Teoría de landscapes en optimización combinatoria: resultados recientes y herramienta software

Francisco Chicano y Enrique Alba

Resumen— **La teoría de *landscapes* es un marco que permite analizar funciones definidas sobre conjuntos de elementos conectados mediante un operador de vecindario. Esta teoría tiene aplicaciones en diversos dominios de conocimiento, entre los que se encuentra la optimización combinatoria. Este trabajo tiene un doble objetivo. Por un lado, presenta los fundamentos de la teoría y los resultados recientes más relevantes aplicados a los problemas de optimización combinatoria, dando al lector una visión global del estado del arte en este contexto. En segundo lugar, presenta una herramienta software, *Landscape Explorer*, que encapsula gran parte del conocimiento adquirido por los autores en teoría de *landscapes*. Esta herramienta pretende ser un primer punto de encuentro entre los investigadores interesados y la teoría. El diseño arquitectónico de la herramienta está pensado para su fácil extensión por parte de cualquier investigador de la comunidad científica.**

Palabras clave— **Teoría de *landscapes*, optimización combinatoria, herramienta software**

I. Introducción

Se denomina *teoría de landscapes* a un conjunto de definiciones, teoremas e hipótesis que permiten analizar problemas de optimización ligados a operadores de vecindario definidos sobre el espacio de búsqueda. Un *landscape* es una terna (X, N, f), donde $f : X \to \mathbb{R}$ define la función objetivo y el *vecindario* $N : X \to 2^X$ es una función que asigna un conjunto de soluciones a cada $x \in X$. Si $y \in N(x)$ entonces decimos que y es un vecino de x.

Aunque nuestro interés en esta teoría se refiere a su relación con la optimización combinatoria, sus aplicaciones se extienden más allá de ésta. Así encontramos aplicaciones de la teoría de *landscapes* en Química [1], Biología [2] y Física [3]. Uno de los principales objetivos de la teoría es comprender mejor la estructura de los problemas de optimización. Gracias a esta comprensión más profunda de los problemas es posible definir nuevos operadores, estrategias de búsqueda o incluso determinar los valores óptimos de los parámetros usados en la ejecución de un algoritmo.

Hay un tipo de *landscape* de especial interés debido a sus propiedades. Son los llamados *landscapes elementales* y se encuentran caracterizados por una *ecuación de onda* debida a Grover [4]:

$$\operatorname*{avg}_{y \in N(x)}\{f(y)\} = f(x) + \frac{k}{d}\left(\bar{f} - f(x)\right), \qquad (1)$$

donde d es el tamaño del vecindario, $|N(x)|$, que asumimos que es el mismo para todas las soluciones en el espacio de búsqueda, $\bar{f}$ es la media de la función objetivo en el espacio completo de búsqueda y k es una constante característica. Tanto k como $\bar{f}$ se pueden calcular de forma exacta y eficiente a partir de los datos del problema o mediante un muestreo aleatorio del espacio de búsqueda.

La ventaja de una expresión como (1) es que permite calcular un estadístico a partir del valor de la función en x: la media del valor que toma la función objetivo en el vecindario de una solución x. Obsérvese que si la ecuación (1) no fuese aplicable, sería necesario evaluar todas las soluciones en $N(x)$ para obtener dicho estadístico. Por tanto, concluimos que la teoría de *landscapes*, por lo pronto, proporciona resultados que permiten calcular de forma eficiente valores estadísticos no triviales relacionados con distribuciones de la función objetivo. Esta es una propiedad que acompaña a la gran mayoría de los resultados de la teoría de *landscapes*, como veremos en la Sección II.

En los últimos años varios investigadores han liderado la investigación en teoría de *landscapes* con aplicaciones en la optimización combinatoria. Cabe destacar la labor de Andrew Sutton y Darrell Whitley en la Universidad de Colorado (EEUU), Bill Langdon en la University College London (Reino Unido), Guanzhou Lu y Xin Yao en la Universidad de Birmingham (Reino Unido) y Francisco Chicano y Enrique Alba en la Universidad de Málaga (España). Entre las aportaciones de estos investigadores encontramos nuevos algoritmos de búsqueda [5] y mecanismos para escapar de *plateaus* [6], algoritmos para calcular estadísticos sobre conjuntos de soluciones cercanas a una dada [7], una metodología para encontrar la descomposición en componentes elementales de un *landscape* [8], un método para estimar las distribuciones de probabilidad de la función objetivo en el vecindario de una solución [9], la descomposición en componentes elementales de diversos problemas [10], [11] y algoritmos eficientes para el cálculo de medidas de autocorrelación en problemas NP-duros a partir de los datos del problema [12].

Entre la lista de logros de la teoría de *landscapes* mencionados en el párrafo anterior, podemos observar que algunas de las aportaciones son de índole práctica o tienen aplicación directa dentro de un algoritmo de búsqueda. Dichas aportaciones, en mu-

E.T.S. Ingeniería Informática, Universidad de Málaga, España. E-mail: {chicano,eat}@lcc.uma.es

chos casos, dan lugar a un algoritmo o método que puede implementarse y que, aunque para su elaboración es necesario un conocimiento relativamente profundo de la teoría, para su uso no es necesario tal conocimiento. Es decir, resulta posible en muchos casos *encapsular* el conocimiento adquirido mediante el estudio de la teoría de *landscapes* en un pequeño trozo de código que puede ser de utilidad práctica, como ya se ha demostrado en varios trabajos. Además, muchos de los resultados de la teoría de *landscapes* son generales y aplicables a todos los problemas de optimización combinatoria o grandes subconjuntos de éstos. Todo ello justifica el interés reciente en dicha teoría.

