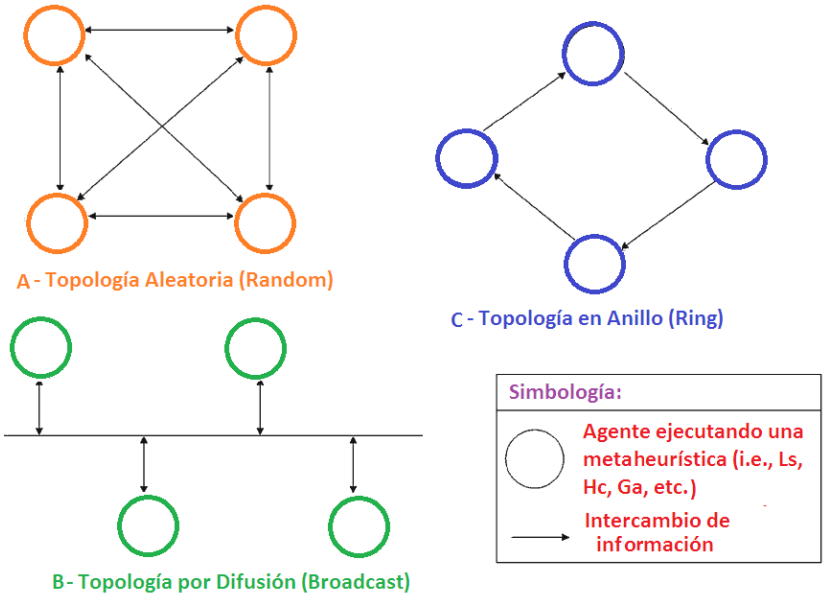
- RANDOM: $\mathbf{T}_R$ está compuesto por n pares (i, j) que han sido seleccionados en forma aleatoria de $\mathbb{N}_n^+ \times \mathbb{N}_n^+$. Este muestreo se realiza cada vez que se produce una comunicación, y por lo tanto, dos agentes eventualmente pueden comunicarse en cualquier etapa del proceso.
- BROADCAST: $\mathbf{T}_R = \mathbb{N}_n^+ \times \mathbb{N}_n^+$, i.e. una topología *go-with-the-winners* en la que la mejor solución general en cada punto de sincronización se transmite a todos los agentes. Esto significa que todos los agentes ejecutan la intensificación sobre la misma región local del espacio de búsqueda al comienzo de cada ciclo.
- RING: $\mathbf{T}_R = \{(i, i(n)+1) \mid i \in \mathbb{N}_n^+ \text{ and } i(n) \text{ denotes } i \text{ modulo } n\}$. Por lo cual, existe una lista circular de agentes en la que cada uno sólo envía (recibe) información a su sucesor (o su predecesor).

Estos esquemas topológicos de comunicación, ya habían sido empleados en [10, 11] para abordar el *problema del intercambio de herramientas* (*tool switching problem*) con cierto éxito; sin embargo, en este trabajo no fue considerada: (a) la ruptura de la simetría del problema, (b) el uso de representaciones alternativas y (c) diferentes políticas para la migración y la aceptación de la solución.

4.3.2 Políticas de Migración/Recepción

Para la elección de la región que será empleada en la exploración hemos adaptado algunas de las ideas propuestas en [229, 231], sobre algoritmos meméticos, por lo que lo hemos trasladado a nuestros algoritmos cooperativos generados a partir del esquema mostrado en el algoritmo 14. En nuestro esquema de colaboración propuesto consideramos una serie de algoritmos que cooperan intercambiando información basados en una red de nodos (topologías de interacción descrito en la sección anterior, 4.3.1), en la cual la solución candidata obtenida por un determinado algoritmo (agente a_i), al finalizar su ciclo de ejecución, será el candidato de entrada para un nuevo algoritmo agente a_j que

Figura 4.1: Topologías de interacción, presentadas por Amaya et al. [11]



inicia un nuevo ciclo de ejecución. Cada agente mantiene un conjunto de soluciones propias (pool de candidatos S_i), por ello debemos proveer a cada agente de un mecanismo que permita determinar qué candidato de este 'pool de soluciones' será seleccionado para ser entregado al nuevo agente (a_j) que inicia su ciclo (políticas de migración). Por otra parte, también es necesario que estos agentes cuenten con un mecanismo que les permita determinar a cuál de las soluciones candidatas de su pool se reemplazará (políticas de recepción). En particular, consideramos una serie de políticas para el envío de candidatos desde el agente i (migración), así como la aceptación de candidatos en el agente j (recepción/aceptación). Por lo tanto, con respecto a la selección del candidato *en el procedimiento de migración*, hemos considerado las siguientes políticas:

- (RANDOM − R): envía una solución candidata del pool, en forma aleatoria, al agente j,
- (DIVERSE − D): se envía un candidato S_i que maximiza la diversidad[1] en S_j, y
- (WORST − W): enviar el peor de los candidatos contenido en el pool de soluciones del agente emisor al agente j.

En lo referente a la *política de recepción/reemplazo* se han considerado, de igual forma, tres alternativas:

- (RANDOM − R): se acepta el candidato aportado y se reemplaza por un individuo del grupo S_j, seleccionado en forma aleatoria,
- (DIVERSE − D): aceptar un nuevo individuo sí, y solo sí, mejora la diversidad del grupo S_j, según el aporte del agente i y reemplaza el peor, y
- (WORST − W): permitirá aceptar el candidato y reemplazarlo por el peor del grupo S_j.

[1] El objetivo es seleccionar los individuos cuya distancia genotípica (en un sentido Hamming) a los individuos en la población receptora es máxima.

4.3.3 Interacción en el Modelo de Cooperación

El algoritmo memético descrito en la sección 4.1.3 y presentado en el algoritmo 12, puede ser considerado como una técnica integrativa e híbrida de acuerdo a Puchinger y Raidl [252], en el cual, un algoritmo de búsqueda local es subordinado por la ejecución de un algoritmo genético externo (GA), es decir que la búsqueda local se ejecuta dentro del GA. Por otra parte, Puchinger and Raidl, también presentaron otro esquema interesante que denominaron **propuesta colaborativa** (**collaborative approach**). Este esquema de colaboración opera por medio de varios optimizadores locales, los cuales son ejecutados en paralelo (o secuencialmente) e intercambian información con cierta frecuencia. Este tipo de cooperación puede ser considerado en sí mismo como un paradigma de la programación, que comprende dos elementos principales [68]: (a) un conjunto de programas autónomos (generalmente llamados agentes), cada uno de los cuales implementa un método de solución particular, de acuerdo a sus características, (b) un esquema o modelo cooperativo que combina estos elementos autónomos en una estrategia simple y unificada para atacar problemas considerados difíciles de resolver.

En el enfoque colaborativo, la idea principal es la utilización de un determinado número de algoritmos de optimización (que posiblemente difieren en su configuración) para que cada uno de ellos explore un espacio de búsqueda específico a través de procesos de intensificación, en procura de un mecanismo efectivo que permita a dichas técnicas escapar de los *mínimos locales*, esto se logra por medio del intercambio de información sobre el espacio de búsqueda que ha sido explorado. Este enfoque ha demostrado ser particularmente eficiente en una serie de problemas combinatorios que han sido abordados en el ámbito científico [10, 11, 72, 196].

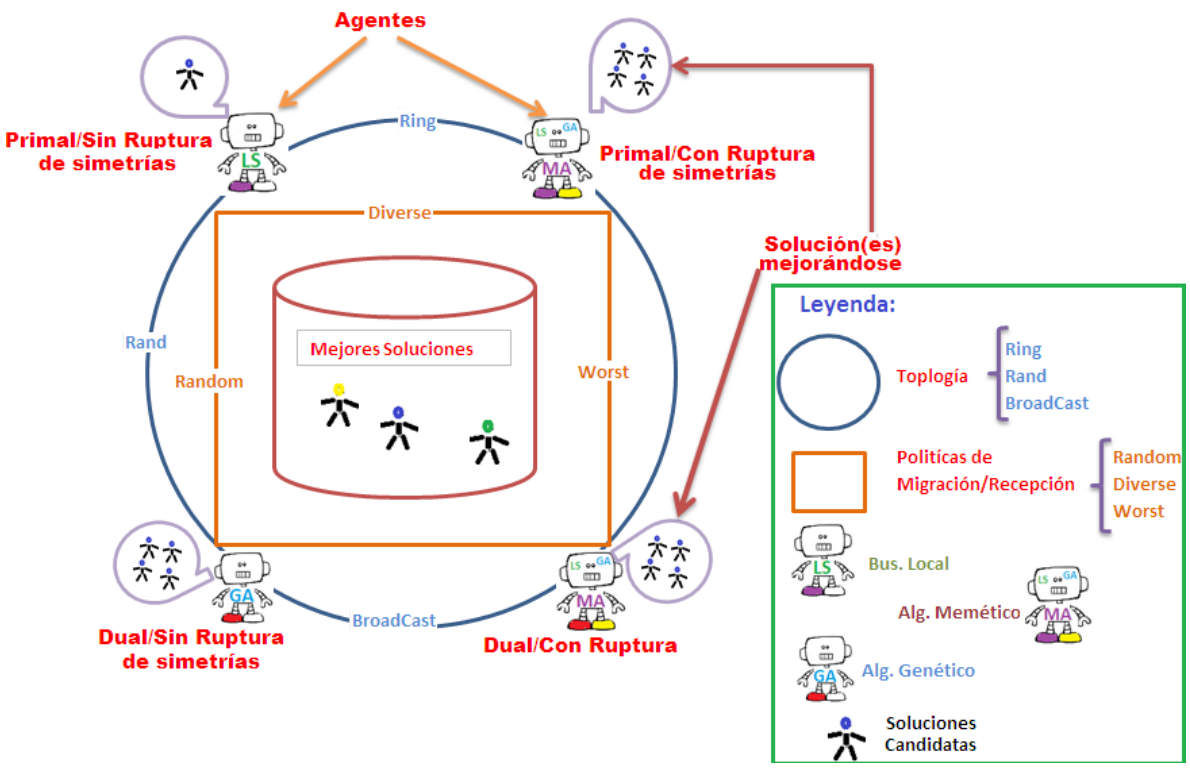
Ahora, a diferencia de lo que se hace a menudo en este tipo de esquema de colaboración nuestro interés es estudiar el efecto de considerar diferentes zonas del espacio de búsqueda para ser evaluados por separado según la topología de la red utilizada. Más precisamente, consideramos una serie de algoritmos que cooperan intercambiando información en los que los agentes son provistos por una de las técnicas metaheurísticas previamente propuestas (Hc, TS, GAs o MAs) o su equivalente, adaptado a la representación correspondiente (como se muestra en las secciones 3.1.8, y 3.2.9) y donde el algoritmo que se ejecuta en cualquier agente dado, también puede estar provisto de algún mecanismo para romper las simetrías (como se explica en las secciones 3.1.8.2 y 3.2.9.2). Esto significa que algunos agentes posiblemente trabajen en diferentes espacios de codificación/búsqueda y utilicen formulaciones distintas del problema, véase la figura 4.2. Los algoritmos dependen de su topología de interacción, de los modelos utilizado para codificar los candidatos y de las políticas de migración/recepción; y estos se analizan a continuación.

4.3.3.1 Defínción Formal del Modelo de Cooperación

Sea R una arquitectura con n agentes; cada agente $a_i (0 \leq i < n)$ en R está determinado por una de las técnicas metaheurísticas descritas anteriormente. Por lo tanto, estos agentes pueden trabajar con soluciones candidatas representadas en uno de los modelos de representación descritos (primal o dual), con o sin ruptura de simetrías. Los agentes participan en períodos de exploración aislada seguida por una comunicación sincronizada.

Denotaremos por Θ el número máximo de ciclos de exploración/comunicación para un cierto modelo de cooperación. Además, dado S_i como el conjunto de soluciones candidatas asociadas con el agente a_i (e.i. si el agente es ejecutado con alguna técnica de búsqueda local, entonces $\#S_i = 1$, y si el agente es ejecutado con una técnica basada en población —e.g. un MA —entonces $\#S_i \geq 1$, donde $\#S_i$ representa la cardinalidad de S_i), y dado $\mathbf{T}_R \subseteq \mathbb{N}_n^+ \times \mathbb{N}_n^+$ será la topología para el intercambio de

Figura 4.2: Modelo cooperativo propuesto para resolver problemas de optimización combinatoria.



información sobre R (i.e. si $(i, j) \in \mathbf{T}_R$ el agente a_i puede enviar información al agente a_j).

La arquitectura general del modelo es descrita en el algoritmo 14. En este algoritmo se tiene como entrada los parámetros del problema (ya sea BIBD o TDP), la topología de la red de agentes (la cual define las políticas de comunicación como se ha explicado antes en la sección 4.3.1), el algoritmo n (i.e. agentes o metaheurísticas) ejecutándose sobre cada nodo del sistema de colaboración (i.e. la red), las políticas de migración de las soluciones candidatas, y los criterios de aceptación de candidatos (como se explicó en la sección 4.3.2). Cada agente también tiene sus propios parámetros (como las tasas de aplicación de los operadores y el tipo de codificación –primal/dual–). Como resultado, el algoritmo proporcionará la mejor solución candidata encontrada durante el proceso (línea 18; nótese que la función de fitness *Fitness* es un concepto bien conocido en computación evolutiva, la cual, básicamente devuelve el valor que mide que tan cerca se encuentra una solución candidata de su óptimo). Primero, todos los agentes son inicializados con soluciones candidatas aleatorias (líneas 1-3). La inicialización del pool S_i está asociada con el agente i en el sistema, y esta es específica para el modelo de representación usado (i.e. primal o dual), que será manejada por la agente a_i. Luego, el algoritmo es ejecutado un número máximo de Θ ciclos de iteración (líneas 5-17) donde, por cada ciclo, la técnica de búsqueda contenida dentro de cada agente se ejecuta para actualizar su conjunto asociado de soluciones (líneas 6-8); nótese además que, si un determinado agente contiene un método de búsqueda local, esto básicamente significaría que se lograría una mejora de su solución (por ser única), pero si el agente contiene un método basado en población, se genera un nuevo conjunto de soluciones. Después, las soluciones son intercambiadas de un agente a otro de acuerdo con la topología considerada (líneas 9-15). En este proceso inicialmente (línea 10), las soluciones candidatas, a ser intercambiadas,

Algoritmo 14: Modelo-Cooperativo$_n$

```
1  for i ← 1 to ℕ_n^+ do
       // Generación ajustada a la representación del problema abordado por el agente a_i
2      S_i ← GeneraPoblaciónInicial();
3  end
4  ciclos ← 1;
5  while ciclos ≤ Θ do
6      for i ← 1 to ℕ_n^+ do
           // Actualiza población
7          S_i ← a_i(S_i);
8      end
9      if (i, j) ∈ T_R then
           // Selección del candidato para la migración basada en las políticas establecidas
10         s_enviado ← seleccioneCandidatoDesde(S_i) ;
           // Ahora, verificar la aceptación del candidato por medio de la
           // política de recepción respectiva
11         if AceptaCandidatoRecibido(s_enviado, S_j) then
               // Seleccione el candidato a reemplazar
12             s_sereemplaza ← SelecioneCandidatoReplazarEn(S_j);
               // Agregar candidato recibido (adaptar la codificación
               // del problema del agente j)
13             S_j ← S_j ∪ {Decodifcar_j(s_enviado)} \ {s_sereemplaza} ;
14         end
15     end
16     ciclis ← ciclos + 1;
17 end
18 return arg min{Fitness(Best(S_i)) | i ∈ ℕ_n^+};
```

son tomadas del pool de soluciones del agente fuente (i.e. nodo o metaheurística i) por medio de una selección basada en las *políticas de migración*. Así, el agente a_j determina (línea 11) que política de recepción será utilizada para la aceptación de la solución entrante del agente a_i (línea 10) (*criterio de aceptación*). Finalmente, si la solución recibida es aceptada, está reemplazará a un individuo del pool de soluciones candidatas del agente j (línea 13); el candidato a ser reemplazado será seleccionado por medio de una función heurística definida previamente (línea 12). Podemos notar que es posible definir varios criterio de migración (desde el agente i hacia el agente j) y aceptación (en el agente j). Por lo tanto, la combinación de diferentes políticas de migración/recepción generan diferentes algoritmos de colaboración (véase la sección 4.3.2). Se debe aclarar que para un máximo número de evaluaciones E_{max} y un específico número de ciclos Θ, cada ciclo de cooperación es equivalente a $E_{ciclos} = E_{max}/\Theta$, con lo cual, un determinado algoritmo de búsqueda de un agente será ejecutado un número de evaluaciones que corresponde con E_{ciclos}/n.

4.4 Métodos Híbridos para Resolver el BIBD

Como ya se ha mencionado antes, la hibridación se basa en el principio de que la sinergia puede generar algoritmos de alto rendimiento que exploten y combinen las ventajas de las estrategias puras

que operan en forma individual, es por ello que hemos abordado el problema del BIBD con dos esquemas colaborativos a saber:

- Algoritmos meméticos.
- Algoritmos cooperativos.

Nuestra MA utiliza un algoritmo genético (GA) como mecanismo subyacente de búsqueda basado en la población, unido a un agente optimizador local. Estos agentes compiten y cooperan de manera sinérgica alternándose en procesos de *diversificación/intensificación* para lograr avanzar hacia soluciones con un alto grado de calidad. Por su parte, los esquemas de colaboración que hemos aplicado, se basan en los métodos de optimización utilizados tanto de los algoritmos de búsqueda locales como de los algoritmos híbridos empleados. Estos agentes participan en períodos de exploración aislada seguida por una comunicación sincrónica. De esta forma se comparte información y posibles soluciones que permiten redirigir los procesos de búsqueda hacia zonas potenciales para lograr alcanzar óptimos globales.

4.4.1 Convenciones de Notación de las Técnicas Híbridas para el BIBD

Para el caso de los algoritmos Meméticos la notación resulta de manera similar que la descrita anteriormente en la sección 3.3.3.1 del capítulo 3, así por ejemplo, $MA.Hc.B.A2.Gd$ corresponde a un algoritmo Memético que implementa mejora local por medio de la técnica de búsqueda local *hill climbing*, hace uso de 2 padres para la recombinación por medio de un operador de cruce de tipo voraz , además de emplear la representación alternativa primal y una configuración sin ruptura de simetrías; y $MA.Ts.D*.A2.Gd$ representa un algoritmo Memético con búsqueda tabú, 2 padres para la recombinación con operador de cruce voraz implementado por medio de la representación dual con ruptura de simetrías.

En cuanto a los métodos cooperativos, estos se componen de algunas de las técnicas anteriores (descritas en la sección 3.3 del capítulo 3) combinadas de acuerdo con la topología y las políticas de migración/recepción descritas en la sección 4.3.2. La notación $\mathbf{T}n(a_1,\ldots,a_n)$MR se utiliza para identificar la configuración empleada para las combinaciones de las técnicas utilizadas, como se describe a continuación:

- $\mathbf{T} \in \{$BROADCAST (Bc), RANDOM (Ra), RING (Ri)$\}$ identifica el esquema topológico utilizado,
- n representa el número de agentes que actúan en la cooperación (i.e. algoritmos) como se ha descrito en la sección 4.3.3,
- a_i corresponde al método de optimización empleado por el agente i (para $1 \leq i \leq n$), y
- M, R $\in \{$RANDOM (R), DIVERSE (D), WORST (W) $\}$ identifica las políticas de migración y recepción de candidatos empleados por los agentes que actúan en la cooperación, como se ha descrito en la sección 4.3.2. En la experimentación hemos considerado las siguientes seis combinaciones para las políticas de *migración-recepción*:
  1. RANDOM-RANDOM (RR), como se indica, utiliza la política de ALEATORIEDAD para la migración y recepción de candidatos.
  2. RANDOM-WORST (RW): emplea una política de ALEATORIEDAD para determinar la migración de candidatos y el reemplazo del PEOR en conjunto de candidatos.
  3. RANDOM-DIVERSE (RD): se utiliza una política de ALEATORIEDAD para la migración y una estrategia de DIVERSIFICACIÓN del conjunto de candidatos para la recepción.
  4. DIVERSE-RANDOM (DR): se selecciona el candidato que mejor DIVERSIFICA el conjunto inicial (política de migración) y se reemplaza un candidato en forma ALEATORIA del con-

junto (política de recepción),

5. DIVERSE-WORST (DW): se selecciona el candidato que mejor DIVERSIFICA y se reemplaza el PEOR, y
6. DIVERSE-DIVERSE (DD): se migra por DIVERSIFICACIÓN y se reemplaza por DIVERSIFICACIÓN.

Nótese que no hemos empleado combinaciones como WD (i.e. WORST-DIVERSE), WR (i.e. WORST-RANDOM) y WW (i.e. WORST-WORST), debido que algunas pruebas preliminares, ejecutadas sobre dichas combinaciones, mostraron que la elección de la política WORST para la migración exhibe un pobre desempeño comparado con las otras combinaciones. En consecuencia no estamos incluyendo esta experimentación en la presente investigación.

Consideremos algunos ejemplos: $Ri2(Ts.B, MA.Ts.D.A2.Gd)RW$ corresponde a la utilización de 2-agentes empleando la topología RING para el esquema colaborativo que conecta los algoritmos (a) basados en búsqueda local Tabú (Ts), trabajando en una representación alternativa Primal (candidatos expresados en forma binaria) y (b) un algoritmo MA, el cual utiliza un esquema de presentación Dual (candidatos expresados en forma decimal), utilizando 2 padres para la recombinación por medio de un operador de cruce voraz (Gd), y que integra un operador de mejora local Ts; para este caso particular, el esquema colaborativo permite enviar un candidato, del conjunto de soluciones del nodo inicial, seleccionado en forma aleatoria (RANDOM política de migración), con reemplazo del peor individuo del conjunto de soluciones del nodo destino (WORST política de aceptación).

De manera similar: $Ra3(Ts.B, MA.Ts.B.A2.Gd, MA.Ts.D.A4.Gd)RD$, denota una cooperación con 3-agentes actuando en forma simultánea, en una topología aleatoria RANDOM, con (a) búsqueda Tabú como técnica integrativa y (b) dos variantes para los algoritmos meméticos (MAs); los candidatos son enviados por medio de una política de migración aleatoria (i.e. a RANDOM), mientras que son aceptados solo si se incrementa la diversidad del conjunto de soluciones del nodo final (i.e. DIVERSE).

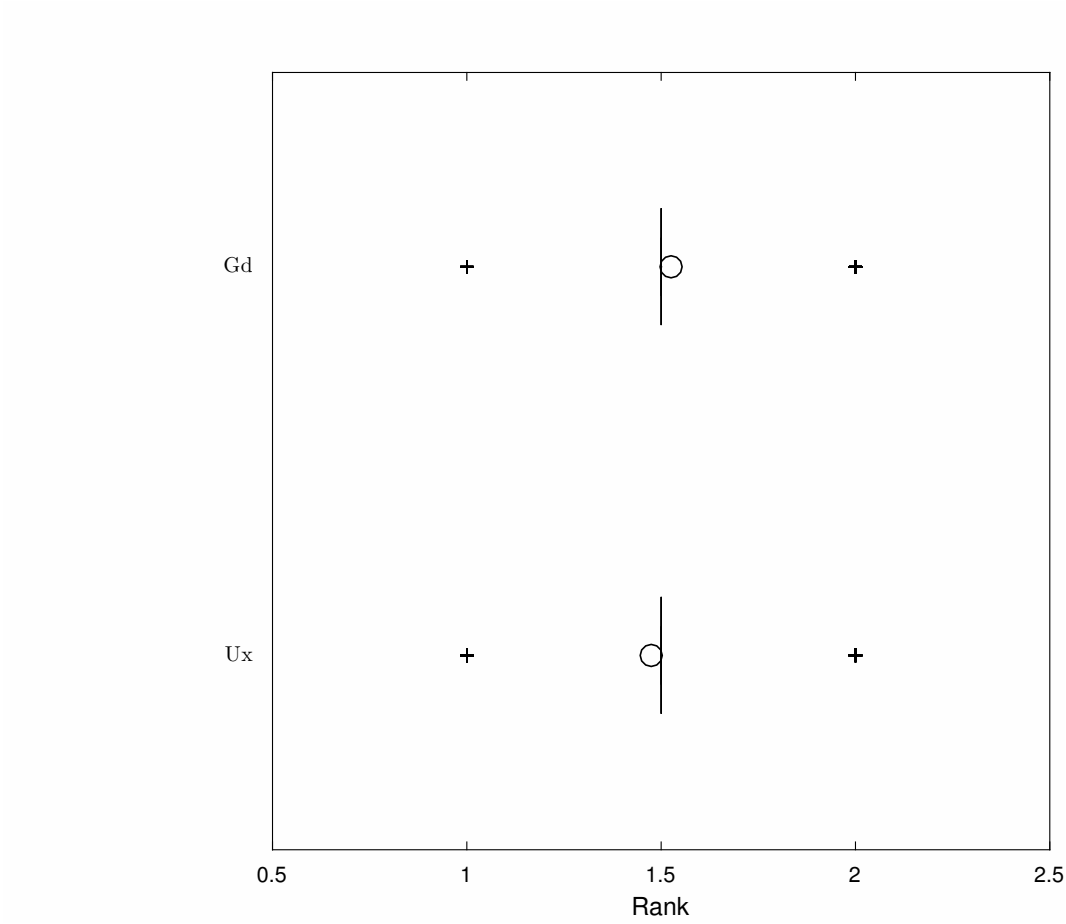
Para simplificar la notación utilizada en las tablas y figuras de las secciones siguientes, de manera similar la representación $\mathbf{T}n(pa, qb)$MR es utilizada para denotar un esquema colaborativo

$$\mathbf{T}n(\underbrace{a,\ldots,a}_{p},\underbrace{b\ldots b}_{q})MR$$

donde los agentes a y b intervienen $\mathbf{p}$ y $\mathbf{q}$ veces respectivamente (y donde p y q son números arbitrarios que cumplen con $n = p + q$); nótese que $\mathbf{p}$ (de igual forma $\mathbf{q}$) no será escrito cuando $\mathbf{p} = 1$ (de igual forma $\mathbf{q} = 1$). Además, para la configuración, $Bc4(3Ts.B, MA.Ts.D * .A4.Gd)RD$ denotaría un esquema, que podría escribirse en forma más precisa como: $Bc4(Ts.B, Ts.B, Ts.B, MA.Ts.D * .A4.Gd)RD$ (i.e donde $p = 3$ y $q = 1$). También, $Ra5(3Ts.B, 2MA.Ts.B.A2.Gd)DW$ representa un esquema de colaboración que hace uso de 5-agentes (algoritmos), que involucra la búsqueda local $Ts.B$ que opera con 3 agentes de la misma versión en conjunto con el algoritmo $MA.Ts.B.A2.Gd$ que actúa con los otros dos agentes.

Las demás notaciones empleadas en las secciones siguientes para representar los esquemas cooperativos utilizados en la totalidad de la experimentación deben quedar aclaradas con la explicación dada. El número de estrategias para el intercambio de información entre los diferentes agentes que actúan en las distintas configuraciones de los esquemas colaborativos pueden ser fácilmente obtenidos por medio de la combinación de las políticas de migración/recepción que ya han sido descritas previamente en esta investigación.

Figura 4.3: Problema del BIBD: Torneo para las diferentes técnicas genéticas agrupadas por tipo de cruce (Ux, GrX) aplicadas en ambos modelos Primal (B) y Dual (D) (Con/Sin Ruptura de simetrías), B identifica el modelo de representación Primal sin ruptura de simetrías, B* modelo Primal con ruptura de simetrías; D el modelo Dual sin ruptura de simetrías, y D* el modelo Dual con ruptura de simetrías. Cada caja comprende desde el primer al tercer cuartil de la distribución, la mediana (2do cuartil) es indicada con una línea vertical, la zona sin valores atípicos (whiskers span) es de 1.5 veces el rango intercuartil, además los valores atípicos están marcados con un signo '+'.



4.4.2 Resultados Experimentales de las Técnicas Meméticas sobre el BIBD

Antes de mostrar los resultados obtenidos para estas pruebas, debemos concentrarnos en realizar un análisis de los dos tipos de cruce empleados en las dos representaciones alternativas (primal/dual), para las versiones con/sin ruptura de simetrías (Ux y GrX), para ello hemos aplicado un torneo agrupando los algoritmos Genéticos de acuerdo al tipo de cruce. Como se puede observar en la figura 4.3, ambos cruces parecen comportarse de manera muy similar.

De igual forma, al aplicar los tests de Friedman e Iman-Davenport, no parece haber diferencias estadísticamente significativas como se aprecia en la tabla 4.1. El test de Holm corrobora lo concluido hasta el momento sobre los dos tipos de cruces aplicados sobre el modelo primal (véase la tabla 4.2).

Para la realización de estas pruebas nos hemos inclinado por la utilización del cruce heurístico basado en filas, donde se descarta las filas idénticas y se verifica cuál de las combinaciones de las filas

Tabla 4.1: Problema del BIBD: Resultados para los tests de Friedman e Iman-Davenport ($\alpha = 0.05$) para las técnicas genéticas agrupadas por tipo de cruce, para las representaciones Con/Sin Ruptura de simetrías de ambos modelos (Primal (B) y Dual (D)), B identifica el modelo de representación Primal sin ruptura de simetrías, B* modelo Primal con ruptura de simetrías; D el modelo Dual sin ruptura de simetrías, y D* el modelo Dual con ruptura de simetrías.

	Friedman value	Critical χ^2 value	Iman-Davenport value	Critical F_F value
All 2	0.840116	3.841459	0.839725	3.868712

Tabla 4.2: Problema del BIBD: Resultados para el test de Holm ($\alpha = 0.05$), usando Ux como algoritmo de control, para las técnicas genéticas agrupadas por tipo de cruce, para las representaciones Con/Sin Ruptura de simetrías de ambos modelos (Primal (B) y Dual (D)), B identifica el modelo de representación Primal sin ruptura de simetrías, B* modelo Primal con ruptura de simetrías; D el modelo Dual sin ruptura de simetrías, y D* el modelo Dual con ruptura de simetrías.

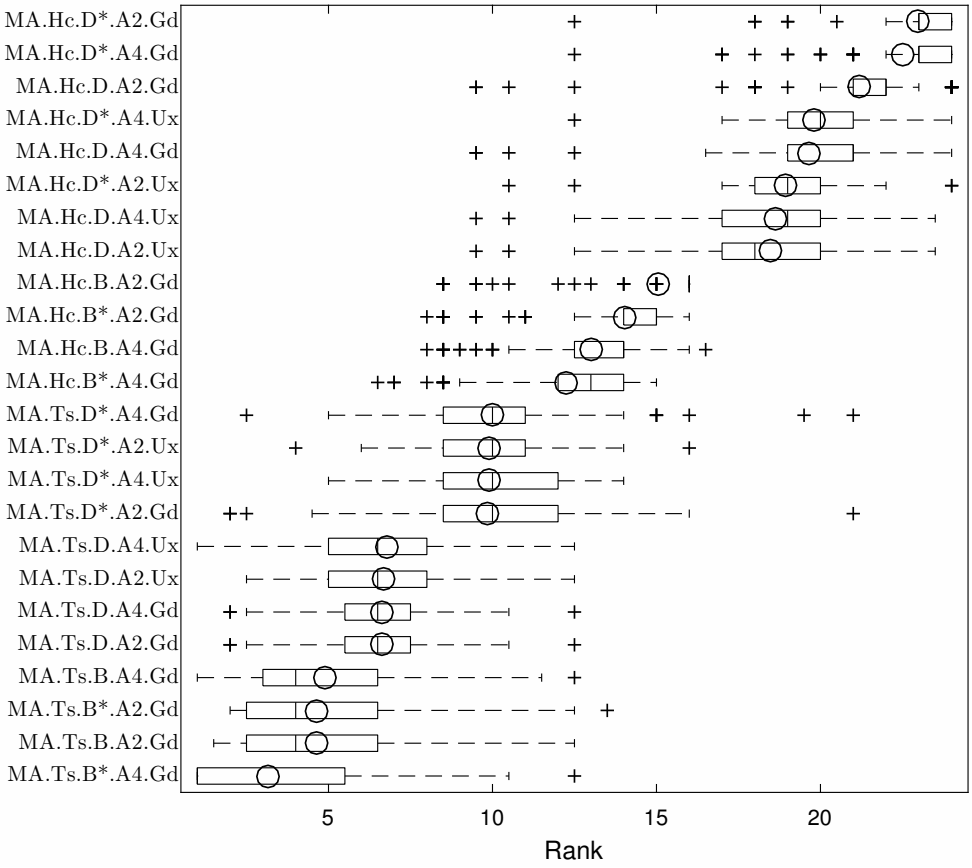
i	strategy	z-statistic	p-value	α/i
1	Gd	9.166e-01	1.797e-01	5.000e-02

Tabla 4.3: Problema del BIBD: Número (y porcentaje) de las instancias resueltas por las técnicas integrativas (meméticos) trabajando en forma separada sobre el conjunto de las 86 instancias tomadas de [32, 246], para los modelos Primal (B) y Dual (D) (Con/Sin Ruptura de simetrías), B Primal sin ruptura de simetrías, B* Primal con ruptura de simetrías; D Dual sin ruptura de simetrías, y D* Dual con ruptura de simetrías.

algoritmo	# (%)	algoritmo	# (%)
MA.Hc.B.A2.Gd	43(50.00%)	MA.Hc.D.A4.Gd	15(17.44%)
MA.Ts.B.A2.Gd	53(61.63%)	MA.Ts.D.A4.Gd	52(60.47%)
MA.Hc.B.A4.Gd	46(53.49%)	MA.Hc.D.A4.Ux	20(23.26%)
MA.Ts.B.A4.Gd	56(65.12%)	MA.Ts.D.A4.Ux	53(61.63%)
MA.Hc.B*.A2.Gd	43(50.00%)	MA.Hc.D*.A2.Gd	10(11.63%)
MA.Ts.B*.A2.Gd	53(61.63%)	MA.Ts.D*.A2.Gd	51(59.30%)
MA.Hc.B*.A4.Gd	46(53.49%)	MA.Hc.D*.A2.Ux	18(20.93%)
MA.Ts.B*.A4.Gd	59(68.60%)	MA.Ts.D*.A2.Ux	48(55.81%)
MA.Hc.D.A2.Gd	14(16.28%)	MA.Hc.D*.A4.Gd	9(10.47%)
MA.Ts.D.A2.Gd	52(60.47%)	MA.Ts.D*.A4.Gd	47(54.65%)
MA.Hc.D.A2.Ux	17(19.77%)	MA.Hc.D*.A4.Ux	16(18.60%)
MA.Ts.D.A2.Ux	52(60.47%)	MA.Ts.D*.A4.Ux	49(56.98%)

resulta ser la mejor GrX (Gd o greedy). Con el objetivo de evitar la ceguera detectada para el cruce Ux y aumentar la posibilidad de obtener mejor rendimiento de las técnicas híbridas aplicadas sobre el modelo Primal.

Figura 4.4: Problema del BIBD: Torneo para las diferentes técnicas meméticas aplicadas para los modelos Primal (B) y Dual (D) (Con/Sin Ruptura de simetrías), B Primal sin ruptura de simetrías, B* Primal con ruptura de simetrías; D Dual sin ruptura de simetrías, y D* Dual con ruptura de simetrías. Cada caja comprende desde el primer al tercer cuartil de la distribución, la mediana (2do cuartil) es indicada con una línea vertical, la zona sin valores atípicos (whiskers span) es de 1.5 veces el rango intercuartil, además los valores atípicos están marcados con un signo '+'.



4.4.2.1 Test Estadísticos de las Técnicas Meméticas del BIBD

En la figura 4.4 se presenta el torneo de los algoritmos basado en su Ranking. Para este caso (técnicas Meméticas), el algoritmo basado en el optimizador local TS, que emplea el modelo de representación Binaria (primal), con operador de cruce voraz y recombinación multi-padre (aridad 4), se muestra como el mejor posicionado en el torneo.

De igual manera la prueba estadística de Friedman e Iman-Davenport (véase la tabla 4.4), nos indica que existen diferencias estadísticamente significativas entre todos los algoritmos utilizados en estas pruebas. Nótese, como los valores obtenidos en los tests son mucho mayores que los valores críticos respectivos.

Según el test de Holm, podemos verificar que existen diferencias significativas para la mayoría de las técnicas meméticas empleadas, salvo para los algoritmos: MA.Ts.B.A2.Gd, MA.Ts.B*.A2.Gd y MA.Ts.B.A4.Gd, para los cuales no es concluyente que superen la prueba. Como podemos apreciar,

Tabla 4.4: Problema del BIBD: Resultados para los tests de Friedman e Iman-Davenport ($\alpha = 0.05$) para las diferentes técnicas meméticas aplicadas sobre los modelos Primal (B) y Dual (D) (Con/Sin Ruptura de simetrías), B Primal sin ruptura de simetrías, B* Primal con ruptura de simetrías; D Dual sin ruptura de simetrías, y D* Dual con ruptura de simetrías.

	Friedman value	Critical χ^2 value	Iman-Davenport value	Critical F_F value
All 24	1627.027907	35.172462	394.040936	1.534780

las técnicas basadas en el operador de cruce *Gd*, son las que están mostrando la pugna del top 4.

