An Analysis of Some Algorithms and Heuristics for Multiobjective Graph Search

UNIVERSIDAD
DE MÁLAGA

TESIS DOCTORAL

Enrique L. Machuca Sánchez

Universidad de Málaga

24 de Julio de 2012

Documento maquetado con TEXIS v.1.0.

Este documento está preparado para ser imprimido a doble cara.

An Analysis of Some Algorithms and Heuristics for Multiobjective Graph Search

Memoria que presenta el doctorando

Enrique L. Machuca Sánchez

para optar al grado académico de Doctor Ingeniero en Informática

Dirigida por el Doctor

Lorenzo Mandow Andaluz

Departamento de Lenguajes y Ciencias de la Computación
Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

24 de Julio de 2012

Tribunal de la tesis / Thesis Committee

Dr. José Luis Pérez de la Cruz Molina - Universidad de Málaga
Dra. Amparo Ruiz Sepúlveda- Universidad de Málaga
Dra. Raquel Fuentetaja Pizán - Universidad Carlos III de Madrid
Dra. Lucie Galand - Université Paris Dauphine
Dra. Camino Rodríguez Vela - Universidad de Oviedo

Evaluadores externos / External Reviewers

Dr. Christos Zaroliagis - University of Patras
Dr. Antonio Iovanella - University of Rome Tor Vergata

E-MAIL: machuca@lcc.uma.es
WWW: `http://www.lcc.uma.es/~machuca`

This work is partially funded by/Este trabajo está financiado por:

Consejería de Economía, Innovación, Ciencia y Empresa.
Junta de Andalucía (España)
Referencia: P07-TIC-03018

El Dr. D. Lorenzo Mandow Andaluz, Profesor Titular de Universidad, del Área de Ciencias de la Computación e Inteligencia Artificial de la Escuela Técnica Superior de Ingeniería Informática de la Universidad de Málaga,

Certifica que,

D. Enrique L. Machuca Sánchez, Ingeniero en Informática, ha realizado en el Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, bajo su dirección, el trabajo de investigación correspondiente a su Tesis Doctoral titulada:

An Analysis of Some Algorithms and Heuristics for Multiobjective Graph Search

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

Fdo.: Dr. Lorenzo Mandow Andaluz

Málaga, 24 de Julio de 2012

A mi padre,
al que debo
haber llegado aquí
Espero me vea
desde el cielo

Agradecimientos

Aunque sea la primera quizás en leer, esta es la última página que escribo, y con ella quiero agradecer de corazón su aprecio, su apoyo y su ayuda a cuantos han hecho posible que este trabajo de tesis doctoral sea una realidad.

En primer lugar, a mi director de tesis, Lawrence, que me ha guiado siempre y en cada momento hacia la meta, con más tenacidad de la que yo he sabido mostrar en muchos momentos. Ha sido mi maestro y mi mentor, pero también agradezco enormemente su gran humanidad, su comprensión y apoyo siempre ante situaciones personales. Ojalá haya adquirido en estos años todas las dotes y habilidades que tan amablemente ha sabido mostrarme para esta ardua tarea de ser científico y poder aportar mi granito de arena al conocimiento de la humanidad.

Por supuesto también a Pepe Luis, al que debo también mi aprendizaje gracias a sus siempre sabias críticas y consejos. Es una suerte haber disfrutado de la opinión y el apoyo de otro gran científico. Compañeros de viaje también han sido Amparo, Javier y mi compañero Francis, entre otros, con los que he podido compartir mis debates sobre NAMOA, la búsqueda heurística o la investigación en general. Es de bien nacido ser agradecido, así que no quiero dejar pasar la ocasión para dar las gracias a la Junta de Andalucía por concederme una beca de investigación (ref. P07-TIC-03018), y por ende a tod@s los andaluces, por hacer realidad este deseo de ser Doctor y llegar al máximo nivel de conocimiento en una disciplina científica.

No quiero olvidarme del Dr. Sanders, la Dra. Wagner y todas las personas que allí en Karlsruhe me acogieron y me ayudaron; también ha sido una interesante experiencia. Entre ellos, no puedo dejar de mencionar a mi querido amigo Alex, a quien siempre llevaré en el corazón. También he de dar las gracias a aquellos investigadores con los que he podido debatir o colaborar de alguna manera en estos años, como la Dra. Raith, el Dr. Iovanella, el Dr. Edelkamp o el Dr. Zaroliagis.

Pero aparte de trabajo, también han sido años de compartir experiencias muy positivas en lo personal, y no quisiera olvidarme de cuantas personas han sufrido mis conversaciones, mis bromas o mis emails en el laboratorio. No voy a nombrar a tod@s porque son much@s, y a cada cual los llevaré siempre en el recuerdo. Pero especialmente tengo que nombrar a Jaime, mi querido compañero de viaje, con el que comparto las penas y las alegrías de esta sufrida profesión, y al que me une, más que trabajo, una gran amistad.

He de recordar también que hasta aquí no habría llegado si no fuera por lo que aprendí de profesores de Universidad, como mi tutor de PFC Edu, de compañer@s y profesor@s de departamento, o de las grandes personas que me formaron en el colegio San José. También debo mi gratitud y mi recuerdo a aquellas personas que marcaron de alguna manera mi personalidad y con las que he compartido tantos momentos de mi vida, compañer@s de clase, de facultad, amig@s, herman@s de fe y familiares, que diré simplemente que son muchos y muy queridos para mí, para no menospreciar a unos y olvidar a otros.

Y por último, a Sarai por sufrir las consecuencias de ser científico en este mundo tan complicado que vivimos, y a mi familia, por darme lo que soy en esta vida, y por mostrarme el camino hacia Dios y hacia los hombres. De ellos recibí el amor, y aprendí a ser mejor persona, a tener unos valores y unos sentimientos que conforman en gran manera lo que soy.

A tod@s, en este momento tan importante de mi vida, de corazón, ¡¡GRACIAS!!

Enrique L. Machuca Sánchez
Málaga, 24 de Julio de 2012

Abstract

Many real problems require the examination of an exponential number of alternatives in order to find the best choice. They are the so-called combinatorial optimization problems. Besides, real problems usually involve the consideration of several conflicting magnitudes. When multiple *objectives* must be simultaneously optimized, there is generally not an optimal value satisfying the requirements for all the criteria at the same time. Solving these multiobjective combinatorial problems commonly results in a large set of Pareto-optimal solutions, which define the optimal tradeoffs between the objectives under consideration.

One of most recurrent multiobjective problems is considered in this thesis: the search for *shortest paths* in a graph, taking into account several objectives at the same time. Many practical applications of multiobjective search in different domains can be pointed out: routing in multimedia networks (Clímaco et al., 2003), satellite scheduling (Gabrel & Vanderpooten, 2002), transportation problems (Pallottino & Scutellà, 1998), routing in railway networks (Müller-Hannemann & Weihe, 2006), route planning in road maps (Jozefowiez et al., 2008), robot surveillance (delle Fave et al., 2009) or domain independent planning (Refanidis & Vlahavas, 2003).

Multiobjective route planning over realistic road maps has been considered as a potential application scenario for the multiobjective algorithms and heuristics considered in this thesis. Hazardous material transportation (Erkut et al., 2007), another related multiobjective routing problem, has also been considered as an interesting potential application scenario.