A lo largo de estos años de investigación en la teoría de *landscapes* hemos desarrollado con frecuencia pequeños fragmentos de código y algoritmos que nos han permitido comprobar empíricamente los resultados de la teoría y estudiar las implicaciones prácticas de ésta. Pensamos que estos algoritmos pueden resultar de gran utilidad para el resto de investigadores del dominio y, por ese motivo, hemos decidido desarrollar una aplicación, a la que hemos llamado *Landscape Explorer*, que incorpora estos algoritmos. Esta aplicación, que presentamos en este artículo por primer vez, está diseñada para permitir una fácil extensión. Nuestro objetivo es que sea una herramienta útil tanto para los investigadores en teoría de *landscapes* como para aquellas personas ajenas a la teoría que quieren comprobar lo que se puede hacer con ella.

El resto del artículo está organizado como sigue. En la Sección II presentamos los fundamentos de la teoría de *landscapes*. La Sección III presenta los resultados más relevantes de la teoría en los últimos años. A continuación se presenta la aplicación *Landscape Explorer* en la Sección IV y se concluye el artículo en la Sección V.

II. Fundamentos de landscapes

En esta sección presentamos algunos conceptos básicos de la teoría de *landscapes* necesarios para entender el resto del artículo. El lector interesado en ampliar estos conceptos puede consultar [13].

Sea (X, N, f) un *landscape*, donde X es un conjunto finito (de soluciones al problema), $f : X \to \mathbb{R}$ es una función real definida sobre X y $N : X \to 2^X$ es el vecindario. Al par (X, N) se le llama *espacio de configuración* y se puede representar mediante un grafo $G(X, E)$ en el que X es el conjunto de vértices y un arco (x, y) está presente en el grafo si $y \in N(x)$ [14]. Podemos caracterizar el vecindario mediante su matriz de adyacencia:

$$\boldsymbol{A}_{xy} = \begin{cases} 1 & \text{si } y \in N(x), \\ 0 & \text{en otro caso.} \end{cases} \tag{2}$$

La matriz de grado $\boldsymbol{D}$ se define como la matriz diagonal

$$\boldsymbol{D}_{xy} = \begin{cases} |N(x)| & \text{si } x = y, \\ 0 & \text{en otro caso.} \end{cases} \tag{3}$$

La matriz Laplaciana del espacio de configuración se define como:

$$\boldsymbol{\Delta} = \boldsymbol{A} - \boldsymbol{D}. \tag{4}$$

Cualquier función discreta, f, definida sobre el conjunto de soluciones candidatas se puede caracterizar con un vector de $\mathbb{R}^{|X|}$. Asimismo, cualquier matriz de dimensión $|X| \times |X|$ se puede interpretar como una aplicación lineal que actúa sobre el espacio vectorial $\mathbb{R}^{|X|}$. Por ejemplo, la matriz de adyacencia $\boldsymbol{A}$ actúa sobre la función f como sigue:

$$\boldsymbol{A}\ f = \begin{pmatrix} \sum_{y \in N(x_1)} f(y) \\ \sum_{y \in N(x_2)} f(y) \\ \vdots \\ \sum_{y \in N(x_{|X|})} f(y) \end{pmatrix}. \tag{5}$$

El componente x de este producto de matriz por vector se puede escribir de la siguiente forma:

$$(\boldsymbol{A}\ f)(x) = \sum_{y \in N(x)} f(y), \tag{6}$$

que es la suma del valor de la función de todos los vecinos de x. En este artículo sólo consideraremos vecindarios regulares (todas las soluciones tienen el mismo número de vecinos) y simétricos (si $y \in N(x)$ entonces $x \in N(y)$). Bajo estas condiciones, la matriz Laplaciana se puede escribir así: $\boldsymbol{\Delta} = \boldsymbol{A} - d\boldsymbol{I}$. Stadler denomina *landscapes elementales* a aquellos *landscapes* en los que la función f es un autovector (o autofunción) de la matriz Laplaciana del espacio de configuración, salvo por una constante [15]. Una definición formal es la siguiente.

Definición 1. *Sea* (X, N, f) *un* landscape *y* $\boldsymbol{\Delta}$ *la matriz Laplaciana del espacio de configuración. La función* f *se dice que es* elemental *si existe una constante* b*, que llamamos* offset*, y un autovalor* λ *de* $-\boldsymbol{\Delta}$ *tal que* $(-\boldsymbol{\Delta})(f - b) = \lambda(f - b)$*. El* landscape *se dice que es elemental si* f *lo es.*

Usamos autovalores de $-\boldsymbol{\Delta}$ en lugar de $\boldsymbol{\Delta}$ simplemente para trabajar con valores positivos [14]. De acuerdo con la definición previa, toda función elemental, f, se puede escribir como la suma de una autofunción de $-\boldsymbol{\Delta}$, que denotamos con g, y una constante b, es decir, $f = g + b$. El autovalor λ es un valor característico del *landscape*: depende de la función objetivo f y del espacio de configuración (X, N). En vecindarios conexos (que son los que consideraremos en este trabajo) el *offset* b es el promedio del valor de la función objetivo sobre el espacio de búsqueda completo: $b = \bar{f}$. En los *landscapes* elementales, el promedio $\bar{f}$ se puede calcular normalmente de forma muy eficiente usando los datos del problema. Es

decir, no se requiere realizar una enumeración completa del espacio de búsqueda.