Tabla 4.5: Problema del BIBD: Resultados del test de Holm ($\alpha = 0.05$) utilizando MA.Ts.B*.A4.Gd como algoritmo de control para las diferentes técnicas meméticas aplicadas sobre los modelos Primal (B) y Dual (D) (Con/Sin Ruptura de simetrías), B Primal sin ruptura de simetrías, B* Primal con ruptura de simetrías; D Dual sin ruptura de simetrías, y D* Dual con ruptura de simetrías.

i	strategy	z-statistic	p-value	α/i
1	MA.Ts.B.A2.Gd	1.375e+00	8.459e-02	5.000e-02
2	MA.Ts.B*.A2.Gd	1.375e+00	8.459e-02	2.500e-02
3	MA.Ts.B.A4.Gd	1.596e+00	5.525e-02	1.667e-02
4	MA.Ts.D.A2.Gd	3.208e+00	6.682e-04	1.250e-02
5	MA.Ts.D.A4.Gd	3.208e+00	6.682e-04	1.000e-02
6	MA.Ts.D.A2.Ux	3.278e+00	5.225e-04	8.333e-03
7	MA.Ts.D.A4.Ux	3.370e+00	3.761e-04	7.143e-03
8	MA.Ts.D*.A2.Gd	6.233e+00	2.292e-10	6.250e-03
9	MA.Ts.D*.A4.Ux	6.238e+00	2.214e-10	5.556e-03
10	MA.Ts.D*.A2.Ux	6.244e+00	2.139e-10	5.000e-03
11	MA.Ts.D*.A4.Gd	6.368e+00	9.605e-11	4.545e-03
12	MA.Hc.B*.A4.Gd	8.433e+00	1.691e-17	4.167e-03
13	MA.Hc.B.A4.Gd	9.160e+00	2.585e-20	3.846e-03
14	MA.Hc.B*.A2.Gd	1.008e+01	3.492e-24	3.571e-03
15	MA.Hc.B.A2.Gd	1.102e+01	1.522e-28	3.333e-03
16	MA.Hc.D.A2.Ux	1.421e+01	4.147e-46	3.125e-03
17	MA.Hc.D.A4.Ux	1.436e+01	4.395e-47	2.941e-03
18	MA.Hc.D*.A2.Ux	1.462e+01	1.015e-48	2.778e-03
19	MA.Hc.D.A4.Gd	1.529e+01	4.411e-53	2.632e-03
20	MA.Hc.D*.A4.Ux	1.545e+01	3.632e-54	2.500e-03
21	MA.Hc.D.A2.Gd	1.673e+01	3.945e-63	2.381e-03
22	MA.Hc.D*.A4.Gd	1.797e+01	1.664e-72	2.273e-03
23	MA.Hc.D*.A2.Gd	1.839e+01	8.584e-76	2.174e-03

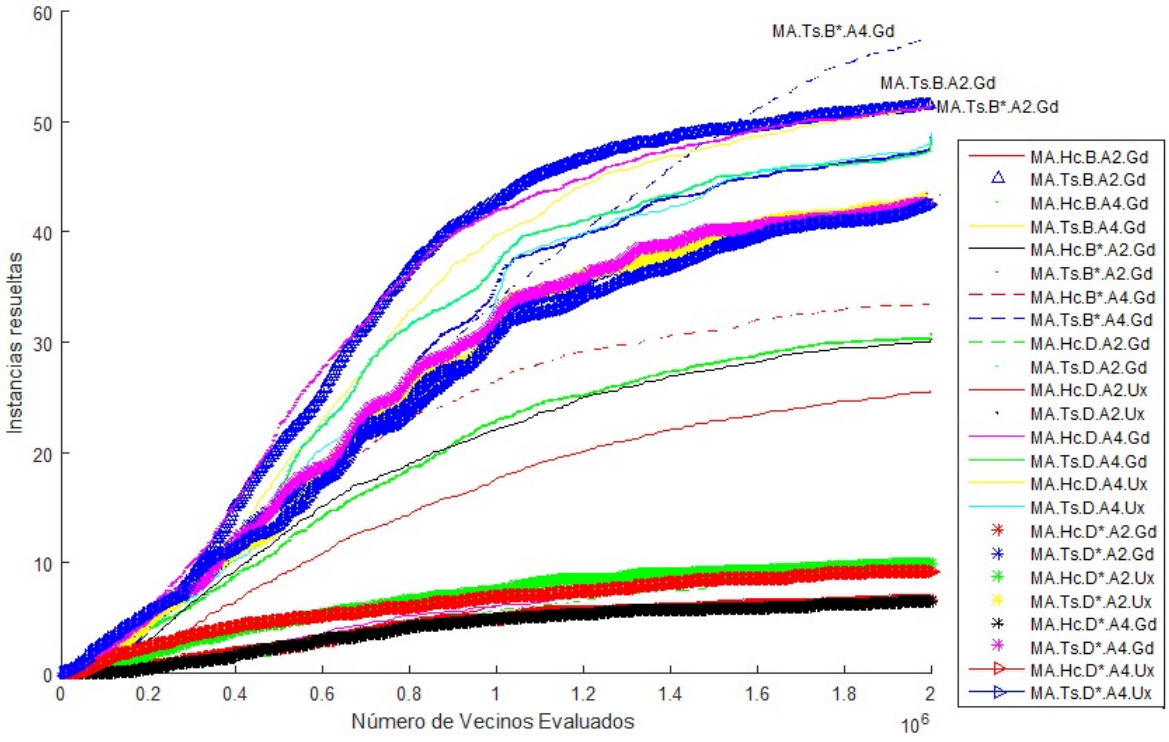
Tabla 4.6: Resumen de resultados del test de significancia estadística para las técnicas meméticas sobre el problema del BIBD para los modelos Primal (B) y Dual (D) (Con/Sin Ruptura de simetrías), B Primal sin ruptura de simetrías, B* Primal con ruptura de simetrías; D Dual sin ruptura de simetrías, y D* Dual con ruptura de simetrías. Cada entrada en la tabla indica el porcentaje de instancias en la cual el algoritmo etiquetado en la fila supera el rendimiento del algoritmo etiquetado en la columna, con una diferencia estadísticamente significativas en el nivel 0.05, según el test de Wilcoxon ranksum.

	MA:Tc:B:A:G:Gd	MA:Tc:B:A:G:Gd	MA:Tc:B:A:G:Ga	MA:Tc:B:A:G:Gd	MA:Tc:B*:A:G:Gd	MA:Tc:B*:A:G:Ga	MA:Tc:B*:A:G:Gd	MA:Tc:D:A:G:Gd	MA:Tc:D:A:G:Ga	MA:Tc:D:A:G:Gd	MA:Tc:D*:A:G:Gd	MA:Tc:D*:A:G:Ga	MA:Tc:D*:A:G:Gd	MA:Tc:D*:A:G:Ga	MA:Tc:D*:A:U:s
MA:Hc:B:A:G:Gd	–	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
MA:Tc:B:A:G:Gd	86.09%	–	69.77%	2.33%	67.44%	1.16%	66.28%	0.00%	91.86%	0.00%	90.70%	0.00%	93.02%	0.00%	90.70%
MA:Hc:B:A:G:Gd	66.28%	15.12%	–	0.00%	32.56%	0.00%	1.16%	0.00%	91.86%	0.00%	90.70%	0.00%	93.02%	0.00%	90.70%
MA:Tc:B:A:G:Gd	86.09%	2.33%	86.05%	–	67.44%	3.49%	66.28%	0.00%	91.86%	37.21%	91.86%	45.35%	90.70%	37.21%	90.70%
MA:Hc:B*:A:G:Gd	41.02%	17.44%	1.16%	16.28%	–	0.00%	0.00%	0.00%	91.86%	0.00%	90.70%	0.00%	90.70%	0.00%	90.70%
MA:Tc:B*:A:G:Gd	86.09%	1.16%	86.05%	13.95%	84.88%	–	66.28%	0.00%	91.86%	34.88%	90.70%	39.53%	90.70%	34.88%	90.70%
MA:Hc:B*:A:G:Gd	73.26%	13.95%	9.30%	13.95%	45.35%	13.95%	–	0.00%	91.86%	0.00%	90.70%	0.00%	93.02%	2.33%	91.86%
MA:Tc:B*:A:G:Gd	86.05%	53.49%	86.05%	54.65%	84.88%	51.16%	81.40%	–	91.86%	50.00%	90.70%	50.00%	93.02%	55.81%	91.86%
MA:Hc:D:A:2:Gd	2.33%	2.33%	1.16%	2.33%	1.16%	2.33%	2.33%	2.33%	–	0.00%	0.00%	0.00%	69.77%	0.00%	0.00%
MA:Tc:D:A2:Gd	86.05%	6.98%	77.91%	9.30%	84.88%	6.98%	72.09%	8.14%	94.19%	–	91.86%	1.16%	90.70%	0.00%	3.49%
MA:Hc:D:A2:1:s	2.33%	2.33%	2.33%	2.33%	2.33%	2.33%	2.33%	2.33%	56.98%	2.33%	–	0.00%	45.35%	0.00%	1.16%
MA:Tc:D:A2:1:s	86.05%	8.14%	81.40%	6.98%	84.88%	5.81%	76.74%	8.14%	94.19%	2.33%	94.19%	–	90.70%	2.33%	91.86%
MA:Hc:D:A4:Gd	1.16%	2.33%	2.33%	2.33%	2.33%	2.33%	2.33%	2.33%	29.07%	2.33%	4.65%	2.33%	–	0.00%	0.00%
MA:Tc:D:A4:Gd	86.05%	6.98%	77.91%	9.30%	84.88%	6.98%	72.09%	8.14%	94.19%	0.00%	94.19%	1.16%	93.02%	–	91.86%
MA:Hc:D:A4:1:s	3.49%	3.49%	3.49%	3.49%	3.49%	3.49%	3.49%	3.49%	63.95%	3.49%	3.49%	3.49%	48.84%	3.49%	–
MA:Tc:D:A4:1:s	86.05%	6.98%	84.88%	6.98%	84.88%	4.65%	77.91%	8.14%	94.19%	2.33%	94.19%	0.00%	93.02%	2.33%	91.86%
MA:Hc:D*:A:2:Gd	3.49%	4.65%	3.49%	4.65%	3.49%	4.65%	4.65%	4.65%	11.63%	4.65%	4.65%	4.65%	6.98%	4.65%	4.65%
MA:Tc:D*:A2:Gd	81.40%	6.98%	58.14%	6.98%	73.26%	6.98%	53.49%	10.47%	94.19%	3.49%	93.02%	2.33%	93.02%	3.49%	95.35%
MA:Hc:D*:A2:1:s	3.49%	3.49%	2.33%	3.49%	3.49%	3.49%	3.49%	3.49%	59.30%	3.49%	1.16%	3.49%	44.19%	3.49%	0.00%
MA:Tc:D*:A2:1:s	82.56%	11.63%	60.47%	8.14%	79.07%	9.30%	56.98%	12.79%	94.19%	5.81%	94.19%	8.14%	93.02%	5.81%	95.35%
MA:Hc:D*:A4:Gd	3.49%	4.65%	3.49%	4.65%	3.49%	4.65%	4.65%	4.65%	12.79%	4.65%	4.65%	4.65%	6.98%	4.65%	4.65%
MA:Tc:D*:A4:Gd	83.72%	10.47%	59.30%	8.14%	76.74%	8.14%	55.81%	11.63%	94.19%	4.65%	93.02%	3.49%	93.02%	4.65%	94.19%
MA:Hc:D*:A4:1:s	3.49%	3.49%	2.33%	3.49%	2.33%	3.49%	3.49%	3.49%	54.65%	3.49%	3.49%	3.49%	36.05%	3.49%	2.33%
MA:Tc:D*:A4:1:s	82.56%	8.14%	56.98%	5.81%	75.58%	8.14%	54.65%	10.47%	94.19%	4.65%	94.19%	1.16%	93.02%	4.65%	95.35%

	MA:Hc:B:A:G:Gd	MA:Tc:B:A:G:Gd	MA:Tc:B:A:G:Ga	MA:Tc:B:A:G:Gd	MA:Tc:B*:A:G:Gd	MA:Tc:B*:A:G:Ga	MA:Tc:B*:A:G:Gd	MA:Tc:D:A:G:Gd	MA:Tc:D:A:G:Ga	MA:Tc:D:A:G:Gd	MA:Tc:D*:A:G:Gd	MA:Tc:D*:A:G:Ga	MA:Tc:D*:A:G:Gd	MA:Tc:D*:A:G:Ga	MA:Tc:D*:A:U:s
MA:Hc:B:A:G:Gd	–	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
MA:Tc:B:A:G:Gd	86.09%	–	69.77%	2.33%	67.44%	1.16%	66.28%	0.00%	91.86%	0.00%	90.70%	0.00%	93.02%	0.00%	90.70%
MA:Hc:B:A:G:Gd	66.28%	15.12%	–	0.00%	32.56%	0.00%	1.16%	0.00%	91.86%	0.00%	90.70%	0.00%	93.02%	0.00%	90.70%
MA:Tc:B:A:G:Gd	86.09%	2.33%	86.05%	–	67.44%	3.49%	66.28%	0.00%	91.86%	37.21%	91.86%	45.35%	90.70%	37.21%	90.70%
MA:Hc:B*:A:G:Gd	41.02%	17.44%	1.16%	16.28%	–	0.00%	0.00%	0.00%	91.86%	0.00%	90.70%	0.00%	90.70%	0.00%	90.70%
MA:Tc:B*:A:G:Gd	86.09%	1.16%	86.05%	13.95%	84.88%	–	66.28%	0.00%	91.86%	34.88%	90.70%	39.53%	90.70%	34.88%	90.70%
MA:Hc:B*:A:G:Gd	73.26%	13.95%	9.30%	13.95%	45.35%	13.95%	–	0.00%	91.86%	0.00%	90.70%	0.00%	93.02%	2.33%	91.86%
MA:Tc:B*:A:G:Gd	86.05%	53.49%	86.05%	54.65%	84.88%	51.16%	81.40%	–	91.86%	50.00%	90.70%	50.00%	93.02%	55.81%	91.86%
MA:Hc:D:A:2:Gd	2.33%	2.33%	1.16%	2.33%	1.16%	2.33%	2.33%	2.33%	–	0.00%	0.00%	0.00%	69.77%	0.00%	0.00%
MA:Tc:D:A2:Gd	86.05%	6.98%	77.91%	9.30%	84.88%	6.98%	72.09%	8.14%	94.19%	–	91.86%	1.16%	90.70%	0.00%	3.49%
MA:Hc:D:A2:1:s	2.33%	2.33%	2.33%	2.33%	2.33%	2.33%	2.33%	2.33%	56.98%	2.33%	–	0.00%	45.35%	0.00%	1.16%
MA:Tc:D:A2:1:s	86.05%	8.14%	81.40%	6.98%	84.88%	5.81%	76.74%	8.14%	94.19%	2.33%	94.19%	–	90.70%	2.33%	91.86%
MA:Hc:D:A4:Gd	1.16%	2.33%	2.33%	2.33%	2.33%	2.33%	2.33%	2.33%	29.07%	2.33%	4.65%	2.33%	–	0.00%	0.00%
MA:Tc:D:A4:Gd	86.05%	6.98%	77.91%	9.30%	84.88%	6.98%	72.09%	8.14%	94.19%	0.00%	94.19%	1.16%	93.02%	–	91.86%
MA:Hc:D:A4:1:s	3.49%	3.49%	3.49%	3.49%	3.49%	3.49%	3.49%	3.49%	63.95%	3.49%	3.49%	3.49%	48.84%	3.49%	–
MA:Tc:D:A4:1:s	86.05%	6.98%	84.88%	6.98%	84.88%	4.65%	77.91%	8.14%	94.19%	2.33%	94.19%	0.00%	93.02%	2.33%	91.86%
MA:Hc:D*:A:2:Gd	3.49%	4.65%	3.49%	4.65%	3.49%	4.65%	4.65%	4.65%	11.63%	4.65%	4.65%	4.65%	6.98%	4.65%	4.65%
MA:Tc:D*:A2:Gd	81.40%	6.98%	58.14%	6.98%	73.26%	6.98%	53.49%	10.47%	94.19%	3.49%	93.02%	2.33%	93.02%	3.49%	95.35%
MA:Hc:D*:A2:1:s	3.49%	3.49%	2.33%	3.49%	3.49%	3.49%	3.49%	3.49%	59.30%	3.49%	1.16%	3.49%	44.19%	3.49%	0.00%
MA:Tc:D*:A2:1:s	82.56%	11.63%	60.47%	8.14%	79.07%	9.30%	56.98%	12.79%	94.19%	5.81%	94.19%	8.14%	93.02%	5.81%	95.35%
MA:Hc:D*:A4:Gd	3.49%	4.65%	3.49%	4.65%	3.49%	4.65%	4.65%	4.65%	12.79%	4.65%	4.65%	4.65%	6.98%	4.65%	4.65%
MA:Tc:D*:A4:Gd	83.72%	10.47%	59.30%	8.14%	76.74%	8.14%	55.81%	11.63%	94.19%	4.65%	93.02%	3.49%	93.02%	4.65%	94.19%
MA:Hc:D*:A4:1:s	3.49%	3.49%	2.33%	3.49%	2.33%	3.49%	3.49%	3.49%	54.65%	3.49%	3.49%	3.49%	36.05%	3.49%	2.33%
MA:Tc:D*:A4:1:s	82.56%	8.14%	56.98%	5.81%	75.58%	8.14%	54.65%	10.47%	94.19%	4.65%	94.19%	1.16%	93.02%	4.65%	95.35%

La curva de rendimiento (figura 4.5), aplicada a las técnicas meméticas, nos muestras el mejor comportamiento para las técnicas MA.Ts.B*.A4.Gd, MA.Ts.B.A2.Gd y MA.Ts.B*.A2.Gd, como ya lo habíamos visto en los tests anteriores, en esta misma sección.

Figura 4.5: Problema del BIBD: Curva de rendimiento de las diferentes técnicas meméticas aplicadas para los modelos Primal (B) y Dual (D) (Con/Sin Ruptura de simetrías), B Primal sin ruptura de simetrías, B* Primal con ruptura de simetrías; D Dual sin ruptura de simetrías, y D* Dual con ruptura de simetrías.



Finalmente, la prueba de suma de rangos, nos indica que el algoritmo con el mejor rendimiento (MA.Ts.B*.A4.Gd) supera en al menos un 50% a las demás técnicas meméticas empleadas en esta experimentación, sobre el problema del BIBD (véase la tabla 4.6).

Tabla 4.7: Problema del BIBD: Número (y porcentaje) de instancias resueltas por las técnicas metaheurísticas básicas e integrativas trabajando en forma separada sobre el conjunto de las 86 instancias tomadas de [32, 246].

algoritmo	# (%)	algoritmo	# (%)
Hc.B	35 (40.70 %)	Ts.B	57 (66.28 %)
Hc.D	6 (6.98 %)	Ts.D	43 (50.00 %)
Hc.B*	25 (29.07 %)	Ts.B*	51 (59.30 %)
Hc.D*	3 (3.49 %)	Ts.D*	46 (53.49 %)
GA.B.A2.Gd	25 (29.07 %)	GA.B.A4.Gd	35 (40.70 %)
GA.D.A2.Gd	35 (40.70 %)	GA.D.A4.Gd	36 (41.86 %)
GA.B*.A2.Gd	38 (44.19 %)	GA.B*.A4.Gd	43 (50.00 %)
GA.D*.A2.Gd	28 (32.56 %)	GA.D*.A4.Gd	31 (36.05 %)
MA.Hc.B.A2.Gd	43 (50.00 %)	MA.Ts.B.A2.Gd	53 (61.63 %)
MA.Hc.B.A4.Gd	46 (53.49 %)	MA.Ts.B.A4.Gd	56 (65.12 %)
MA.Hc.D.B.A2.Gd	14 (16.28 %)	MA.Ts.D.B.A2.Gd	52 (60.47 %)
MA.Hc.D.B.A4.Gd	15 (17.44 %)	MA.Ts.D.B.A4.Gd	52 (60.47 %)
MA.Hc.B*.A2.Gd	43 (50.00 %)	MA.Ts.B*.A2.Gd	53 (61.63 %)
MA.Hc.B*.A4.Gd	46 (53.49 %)	**MA.Ts.B*.A4.Gd**	**59 (68.60 %)**
MA.Hc.D*.A2.Gd	10 (11.63 %)	MA.Ts.D*.A2.Gd	51 (59.30 %)
MA.Hc.D*.A4.Gd	9 (10.47 %)	MA.Ts.D*.A4.Gd	47 (54.65 %)

4.4.3 Análisis de las Técnicas Básicas e Integrativas

Para realizar este análisis hemos considerado treinta y dos algoritmos básicos e integrativos, es decir, 8 algoritmos de búsqueda locales (resultantes de los dos métodos LS considerados en este documento –HC y TS– y las combinaciones resultantes del uso de las representaciones primal/dual y con/sin ruptura de simetrías, B Primal sin ruptura de simetrías, B* Primal con ruptura de simetrías; D Dual sin ruptura de simetrías, y D* Dual con ruptura de simetrías), 8 GA (resultantes de los cuatro anteriores escenarios más el uso de dos aridades diferentes y el uso del operador de cruce voraz) y 16 MAs (como resultado de dotar a cada uno de los GA anteriores, con HC o TS para realizar mejoras locales).

Los resultados de rendimiento obtenidos por todas estas metaheurísticas se muestran en la tabla 4.7. En general, las variantes de los algoritmos TS superan a sus contrapartes HC y las versiones de GA. Más específicamente, se puede ver que la versión del algoritmo TS que trabaja en el entorno *swap*, bajo la representación primal (identificado como $Ts.B$) presenta el mayor número de instancias resueltas (lo cual no es de sorprender, ya que confirma los resultados mostrados [268]). No obstante, hemos encontrado un resultado interesante: el algoritmo memético que hace uso de ruptura de simetrías, bajo la representación primal, con operador de mejora local TS, usando 4 padres y operador de cruce *voraz*, obtiene el mayor número de instancias resueltas (con respecto de su contraparte: $MA.Ts.B.A4.Gd$, que no utiliza la ruptura de simetrías). Así, $MA.Ts.B*.A4.Gd$ resuelve un total de 59 instancias versus las 56 instancias que logra resolver su equivalente sin ruptura de simetrías.

Debido a la gran cantidad de variantes de algoritmos, no es fácil determinar el rendimiento de cada uno de ellos en comparación con el resto, simplemente con revisar el número de instancias resueltas, por lo que hemos optado por un torneo basado en rangos. Más precisamente, hemos obtenido el rango r_j^i de cada grupo de algoritmos j en cada instancia i (se le asigna el valor 1 a la mejor técnica, y el

Tabla 4.8: Resultados de los tests de Friedman e Iman-Davenport ($\alpha = 0.05$), para las diferentes técnicas metaheurísticas básicas e integrativas (32 en total), aplicadas sobre el problema del BIBD para las 86 instancias tomadas de [32, 246], para los modelos Primal (B) y Dual (D) (Con/Sin Ruptura de simetrías), B Primal sin ruptura de simetrías, B* Primal con ruptura de simetrías; D Dual sin ruptura de simetrías, y D* Dual con ruptura de simetrías.

	Friedman value	Critical χ^2 value	Iman-Davenport value	Critical F_F value
All 32	2179.571749	44.985343	380.865212	1.455540

rango k a la peor de ellas, k es el número de algoritmos utilizados en el torneo el cual corresponde a 32 para este caso particular).

Inicialmente, en todos los casos, habíamos utilizado para fines de clasificación sólo la cantidad de soluciones alcanzadas en cada instancia (del conjunto de 30 ejecuciones) pero notamos que al hacerlo no consideramos la solidez de los algoritmos (es decir, su capacidad para encontrar soluciones candidatas cercanas al óptimo). Entonces, al final, hemos utilizado la cantidad de soluciones encontradas en cada instancia y hemos empleado el *fitness* medio para tratar los empates. La distribución del torneo se muestra en la figura 4.6. A simple vista, los algoritmos integrativos (i.e. las versiones de MAs) parecen mostrar el mejor rendimiento, respecto de sus contrapartes básicas. Los resultados confirman que las versiones de MA con ruptura de simetrías (i.e. $MA.Ts.B*.A4.Gd$) supera a la versión que no utiliza la ruptura de simetrías. Un análisis estadístico más detallado indica que existen diferencias significativas entre los diferentes algoritmos de acuerdo con los tests de Friedman e Iman-Davenport (véase la tabla 4.8). Los resultados para el test de Holm la podemos visualizar en la tabla 4.9, en la cual podemos observar el rendimiento de estas técnicas, estadísticamente hablando. Debemos mencionar que la versión del algoritmo $Ts.B$ es la técnica básica que parece mostrar el mejor desempeño, mientras que los otros cuatro algoritmos en estas colecciones representan los mejores métodos integrativos a lo largo de los dominios B*, B, D y D*.

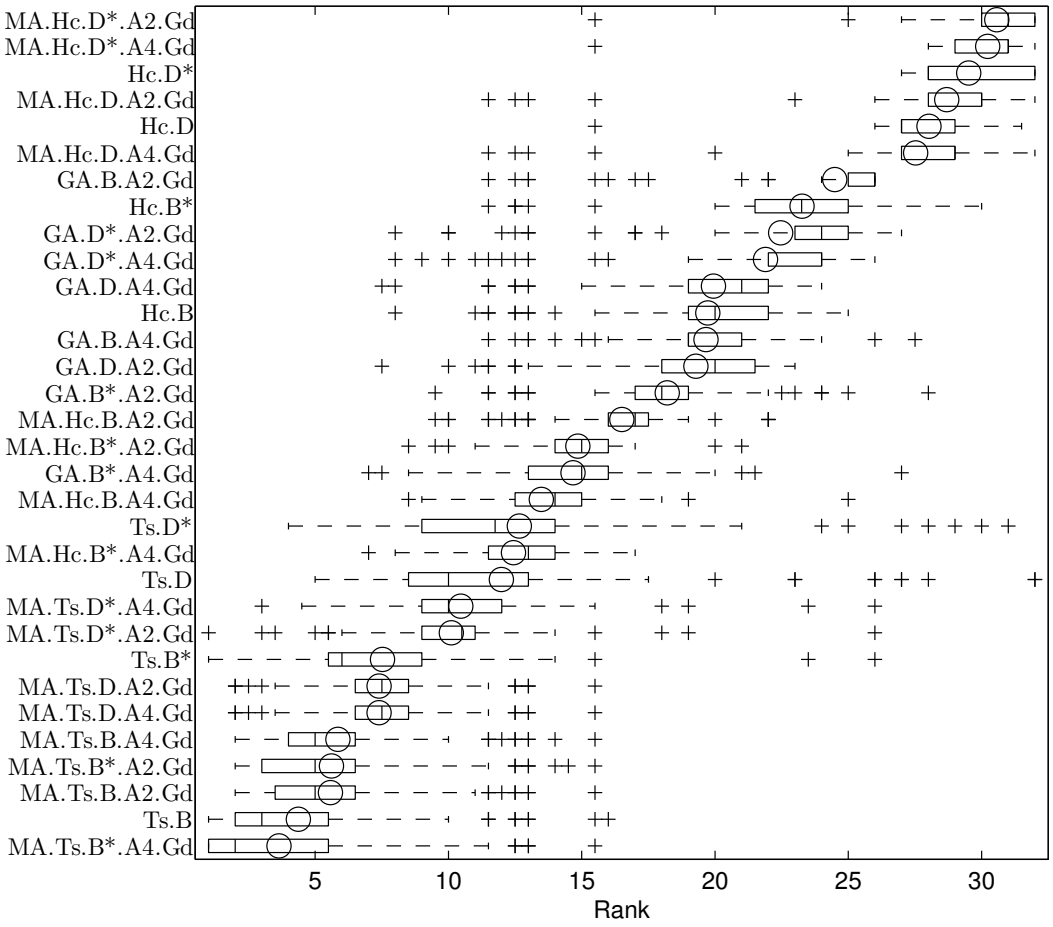
Algunas aspectos interesantes surgen cuando los datos se analizan a lo largo de dimensiones particulares. Consideremos un análisis conjunto de los modelos de representación y el uso de técnicas par la ruptura de simetrías, dividiendo los datos en cuatro grupos: {primal, dual } $\times$ {con ruptura de simetrías, sin ruptura de simetrías } (es decir: B, B*, D y D*). Como se puede ver en la figura 4.7, el uso del modelo primal con ruptura de simetrías produce el mejor rango global y supera significativamente a las otras combinaciones. Los tests de Friedman e Iman-Davenport confirman que existen diferencias estadísticamente significativas (en un valor estándar $\alpha = 0.05$) entre los diferentes grupos (véase la tabla 4.10). Los resultados del test de Holm aplicado, usando $\mathbf{B}^*$ como algoritmo de control, corroboran la tesis de que existen diferencias estadísticas significativas ($\alpha = 0.05$) con respecto a los demás algoritmos (es decir, en todos los casos, valor p $< \alpha/i$) (véase la tabla 4.11).

4.4.4 Resultados Experimentales de las Técnicas Cooperativas sobre el BIBD

En esta sección se evalúa el rendimiento de los diferentes esquemas de colaboración que han sido implementados para resolver el problema del BIBD, considerando los diferentes aspectos que ya se han mencionado en las secciones previas 2.6.3, 4.3 y 4.3.3.

La idea principal es el aprovechamiento de la sinergia entre las metaheurísticas cuando estas fun-

Figura 4.6: Torneo para las diferentes técnicas metaheurísticas básicas e integrativas (32 en total), aplicadas sobre el problema del BIBD para las 86 instancias tomadas de [32, 246], para los modelos Primal (B) y Dual (D) (Con/Sin Ruptura de simetrías), B Primal sin ruptura de simetrías, B* Primal con ruptura de simetrías; D Dual sin ruptura de simetrías, y D* Dual con ruptura de simetrías. Cada caja comprende desde el primer al tercer cuartil de la distribución, la mediana (2do cuartil) es indicada con una línea vertical, la zona sin valores atípicos (whiskers span) es de 1.5 veces el rango intercuartil, además los valores atípicos están marcados con un signo '+'.



cionan bajo un esquema colaborativo. Hemos considerado las tres topologías propuestas, descritas anteriormente en la sección 4.3.1, con un número variable de n de agentes entre 2 y 5 (según lo descrito en [11]), un número de ciclos $\Theta = 5$ (este valor ha sido establecido con base a los resultados obtenidos, luego de realizar pruebas preliminares, donde se consideró variar este parámetro con valores de $\Theta \in \{5, 10, 15\}$). Los modelos cooperativos considerados son variaciones de la configuración $\mathbf{T}n(pa, qb)$MR, donde $p+q=n$ y $a, b \in \mathcal{A}$ para una cierta colección $\mathcal{A}$ que contiene dos de los métodos considerados. Hemos empleado las siguientes cuatro colecciones de este tipo:

- $\mathcal{A}_1 = \{$Ts.B,MA.Ts.B*.A4.Gd$\}$
- $\mathcal{A}_2 = \{$Ts.B,MA.Ts.B.A2.Gd$\}$
- $\mathcal{A}_3 = \{$Ts.B,MA.Ts.D.A4.Gd$\}$

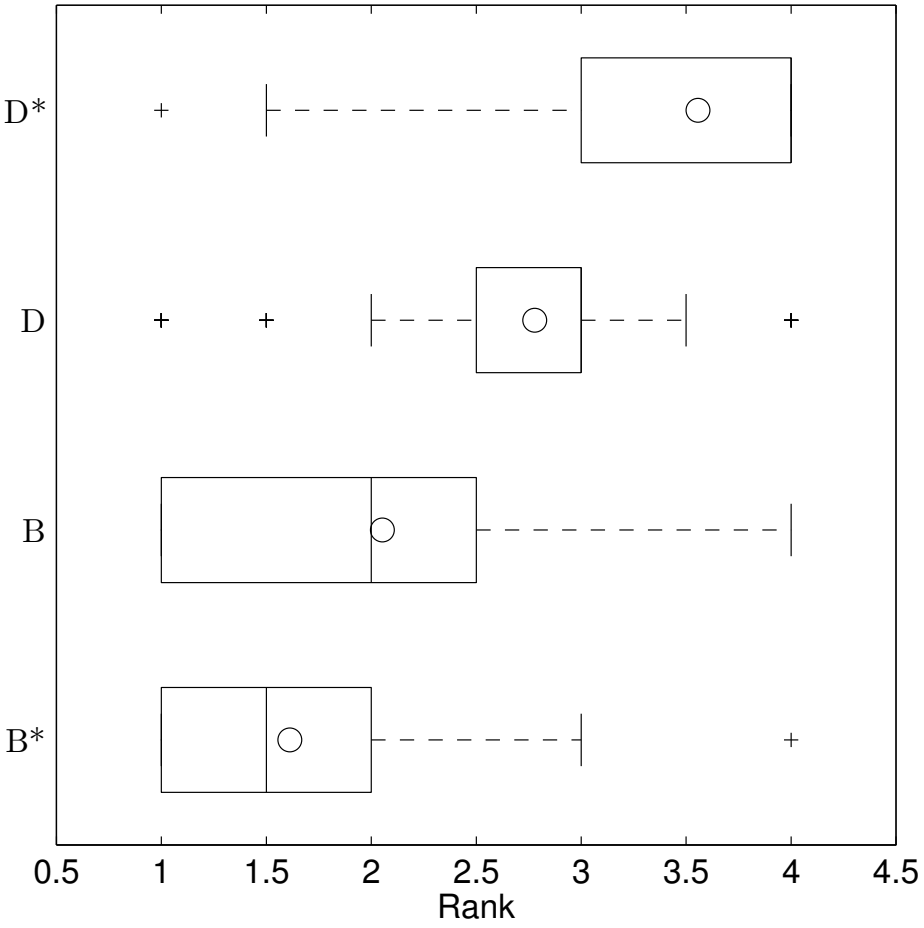
Tabla 4.9: Problema del BIBD: Resultados del test de Holm aplicado sobre las técnicas básicas e integrativas propuestas (32 en total), para los modelos Primal (B) y Dual (D) (Con/Sin Ruptura de simetrías), B Primal sin ruptura de simetrías, B* Primal con ruptura de simetrías; D Dual sin ruptura de simetrías, y D* Dual con ruptura de simetrías, usando MA.Ts.B*.A4.Gd como algoritmo de control - en el nivel estándar de $\alpha = 0.05$.

i	algoritmo	z-statistic	p-value	α/i
1	Ts.B	5.039e-001	3.071e-001	5.000e-002
2	MA.Ts.B.A2.Gd	1.345e+000	8.928e-002	2.500e-002
3	MA.Ts.B*.A2.Gd	1.374e+000	8.477e-002	1.667e-002
4	MA.Ts.B.A4.Gd	1.544e+000	6.125e-002	1.250e-002
5	MA.Ts.D.A4.Gd	2.621e+000	4.379e-003	1.000e-002
6	MA.Ts.D.A2.Gd	2.621e+000	4.379e-003	8.333e-003
7	Ts.B*	2.711e+000	3.357e-003	7.143e-003
8	MA.Ts.D*.A2.Gd	4.519e+000	3.103e-006	6.250e-003
9	MA.Ts.D*.A4.Gd	4.767e+000	9.341e-007	5.556e-003
10	Ts.D	5.828e+000	2.806e-009	5.000e-003
11	MA.Hc.B*.A4.Gd	6.145e+000	4.000e-010	4.545e-003
12	Ts.D*	6.299e+000	1.494e-010	4.167e-003
13	MA.Hc.B.A4.Gd	6.876e+000	3.068e-012	3.846e-003
14	GA.B*.A4.Gd	7.710e+000	6.311e-015	3.571e-003
15	MA.Hc.B*.A2.Gd	7.836e+000	2.333e-015	3.333e-003
16	MA.Hc.B.A2.Gd	8.986e+000	1.285e-019	3.125e-003
17	GA.B*.A2.Gd	1.018e+001	1.211e-024	2.941e-003
18	GA.D.A2.Gd	1.093e+001	4.218e-028	2.778e-003
19	GA.B.A4.Gd	1.119e+001	2.218e-029	2.632e-003
20	Hc.B	1.124e+001	1.278e-029	2.500e-003
21	GA.D.A4.Gd	1.139e+001	2.300e-030	2.381e-003
22	GA.D*.A4.Gd	1.276e+001	1.421e-037	2.273e-003
23	GA.D*.A2.Gd	1.315e+001	8.349e-040	2.174e-003
24	Hc.B*	1.371e+001	4.286e-043	2.083e-003
25	GA.B.A2.Gd	1.457e+001	2.186e-048	2.000e-003
26	MA.Hc.D.A4.Gd	1.669e+001	7.587e-063	1.923e-003
27	Hc.D	1.704e+001	2.046e-065	1.852e-003
28	MA.Hc.D.A2.Gd	1.751e+001	6.209e-069	1.786e-003
29	Hc.D*	1.809e+001	2.082e-073	1.724e-003
30	MA.Hc.D*.A4.Gd	1.859e+001	2.120e-077	1.667e-003
31	MA.Hc.D*.A2.Gd	1.882e+001	2.548e-079	1.613e-003

- $\mathcal{A}_4 = \{$Ts.B,MA.Ts.D*.A2.Gd$\}$

Los algoritmos en estas colecciones, no ha sido seleccionados en forma arbitraria sino que por el contrario, los hemos escogidos debido a su buen desempeño al operar en forma individual, de acuerdo con la tabla 4.9: así que, Ts.B es técnica básica que parece mostrar el mejor desempeño, mientras que los otros cuatro algoritmos en estas colecciones representan los mejores métodos integrativos a lo largo de los dominios B*, B, D y D*. Además, cada colección representa una posible forma de re-

Figura 4.7: Torneo para las diferentes técnicas metaheurísticas básicas e integrativas, aplicadas sobre el problema del BIBD para las 86 instancias tomadas de [32, 246], reunidas en cuatro grupos: {primal, dual}×{sin ruptura de simetrías, con ruptura de simetrías}. B identifica el modelo de representación Primal sin ruptura de simetrías, B* modelo Primal con ruptura de simetrías; D el modelo Dual sin ruptura de simetrías, y D* el modelo Dual con ruptura de simetrías. Cada caja comprende desde el primer al tercer cuartil de la distribución, la mediana (2do cuartil) es indicada con una línea vertical, la zona sin valores atípicos (whiskers span) es de 1.5 veces el rango intercuartil, además los valores atípicos están marcados con un signo '+'.



combinar las diferentes técnicas que han sido consideradas en esta investigación: así, $\mathcal{A}_1$ representa la cooperación de uno de los esquemas de representación alternativa (en este caso, el modelo que hemos llamado Binario) con las correspondientes configuraciones de con/sin ruptura de simetría (es decir, B-B*), $\mathcal{A}_2$ representa la cooperación de técnicas que trabajan en los mismos dominios de cómputo sin ruptura de simetría (es decir, B-B), $\mathcal{A}_3$ representa la cooperación de métodos que trabajan en los distintos dominios de computo sin ruptura de simetría (es decir, B-D) y $\mathcal{A}_4$ el esquema en el que los métodos trabajan en dominios de computo diferentes con políticas distintas para la ruptura de simetría (es decir, B-D*).

Tomando en cuenta todas las posibles combinaciones de topología, número de agentes, así como las diferentes políticas de migración/recepción consideradas, se diseñaron un total de 288 variantes de las técnicas empleadas. Estas 288 variantes resultan de sumar todos los algoritmos que se pueden

Tabla 4.10: Resultados de los tests de Friedman e Iman-Davenport, aplicado sobre las diferentes técnicas metaheurísticas básicas e integrativas, aplicadas sobre el problema del BIBD para las 86 instancias tomadas de [32, 246], reunidas en cuatro grupos: {primal, dual}×{sin ruptura de simetrías, con ruptura de simetrías}. B identifica el modelo de representación Primal sin ruptura de simetrías, B* modelo Primal con ruptura de simetrías; D el modelo Dual sin ruptura de simetrías, y D* el modelo Dual con ruptura de simetrías.

	Friedman value	Critical χ^2 value	Iman-Davenport value	Critical F_F value
All 4	900.940552	7.814728	532.170699	2.609220

Tabla 4.11: Resultados del test de Holm, aplicado sobre las diferentes técnicas metaheurísticas básicas e integrativas, aplicadas sobre el problema del BIBD para las 86 instancias tomadas de [32, 246], reunidas en cuatro grupos: {primal, dual}×{sin ruptura de simetrías, con ruptura de simetrías}. B identifica el modelo de representación Primal sin ruptura de simetrías, B* modelo Primal con ruptura de simetrías; D el modelo Dual sin ruptura de simetrías, y D* el modelo Dual con ruptura de simetrías, usando $B*$ como algoritmo de control.

i	algorithm	z-statistic	p-value	α/i
1	B	6.338e+000	1.167e-010	5.000e-002
2	D	1.676e+001	2.492e-063	2.500e-002
3	D*	2.794e+001	4.391e-172	1.667e-002

generar al combinar: (1) dos modelos diferentes de representación (es decir primal, dual), (2) dos formas distintas de gestionar la resolución de problemas (es decir, con o sin ruptura de simetría) , (3) tres topologías de comunicación entre los agentes (es decir, RING, RAND o BROADCAST), (4) seis reglas para las políticas de migración/recepción, y (5) 4 posibles formas de combinación para cargar los algoritmos en los agentes, en un total de 4 esquema de agentes (es decir, colecciones $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ y $\mathcal{A}_4$). Así tendríamos que $2 \times 2 \times 3 \times 6 \times 4 = 288$.

Debido a la gran cantidad algoritmos que han resultado de las combinaciones descritas anteriormente, hemos considerado un conjunto reducido de 29 instancias para evaluar el rendimiento de estas técnicas. De igual manera, este grupo no ha sido seleccionado en forma arbitraria, sino que es el resultado de seleccionar, del conjunto anterior de 86 instancias, sólo aquellas que no pudieron ser resueltas por los métodos metaheurísticos (no constructivos) reportados en la literatura científica como se menciona en las secciones anteriores (véase la sección 3.3). Estas 29 instancias, consideradas duras, se pueden ver en la tabla 4.12.