Single criterion shortest path methods are well known and have been widely studied. Heuristic Search allows the reduction of the space and time requirements of these methods, exploiting estimates of the actual distance to the goal. Multiobjective problems are much more complex than their single-objective counterparts, and require specific methods. These range from exact solution techniques to approximate ones, including the metaheuristic approximate methods usually found in the literature. This thesis is concerned with exact best-first algorithms, and particularly, with the use of heuristic information to improve their performance.

This thesis contributes both formal and empirical analysis of algorithms and heuristics for multiobjective search. The formal characterization of algorithms is important for the field. However, empirical evaluation is also of great importance for the real application of these methods. Several well known classes of problems have been used to test their performance, including some realistic scenarios as described above.

The results of this thesis provide a better understanding of which of the available methods are better in practical situations. Formal and empirical explanations of their behaviour are presented. Heuristic search is shown to reduce considerably space and time requirements in most situations. In particular, the first systematic results showing the advantages of the application of precalculated multiobjective heuristics are presented. The thesis also contributes an improved method for heuristic precalculation, and explores the convenience of more informed precalculated heuristics.

Contents

List of Figures

List of Tables

Part I

Motivation and Fundamentals

This part introduces the motivation, goals, and contributions of this thesis. The fundamentals of multiobjective graph search are presented, and previous relevant works are described in detail. In particular, the three algorithms analyzed in this thesis ($NAMOA^*$, MOA^* and TC) are described using a common framework. A review of previous benchmarks on multiobjective search is conducted, and a set of relevant problem classes and instances are identified for the empirical analysis performed in subsequent chapters.

- Chapter 1 gives an overview of the contents, goals and contributions of this thesis.
- Chapter 2 presents the problem and the algorithms that will be subject of analysis.
- Chapter 3 details the empirical methods and benchmarks used.

Chapter 1

Introduction

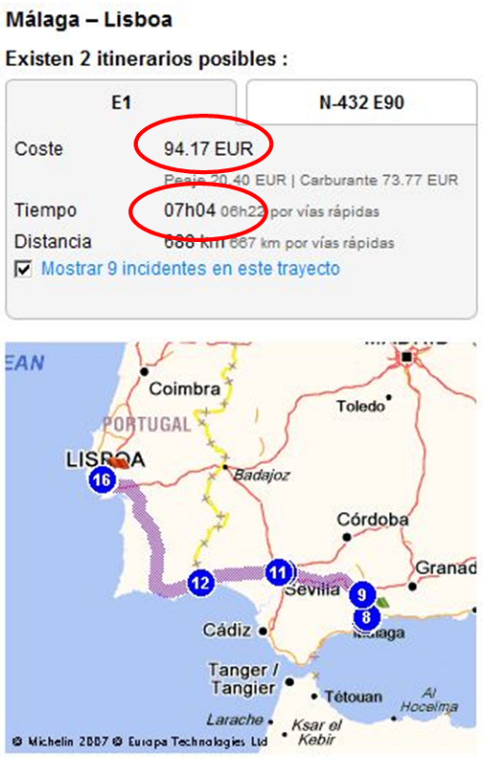
The context of this thesis is a common problem in the fields of Artificial Intelligence (AI) and Operational Research (OR). Shortest Path Problems appear in many everyday life situations, e.g. in car navigation systems, which plan the optimal route between some source and a specified destination point. Most available algorithms to solve this problem usually optimize a single criterion. The development and analysis of multiobjective shortest path algorithms is a relatively less explored area. The goal of this thesis is to deepen our undertanding of the formal and empirical behaviour of available exact multiobjective shortest path algorithms, with particular attention to heuristic search techniques. We also seek to analyze the benefits of heuristic search and to improve the performance of existing methods.

Section 1.1 introduces the motivation of this work. Scope and orientation are presented in section 1.2. The goals of this thesis are summarized in section 1.3. Contributions are summarized in section 1.4. Related publications derived from this research can be found in section 1.5. Finally, an outline of the structure of this thesis is presented in section 1.6.

1.1 Motivation and Significance

Artificial Intelligence (AI) is one of the main branches of Computer Science (CS). Since the establishment of AI as a formal discipline, many of the most difficult problems in CS have been its subject of study. One recurrent problem in the AI literature is the Shortest Path Problem (SP). A minimal cost route between two points in a network can be obtained by algorithms like the one devised by Dijkstra (1959). Heuristic Search (HS) is an AI subdiscipline aiming to obtain more efficient algorithms, exploiting specific problem knowledge. An important reference is the A^* algorithm (Hart et al., 1968), that uses cost estimates to guide search to the goal, improving efficiency.

Realistic decision problems frequently involve the consideration of multiple criteria at the same time. Figure 1.1 shows a sample scenario: planning a travel from Málaga to Lisbon seeking to minimize both time and economic cost. Among all feasible routes only a subset can be considered minimal in terms of both objectives. This is the so-called Pareto set. An alternative is Pareto-optimal if it cannot be improved in one objective without worsening some of the others, i.e. it represents an optimal tradeoff between the



Málaga – Lisboa

Existen 2 itinerarios posibles :

E1 | N-432 E90

Coste 94.17 EUR

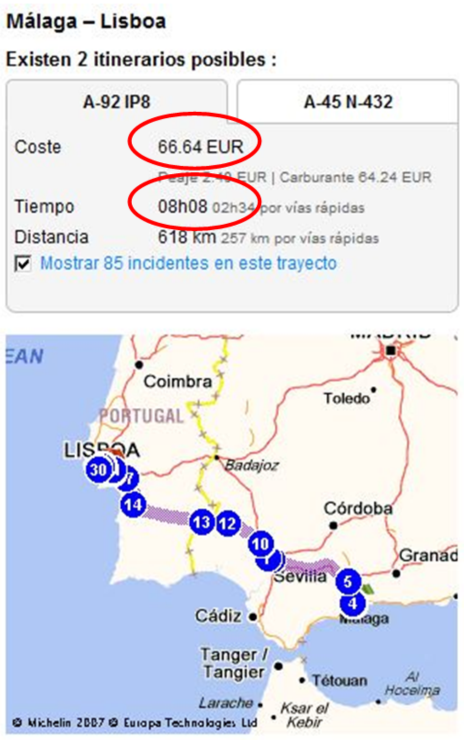
Peaje 20.40 EUR | Carburante 73.77 EUR

Tiempo 07h04 06h22 por vías rápidas

Distancia 688 km 667 km por vías rápidas

☑ Mostrar 9 incidentes en este trayecto

(a) Best route concerning time



Málaga – Lisboa

Existen 2 itinerarios posibles :

A-92 IP8 | A-45 N-432

Coste 66.64 EUR

Peaje 2.40 EUR | Carburante 64.24 EUR

Tiempo 08h08 02h34 por vías rápidas

Distancia 618 km 257 km por vías rápidas

☑ Mostrar 85 incidentes en este trayecto

(b) Best route concerning economic cost

Figure 1.1: Screenshots from a sample route planning application, showing alternative routes from Málaga to Lisbon.

objectives. The multiobjective search problem is known to be computationally more complex than single-objective search. Several methods and algorithms for this problem have been considered since the pioneering work of Hansen (1979).

In particular, several multiobjective heuristic search algorithms have been described in the literature, namely MOA^* (Stewart & White, 1991), $NAMOA^*$ (Mandow & Pérez de la Cruz, 2005) and TC (Tung & Chew, 1992) algorithms. However, their formal and empirical analysis is relatively unexplored when compared to analogous single-objective algorithms.