Usando resultados básicos de álgebra lineal, se puede demostrar que si f es elemental con autovalor λ, entonces $af+b$ es también elemental con el mismo autovalor λ. Es más, en los vecindarios regulares, si g es una autofunción de $-\boldsymbol{\Delta}$ con autovalor λ entonces g es también autofunción de la matriz de adyacencia $\boldsymbol{A}$ con autovalor $d-\lambda$. El promedio del valor de la función objetivo en el vecindario de una solución x se puede calcular usando la siguiente expresión:

$$\operatorname*{avg}_{y\in N(x)}\{f(y)\} = \frac{1}{d}(\boldsymbol{A}\ f)(x). \tag{7}$$

Si f es una función elemental con autovalor λ, entonces dicho promedio se puede calcular como:

$$\begin{aligned}\operatorname*{avg}_{y\in N(x)}\{f(y)\} &= \operatorname*{avg}_{y\in N(x)}\{f(y)-\bar{f}\}+\bar{f}\\ &= \frac{1}{d}(\boldsymbol{A}\ (f-\bar{f}))(x)+\bar{f}\\ &= \frac{d-\lambda}{d}(f(x)-\bar{f})+\bar{f}\\ &= f(x)+\frac{\lambda}{d}(\bar{f}-f(x)),\end{aligned} \tag{8}$$

y obtenemos de esta forma la ecuación de onda de Grover (1). En la expresión anterior hemos utilizado el hecho de que $f-\bar{f}$ es autofunción de $\boldsymbol{A}$ con autovalor $d-\lambda$.

Un *landscape* (X, N, f) no siempre es elemental, pero incluso en ese caso es posible caracterizar la función f como una suma de funciones que sí son elementales y todo problema combinatorio es pues susceptible de tal descomposición. Llamamos a estas funciones *componentes elementales* de f. Tal descomposición es útil desde un punto de vista teórico y práctico. En la teoría, la descomposición de un problema se puede usar para calcular la expresión exacta de las funciones de autocorrelación, el coeficiente de autocorrelación y la longitud de autocorrelación [16]. Las anteriores medidas de autocorrelación permiten caracterizar la función objetivo de tal forma que se pueden realizar predicciones del rendimiento de determinadas técnicas de búsqueda local. Desde un punto de vista práctico, la descomposición en *landscapes* elementales junto con la ecuación de onda de Grover se puede usar también para calcular el valor medio de la función objetivo en el vecindario de una solución, lo cual puede ser la base para el desarrollo de nuevos operadores o algoritmos [5], [6], [17]. El lector interesado en la descomposición de *landscapes* puede consultar [8].

III. Resultados recientes

En la sección precedente hemos presentado conceptos y resultados generales de la teoría de *landscapes*. En esta, vamos a centrarnos en ciertos espacios de configuraciones particulares. Una vez fijado el espacio de soluciones y el vecindario empleado, es posible aportar bastantes más resultados que son de interés teórico y práctico. En las siguientes secciones presentaremos los principales avances de la teoría de *landscapes* en tres espacios de configuración diferentes: los hipercubos binarios, los hipercubos generales y el espacio de permutaciones.

A. Hipercubo binario

Llamamos *hipercubo binario* a un espacio de configuración en el que el conjunto de soluciones X está formado por cadenas binarias de longitud n, esto es, $X=\mathbb{B}^n$; y dos soluciones cualesquiera x e y son vecinas si la distancia de Hamming entre ellas es 1, es decir, hay un dígito que difiere entre las cadenas. El grafo asociado a un hipercubo binario es conexo, simétrico y regular. Como consecuencia, existe al menos una base para el espacio de posibles funciones objetivo, $\mathbb{R}^{|X|}$, que está formada por autovectores de la matriz Laplaciana.

Una de estas bases, posiblemente la más popular, está formada por las funciones de Walsh [18]. Dada una cadena binaria $w\in\mathbb{B}^n$, la función de Walsh con parámetro w está definida de la siguiente forma:

$$\psi_w(x) = \prod_{i=1}^{n}(-1)^{w_i x_i} = (-1)^{\sum_{i=1}^{n} w_i x_i}. \tag{9}$$

Esta función es autovector de la matriz Laplaciana con autovalor $2|w|$, donde $|w|$ denota el número de unos de la cadena w. Dada una función cualquiera f, podemos siempre escribirla como suma ponderada de funciones de Walsh:

$$f = \sum_{w\in\mathbb{B}^n} z_w\psi_w, \tag{10}$$

donde z_w son los denominados *coeficientes de Walsh*.

Además, siempre es posible agrupar las funciones de Walsh con el mismo autovalor para formar una *componente elemental* de f y así obtener:

$$f = \sum_{j=0}^{n}\sum_{\substack{w\in\mathbb{B}^n\\|w|=j}} z_w\psi_w = \sum_{j=0}^{n} f_{[j]}, \tag{11}$$

donde $f_{[j]}$ denota la componente elemental de f con autovalor $2j$.

Sutton *et al.* [7] analizan los *landscapes* en hipercubos binarios. Demuestran que las funciones que son elementales en dicho espacio de configuración también lo son cuando el vecindario de una solución está formado por todas aquellas soluciones que se encuentran a una distancia de Hamming arbitraria r de la solución original x (esfera de radio r). Este conjunto de soluciones lo denotaremos con $S^{(r)}(x)$. Como consecuencia directa, resulta posible calcular el promedio del valor de la función objetivo en $S^{(r)}(x)$. Este valor es:

$$\operatorname*{avg}_{y\in S^{(r)}(x)}\{f(y)\} = \sum_{j=0}^{n}\mathcal{K}_{rj}^{(n)} f_{[j]}(x), \tag{12}$$

donde $\mathcal{K}_{rj}^{(n)}$ es el elemento (r, j) de la matriz de Krawtchouk de orden n, que viene dado por:

$$\mathcal{K}_{rj}^{(n)} = \sum_{l=\text{máx}(0,r+j-n)}^{\text{mín}(r,j)} (-1)^l \left(\begin{array}{c} n-j \\ r-l \end{array} \right) \left(\begin{array}{c} j \\ l \end{array} \right). \tag{13}$$

La expresión anterior permite realizar un cálculo eficiente de $\text{avg}\{f(y)\}_{y \in S^{(r)}(x)}$ siempre que conozcamos la descomposición en componentes elementales de f. Si no se usa (12) el cálculo sería muy ineficiente, especialmente para valores altos de r. La expresión anterior puede extenderse también para considerar bolas de tamaño arbitrario r, es decir, conjuntos de soluciones que tienen distancia de Hamming r o menos a x. Estos resultados permiten generalizar la ecuación (1), extendiendo el cálculo del promedio a estas esferas y bolas de radio arbitrario.