4.4.4.1 Análisis de Factores de Diseño de los Modelos Cooperativos del BIBD

Dada la gran cantidad de algoritmos que se han abordado como resultado de las diferentes combinaciones resultantes, es conveniente realizar un análisis de diferentes aspectos a lo largo de un conjunto de dimensiones correspondientes a la variedad de decisiones de diseño con respecto al número de agentes involucrados, su topología o las políticas de comunicación. Comencemos con este último,

Tabla 4.12: 29 instancias del problema del BIBD consideradas muy difíciles de resolver, tomadas de entre las 86 instancias presentadas en [32, 246]. La primera columna corresponde a la identificación de la instancia indicada en [246], las columnas desde la 2–6 muestran los parámetros correspondientes a la instancia, y la columna 7 nos indica el tamaño de la instancia respectiva.

ID	v	b	r	k	λ	vb
21	14	26	13	7	6	364
27	15	30	14	7	6	450
28	16	30	15	8	7	480
33	16	32	12	6	4	512
34	15	35	14	6	5	525
39	17	34	16	8	7	578
43	18	34	17	9	8	612
44	25	25	9	9	3	625
46	21	30	10	7	3	630
48	16	40	15	6	5	640
50	15	45	21	7	9	675
54	19	38	18	9	8	722
56	22	33	12	8	4	726
57	14	52	26	7	12	780
58	27	27	13	13	6	729

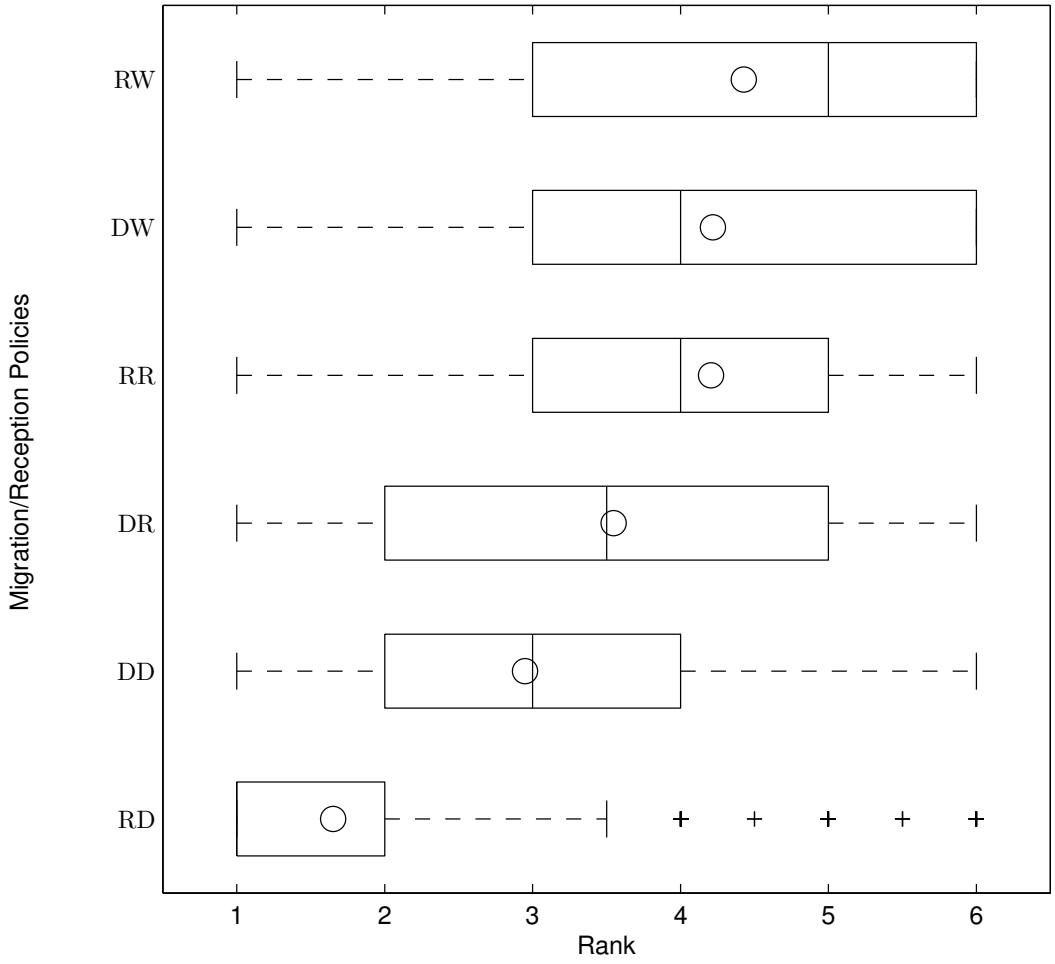
ID	v	b	r	k	λ	vb
59	21	35	15	9	6	735
62	20	38	19	10	9	760
63	16	48	18	6	6	768
70	21	42	10	5	2	882
71	21	42	12	6	3	882
72	21	42	20	10	9	882
73	16	56	21	6	7	896
76	18	51	17	6	5	918
77	22	42	21	11	10	924
80	16	60	30	8	14	960
82	31	31	10	10	3	961
83	31	31	15	15	7	961
85	22	44	14	7	4	968
86	25	40	16	10	6	1000

considerando las seis combinaciones (es decir, DD, DR, DW, RD, RR y RW) utilizadas.

Hemos realizado una revisión de los datos a lo largo de muchas dimensiones, pero consideramos necesario comparar los resultados de estas seis políticas en una única variante algorítmica. Más precisamente, para cada una de las 288/6 = 48 variantes (resultantes de diferentes combinaciones de topología, número de agentes y algoritmos individuales utilizados), clasificamos las seis políticas de migración/recepción (M/R) según el número de soluciones óptimas encontradas (usando el fitness medio para romper con aquellos resultados que son muy similares). La figura 4.8 muestra la distribución resultante del torneo para la diferentes políticas empleadas. Según el test aplicado, la mejor política de M/R es RD (realizar envío de información, es decir solución candidata, seleccionada al azar, reemplazo de nuevas soluciones considerando si esta diversifica la población). Esto se confirma con las pruebas de Friedman e Iman-Davenport (tabla 4.13, en el nivel estándar de $\alpha = 0.05$), esto indica que hay diferencias estadísticamente significativas entre las políticas, lo cual es corroborado al aplicar el test de Holm (tabla 4.14). Aquí se muestra que RD es significativamente mejor que las políticas restantes.

Ahora, vamos a considerar realizar un análisis a lo largo del eje de las topologías. Para este aspecto, vamos a poner a competir las tres topologías mencionadas, para las 288/3=96 variantes de algoritmos resultantes. Los resultados que podemos ver en la tabla 4.15, corresponde al test de Friedman e Iman-Davenport ($\alpha = 0.05$) sobre las topologías estudiadas, como podemos apreciar, parece haber diferencias estadísticamente significativas ya que los valores estadísticos encontrados están por encima del nivel crítico. De hecho, la topología BROADCAST parece mostrar un mejor desempeño frente a las topologías restantes, esto se puede corroborar fácilmente, por los resultados obtenidos con el test de Holm para las topologías utilizadas que emplean BROADCAST como algoritmo de control indicado en la tabla 4.16.

Figura 4.8: Torneo para las diferentes técnicas colaborativas, aplicadas sobre el problema del BIBD para las 29 instancias de la tabla 4.12, agrupadas a lo largo de la dimensión: políticas de migración/recepción y reemplazo. Considerando las seis combinaciones (es decir, DD, DR, DW, RD, RR y RW) utilizadas (véase la sección 4.4.1) . Cada caja comprende desde el primer al tercer cuartil de la distribución, la mediana (2do cuartil) es indicada con una línea vertical, la zona sin valores atípicos (whiskers span) es de 1.5 veces el rango intercuartil, además los valores atípicos están marcados con un signo '+'.



Esta topología parece contribuir a la difusión parsimoniosa de la información entre los agentes, que en este caso resulta ser globalmente más apropiada. Ahora dirigimos nuestra atención a la cantidad de agentes utilizados en el modelo cooperativo.

Hemos considerado $n \in [2,5]$ y los cuatro valores de este parámetro se clasifican en 288/4 = 72 diferentes variantes de los algoritmos empleados en esta investigación. Tanto las pruebas de Friedman como las pruebas de Iman-Davenport indican que se presentan diferencias significativas entre los diferentes valores utilizados, – vea la tabla 4.17.

Para afianzar de mejor manera los argumentos aquí expuestos hemos optado por realizar el test de Holm, usando $n = 2$ (el mejor valor de la clasificación) como algoritmo de control. Como se muestra

Tabla 4.13: Resultados de los tests de Friedman e Iman-Davenport ($\alpha = 0.05$) aplicadas sobre las diferentes técnicas colaborativas del problema del BIBD para las 29 instancias de la tabla 4.12, agrupadas a lo largo de la dimensión: políticas de migración/recepción y reemplazo. Considerando las seis combinaciones (es decir, DD, DR, DW, RD, RR y RW) utilizadas (véase la sección 4.4.1).

	Friedman value	Critical χ^2 value	Iman-Davenport value	Critical F_F value
All 6	2229.396552	11.070498	655.538059	2.215385

Tabla 4.14: Resultados del test de Holm ($\alpha = 0.05$) aplicadas sobre las diferentes técnicas colaborativas del problema del BIBD para las 29 instancias de la tabla 4.12, agrupadas a lo largo de la dimensión: políticas de migración/recepción y reemplazo. Considerando las seis combinaciones (es decir, DD, DR, DW, RD, RR y RW) utilizadas (véase la sección 4.4.1), usando RD como algoritmo de control.

i	algorithm	z-statistic	p-value	α/i
1	DD	1.830e+001	4.477e-075	5.000e-002
2	DR	2.675e+001	5.446e-158	2.500e-002
3	RR	3.604e+001	1.015e-284	1.667e-002
4	DW	3.623e+001	1.156e-287	1.250e-002
5	RW	3.917e+001	0.000e+000	1.000e-002

Tabla 4.15: Resultados de los tests de Friedman e Iman-Davenport ($\alpha = 0.05$) para las diferentes técnicas colaborativas, aplicadas sobre el problema del BIBD para las 29 instancias de la tabla 4.12, agrupadas a lo largo de la dimensión: Topologías (Ra, Ri y BC, véase la sección 4.4.1).

	Friedman value	Critical χ^2 value	Iman-Davenport value	Critical F_F value
All 3	162.643499	5.991465	83.738576	2.997345

Tabla 4.16: Resultados del test de Holm, para las diferentes técnicas colaborativas, aplicadas sobre el problema del BIBD para las 29 instancias de la tabla 4.12, agrupadas a lo largo de la dimensión: Topologías (Ra, Ri y BC, véase la sección 4.4.1), usando **Broadcast** como algoritmo de control.

i	algorithm	z-statistic	p-value	α/i
1	Rand	2.225e+000	1.305e-002	5.000e-002
2	Ring	1.199e+001	2.064e-033	2.500e-002

en la tabla 4.18, el resultado ha sido que la diferencia es significativa frente a los valores restantes de n.

Interpretamos este resultado como una señal que nos indica que la cantidad de agentes es un factor

Figura 4.9: Torneo para las diferentes técnicas colaborativas, aplicadas sobre el problema del BIBD para las 29 instancias de la tabla 4.12, agrupadas a lo largo de la dimensión: Topologías (Ra, Ri y BC, véase la sección 4.4.1). Cada caja comprende desde el primer al tercer cuartil de la distribución, la mediana (2do cuartil) es indicada con una línea vertical, la zona sin valores atípicos (whiskers span) es de 1.5 veces el rango intercuartil, además los valores atípicos están marcados con un signo '+'.

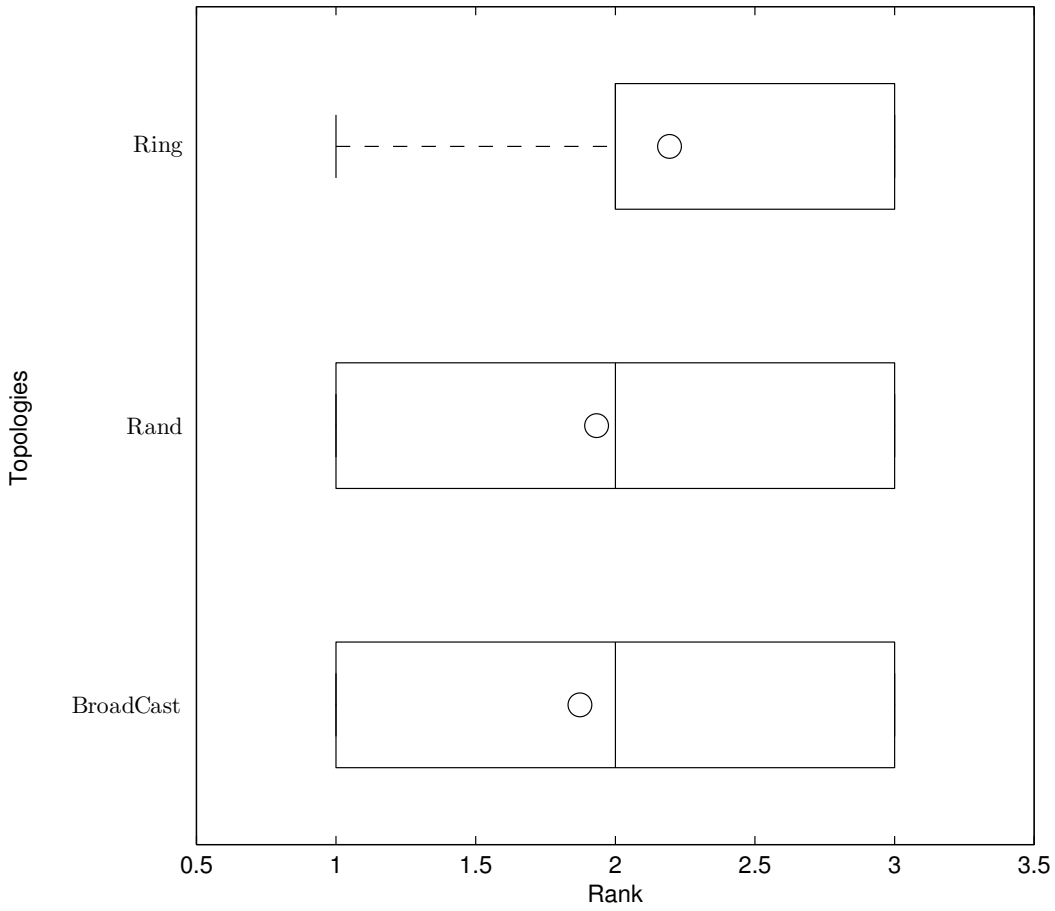
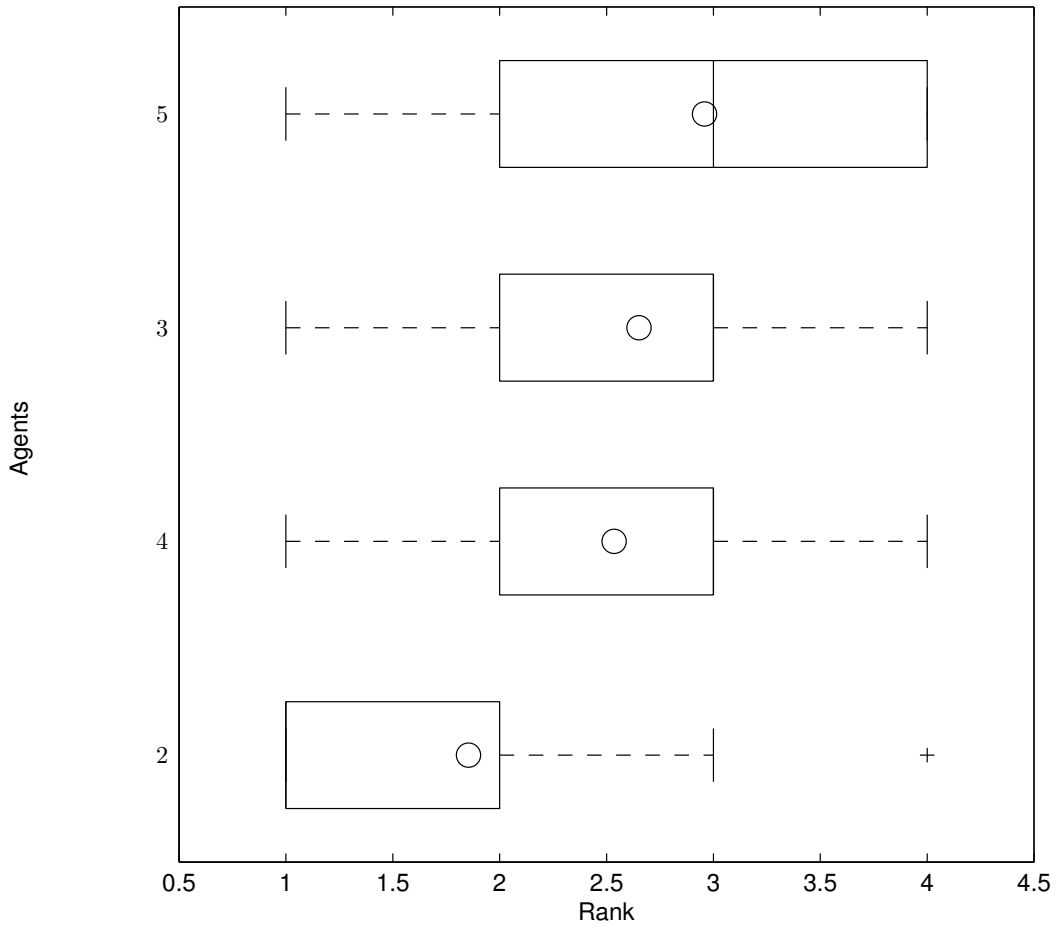


Tabla 4.17: Resultados de los tests de Friedman e Iman-Davenport ($\alpha = 0.05$) para las diferentes técnicas colaborativas, aplicadas sobre el problema del BIBD para las 29 instancias de la tabla 4.12, agrupadas a lo largo de la dimensión: Número de agentes ($n \in [2,5]$).

	Friedman value	Critical χ^2 value	Iman-Davenport value	Critical F_F value
All 4	817.445833	7.814728	313.227300	2.606327

que ejerce una influencia significativa en el rendimiento del modelo cooperativo, lo que sugiere que un gran número de ellos puede contribuir a diversificar la búsqueda a expensas de una intensificación más fuerte, llevando al deterioro en su rendimiento.

Figura 4.10: Torneo para las diferentes técnicas colaborativas, aplicadas sobre el problema del BIBD para las 29 instancias de la tabla 4.12, agrupadas a lo largo de la dimensión: Número de agentes ($n \in [2,5]$). Cada caja comprende desde el primer al tercer cuartil de la distribución, la mediana (2do cuartil) es indicada con una línea vertical, la zona sin valores atípicos (whiskers span) es de 1.5 veces el rango intercuartil, además los valores atípicos están marcados con un signo '+'.



.

Tabla 4.18: Resultados del test de Holm ($\alpha = 0.05$) para las diferentes técnicas colaborativas, aplicadas sobre el problema del BIBD para las 29 instancias de la tabla 4.12, agrupadas a lo largo de la dimensión: Número de agentes ($n \in [2,5]$), usando $n = 2$ como algoritmo de control.

i	algorithm	z-statistic	p-value	α/i
1	$n = 4$	1.707e+001	1.141e-065	5.000e-002
2	$n = 3$	1.997e+001	5.058e-089	2.500e-002
3	$n = 5$	2.766e+001	1.086e-168	1.667e-002

Como podemos observar en la figura del torneo 4.10, el grupo que corresponde a las combinaciones de algoritmos $\mathcal{A}_2$ ($\{Ts.B, MA.Ts.B.A2.Gd\}$), obtienen la mejor posición en el torneo. Al parecer, para este problema en particular, la combinación de agentes basados en el modelo Binario (Primal), sin ruptura de simetrías, en donde trabajan de forma mancomunada una búsqueda local y

Figura 4.11: Torneo para las diferentes técnicas colaborativas, aplicadas sobre el problema del BIBD para las 29 instancias de la tabla 4.12, considerando los grupos de agentes: $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ y $\mathcal{A}_4$, véase la sección 4.4.4.

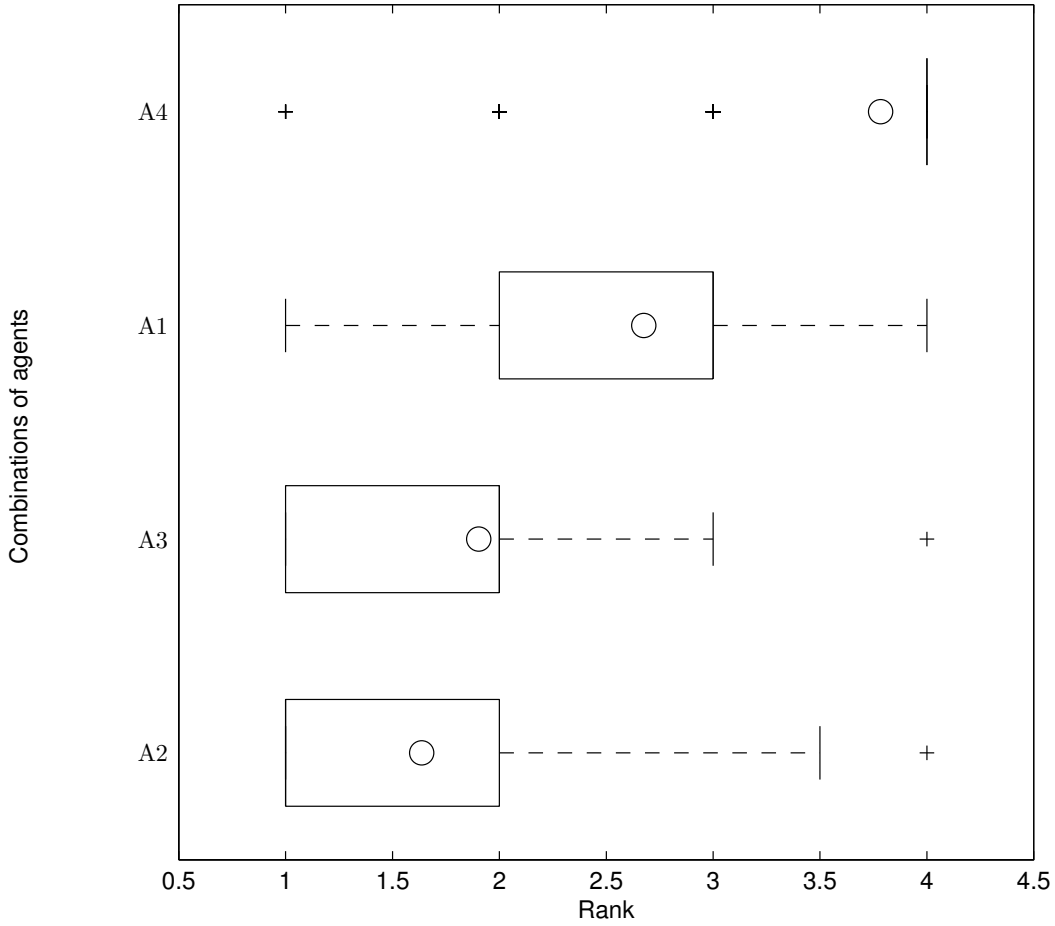


Tabla 4.19: Resultados de los tests de Friedman e Iman-Davenport ($\alpha = 0.05$) para las diferentes técnicas colaborativas, aplicadas sobre el problema del BIBD para las 29 instancias de la tabla 4.12, considerando los grupos de agentes: $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ y $\mathcal{A}_4$, véase la sección 4.4.4.

	Friedman value	Critical χ^2 value	Iman-Davenport value	Critical F_F value
All 4	3483.078879	7.814728	2613.948870	2.606327

un algoritmo memético, obtienen los mejores resultados, como se puede ver en los tests aplicados y cuyos resultados se pueden ver en las tablas 4.19 y 4.20. Los tests muestran la superioridad de la combinación $\mathcal{A}_2$, pues muestran diferencia estadísticamente significativas con respecto a las demás combinaciones empleadas en esta investigación.

Tabla 4.20: Resultados del test de Holm ($\alpha = 0.05$) para las diferentes técnicas colaborativas, aplicadas sobre el problema del BIBD para las 29 instancias de la tabla 4.12, considerando los grupos de agentes: $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ y $\mathcal{A}_4$, véase la sección 4.4.4, usando $\mathcal{A}_2$ como algoritmo de control.

i	algorithm	z-statistic	p-value	α/i
1	$\mathcal{A}_3$	6.683e+000	1.174e-011	5.000e-002
2	$\mathcal{A}_1$	2.600e+001	2.548e-149	2.500e-002
3	$\mathcal{A}_4$	5.374e+001	0.000e+000	1.667e-002

4.4.4.2 Análisis de los Mejores Modelos Cooperativos del BIBD

Ahora nos enfocamos en los modelos cooperativos más efectivos. Más precisamente, en los siguientes párrafos, consideramos todos los modelos cooperativos que pudieron resolver al menos 9 de las instancias problemáticas (de las 29 mencionadas anteriormente) en al menos una ejecución. Este conjunto de algoritmos está compuesto por 41 variantes (los resultados se pueden ver en la tabla 4.21). Esta tabla muestra el número de instancias del problema resueltas y el porcentaje de éxito correspondiente.

Como podemos observar, los diseños correspondientes al grupo $\mathcal{A}_2$ (algoritmos **Ri2(Ts.B,MA.Ts.B.A2.Gd)DR y Ra3(2Ts.B,MA.Ts.B.A2.Gd)RD** tienen la mayor ventaja, en cuanto al número de instancias resueltas se refiere, pues logran resolver el 44.83% del total de las 29 instancias consideradas muy difíciles de resolver.

Si comprobamos la frecuencia relativa de cada parámetro de diseño particular entre estos 41 modelos algorítmicos seleccionados, obtenemos los resultados que se muestran en la tabla 4.22. Los resultados son consistentes con el análisis estadístico de la sección anterior. Por lo tanto, podemos ver que la política de RD para migración/recepción muestra una mayor presencia en los modelos algorítmicos. En cuanto a la topología, BROADCAST está presente en más de la mitad de los modelos. Finalmente, como se puede notar, un bajo número de agentes parece ofrecer un buen rendimiento con mayor frecuencia, según las pruebas realizadas.

La figura 4.12 muestra la distribución de rango correspondiente para estos 41 variaciones de modelos. Se ejecutó un análisis estadístico al aplicar los tests estadísticos de Friedman e Iman-Davenport ($\alpha = 0.05$) para los algoritmos mostrados en la tabla 4.21.

En este caso, los valores estadísticos obtenidos fueron claramente más altos que los valores críticos, y por lo tanto podemos indicar que existen diferencias significativas en sus rangos al nivel estándar $\alpha = 0.05$, véase la tabla 4.23.

Posteriormente, hemos llevado a cabo el test de Holm para determinar si existen diferencias significativas con respecto a un algoritmo de control (en este caso, el algoritmo cooperativo en el que participan 3 agentes y emplea la topología BROADCAST) Bc3 (2Ts.B,MA.Ts.B.A2.Gd)RD, es el algoritmo con el mejor rango medio, según la figura 4.12. Los resultados se muestran en la tabla 4.24, donde vemos que hay diferencias estadísticas significativas entre los primeros cuatro esquemas de colaboración, pero no hay muestras de diferencias estadísticamente significativas con respecto a los demás esquemas.

Tabla 4.21: Columna central, número (#) y porcentaje (%) de instancias para el problema del BIBD, tomadas de la tabla 4.12, resueltas por los modelos cooperativos (identificados en la primera columna). En la columna derecha se muestran el grupo de algoritmos que operan en el modelo. El número de ejecuciones de cada algoritmo corresponde a $n * 10$, donde $n =$ Número de agentes.

Algorithms	# (%)	Collection
Bc2(Ts.B,MA.Ts.B.A2.Gd)DD	9 (31.03 %)	$\mathcal{A}_2$
Bc2(Ts.B,MA.Ts.B.A2.Gd)DR	9 (31.03 %)	$\mathcal{A}_2$
Bc2(Ts.B,MA.Ts.B.A2.Gd)DW	12 (41.38 %)	$\mathcal{A}_2$
Bc2(Ts.B,MA.Ts.B.A2.Gd)RD	11 (37.93 %)	$\mathcal{A}_2$
Bc2(Ts.B,MA.Ts.B.A2.Gd)RR	12 (41.38 %)	$\mathcal{A}_2$
Bc2(Ts.B,MA.Ts.B.A2.Gd)RW	9 (31.03 %)	$\mathcal{A}_2$
Ra2(Ts.B,MA.Ts.B.A2.Gd)DD	12 (41.38 %)	$\mathcal{A}_2$
Ra2(Ts.B,MA.Ts.B.A2.Gd)DR	10 (34.48 %)	$\mathcal{A}_2$
Ra2(Ts.B,MA.Ts.B.A2.Gd)DW	11 (37.93 %)	$\mathcal{A}_2$
Ra2(Ts.B,MA.Ts.B.A2.Gd)RD	10 (34.48 %)	$\mathcal{A}_2$
Ra2(Ts.B,MA.Ts.B.A2.Gd)RR	12 (41.38 %)	$\mathcal{A}_2$
Ra2(Ts.B,MA.Ts.B.A2.Gd)RW	12 (41.38 %)	$\mathcal{A}_2$
Ri2(Ts.B,MA.Ts.B.A2.Gd)DD	9 (31.03 %)	$\mathcal{A}_2$
Ri2(Ts.B,MA.Ts.B.A2.Gd)DR	**13 (44.83 %)**	$\mathcal{A}_2$
Ri2(Ts.B,MA.Ts.B.A2.Gd)RD	10 (34.48 %)	$\mathcal{A}_2$
Ri2(Ts.B,MA.Ts.B.A2.Gd)RR	10 (34.48 %)	$\mathcal{A}_2$
Ri2(Ts.B,MA.Ts.B.A2.Gd)RW	11 (37.93 %)	$\mathcal{A}_2$
Bc3(2Ts.B,MA.Ts.B.A2.Gd)DD	9 (31.03 %)	$\mathcal{A}_2$
Bc3(2Ts.B,MA.Ts.B.A2.Gd)DR	9 (31.03 %)	$\mathcal{A}_2$
Bc3(2Ts.B,MA.Ts.B.A2.Gd)DW	9 (31.03 %)	$\mathcal{A}_2$
Bc3(2Ts.B,MA.Ts.B.A2.Gd)RD	11 (37.93 %)	$\mathcal{A}_2$
Bc3(2Ts.B,MA.Ts.B.A2.Gd)RR	9 (31.03 %)	$\mathcal{A}_2$
Bc3(2Ts.B,MA.Ts.B.A2.Gd)RW	9 (31.03 %)	$\mathcal{A}_2$
Ra3(2Ts.B,MA.Ts.B.A2.Gd)DD	9 (31.03 %)	$\mathcal{A}_2$
Ra3(2Ts.B,MA.Ts.B.A2.Gd)RD	**13 (44.83 %)**	$\mathcal{A}_2$
Ra3(2Ts.B,MA.Ts.B.A2.Gd)RR	9 (31.03 %)	$\mathcal{A}_2$
Ri3(2Ts.B,MA.Ts.B.A2.Gd)DD	9 (31.03 %)	$\mathcal{A}_2$
Ri3(2Ts.B,MA.Ts.B.A2.Gd)DR	10 (34.48 %)	$\mathcal{A}_2$
Ri3(2Ts.B,MA.Ts.B.A2.Gd)RD	10 (34.48 %)	$\mathcal{A}_2$
Ri3(2Ts.B,MA.Ts.B.A2.Gd)RR	9 (31.03 %)	$\mathcal{A}_2$
Bc4(2Ts.B,2MA.Ts.B.A2.Gd)DR	10 (34.48 %)	$\mathcal{A}_2$
Bc4(2Ts.B,2MA.Ts.B.A2.Gd)RR	9 (31.03 %)	$\mathcal{A}_2$
Ra4(2Ts.B,2MA.Ts.B.A2.Gd)DW	9 (31.03 %)	$\mathcal{A}_2$
Bc5(3Ts.B,2MA.Ts.B.A2.Gd)DD	9 (31.03 %)	$\mathcal{A}_2$
Bc5(3Ts.B,2MA.Ts.B.A2.Gd)DR	10 (34.48 %)	$\mathcal{A}_2$
Bc5(3Ts.B,2MA.Ts.B.A2.Gd)RD	9 (31.03 %)	$\mathcal{A}_2$
Bc5(3Ts.B,2MA.Ts.B.A2.Gd)RW	9 (31.03 %)	$\mathcal{A}_2$
Ra5(3Ts.B,2MA.Ts.B.A2.Gd)DW	9 (31.03 %)	$\mathcal{A}_2$
Bc2(Ts.B,MA.Ts.D.A4.Gd)RD	9 (31.03 %)	$\mathcal{A}_3$
Ra5(3Ts.B,2MA.Ts.D.A4.Gd)RD	9 (31.03 %)	$\mathcal{A}_3$
Ri5(3Ts.B,2MA.Ts.D.A4.Gd)RD	9 (31.03 %)	$\mathcal{A}_3$

4.4.5 Discusión: Integrativos vs Cooperativos para el Problema del BIBD

Esta sección la dedicaremos a discutir (y comparar) el desempeño de los diferentes enfoques colaborativos, así como las técnicas integrativas que hemos empleado para abordar el problema de Diseño de Bloques Incompletos Balanceados (BIBD). Es importante señalar algunos aspectos que consideramos relevantes:

Tabla 4.22: Frecuencia relativa de cada diseño cooperativo particular sobre las 29 instancias del BIBD, para ello se han seleccionado los algoritmos cooperativos de la tabla 4.21. La columna de la izquierda indica la combinación MR de las políticas de migración(M) / recepción(R), donde M, R $\in$ {RANDOM (R), DIVERSE (D), WORST (W) como se explica en la sección 4.4.1. La columna central muestra la topología de comunicación, donde Bc = BROADCAST, Ra = RANDOM, and Ri = RING. La columna de la derecha nos indica el número de agentes que intervienen.

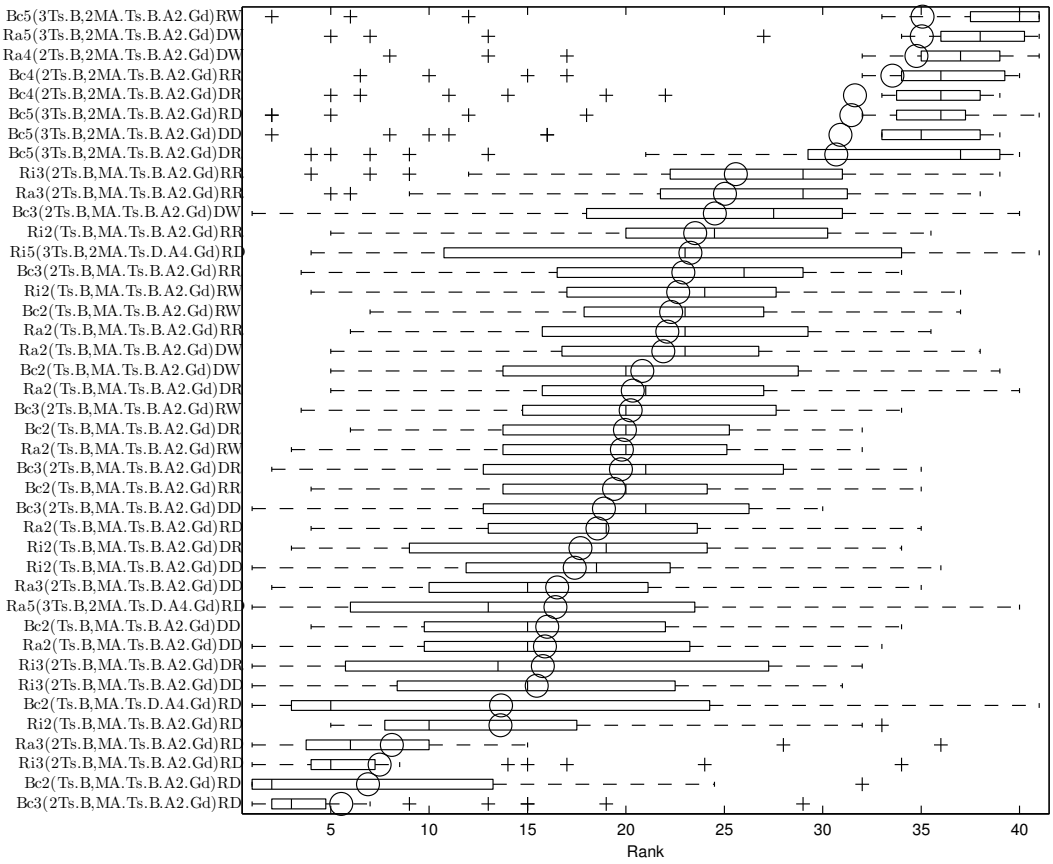
M/R Policy			Topology			Number of agents		
DD	7	(17.07 %)	Bc	19	(46.34 %)	$n = 2$	18	(43.90 %)
DR	7	(17.07 %)	Ra	12	(29.27 %)	$n = 3$	13	(31.71 %)
DW	5	(12.19 %)	Ri	10	(24.39 %)	$n = 4$	3	(7.32 %)
RD	10	(24.39 %)				$n = 5$	7	(17.07 %)
RR	7	(17.07 %)						
RW	5	(12.19 %)						

Tabla 4.23: Problema del BIBD: Resultados de los tests de Friedman e Iman-Davenport ($\alpha = 0.05$) de los diferentes algoritmos cooperativos, seleccionados de la tabla 4.21.

	Friedman value	Critical χ^2 value	Iman-Davenport value	Critical F_F value
All 41	467.822059	55.758479	18.924350	1.404986

- De toda esta experimentación hemos comparado una colección de 32 metaheurísticas integrativas que operan sobre 86 instancias clásicas del problema que hemos abordado. El resultado obtenido nos lleva a pensar que el mejor enfoque híbrido descrito anteriormente en la literatura, a saber, el cual corresponde a un algoritmo memético, lo hemos podido mejorar mediante la incorporación de una configuración basada en la ruptura de simetría. Esta nueva propuesta ha podido resolver 59 de las 86 instancias (es decir, 68.60% de efectividad). A pesar de esta mejora, se presentaron instancias que no fue posible resolver (un total de 27 de ellas). Estas instancias son parte de ese subconjunto de 29 instancias que hemos consideran como difíciles de resolver. Para abordar estos casos, que nos han resultado más difíciles, hemos propuesto una serie de esquemas colaborativos (algoritmos cooperativos). De estas nuevas propuestas hemos encontrado que pueden resolver un total de 13 de las instancias difíciles, y si incorporamos el accionar de las mejores propuestas colaborativas se logra resolver un total de 14 instancias diferentes del conjunto considerado duro. Los identificadores de estas instancias resueltas corresponden a: 21, 27, 28, 33, 34, 39, 44, 48, 50, 57 , 58, 63, 73 y 76 (véase la tabla 4.12). En resumen, esto significa que hemos logrado resolver un total de 71 de las 86 instancias (i.e. 82.56% de efectividad), con respecto al primer método mejor colocado que ha sido reportado en la literatura (i.e. 57 de las instancias de un total 86, que forman el conjunto inicialmente abordado), nos deja con un porcentaje de efectividad de 48.28%, respecto a las instancias consideradas duras.
- Lo que resulta aún más interesante, es el hecho de que el mejor esquema colaborativo combina diferentes técnicas integrativas explorando diferentes zonas del espacio de búsqueda. Esto nos lleva a pensar en la gran utilidad que representa la combinación de las técnicas de búsqueda sobre paisajes complementarios. Además, demos subrayar que un gran número de algoritmos

Figura 4.12: Problema del BIBD: Torneo de los diferentes algoritmos cooperativos, seleccionados de la tabla 4.21. Cada caja comprende desde el primer al tercer cuartil de la distribución, la mediana (2do cuartil) es indicada con una línea vertical, la zona sin valores atípicos (whiskers span) es de 1.5 veces el rango intercuartil, además los valores atípicos están marcados con un signo '+'.



cooperativos tuvieron éxito en muchas instancias problemáticas donde las metaheurísticas más simples no tuvieron éxito. Esto evidencia que la cooperación podría ser un factor determinante para mejorar el proceso de búsqueda. Sin embargo, mientras que la cooperación es un punto clave que ayuda a resolver nuevas instancias, también es cierto que los algoritmos no cooperativos muestran una mayor solidez debido a que logran un margen muy estrecho respecto a las soluciones óptimas en las instancias no resueltas.