Recent formal analyses have begun to clarify the benefits of heuristic information in multiobjective search (Mandow & Pérez de la Cruz, 2010a). However, little experimental evaluation has been performed prior to this thesis to systematically compare the performance of multiobjective heuristic search algorithms.

The main goal of this thesis is to deepen our understanding of the performance of these algorithms. Research has been directed to complete formal analysis where needed, and to evaluate empirically the performance of heuristic search in general and realistic situations. In particular, the advantages of precalculated heuristics in multiobjective search are analyzed, and improved methods for heuristic precalculation are provided. The results obtained can be a useful guide for practitioners, and also point out clear lines of future research.

Finally, empirical evaluation is performed in different kinds of settings. Randomly generated problems provide an important testbed to control problem parameters, like solution depth or correlation between objectives. On the other hand, this thesis focuses on realistic route planning scenarios. These have been the subject of considerable research in the past few years for single-objective problems (Geisberger et al., 2008; Bast et al., 2007; Bauer & Delling, 2009; Goldberg & Harrelson, 2005), and emerge as an important potential area of application of multiobjective search.

1.2 Scope and Orientation

The boundaries of this doctoral dissertation are defined by the following terms:

Shortest path problems Problem instances involve the determination of shortest paths in graphs.

Additive costs Arcs are labelled with costs representing magnitudes to be minimized. The cost of a route between two nodes in a graph in terms of some magnitude can be obtained adding the corresponding magnitude costs of all arcs in the path.

Multicriteria scenarios Vector costs are considered in order to handle multiple objectives at the same time. In general, practical analyses consider two objectives, except for some simple scenarios where three objectives were considered.

Pareto optimality The scalar concept of minimum is no longer valid when multiple criteria are considered. Solutions must satisfy that no other feasible alternative can improve according to one objective whithout worsening at least one of the others.

Best-first exact algorithms All the algorithms and heuristics evaluated aim at the determination of the full Pareto-optimal set of solutions. Only best-first heuristic search is considered.

Empirical approach The research was guided mainly by practical experimentation with multiobjective heuristics and algorithms. Nevertheless, some theoretical results have been developed when necessary in order to explain the observed behaviour of algorithms, or complete previous formal analyses.

Tractable testbeds Even in simple biobjective problems, the number of Pareto-optimal solution costs can grow exponentially with graph size. However, under reasonable assumptions, the number of such solutions can be much smaller in practice (Müller-Hannemann & Weihe, 2006; Mandow & Pérez de la Cruz, 2009). In particular, when costs are discrete and bounded, and graph size grows polynomially with depth, the number of solutions can be shown to grow only polynomially in the worst case. Only such kind of problems are considered as testbeds for evaluation. Both randomly generated and realistic scenarios have been considered. Different problem factors can be gradually controlled in randomly generated problems. On the other hand, realistic scenarios evaluate more accurately the potential of multiobjective search in practical situations.

1.3 Research Goals

The main goals of this thesis can be summarized as follows:

Theoretical formalization One of the goals of this thesis is to complete the formal analysis of the heuristic performance of MOA^*, following the recent formal developments achieved for $NAMOA^*$. Additionally, some fundamental characterization of the TC heuristics is provided.

Empirical evaluation A second goal is to perform a systematic comparison of algorithms in order to determine which one performs better according to various problem parameters, like solution depth or correlation between objectives. Additionally, we evaluate the performance of multiobjective search in realistic route planning domains. In all cases, we seek a deep understanding of the causes of the observed behaviours.

Effectiveness of heuristic search A third goal is to establish under which conditions heuristic search can actually improve the performance of multiobjective best-first algorithms from a practical point of view.

Improvements on current techniques A final goal is to use the knowledge gained through the formal and empirical analysis to explore new ways to improve algorithm performance. Special attention is paid to certain alternatives in algorithm implementation, like the order of selection of alternatives for exploration.

1.4 Contributions of this Thesis

The main contributions of this thesis can be summarized as follows:

Analysis of MOA^* The thesis completes previous formal analyses of MOA^*. Previous analyses were unable to establish clearly the importance of heuristic informedness in algorithm performance. We show that the number of label expansions of MOA^* can be much larger with heuristic search than with blind search. In fact, performance can become worse even with the use of more informed consistent heuristics. This phenomenon is formally related to the node selection strategy used by MOA^*. In addition, empirical results show that this situation can easily appear in practice. As a result, MOA^* can be discarded in general as a suitable alternative for multiobjective heuristic search.

Analysis of TC A simple characterization is provided to show that the precalculated heuristic devised by Tung & Chew (1992) is consistent. This has been recently identified as an important formal property for multiobjective heuristic search (Mandow & Pérez de la Cruz, 2010a). Systematic empirical analyses show for the first time that this heuristic can improve considerably the performance of both TC and $NAMOA^*$ over blind search. However, TC was found to perform somewhat worse than $NAMOA^*$ in spite of the use of an additional heuristic for alternative selection. This phenomenon has been adequately explained: finding solutions early can penalyze the time requirements of heuristic search.

Analysis of $NAMOA^*$ The algorithm $NAMOA^*$ has been found the most effective approach in most situations. However, certain situations have been identified where heuristic search can represent an overhead in time performance. These are the cases where the reduction of the number of alternatives considered does not compensate for the time penalty of early solution determination inherent to heuristic search. More precisely, this can be traced to the number of dominance checks needed by the algorithm. Additionally, empirical evaluation suggests that a linear selection rule significantly improves the time performance of $NAMOA^*$ when compared to a traditional lexicographic one.

Precalculated heuristics The original precalculation method proposed by Tung & Chew (1992) can be improved in several ways. The original method requires the calculation of a one-to-all single-objective search for each objective under consideration. We have shown how the formal properties of $NAMOA^*$ let us bound the nodes that will be visited by $NAMOA^*$. This eliminates the need to consider in the precalculation stage those that will never be reached in the multiobjective search stage.

The original precalculated TC heuristic provides a single vector estimate for each node. $NAMOA^*$ accepts general heuristic functions $H(n)$ with multiple heuristic vector estimates. A new calculation method, called $KDLS$, is presented in this thesis. More informed, multiple vector, heuristic functions can be precalculated by $KDLS$. The precision of the new heuristic is determined by a parameter k. Larger values of k result in more informed heuristics, but at the same time, require more precalculation effort. In general, better heuristics considerably reduce the space requirements of $NAMOA^*$. However, time requirements are steadily increased in random grids, and only the smallest values of k are competitive with the original approach in this sense. Nevertheless, in route planning problems multivalued heuristics can offer savings in both space and time requirements for some instances.

Applications in realistic scenarios The application of $NAMOA^*$ with informed heuristics has been shown to be competitive with state-of-the-art approaches to multiobjective search. Several different testbeds have been considered for route planning in road maps, a potential application area. Combinations of objectives distance/time, and the more complex time/economic cost, have been solved exactly with available resources over large road maps.

1.5 Related Publications

Several of the contributions presented in this thesis have been already published in international peer-reviewed conferences and journals:

- **Journals:**

Machuca, E., Mandow, L., Pérez de la Cruz, J., & Ruiz-Sepulveda, A. (2012). A comparison of heuristic best-first algorithms for bicriterion shortest path problems. *European Journal of Operational Research*, 217(1), 44–53.

Machuca, E. & Mandow, L. (2012). Multiobjective heuristic search in road maps. *Expert Systems with Applications*, 39, 6435–6445.