Pero los resultados de Sutton *et al.* van más allá y consiguen calcular sobre estos nuevos vecindarios momentos de orden superior al primero (media). Así pues, además de la media, pueden calcular los momentos de segundo orden, tercero, cuarto y, en general, de cualquier otro orden, todo ello en tiempo polinomial. Esto les permite calcular de forma exacta la varianza, el sesgo, la curtosis, etc. Con ayuda de estos estadísticos, son capaces de estimar la distribución de probabilidad de los valores de la función objetivo en torno a una solución [9]. Desde un punto de vista práctico, el cálculo de estos estadísticos se ha usado para proponer nuevas estrategias de búsqueda [5], nuevos operadores [17] y estrategias para escapar de plateaus [6].

Tomando como base los resultados de Sutton *et al.* acerca de las funciones elementales en esferas de radio arbitrario, Chicano y Alba [19] demostraron que la esperanza del valor de la función objetivo obtenido tras aplicar la mutación por inversión de bits a una solución x viene dada por la expresión

$$\mathbb{E}[f]_x = \bar{f} + \sum_{j=1}^{n} (1-2p)^j f_{[j]}(x), \tag{14}$$

donde p es la probabilidad de cambiar un bit de la cadena. Se puede observar que $\mathbb{E}[f]_x$ es un polinomio en p. Fijado x es posible, en teoría, calcular el valor p para optimizar la esperanza del valor objetivo de la solución tras la mutación. De esta forma se podría proporcionar un valor de p ajustado para cada solución x y óptimo con respecto a esta esperanza. Dicho operador de mutación adaptativo ha sido analizado por Sutton *et al.* en [20] y los resultados sugieren que puede ser muy útil en las primeras generaciones de un algoritmo genético, acelerando la búsqueda al principio.

B. Hipercubos generalizados

Se denomina *hipercubo generalizado* o, más concretamente, *hipercubo q-ario* al espacio de configuración en el que el conjunto de soluciones X está formado por cadenas de un alfabeto finito Γ de cardinal q; y dos soluciones cualesquiera x e y son vecinas cuando sólo difieren en un componente de la solución. Este espacio de configuración es una generalización del hipercubo binario, en la cual cada componente de la solución puede tomar q valores diferentes, en lugar de sólo dos.

Existen funciones de Walsh generalizadas de valores complejos que forman una base de este espacio de configuración [3]. Sin embargo, los trabajos recientes que encontramos basados en hipercubos generalizados no usan dichas bases para encontrar la descomposición en componentes elementales de los problemas de optimización combinatoria.

Whitley y Sutton [21] demuestran que el problema de coloreado de grafos es un *landscape* elemental. Para ello usan un modelo basado en componentes del problema. Usando este modelo de componentes junto con algunos resultados algebraicos, Whitley *et al.* [22] demuestran que el problema de asignación de frecuencias en una red celular (*Frequency Assignment Problem*, FAP) se puede descomponer en suma de dos componentes elementales. El problema de coloreado de grafos es un caso particular de FAP. Más tarde, Chicano *et al.* [10] demuestran usando argumentos puramente algebraicos que la versión más general de FAP es también suma de dos componentes elementales y caracterizan los casos en que el problema se convierte en elemental. Uno de estos casos es precisamente aquél en que el problema representa una instancia del problema de coloreado de grafos.

Recientes investigaciones aún no publicadas sugieren que los resultados mencionados en el apartado anterior para los hipercubos binarios se pueden generalizar al caso de los hipercubos q-arios. De esta forma, sería posible calcular momentos de cualquier orden sobre esferas y bolas de radio arbitrario así como calcular la esperanza del valor de la función objetivo tras aplicar un tipo concreto de mutación a una solución arbitraria.

C. Permutaciones

Por último, consideramos el caso en el que el conjunto de soluciones X es el conjunto de todas las permutaciones de tamaño n, $X = S_n$. Se han definido varios espacios de configuración basados en permutaciones entre los que distinguimos dos. El primero utiliza como vecindario el conjunto $N(x)$ formado por todas las soluciones que se pueden construir a partir de x aplicando un movimiento 2-opt. En el segundo, el conjunto de vecinos $N(x)$ está formado por todas las soluciones que se obtienen a partir x mediante el intercambio de dos posiciones (*swap*).

Whitley y Sutton [21] demuestran que el problema del viajante de comercio (TSP) es un *landscape* elemental bajo el vecindario 2-opt. Más tarde, Whitley y Ochoa [23] presentan expresiones para optimizar el cálculo del promedio de los valores de la función objetivo en vecindarios parciales para este problema. Denominan vecindario parcial de una solución a un

subconjunto de $N(x)$ (el vecindario completo) para el cual sigue siendo válida la ecuación de onda de Grover (1) con ligeras variaciones.

Por otro lado, Chicano *et al.* estudian el problema de asignación cuadrática (*Quadratic Assignment Problem*, QAP) sobre el espacio de configuración inducido por el vecindario basado en intercambio. Demuestran en [24] que dicho problema se puede descomponer como suma de tres componentes elementales y presentan en [12] un algoritmo eficiente para el cálculo de las medidas de autocorrelación de las instancias de dicho problema. TSP es un caso particular de QAP y, por tanto, le son aplicables todos estos resultados. De hecho, Chicano *et al.* demuestran que TSP se puede descomponer en suma de, a lo sumo, dos componentes elementales. En particular, TSP es elemental cuando las distancias entre ciudades son simétricas (o antisimétricas).