- También debemos prestar atención al rendimiento logrado como consecuencia de la utilización de diferentes políticas para el intercambio de información entre los diferentes agentes que intervienen en la colaboración. Hemos probado 6 combinaciones para estas políticas, a saber, RANDOM-DIVERSE (RD), RANDOM-RANDOM (RR), RANDOM-WORST (RW), DIVERSE-DIVERSE (DD), DIVERSE-RANDOM (DR), and DIVERSE-WORST (DW). También observamos que el uso de la política WORST para la migración de candidatos deterioró el rendimiento del algoritmo. En general, la combinación RD claramente tiene un efecto positivo en la cooperación, ya que los primeros seis algoritmos de las diez mejores técnicas cooperativas se basan en ella.
- Consideremos también que hemos ejecutado una gran cantidad de experimentos y hemos con-

Tabla 4.24: Problema del BIBD: Resultados del test de Holm, de los diferentes algoritmos cooperativos, seleccionados de la tabla 4.21, usando **Bc3(2Ts.B,MA.Ts.B.A2.Gd)RD** como algoritmo de control, en el nivel estándar de $\alpha = 0.05$.

i	algorithm	z-statistic	p-value	α/i
1	Bc2(Ts.B,MA.Ts.B.A2.Gd)RD	4.330e-001	3.325e-001	5.000e-002
2	Ri3(2Ts.B,MA.Ts.B.A2.Gd)RD	6.193e-001	2.679e-001	2.500e-002
3	Ra3(2Ts.B,MA.Ts.B.A2.Gd)RD	8.166e-001	2.071e-001	1.667e-002
4	Ri2(Ts.B,MA.Ts.B.A2.Gd)RD	2.570e+000	5.079e-003	1.250e-002
5	Bc2(Ts.B,MA.Ts.D.A4.Gd)RD	2.581e+000	4.920e-003	1.000e-002
6	Ri3(2Ts.B,MA.Ts.B.A2.Gd)DD	3.157e+000	7.974e-004	8.333e-003
7	Ri3(2Ts.B,MA.Ts.B.A2.Gd)DR	3.255e+000	5.660e-004	7.143e-003
8	Ra2(Ts.B,MA.Ts.B.A2.Gd)DD	3.288e+000	5.038e-004	6.250e-003
9	Bc2(Ts.B,MA.Ts.B.A2.Gd)DD	3.327e+000	4.393e-004	5.556e-003
10	Ra5(3Ts.B,2MA.Ts.D.A4.Gd)RD	3.458e+000	2.718e-004	5.000e-003
11	Ra3(2Ts.B,MA.Ts.B.A2.Gd)DD	3.486e+000	2.454e-004	4.545e-003
12	Ri2(Ts.B,MA.Ts.B.A2.Gd)DD	3.771e+000	8.141e-005	4.167e-003
13	Ri2(Ts.B,MA.Ts.B.A2.Gd)DR	3.864e+000	5.581e-005	3.846e-003
14	Ra2(Ts.B,MA.Ts.B.A2.Gd)RD	4.138e+000	1.753e-005	3.571e-003
15	Bc3(2Ts.B,MA.Ts.B.A2.Gd)DD	4.242e+000	1.108e-005	3.333e-003
16	Bc2(Ts.B,MA.Ts.B.A2.Gd)RR	4.406e+000	5.255e-006	3.125e-003
17	Bc3(2Ts.B,MA.Ts.B.A2.Gd)DR	4.516e+000	3.150e-006	2.941e-003
18	Ra2(Ts.B,MA.Ts.B.A2.Gd)RW	4.532e+000	2.915e-006	2.778e-003
19	Bc2(Ts.B,MA.Ts.B.A2.Gd)DR	4.582e+000	2.305e-006	2.632e-003
20	Bc3(2Ts.B,MA.Ts.B.A2.Gd)RW	4.675e+000	1.470e-006	2.500e-003
21	Ra2(Ts.B,MA.Ts.B.A2.Gd)DR	4.708e+000	1.252e-006	2.381e-003
22	Bc2(Ts.B,MA.Ts.B.A2.Gd)DW	4.861e+000	5.830e-007	2.273e-003
23	Ra2(Ts.B,MA.Ts.B.A2.Gd)DW	5.201e+000	9.905e-008	2.174e-003
24	Ra2(Ts.B,MA.Ts.B.A2.Gd)RR	5.267e+000	6.938e-008	2.083e-003
25	Bc2(Ts.B,MA.Ts.B.A2.Gd)RW	5.327e+000	4.988e-008	2.000e-003
26	Ri2(Ts.B,MA.Ts.B.A2.Gd)RW	5.442e+000	2.630e-008	1.923e-003
27	Bc3(2Ts.B,MA.Ts.B.A2.Gd)RR	5.524e+000	1.652e-008	1.852e-003
28	Ri5(3Ts.B,2MA.Ts.D.A4.Gd)RD	5.640e+000	8.524e-009	1.786e-003
29	Ri2(Ts.B,MA.Ts.B.A2.Gd)RR	5.716e+000	5.444e-009	1.724e-003
30	Bc3(2Ts.B,MA.Ts.B.A2.Gd)DW	6.034e+000	7.989e-010	1.667e-003
31	Ra3(2Ts.B,MA.Ts.B.A2.Gd)RR	6.199e+000	2.849e-010	1.613e-003
32	Ri3(2Ts.B,MA.Ts.B.A2.Gd)RR	6.374e+000	9.210e-011	1.563e-003
33	Bc5(3Ts.B,2MA.Ts.B.A2.Gd)DR	7.996e+000	6.414e-016	1.515e-003
34	Bc5(3Ts.B,2MA.Ts.B.A2.Gd)DD	8.062e+000	3.753e-016	1.471e-003
35	Bc5(3Ts.B,2MA.Ts.B.A2.Gd)RD	8.237e+000	8.801e-017	1.429e-003
36	Bc4(2Ts.B,2MA.Ts.B.A2.Gd)DR	8.298e+000	5.309e-017	1.389e-003
37	Bc4(2Ts.B,2MA.Ts.B.A2.Gd)RR	8.901e+000	2.779e-019	1.351e-003
38	Ra4(2Ts.B,2MA.Ts.B.A2.Gd)DW	9.290e+000	7.739e-021	1.316e-003
39	Ra5(3Ts.B,2MA.Ts.B.A2.Gd)DW	9.377e+000	3.383e-021	1.282e-003
40	Bc5(3Ts.B,2MA.Ts.B.A2.Gd)RW	9.388e+000	3.049e-021	1.250e-003

siderado otras combinaciones y algoritmos, muchos de los cuales no se han tabulado en esta investigación para evitar colocar data poco relevante, dado que su rendimiento no es prometedor.

- Otro resultado muy interesante que puede extraerse de nuestros experimentos es que la cooperación funciona mejor cuando todos los algoritmos conectados funcionan en el mismo dominio de cálculo y bajo la misma formulación del problema.

Tabla 4.25: Problema del BIBD: Comparación del tiempo de ejecución entre la técnica híbrida MA=$Ma.Ts.B*.A4.Gd$ (i.e. el mejor MA) y el modelo de cooperación $Ra3(2Ts.B, Ma.Ts.B.A2.Gd-)RD$ (i.e. uno de los mejores métodos colaborativos, que ha mostrado un rendimiento similar, estadísticamente hablando y de acuerdo a la tabla 4.24, para el mejor algoritmo colaborativo (i.e. $Bc3(2Ts.B, MA.Ts.B.A2.Gd)RD$)). Las dos instancias del problema: $I_1 = \langle 14,26,13,7,6\rangle$ y $I_2 = \langle 25,25,9,9,3\rangle$ (del conjunto de las 29 instancias consideradas difíciles en esta investigación) que fueron resueltas por ambos métodos se seleccionan como puntos de referencia para la comparación. T_{max} y T_{min} muestra el tiempo máximo y mínimo (en segundos) consumido por la mejor ejecución de cada técnica para encontrar una solución. La última columna muestra la mejora del método cooperativo con respecto al MA.

	Instancias					
	I_1		I_2		Coop/MA(%)	
Tiempo(s)	MA	Coop	MA	Coop	I_1	I_2
T_{min}	23.00	0.02	20.08	0.03	1150	669
T_{max}	5754.78	24.67	9317.03	34.91	233	266

Si nos referimos a los tiempos de ejecución, hemos comparado el mejor enfoque integrador (digamos, $Ma.Ts.B*.A4.Gd$) con uno de nuestros mejores métodos de cooperación (digamos, $Ra3(2Ts.B, Ma.Ts.B.A2).Gd)RD$). Inicialmente, consideramos todas las 29 instancias del problema difíciles que se muestran en la tabla 4.12 pero nos ha parecido bastante injusta esta comparativa, ya que el algoritmo memético sólo pudo resolver 2 de ellas (contra 13 que logra resolver el esquema colaborativo $Ra3(2Ts.B, Ma.Ts.B.A2.Gd)RD$ como se muestra en la tabla 4.21). Esto básicamente significó que el MA consumió el número máximo de evaluaciones en cada una de sus corridas tratando de resolver las otras 27 instancias.

Sin embargo, para la técnica cooperativa, en una ejecución completa sin registrar éxito alguno, solo ocurrió en 16 instancias del problema. Lo que ha permitido mostrar una gran diferencia a favor del método cooperativo. Para ser justos, solo consideramos las dos instancias resueltas por ambos métodos y los resultados se muestran en la tabla 4.25 . El algoritmo memético es notablemente más lento que el cooperativo. Una explicación para este fenómeno la podemos deducir debido a que los esquemas colaborativos donde intervienen distintos algoritmos produce una sinergia positiva para que la cooperación entre las técnicas rinda mejor que cada una de sus partes constituyentes.

4.5 Métodos Híbridos para Resolver el TDP

Al igual que se ha hecho con el BIBD, para abordar el problema del Diseño de Plantillas (TDP), hemos empleados los dos esquemas híbridos descritos anteriormente: (a) Algoritmos meméticos y (b) Algoritmos cooperativos. De igual forma, el método de optimización híbrido que estamos proponiendo consiste en ejecutar una búsqueda local subordinada a la ejecución de un algoritmo genético externo (GA). En otras palabras, proponemos emplear un algoritmo memético (MA). La entrada de este algoritmo está constituida por los parámetros del problema (i.e., T, V, s), así como los propios parámetros de la técnica Genética que se emplea, es decir, los operadores genéticos y sus parámetros asociados (tales como, por ejemplo, probabilidad de mutación y/o aplicación de mejora local); el resultado del algoritmo es el mejor individuo encontrado durante el proceso de búsqueda. La selección

se realiza mediante torneo binario con reemplazo del peor individuo de la población. La búsqueda local está restringida para explorar n_v vecinos, lo que en otras palabras significa evaluar un total n_v candidatos. Para el caso del esquema colaborativo se emplea un grupo de agentes que operan en periodos de ejecución e intercambio de información entre ellos, esto permitirá que cada uno de ellos explore un espacio de búsqueda específico a través de procesos de intensificación, en procura de un mecanismo efectivo que permita a dichas técnicas escapar de los *mínimos locales*. Los agentes que intervienen son el grupo de técnicas integrativas que se han aplicado en forma individual. La idea es aprovechar la sinergia entre estas técnicas para la exploración de zonas del espacio de búsqueda que resulten prometedoras para alcanzar la mayor cantidad de óptimos globales posibles. Este enfoque ha demostrado ser particularmente eficiente en una serie de problemas combinatorios que han sido abordados en el ámbito científico [10, 11, 72, 196]. Ahora, a diferencia de lo que se hace a menudo en este tipo de esquema de colaboración nuestro interés es estudiar el efecto de considerar diferentes zonas del espacio de búsqueda para ser evaluados por separado según la topología de la red utilizada. Más precisamente, consideramos una serie de algoritmos que cooperan intercambiando información en los que los agentes son provistos por una de las técnicas metaheurísticas previamente propuestas (Hc, TS, GAs o MAs) o su equivalente, adaptado a la representación correspondiente (como se muestra en las secciones 3.1.8, y 3.2.9) y donde el algoritmo que se ejecuta en cualquier agente dado, también puede estar provisto de algún mecanismo para romper las simetrías (como se explica en las secciones 3.1.8.2 y 3.2.9.2). Esto significa que algunos agentes posiblemente trabajen en diferentes espacios de codificación/búsqueda y utilicen formulaciones distintas del problema. Los algoritmos dependen de su topología de interacción, de los modelos utilizado para codificar los candidatos y de las políticas de migración/recepción.

4.5.1 Convenciones de Notación de las Técnicas Híbridas para el TDP

La notación empleada para identificar las diferentes técnicas y combinaciones de ellas se ha codificado de manera similar a lo descrito en las sección 3.4.3.1, con la salvedad de que al modelo Primal lo identificamos con la letra **P** a diferencia del dual que se identifica con la letra **D**. Asi, por ejemplo: $MA.Hc.P.A2.Gd$ corresponde a un algoritmo Memético que implementa mejora local por medio de la técnica de búsqueda local *hill climbing*, hace uso de 2 padres para la recombinación por medio de un operador de cruce de tipo voraz, además de emplear la representación alternativa primal y una configuración sin ruptura de simetrías, de igual forma, $Ri2(Ts.P*, MA.Ts.D.A2.Gd)RW$ corresponde a la utilización de 2-agentes empleando la topología RING para el esquema colaborativo que conecta los algoritmos (a) basados en búsqueda local Tabú (Ts), trabajando en una representación alternativa Primal (candidatos expresados con base a diseños) bajo el esquema de ruptura de simetrías y (b) un algoritmo MA, el cual utiliza un esquema de presentación Dual (candidatos expresados con base a Slots), utilizando 2 padres para la recombinación por medio de un operador de cruce voraz (Gd), y que integra un operador de mejora local Ts; para este caso particular, el esquema colaborativo permite enviar un candidato, del conjunto de soluciones del nodo inicial, seleccionado en forma aleatoria (RANDOM política de migración), con reemplazo del peor individuo del conjunto de soluciones del nodo destino (WORST política de aceptación).

4.5.1.1 Resultados Experimentales de las Técnicas Meméticas del TDP

Esta sección evalúa el rendimiento de varios algoritmos híbridos diferentes creados a partir del esquema descrito en la Sección 3.4. La idea es aprovechar la sinergia entre las metaheurísticas cuando

estas funcionan a modo integrativo. Para este propósito, nuestra integración usa el paradigma de optimización metaheurística basado en la explotación sistemática del conocimiento sobre el problema que se está resolviendo, y la combinación sinérgica de ideas tomadas de otras metaheurísticas basadas en la población (es decir, GA) y trayectoria (es decir, HC y TS). Como ya se indicó anteriormente, consideramos, además, las dos representaciones alternativas y los esquemas basados en ruptura, sin-ruptura de simetrías.

Para este propósito, se han considerado 32 variantes de algoritmos para la realización de las pruebas correspondientes. Este número de propuestas es el resultado de considerar dos representaciones alternativas para el problema distintas (es decir, primal / dual), dos operadores de recombinación (es decir, UX / Gd), dos técnicas de mejora local (es decir, HC y TS), la posibilidad de aplicar procedimientos de ruptura, no-ruptura de simetría y la elección de 2 o 4 padres para la recombinación. Todas las combinaciones posibles producen 32 escenarios distintos que están asociados a los algoritmos meméticos. Los resultados de rendimiento obtenidos por todas estas propuestas híbridas se muestran en la tabla 4.26. El concepto de **dominio** (como se usa en este documento) se basa en el *Dominio de Pareto* presentado en optimización multiobjetivo [339]. Más precisamente en el contexto de este documento, maximizar el número de soluciones encontradas para cada instancia de problema podría considerarse como un objetivo dentro de un enfoque multiobjetivo. La quinta columna de la tabla 4.26 indica el valor de clasificación promedio obtenido de las distribuciones de los rangos de cada algoritmo. Allí, también se muestra para cada uno de los 32 algoritmos híbridos (identificados en la primera columna) la cantidad de veces que se resolvió una instancia de problema en sus 20 ejecuciones (en las columnas dos a cuatro, entre paréntesis también se muestra el porcentaje de éxito correspondiente).

En general, los métodos meméticos funcionan razonablemente bien en todas las instancias del problema. Tenga en cuenta que más de 59% de estos algoritmos encuentran soluciones factibles en todos ellos. Considerando la suma del número de soluciones factibles encontradas, se puede observar que las versiones de los algoritmos mejor ubicados son Ma.Hc.P*.A2.Gd, Ma.Hc.P*.A2.Ux, Ma.Hc.P*.A4.Gd y Ma.Ts.P.A2.Ux.

Aplicando un torneo entre los algoritmos utilizados – para los fines de clasificación se utiliza la suma del número de soluciones factibles encontradas en cada instancia de problema (del conjunto de 20 ejecuciones)–, las distribuciones de estos rangos se muestran en la figura 4.13.

Además, la mayoría de las técnicas no parecen mostrar diferencias significativas (cerca de 90%), como lo confirman los tests de Friedman e Iman-Davenport, así como el de Holm. Los resultados de estas pruebas se muestran en las tablas 4.27 y 4.28, respectivamente. Esto es una prueba de la solidez de las técnicas Meméticas.

Tabla 4.26: Algoritmos integrativos (meméticos) para resolver el TDP: Número (y porcentaje) de soluciones factibles alcanzadas considerando el grupo de escenarios tomados de [251]. Las filas son ordenadas de acuerdo con el valor del rango asignado a cada algoritmo (mostrado en la penúltima columna). En la última columna el simbolo ✗indica los algoritmos que no son dominados por otra técnica. En general, una solución no está dominada si no hay otra solución que mejore alguno de sus valores objetivos y no sea peor en los valores objetivos restantes

Metaheurísticas	Cat Food	Herbs C.	Magazine I.	Ranking promedio	N.D
Ma.Hc.P*.A2.Ux	17 (85.00 %)	**19 (95.00 %)**	**10 (50.00 %)**	4.83333	✗
Ma.Hc.P*.A4.Gd	17 (85.00 %)	18 (90.00 %)	8 (40.00 %)	6.00000	
Ma.Ts.P*.A2.Gd	17 (85.00 %)	**19 (95.00 %)**	6 (30.00 %)	6.66667	
Ma.Hc.P*.A2.Gd	17 (85.00 %)	17 (85.00 %)	9 (45.00 %)	7.16667	
Ma.Ts.P.A2.Ux	17 (85.00 %)	7 (35.00 %)	9 (45.00 %)	8.00000	
Ma.Hc.P.A4.Gd	17 (85.00 %)	12 (60.00 %)	5 (25.00 %)	9.66667	
Ma.Ts.P*.A4.Gd	16 (80.00 %)	**19 (95.00 %)**	3 (15.00 %)	11.00000	
Ma.Hc.P.A4.Ux	17 (85.00 %)	3 (15.00 %)	4 (20.00 %)	11.33333	
Ma.Ts.P*.A2.Ux	15 (75.00 %)	**19 (95.00 %)**	4 (20.00 %)	11.66667	
Ma.Hc.D*.A2.Gd	14 (70.00 %)	14 (70.00 %)	7 (35.00 %)	12.00000	
Ma.Hc.P.A2.Ux	17 (85.00 %)	4 (20.00 %)	3 (15.00 %)	12.66667	
Ma.Hc.D*.A4.Gd	15 (75.00 %)	13 (65.00 %)	6 (30.00 %)	12.66667	
Ma.Ts.P*.A4.Ux	16 (80.00 %)	17 (85.00 %)	3 (15.00 %)	13.00000	
Ma.Hc.P*.A4.Ux	17 (85.00 %)	18 (90.00 %)	2 (10.00 %)	13.33333	
Ma.Hc.D.A4.Gd	**18 (90.00 %)**	0 (0.00 %)	7 (35.00 %)	13.66667	✗
Ma.Ts.P.A4.Ux	17 (85.00 %)	3 (15.00 %)	2 (10.00 %)	14.66667	
Ma.Hc.D.A2.Gd	14 (70.00 %)	8 (40.00 %)	4 (20.00 %)	16.00000	
Ma.Ts.P.A4.Gd	16 (80.00 %)	15 (75.00 %)	1 (5.00 %)	16.33333	
Ma.Ts.P.A2.Gd	**18 (90.00 %)**	1 (5.00 %)	0 (0.00 %)	17.16667	✗
Ma.Hc.D.A2.Ux	13 (65.00 %)	3 (15.00 %)	4 (20.00 %)	18.66667	
Ma.Hc.D*.A2.Ux	10 (50.00 %)	0 (0.00 %)	5 (25.00 %)	19.66667	
Ma.Ts.D*.A2.Gd	16 (80.00 %)	1 (5.00 %)	2 (10.00 %)	20.33333	
Ma.Hc.P.A2.Gd	**18 (90.00 %)**	0 (0.00 %)	0 (0.00 %)	20.83333	
Ma.Hc.D*.A4.Ux	13 (65.00 %)	0 (0.00 %)	3 (15.00 %)	21.33333	
Ma.Ts.D*.A4.Gd	16 (80.00 %)	0 (0.00 %)	0 (0.00 %)	23.66667	
Ma.Hc.D.A4.Ux	12 (60.00 %)	0 (0.00 %)	2 (10.00 %)	24.00000	
Ma.Ts.D.A4.Ux	8 (40.00 %)	0 (0.00 %)	4 (20.00 %)	24.33333	
Ma.Ts.D.A2.Gd	12 (60.00 %)	0 (0.00 %)	2 (10.00 %)	25.66667	
Ma.Ts.D*.A4.Ux	8 (40.00 %)	0 (0.00 %)	3 (15.00 %)	26.33333	
Ma.Ts.D*.A2.Ux	12 (60.00 %)	0 (0.00 %)	1 (5.00 %)	27.33333	
Ma.Ts.D.A2.Ux	10 (50.00 %)	0 (0.00 %)	1 (5.00 %)	28.00000	
Ma.Ts.D.A4.Gd	5 (25.00 %)	0 (0.00 %)	1 (5.00 %)	30.00000	

Tabla 4.27: Resultados para los tests de Friedman e Iman-Davenport, de los diferentes algoritmos integrativos (meméticos) propuestos trabajando sobre el problema del TDP, para los escenarios tomados de [251].

	Friedman value	Critical χ^2 value	Iman-Davenport value	Critical F_F value
All 32	53.405303	44.985343	2.697599	1.636151

Figura 4.13: Torneo de los diferentes algoritmos integrativos (meméticos) propuestos trabajando sobre el problema del TDP, para los escenarios tomados de [251]. Cada caja comprende desde el primer al tercer cuartil de la distribución, la mediana (2do cuartil) es indicada con una línea vertical, la zona sin valores atípicos (whiskers span) es de 1.5 veces el rango intercuartil, además los valores atípicos están marcados con un signo '+'.

.

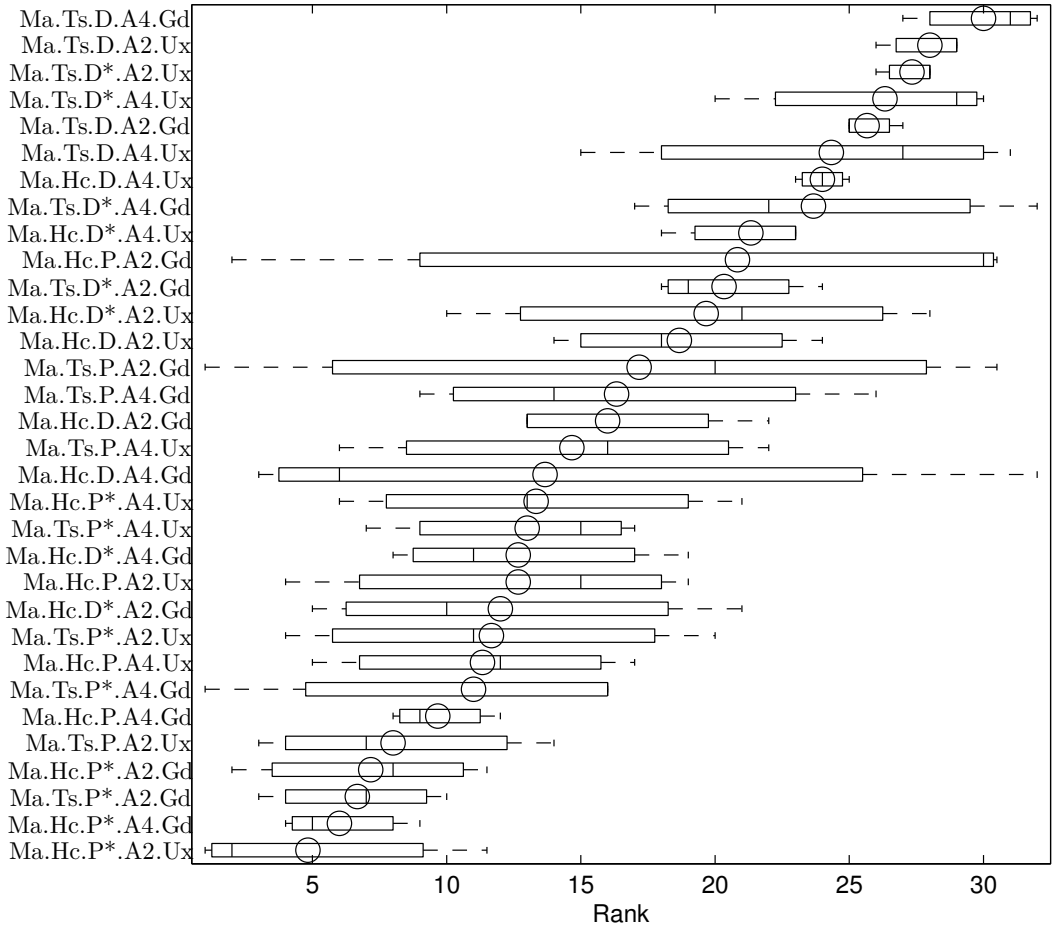


Tabla 4.28: Resultados del test de Holm, de los diferentes algoritmos integrativos (meméticos), propuestos trabajando sobre el problema del TDP, para los escenarios tomados de [251], considerando Ma.Hc.P*.A2.Ux como algoritmo de control.

i	algorithm	z-statistic	p-value	α/i
1	Ma.Hc.P*.A4.Gd	1.523e-001	4.395e-001	5.000e-002
2	Ma.Ts.P*.A2.Gd	2.394e-001	4.054e-001	2.500e-002
3	Ma.Hc.P*.A2.Gd	3.046e-001	3.803e-001	1.667e-002
4	Ma.Ts.P.A2.Ux	4.134e-001	3.396e-001	1.250e-002
5	Ma.Hc.P.A4.Gd	6.310e-001	2.640e-001	1.000e-002
6	Ma.Ts.P*.A4.Gd	8.051e-001	2.104e-001	8.333e-003
7	Ma.Hc.P.A4.Ux	8.486e-001	1.980e-001	7.143e-003
8	Ma.Ts.P*.A2.Ux	8.921e-001	1.862e-001	6.250e-003
9	Ma.Hc.D*.A2.Gd	9.357e-001	1.747e-001	5.556e-003
10	Ma.Hc.D*.A4.Gd	1.023e+000	1.532e-001	5.000e-003
11	Ma.Hc.P.A2.Ux	1.023e+000	1.532e-001	4.545e-003
12	Ma.Ts.P*.A4.Ux	1.066e+000	1.432e-001	4.167e-003
13	Ma.Hc.P*.A4.Ux	1.110e+000	1.336e-001	3.846e-003
14	Ma.Hc.D.A4.Gd	1.153e+000	1.244e-001	3.571e-003
15	Ma.Ts.P.A4.Ux	1.284e+000	9.960e-002	3.333e-003
16	Ma.Hc.D.A2.Gd	1.458e+000	7.243e-002	3.125e-003
17	Ma.Ts.P.A4.Gd	1.501e+000	6.662e-002	2.941e-003
18	Ma.Ts.P.A2.Gd	1.610e+000	5.368e-002	2.778e-003
19	Ma.Hc.D.A2.Ux	1.806e+000	3.545e-002	2.632e-003
20	Ma.Hc.D*.A2.Ux	1.937e+000	2.640e-002	2.500e-003
21	Ma.Ts.D*.A2.Gd	2.024e+000	2.150e-002	2.381e-003
22	Ma.Hc.P.A2.Gd	2.089e+000	1.836e-002	2.273e-003
23	Ma.Hc.D*.A4.Ux	2.154e+000	1.561e-002	2.174e-003
24	Ma.Ts.D*.A4.Gd	2.459e+000	6.969e-003	2.083e-003
25	Ma.Hc.D.A4.Ux	2.502e+000	6.168e-003	2.000e-003
26	Ma.Ts.D.A4.Ux	2.546e+000	5.450e-003	1.923e-003
27	Ma.Ts.D.A2.Gd	2.720e+000	3.264e-003	1.852e-003
28	Ma.Ts.D*.A4.Ux	2.807e+000	2.500e-003	1.786e-003
29	Ma.Ts.D*.A2.Ux	2.938e+000	1.654e-003	1.724e-003
30	Ma.Ts.D.A2.Ux	3.025e+000	1.245e-003	1.667e-003
31	Ma.Ts.D.A4.Gd	3.286e+000	5.086e-004	1.613e-003

4.5.1.2 Test Estadísticos de las Técnicas Meméticas del TDP

Algunas observaciones interesantes surgen cuando los datos se factorizan a lo largo de dimensiones particulares. Por esta razón, hemos llevado a cabo un análisis conjunto considerando tanto la representación como el posible uso de la ruptura de simetría. Esto produce cuatro grupos, {primal, dual}×{sin ruptura de simetrías, con ruptura de simetrías}. De nuevo, hemos optado por un torneo basado en rangos. Entonces, hemos calculado el rango r_j^i de cada grupo de algoritmos j en cada instancia i. En todos los casos, hemos utilizado a efectos de clasificación la suma del número de soluciones factibles encontradas en cada instancia de problema por cada algoritmo que pertenece al grupo. El mejor grupo recibe el rango 1 y el peor recibe el rango k, donde $k = 4$ que corresponde al número de grupos involucrados en el ranking. La figura 4.14 muestra las distribuciones de estos rangos. Observe que cuando se ha utilizado el modelo Primal con ruptura de simetría produce el mejor rango global. Además, como se muestra en tablas 4.29 - 4.30, existen diferencias significativas (estadísticamente hablando) con respecto a las otras propuestas.

Tabla 4.29: Resultados de los tests de Friedman e Iman-Davenport, para las diferentes técnicas integrativas (meméticos), aplicadas sobre el problema del TDP para los escenarios tomados de [251], reunidas en cuatro grupos: {primal, dual}×{sin ruptura de simetrías, con ruptura de simetrías}. P identifica el modelo de representación Primal sin ruptura de simetrías, P* modelo Primal con ruptura de simetrías; D el modelo Dual sin ruptura de simetrías, y D* el modelo Dual con ruptura de simetrías.

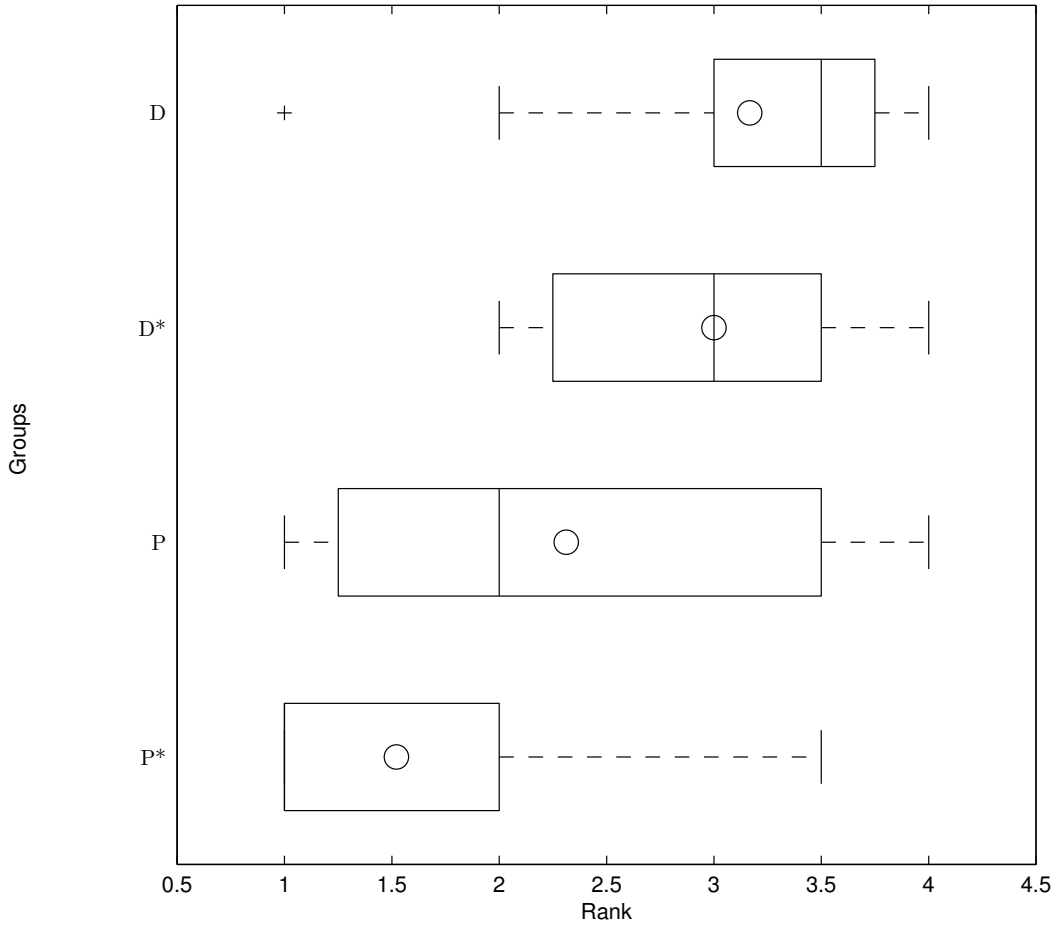
	Friedman value	Critical χ^2 value	Iman-Davenport value	Critical F_F value
All 4	27.007143	7.814728	11.185583	2.678301

Tabla 4.30: Resultados del test de Holm para las diferentes técnicas integrativas (meméticos), aplicadas sobre el problema del TDP para los escenarios tomados de [251], reunidas en cuatro grupos: {primal, dual}×{sin ruptura de simetrías, con ruptura de simetrías}. P identifica el modelo de representación Primal sin ruptura de simetrías, P* modelo Primal con ruptura de simetrías; D el modelo Dual sin ruptura de simetrías, y D* el modelo Dual con ruptura de simetrías. Usando P* como algoritmo de control.

i	algorithm	z-statistic	p-value	α/i
1	P	2.124e+000	1.682e-002	5.000e-002
2	D*	3.969e+000	3.608e-005	2.500e-002
3	D	4.416e+000	5.022e-006	1.667e-002

Otra dimensión que hemos considerado es la elección del operador de recombinación (es decir, UX / Gd). La combinación de todos los valores posibles en estas dimensiones produce un conjunto de 8 grupos diferentes dependiendo de si se emplea un enfoque basado en el modelo primal o dual, si se decide o no emplear la ruptura de simetría, y selecciona uno de los dos posibles operadores de cruce. De acuerdo con la figura 4.15, el operador de cruce Gd, en combinación de la representación alternativa primal y que aplica ruptura de simetría aparece en la primera posición del rango. Sin embargo, no detectamos diferencias significativas (nuevamente a nivel estadístico) con respecto a su contraparte, la combinación P*.Ux. Este resultado se puede extraer de los datos que se muestran en

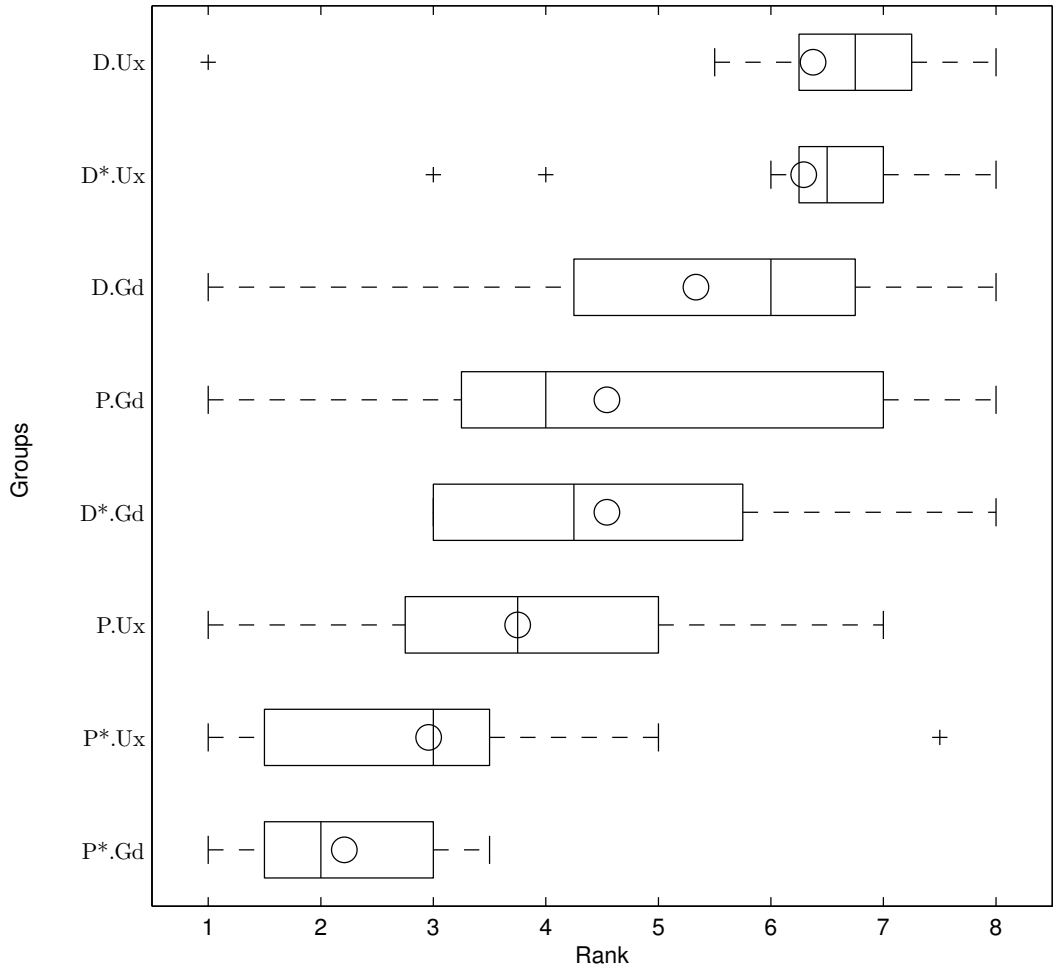
Figura 4.14: Torneo para las diferentes técnicas integrativas (meméticos), aplicadas sobre el problema del TDP para los escenarios tomadas de [251], reunidas en cuatro grupos: {primal, dual}×{sin ruptura de simetrías, con ruptura de simetrías}. P identifica el modelo de representación Primal sin ruptura de simetrías, P* modelo Primal con ruptura de simetrías; D el modelo Dual sin ruptura de simetrías, y D* el modelo Dual con ruptura de simetrías. Cada caja comprende desde el primer al tercer cuartil de la distribución, la mediana (2do cuartil) es indicada con una línea vertical, la zona sin valores atípicos (whiskers span) es de 1.5 veces el rango intercuartil, además los valores atípicos están marcados con un signo '+'.



tablas 4.31 - 4.32 que muestran los resultados obtenidos para la dimensión analizada.