- **Conferences:**

Machuca, E., Mandow, L., & Pérez de la Cruz, J. L. (2009). An evaluation of heuristic functions for bicriterion shortest path problems. In L. Seabra Lopes, N. Lau, P. Mariano, & L. Rocha (Eds.), *New Trends in Artificial Intelligence. Proceedings of EPIA'09* (pp. 205–216).: Universidade de Aveiro, Portugal.

Machuca, E., Mandow, L., Pérez de la Cruz, J. L., & Ruiz-Sepúlveda, A. (2010). An empirical comparison of some multiobjective graph search algorithms. In R. Dillmann, J. Beyerer, U. D. Hanebeck, & T. Schultz (Eds.), *Advances in Artificial Intelligence (Proceedings of KI'2010, 33rd Annual German Conference on AI, Karlsruhe, Germany, September 21-24)*, volume 6359 of *Lecture Notes in Computer Science* (pp. 238–245). Springer.

Machuca, E. & Mandow, L. (2011). Multiobjective route planning with precalculated heuristics. In L. Antunes, H. Pinto, R. Prada, & P. Trigo (Eds.), *Proc. of the 15th Portuguese Conference on Artificial Intelligence (EPIA 2011)* (pp. 98–107).

Machuca, E., Mandow, L., Pérez De La Cruz, J. L., & Iovanella, A. (2011). Heuristic multiobjective search for hazmat transportation problems. In J. Lozano, J. Gómez, & J. Moreno (Eds.), *Advances in Artificial Intelligence (Proceedings of the 14th international conference on Advances in Artificial Intelligence: Spanish Association for Artificial Intelligence - CAEPIA'11)*, volume 7023 of *Lecture Notes in Computer Science* (pp. 243–252). Berlin, Heidelberg: Springer-Verlag.

- **Doctoral Consortium:**

Machuca, E. (2009). Heuristics in best-first algorithms for multiobjective shortest path problems. In *Doctoral Consortium, XIII Conference of the Spanish Association for Artificial Intelligence (CAEPIA'09).*

Machuca, E. (2011). An analysis of multiobjective search algorithms and heuristics. In *Doctoral Consortium, Proceedings of 22nd International Joint Conference on Artificial Intelligence (IJCAI'11), Barcelona, 15-22 July* (pp. 2822–2823).

1.6 Outline

This thesis is structured in eight chapters, grouped in three parts. The first part comprises this introductory chapter, and chapters 2 and 3. Chapter 2 introduces the reader to the fundamentals of multiobjective optimization, reviews the different kinds of multiobjective search methods and presents the specific algorithms and heuristics analyzed in this thesis under a common framework. Chapter 3 introduces the reader to the task of empirical evaluation of algorithms. A literature review of previous relevant multiobjective benchmarks is presented. Finally, the different kinds of benchmark problems used in the empirical evaluations of this thesis are described in detail.

The second part groups all contributions of this thesis. Formal properties of MOA^* and the TC heuristic are established in chapter 4. A class of problems is devised to show that the complexity of MOA^* can become worse with perfectly informed heuristics when compared to blind search.

Chapter 5 performs a systematic evaluation and comparison of $NAMOA^*$, MOA^*, and TC. In the first place, the use of heuristic information is evaluated against blind search. Then, pair comparisons between the heuristic algorithms are performed. Some previously unnoticed phenomena are properly analyzed and explained. Moreover, the empirical results of MOA^* confirm the bad results expected from the theoretical analysis of chapter 4.

Chapter 6 exploits the result obtained in previous chapters to evaluate the performance of multiobjective search in the domain of route planning in road maps. A bounded procedure for the calculation of the TC heuristic is presented. General time/distance and time/economic cost problem instances are evaluated. Additionally, a set of hazardous material transportation problems is also considered.

Finally, chapter 7 explores the possibility of more informed precalculated heuristics, extending the precalculation process of the TC heuristic to multiple-vector heuristic estimates. The new heuristic is tested on a set of selected problem benchmarks.

Part three summarizes the main conclusions and future work in chapter 8.

Chapter 2

MultiObjective Graph Search: Problems and Algorithms

This chapter introduces the main concepts that will be used throughout the rest of this thesis. A definition of single-objective search problems and a brief review of relevant search algorithms is presented in section 2.1. A description of the multiobjective shortest path problem and basic multiobjective concepts follows in section 2.2. A review of the relevant blind algorithmic approaches for multiobjective search can be found in section 2.3. The heuristic algorithms analyzed in this thesis are described in section 2.4. Some formal properties about them and the heuristic functions used through most of the chapters of this thesis are also presented in this section. This includes the consideration of inconsistent heuristics in some detail. Finally, a review of relevant related heuristic works and a summary on multiobjective algorithms and heuristics is located at the end of the chapter.

2.1 Single-objective search

2.1.1 The Shortest Path Problem

The shortest path problem is probably one of the most studied problems by the Artificial Intelligence (AI) and Operational Research (OR) communities (Gallo & Pallottino, 1988; Cherkassky et al., 1996; Pearl, 1984). Many real problems can be modelled as finding the shortest path between two nodes in a graph. Let us see a formal description of the problem.

Let G be a locally finite labeled directed graph $G = (N, A, c)$ with $|N|$ nodes and $|A|$ arcs (n, n') labeled with positive values $\vec{c}(n, n') \in \mathbb{R}$, where $n, n' \in N$.

Definition 2.1 *A* ***path*** *P in (N, A) is a sequence $\langle n_1, n_2, \ldots, n_l \rangle$, where $n_i \in N, \forall i \in [1, l]$ and $(n_j, n_{j+1}) \in A, \forall j \in [1, l-1]$. The set of all possible paths in G is denoted by $\mathbb{P}$.*

Definition 2.2 *The* ***cost of a path*** *P is defined as the sum of the costs of its compo-*

nent arcs,

$$c(P) = \sum_{(n,n')\in P} c(n,n') \tag{2.1}$$

Definition 2.3 *Given a start node $s \in N$ and a set of goal nodes $\Gamma \subseteq N$, the **Shortest Path problem** (SP) consists of finding the path $P \in \mathbb{P}$ in G with the minimum cost $c(P)$ from s to a goal node $\gamma \in \Gamma$.*

2.1.2 Shortest path algorithms

Shortest path algorithms can be classified in terms of different parameters: best-first/depth-first, blind/heuristic or exact/approximate, among others. A complete description and a classification of different strategies has been recently devised by Korf (2010).

Best-first search is based in the principle of optimality stated by Bellman (1954). This principle applied to shortest path problems means that every optimal path must be composed by optimal subpaths. Thus, best-first algorithms achieve the optimal solution choosing at each step the best promising alternative. Whenever more than one path reaches the same node, the best one is preserved while the others are discarded or pruned. Best-first search is a general strategy for solving shortest path problems in graphs. The main drawback is that it can exhaust memory resources for many problems, as all promising alternatives must be kept in memory.

On the contrary, **depth-first search**(Korf, 1985) only considers the best next promising alternative, forgetting other feasible ones which can be in fact part of the optimal path. Thus, the algorithm must backtrack to find the least cost solution path. Depth-first search is generally applied when there are not enough space resources to solve the problem and is specially adequate for solving tree problems.