IV. Landscape Explorer

Hasta ahora, los investigadores interesados en analizar un *landscape* usando los resultados que hemos presentado anteriormente debían tener un conocimiento sólido de la teoría de *landscapes*. El principal objetivo perseguido con la herramienta software que presentamos en esta sección es el de ocultar, en la medida de lo posible, detalles matemáticos a los investigadores que deseen analizar el *landscape* del problema que les interesa. Pretendemos que nuestra herramienta, *Landscape Explorer*, sea una gran aliada de estos investigadores. Esta herramienta está disponible para su descarga en la URL `http://neo.lcc.uma.es/software/landexplorer`.

El principal requisito que hemos considerado a la hora de elaborar el diseño arquitectónico de *Landscape Explorer* es su extensibilidad. Ello está motivado por el hecho de que la teoría de *landscapes* no es algo cerrado, sino que crece continuamente con aportaciones de distintos equipos de investigación. Por tanto, si queremos que la herramienta resulte siempre útil debemos dotarla de mecanismos sencillos de extensión. Además, la posibilidad de extensión debe ir acompañada de una gran modularidad del código, de forma que las nuevas extensiones o variaciones de la herramienta original se puedan conseguir fácilmente incorporando o modificando los módulos que componen la herramienta.

Un segundo requisito de la herramienta es su ejecución multiplataforma. El desarrollo de aplicaciones en lenguaje Java o en la plataforma .NET evita tener que tratar con detalles de implementación que dependen de los distintos sistemas operativos o arquitecturas de computadores, facilitando así la portabilidad de la herramienta.

Para cubrir los dos requisitos mencionados en los párrafos anteriores hemos basado *Landscape Explorer* en la *Rich Client Platform* (RCP) de *Eclipse*, que usa Java como lenguaje de programación. De esta forma, podemos pensar en *Landscape Explorer* como un conjunto de módulos desarrollados en Java (*plug-ins*) con dependencias e interacciones entre ellos. Para extender la aplicación con una nueva característica, tan sólo es necesario implementar dicha característica en un nuevo módulo y añadirlo con los demás. De la misma forma es posible eliminar módulos o sustituirlos por otras versiones. Algunos ejemplos de otras aplicaciones basadas en la plataforma RCP son: el IDE de Eclipse, Azureus, BaZaa y CCTVnet.

En las siguientes secciones se presenta una breve introducción a la RCP de *Eclipse*, se describe la arquitectura de la aplicación y se mencionan los distintos tipos de análisis incluidos en la versión actual de la herramienta.

A. Eclipse RCP

La *Rich Client Platform* (RCP) de *Eclipse* es un conjunto de módulos que ofrecen servicios para que los desarrolladores puedan construir herramientas software extensibles y modulares. La RCP está implementada encima de *Equinox*, que es a su vez una implementación de la *OSGi R4 core framework specification*. El objetivo de OSGi con su especificación, y de Equinox con su implementación, es ofrecer una plataforma para el desarrollo de *ecosistemas software*, que son un conjunto de módulos, denominados *bundles* en la jerga de OGSi y *plug-ins* en la de *Eclipse*, con dependencias e interacciones entre ellos. Cada uno de estos módulos debe especificar de qué módulos dependen y qué funcionalidad exportan para que otros módulos puedan usarlas. *Equinox* proporciona el entorno de tiempo de ejecución para que todos estos módulos puedan enlazarse entre ellos y colaborar para realizar una determinada tarea.

Además de los servicios propios de *Equinox*, la RCP proporciona dos *plug-ins*, `org.eclipse.ui` y `org.eclipse.core.runtime`, que ofrecen bibliotecas de clases para crear interfaces gráficas de usuario (por medio de las bibliotecas SWT y JFace), acceder a las funciones exportadas por otros *plug-ins*, extender las funciones de otros *plug-ins* y ejecutar tareas intensivas en CPU, entre otras. Una aplicación desarrollada con RCP constará de un conjunto de *plug-ins* (entre los que se encuentran los dos mencionados anteriormente y todos aquéllos de los que dependen) y un fichero ejecutable que será el encargado de señalar la clase que actuará como principal (ver Figura 1).

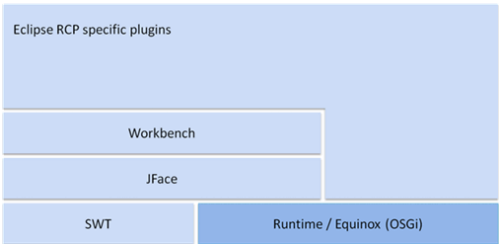

Fig. 1. Estructura en bloques de una aplicación RCP.

El principal mecanismo de extensión en la RCP está basado en los conceptos de *punto de extensión* y *extensión*. Una *extensión* representa una nueva funcionalidad que se incorpora al software. Esta extensión debe conectarse a uno de los *puntos de extensión* existentes en el mismo. Como requisito para que la conexión sea correcta, la extensión debe proporcionar la información requerida por el punto de extensión en el formato correcto. Esta información puede ser desde cadenas de texto hasta clases Java que implementan una determinada interfaz.

Los *plug-ins* pueden definir tanto extensiones como puntos de extensión. De esta forma, si un *plug-in* define un punto de extensión favorece la extensión del software. Por otro lado, los *plug-ins* que definen extensiones están contribuyendo a ampliar la funcionalidad del software. La descripción de las extensiones y los puntos de extensión presentes en un *plug-in* se encuentra en un fichero dentro del propio archivo `jar` del *plug-in* y el entorno de ejecución se encarga de leerlo para proporcionar dicha información a los *plug-ins* que deseen consultarla. Sin embargo, es responsabilidad de los *plug-ins* que definieron los puntos de extensión el comprobar que hay otros *plug-ins* que han creado extensiones para éstos y realizar las acciones oportunas para que dichas funciones estén disponibles de cara al usuario final.