Si nos concentramos en la obtención del mejor resultado encontrado para el problema, es decir, la solución que minimiza el número de plantillas con un desperdicio mínimo, uno de nuestros modelos híbridos parece ser el mejor: concretamente, la técnica que hace uso de la representación alternativa primal con ruptura de simetrías, que utiliza Ux como operador de cruce, una recombinación de 2 padres y algoritmo HC para la mejora local. Este algoritmo (es decir, Ma.Hc.P*.A2.Ux) obtiene las mejores configuraciones para los tres escenarios del problema. Las soluciones se muestran en la tabla 4.33.

Figura 4.15: Torneo para las diferentes técnicas integrativas (meméticos), aplicadas sobre el problema del TDP para los escenarios tomados de [251], reunidas en ocho grupos: {primal, dual}×{sin ruptura de simetrías, con ruptura de simetrías}×{operadores de cruce Ux/Gd}. P identifica el modelo de representación Primal sin ruptura de simetrías, P* modelo Primal con ruptura de simetrías; D el modelo Dual sin ruptura de simetrías, y D* el modelo Dual con ruptura de simetrías. Cada caja comprende desde el primer al tercer cuartil de la distribución, la mediana (2do cuartil) es indicada con una línea vertical, la zona sin valores atípicos (whiskers span) es de 1.5 veces el rango intercuartil, además los valores atípicos están marcados con un signo '+'.



4.5.2 Comparando las Técnicas Metaheurísticas y las Meméticas Aplicadas al TDP

Ahora nos vamos a concentrar en la realización de una comparación cruzada entre los mejores métodos considerados en esta experimentación para atacar el problema del TDP, es decir, enfoques básicos (i.e. metaheurísticas básicas LS y GA) e híbridas (i.e. algoritmos meméticos). Para este fin, combinamos los resultados de los mejores algoritmos de cada clase. En particular, hemos seleccionado los 21 métodos que producen mejores resultados y que no tienen diferencias de rendimiento significativas según se muestra y los mejores 29 de la sección 4.5.1.1 (consulte las tablas 3.24 y 4.28). Debemos mencionar que la selección de estos algoritmos no se ha realizado en forma arbitraria; para cada clase de algoritmo, aplicamos previamente el test de Holm tomando como algoritmo de control el mejor

Tabla 4.31: Resultados de los tests de Friedman e Iman-Davenport, para las diferentes técnicas integrativas (meméticos), aplicadas sobre el problema del TDP para los escenarios tomados de [251], reunidas en ocho grupos: {primal, dual}×{sin ruptura de simetrías, con ruptura de simetrías}×{operadores de cruce Ux/Gd}. P identifica el modelo de representación Primal sin ruptura de simetrías, P* modelo Primal con ruptura de simetrías; D el modelo Dual sin ruptura de simetrías, y D* el modelo Dual con ruptura de simetrías.

	Friedman value	Critical χ^2 value	Iman-Davenport value	Critical F_F value
All 8	31.229167	14.067140	6.509672	2.130990

Tabla 4.32: Resultados del test de Holm, para las diferentes técnicas integrativas (meméticos), aplicadas sobre el problema del TDP para los escenarios tomados de [251], reunidas en ocho grupos: {primal, dual}×{sin ruptura de simetrías, con ruptura de simetrías}×{operadores de cruce Ux/Gd}. P identifica el modelo de representación Primal sin ruptura de simetrías, P* modelo Primal con ruptura de simetrías; D el modelo Dual sin ruptura de simetrías, y D* el modelo Dual con ruptura de simetrías, usando P*.Gd como algoritmo de control.

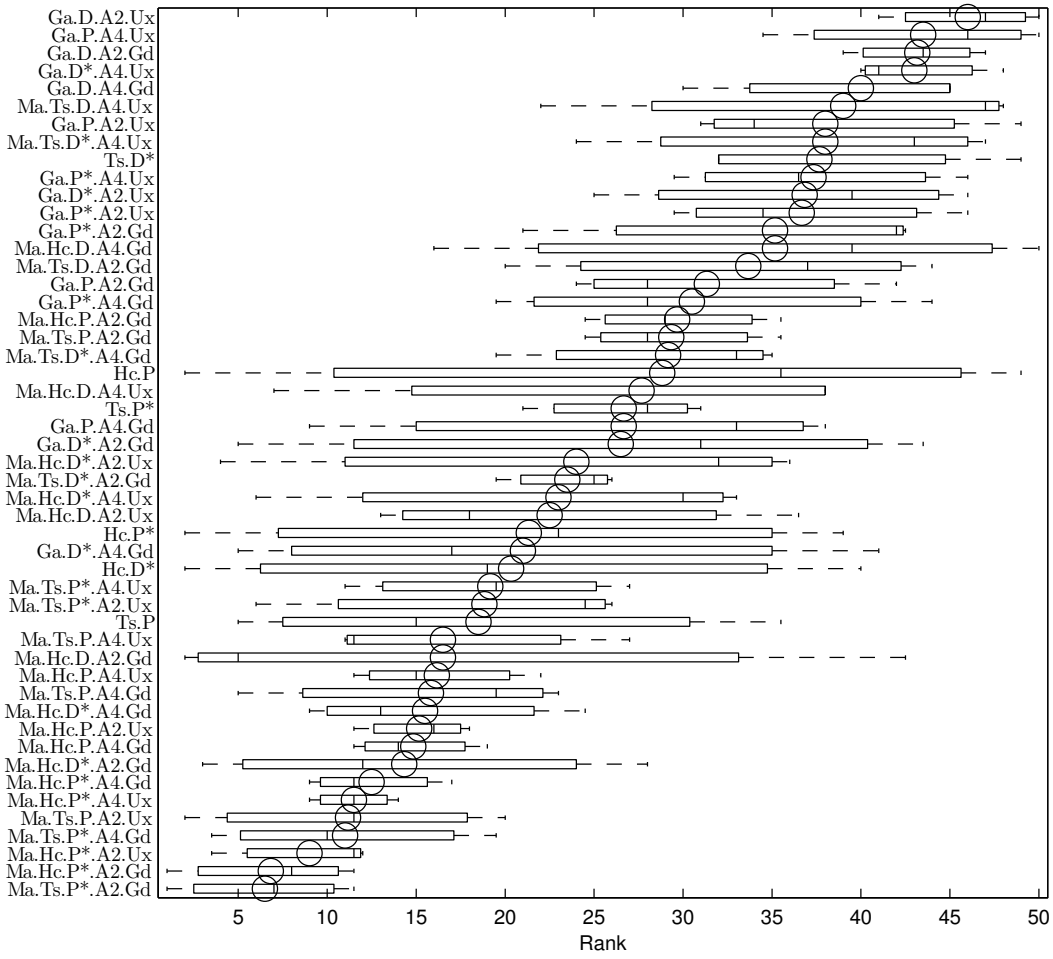
i	algorithm	z-statistic	p-value	α/i
1	P*.Ux	7.500e-001	2.266e-001	5.000e-002
2	P.Ux	1.542e+000	6.158e-002	2.500e-002
3	P.Gd	2.333e+000	9.815e-003	1.667e-002
4	D*.Gd	2.333e+000	9.815e-003	1.250e-002
5	D.Gd	3.125e+000	8.890e-004	1.000e-002
6	D*.Ux	4.083e+000	2.220e-005	8.333e-003
7	D.Ux	4.167e+000	1.545e-005	7.143e-003

Tabla 4.33: Mejores configuraciones encontradas para el problema del TDP, de los 3 escenarios evaluados tomados de [251], del mejor algoritmo mémetico $Ma.Hc.P*.A2.Ux$ (mínino desperdicio, utilizando el menor número de plantillas).

Problem	No. Template	Templates	Pressings	Overall Desv.	T.Desv	Waste
Cat Food	2	[1,1,1,2,2,2,0]	250000			
		[0,0,0,0,0,2,7]	157143	0.80	-3.85/ 1.79	29287
Herbs Cartoon	2	[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1, 0,1,1,1,1,1,1,1,2,2,2,3,3,4,4]	65911			
		[0,0,0,0,0,0,0,0,0,1,1,1,1,1,1, 6,1,1,1,2,2,2,2,1,6,6,2,2,1,1]	16363	2.99	-8.58/ 9.85	104548
Mz.Inserts	3	[1,1,1,1,1,1,2,2,2,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,0,0,0,0,1,0,0,0]	54000			
		[0,0,0,0,1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,1,1, 0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,3,3,3,3,2,2,2,12]	33750			
		[0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1, 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,0]	146000	2.97	-10.00/ 8.00	277500

de acuerdo con el torneo basado en el rangos. Se han considerado los algoritmos que no mostraron diferencias estadísticas contra el algoritmo de control. Sin embargo, debemos tomar en cuenta que el bajo número de instancias problemáticas podría haber tenido una gran influencia en los resultados de estas pruebas y, como consecuencia, podríamos haber obtenido un resultado no concluyente. Por lo tanto, volvemos al torneo basado en rangos y lo aplicamos al conjunto de algoritmos para los cuales no se observó diferencias estadísticamente significativas.

Figura 4.16: Torneo de las diferentes técnicas de alto rendimiento, aplicadas sobre el TDP, para las 21 metaheurísticas básicas de la tabla 3.24 y 29 metaheurísticas integrativas de la tabla 4.28, que no muestran diferencias significativas de rendimiento. Cada caja comprende desde el primer al tercer cuartil de la distribución, la mediana (2do cuartil) es indicada con una línea vertical, la zona sin valores atípicos (whiskers span) es de 1.5 veces el rango intercuartil, además los valores atípicos están marcados con un signo '+'.



El resultado de este nuevo análisis proporciona una observación interesante sobre el comportamiento de los algoritmos. En primer lugar, consideremos la clasificación que se muestra en la figura 4.16. Este ranking apunta que Ma.Ts.P*.A2.Gd ha resultado el algoritmo mejor clasificado.

El análisis estadístico que hemos aplicado nos está mostrando que hay diferencias estadísticas

Tabla 4.34: Resultados para los tests de Friedman e Iman-Davenport, de las diferentes técnicas de alto rendimiento, aplicadas sobre el TDP, para las 21 metaheurísticas básicas de la tabla 3.24 y 29 metaheurísticas integrativas de la tabla 4.28, que no muestran diferencias significativas de rendimiento.

	Friedman value	Critical χ^2 value	Iman-Davenport value	Critical F_F value
All 50	81.261176	66.338649	2.472243	1.482944

significativas entre las técnicas; consulte la tabla 4.34, de igual forma, el test de Holm (véase la tabla 4.35), nos indica que las diferencias para este algoritmo de control, respecto a la mayoría de los algoritmos, no son estadísticamente significativas.

Esto nos ha llevado a profundizar un poco más en el análisis de estos resultados, ya que aún abrigamos la idea de que podemos aclarar mejor este panorama y obtener un resultado más concluyente. Es por ello que nos vamos a concentrar ahora en aquellos algoritmos que no parecen estar dominados por otras técnicas, como se puede ver en las tablas 3.22 y 4.26.

Para la comparación, hemos seleccionado aquellos algoritmos que no están dominados por ningún otro método en su clase (es decir, básico o integrativo). Las columnas de la extrema derecha en tablas 3.22 y 4.26 marcan los algoritmos no diferenciados para las clases de métodos básicos e integradores, respectivamente. Así, consideramos los 6 métodos básicos no dominados y los 3 métodos híbridos no dominados, la idea es compararlos de nuevo en un torneo basado en rangos. Como podemos observar en la figura 4.17 la técnica que se coloca en la primera posición corresponde a la versión del algoritmo memético Ma.Hc.P*.A2.Ux, la cual trabaja bajo la representación alternativa que hemos llamado Primal, que emplea la ruptura de simetrías, junto con el operador de cruce Ux y la recombinación por medio de dos padres. Nótese que este algoritmo está ubicado en la segunda posición del ranking que se muestra en la tabla 4.16.

Como los valores resultantes para los tests de Friedman e Iman-Davenport son menores a los valores críticos, es evidente que no hay diferencias significativas entre los algoritmos que son objeto de este estudio. Los resultados del test de Holm pueden aclarar aún más esta conclusión. Así que, podemos concluir que no hay argumentos para considerar diferencias significativas entre estas técnicas (véase las tablas 4.36 y 4.37).

Finalmente, hemos considerado realizar una comparación cruzada entre los enfoques básico e integrativos (metaheurísticas vs. meméticos). Debido a la considerable cantidad de métodos que podemos incluir en esta comparativa, nos inclinamos por una selección de los métodos que se incluirán en la comparación basada en el concepto de *Dominio de Pareto* presentado en optimización multiobjetivo [339]. Es decir, enfocándonos en el contexto de este trabajo de investigación, el procedimiento de maximizar el número de soluciones encontradas para cada instancia de problema podría considerarse como un objetivo dentro de un enfoque multiobjetivo.

Para la comparación, hemos seleccionado aquellos algoritmos que no están dominados por ningún otro método en su clase (es decir, básico o integrativo). Específicamente, hemos seleccionado seis métodos básicos de la tabla 3.22 y tres técnicas de la tabla 4.26. Como podemos observar en la figura 4.18, estos también se representan en un eje tridimensional, donde podemos encontrar la relación entre los escenarios y los algoritmos utilizados.

Usando el mismo torneo basado en rangos, para el conjunto de algoritmos no dominados (ver

Tabla 4.35: Resultados del test de Holm, de las diferentes técnicas de alto rendimiento, aplicadas sobre el TDP, para las 21 metaheurísticas básicas de la tabla 3.24 y 29 metaheurísticas integrativas de la tabla 4.28, que no muestran diferencias significativas de rendimiento, usando Ma.Ts.P*.A2.Gd como algoritmo de control.

i	algorithm	z-statistic	p-value	α/i
1	Ma.Hc.P*.A2.Gd	2.801e-002	4.888e-001	5.000e-002
2	Ma.Hc.P*.A2.Ux	2.100e-001	4.168e-001	2.500e-002
3	Ma.Ts.P*.A4.Gd	3.781e-001	3.527e-001	1.667e-002
4	Ma.Ts.P.A2.Ux	3.921e-001	3.475e-001	1.250e-002
5	Ma.Hc.P*.A4.Ux	4.201e-001	3.372e-001	1.000e-002
6	Ma.Hc.P*.A4.Gd	5.041e-001	3.071e-001	8.333e-003
7	Ma.Hc.D*.A2.Gd	6.581e-001	2.552e-001	7.143e-003
8	Ma.Hc.P.A4.Gd	7.001e-001	2.419e-001	6.250e-003
9	Ma.Hc.P.A2.Ux	7.281e-001	2.333e-001	5.556e-003
10	Ma.Hc.D*.A4.Gd	7.562e-001	2.248e-001	5.000e-003
11	Ma.Ts.P.A4.Gd	7.842e-001	2.165e-001	4.545e-003
12	Ma.Hc.P.A4.Ux	8.122e-001	2.083e-001	4.167e-003
13	Ma.Hc.D.A2.Gd	8.402e-001	2.004e-001	3.846e-003
14	Ma.Ts.P.A4.Ux	8.402e-001	2.004e-001	3.571e-003
15	Ts.P	1.008e+000	1.567e-001	3.333e-003
16	Ma.Ts.P*.A2.Ux	1.036e+000	1.501e-001	3.125e-003
17	Ma.Ts.P*.A4.Ux	1.064e+000	1.436e-001	2.941e-003
18	Hc.D*	1.162e+000	1.226e-001	2.778e-003
19	Ga.D*.A4.Gd	1.218e+000	1.116e-001	2.632e-003
20	Hc.P*	1.246e+000	1.063e-001	2.500e-003
21	Ma.Hc.D.A2.Ux	1.344e+000	8.943e-002	2.381e-003
22	Ma.Hc.D*.A4.Ux	1.386e+000	8.283e-002	2.273e-003
23	Ma.Ts.D*.A2.Gd	1.428e+000	7.660e-002	2.174e-003
24	Ma.Hc.D*.A2.Ux	1.470e+000	7.074e-002	2.083e-003
25	Ga.D*.A2.Gd	1.680e+000	4.645e-002	2.000e-003
26	Ts.P*	1.694e+000	4.510e-002	1.923e-003
27	Ga.P.A4.Gd	1.694e+000	4.510e-002	1.852e-003
28	Ma.Hc.D.A4.Ux	1.778e+000	3.767e-002	1.786e-003
29	Hc.P	1.876e+000	3.030e-002	1.724e-003
30	Ma.Ts.D*.A4.Gd	1.904e+000	2.843e-002	1.667e-003
31	Ma.Ts.P.A2.Gd	1.918e+000	2.753e-002	1.613e-003
32	Ma.Hc.P.A2.Gd	1.946e+000	2.580e-002	1.563e-003
33	Ga.P*.A4.Gd	2.016e+000	2.188e-002	1.515e-003
34	Ga.P.A2.Gd	2.086e+000	1.847e-002	1.471e-003
35	Ma.Ts.D.A2.Gd	2.282e+000	1.123e-002	1.429e-003
36	Ma.Hc.D.A4.Gd	2.408e+000	8.010e-003	1.389e-003
37	Ga.P*.A2.Gd	2.408e+000	8.010e-003	1.351e-003
38	Ga.P*.A2.Ux	2.535e+000	5.630e-003	1.316e-003
39	Ga.D*.A2.Ux	2.549e+000	5.409e-003	1.282e-003
40	Ga.P*.A4.Ux	2.591e+000	4.792e-003	1.250e-003
41	Ts.D*	2.619e+000	4.416e-003	1.220e-003
42	Ga.P.A2.Ux	2.647e+000	4.066e-003	1.190e-003
43	Ma.Ts.D*.A4.Ux	2.647e+000	4.066e-003	1.163e-003
44	Ma.Ts.D.A4.Ux	2.731e+000	3.161e-003	1.136e-003
45	Ga.D.A4.Gd	2.815e+000	2.442e-003	1.111e-003
46	Ga.D*.A4.Ux	3.067e+000	1.082e-003	1.087e-003
47	Ga.D.A2.Gd	3.081e+000	1.033e-003	1.064e-003
48	Ga.P.A4.Ux	3.109e+000	9.398e-004	1.042e-003
49	Ga.D.A2.Ux	3.319e+000	4.522e-004	1.020e-003

figura 4.19), podemos observar que la técnica Ma.Hc.P*.A2.Ux es el algoritmo que obtiene la mejor clasificación. Este algoritmo muestra el mejor desenvolvimiento en dos de los escenarios abordados en este problema, a saber, Herbs y Magazine. Como se muestra en la tabla 4.38, esta superioridad es estadísticamente significativas en todos los casos (a excepción de Ma.Hc.D.A4.Gd en el escenario

Figura 4.17: Torneo de las diferentes técnicas de alto rendimiento, aplicadas sobre el TDP, para las 6 metaheurísticas básicas de la tabla 3.22 y 3 metaheurísticas integrativas de la tabla 4.26, que no son dominados por otra técnica. Cada caja comprende desde el primer al tercer cuartil de la distribución, la mediana (2do cuartil) es indicada con una línea vertical, la zona sin valores atípicos (whiskers span) es de 1.5 veces el rango intercuartil, además los valores atípicos están marcados con un signo '+'.

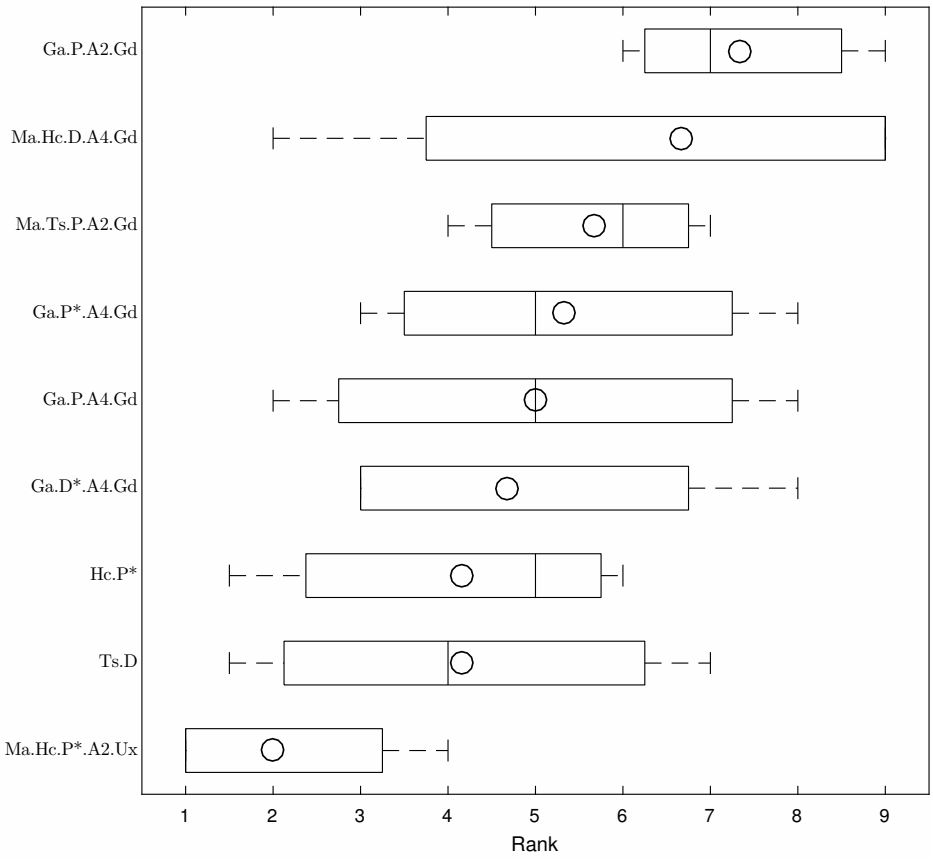


Tabla 4.36: Resultados de los tests de Friedman e Iman-Davenport, de las diferentes técnicas de alto rendimiento, aplicadas sobre el TDP, para las 6 metaheurísticas básicas de la tabla 3.22 y 3 metaheurísticas integrativas de la tabla 4.26, que no son dominados por otra técnica.

	Friedman value	Critical χ^2 value	Iman-Davenport value	Critical F_F value
All 9	7.711111	15.507313	0.946794	2.591096

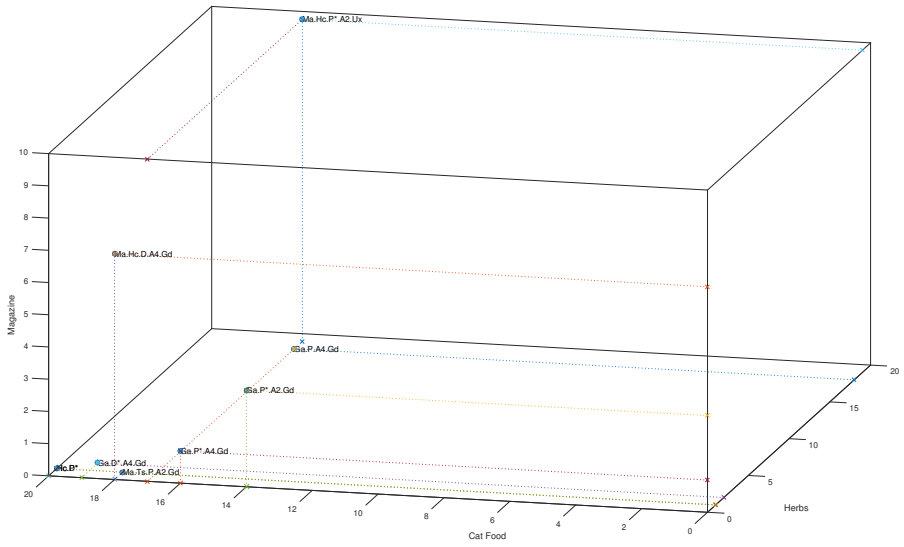
Magazine). Para el caso del escenario de menor complejidad (Cat Food Cartons), las variantes de la técnica Hill Climbing, parecen mostrar los mejores resultados, superando los enfoques integradores cuya complejidad adicional no compensa en este escenario.

Debemos tomar en cuenta que las versiones algorítmicas con ruptura de simetría muestran una tendencia a lograr una mejor clasificación. Este es un resultado muy interesante que nos permite

Tabla 4.37: Resultados del test de Holm, de las diferentes técnicas de alto rendimiento, aplicadas sobre el TDP, para las 6 metaheurísticas básicas de la tabla 3.22 y 3 metaheurísticas integrativas de la tabla 4.26, que no son dominados por otra técnica, usando Ma.Hc.P*.A2.Ux como algoritmo de control.

i	algorithm	z-statistic	p-value	α/i
1	Ts.D	9.690e-01	1.663e-01	5.000e-02
2	Hc.P*	9.690e-01	1.663e-01	2.500e-02
3	Ga.D*.A4.Gd	1.193e+00	1.165e-01	1.667e-02
4	Ga.P.A4.Gd	1.342e+00	8.986e-02	1.250e-02
5	Ga.P*.A4.Gd	1.491e+00	6.802e-02	1.000e-02
6	Ma.Ts.P.A2.Gd	1.640e+00	5.053e-02	8.333e-03
7	Ma.Hc.D.A4.Gd	2.087e+00	1.844e-02	7.143e-03
8	Ga.P.A2.Gd	2.385e+00	8.536e-03	6.250e-03

Figura 4.18: Representación del rendimiento de los algoritmos de alto rendimiento, aplicados sobre el TDP, para las 6 metaheurísticas básicas de la tabla 3.22 y 3 metaheurísticas integrativas de la tabla 4.26, que no son dominados por otra técnica.



respaldar la utilidad de la ruptura de la simetría para mejorar las capacidades de resolución en el TDP, lo que podría conducir al diseño de otros algoritmos para hacer frente a este problema.

4.5.3 Resultados Experimentales para los Esquemas Cooperativos Sobre el TDP

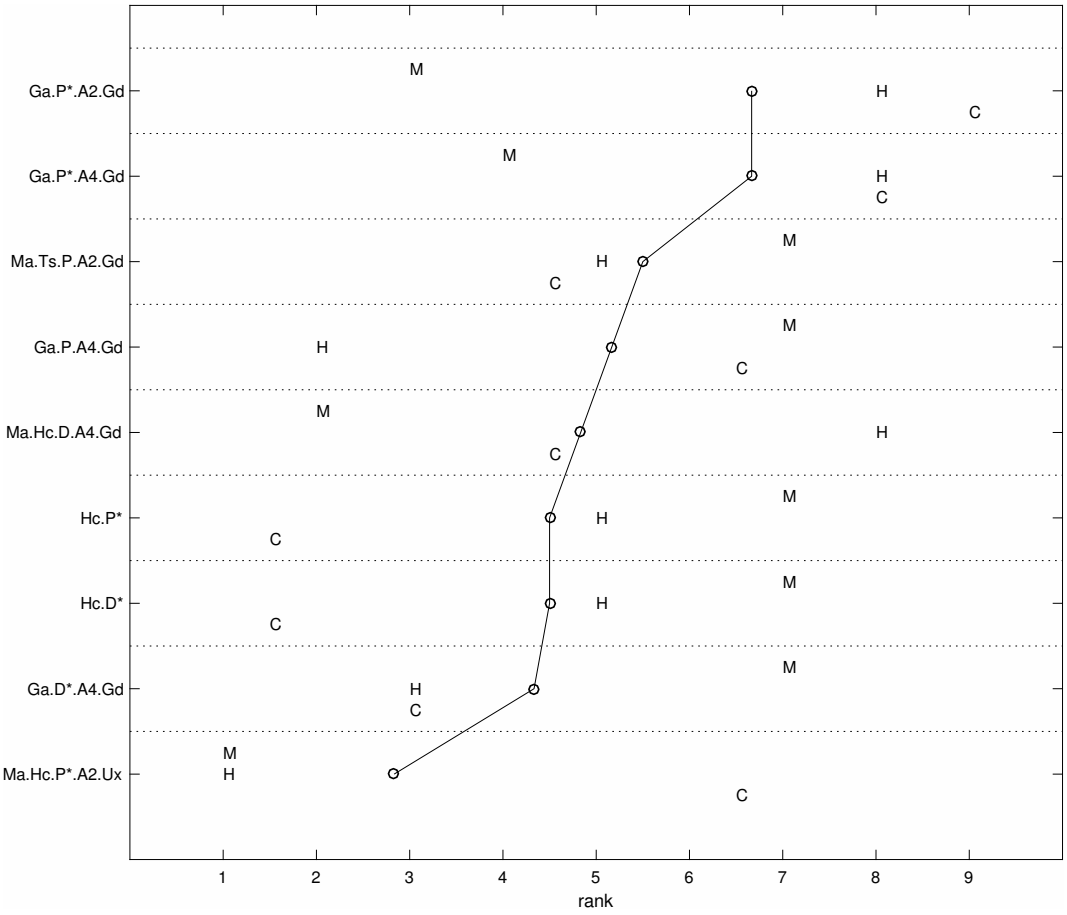
Los esquemas colaborativos que vamos a emplear en esta experimentación ya han sido explicados ampliamente a partir de la sección 4.3, así como la sección 4.4.4 en este capítulo.

Tabla 4.38: Comparación de la superioridad estadística de Ma.Hc.P*.A2.Ux con los algoritmos restantes en cada una de los tres escenarios del TDP tomados de la tabla 3.17. Cada entrada en la tabla contiene tres símbolos que corresponden (de izquierda a derecha) a Cat Food Cartons, Herbs Cartons and Magazine Inserts: • (resp. ○) indica la diferencia de rendimiento sobre el escenario correspondiente (resp. no satisface) es estadísticamente significativas en $\alpha = 0.05$ usando el test de ranksum.

Hc.P*	Hc.D*	Ga.P.A4.Gd	Ga.P*.A4.Gd
•/•/•	•/•/•	•/•/•	•/•/•

Ga.P*.A2.Gd	Ga.D*.A4.Gd	Ma.Ts.P.A2.Gd	Ma.Hc.D.A4.Gd
•/•/•	•/•/•	○/•/•	•/•/○

Figura 4.19: Distribución de rangos para las 6 metaheurísticas básicas de la tabla 3.22 y 3 metaheurísticas integrativas de la tabla 4.26, aplicados sobre el TDP. Las letras C, H, M representan el rango en cada uno de los tres escenarios del problema, y el círculo indica el rango medio del algoritmo respectivo.



Al igual que como se ha hecho en la sección relacionada con el modelo de cooperación aplicado al problema del BIBD, los algoritmos en estas colecciones, no han sido seleccionados en forma arbitraria sino que por el contrario, los hemos escogido debido a su buen desempeño al operar en forma individual, de acuerdo con la figura 4.17: así que, Ts.D corresponde a la técnica básica que parece posicionarse de mejor manera en el torneo, junto con el algoritmo Ga.D*.A4.Gd, mientras que para las técnicas híbridas, es la versión Ma.Hc.P*.A2.Ux del algoritmo Memético que se ha posicionado de mejor manera. Sin embargo, hemos optado por utilizar el algoritmo Ma.Ts.P.A2.Gd, para formar un grupo con los que parecen ser las cuatro mejores técnicas, además, de que en esta colección están representados los mejores métodos integrativos a lo largo de los dominios P*, P, D y D*. Hemos empleado las siguientes cuatro colecciones, esto basándonos en la experimentación que se ha realizado sobre el problema del BIBD, para seguir la misma metodología, puesto que esta nos ha indicado que su utilización puede conducirnos a la obtención de resultados interesantes, y así poder generalizar y proponer dicha metodología para futuras investigaciones que se apliquen a problemas de optimización combinatoria o de difícil solución.

- $\mathcal{A}_1$ ={Ts.D,MA.Hc.P*.A2.Ux}
- $\mathcal{A}_2$ ={Ts.D,GA.D*.A4.Gd}
- $\mathcal{A}_3$ ={Ts.D,MA.Ts.P.A2.Gd}
- $\mathcal{A}_4$ ={MA.Hc.P*.A2.Ux,MA.Ts.P.A2.Gd}

Además, cada colección representa una posible forma de recombinar las diferentes técnicas que han sido consideradas en esta investigación: así, $\mathcal{A}_1$ representa la cooperación entre los dos esquemas de representación alternativa (en este caso, los modelos que hemos llamado Primal y Dual) con las correspondientes configuraciones de con/sin ruptura de simetría (es decir, D-P*), $\mathcal{A}_2$ representa la cooperación de técnicas que trabajan en los mismos dominios de computo del modelo Dual (es decir, D-D*), $\mathcal{A}_3$ representa la cooperación de métodos que trabajan en los distintos dominios de computo sin ruptura de simetría (es decir, P-D) y $\mathcal{A}_4$ el esquema en el que los métodos trabajan en igual dominio de computo pero con diferentes políticas para la ruptura de simetría (es decir, P-P*).

4.5.3.1 Análisis de Factores de Diseño de los Esquemas Cooperativos Sobre el TDP

De igual forma, como lo hemos aplicado sobre el problema del BIBD (en el cual se implementó un importante grupo de técnicas), hemos combinado una gran cantidad de algoritmos que son el resultado de las diferentes combinaciones posibles, entre los elementos considerados (i.e. representaciones, políticas de migración/recepción, etc.), es conveniente realizar un análisis de diferentes aspectos a lo largo de un conjunto de dimensiones correspondientes a la variedad de decisiones de diseño con respecto al número de agentes involucrados, su topología o las políticas de comunicación.

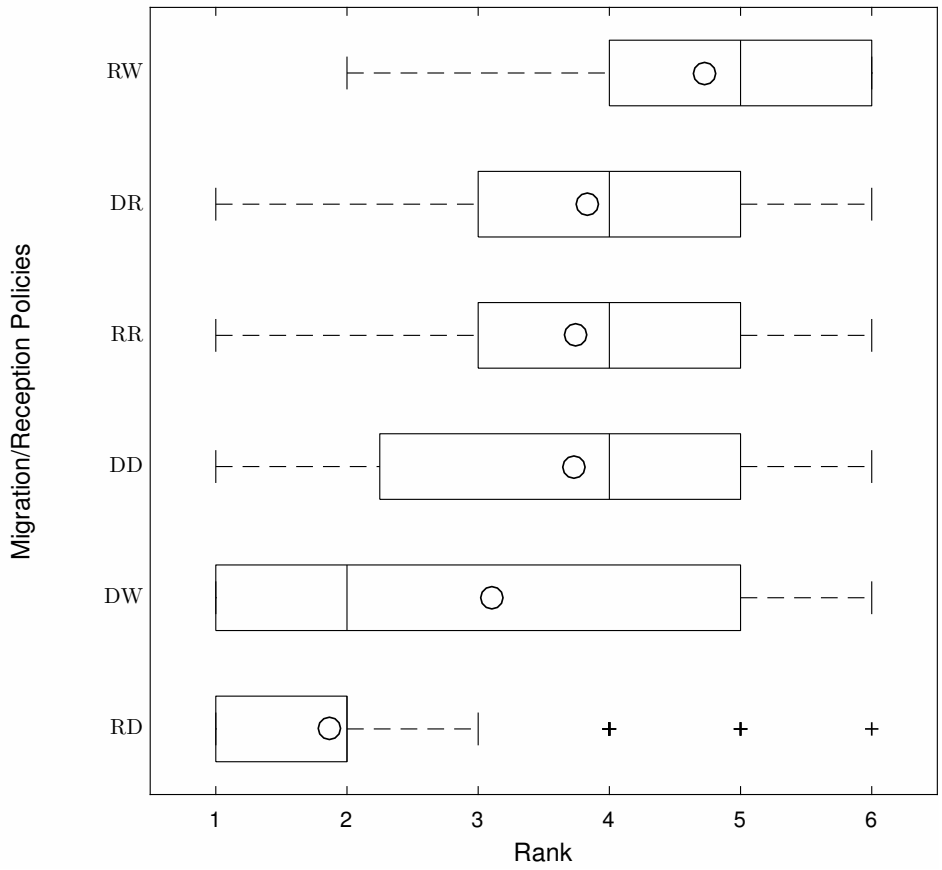
En la figura 4.20 se muestra la distribución resultante del torneo para la diferentes políticas empleadas. Según el torneo basado en rangos aplicado, la mejor política de M/R es RD (realizar envío de información, es decir solución candidata, seleccionada al azar, reemplazo de nuevas soluciones considerando si esta diversifica la población).

Esto se confirma con las pruebas de Friedman e Iman-Davenport (tabla 4.39, en el nivel estándar de $\alpha = 0.05$) esto indica que hay diferencias estadísticamente significativas entre las políticas, lo cual es corroborado al aplicar el test de Holm (tabla 4.40).

Aquí se muestra que la configuración de la política M/R correspondiente a RD, es significativamente mejor que las políticas restantes.

Ahora, vamos a considerar realizar un análisis a lo largo del eje de las topologías. Para este

Figura 4.20: Torneo para las diferentes técnicas colaborativas, aplicadas sobre el problema del TDP para los escenarios tomados de [251], agrupadas a lo largo de la dimensión: políticas de migración/recepción y reemplazo. Considerando las seis combinaciones (es decir, DD, DR, DW, RD, RR y RW) utilizadas (véase la sección 4.4.1). Cada caja comprende desde el primer al tercer cuartil de la distribución, la mediana (2do cuartil) es indicada con una línea vertical, la zona sin valores atípicos (whiskers span) es de 1.5 veces el rango intercuartil, además los valores atípicos están marcados con un signo '+'.



.

Tabla 4.39: Resultados de los tests de Friedman e Iman-Davenport ($\alpha = 0.05$), para las diferentes técnicas colaborativas, aplicadas sobre el problema del TDP para los escenarios tomados de [251], agrupadas a lo largo de la dimensión: políticas de migración/recepción y reemplazo. Considerando las seis combinaciones (es decir, DD, DR, DW, RD, RR y RW) utilizadas (véase la sección 4.4.1).