Among the huge variety of algorithms solving the Shortest Path Problem, this thesis deals with exact best-first search strategies for graphs. Dijkstra's algorithm (Dijkstra, 1959) is the most known blind algorithm in this sense for single-objective shortest path problems. However, the reference algorithm for shortest paths in AI is the A^* algorithm (Hart et al., 1968; Pearl, 1984) which can be seen as a generalization of Dijkstra's algorithm that uses heuristic cost estimates of the distance to goal(s) to improve search efficiency. The term *heuristic search* is used by the AI community for techniques that exploit knowledge about the problem in order to accelerate the solution of tough combinatorial problems.

A^* and Dijkstra-like algorithms work in a similar way: several feasible alternatives are evaluated at each step for the **selection** of the most promising one, according to a characteristic evaluation function for each node n. Let $g(n)$ denote the accrued cost of a path from s to n, while the heuristic function $h(n)$ an estimated cost of a solution from n to a goal node γ. For blind labelling strategies like Dijkstra's algorithm, the evaluation function is $f(n) = g(n)$, as $h(n) = 0$. Heuristic algorithms like A^* incorporate with $h(n)$ an estimate of the distance to a goal node, i.e. for A^* the evaluation function is $f(n) = g(n) + h(n)$ for each node n.

All these alternative costs from paths beginning in source node s are stored in a search structure (usually a search tree for single-objective search) within **labels**.

Definition 2.4 *A **label** stands for the cost of a path found to a given node. In single-objective algorithms each node n is labeled with a single-valued label, that stands for the cost of the current best known path to n.*

A reference to the ancestor(s) of the label can be included, in order to recover the solution path, once the algorithm finishes. At each step a label from some node n is selected for **expansion**, and all the possible extensions via outgoing arcs of node n are **generated** and compared with actual stored labels in successor nodes. The label stored for an adjacent node n' is updated when the extension of the selected path through some outgoing arc of n represent a least cost path to the node n'. Labels of dominated paths reaching a given node are discarded or **pruned** by virtue of the optimality principle (Bellman, 1954).

When a best-first strategy is used, the expanded label becomes permanent as the least cost to the node has been found. Otherwise, this is not guaranteed until all nodes have been examined, as the label can be corrected by a new better one. The former are known as **label-setting** algorithms (Dijkstra, 1959) and the later as **label-correcting** (Zhan & Noon, 2000).

This iterative node labelling process is repeated until the stopping criterion is fulfilled. The optimal cost of the path can be found in the label of the goal node when this is selected for expansion and the actual optimal path can be recovered tracing back pointers to parents.

The labels still pending evaluation are typically stored in an $OPEN$ queue. A distinct set of $CLOSED$ nodes can be used to store already evaluated alternatives, in order to perform duplicate detection and recover the solution path. In Dijkstra's algorithm, each time a label is expanded, as best-first search is used, the optimal shortest path distance from s to the node is found. In the case of A^*, the label will be permanent and a node will not come back to $OPEN$ queue under some assumptions related to the heuristic function $h(n)$.

2.1.2.1 Formal properties

Definition 2.5 *Let $h^*(n)$ be the actual optimal cost of a path from n to goal nodes $\gamma \in \Gamma$. A heuristic function $h(n)$ is **optimistic** when*

$$h(n) \leq h^*(n) \quad \forall n \in N \tag{2.2}$$

Definition 2.6 *Let $k(n, n')$ denote the cost of an optimal path in G from a node n to another node n'. A heuristic function $h(n)$ is **consistent** when*

$$h(n) + k(n, n') \leq h(n') \quad \forall n, n' \in N \tag{2.3}$$

Definition 2.7 *Equivalently, a heuristic function $h(n)$ is said to be **monotone** when*

$$h(n) + c(n, n') \leq h(n') \quad \forall (n, n') \in A \tag{2.4}$$

Definition 2.8 *A heuristic function $h_1(n)$ is said to be **more informed** than another heuristic function $h_2(n)$ when both are optimistic and*

$$h_1(n) > h_2(n) \quad \forall n \in N \wedge n \notin \Gamma \tag{2.5}$$

There is a strong relationship between the properties of $h(n)$ and the efficiency and quality of results produced by A^* (Pearl, 1984).

Property 2.1 (Admissibility) *When $h(n)$ is a lower bound (optimistic) the search is considered **admissible**, i.e. it is guaranteed to find an optimal solution if this solution exists. A^* is admissible even on infinite graphs with some additional assumptions:*

$$\begin{aligned} \forall n \in N,\ h(n) \geq 0 \\ \forall (n,n') \in A,\ c(n,n') \geq \epsilon > 0 \end{aligned} \tag{2.6}$$

Property 2.2 (Efficiency) *When $\forall n \in N,\ h(n) = 0$, A^* is equivalent to Dijkstra's algorithm. When $h(n)$ is **consistent** or **monotone**, A^* requires in the worst case $O(|N|)$ iterations, storing $O(|N|)$ nodes in memory. If the cost of the optimal solution is denoted by $c^* = k(s,\gamma)$, A^* will always expand for sure all labels with $f(n) < c^*$. For those with $f(n) = c^*$, only those belonging to the returned optimal solution path will be necessarily expanded. Given an optimistic heuristic function, more actual suboptimal alternatives can be pushed out the search frontier $f(n) = c^*$ with **more informed** heuristics, i.e. bigger values of $h(n)$, reducing search effort.*

Property 2.3 (Optimality) *When the heuristic function $h(n)$ is **monotone**, the cost of a path found by the algorithm is known to be optimal and it is not necessary to reopen nodes (an expanded node will not come back to $OPEN$). Moreover, A^* is proven to be **optimal** among the class of admissible best-first algorithms[1] (Dechter & Pearl, 1985), both in the number of expanded nodes and in the number of necessary iterations to find the solution. This means, that any expansion performed by the A^* algorithm must be also performed by another algorithm in this class to preserve admissibility. Otherwise, there is no guarantee to find the optimal solution in all cases.*

2.1.3 Application to route planning

Applied research in single-objective shortest paths includes the search for optimal routes in real road networks with Dijkstra's algorithm (Zhan & Noon, 1998), A^* (Zeng & Church, 2009) or even bidirectional A^* (Klunder & Post, 2006). Efficient speedup techniques have been recently devised for route planning in road maps (Schultes, 2008) or train timetables (Schulz et al., 2000; Schulz, 2005).

In general, speedup techniques exploit information gathered in previous extensive searches of the map. The challenge is to achieve fast shortest-path queries with practical preprocessing time and memory. Two main categories can be found:

- *Hierarchical techniques* exploit the structure of the problem to prune unimportant nodes (or arcs). The most fruitful hierarchical technique, namely *contraction hierarchies*, is based in node contraction (Geisberger et al., 2008): nodes are removed or *contracted* from the graph in some order while the shortest paths are preserved with additional arcs called *shortcuts*. However multilevel graphs (Schulz et al., 2002), routing based in transit-nodes (Bast et al., 2007) or customizable

[1]These are defined as the class of unidirectional search algorithms that begin on a start node s and are guided by path-dependent evaluation functions (Dechter & Pearl, 1985)

route planning (Delling et al., 2011a) based in advanced partitioning techniques of the graph like PUNCH (Delling et al., 2011b), have been also catalogued among other speedup techniques as effective for an improvement in the preprocessing of the graph or the time devoted to a query.

- *Goal directed techniques* compute distance bounds or make some preprocessing on arcs to exclude those arcs that do not belong to an optimal path. SHARC (Bauer & Delling, 2009) or ALT (Goldberg & Harrelson, 2005) belong to this category. While the former precomputes *arc-flags*, the later precalculates optimal distances to certain *landmarks* to provide distance bounds using the triangle inequality.