Además de los puntos de extensión y las extensiones, los *plug-ins* pueden declarar públicos ciertos paquetes definidos en ellos, de esta forma se pueden añadir con facilidad bibliotecas de clases necesarias para determinadas funciones de la herramienta.

B. Arquitectura de la aplicación

Dado el carácter abierto de la teoría de *landscapes* no es posible predecir el tipo de análisis y de elementos software que serán necesarios en el futuro para investigar en este campo. Por este motivo, se ha diseñado la arquitectura de *Landscape Explorer* de forma que permita casi cualquier extensión posible. Tan sólo se ha especificado lo que pensamos que puede ser un conjunto mínimo de clases y procedimientos que cualquier análisis podría necesitar. Sin embargo, incluso este conjunto mínimo podría cambiar con el paso del tiempo y la modularización de la aplicación permitiría realizar este cambio con relativa facilidad.

Las clases y procedimientos básicos de la aplicación están repartidos en dos *plug-ins* (Figura 2):

- `neo.landscapes.theory.kernel`: proporciona las interfaces y clases que definen lo que es un *landscape* (desde el punto de vista de la implementación) y ofrece varios ejemplos de ellos.
- `neo.landscapes.theory.tool`: es el *plug-in* que contiene el punto de entrada a la aplicación y es responsable de mostrar la interfaz gráfica de usuario (GUI) de la aplicación así como de definir los principales puntos de extensión de *Landscape Explorer*.

El primer *plug-in*, al que llamaremos `kernel` para ahorrar notación, define un punto de extensión,

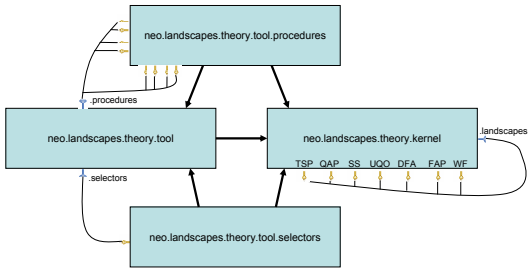

Fig. 2. Conjunto de *plug-ins* que componen la aplicación.

`neo.landscapes.theory.kernel.landscapes`, que debe ser extendido por todos aquellos *plug-ins* que quieran definir *landscapes* para ser usados en los distintos análisis que proporcione la herramienta. Cuando un investigador desea aplicar todos los análisis implementados en la herramienta a un problema concreto en que está interesado, debe crear un *plug-in* que defina una extensión para este punto de extensión. En la fecha de escritura de este artículo, el propio `kernel` define seis extensiones para este punto que se corresponden con los seis problemas de optimización que mencionaremos en la Sección IV-C.

El segundo *plug-in* del núcleo de *Landscape Explorer*, al que denominaremos `tool`, define los siguientes puntos de extensión:

- `neo.landscapes.theory.tool.procedures`: permite a otros *plug-ins* definir métodos de análisis de *landscapes*. Los *plug-ins* que extiendan este punto de extensión deben proporcionar una clase que implemente la interfaz `IProcedure`. Además de proporcionar el nombre y una descripción del método de análisis, la clase debe construir un componente de GUI para que el usuario interaccione con el método de análisis y pueda observar los resultados. Permitiendo que sea el desarrollador de la extensión el que defina la GUI para el método de análisis conseguimos minimizar los requisitos que dicho método debe cumplir.
- `neo.landscapes.theory.tool.selectors`: uno de los pasos básicos para realizar el análisis de un *landscape* es seleccionar dicho *landscape*. Este punto de extensión permite a los desarrolladores proporcionar procedimientos de selección de *landscapes* para que los desarrolladores de nuevos métodos de análisis no tengan que hacerlo en cada método. Los desarrolladores que deseen proporcionar un procedimiento de selección deberán proporcionar una clase que implemente la interfaz `ISelector` que, entre otras cosas, devuelve un componente gráfico asociado a dicho procedimiento de selección.

Además de los mencionados puntos de extensión, el *plug-in* `tool` exporta una clase, llamada `SelectionServices`, que ofrece operaciones públicas para la selección de *landscapes*. Esta clase está pensada para ser usada por parte de los métodos de análisis. Cuando un método de análisis necesita que el usuario escoja un *landscape* para trabajar con él, éste debe pedir el *landscape* usan-

do la clase `SelectionServices`, la cual, a su vez, usará alguno de los procedimientos de selección registrados en el sistema bajo el punto de extensión `neo.landscapes.theory.tool.selectors`. El *landscape* elegido puede tener parámetros que el usuario debe proporcionar. En este caso, es responsabilidad del método de análisis consultar dichos parámetros y ofrecer al usuario la oportunidad de rellenarlos.

Para añadir un nuevo *landscape*, método de análisis o procedimiento de selección, el desarrollador debe crear un *plug-in* donde defina la extensión correspondiente y colocar el archivo `jar` del *plug-in* en la carpeta *plugins* de la aplicación, ubicada en el directorio principal de la misma.

C. Funciones incluidas

En la sección anterior hemos descrito los *plug-ins* que forman el núcleo de *Landscape Explorer*. En ésta vamos a describir el procedimiento de selección y los métodos de análisis que se proporcionan junto con la aplicación en el momento de escritura de esta líneas.