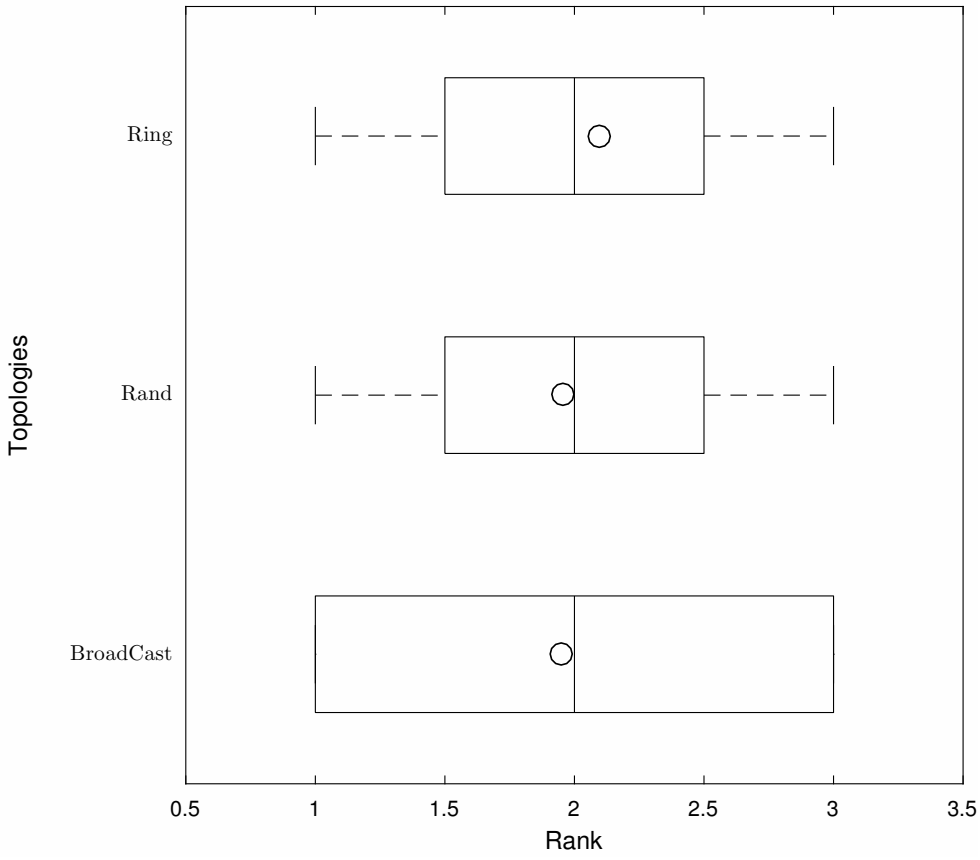
	Friedman value	Critical χ^2 value	Iman-Davenport value	Critical F_F value
All 6	184.779221	11.070498	49.486271	2.226720

aspecto, vamos a poner a competir las tres topologías mencionadas, para las 288/3=96 variantes de algoritmos resultantes. Los resultados que podemos ver en la tabla 4.15, corresponde al test de Fried-

Tabla 4.40: Resultados del test de Holm ($\alpha = 0.05$), para las diferentes técnicas colaborativas, aplicadas sobre el problema del TDP para los escenarios tomados de [251], agrupadas a lo largo de la dimensión: políticas de migración/recepción y reemplazo. Considerando las seis combinaciones (es decir, DD, DR, DW, RD, RR y RW) utilizadas (véase la sección 4.4.1), usando RD como algoritmo de control.

i	algorithm	z-statistic	p-value	α/i
1	DW	5.579e+00	1.212e-08	5.000e-02
2	DD	8.392e+00	2.397e-17	2.500e-02
3	RR	8.455e+00	1.397e-17	1.667e-02
4	DR	8.882e+00	3.296e-19	1.250e-02
5	RW	1.288e+01	2.923e-38	1.000e-02

Figura 4.21: Torneo para las diferentes técnicas colaborativas, aplicadas sobre el problema del TDP para los escenarios tomados de [251], agrupadas a lo largo de la dimensión: Topologías (Ra, Ri y BC, véase la sección 4.4.1). Cada caja comprende desde el primer al tercer cuartil de la distribución, la mediana (2do cuartil) es indicada con una línea vertical, la zona sin valores atípicos (whiskers span) es de 1.5 veces el rango intercuartil, además los valores atípicos están marcados con un signo '+'.



.

Tabla 4.41: Resultados de los tests de Friedman e Iman-Davenport ($\alpha = 0.05$), Torneo para las diferentes técnicas colaborativas, aplicadas sobre el problema del TDP para los escenarios tomados de [251], agrupadas a lo largo de la dimensión: Topologías (Ra, Ri y BC, véase la sección 4.4.1).

	Friedman value	Critical χ^2 value	Iman-Davenport value	Critical F_F value
All 3	3.956446	5.991465	1.985012	3.011477

man e Iman-Davenport ($\alpha = 0.05$) sobre las topologías estudiadas, como podemos apreciar, no parece haber diferencias estadísticamente significativas.

Tabla 4.42: Resultados del test de Holm, para las diferentes técnicas colaborativas, aplicadas sobre el problema del TDP para los escenarios tomados de [251], agrupadas a lo largo de la dimensión: Topologías (Ra, Ri y BC, véase la sección 4.4.1), usando Broadcast como algoritmo de control.

i	algorithm	z-statistic	p-value	α/i
1	Rand	6.261e-02	4.750e-01	5.000e-02
2	Ring	1.753e+00	3.980e-02	2.500e-02

Los resultados obtenidos para el test de Holm según las topologías utilizadas, empleando BROADCAST como algoritmo de control se pueden revisar en la tabla 4.42 y corrobora la tesis de que no existen diferencias entre las topologías.

Ahora si fijamos nuestra atención en la cantidad de agentes utilizados en el modelo cooperativo, que en este caso hemos considerado $n \in [2,5]$ (288/4 = 72 diferentes variantes de los algoritmos empleados).

Tanto las pruebas de Friedman como las pruebas de Iman-Davenport indican que se presentan diferencias significativas entre el número de agentes aplicados, – véase la tabla 4.43–. Para afianzar de mejor manera los argumentos aquí expuestos hemos optado por realizar el test de Holm, usando $n = 5$ (que corresponde al mejor valor de la clasificación) como algoritmo de control.

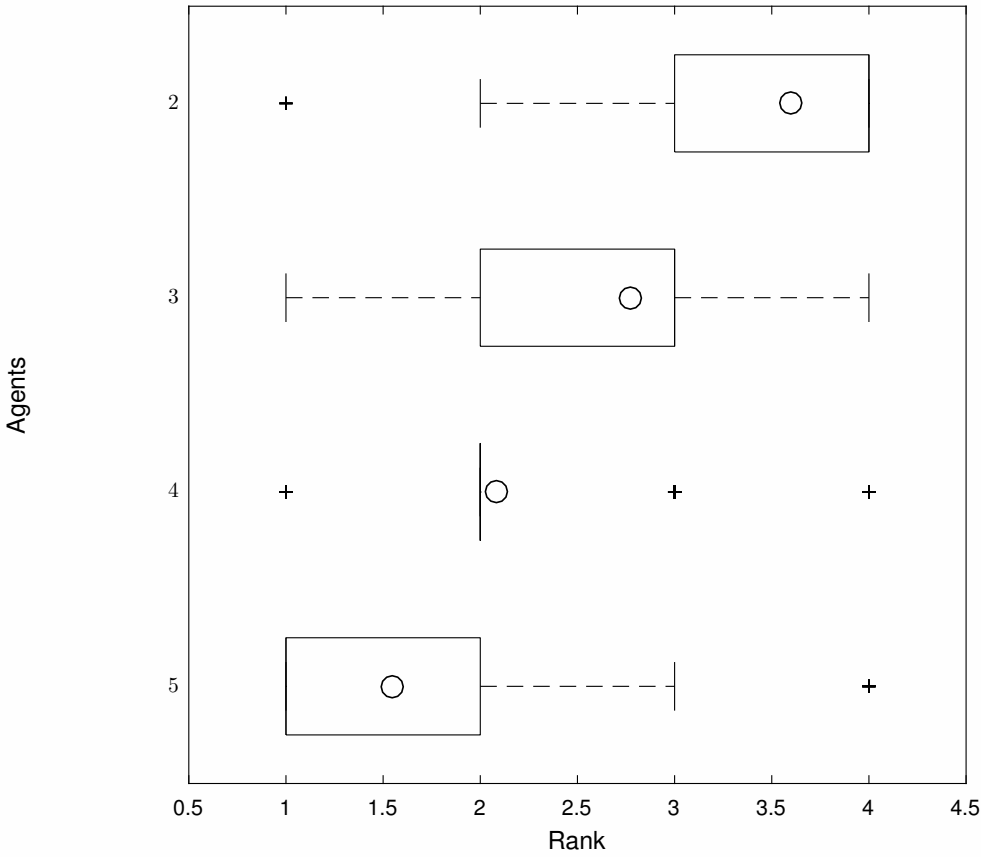
Como se muestra en la tabla 4.44, el resultado nos reafirma que efectivamente existen diferencias estadísticas significativas para $n = 5$, respecto a las demás configuraciones del número de agentes.

Como podemos observar en la figura 4.23, la combinación que corresponde a $\mathcal{A}_2$ tiene el mejor lugar en el torneo, sin embargo no es concluyente que éste sea el mejor grupo, ya que los tests de Friedman e Iman-Davenport y el de Holm nos indica que hay diferencias significativas con respecto a los grupos $\mathcal{A}_1$ y $\mathcal{A}_4$, pero no respecto al grupo $\mathcal{A}_3$ (véase las tablas 4.45 y 4.46).

4.5.3.2 Análisis de los Mejores Modelos Cooperativos para el TDP

Ahora consideremos los modelos cooperativos más efectivos. En los siguientes párrafos, consideramos todos los esquemas colaborativos que logran encontrar al menos un 70% de soluciones factibles en cada uno de los escenarios abordados (Cat Food, Herbs y Magazine), para cada ejecución lanzada sobre el problema del Diseño de Plantillas. Este conjunto de algoritmos está compuesto por 17 variantes (los resultados se pueden ver en la tabla 4.47). La tabla muestra el porcentaje de soluciones

Figura 4.22: Torneo para las diferentes técnicas colaborativas, aplicadas sobre el problema del TDP para los escenarios tomados de [251], agrupadas a lo largo de la dimensión: Número de agentes ($n \in [2,5]$). Cada caja comprende desde el primer al tercer cuartil de la distribución, la mediana (2do cuartil) es indicada con una línea vertical, la zona sin valores atípicos (whiskers span) es de 1.5 veces el rango intercuartil, además los valores atípicos están marcados con un signo '+'.



.

Tabla 4.43: Resultados de los tests de Friedman e Iman-Davenport ($\alpha = 0.05$), para las diferentes técnicas colaborativas, aplicadas sobre el problema del TDP para los escenarios tomados de [251], agrupadas a lo largo de la dimensión: Número de agentes ($n \in [2,5]$).

	Friedman value	Critical χ^2 value	Iman-Davenport value	Critical F_F value
All 4	305.204651	7.814728	192.215095	2.618779

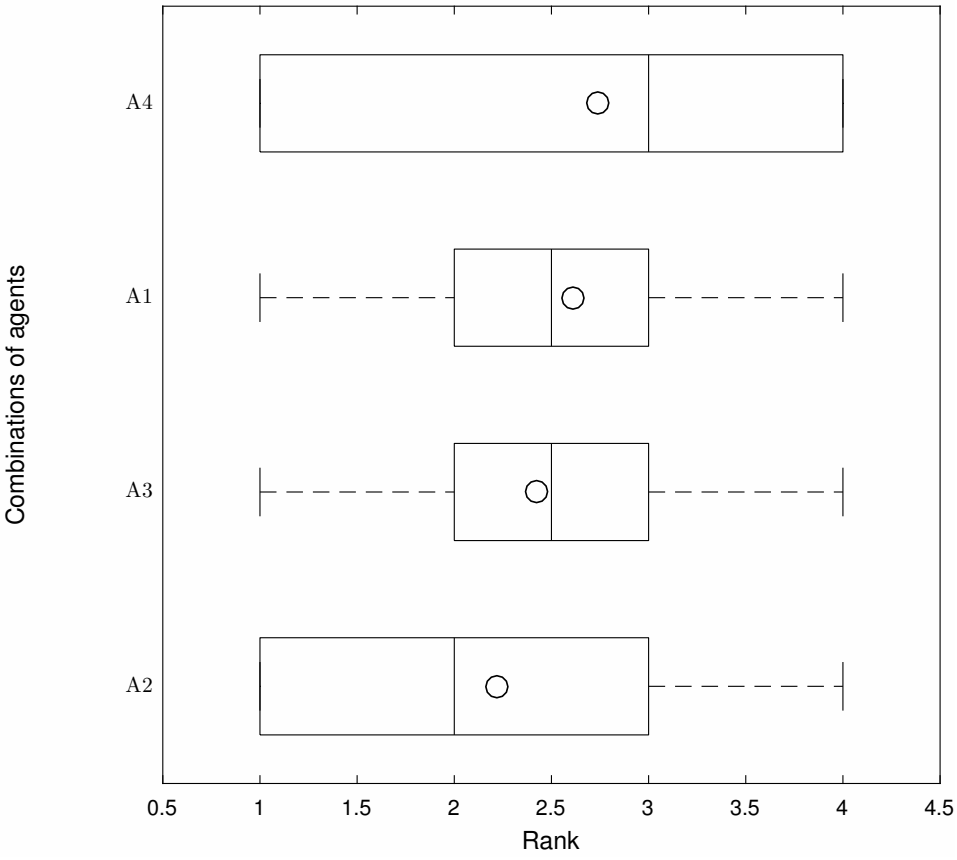
factibles que han alcanzado los esquemas colaborativos utilizados (expresado en porcentaje de éxito logrado en alguno de los escenarios tratados).

Si comprobamos la frecuencia relativa de cada parámetro de diseño particular entre estos 17 modelos algorítmicos seleccionados, obtenemos los resultados que se muestran en la tabla 4.48. Los resultados son consistentes con el análisis estadístico de la sección anterior. Por lo tanto, podemos

Tabla 4.44: Resultados del test de Holm ($\alpha = 0.05$), para las diferentes técnicas colaborativas, aplicadas sobre el problema del TDP para los escenarios tomados de [251], agrupadas a lo largo de la dimensión: Número de agentes ($n \in [2,5]$), usando $n = 5$ como algoritmo de control.

i	algorithm	z-statistic	p-value	α/i
1	4	4.258e+00	1.029e-05	5.000e-02
2	3	9.824e+00	4.429e-23	2.500e-02
3	2	1.647e+01	2.855e-61	1.667e-02

Figura 4.23: Torneo para las diferentes técnicas colaborativas, aplicadas sobre el problema del TDP para los escenarios tomados de [251], considerando los grupos de agentes: $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ y $\mathcal{A}_4$, véase la sección 4.5.3. Cada caja comprende desde el primer al tercer cuartil de la distribución, la mediana (2do cuartil) es indicada con una línea vertical, la zona sin valores atípicos (whiskers span) es de 1.5 veces el rango intercuartil, además los valores atípicos están marcados con un signo '+'.



.

ver que la política de RD para migración/recepción muestra una mayor presencia en los modelos algorítmicos. En cuanto a la topología, BROADCAST está presente en más de la mitad de los modelos. Finalmente, como se puede notar para este caso particular, el número de agentes que parece ofrecer un buen rendimiento con mayor frecuencia corresponde a la configuración de 5 algoritmos trabajando

Tabla 4.45: Resultados de los tests de Friedman e Iman-Davenport ($\alpha = 0.05$), para las diferentes técnicas colaborativas, aplicadas sobre el problema del TDP para los escenarios tomados de [251], considerando los grupos de agentes: $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ y $\mathcal{A}_4$, véase la sección 4.5.3.

	Friedman value	Critical χ^2 value	Iman-Davenport value	Critical F_F value
All 4	19.527907	7.814728	6.681309	2.618779

Tabla 4.46: Resultados del test de Holm ($\alpha = 0.05$), para las diferentes técnicas colaborativas, aplicadas sobre el problema del TDP para los escenarios tomados de [251], considerando los grupos de agentes: $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ y $\mathcal{A}_4$, véase la sección 4.5.3, usando $\mathcal{A}_2$ como algoritmo de control.

i	algorithm	z-statistic	p-value	α/i
1	$\mathcal{A}_3$	1.625e+00	5.209e-02	5.000e-02
2	$\mathcal{A}_1$	3.138e+00	8.512e-04	2.500e-02
3	$\mathcal{A}_4$	4.128e+00	1.832e-05	1.667e-02

en forma simultánea, según las pruebas realizadas.

La figura 4.24 muestra la distribución de los rangos correspondiente para estas 17 variaciones de modelos. Se ejecutó un análisis estadístico al aplicar los tests estadísticos de Friedman e Iman-Davenport ($\alpha = 0.05$) para los algoritmos mostrados en la tabla 4.47.

En este caso, los valores estadísticos obtenidos fueron ligeramente más altos que los valores críticos, y por lo tanto podemos indicar que existen diferencias significativas en sus rangos al nivel estándar $\alpha = 0.05$, véase la tabla 4.49.

Posteriormente, hemos llevado a cabo el test de Holm para determinar si existen diferencias significativas con respecto a un algoritmo de control (en este caso, el algoritmo cooperativo en el que participan 5 agentes y emplea la topología BROADCAST), Bc5(Ts.D,Ma.Hc.P*.A2.Ux)RD, el algoritmo con el mejor rango medio según figura 4.24. Los resultados se muestran en la tabla 4.50 donde vemos que hay diferencias estadísticas significativas para el 35.29% de los esquemas de colaboración utilizados, pero no hay muestras de diferencias estadísticamente significativas en más del 70%.

4.5.4 Discusión: Integrativos vs Cooperativos para el Problema del TDP

Esta sección la dedicaremos a discutir (y comparar) el desempeño de los diferentes enfoques colaborativos, así como las técnicas integrativas que hemos empleado para abordar el problema de Diseño de Plantillas (TDP). Para ello hemos seleccionado las 9 versiones de las técnicas integrativas (catalogadas como no-dominadas) que no muestran diferencias estadísticamente significativas en su desempeño al compararlas entre sí (véase la tabla 4.37). En cuanto a las técnicas colaborativas, estamos tomando las 11 configuraciones de los modelos de cooperación que de igual manera no han mostrado diferencias entre ellos (véase la tabla 4.50).

Para ello, y al igual que como se ha abordado en las secciones anteriores de esta investigación, hemos aplicado un torneo de rangos, para determinar cuáles de estas técnicas se ubican en la mejor

Tabla 4.47: Columna central, porcentaje (%) de cada diseño cooperativo que logró encontrar más del 70% de soluciones factibles para el problema del TDP, de los escenarios tomados de la tabla 3.17, resueltas por los modelos cooperativos (identificados en la primera columna). En la columna derecha se muestran el grupo de algoritmos que operan en el modelo. El número de ejecuciones de cada algoritmo corresponde a $n * 10$, donde $n =$ Número de agentes. Incluye los modelos primal/dual, con/sin ruptura de simetrías: P Primal sin ruptura de simetrías, P* Primal con ruptura de simetrías; D Dual sin ruptura de simetrías, y D* Dual con ruptura de simetrías.

Algorithms	> % Fact.	Collection
Ra2(Ts.D,Ga.D*.A4.Gd)DW	70.00 %	$\mathcal{A}_2$
Bc5(Ts.D,Ga.D*.A4.Gd)RD	72.00 %	$\mathcal{A}_2$
Bc4(Ts.D,Ma.Ts.P.A2.Gd)RD	72.50 %	$\mathcal{A}_3$
Ri3(Ts.D,Ma.Hc.P*.A2.Ux)RD	73.33 %	$\mathcal{A}_1$
Ri5(Ts.D,Ga.D*.A4.Gd)RD	74.00 %	$\mathcal{A}_2$
Ra5(Ts.D,Ma.Ts.P.A2.Gd)RD	74.00 %	$\mathcal{A}_3$
Ra5(Ts.D,Ga.D*.A4.Gd)RD	76.00 %	$\mathcal{A}_2$
Ra3(Ts.D,Ma.Hc.P*.A2.Ux)RD	76.67 %	$\mathcal{A}_1$
Bc4(Ts.D,Ma.Hc.P*.A2.Ux)RD	77.50 %	$\mathcal{A}_1$
Bc4(Ts.D,Ga.D*.A4.Gd)RD	77.50 %	$\mathcal{A}_2$
Ri5(Ts.D,Ma.Ts.P.A2.Gd)RD	78.00 %	$\mathcal{A}_3$
Bc2(Ts.D,Ma.Hc.P*.A2.Ux)RD	80.00 %	$\mathcal{A}_1$
Ri5(Ts.D,Ma.Hc.P*.A2.Ux)RD	80.00 %	$\mathcal{A}_1$
Bc5(Ts.D,Ma.Ts.P.A2.Gd)RD	82.00 %	$\mathcal{A}_3$
Ri3(Ts.D,Ga.D*.A4.Gd)RD	83.33 %	$\mathcal{A}_2$
Bc2(Ts.D,Ma.Ts.P.A2.Gd)RD	85.00 %	$\mathcal{A}_3$
Bc5(Ts.D,Ma.Hc.P*.A2.Ux)RD	94.00 %	$\mathcal{A}_1$

Tabla 4.48: Frecuencia relativa de cada diseño cooperativo particular sobre el problema del TDP, que logran encontrar al menos un 70% de soluciones factibles en cada uno de los escenarios abordados tomados de la tabla 3.17, y seleccionando los algoritmos de la tabla 4.47. La columna de la izquierda indica la combinación MR de las políticas de migración(M)/recepción(R), donde M, R $\in$ {RANDOM (R), DIVERSE (D), WORST (W) como se explica en la sección 4.4.1. La columna central muestra la topología de comunicación, donde Bc = BROADCAST, Ra = RANDOM, and Ri = RING. La columna de la derecha nos indica el número de agentes que intervienen.

M/R Policy			Topology			Number of agents		
DW	1	(5.88 %)	Bc	8	(47.06 %)	$n = 2$	3	(17.65 %)
RD	16	(94.12 %)	Ra	4	(23.53 %)	$n = 3$	3	(17.65 %)
			Ri	5	(29.41 %)	$n = 4$	3	(17.65 %)
						$n = 5$	8	(47.05 %)

clasificación. En la figura 4.25 podemos apreciar, los resultados de esta prueba, donde se puede notar que la técnica mejor posicionada es la configuración colaborativa correspondiente a la versión $\mathcal{A}_1$, de la topología Broadcast, con 5 agentes operando en forma simultánea. Además, podemos asegurar que

Figura 4.24: Torneo de los diseños cooperativos que lograron encontrar más del 70% de soluciones factibles para el problema del TDP, de los escenarios tomados de la tabla 3.17, y seleccionando los algoritmos de la tabla 4.47. Incluye los modelos primal/dual, con/sin ruptura de simetrías: P Primal sin ruptura de simetrías, P* Primal con ruptura de simetrías; D Dual sin ruptura de simetrías, y D* Dual con ruptura de simetrías. Cada caja comprende desde el primer al tercer cuartil de la distribución, la mediana (2do cuartil) es indicada con una línea vertical, la zona sin valores atípicos (whiskers span) es de 1.5 veces el rango intercuartil, además los valores atípicos están marcados con un signo '+'.

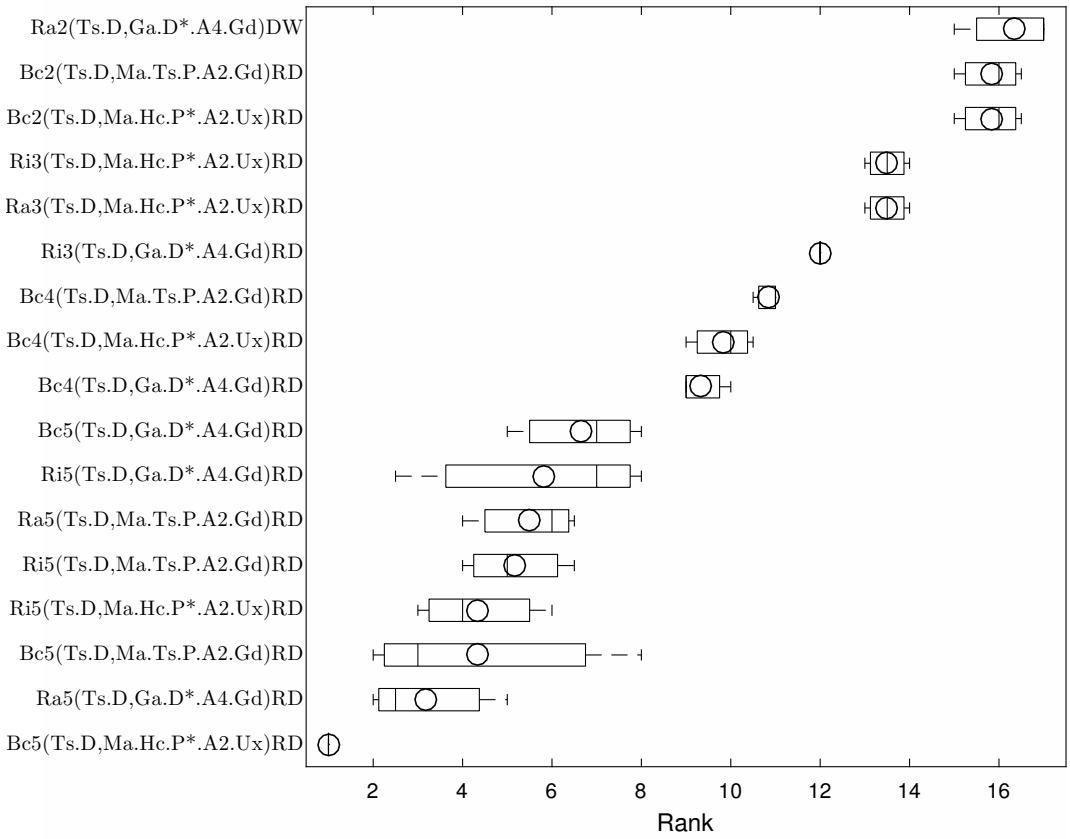


Tabla 4.49: Resultados de los tests de Friedman e Iman-Davenport ($\alpha = 0.05$), de los diseños cooperativos que lograron encontrar más del 70% de soluciones factibles para el problema del TDP, de los escenarios tomados de la tabla 3.17, y seleccionando los algoritmos de la tabla 4.47.

	Friedman value	Critical χ^2 value	Iman-Davenport value	Critical F_F value
All 17	45.274510	26.296228	33.223022	1.971683

todas las versiones correspondientes a los esquemas colaborativos, se han posicionado en los primeros lugares del torneo, quedando los últimos lugares para las técnicas integrativas.

Las pruebas correspondientes a Friedman, así como las de Iman-Davenport indican que se presentan diferencias estadísticamente significativas entre el grupo de técnicas que estamos comparando en

Tabla 4.50: Resultados del test de Holm, de los diseños cooperativos que lograron encontrar más del 70% de soluciones factibles para el problema del TDP, de los escenarios tomados de la tabla 3.17, y seleccionando los algoritmos de la tabla 4.47, usando Bc5(Ts.D,Ma.Hc.P*.A2.Ux)RD como algoritmo de control. Incluye los modelos primal/dual, con/sin ruptura de simetrías: P Primal sin ruptura de simetrías, P* Primal con ruptura de simetrías; D Dual sin ruptura de simetrías, y D* Dual con ruptura de simetrías.

i	algorithm	z-statistic	p-value	α/i
1	Ra5(Ts.D,Ga.D*.A4.Gd)RD	5.255e-01	2.996e-01	5.000e-02
2	Bc5(Ts.D,Ma.Ts.P.A2.Gd)RD	8.085e-01	2.094e-01	2.500e-02
3	Ri5(Ts.D,Ma.Hc.P*.A2.Ux)RD	8.085e-01	2.094e-01	1.667e-02
4	Ri5(Ts.D,Ma.Ts.P.A2.Gd)RD	1.011e+00	1.561e-01	1.250e-02
5	Ra5(Ts.D,Ma.Ts.P.A2.Gd)RD	1.091e+00	1.375e-01	1.000e-02
6	Ri5(Ts.D,Ga.D*.A4.Gd)RD	1.172e+00	1.205e-01	8.333e-03
7	Bc5(Ts.D,Ga.D*.A4.Gd)RD	1.374e+00	8.466e-02	7.143e-03
8	Bc4(Ts.D,Ga.D*.A4.Gd)RD	2.021e+00	2.163e-02	6.250e-03
9	Bc4(Ts.D,Ma.Hc.P*.A2.Ux)RD	2.142e+00	1.608e-02	5.556e-03
10	Bc4(Ts.D,Ma.Ts.P.A2.Gd)RD	2.385e+00	8.541e-03	5.000e-03
11	Ri3(Ts.D,Ga.D*.A4.Gd)RD	2.668e+00	3.816e-03	4.545e-03
12	Ra3(Ts.D,Ma.Hc.P*.A2.Ux)RD	3.032e+00	1.216e-03	4.167e-03
13	Ri3(Ts.D,Ma.Hc.P*.A2.Ux)RD	3.032e+00	1.216e-03	3.846e-03
14	Bc2(Ts.D,Ma.Hc.P*.A2.Ux)RD	3.598e+00	1.606e-04	3.571e-03
15	Bc2(Ts.D,Ma.Ts.P.A2.Gd)RD	3.598e+00	1.606e-04	3.333e-03
16	Ra2(Ts.D,Ga.D*.A4.Gd)DW	3.719e+00	1.001e-04	3.125e-03

Tabla 4.51: Resultados de los tests de Friedman e Iman-Davenport ($\alpha = 0.05$), de las 9 técnicas integrativas tomadas de la tabla 4.50 y 11 colaborativas tomadas de la tabla 4.37, catalogadas como no-dominadas, que no muestran diferencias estadísticamente significativas y mejor posicionadas al abordar el problema del TDP, sobre los escenarios de la tabla 3.17.

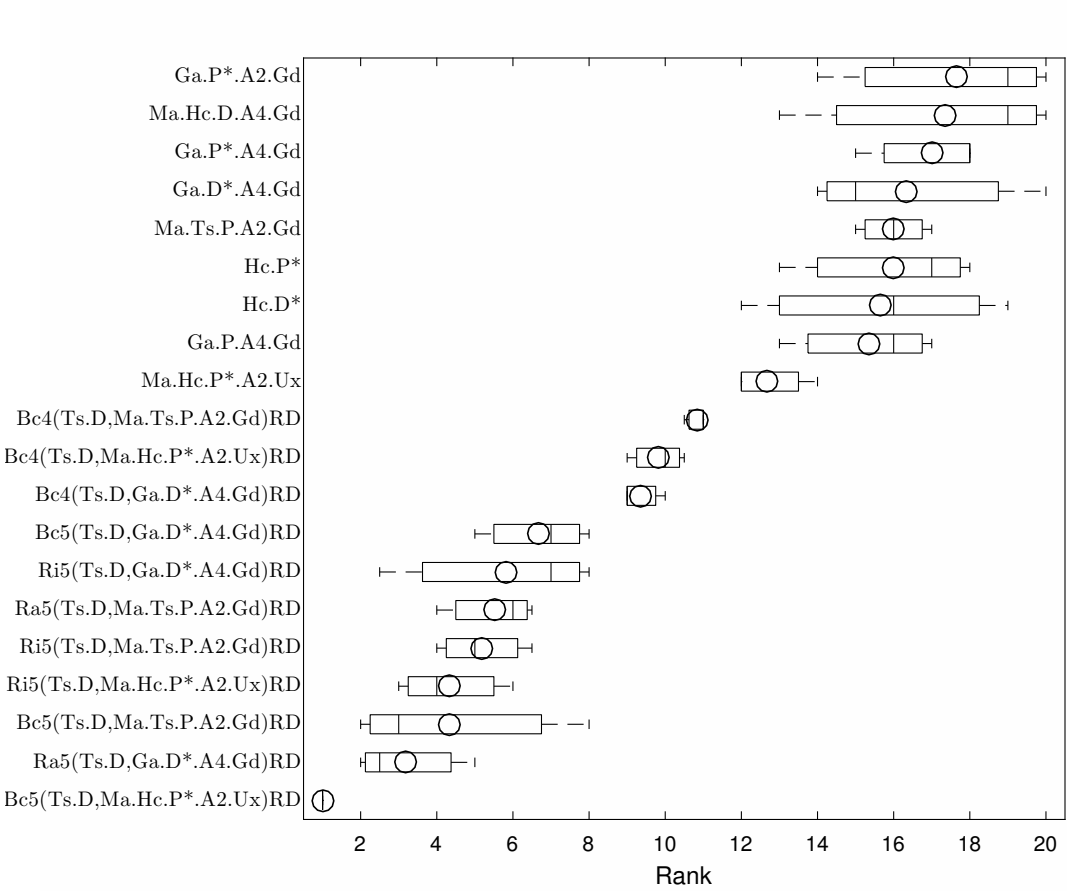
	Friedman value	Critical χ^2 value	Iman-Davenport value	Critical F_F value
All 20	51.557143	30.143527	18.944882	1.867332

ésta sección. Como podemos observar los valores criticos son menores a los valores obtenidos en los tests respectivos, – véase la tabla 4.51 –. Para afianzar de mejor manera los argumentos aquí expuestos hemos optado por realizar el test de Holm usando *Bc5(Ts.D,Ma.Hc.P*.A2.Ux)RD* (que corresponde al mejor valor de la clasificación) como algoritmo de control.

Los resultados obtenidos para este test los encontramos en la tabla 4.52. Los valores nos indican que hay diferencias estadísticas significativas para el 11.11% de los algoritmos integrativos utilizados, si los comparamos con los algoritmos colaborativos, el resto de estos (88.89%) no muestra diferencias estadísticamente significativas.

Considerando el test de suma de rangos (véase la tabla 4.53), existen diferencias estadísticamente

Figura 4.25: Torneo de las 9 técnicas integrativas tomadas de la tabla 4.50 y 11 colaborativas tomadas de la tabla 4.37, catalogadas como no-dominadas, que no muestran diferencias estadísticamente significativas y mejor posicionadas al abordar el problema del TDP, sobre los escenarios de la tabla 3.17. Incluye los modelos primal/dual, con/sin ruptura de simetrías: P Primal sin ruptura de simetrías, P* Primal con ruptura de simetrías; D Dual sin ruptura de simetrías, y D* Dual con ruptura de simetrías. Cada caja comprende desde el primer al tercer cuartil de la distribución, la mediana (2do cuartil) es indicada con una línea vertical, la zona sin valores atípicos (whiskers span) es de 1.5 veces el rango intercuartil, además los valores atípicos están marcados con un signo '+'.



significativas, respecto a la mejor técnica integrativa (digamos: *Ma.Hc.P*.A2.Ux*) utilizada en esta experimentación. Por lo demás, podemos afirmar que seis de las técnicas colaborativas no parecen mostrar diferencias, en al menos, 6 de ellas, respecto al mejor esquema colaborativo utilizado.

4.6 Revisión de Resultados: BIBD y TDP para las Técnicas Híbridas

En este capítulo se muestra una extensa experimentación aplicando las diferentes técnicas metaheurísticas híbridas que hemos seleccionado, al abordar los problemas de diseño de bloques incompletos balanceados y el de diseño plantillas. Los diferentes análisis realizados nos muestran que las técnicas

Tabla 4.52: Resultados del test de Holm ($\alpha = 0.05$), de las 9 técnicas integrativas tomadas de la tabla 4.50 y 11 colaborativas tomadas de la tabla 4.37, catalogadas como no-dominadas, que no muestran diferencias estadísticamente significativas y mejor posicionadas al abordar el problema del TDP, sobre los escenarios de la tabla 3.17, usando Bc5(Ts.D,Ma.Hc.P*.A2.Ux)RD como algoritmo de control. Incluye los modelos primal/dual, con/sin ruptura de simetrías: P Primal sin ruptura de simetrías, P* Primal con ruptura de simetrías; D Dual sin ruptura de simetrías, y D* Dual con ruptura de simetrías.

i	algorithm	z-statistic	p-value	α/i
1	Ra5(Ts.D,Ga.D*.A4.Gd)RD	4.485e-01	3.269e-01	5.000e-02
2	Bc5(Ts.D,Ma.Ts.P.A2.Gd)RD	6.901e-01	2.451e-01	2.500e-02
3	Ri5(Ts.D,Ma.Hc.P*.A2.Ux)RD	6.901e-01	2.451e-01	1.667e-02
4	Ri5(Ts.D,Ma.Ts.P.A2.Gd)RD	8.626e-01	1.942e-01	1.250e-02
5	Ra5(Ts.D,Ma.Ts.P.A2.Gd)RD	9.316e-01	1.758e-01	1.000e-02
6	Ri5(Ts.D,Ga.D*.A4.Gd)RD	1.001e+00	1.585e-01	8.333e-03
7	Bc5(Ts.D,Ga.D*.A4.Gd)RD	1.173e+00	1.204e-01	7.143e-03
8	Bc4(Ts.D,Ga.D*.A4.Gd)RD	1.725e+00	4.225e-02	6.250e-03
9	Bc4(Ts.D,Ma.Hc.P*.A2.Ux)RD	1.829e+00	3.372e-02	5.556e-03
10	Bc4(Ts.D,Ma.Ts.P.A2.Gd)RD	2.036e+00	2.089e-02	5.000e-03
11	Ma.Hc.P*.A2.Ux	2.415e+00	7.863e-03	4.545e-03
12	Ga.P.A4.Gd	2.967e+00	1.502e-03	4.167e-03
13	Hc.D*	3.036e+00	1.198e-03	3.846e-03
14	Ma.Ts.P.A2.Gd	3.105e+00	9.504e-04	3.571e-03
15	Hc.P*	3.105e+00	9.504e-04	3.333e-03
16	Ga.D*.A4.Gd	3.174e+00	7.510e-04	3.125e-03
17	Ga.P*.A4.Gd	3.312e+00	4.626e-04	2.941e-03
18	Ma.Hc.D.A4.Gd	3.381e+00	3.607e-04	2.778e-03
19	Ga.P*.A2.Gd	3.450e+00	2.800e-04	2.632e-03

Tabla 4.53: Comparación de la superioridad estadística de Bc5(Ts.D,Ma.Hc.P*.A2.Ux)RD con los algoritmos restantes en cada una de los tres escenarios del TDP tomados de la tabla 3.17. Cada entrada en la tabla contiene tres símbolos que corresponden (de izquierda a derecha) a Cat Food Cartons, Herbs Cartons and Magazine Inserts: • (resp. ○) indica la diferencia de rendimiento sobre el escenario correspondiente (resp. no satisface) es estadísticamente significativas en $\alpha = 0.05$ usando el test de ranksum. Incluye los modelos primal/dual, con/sin ruptura de simetrías: P Primal sin ruptura de simetrías, P* Primal con ruptura de simetrías; D Dual sin ruptura de simetrías, y D* Dual con ruptura de simetrías.

Ma.Hc.P*.A2.Ux	Ra5(Ts.D,Ga.D*.A4.Gd)RD	Bc5(Ts.D,Ma.Ts.P.A2.Gd)RD	Ri5(Ts.D,Ma.Hc.P*.A2.Ux)RD
•/•/•	•/○/○	•/○/○	•/•/○
Ri5(Ts.D,Ma.Ts.P.A2.Gd)RD	**Ra5(Ts.D,Ma.Ts.P.A2.Gd)RD**	**Ri5(Ts.D,Ga.D*.A4.Gd)RD**	**Bc5(Ts.D,Ga.D*.A4.Gd)RD**
•/○/○	•/○/○	•/•/○	•/•/•
Bc4(Ts.D,Ga.D*.A4.Gd)RD	**Bc4(Ts.D,Ma.Hc.P*.A2.Ux)RD**	**Bc4(Ts.D,Ma.Ts.P.A2.Gd)RD**	
•/•/•	•/○/○	•/○/○	

Tabla 4.54: Mejores configuraciones encontradas para el problema del TDP, de los 3 escenarios evaluados tomados de la tabla 3.17, para el diseño colaborativo $Bc5(Ts.D, Ma.Hc.P*.A2.Ux)RD$ (mínino desperdicio, utilizando el menor número de plantillas).