Both approaches, hierarchical and goal directed can be combined. A good overview of the possibilities is described by Bauer et al. (2010b). Moreover, some of the vast quantity of recent speedup techniques have been extended to other scenarios, like mobile or dynamic routing (Sanders et al., 2008), time-dependent planning (Batz et al., 2009) or multicriteria route planning (Geisberger et al., 2010; Delling & Wagner, 2009) for example.

The actual performance of speedup techniques depends on some decisions, taking into account the structural properties of the particular problem instance, e.g. the decision on which node is to be contracted in *contraction hierarchies*. A recent work established that the optimal adjustment for every instance in many recent techniques (for example, the assignment of landmarks to a graph in the ALT technique) is NP-hard (Bauer et al., 2010a). In practice, these adjustments are frequently settled experimentally with heuristics. Besides, most of these techniques are based in bidirectional search, in order to reduce time requirements.

A complete review of the literature on these techniques is out of the scope of this thesis. Good overviews of this field can be found elsewhere (Schultes, 2008; Delling et al., 2009; Bauer et al., 2010b).

Limited experiments have been performed on multiobjective route planning (Geisberger et al., 2010; Delling & Wagner, 2009). This will be used as one of the benchmark problems used in this thesis (see chapter 3).

2.2 MultiObjective Shortest Path Problems

The Multiobjective Search Problem (MSP) is an extension of the Shortest Path Problem where arcs are labelled with vector costs. Each component in a cost vector represents a different relevant attribute to be minimized, e.g. distance, time, risk or monetary cost. Bicriterion Shortest Path problems (BSP) are a particular class where only two attributes are considered. These problems rarely have a single optimal solution. Most frequently, a set of *nondominated* (Pareto-optimal) solutions can be found, each one presenting a particular trade-off between the objectives under consideration. Multiple techniques have been devised to solve multiobjective shortest path problems. Exact methods find the set of *all* Pareto-optimal paths to the problem. The methods described in this thesis fall into this category.

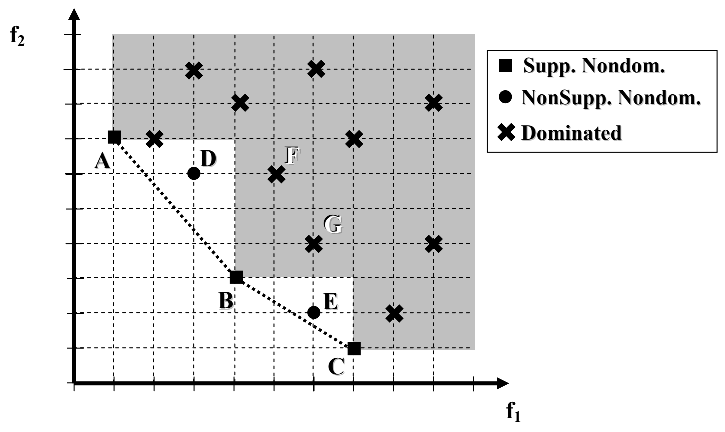


Figure 2.1: Supported, nonsupported and dominated points in a biobjective cost space

2.2.1 Basic Multiobjective Concepts

Multiobjective search algorithms differ from their scalar counterparts in several ways. First of all, the use of cost vectors in multiobjective problems induces only a partial order relation.

Definition 2.9 *Let us consider two q-dimensional vectors $\vec{v}, \vec{v}' \in \mathbb{R}^q$. A partial order relation $\prec$ denominated **dominance** is defined as follows,*

$$\forall \vec{v}, \vec{v}' \in \mathbb{R}^q, \quad \vec{v} \prec \vec{v}' \quad \Leftrightarrow \quad \forall i \ (1 \leq \ i \leq \ q), \quad v_i \ \leq \ v'_i \wedge \ \vec{v} \neq \vec{v}' \tag{2.7}$$

where v_i denotes the i-th component of vector $\vec{v}$. Likewise, the symbol $\preceq$ denotes the relation "dominates or equals".

Now, given two q-dimensional vectors $\vec{v}$ and $\vec{v}'$ (where $q > 1$), it is not always possible to say that one is better than the another. For example in a bidimensional cost space (see figure 2.1) a vector $(4,3)$ (point B) dominates $(5,6)$ and $(6,4)$ (points F and G), but no dominance relation exists between $(4,3)$ and $(1,7)$ or $(7,1)$ (points A and C). They are said to be *nondominated.*

Definition 2.10 *Given a set of vectors X, we shall define $nd(X)$ as the **set of non-dominated vectors** in X, i. e.,*

$$nd(X) = \{\vec{x} \in X \mid \nexists \vec{y} \in X, \quad \vec{y} \prec \vec{x}\} \tag{2.8}$$

In the example of figure 2.1 the set nd is $\{(1,7),(4,3),(7,1)\}$. Sometimes it becomes necessary to choose among nondominated vectors. Total orders can be defined in order to rank vectors.

Definition 2.11 *Let us consider two q-dimensional vectors $\vec{v}, \vec{v}' \in \mathbb{R}^q$. A total order relation $\prec_{lex}$ denominated **lexicographic order** is defined as follows,*

$$\vec{v} \prec_{lex} \vec{v}' \ \Leftrightarrow \ \exists j \ (1 \leq \ j \leq \ q), \ \ v_j < v'_j \ \ \wedge \ \forall i < j, \ \ v_i = v'_i \tag{2.9}$$

where v_i denotes the i-th component of vector $\vec{v}$.

Definition 2.12 *Let us consider two q-dimensional vectors $\vec{v}, \vec{v}' \in \mathbb{R}^q$. A total order relation $\prec_{lin}$ denominated **linear order** is defined as follows,*

$$\vec{v} \prec_{lin} \vec{v}' \;\Leftrightarrow\; \sum_i v_i \;<\; \sum_i v_i' \;\;,\; 1 \;\leq\; i \;\leq\; q \tag{2.10}$$

where v_i denotes the i-th component of vector $\vec{v}$.

Definition 2.13 *Let us consider two q-dimensional vectors $\vec{v}, \vec{v}' \in \mathbb{R}^q$. A total order relation $\prec_{wlin}$ denominated **weighted linear order** is defined as follows,*

$$\vec{v} \prec_{wlin} \vec{v}' \;\Leftrightarrow\; \sum_i \lambda_i v_i \;<\; \sum_i \lambda_i v_i' \;\;,\; 1 \;\leq\; i \;\leq\; q \tag{2.11}$$

where v_i, λ_i denote the i-th component of vectors $\vec{v}, \vec{\lambda}$.

A useful property of the lexicographic, linear or weighted linear order is that their optimum in a set of vectors is also a nondominated vector. The order relations defined by $\prec_{lex}$, $\prec_{lin}$ or $\prec_{wlin}$ are total orders. Now, given two q-dimensional vectors $\vec{v}$ and $\vec{v}'$ (where $q > 1$), it is possible to say that one is better than the another. For example, in a bidimensional cost space no dominance relation exists between $(2,3)$ and $(4,2)$. However, $(2,3) \prec_{lex} (4,2)$ and $(2,3) \prec_{lin} (4,2)$ (as $2+3 < 4+2$). Note that a derived total order can not always rank among two q-dimensional vectors, e.g. we can not say that $(2,3) \prec_{wlin} (4,2)$ for $\lambda = (1,2)$ (as $2*1+3*2 = 4*1+2*2$). Some additional tie-breaking rule must be used.