Un tercer *plug-in* incluido en *Landscape Explorer* es `neo.landscapes.theory.tool.selectors`, que contiene un procedimiento de selección de *landscapes* básico. En dicho procedimiento se muestra al usuario los distintos *landscapes* definidos como extensiones del punto `neo.landscapes.kernel.landscapes`. Estos *landscapes* son siete actualmente, todos ellos parametrizables:

- *Quadratic Assignment Problem* (QAP): es un problema clásico de optimización combinatoria. El usuario debe proporcionar una ruta a un fichero de texto codificado en el mismo formato que las instancias de QAPLIB [25].
- *Traveling Salesman Problem* (TSP): el problema del viajante de comercio. El usuario debe proporcional una ruta a un fichero con la instancia.
- *Unconstrained Quadratic Optimizaion* (UQO): un problema de optimización cuadrática binaria NP-duro [26]. El usuario debe proporcionar un fichero de texto con la instancia, que debe contener la matriz de coeficientes.
- *Subset Sum* (SS): el problema de la suma de subconjunto. El usuario debe proporcionar el tamaño del problema y una semilla aleatoria. La instancia del problema será aleatoriamente generada.
- *Frequency Assignment Problem* (FAP): problema de asignación de frecuencias. Es un problema de telecomunicaciones que está relacionado con el coloreado de grafos. El usuario debe proporcionar el número de nodos de la red y el número de frecuencias que se pueden utilizar. El *landscape* que se analiza es el correspondiente a un eje del grafo.
- *DNA Fragment Assembly* (DFA): problema de ensamblado de fragmentos de ADN. Se trata de un problema de bioinformática relacionado con el QAP para el que hay definidas dos posibles funciones de objetivo. El usuario debe proporcionar el tamaño del problema y la función objetivo que quiere analizar.
- *Walsh Functions* (WF): suma controlada de funciones de Walsh. Este *landscape* no se corresponde con ningún problema concreto de optimización. Sin embargo, cualquier problema definido sobre un hipercubo binario se puede escribir como suma de funciones de Walsh. El usuario debe proporcionar el tamaño del espacio de búsqueda, el orden de las funciones de Walsh que se usarán en la suma y una semilla aleatoria. Se generará una suma aleatoria de las funciones de Walsh de orden indicado.

El último *plug-in* incluido en la aplicación es `neo.landscapes.theory.tool.procedures` que define varios métodos de análisis de landscapes. Algunos de estos métodos son aplicables a todos los *landscapes* o una gran familia de ellos. Estos son:

- **Medida empírica de la autocorrelación**. Este método realiza un recorrido aleatorio del espacio de búsqueda saltando de una solución x a un vecino $y \in N(x)$. A la vez que recorre el espacio de búsqueda calcula la autocorrelación de los valores de la función objetivo, obteniendo así los valores de la función de autocorrelación de Weinberger [2].
- **Chequeo de la condición de elemental**. Este método realiza un muestreo aleatorio de un conjunto de soluciones y sus vecinas para comprobar si el *landscape* es elemental. Puesto que el método no es exhaustivo, si responde afirmativamente al chequeo no puede asegurar que el *landscape* sea elemental. Por otro lado, si responde negativamente al chequeo, podemos estar seguros de que el *landscape* no es elemental. Si determina que el *landscape* podría ser elemental devuelve el autovalor asociado y la media de la función objetivo en el espacio de búsqueda.
- **Programa para** *Mathematica*. Dado un *landscape* cualquiera, este método devuelve un *script* escrito para la aplicación *Mathematica* que permite realizar la descomposición en *landscapes* elementales del *landscape* original. Este método resulta ser una pieza fundamental dentro de la metodología algebraica para la descomposición de *landscapes* [8].
- **Estimación del número de** *landscapes* **elementales**. Dado un *landscape* definido sobre un espacio de configuración binario, el método determina de forma empírica el número de *landscapes* elementales en que se puede descomponer el problema de optimización original o cualquiera de sus potencias. El resultado que ofrece no es un valor exacto pero se puede aproximar a éste conforme se aumente el número de muestras aleatorias utilizadas.

Además de los anteriores, se incluyen algunos métodos de medida de autocorrelación específicos para algunos problemas concretos. Estos problemas son QAP, TSP, UQO y SS. En todos los casos los métodos toman una instancia del problema y calculan de forma exacta el coeficiente de autocorrelación, la longitud de autocorrelación y la función de autocorrelación de Weinberger [2].

V. Conclusiones

Este artículo tiene un doble objetivo. Por un lado, proporciona una breve introducción a la *teoría de landscapes* y presenta los principales logros conseguidos en este campo en los últimos años. En segundo lugar, se presenta una nueva herramienta software, denominada *Landscape Explorer*, que facilita el análisis de *landscapes*.

Al investigador interesado en teoría de *landscapes* la herramienta aquí presentada le permite hacer un análisis rápido de los problemas de optimización para determinar ciertas propiedades de éstos. También le permite comprobar empíricamente resultados teóricos obtenidos mediante desarrollos matemáticos. Los investigadores interesados en un determinado problema de optimización pueden usar la herramienta para determinar la descomposición en componentes elementales del problema y poder aplicar los resultados prácticos que se derivan de tal descomposición.

En su estado actual, *Landscape Explorer* es un prototipo que puede crecer muy rápidamente con la ayuda de otros investigadores interesados en ella. Aunque ya está disponible en la Web, nuestro objetivo es colgarla próximamente en un sitio donde pueda tener visibilidad, como `sourceforge`. También planeamos incorporar a la herramienta todos los métodos de análisis que surjan nuevos con motivo del avance en los aspectos teóricos.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Ministerio de Ciencia e Innovación y fondos FEDER con números de proyecto TIN2008-06491-C04-01 (M*) y TIN2011-28194 (roadME) y por la Junta de Andalucía con número de proyecto P07-TIC-03044 (DIRICOM).