Problem	No. Template	Templates	Pressings	Overall Desv.	T.Desv	Waste
Cat Food	2	[0,0,0,0,0,2,7]	157143			
		[1,1,1,2,2,2,0]	250000	0.80	-3.85/ 1.79	29287
Herbs Cartoon	2	[1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,				
		1,1,1,0,1,1,1,1,2,3,3,3,2,4,4]	66000			
		[0,0,0,0,0,0,0,0,0,1,5,1,1,1,1,				
		1,1,1,6,2,2,2,2,1,2,2,2,6,1,1]	16000	2.97	-8.89/ 10.00	104000
Mz.Inserts	3	[0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,				
		1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,0]	150000			
		[1,1,1,0,1,1,2,2,2,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,				
		1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,1,0,0,0,1,0,0,0]	50000			
		[0,0,0,2,1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,				
		0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,1,3,3,3,2,2,2,12]	33000	2.63	-9.09/ 10.00	246000

empleadas operan de manera adecuada para resolver los problemas utilizados, ofreciendo resultados alentadores.

En general los esquemas colaborativos aplicados sobre el problema del TDP, resultan mucho más rápidos que las técnicas al operar en forma individual (al igual que ha sucedido con el problema del BIBD, anteriormente descrito). Esto lo podemos concluir ya que para los escenarios más complejos, las técnicas colaborativas han logrado superar a sus contrapartes integrativas, como se puede ver en la tabla 4.54. Podemos observar que para el escenario Herbs Cartoon, se logra superar el resultado alcanzado por las técnicas integrativas, ya que se desperdician 548 unidades menos de producto terminado. Para el escenario Magazine, se logra ahorrar un total de 31500 unidades de producto terminado, en comparación con la mejor técnica integrativa.

4.7 Contribuciones

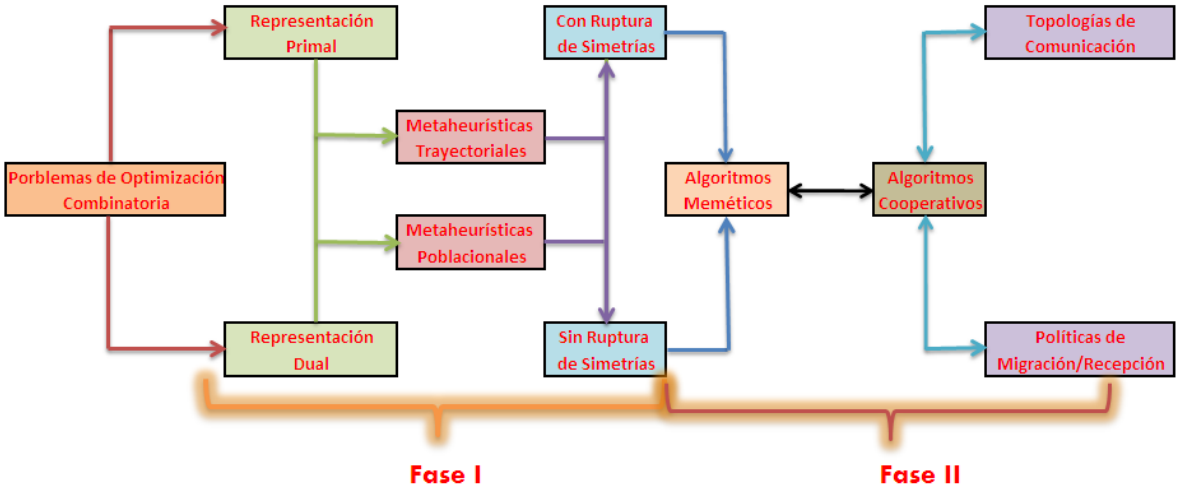
En este capítulo presentamos las diferentes propuestas que hemos utilizado para la realización de la presente investigación, en lo que se refiere a las técnicas híbridas. Para ello hemos descrito, en detalle, los algoritmos Meméticos y los esquemas de colaboración que se han empleado[2]. Se realiza un análisis de las bondades de nuestra propuesta por medio de la discusión del comportamiento de los métodos *Integrativos vs Cooperativos*, para los problemas objeto de estudio. Las contribuciones que consideramos importantes mencionar al concluir esta parte del trabajo son:

- Hemos mostrado la bondad de las metaheurísticas híbridas para hacer frente a problemas de optimización que tradicionalmente se han abordado eficientemente mediante métodos LP y CPS.
- Abordamos tres temas principales para el diseño de técnicas híbridas: simetrías del problema, formulación del problema (y aspectos reaccionados como la representación del espacio de búsqueda) y diferentes formas de colaboración (es decir, esquemas integradores versus cooperativos); ya que comúnmente las técnicas colaborativas operan sobre la misma representación del problema tratado.
- Hemos considerado el uso de diferentes políticas de migración/recepción y reemplazo, que permite a los diferentes algoritmos el intercambio de información de manera más inteligente.

[2]Parte del código fuente puede ser revisado en: https://github.com/drrueda/UMA-Code

- Algunos de los métodos cooperativos que hemos propuesto en este trabajo, pueden constituir el state-of-the-art para resolver problemas de optimización. Además, proporcionamos un esquema general a partir del cual se pueden generar otras metaheurísticas (posiblemente cooperativas).
- El trabajo desarrollado nos lleva a proponer una metodología robusta para abordar problemas de optimización combinatoria en general, desde un enfoque metaheurístico. Proponemos dos fases para enfrentar este tipo de problemas, la primera de ellas implica un proceso de análisis y diseño de los elementos principales de nuestra propuesta; es decir debemos considerar las representaciones del problema, las metaheurísticas básicas (i.e. LSs y GAs) que se utilizarán en el modelo y las estrategias para la ruptura de las simetrías. En esta fase los algoritmos operaran de manera individual (o aislada) sin intercambiar información entre ellos. En la segunda fase, es necesario determinar, luego de su respectivo análisis de resultados, el comportamiento de cada una las técnicas básicas empleadas para poder agrupar estas, de acuerdo a su rendimiento y proponer las diferentes técnicas integrativas que serán utilizadas en esta fase (i.e. MAs y cooperativos). En esta fase se deben determinar aspectos como: (a) topologías de comunicación, (b) políticas de migración/recepción/reemplazo. (c) número de agentes a utilizar para el intercambio de información y (d) número de grupos de agentes. Un gráfico de nuestra propuesta lo podemos revisar en la figura 4.26.

Figura 4.26: Propuesta de metodología resultante del estudio realizado en esta tesis



Todo este trabajo nos ha permitido describir aportaciones publicadas en la literatura científica, y que además han sido incorporadas en este capítulo:

- David Rodríguez Rueda, Enrique Darghan, y Julio Monroy [272].
- David Rodríguez Rueda, Carlos Cotta y Antonio J. Fernández-Leiva [273, 274].

Adicionalmente, los siguientes trabajos están en proceso de revisión/preparación:

- David Rodríguez Rueda, Carlos Cotta y Antonio J. Fernández-Leiva [275].

CAPÍTULO 5

CONCLUSIONES Y TRABAJO FUTURO

En este capítulo se presenta un conjunto de conclusiones y las posibles líneas a seguir para enfrentar los principales problemas aquí estudiados, que en general pueden también ser extensibles a una gran gama de problemas combinatorios en general, que posean características similares en lo que se refiere a problemas de optimización/satisfacción combinatoria, cuya formulación ha sido utilizada especialmente para ser abordados por medio de técnicas de ILP o CP, que además poseen naturaleza simétrica y presentan un gran número de instancias/escenarios de diversa complejidad, que incluyen fácil, media y gran dificultad para su resolución. En esta investigación hemos abordado dos problemas académicos, que además son aplicables a la resolución problemas reales en la industria (para la optimización de desperdicios y de material terminado), como lo es el caso del Diseño de Plantillas (TDP), y a la criptografía, agricultura, biología, ingeniería, etc., como es el caso del problema de Diseño de Bloques Incompletos Balanceados (BIBD). En particular, hemos realizado una serie de estudios sobre diversas técnicas en la rama de los algoritmos genéticos, las búsquedas locales y los procedimientos híbridos que han sido utilizados en años recientes. Se ha abordado también, esquemas alternativos considerando diferentes formas de representación enmarcados en la teoría de la dualidad, e intentando emplear mecanismos que permitan la reducción del paisaje de búsqueda por medio de la ruptura de los estados simétricos del problema. Además, se han aplicado esquemas de colaboración, utilizando diferentes modelos de arquitectura, así como algoritmos híbridos evolutivos con diferentes métodos de búsqueda local, concentrándonos en la hibridación desde el punto de vista de la "simbiosis" de algoritmos basados en población y en técnicas de búsqueda trayectoriales, es decir un enfoque memético e incorporando enfoques de cooperación entre las metaheurísticas propuestas a través de la definición de topologías de comunicación entre los diferentes componentes que participan en el esquema colaborativo por medio de la incorporación de políticas de migración/recepción y reemplazo que contribuyan, de manera inteligente, al intercambio de soluciones candidatas entre los diferentes nodos de la red.

En general, se ha realizado un estudio sobre diversos métodos en forma independiente o formando parte de un algoritmo evolutivo híbrido, sumado al uso varias formas de representación del problema –representaciones alternativas, **Primal**, **Dual**–, el empleo de esquemas colaborativos, que hacen uso de una variedad de políticas para el intercambio de información entre los elementos involucrados en el proceso – **agentes** –, basados en la integración por medio de diversas topologías que operan en forma de anillo (**Ring**), aleatoria (**Random**) o difusión (**Broadcast**), así como la incorporación de mecanismo que permiten la ruptura de las simetrías existentes en el problema, con el fin verificar

la efectividad de las técnicas metaheurísticas para resolver problemas de optimización/satisfacción combinatoria, cuya resolución ha sido abordada tradicionalmente por medio de técnicas de ILP o CP.

5.1 Conclusiones

Se han abordado dos problemas que han resultado difíciles de resolver y que han sido resueltos tradicionalmente por medio del uso de técnicas de ILP, CP y CSP, con una variedad de instancias que van desde complejidad media a alta complejidad, que además poseen naturaleza simétrica. Para ello se ha realizado una extensa y exhaustiva experimentación, aunado a un amplio estudio estadístico de las diferentes técnicas empleadas, tanto al operar en modo stand-alone, asi como, en formato de hibridación y colaboración que permitan validar la adecuación y la eficacia de las técnicas metaheurísticas para abordar estos dos problemas. Hemos podido probar que la mayoría de los esquemas de hibridación responden a un esquema de integración que ha permitido extender la capacidad de los algoritmos básicos por medio de un mejor balance entre los procesos de intensificación/diversificación. Para este estudio en particular, se han empleado técnicas de hibridación basadas en: (a) trayectoria (i.e. Hill Climbing, Búsqueda Tabú) y (b) poblacionales (i.e. Algoritmos Genéticos) con el fin de aprovechar las bondades de cada una de ellas en términos de su capacidad de intensificación/diversificación; la combinación de estos procesos nos permiten la exploración de paisajes de búsqueda complementarios. Los algoritmos meméticos resultantes han mostrado un rendimiento superior a sus componentes cuando operan en forma individual.

La puesta a punto de los algoritmos es un elemento que debemos considerar con mucho cuidado y paciencia, cuando se desea utilizar las técnicas metaheurísticas híbridas, para abordar la ruptura de las simetrías en problemas de optimización/resolución combinatoria con restricciones, ya que los parámetros que se empleen influyen directamente sobre el rendimiento de los algoritmos híbridos resultantes. Más concretamente, hemos podido observar que la calibración del algoritmo memético resulta fundamental, ya que debimos realizar una amplia experimentación para poder conseguir los valores más adecuados de estos parámetros. Los detalles de estos experimentos los podemos encontrar en los capítulos referido a la experimentación (3 y 4).

A lo largo de la experimentación aplicada sobre los dos problemas que han sido abordados en este trabajo de investigación, hemos podido observar que las metaheurísticas integrativas mantienen un rendimiento aceptable, sin embargo, focalizamos muchos aspectos limitativos al momento de atacar instancias/escenarios de los problemas cuyo grado de complejidad aumenta las posibilidades de obtener óptimos globales. Por esta razón, hemos propuesto un esquema para la configuración de modelos colaborativos que combinan las ventajas de los algoritmos integradores utilizados en la realización de las pruebas ejecutadas. Además, algunas de nuestras propuestas cooperativas se pueden considerar, en la actualidad, como los métodos metaheurísticos de última generación para enfrentar un gran número de problemas de optimización combinatoria.

Otro aspecto, que hemos encontrado muy interesante, ha sido la incorporación de representaciones alternativas del problema. La idea considerada en nuestra propuesta está relacionada con la capacidad que posee la representación dual de permitir una exploración del espacio de búsqueda complementario del problema frente a la representación primal (tradicionalmente asociada a la LP y CSP). Además, es común que los algoritmos híbridos/colaborativos trabajen siempre sobre la misma representación del problema, esta tesis propone la incorporación de una representación alternativa, que permita a los esquemas de colaboración aumentar la posibilidad de exploración de las diferentes zonas del paisaje de búsqueda que se generan sobre el problema en particular que se está abordando. Muchas de las

técnicas que se emplean no tiene éxito debido a que sólo son capaces de explorar pequeñas zonas de la totalidad del espacio de búsqueda, aumentando la posibilidad de quedar atrapadas en mínimos locales (o máximos), hemos podido observar que el empleo de varias representaciones del problema ayuda enormemente en la exploración más efectiva (directamente proporcional al ahorro de tiempos de respuesta) de los paisajes de búsqueda del dominio del problema.

Las simetrías del problema imprimen mayor dificultad a las técnicas de búsqueda, ya sea al operar en forma individual o en forma mancomunada. El empleo de modelos donde se considere atender la ruptura de simetrías, así como la incorporación de un pool de restricciones, conducen a un mejor rendimiento. En este estudio hemos empleado mecanismos para la ruptura de las simetrías como: *Value symmetry breaking* y un ordenamiento lexicográfico , que suelen ser empleados en la programación con restricciones, pero no así en las técnicas metaheurísticas. Apoyándonos en el análisis estadístico aplicado sobre la basta experimentación realizada, hemos podido constatar que la aplicación de políticas para la ruptura de simetrías juega un papel relevante al momento de abordar los problemas que presentan espacios de búsqueda de tamaño considerable, es por ello que hemos propuesto utilizar mecanismos estándar para la ruptura de simetrías que se emplean en CP (Constraint Programming), ya que las técnicas metaheurísticas suelen emplear mecanismos diferentes. En esta tesis, hemos analizado cómo se comportan las metaheurísticas cuando emplean este mecanismo estándar sobre los problemas objeto de estudio.

Al incorporar mecanismos colaborativos que involucren varias técnicas de búsqueda y diferentes espacios de codificación/formulación del problema (es común el uso de una sola formulación), se hace necesario determinar qué mecanismo de intercambio de información entre los diferentes actores del proceso resultan más adecuados. Para este estudio hemos empleados un conjunto de configuraciones para el intercambio de la información necesaria y suficiente, que permita a los algoritmos –agentes– actuar de manera adecuada para aportar información relevante al proceso colaborativo, por ello los algoritmos dependen de su topología de interacción (i.e. **Ring**, **Random** y **Broadcast**), de los modelos utilizado para codificar los candidatos (representación primal y dual) y de las políticas de migración/recepción/reemplazo. En esta tesis hemos estudiado cómo se comportan las metaheurísticas cuando se emplean esquemas colaborativos sobre diferentes formulaciones del problema.

El intercambio de información por medio de las topologías aumenta la potencia en los procesos colaborativos, pero se hace necesario la utilización de diferentes políticas que coadyuven en forma acertada los criterios a utilizar para el envió – *migración* – de las posibles soluciones candidatas que serán utilizadas como punto de partida en los procesos de intensificación de los agentes. Además de contar con políticas de aceptación –*recepción* – robustas al momento de determinar cuáles de las soluciones aportadas por un determinado agente deberían ser incorporadas en el pool de soluciones candidatas. Para esta investigación hemos empleado políticas de migración/recepción como: aleatoria (**Random**), por diversificación (**Diverse**) y el peor individuo (**Worst**).

Los resultados obtenidos nos han permitido verificar que los esquemas colaborativos combinan diferentes técnicas integrativas que exploran diferentes zonas del espacio de búsqueda. Esto nos lleva a pensar en la gran utilidad que representa la combinación de las técnicas de búsqueda sobre paisajes complementarios del problema. Además, debemos subrayar que un gran número de algoritmos cooperativos tuvieron éxito en muchas instancias problemáticas donde las metaheurísticas más simples no tuvieron éxito.

El trabajo aquí descrito se centra en la solución de los problemas del BIBD y TDP, sin embargo, queremos subrayar que una gran parte de nuestra propuesta se puede generalizar para resolver cualquier problema combinatorio simétrico. Por esta razón, nuestros esquemas cooperativos son ge-

nerales y no dependen de algoritmos específicos, sino de factores de diseño que se han analizado a lo largo de nuestro trabajo de investigación. En este sentido, las metaheurísticas (especialmente las versiones cooperativas) descritas no deberían aplicarse directamente para manejar instancias de problemas abiertos en la teoría del diseño. Para hacer frente a instancias abiertas, nuestras propuestas deben ajustarse específicamente. Esto básicamente significa que debemos diseñar nuestras metaheurísticas para adaptarlas a las características particulares de la instancia abierta con el objetivo de explotar la estructura disponible tanto como sea posible. Esto requiere no solo tomar decisiones sobre los factores de diseño (por ejemplo, naturaleza de los agentes, topología del sistema cooperativo, políticas de migración/recepción, etc., solo por nombrar algunos), sino también incorporar conocimientos específicos sobre los escenarios/instancias del problema.

La integración de los diferentes aspectos mencionados anteriormente, nos ha permitido visualizar un metodología robusta para abordar problemas de optimización combinatoria en general, desde un enfoque metaheurístico. Queda demostrado, por la experimentación exhaustiva realizada, así como el análisis estadístico aplicado, la viabilidad de la aplicación de esta metodología para enfrentar este tipo de problemas.

5.2 Trabajo Futuro

Centrándose en los modelos meméticos integradores, en cuanto al trabajo futuro, planeamos profundizar en el estudio del equilibrio de los parámetros de intensificación/diversificación del algoritmo, con el objetivo de mejorar su rendimiento en las instancias/escenarios que han mostrado mayor dificultad para resolverlos. Un objetivo a medio plazo sería dotar a los MAs de capacidades auto-adaptativas [285] para mejorar sus capacidades de búsqueda. Este tema también se puede explorar en modelos cooperativos, muchos de los parámetros que definen podrían ser ajustados durante el tiempo de ejecución en respuesta al estado de la búsqueda.

Resulta interesante ampliar el análisis de los modelos híbridos propuestos, por medio de su aplicación sobre otros problemas de optimización combinatoria como es el caso del enrutamiento de vehículos con ventanas de tiempo [36, 316], empleando de igual forma representaciones alternativas del problema.

Otro aspecto interesante que hemos podido divisar está relacionado con el empleo de representaciones asimétricas, que han demostrado ser efectivas para resolver otros problemas combinatorios (véase [163, 318]) en la formulación de nuestro modelo dual. Esta es una línea de trabajo interesante que podríamos abordar en el futuro cercano.

Ha resultado interesante el empleo de dos regiones diferentes del espacio de búsqueda (representación primal y dual), pensamos ampliar este parámetro a más de dos representaciones alternativas, lo que nos podría conducir a nuevas regiones inexploradas, potenciado la posibilidad de lograr soluciones óptimas en el proceso colaborativo.

Se requiere de un estudio adicional para definir una formulación que permita adecuar los parámetros involucrados en los esquemas colaborativos como el número de agentes, el número de ciclos, número de soluciones candidatas en el pool, entre otros, y revisar el uso de nuevos parámetros como es el caso del establecimiento de un valor α que permita determinar cuando un agente resulta una distorción al no realizar nuevos aportes al modelo.

Incorporar el uso de modelos meta-colaborativos basados en landscape aplicados sobre los problemas que han sido abordados en este trabajo, así como la incorporación de mecanismo auto-adaptativos

de los parámetros del modelo meta-colaborativo, analizando el comportamiento de estos parámetros sobre las técnicas utilizadas al abordar estos problemas.

Consideración de esquemas colaborativos que puedan adaptarse dinámicamente al cambiar sus características particulares como el empleo de una topología u otra, cuándo debe utilizarse una determinada política de migración/recepción, etc., de acuerdo al desempeño detectado en cuanto al aporte de estas al modelo colaborativo.

Una línea de trabajo futura podría incorporar el uso de un resolutor ILP (i.e. LPSolve) para la inicialización del pool de candidatos de cada uno de los agentes que operan sobre el problema del BIBD. Este tipo de colaboración podría arrojar resultados interesantes.

Bibliografía

[1] D. Abramson, M. Krishnamoorthy, and H. Dang. Simulated annealing cooling schedules for the school timetabling problem. *Asia-Pacific Journal of Operational Research*, 16(1):1–22, 1999.

[2] Z. Adak and A. Demiriz. Hybridization of population-based ant colony optimization via data mining. *Intelligent Data Analysis*, 24:291–307, 03 2020.

[3] M. Affenzeller, S. Wagner, S. Winkler, and A. Beham. *Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications*. Numerical Insights. CRC Press, 2009.

[4] K. Agarwal, A. Sinha, and M. Hima Bindu. A novel hybrid approach to n-queen problem. In D. C. Wyld, J. Zizka, and D. Nagamalai, editors, *Advances in Computer Science, Engineering and Applications*, volume 166 of *Advances in Intelligent and Soft Computing*, pages 519–527. Springer Berlin Heidelberg, 2012.

[5] E. Alba. *Parallel metaheuristics: a new class of algorithms*, volume 47. John Wiley & Sons, 2005.

[6] E. Alba, G. Leguizamón, and G. Ordoñez. Evolutionary algorithms for the minimum tardy task problem. In *Proceedings of the International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications*, pages 401–413, 2003.

[7] E. Alba, G. Leguizamón, and G. Ordoñez. Two models of parallel aco algorithms for the minimum tardy task problem. *International Journal of High Performance Systems Architecture*, 1(1):50–59, 2007.

[8] D. J. Aloise, D. Aloise, C. T. M. Rocha, C. C. Ribeiro, J. C. R. Filho, and L. S. S. Moura. Scheduling workover rigs for onshore oil production. *Discrete Applied Mathematics*, 154(5): 695 – 702, 2006. IV ALIO/EURO Workshop on Applied Combinatorial Optimization.

[9] J. E. Amaya. *Modelos meméticos cooperativos para la optimización de problemas combinatorios*. PhD thesis, Universidad de Malaga, España, 2011.

[10] J. E. Amaya, C. Cotta, and A. J. Fernández-Leiva. A memetic cooperative optimization schema and its application to the tool switching problem. In R. Schaefer et al., editors, *Parallel Problem Solving from Nature - PPSN XI*, volume 6238 of *Lecture Notes in Computer Science (LNCS)*, pages 445–454. Springer Berlin / Heidelberg, 2011.

[11] J. E. Amaya, C. Cotta, and A. J. Fernández-Leiva. Memetic cooperative models for the tool switching problem. *Memetic Computing*, 3(3):199–216, Oct 2011.

[12] J. E. Amaya, C. Cotta, A. J. Fernández-Leiva, and P. García-Sánchez. Deep memetic models for combinatorial optimization problems: application to the tool switching problem. *Memetic Computing*, 12(1):3–22, 2020. doi: 10.1007/s12293-019-00294-1.

[13] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[14] W. R. Ashby, C. Shannon, and J. McCarthy. *Automata Studies: Annals of Mathematics Studies. Number 34*. Annals of mathematics studies. Princeton University Press, 1956.

[15] J. B. Atkinson. A greedy randomised search heuristic for time-constrained vehicle scheduling and the incorporation of a learning strategy. *Journal of the Operational Research Society*, pages 700–708, 1998.

[16] J. P. Aubin and I. Ekeland. *Applied Nonlinear Analysis*. Dover Books on Mathematics Series. Dover Publications, 2006.

[17] V. Bachelet and E.-G. Talbi. Cosearch: a co-evolutionary metaheuristic. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, volume 2, pages 1550–1557. IEEE, 2000.

[18] T. Bäck, D. B. Fogel, and Z. Michalewicz. *Evolutionary Computation 1: Basic Algorithms and Operators*. Taylor & Francis, 2000.

[19] R. Backofen and S. Will. Excluding symmetries in constraint-based search. *Constraints*, 7 (3-4):333–349, 2002.

[20] F. Barber and M. A. Salido. Introducción a la programación de restricciones. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 7(20):13–30, 2003.

[21] R. Battiti and G. Tecchiolli. The reactive tabu search. *ORSA journal on computing*, 6(2): 126–140, 1994.

[22] E. A. Bender and H. S. Wilf. A theoretical analysis of backtracking in the graph coloring problem. *Journal of Algorithms*, 6(2):275–282, 1985.

[23] B. Benhamou. Study of symmetry in constraint satisfaction problems. In *Proceedings of the 2nd Workshop on Principles and Practice of Constraint Programming, PPCP 94*, pages 246–254. DTIC Document, 1994.

[24] B. Benhamou and M. Saïdi. Local symmetry breaking during search in csps. In C. Bessire, editor, *Principles and Practice of Constraint Programming CP 2007*, volume 4741 of *Lecture Notes in Computer Science (LNCS)*, pages 195–209. Springer Berlin Heidelberg, 2007.

[25] M. Berkelaar, K. Eikland, and P. Notebaert. `lp_solve`, open source (mixed-integer) linear programming system. http://lpsolve.sourceforge.net/5.0/, 2004.

[26] H.-G. Beyer and H.-P. Schwefel. Evolution strategies: A comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[27] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, Sept. 2003.

[28] C. Blum, C. Cotta, A. J. Fernández-Leiva, J. E. Gallardo, and M. Mastrolilli. Hybridizations of metaheuristics with branch & bound derivates. In *Hybrid Metaheuristics*, pages 85–116. Springer, 2008.

[29] C. Blum, J. Puchinger, G. R. Raidl, and A. Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151, 2011.

[30] M. Blum. A machine-independent theory of the complexity of recursive functions. *Journal of the ACM (JACM)*, 14(2):322–336, 1967.

[31] P. Bofill and C. Torras. Neural cost functions and search strategies for the generation of block designs: an experimental evaluation. *International Journal of Neural Systems*, 11(02):187–202, 2001.

[32] P. Bofill, R. Guimerà, and C. Torras. Comparison of simulated annealing and mean field annealing as applied to the generation of block designs. *Neural Networks*, 16(10):1421–1428, 2003.

[33] J. Borghoff, L. R. Knudsen, and K. Matusiewicz. Hill climbing algorithms and trivium. In *Selected Areas in Cryptography*, pages 57–73. Springer, 2011.

[34] S. Bouktif, H. Sahraoui, and G. Antoniol. Simulated annealing for improving software quality prediction. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1893–1900. ACM, 2006.

[35] K. Brading and E. Castellani. *Symmetries in Physics: Philosophical Reflections*. Cambridge University Press, 2003.

[36] K. Braekers, K. Ramaekers, and I. V. Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300 – 313, 2016.

[37] S. C. Brailsford, C. N. Potts, and B. M. Smith. Constraint satisfaction problems: Algorithms and applications. *European Journal of Operational Research*, 119(3):557–581, 1999.

[38] D. Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4): 251–256, 1979.

[39] M. Buratti. Some (17q, 17, 2) and (25q, 25, 3)BIBD constructions. *Designs, Codes and Cryptography*, 16(2):117–120, 1999.

[40] M. Campêlo, V. A. Campos, and R. C. Corrêa. On the asymmetric representatives formulation for the vertex coloring problem. *Discrete Applied Mathematics*, 156(7):1097 – 1111, 2008. GRACO 2005.

[41] M. J. C. Cánovas and V. Navarro. *Optimización matemática aplicada. Enunciados, ejercicios y aplicaciones del mundo real con MATLAB*. Editorial Club Universitario, 2011.

[42] E. Castellani. On the meaning of symmetry breaking, 2002. URL http://philsci-archive.pitt.edu/927/.

[43] A. F. Chalmers. Curie's principle. *British Journal for the Philosophy of Science*, pages 133–148, 1970.

[44] H. Chen, N. S. Flann, and D. W. Watson. Parallel genetic simulated annealing: a massively parallel simd algorithm. *Parallel and Distributed Systems, IEEE Transactions on*, 9(2):126–136, 1998.

[45] T.-C. Chen and Y.-W. Chang. Modern floorplanning based on b*-tree and fast simulated annealing. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25 (4):637–650, 2006.

[46] M. Chiarandini, T. Stützle, et al. An application of iterated local search to graph coloring problem. In *Proceedings of the Computational Symposium on Graph Coloring and its Generalizations*, pages 112–125, 2002.

[47] B. Chopard and M. Tomassini. *An Introduction to Metaheuristics for Optimization*. Natural Computing Series. Springer International Publishing, 2018.

[48] N. Christofides. The vehicle routing problem. *RAIRO-Operations Research-Recherche Opérationnelle*, 10(V1):55–70, 1976.

[49] N. Christofides. *Combinatorial optimization*. A Wiley-Interscience publication. John Wiley & Sons Canada, Limited, 1979.

[50] W. G. Cochran and G. M. Cox. *Experimental Design*. John Wiley, New York, 1957.

[51] C. Colbourn. *CRC Handbook of Combinatorial Designs*. Discrete Mathematics and Its Applications. CRC Press, 2010.

[52] F. E. Colomine Durán, C. Cotta, and A. J. Fernández-Leiva. A comparative study of multiobjective evolutionary algorithms to optimize the selection of investment portfolios with cardinality constraints. In *Applications of Evolutionary Computation*, pages 165–173. Springer, 2012.

[53] A. Colorni, M. Dorigo, V. Maniezzo, and M. Trubian. Ant system for job-shop scheduling. *Belgian Journal of Operations Research, Statistics and Computer Science*, 34(1):39–53, 1994.

[54] R. K. Congram, C. N. Potts, and S. L. van de Velde. An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing*, 14(1):52–67, 2002.

[55] D. T. Connolly. An improved annealing scheme for the qap. *European Journal of Operational Research*, 46(1):93–100, 1990.

[56] W. Cook, W. Cunningham, W. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 2011.

[57] D. G. Corneil and R. Mathon. Algorithmic techniques for the generation and analysis of strongly regular graphs and other combinatorial configurations. *Annals of Discrete Mathematics*, 2:1–32, 1978.

[58] C. Cotta. Scatter search with path relinking for phylogenetic inference. *European Journal of Operational Research*, 169(2):520 – 532, 2006.

[59] C. Cotta and A. Fernández. A hybrid GRASP - evolutionary algorithm approach to golomb ruler search. In X. Yao et al., editors, *Parallel Problem Solving From Nature VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 481–490, Berlin, 2004. Springer-Verlag.

[60] C. Cotta and J. Troya. A hybrid genetic algorithm for the 0-1 multiple knapsack problem. In G. Smith, N. Steele, and R. Albrecht, editors, *Artificial Neural Nets and Genetic Algorithms 3*, pages 251–255, Wien New York, 1998. Springer-Verlag.

[61] C. Cotta and J. M. Troya. A comparison of several evolutionary heuristics for the frequency assignment problem. In J. Mira and A. Prieto, editors, *Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence*, pages 474–481, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-45720-6.

[62] C. Cotta, E. Talbi, and E. Alba. Parallel hybrid metaheuristics. In E. Alba, editor, *Parallel Metaheuristics*, pages 347–370. Wiley-Interscience, Hoboken NJ, 2005.

[63] C. Cotta, I. Dotú, A. J. Fernández-Leiva, and P. V. Hentenryck. Local search-based hybrid algorithms for finding golomb rulers. *Constraints*, 12(3):263–291, 2007.

[64] C. Cotta, L. Mathieson, and P. Moscato. Memetic algorithms. In R. Martí, P. M. Pardalos, and M. G. C. Resende, editors, *Handbook of Heuristics*, pages 607–638. Springer, 2018.

[65] D. Cox and N. Reid. *The Theory of the Design of Experiments*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. CRC Press, 2000.

[66] T. G. Crainic and M. Gendreau. Cooperative parallel tabu search for capacitated network design. *Journal of Heuristics*, 8(6):601–627, 2002.

[67] T. G. Crainic and M. Toulouse. *Parallel Strategies for Meta-Heuristics*, pages 475–513. Springer US, Boston, MA, 2003.

[68] T. G. Crainic and M. Toulouse. Explicit and emergent cooperation schemes for search algorithms. In V. Maniezzo, R. Battiti, and J.-P. Watson, editors, *Learning and Intelligent Optimization – LION 2007 II*, pages 95–109, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[69] T. G. Crainic, M. Gendreau, P. Hansen, and N. Mladenović. Cooperative parallel variable neighborhood search for the p-median. *Journal of Heuristics*, 10(3):293–314, 2004.

[70] J. Crawford, M. Ginsberg, E. Luks, and A. Roy. Symmetry-breaking predicates for search problems. *In Knowledge Representation 96: Principles of Kenowledge. Representation and ReasoningMorgan Kaufman, San Francisco, California*, pages 148–159, 1996.

[71] K. D. Crawford. Solving the n-queens problem using genetic algorithms. In *Proceedings of the 1992 ACM/SIGAPP symposium on Applied computing: technological challenges of the 1990's*, pages 1039–1047. ACM, 1992.

[72] C. Cruz and D. Pelta. Soft computing and cooperative strategies for optimization. *Applied Soft Computing*, 9(1):30–38, 2009.

[73] C. Darwin and A. R. Wallace. *La teoría de la evolución de las especies*. Clásicos de la Ciencia y la Tecnología. Editorial Crítica, 2006.

[74] K. Deb. *Optimization for engineering design: Algorithms and Examples*. PHI Learning, 2012.

[75] E. Demeulemeester and W. Herroelen. A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management science*, 38(12):1803–1818, 1992.

[76] M. Den Besten, T. Stützle, and M. Dorigo. Design of iterated local search algorithms. In *Applications of Evolutionary Computing*, pages 441–451. Springer, 2001.

[77] J. Denes and A. Keedwell. *Latin Squares: New Developments in the Theory and Applications*. Annals of Discrete Mathematics. Elsevier Science, 1991.

[78] B. Dengiz and C. Alabas. Simulation optimization using tabu search. In *Proceedings of the 32nd conference on Winter simulation*, pages 805–810. Society for Computer Simulation International, 2000.

[79] C. Dhaenens and L. Jourdan. *Metaheuristics and Parallel Optimization*, pages 53–61. John Wiley & Sons, Inc., 2016.

[80] M. Dorigo and M. Birattari. *Ant Colony Optimization*, pages 36–39. Springer US, Boston, MA, 2010.

[81] K. A. Dowsland. Some experiments with simulated annealing techniques for packing problems. *European Journal of Operational Research*, 68(3):389–399, 1993.

[82] K. A. Dowsland and B. Adenso-Díaz. Diseño de heurísticas y fundamentos del recocido simulado. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 7(19):93–102, 2003.

[83] J. Dréo, A. Chatterjee, A. Pétrowski, P. Siarry, and E. Taillard. *Metaheuristics for Hard Optimization: Methods and Case Studies*. Springer Berlin Heidelberg, 2006.

[84] D. Du and K. Ko. *Theory of Computational Complexity*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 2014.

[85] D.-Z. Du and P. M. Pardalos. *Handbook of Combinatorial Optimization: Supplement Volume B*. Handbook of Combinatorial Optimization: Supplement. Springer, 2006.

[86] A. Duarte, M. Laguna, and R. Marti. *Metaheuristics for Business Analytics: A Decision Modeling Approach*. EURO Advanced Tutorials on Operational Research. Springer International Publishing, 2017.

[87] H. Dyckhoff. A typology of cutting and packing problems. *European Journal of Operational Research*, 44(2):145–159, 1990.

[88] M. El-Abd and M. Kamel. A taxonomy of cooperative search algorithms. In *Hybrid Metaheuristics*, pages 32–41. Springer, 2005.

[89] A. A. El Gamal, L. A. Hemachandra, I. Shperling, and V. K. W. Wei. Using simulated annealing to design good codes. *IEEE Transactions on Information Theory*, 33(1):116–123, 1987.

[90] P. Erdős and A. Gyárfás. A variant of the classical ramsey problem. *Combinatorica*, 17(4): 459–467, 1997.

[91] D. E. Evans and Y. Kawahigashi. *Quantum symmetries on operator algebras*, volume 147. Clarendon Press Oxford, 1998.

[92] T. Fahle, S. Schamberger, and M. Sellmann. Symmetry breaking. In *Principles and Practice of Constraint Programming — CP 2001*, pages 93–107, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[93] A. J. Fernández-Leiva and Á. Gutiérrez-Rodríguez. On distributed user-centric memetic algorithms. *Soft Comput.*, 23(12):4019–4039, 2019.

[94] R. A. Fisher. The arrangement of field experiments. *Journal of the Ministry of Agriculture Great Britain*, 33, 1926.

[95] R. A. Fisher. The arrangement of field experiments. In *Breakthroughs in Statistics*, pages 82–91. Springer, 1992.

[96] R. A. Fisher and F. Yates. Statistical tables for biological, agricultural and medical research. *Statistical tables for biological, agricultural and medical research.*, 1(Ed. 3.), 1949.

[97] R. A. Fisher and F. Yates. *Statistical Tables for Biological, Agricultural and Medical Research*. Oliver & Boy, 3 edition, 1949.

[98] M. Fleischer. Simulated annealing: past, present, and future. In *Proceedings of the 27th conference on Winter simulation*, pages 155–161. IEEE Computer Society, 1995.

[99] P. Flener, A. M. Frisch, B. Hnich, Z. Kzltan, I. Miguel, and T. Walsh. Matrix modelling. In *Proc. of the CP-01 Workshop on Modelling and Problem Formulation. International Conference on the Principles and Practice of Constraint Programming*, 2001. Available from: http://citeseerx.ist.psu.edu/viewdoc/summary?optdoi=10.1.1.21.5946.

[100] P. Flener, A. M. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, J. Pearson, and T. Walsh. Breaking row and column symmetries in matrix models. In *Principles and Practice of Constraint Programming-CP 2002*, pages 462–477. Springer, 2002.

[101] F. Focacci and M. Milano. Global cut framework for removing symmetries. In *Principles and Practice of Constraint Programming CP 2001*, pages 77–92. Springer, 2001.

[102] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: Formulationdiscussion and generalization. In *ICGA*, volume 93, pages 416–423. Citeseer, 1993.

[103] M. Fox and D. Long. Extending the exploitation of symmetries in planning. In *AIPS*, pages 83–91, 2002.

[104] E. C. Freuder. Eliminating interchangeable values in constraint satisfaction problems. In *AAAI*, volume 91, pages 227–233, 1991.

[105] J. E. Gallardo and C. Cotta. A GRASP-based memetic algorithm with path relinking for the far from most string problem. *Engineering Applications of Artificial Intelligence*, 41:183 – 194, 2015.

[106] S. García, D. Molina, M. Lozano, and F. Herrera. A study on the use of non-parametric tests for analyzing the evolutionary algorithm's behaviour: a case study on the cec'2005 special session on real parameter optimization. *J. Heuristics*, 15(6):617–644, 2009.

[107] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Mathematical Sciences Series. W. H. Freeman, 1979.

[108] S. I. Gass. *Linear Programming: Methods and Applications*. Dover Books on Computer Science Series. Dover Publications, 2003.

[109] M. Gendreau and J. Potvin, editors. *Handbook of Metaheuristics*, volume 272 of *International Series in Operations Research and Management Science*. Springer International Publishing AG, 3 edition, 2019.

[110] M. Gendreau and J.-Y. Potvin. Metaheuristics in combinatorial optimization. *Annals of Operations Research*, 140(1):189–213, Nov 2005.