Definition 2.14 *Let X be the set of feasible solutions to a problem and let $f^k : X \to \mathbb{R}$ be k functions assigning a real value as image to a solution in X, being $k = 1, 2, \ldots, q$. A **multiobjective problem** in X can be formulated as a minimization problem,*

$$\min \vec{f}(x) = (f^1(x), f^2(x), \ldots, f^q(x)) \tag{2.12}$$
$$s.t.\ \vec{x} \in X$$

The criteria to be minimized[2] at the same time (called **objectives**) are usually conflicting so that there does not exist $\vec{x} \in X$ optimal for the q dimensions. Thus, a multiobjective problem has in general more than one **nondominated solution**, rather than a single optimal solution. These are also called **Pareto-optimal** or **efficient solutions** in the literature

Let X_E be the set of efficient or Pareto-optimal solutions to the minimization problem of (2.12), and F_E the set of nondominated image values on the objective space. Among the set X_E, we can distinguish two different kinds of nondominated solutions:

Supported These ones can be obtained as optimal solutions to a single-objective weighted sum problem (WSP). For instance, for the biobjective case (i.e. $q = 2$), where $\vec{x} = (x_1, x_2)$

$$\min_{x \in X} \lambda_1 x_1 + \lambda_2 x_2 \tag{2.13}$$

[2]Note that there is no loss of generality considering minimization as the maximization case can be transformed to a minimization problem.

for some $\vec{\lambda} = (\lambda_1, \lambda_2)$. The set of all supported solutions can be denoted by X_S, and the set of nondominated image values by F_S. For example, see figure 2.1, where a sample bidimensional image space is depicted. Points $A = (1, 7)$, $B = (4, 3)$ and $C = (7, 1)$ represent supported solutions. The *extreme* nondominated solutions are supported solutions that have the minimum possible value in at least one of the objectives (points A and C).

Non-Supported All the Pareto-optimal solutions in a continuous linear space $\mathbb{R}^q$ are supported. However, there can be remaining solutions, when dealing with discrete spaces (see figure 2.1). These remaining solutions in $X_{NS} = X_E \backslash X_S$ are called *non-supported* solutions as they cannot be obtained with linear combinations (WSPs). When solving a WSP, another supported solution will be found first, regardless of the slope used (i.e. regardless of the value of $\vec{\lambda}$). These solutions are located in the interior of triangles formed by two adjacent supported solutions, as depicted in figure 2.1. These areas are denominated by some authors as *duality gaps* (see the thesis of Raith (2009) for further details). The set of nondominated image values of X_{NS} is denoted by F_{NS}. These image values (points A, B and C) dominate shaded areas in the figure, but not points $D = (3, 6)$ or $E = (6, 2)$.

Definition 2.15 *Two feasible solutions $\vec{x}$ and $\vec{x}'$ are called* ***equivalent****, denoted by $\vec{x} =_{\prec} \vec{x}'$, if their image values are the same, i.e. $\vec{f}(\vec{x}) = \vec{f}(\vec{x}')$.*

Definition 2.16 *A* ***complete set*** *$X_C \subseteq X_E$ is a set of nondominated solutions whose image values, denoted by F_C form the minimal set of distinct nondominated $\vec{f}$ values, such that*

$$\forall \vec{x} \in X \backslash X_E, \quad \vec{x} \notin nd(X) \ \vee \ \exists \vec{x}' \in X_C, \ \vec{x} =_{\prec} \vec{x}' \tag{2.14}$$

For instance, the set $X_E = \{\vec{x}_1, \vec{x}_2, \vec{x}_3, \vec{x}_4\}$ denotes the set of *all* efficient or Pareto-optimal solutions to a multiobjective problem. The nondominated image set $F_E = \{(1, 2), (2, 1)\}$ has only two different vector values. A *complete set* $X_C = \{\vec{x}_1, \vec{x}_2\}$ could include all different nondominated image values as $F_C = F_E$. However, two additional *equivalent* solutions can be found in X_E, with $\vec{x}_1 =_{\prec} \vec{x}_3$ and $\vec{x}_2 =_{\prec} \vec{x}_4$. A different set $X_{NC} = \{\vec{x}_1, \vec{x}_3\}$ with nondominated image $F_{NC} = \{(1, 2)\}$ would not include all nondominated solution values.

2.2.2 The Multiobjective Search Problem

Let G be a locally finite labeled directed graph $G = (N, A, \vec{c})$ with $|N|$ nodes and $|A|$ arcs (n, n') where $n, n' \in N$, labeled with positive vectors $\vec{c}(n, n') \in \mathbb{R}^q$. The concept of path in multiobjective problems remains the same as for single-objective case (see definition 2.1). However, the definition of the cost of a path for the multiobjective case must take into account that arcs are labelled with vectors.

Definition 2.17 *The* ***cost of a path*** *P in multiobjective problems is a q-dimensional vector $\vec{C}_P$, and is defined as the sum of the costs of its arcs,*

$$\vec{C}_P = \vec{c}(P) = \sum_{(n,n') \in P} \vec{c}(n, n') \tag{2.15}$$

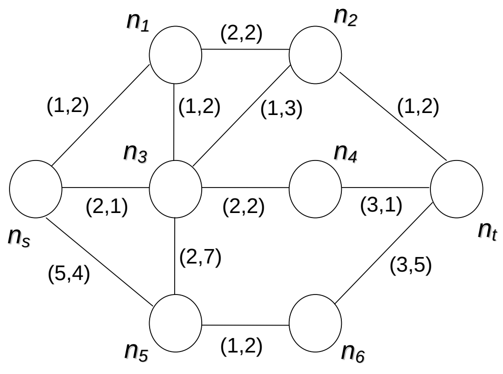


Figure 2.2: A sample graph with some nondominated solution paths and equivalent solutions

Let $\mathbb{P}$ be the set of all possible paths in G. The set of all possible costs $\vec{C}_P$ of paths in G will be denoted by $\mathbb{C}_{\mathbb{P}}$.

Definition 2.18 *Given a start node $s \in N$ and a set of goal nodes $\Gamma \subseteq N$, let $\mathbb{P}_{s\Gamma}$ be the set of all paths $P_{s\Gamma} \in \mathbb{P}$ joining s and nodes in Γ and let $\mathbb{C}_{s\Gamma}$ be the set of all possible costs $\vec{C}_{s\Gamma} \in \mathbb{C}_{\mathbb{P}}$ for paths in $\mathbb{P}_{s\Gamma}$. A multiobjective search problem (MSP) in G consists of finding the set of* ***all*** *nondominated paths $P_E \in \mathbb{P}_{s\Gamma}$ in G such that $\vec{c}(P_E) \in nd(\mathbb{C}_{s\Gamma})$.*

This means that we are looking for efficient paths P_E in G such that

1. go from source node s to a node in Γ, i.e. $P_E \in \mathbb{P}_{s\Gamma}$
2. their cost is nondominated with other costs from paths in $\mathbb{P}_{s\Gamma}$

The Multiobjective Shortest Path Problem is also a minimization problem and can be formulated in a similar way to (2.12), [3]

$$\begin{aligned} \min \vec{c}(P) = \vec{c}(\langle n_1, n_2, \dots, n_l \rangle) \\ s.t.\ P \in \mathbb{P}_{s\Gamma} \end{aligned} \tag{2.17}$$

The Bicriterion Shortest Path Problem (BSP) is a particular case of MSP in which $q = 2$, i. e., arc costs have two real components.