Referencias

[1] P F. Stadler, "Landscapes and their correlation functions," *Journal of Mathematical Chemistry*, vol. 20, pp. 1–45, 1996.

[2] E. Weinberger, "Correlated and uncorrelated fitness landscapes and how to tell the difference," *Biological Cybernetics*, vol. 63, no. 5, pp. 325–336, 1990.

[3] Ricardo García-Pelayo and Peter Stadler, "Correlation length, isotropy and meta-stable states," *Physica D: Nonlinear Phenomena*, vol. 107, no. 2-4, pp. 240–254, Sep 1997.

[4] L. K. Grover, "Local search and the local structure of NP-complete problems," *Operations Research Letters*, vol. 12, pp. 235–243, 1992.

[5] Guanzhou Lu, Rami Bahsoon, and Xin Yao, "Applying elementary landscape analysis to search-based software engineering," in *Proceedings of SSBSE*, 2010, pp. 3–8.

[6] A. M. Sutton, A. E. Howe, and L. D. Whitley, "Directed plateau search for MAX-k-SAT," in *Proceedings of SoCS*, Atlanta, GA, USA, July 2010.

[7] Andrew M. Sutton, L. Darrell Whitley, and Adele E. Howe, "Computing the moments of k-bounded pseudo-boolean functions over Hamming spheres of arbitrary radius in polynomial time," *Theoretical Computer Science*, in press (doi: 10.1016/j.tcs.2011.02.006).

[8] Francisco Chicano, L. Darrell Whitley, and Enrique Alba, "A methodology to find the elementary landscape decomposition of combinatorial optimization problems," *Evolutionary Computation*, vol. 19, no. 4, pp. 597–637, 2011.

[9] Andrew M. Sutton, Darrell Whitley, and Adele E. Howe, "Approximating the distribution of fitness over hamming regions," in *Proceedings of the 11th workshop proceedings on Foundations of genetic algorithms*, New York, NY, USA, 2011, FOGA '11, pp. 93–104, ACM.

[10] Francisco Chicano, L. Darrell Whitley, Enrique Alba, and Francisco Luna, "Elementary landscape decomposition of the frequency assignment problem," *Theoretical Computer Science*, vol. 412, no. 43, pp. 6002–6019, 2011.

[11] Francisco Chicano and Enrique Alba, "Elementary landscape decomposition of the 0-1 unconstrained quadratic optimization," *Journal of Heuristics*, 2011, in press (doi: 10.1007/s10732-011-9170-6).

[12] Francisco Chicano, Gabriel J. Luque, and Enrique Alba, "Autocorrelation measures for the quadratic assignment problem," *Applied Mathematics Letters*, 2011, in press (doi :10.1016/j.aml.2011.09.053).

[13] Christian M. Reidys and Peter F. Stadler, "Combinatorial landscapes," *SIAM Review*, vol. 44, no. 1, pp. 3–54, 2002.

[14] Türker Biyikoglu, Josef Leyold, and Peter F. Stadler, *Laplacian Eigenvectors of Graphs*, Lecture Notes in Mathematics. Springer-Verlag, 2007.

[15] P. F. Stadler, "Toward a theory of landscapes," in *Complex Systems and Binary Networks*, R. López-Peña, R. Capovilla, R. García-Pelayo, H.Waelbroeck, and F. Zertruche, Eds. 1995, pp. 77–163, Springer.

[16] E. Angel and V. Zissimopoulos, "On the classiffication of NP-complete problems in terms of their correlation coefficient," *Discr. App. Maths.*, vol. 99, pp. 261–277, 2000.

[17] Francisco Chicano, Javier Ferrer, and Enrique Alba, "Elementary landscape decomposition of the test suite minimization problem," in *Proccedings of SSBSE*, 2011, vol. 6956 of *LNCS*, pp. 48–63.

[18] Andrew M. Sutton, L. Darrell Whitley, and Adele E. Howe, "A polynomial time computation of the exact correlation structure of k-satisfiability landscapes," in *Proceedings of GECCO*, New York, NY, USA, 2009, pp. 365–372, ACM.

[19] Francisco Chicano and Enrique Alba, "Exact computation of the expectation curves of the bit-flip mutation using landscapes theory," in *Proceedings of GECCO*, 2011, pp. 2027–2034.

[20] Andrew M. Sutton, Darrell Whitley, and Adele E. Howe, "Mutation rates of the (1+1)-EA on pseudo-boolean functions of bounded epistasis," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 2011, GECCO '11, pp. 973–980.

[21] L. Darrell Whitley and Andrew M. Sutton, "Partial neighborhoods of elementary landscapes," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, New York, NY, USA, 2009, GECCO '09, pp. 381–388, ACM.

[22] L. Darrell Whitley, Francisco Chicano, Enrique Alba, and Francisco Luna, "Elementary landscapes of frequency assignment problems," in *In Proceedings of GECCO*, 2010, pp. 1409–1416.

[23] Darrell Whitley and Gabriela Ochoa, "Partial neighborhoods of the traveling salesman problem," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, New York, NY, USA, 2011, GECCO '11, pp. 529–536, ACM.

[24] Francisco Chicano, Gabriel Luque, and Enrique Alba, "Elementary landscape decomposition of the quadratic assignment problem," in *Proceedings of GECCO*, New York, NY, USA, 2010, pp. 1425–1432, ACM.

[25] R.E. Burkard, S.E. Karisch, and F. Rendl, "QAPLIB - a quadratic assignment problem library," *Journal of Global Optimization*, vol. 10, pp. 391–403, 1997.

[26] Fred Glover, Bahram Alidaee, César Rego, and Gary Kochenberger, "One-pass heuristics for large-scale unconstrained binary quadratic problems," *European Journal of Operational Research*, vol. 137, no. 2, pp. 272–287, Mar 2002.