[111] I. P. Gent and B. Smith. Symmetry breaking during search in constraint programming. In *Proceedings ECAI 2000*, pages 599–603, 1999.

[112] I. P. Gent, W. Harvey, and T. Kelsey. Groups and constraints: Symmetry breaking during search. In *Principles and Practice of Constraint Programming-CP 2002*, pages 415–430. Springer, 2002.

[113] I. P. Gent, I. McDonald, and B. M. Smith. Conditional symmetry in the all-interval series problem. In *Proc. SymCon*, volume 3, pages 55–65, 2003.

[114] I. P. Gent, T. Kelsey, S. A. Linton, I. McDonald, I. Miguel, and B. M. Smith. Conditional symmetry breaking. In P. Van Beek, editor, *Principles and Practice of Constraint Programming - CP 2005*, volume 3709 of *Lecture Notes in Computer Science (LNCS)*, pages 256–270. Springer Berlin Heidelberg, 2005.

[115] P. B. Gibbons and P. R. J. Östergård. *Computational Methods in Design Theory*, pages 755–790. CRC Press, 2010.

[116] J. W. Gibbs. *Elementary Principles in Statistical Mechanics*. Dover Books on Physics. Dover Publications, 2014.

[117] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem. *Operations research*, 9(6):849–859, 1961.

[118] P. A. Giraldo, S. I. Uribe, and A. López. Análisis de secuencias de adn mitocondrial (cytb y nd1) en lucilia eximia (diptera: Calliphoridae). *Revista Colombiana de Entomología*, 37(2): 273–278, 2011.

[119] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549, 1986.

[120] F. Glover. Tabu search part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[121] F. Glover. Tabu search part II. *ORSA Journal on Computing*, 2(1):4–32, 1990.

[122] F. Glover. Tabu search for nonlinear and parametric optimization (with links to genetic algorithms). *Discrete Applied Mathematics*, 49(1-3):231–255, 1994.

[123] F. Glover. A template for scatter search and path relinking. In *Artificial evolution*, pages 1–51. Springer, 1997.

[124] F. Glover and G. A. Kochenberger. *Handbook of Metaheuristics*. International series in operations research & management science. Kluwer Academic Publishers, 2003.

[125] F. Glover and M. Laguna. *Tabu Search*, pages 3261–3362. Springer, New York, NY, 2013.

[126] F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. *Control and cybernetics*, 29(3):653–684, 2000.

[127] F. Glover, M. Laguna, and R. Marti. Principles of tabu search. *Approximation Algorithms and Metaheuristics*, 23:1–12, 2007.

[128] F. W. Glover and M. Laguna. *Tabu Search*. Number v. 1 in Tabu Search. Springer US, 1998.

[129] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. *Foundations of genetic algorithms*, 1:69–93, 1991.

[130] D. E. Goldberg and J. H. Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99, 1988.

[131] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[132] M. Gomez-Barrero, J. Galbally, J. Fierrez, and J. Ortega-Garcia. Hill-climbing attack based on the uphill simplex algorithm and its application to signature verification. In *Biometrics and ID Management*, pages 83–94. Springer, 2011.

[133] J. F. Gonçalves, J. J. de Magalhães Mendes, and M. G. C. Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, 167(1):77 – 95, 2005.

[134] A. González and J. López. *Curso de geometría afín y geometría euclidiana*. Sanz y Torres, 2011.

[135] T. González, editor. *Handbook of Approximation Algorithms and Metaheuristics*. Chapman & Hall/Crc Computer & Information Science Series, 2007.

[136] G. Gopalakrishnan. Turing machines. In *Computation Engineering*, pages 271–290. Springer US, 2006.

[137] I. Griva, S. Nash, and A. Sofer. *Linear and Nonlinear Optimization: Second Edition*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2009.

[138] J. Gu and X. Huang. Efficient local search with search space smoothing: a case study of the traveling salesman problem (tsp). *Systems, Man and Cybernetics, IEEE Transactions on*, 24 (5):728–735, May 1994.

[139] R. W. Haessler and P. E. Sweeney. Cutting stock problems and solution procedures. *European Journal of Operational Research*, 54(2):141–150, 1991.

[140] P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *European journal of operational research*, 130(3):449–467, 2001.

[141] P. Hansen, N. Mladenović, and J. A. Moreno-Pérez. Búsqueda de entorno variable. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 7(19):77–92, 2003.

[142] X. Hao. Optimization models and heuristic method based on simulated annealing strategy for traveling salesman problem. *Applied Mechanics and Materials*, 34-35:1180–1184, 2010.

[143] W. E. Hart. *Adaptive Global Optimization with Local Search*. PhD thesis, University of California, 1994.

[144] W. E. Hart, N. Krasnogor, and J. E. Smith. *Recent Advances in Memetic Algorithms*, volume 166 of *Studies in Fuzziness and Soft Computing*. Springer-Verlag, Berlin Heidelberg, 2005.

[145] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, pages 285–306, 1965.

[146] A. Haselböck. Exploiting interchangeabilities in constraint satisfaction problems. In *IJCAI*, volume 93, pages 282–289, 1993.

[147] R. Haupt and S. Haupt. *Practical Genetic Algorithms*. Wiley InterScience electronic collection. Wiley, 2004.

[148] A. Hertz and D. de Werra. The tabu search metaheuristic: how we used it. *Annals of Mathematics and Artificial Intelligence*, 1(1):111–121, 1990.

[149] Z. C. S. S. Hlaing and M. A. Khine. An ant colony optimization algorithm for solving traveling salesman problem. In *International Conference on Information Communication and Management, Singapore, IPCSIT*, volume 16, 2011.

[150] K. Hoffman, M. Padberg, and G. Rinaldi. Traveling salesman problem. In S. Gass and M. Fu, editors, *Encyclopedia of Operations Research and Management Science*, pages 1573–1578. Springer US, 2013.

[151] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Bradford book. MIT Press, 1992.

[152] C. Hoyweghen, B. Naudts, and D. Goldberg. Spin-flip symmetry and synchronization. *Evolutionary computation*, 10:317–44, 02 2002.

[153] J. Hu, J. Xu, and L. Xie. Cooperative search and exploration in robotic networks. *Unmanned Systems*, 1:121–142, 07 2013.

[154] T. C. Hu and A. B. Kahng. *Linear and Integer Programming Made Easy*. Springer International Publishing, 1 edition, 2016.

[155] X. Hu and P. Li. A fast and practical algorithm for absolute dominators searching for very large scale integration. In W. Du, editor, *Informatics and Management Science IV*, volume 207 of *Lecture Notes in Electrical Engineering*, pages 195–203. Springer London, 2013.

[156] X. Huang. *Cooperative optimization for solving large scale combinatorial problems*, pages 117 – 156. Series on Computers and Operations Research, Theory and Algorithms for Cooperative Systems, 08 2004.

[157] S. Imahori, M. Yagiura, and T. Ibaraki. Local search heuristics for the rectangle packing. In *4th Metaheuristics International Conference, MIC'2001*, pages 471–476, 2001.

[158] L. Ingber. Simulated annealing: Practice versus theory. *Mathematical and Computer Modelling*, 18(11):29 – 57, 1993.

[159] L. Ingber. Adaptive simulated annealing (ASA): Lessons learned. *Control and cybernetics*, 25: 33–54, 1996.

[160] F. Jaeger. *Lectures on the Principle of Symmetry and Its Applications in All Natural Sciences*. Alpha Ed, 2019.

[161] S. Jakobs. On genetic algorithms for the packing of polygons. *European Journal of Operational Research*, 88(1):165–181, 1996.

[162] R. Jans and J. Desrosiers. Binary clustering problems: Symmetric, asymmetric and decomposition formulations. *GERAD Technical Report G-2010-44*, pages 1 – 15, 2010.

[163] R. Jans and J. Desrosiers. Efficient symmetry breaking formulations for the job grouping problem. *Computers & Operations Research*, 40(4):1132 – 1142, 2013.

[164] P. W. M. John. *Statistical Design and Analysis of Experiments*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 1998.

[165] L. Jourdan, M. Basseur, and E.-G. Talbi. Hybridizing exact methods and metaheuristics: A taxonomy. *European Journal of Operational Research*, 199(3):620–629, 2009.

[166] J. Kennedy. *Particle Swarm Optimization*, pages 760–766. Springer US, Boston, MA, 2010.

[167] G. W. Kinney, J. W. Barnes, and B. W. Colletti. A reactive tabu search algorithm with variable clustering for the unicost set covering problem. *International Journal of Operational Research*, 2(2):156–172, 2007.

[168] S. Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of statistical physics*, 34(5-6):975–986, 1984.

[169] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simmulated annealing. *science*, 220(4598):671–680, 1983.

[170] G. A. Kochenberger, J.-K. Hao, Z. Lü, H. Wang, and F. Glover. Solving large scale max cut problems via tabu search. *Journal of Heuristics*, 19(4):565–571, Aug 2013.

[171] O. Kramer. *Iterated Local Search: A Brief Introduction to Continuous Evolutionary Optimization*, pages 45–54. Springer International Publishing, Cham, 2014.

[172] N. Krasnogor. Memetic algorithms. In *In: Metaheuristics in Neural Networks Learning*, pages 225–247. Springer, 2006.

[173] N. Krasnogor and S. Gustafson. A study on the use of self-generation in memetic algorithms. *Natural Computing*, 3(1):53–76, 2004.

[174] N. Krasnogor and S. Gustafson. Self-assembling of local searchers in memetic algorithms. In W. E. Hart, J. E. Smith, and N. Krasnogor, editors, *Recent Advances in Memetic Algorithms*, volume 166 of *Studies in Fuzziness and Soft Computing*, pages 229–257. Springer Berlin Heidelberg, 2005.

[175] M. Laguna, J. W. Barnes, and F. W. Glover. Tabu search methods for a single machine scheduling problem. *Journal of Intelligent Manufacturing*, 2(2):63–73, 1991.

[176] A. Y. S. Lam and V. O. K. Li. Generalization of the no-free-lunch theorem. In *2009 IEEE International Conference on Systems, Man and Cybernetics*, pages 4322–4328, 2009.

[177] L. Lan, Y. Y. Tai, S. Lin, B. Memari, and B. Honary. New constructions of quasi-cyclic LDPC codes based on special classes of BIDBs for the AWGN and binary erasure channels. *IEEE Transactions on Communications*, 56(1):39–48, 2008.

[178] B. Lawal. Incomplete block design. In *Applied Statistical Methods in Agriculture, Health and Life Sciences*, pages 639–659. Springer International Publishing, 2014.

[179] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys. *The traveling salesman problem: a guided tour of combinatorial optimization*, volume 3. Wiley New York, 1985.

[180] J. Lee. *A First Course in Combinatorial Optimization*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2004.

[181] A. Lodi, S. Martello, and D. Vigo. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS Journal on Computing*, 11(4):345–357, 1999.

[182] A. Lodi, S. Martello, and M. Monaci. Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141(2):241–252, 2002.

[183] H. R. Lourenço. Job-shop scheduling: Computational study of local search and large-step optimization methods. *European Journal of Operational Research*, 83(2):347 – 364, 1995. EURO Summer Institute Combinatorial Optimization.

[184] H. R. Lourenço, O. C. Martin, and T. Stützle. *Iterated Local Search*, pages 320–353. Springer US, Boston, MA, 2003.

[185] G. Luque. *Resolución de Problemas Combinatorios con Aplicación Real en Sistemas Distribuidos*. PhD thesis, Universidad de Málaga, 2006.

[186] K. Mainzer. *Symmetries of Nature: A Handbook for Philosophy of Nature and Science*. Walter de Gruyter, 1996.

[187] K.-F. Man, K.-S. Tang, and S. Kwong. Genetic algorithms: concepts and applications. *IEEE transactions on Industrial Electronics*, 43(5):519–534, 1996.

[188] B. N. Mandal. Linear integer programming approach to construction of balanced incomplete block designs. *Communications in Statistics - Simulation and Computation*, 44(6):1405–1411, 2015.

[189] J. Mandziuk. Solving the n-queens problem with a binary hopfield-type network. synchronous and asynchronous model. *Biological Cybernetics*, 72(5):439–446, 1995.

[190] V. Maniezzo and A. Carbonaro. An ANTS heuristic for the frequency assignment problem. *Future Generation Computer Systems*, 16(8):927 – 935, 2000.

[191] V. Maniezzo and A. Colorni. The ant system applied to the quadratic assignment problem. *Knowledge and Data Engineering, IEEE Transactions on*, 11(5):769–778, Sep 1999.

[192] Y. Marinakis. Multiple phase neighborhood search-GRASP for the capacitated vehicle routing problem. *Expert Systems with Applications*, 39(8):6807–6815, 2012.

[193] O. C. Martin and S. W. Otto. Partitioning of unstructured meshes for load balancing. *Concurrency: Practice and Experience*, 7(4):303–314, 1995.

[194] I. Martinjak and M. Golub. Comparison of heuristic algorithms for the n-queen problem. In *Information Technology Interfaces, 2007. ITI 2007. 29th International Conference on*, pages 759–764. IEEE, June 2007.

[195] A. Masegosa, F. Mascia, D. Pelta, and M. Brunato. Cooperative strategies and reactive search: A hybrid model proposal. In T. Stützle, editor, *Learning and Intelligent Optimization*, volume 5851 of *Lecture Notes in Computer Science (LNCS)*, pages 206–220. Springer Berlin / Heidelberg, 2009.

[196] A. D. Masegosa, D. Pelta, I. G. del Amo, and J. L. Verdegay. *On the Performance of Homogeneous and Heterogeneous Cooperative Search Strategies*, pages 287–300. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[197] I. McDonald. Unique symmetry breaking in csps using group theory. In *SymCon*, volume 1, pages 75–78, 2001.

[198] I. McDonald. Nusbds: An easy to use symmetry breaking system. In F. Rossi, editor, *Principles and Practice of Constraint Programming CP 2003*, volume 2833 of *Lecture Notes in Computer Science (LNCS)*, pages 985–985. Springer Berlin Heidelberg, 2003.

[199] I. McDonald and B. Smith. Partial symmetry breaking. In P. Van Hentenryck, editor, *Principles and Practice of Constraint Programming - CP 2002*, pages 431–445, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[200] B. D. McKay. Nauty users guide (version 2.4). *Computer Science Dept., Australian National University*, 2007.

[201] R. Mead. *Design of Experiments: Statistical Principles for Practical Applications*. Cambridge University Press, 1993.

[202] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on optimization*, 2(4):575–601, 1992.

[203] B. Melián, J. A. M. Pérez, and J. M. M. Vega. Metaheurísticas: una visión global. *Inteligencia Artificial*, 7(19):7–28, 2003.

[204] P. Merz. *Memetic algorithms for combinatorial optimization problems: fitness landscapes and effective search strategies*. PhD thesis, University of Siegen, Germany, 2000.

[205] P. Merz and B. Freisleben. Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. *IEEE Trans. Evolutionary Computation*, 4(4):337–352, 2000. doi: 10.1109/4235.887234.

[206] P. Meseguer and C. Torras. Exploiting symmetries within constraint satisfaction search. *Artificial Intelligence*, 129(129):133 – 163, 2001.

[207] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[208] Z. Michalewicz and D. Fogel. *How to Solve It: Modern Heuristics*. Springer Berlin Heidelberg, 2004.

[209] T. Michel, T. Krishnaiyan, and G. Fred. Multi-level cooperative search: a new paradigm for combinatorial optimization and an application to graph partitioning. In *Euro-Par99 Parallel Processing*, pages 533–542. Springer, 1999.

[210] M. Milano and A. Roli. Magma: a multiagent architecture for metaheuristics. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(2):925–941, April 2004.

[211] B. L. Miller and D. E. Goldberg. Genetic algorithms, selection schemes, and the varying effects of noise. *Evolutionary Computation*, 4(2):113–131, 1996.

[212] H. D. Mills, R. C. Linger, and A. R. Hevner. *Principles of Information Systems Analysis and Design*. Academic Press, 1986.

[213] T. Miyazaki. The complexity of mckays canonical labeling algorithm. In *Groups and Computation II*, volume 28, pages 239–256. Aer. Math. Soc.: Providence, RI, 1997.

[214] N. Mladenović, M. Labbé, and P. Hansen. Solving the p-center problem with tabu search and variable neighborhood search. *Networks*, 42(1):48–64, 2003.

[215] J. D. Monk. Turing machines. In *Mathematical Logic*, volume 37 of *Graduate Texts in Mathematics*, pages 14–25. Springer New York, 1976.

[216] R. Moorthy and C.-P. Teo. Berth management in container terminal: the template design problem. *OR Spectrum, Springer*, 28(44):495–518, Octuber 2006.

[217] P. Moscato. On evolution, search, optimization algorithms and martial arts: Towards memetic algorithms. Report 826, Caltech Concurrent Computation Program, California Institute of Technology, Pasadena, 1989.

[218] P. Moscato. Memetic algorithms: A short introduction. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 219–234. McGraw-Hill, Maidenhead, Berkshire, England, UK, 1999.

[219] P. Moscato and C. Cotta. A gentle introduction to memetic algorithms. In *Handbook of metaheuristics*, pages 105–144. Springer, 2003.

[220] P. Moscato and C. Cotta. An accelerated introduction to memetic algorithms. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 272 of *International Series in Operations Research & Management Science*, pages 275–309. Springer International Publishing AG, 2019.

[221] H. Muhlenbein. Evolution in time and space-the parallel genetic algorithm. In *Foundations of genetic algorithms*. Citeseer, 1991.

[222] H. Muhlenbein. How genetic algorithms really work: I. mutation and hillclimbing. parallel problem solving from nature 2. b. manderick, 1992.

[223] A. D. Muñoz. *Metaheurísticas*. Ciencias Experimentales y Tecnología. Editorial Dykinson, S.L., 2007.

[224] R. Murphey. Frequency assignment problem. In C. A. Floudas and P. M. Pardalos, editors, *Encyclopedia of Optimization*, pages 1097–1101. Springer US, 2009.

[225] B. Naderi and R. Ruiz. A scatter search algorithm for the distributed permutation flowshop scheduling problem. *European Journal of Operational Research*, 239(2):323 – 334, 2014.

[226] F. Neri and C. Cotta. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, 2:1–14, 2012.

[227] F. Neri, C. Cotta, and P. Moscato, editors. *Handbook of Memetic Algorithms*, volume 379 of *Studies in Computational Intelligence*. Springer-Verlag, Berlin Heidelberg, 2012.

[228] F. Neumann and C. Witt. *Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity*. Natural Computing Series. Springer Berlin Heidelberg, 2010.

[229] R. Nogueras and C. Cotta. An analysis of migration strategies in island-based multimemetic algorithms. In T. Bartz-Beielstein, J. Branke, B. Filipič, and J. Smith, editors, *Parallel Problem Solving from Nature – PPSN XIII*, volume 8672 of *Lecture Notes in Computer Science (LNCS)*, pages 731–740. Springer International Publishing, 2014.

[230] R. Nogueras and C. Cotta. A study on meme propagation in multimemetic algorithms. *International Journal of Applied Mathematics and Computer Science*, 25(3):499–512, 2015.

[231] R. Nogueras and C. Cotta. Studying self-balancing strategies in island-based multimemetic algorithms. *Journal of Computational and Applied Mathematics*, 293:180 – 191, 2016. Efficient Numerical Methods for Large-scale Scientific Computations.

[232] Y. S. Ong and A. Keane. Meta-lamarckian learning in memetic algorithms. *IEEE Trans Evol Comput*, 8(2):99–110, 2004.

[233] Y. S. Ong, M.-H. Lim, N. Zhu, and K.-W. Wong. Classification of adaptive memetic algorithms: a comparative study. *IEEE Trans Syst Man Cybern B*, 36(2):141–152, 2006.

[234] I. H. Osman and J. P. Kelly. *Meta-Heuristics: Theory and Applications*. Springer, 1996.

[235] M. Ozawa. Halting of quantum turing machines. In *Unconventional Models of Computation*, volume 2509 of *Lecture Notes in Computer Science (LNCS)*, pages 58–65. Springer Berlin Heidelberg, 2002.

[236] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Books on Computer Science. Dover Publications, 2013.

[237] L. Paquete and T. Stützle. An experimental investigation of iterated local search for coloring graphs. In *Applications of Evolutionary Computing*, pages 122–131. Springer, 2002.

[238] M. Pelikan and D. E. Goldberg. Genetic algorithms, clustering, and the breaking of symmetry. In *Parallel Problem Solving from Nature PPSN VI*, pages 385–394, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[239] M. Pelikan and D. E. Goldberg. Genetic algorithms, clustering, and the breaking of symmetry. In *Parallel Problem Solving from Nature PPSN VI*, pages 385–394. Springer, 2000.

[240] M. Pelikan, M. W. Hauschild, and F. G. Lobo. *Estimation of Distribution Algorithms*, pages 899–928. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[241] D. Pelta, A. Sancho-Royo, C. Cruz, and J. L. Verdegay. Using memory and fuzzy rules in a co-operative multi-thread strategy for optimization. *Information Sciences*, 176(13):1849–1868, 2006.

[242] J. A. M. Pérez, N. Mladenović, B. M. Batista, and I. J. G. del Amo. *Variable Neighbourhood Search*, pages 71–86. Springer US, Boston, MA, 2006.

[243] G. Pólya. *How to Solve It*. Princeton University, 1945.

[244] S. Prestwich. CSPLib problem 028: Balanced incomplete block designs. http://www.csplib.org/Problems/prob028, 1999.

[245] S. Prestwich. Balanced incomplete block design as satisfiability. In *Proceedings of the 12th Irish Conference on Artificial Intelligence and Cognitive Science*, 2001.

[246] S. Prestwich. A local search algorithm for balanced incomplete block designs. In F. Rossi, editor, *9th International Conference on Principles and Practices of Constraint Programming (CP2003)*, Lecture Notes in Computer Science (LNCS), pages 53–64. Springer, 2003.

[247] S. Prestwich. Negative effects of modeling techniques on search performance. *Annals of Operations Research*, 18:137–150, 2003.

[248] S. Prestwich and A. Roli. Symmetry breaking and local search spaces. In R. Barták and M. Milano, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 273–287, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[249] S. Prestwich, S. A. Tarim, and B. Hnich. Template design under demand uncertainty by integer linear local search. *International Journal of Production Research*, 44(22):4915–4928, November 2006.

[250] L. Proll and B. Smith. Integer linear programming and constraint programming approaches to a template design problem. *INFORMS Journal on Computing*, 10(3):265–275, 1998.

[251] L. Proll and B. Smith. ILP and constraint programming approaches to a template design problem. *INFORMS Journal of Computing*, 10(3):265–275, 1998.

[252] J. Puchinger and G. R. Raidl. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In J. Mira and J. Álvarez, editors, *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*, volume 3562 of *Lecture Notes in Computer Science (LNCS)*, pages 113–124. Springer, Berlin Heidelberg, 2005.

[253] J.-F. Puget. On the satisfiability of symmetrical constrained satisfaction problems. In J. Komorowski and Z. W. Ras, editors, *Methodologies for Intelligent Systems*, volume 689 of *Lecture Notes in Computer Science (LNCS)*, pages 350–361. Springer Berlin Heidelberg, 1993.

[254] J.-F. Puget. Symmetry breaking revisited. *Constraints*, 10(1):23–46, 2005.

[255] A. Rahimi-Vahed, T. G. Crainic, M. Gendreau, and W. Rei. A path relinking algorithm for a multi-depot periodic vehicle routing problem. *Journal of Heuristics*, 19(3):497–524, 2013.

[256] G. R. Raidl. A unified view on hybrid metaheuristics. In F. Almeida, M. J. Blesa Aguilera, C. Blum, J. M. Moreno Vega, M. Pérez Pérez, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics*, pages 1–12, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-46385-6.

[257] E. B. Rainer. Quadratic assignment problems. *European Journal of Operational Research*, 15 (3):283 – 289, 1984.

[258] R. R. Ranjan, V. Gupta, M. Saikia, and J. Bora. A vlsi based scheme for implementation of BIBD. In V. V. Das, J. Stephen, and Y. Chaba, editors, *Computer Networks and Information Technologies*, volume 142 of *Communications in Computer and Information Science*, pages 609–614. Springer Berlin Heidelberg, 2011.

[259] M. Ranjbar. A path-relinking metaheuristic for the resource levelling problem. *Journal of the Operational Research Society*, 64(7):1071–1078, 2013.

[260] S. Rao. *Engineering Optimization: Theory and Practice*. New Age International, 2000.

[261] S. Rao and S. Rao. *Engineering Optimization: Theory and Practice*. Wiley, 2009.

[262] G. Reinelt. *The Traveling Salesman, Computational Solutions for TSP Applications*, volume 840 of *Lecture Notes in Computer Science*. Springer, 1994. ISBN 3-540-58334-3. doi: 10.1007/3-540-48661-5.

[263] M. Resende, R. Marti, and P. Pardalos, editors. *Handbook of Heuristics*. Springer International Publishing, Cham, 2018.

[264] T. Riaz, Y. Wang, and K.-B. Li. Multiple sequence alignment using tabu search. In *Proceedings of the second conference on Asia-Pacific bioinformatics-Volume 29*, pages 223–232. Australian Computer Society, Inc., 2004.

[265] A. Rogers, A. Prugel-Bennett, and N. Jennings. Phase transitions and symmetry breaking in genetic algorithms with crossover. *Theor. Comput. Sci.*, 358:121–141, 07 2006.

[266] F. Rossi, P. Van Beek, and T. Walsh. *Handbook of Constraint Programming*. Foundations of Artificial Intelligence. Elsevier Science, 2006.

[267] D. R. Rueda, C. Cotta, and A. J. Fernández-Leiva. Una comparativa de metaheurísticas para el problema del diseño de bloques incompletos equilibrados. In *VI Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB'09)*, pages 191–197, 2009.

[268] D. R. Rueda, C. Cotta, and A. J. Fernández-Leiva. Finding balanced incomplete block designs with metaheuristics. In *Evolutionary Computation in Combinatorial Optimization, 9th European Conference, EvoCOP 2009, Tübingen, Germany, April 15-17, 2009. Proceedings*, volume 5482 of *Lecture Notes in Computer Science (LNCS)*, pages 156–167. Springer, 2009.

[269] D. R. Rueda, C. Cotta, and A. J. Fernández-Leiva. Un problema de diseño de plantillas: Un enfoque metaheurístico basado en búsqueda local. In V. C. et al., editor, *VII Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB2010)*, pages 743–750, Garceta, Valencia, 2010.

[270] D. R. Rueda, C. Cotta, and A. J. Fernández-Leiva. A memetic algorithm for designing balanced incomplete blocks. *International Journal of Combinatorial Optimization Problems and Informatics*, 2(1):14–22, 2011.

[271] D. R. Rueda, C. Cotta, and A. J. Fernández-Leiva. The template design problem: A perspective with metaheuristics. In K. K. Sabelfeld and I. Dimov, editors, *Eighth IMACS Seminar on Monte Carlo Methods*, pages 181–192, Borovets, Bulgaria, 2011, 2012. De Gryuter, Berlin/Boston.

[272] D. R. Rueda, E. Darghan, and J. Monroy. A multi-agent proposal in the resolution of instances of BIBD. *Revista Colombiana de Estadística*, 39(2):267–280, July 2016.

[273] D. R. Rueda, C. Cotta, and A. J. Fernández-Leiva. Memetic collaborative approaches for finding balanced incomplete block designs. *Computers & Operations Research*, 114:1–14, 2020.

[274] D. R. Rueda, C. Cotta, and A. J. Fernández-Leiva. Metaheuristics for the template design problem: Encoding, symmetry and hybridisation. *Journal of Intelligent Manufacturing*, pages 1–29, 2020. Aceptado para su publicación.

[275] D. R. Rueda, C. Cotta, and A. J. Fernández-Leiva. Data for the search of 2-designs: solutions and metaheuritics code. *Data in Brief (Computers & Operations Research)*, 2020. Por enviar.

[276] S. Sahu and B. Nayak. An adaptive genetic algorithm method for damage detection in structural elements. *Materials Today: Proceedings*, 2020.

[277] R. Santana, R. I. McKay, and J. A. Lozano. Symmetry in evolutionary and estimation of distribution algorithms. In *2013 IEEE Congress on Evolutionary Computation*, pages 2053–2060, 2013.

[278] H. Sayoud, K. Takahashi, and B. Vaillant. Designing communication network topologies using steady-state genetic algorithms. *IEEE Communications Letters*, 5(3):113–115, March 2001.

[279] U. Schneider. A tabu search tutorial based on a real-world scheduling problem. *Central European Journal of Operations Research*, 19(4):467–493, 2011.

[280] C. R. Scrich, V. A. Armentano, and M. Laguna. Tardiness minimization in a flexible job shop: a tabu search approach. *Journal of Intelligent Manufacturing*, 15(1):103–115, 2004.

[281] B. Selman, H. J. Levesque, D. G. Mitchell, et al. A new method for solving hard satisfiability problems. In *AAAI*, volume 92, pages 440–446, 1992.

[282] V. P. Shylo, O. V. Shylo, and V. Roschyn. Solving weighted max-cut problem by global equilibrium search. *Cybernetics and Systems Analysis*, 48(4):563–567, 2012.

[283] S. Sivanandam and S. Deepa. *Introduction to Genetic Algorithms*. Springer Berlin Heidelberg, 2007.

[284] B. M. Smith. Reducing symmetry in a combinatorial design problem. *Research report series-university of leeds school of computer studies LU SCS RR*, 1(1), 2001.

[285] J. E. Smith. Self-adaptative and coevolving memetic algorithms. In F. Neri, C. Cotta, and P. Moscato, editors, *Handbook of Memetic Algorithms*, volume 379 of *Studies in Computational Intelligence*, pages 167–188. Springer Berlin Heidelberg, 2012.

[286] K. Smyth, H. H. Hoos, and T. Stützle. Iterated robust tabu search for max-sat. In Y. Xiang and B. Chaib-draa, editors, *Advances in Artificial Intelligence*, pages 129–144. Springer, Berlin, Heidelberg, 2003. ISBN 978-3-540-44886-0.

[287] R. Sosic and J. Gu. Fast search algorithms for the n-queens problem. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(6):1572–1576, 1991.

[288] A. Steven Rudich. *Computational Complexity Theory*. American Mathematical Soc., 2004.

[289] D. R. Stinson. Introduction to balanced incomplete block designs. In *Combinatorial Designs*, pages 1–21. Springer New York, 2004.

[290] T. Stützle. *Local search algorithms for combinatorial problems: analysis, improvements, and new applications*, volume 220. Infix Sankt Augustin, Germany, 1999.

[291] T. Stützle. Iterated local search for the quadratic assignment problem. *European Journal of Operational Research*, 174(3):1519–1539, 2006.

[292] W. Sun and Y. X. Yuan. *Optimization Theory and Methods: Nonlinear Programming*. Springer Optimization and Its Applications. Springer, 2006.

[293] A. Sureka and P. R. Wurman. Using tabu best-response search to find pure strategy nash equilibria in normal form games. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1023–1029. ACM, 2005.

[294] G. Syswerda. Uniform crossover in genetic algorithms. In J. D. Schaffer, editor, *3rd International Conference on Genetic Algorithms*, pages 2–9, San Mateo, CA, 1989. Morgan Kaufmann.

[295] G. Syswerda. A study of reproduction in generational and steady state genetic algorithms. *Foundations of genetic algorithms*, 2:94–101, 1991.

[296] H. Tad and W. Colin P. Solving the really hard problems with cooperative search. In *Proceedings of the national conference on artificial intelligence*, pages 231–231. John Wiley & Sons ltd, 1993.

[297] H. Tad and B. A. Huberman. Better than the best: The power of cooperation. *In SFI 1992 Lectures in Complex Systems*, pages 163–184, 1992.

[298] E.-G. Talbi. A taxonomy of hybrid metaheuristics. *Journal of heuristics*, 8(5):541–564, 2002.

[299] E.-G. Talbi. *Metaheuristics: From Design to Implementation*. Wiley Series on Parallel and Distributed Computing. Wiley, 2009.

[300] E.-G. Talbi and B. Vincent. Cosearch: A parallel cooperative metaheuristic. *Journal of Mathematical Modelling and Algorithms*, 5(1):5–22, 2006.

[301] E.-G. Talbi, O. Roux, C. Fonlupt, and D. Robillard. Parallel ant colonies for the quadratic assignment problem. *Future Generation Computer Systems*, 17(4):441–449, 2001.

[302] J. Tang, J. Zhang, and Z. Pan. A scatter search algorithm for solving vehicle routing problem with loading cost. *Expert Systems with Applications*, 37(6):4073 – 4083, 2010.

[303] M. A. Tawhid and A. F. Ali. A hybrid grey wolf optimizer and genetic algorithm for minimizing potential energy function. *Memetic Computing*, 9(4):347–359, 2017.

[304] P. T. Thach, H. Konno, and D. Yokota. Dual approach to minimization on the set of pareto-optimal solutions. *Journal of Optimization Theory and Applications*, 88(3):689–707, 1996.

[305] L. Tian and C. Collins. An effective robot trajectory planning method using a genetic algorithm. *Mechatronics*, 14(5):455–470, 2004.

[306] M. Toulouse, T. G. Crainic, B. Sanso, and K. Thulasiraman. Self-organization in cooperative tabu search algorithms. In *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics*, volume 3, pages 2379–2384, Oct 1998.

[307] M. Toulouse, T. G. Crainic, B. Sansó, and K. Thulasiraman. Self-organization in cooperative tabu search algorithms. In *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, volume 3, pages 2379–2384. IEEE, 1998.

[308] M. Toulouse, T. G. Crainic, and B. Sansó. *An Experimental Study of Systemic Behavior of Cooperative Search Algorithms*, pages 373–392. Springer US, Boston, MA, 1999.

[309] M. Toulouse, K. Thulasiraman, and F. Glover. Multi-level cooperative search: A new paradigm for combinatorial optimization and an application to graph partitioning. In P. Amestoy, P. Berger, M. Daydé, D. Ruiz, I. Duff, V. Frayssé, and L. Giraud, editors, *Euro-Par'99 Parallel Processing*, pages 533–542, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[310] M. Triska and N. Musliu. An improved sat formulation for the social golfer problem. *Annals of Operations Research*, 194(1):427–438, 2012.

[311] M. M. Van Hulle. Hill-climbing, density-based clustering and equiprobabilistic topographic maps. *Journal of VLSI signal processing systems for signal, image and video technology*, 26 (1-2):79–94, 2000.

[312] P. J. M. Van Laarhoven and E. H. L. Aarts. *Simulated annealing*, pages 7–15. Springer Netherlands, Dordrecht, 1987.

[313] P. J. M. Van Laarhoven, E. H. L. Aarts, and J. K. Lenstra. Job shop scheduling by simulated annealing. *Operations research*, 40(1):113–125, 1992.

[314] J. H. van Lint and R. Wilson. *A Course in Combinatorics*. Cambridge University Press, 1992.

[315] M. Vasile and L. Ricciardi. *Multi Agent Collaborative Search*, pages 223–252. Springer International Publishing, Cham, 2017.

[316] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, 40(1):475 – 489, 2013.

[317] P. F. Vinué. *Optimización de productos y procesos industriales*. Gestión 2000, 2006.

[318] N. Vo-Thanh, R. Jans, E. D. Schoen, and P. Goos. Symmetry breaking in mixed integer linear programming formulations for blocking two-level orthogonal experimental designs. *Computers & Operations Research*, 97:96 – 110, 2018. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2018.04.001.

[319] S. žák. A turing machine time hierarchy. *Theoretical Computer Science*, 26(3):327 – 333, 1983.

[320] M. Wallace, S. Novello, and J. Schimpf. Ecl i ps e: A platform for constraint logic programming. *ICL Systems Journal*, 12(1):159–200, 1997.

[321] T. Walsh. Constraint patterns. In F. Rossi, editor, *9th International Conference in Principles and Practice of Constraint Programming (CP 2003)*, volume 2833 of *Lecture Notes in Computer Science (LNCS)*, pages 53–64. Springer, 2003.

[322] T. Walsh. General symmetry breaking constraints. In F. Benhamou, editor, *Principles and Practice of Constraint Programming - CP 2006*, volume 4204 of *Lecture Notes in Computer Science (LNCS)*, pages 650–664. Springer Berlin Heidelberg, 2006.

[323] D. C. Walters and G. B. Sheble. Genetic algorithm solution of economic dispatch with valve point loading. *Power Systems, IEEE Transactions on*, 8(3):1325–1332, 1993.

[324] K.-P. Wang, L. Huang, C.-G. Zhou, and W. Pang. Particle swarm optimization for traveling salesman problem. In *Machine Learning and Cybernetics, 2003 International Conference on*, volume 3, pages 1583–1585 Vol.3, Nov 2003.

[325] Y. Wang, Z. Lü, F. Glover, and J.-K. Hao. Path relinking for unconstrained binary quadratic programming. *European Journal of Operational Research*, 223(3):595 – 604, 2012.

[326] Y. Wang, Z. Zhang, L. Y. Zhang, J. Feng, J. Gao, and P. Lei. A genetic algorithm for constructing bijective substitution boxes with high nonlinearity. *Information Sciences*, 523:152 – 166, 2020.

[327] J. M. Ware, I. D. Wilson, J. A. Ware, and C. B. Jones. A tabu search approach to automated map generalisation. In *Proceedings of the 10th ACM international symposium on Advances in geographic information systems*, pages 101–106. ACM, 2002.

[328] G. Wäscher and T. Gau. Heuristics for the integer one-dimensional cutting stock problem: A computational study. *Operations-Research-Spektrum*, 18(3):131–144, 1996.

[329] G. Wäscher, H. Haußner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130, 2007.

[330] D. Whitley and J. P. Watson. *Complexity Theory and the No Free Lunch Theorem*, pages 317–339. Springer US, Boston, MA, 2005.

[331] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[332] L. A. Wolsey. Integer programming duality: Price functions and sensitivity analysis. *Mathematical Programming*, 20(1):173–195, 1981.

[333] Yanli Yang, A. A. Minai, and M. M. Polycarpou. Decentralized cooperative search by networked uavs in an uncertain environment. In *Proceedings of the 2004 American Control Conference*, volume 6, pages 5558–5563 vol.6, June 2004.

[334] F. Yates. Incomplete randomized blocks. *Annals of Eugenics*, 7(2):121–140, 1936.

[335] M. Yavuz, E. Akcali, and S. Tufekci. A hybrid meta-heuristic for the batching problem in just-in-time flow shops. *Journal of Mathematical Modelling and Algorithms*, 5(3):371–393, 2006.

[336] D. Yokoya and T. Yamada. A mathematical programming approach to the construction of BIBDs. *International Journal of Computer Mathematics*, 88(5):1067–1082, 2011.

[337] C. Zhang, Z. Lin, and Z. Lin. Variable neighborhood search with permutation distance for qap. In R. Khosla, R. J. Howlett, and L. C. Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems*, volume 3684 of *Lecture Notes in Computer Science (LNCS)*, pages 81–88. Springer Berlin Heidelberg, 2005.

[338] L. Zhang, S. Guo, Y. Zhu, and A. Lim. A tabu search algorithm for the safe transportation of hazardous materials. In *Proceedings of the 2005 ACM symposium on Applied computing*, pages 940–946. ACM, 2005.

[339] E. Zitzler. *Evolutionary Multiobjective Optimization*, pages 871–904. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[340] K. G. Zografos and K. N. Androutsopoulos. A heuristic algorithm for solving hazardous materials distribution problems. *European Journal of Operational Research*, 152(2):507–519, 2004.

Índice Alfabético