For graph search problems, the set X of feasible solutions to (2.13) is $\mathbb{P}_{s\Gamma}$ the subset of paths in $\mathbb{P}$ joining s and nodes in Γ the graph G, and the image function $\vec{f}$ to be minimized from (2.12) is the cost of each feasible path $P_{s\Gamma} \in \mathbb{P}_{s\Gamma}$, i.e $\vec{f}(x) = \vec{c}(P_{s\Gamma})$. For instance, see the path $\langle n_s, n_5, n_6, n_t \rangle$ from source node n_s to goal node n_t across

[3]It was denominated by Hansen (1979) for the bicriteria case, where arcs are labelled with costs $(c_1(n,n'), c_2(n,n'))$, as MINSUM-MINSUM because it tries to find

$$\min \sum_{(n,n') \in P} c_1(n,n') \ \wedge \ \min \sum_{(n,n') \in P} c_2(n,n') \tag{2.16}$$

nodes n_5 and n_6 in figure 2.2. Arcs are labelled with biobjective costs $\vec{c}(n,n') = (c_1(n,n'), c_2(n,n')) \in \mathbb{R}^2$ in the graph. The cost of this path is obtained as the sum for each component $c_i(n,n')$ of the arcs traversed, [4]

$$c(\langle n_s, n_5, n_6, n_t\rangle) = c(\langle n_s, n_5\rangle) + c(\langle n_5, n_6\rangle) + c(\langle n_6, n_t\rangle) =$$

$$= (5,4) + (1,2) + (3,5) = (8,15)$$

There can be multiple paths traversing the graph between a source node and a goal node with the same cost $\vec{c}(P)$, i.e. they are *equivalent solutions*. For instance, in the figure 2.2 the paths $P_1 = \langle n_s, n_1, n_2, n_t\rangle$ and $P_2 = \langle n_s, n_3, n_2, n_t\rangle$ have the same cost, i.e. $c(P_1) = c(P_2) = (4,6)$ or equivalently $P_1 =_{\prec} P_2$.

Notice also, that many different non-dominated paths may reach every node. For example, two different paths $\langle n_s, n_5\rangle$ and $\langle n_s, n_3, n_5\rangle$ reach n_5 with respective costs $(5,4)$ and $(4,8)$, each one nondominated.

In the sample graph of figure 2.2, besides the former paths P_1 and P_2, we can find another nondominated solution path $P_3 = \langle n_s, n_3, n_4, n_t\rangle$ with $c(P_3) = (7,4)$, while the paths $P_4 = \langle n_s, n_5, n_6, n_t\rangle$ or $P_5 = \langle n_s, n_3, n_5, n_6, n_t\rangle$ with respective costs $c(P_4) = (9,11)$ and $c(P_5) = (8,15)$ are not efficient solution paths as their costs are dominated, for example $c(P_3) \prec c(P_4)$ and $c(P_3) \prec c(P_5)$. A minimal complete set of solutions could be the set of paths $\{P_1, P_3\}$ or $\{P_2, P_3\}$ whose nondominated costs are the same, $\{(4,6),(7,4)\}$. In this thesis we consider only algorithms computing the whole set of alternatives, i.e. all Pareto-optimal vector costs of nondominated solution paths to the problem, e.g. the set of paths $\{P_1, P_2, P_3\}$ in the example.

Definition 2.18 deals with most usual MSPs, where the objective functions to be minimized are of additive type [5] like in (2.15). Other related problems can be found in the literature like bottleneck/minmax problems (de Lima Pinto et al., 2009; Iori et al., 2010) or the analogous maxmin(Martins, 1984c; Gandibleux et al., 2006) formulated for example as

$$\min c(P), \quad c(P) = \max_{(n,n')\in P} \{\vec{c}(n,n')\} \tag{2.18}$$

A third category includes methods dedicated to combinatorial problems with non-deterministic, fuzzy or imprecise values for arcs, which make use of multiobjective shortest path techniques. A classification of all types of multiobjective problems and techniques (including linear programming, metaheuristics or approximate approachs) is out the scope of the thesis. See (Clímaco & Pascoal, 2012; Tarapata, 2007; Ehrgott & Gandibleux, 2000, 2004; Ehrgott, 2005) for further details. Clímaco & Pascoal (2012) give an overview of existing techniques for solving the Multiobjective Shortest Path Problem and the Multicriteria Minimum Spanning Tree Problem. Tarapata (2007) recently completed an interesting survey on the different methods of solving MSP problems. Ehrgott & Gandibleux (2000) performed a survey on the more general class of

[4] Notice that the operator '+' denotes here the vectorial sum

[5] Applying logarithms, a reduction to (2.15) can be applied to multiplicative criteria

$$\vec{c}(P) = \prod_{(n,n')\in P} \vec{c}(n,n')$$

MOCO problems, and annotated relevant bibliography on the topic. More recently, the same authors (Ehrgott & Gandibleux, 2004) gave an overview on approximative solutions to MOCO problems. Ehrgott (2005) recently published a complete book about multicriteria optimization including linear programming techniques or a whole bunch of metaheuristic methods (e.g. ant colonies or genetic algorithms among others).

2.2.3 About complexity of the Multiobjective Search Problem

The MSP problem and the most popular BSP problem for the simplest case of two objectives, have been studied since Hansen (1979) proved that the problem is intractable. Hansen (1979) presented a family of sample graphs for the BSP case where the number of efficient solutions grows exponentially with size of the graph.

Theorem 2.1 *(Hansen, 1979, Theorem 1) In the worst case, the number of nondominated solution paths grows exponentially with solution depth (or equivalently with the number of nodes in the graph) even for the two-objective case.*

Moreover, Martins & Santos (1999) analyzed the behaviour respect of the number of *dominated* paths. They proposed a similar analysis for an analogous family of graphs, where all possible generated paths from source node s to a goal node γ are dominated, except for a single nondominated path.

Theorem 2.2 *(Martins & Santos, 1999, Section 5) In the worst case, the number of dominated paths grows exponentially with solution depth (or equivalently with the number of nodes in the graph) even for the two-objective case.* [6]

These results above are based in sample graphs showing the worst case scenario. A theoretical proof on the complexity of the problem was given by Serafini (1986), who transformed BSP into a Binary Knapsack Problem, identified before by Garey & Johnson (1979) as NP-complete.

Theorem 2.3 *(Serafini, 1986, Section 3) The Shortest Path Problem with nonnegative costs is NP-complete.*

However, not all multiobjective problems present this worst case difficulty (Müller-Hannemann & Weihe, 2006, section 1). In polynomial state spaces with bounded integer costs, this number is known to grow only polynomially with solution depth, although still exponentially with the number of objectives in the worst case (Mandow & Pérez de la Cruz, 2009, section 5). Besides these formal results, some complexity properties about the problem have been also presented,

Theorem 2.4 *(Martins & Santos, 1999, Theorem 6)The MSP problem is finite iff there are no negative cycles for any of the objective functions*

Theorem 2.5 *(Martins & Santos, 1999, Theorem 5) The MSP problem is bounded iff it is finite*

[6]They also point out that for some modified graphs, the label-setting algorithm even determine an exponential number of labels while no dominated labels would be determined with a label-correcting algorithm (see section 2.3.2 for explanations of both types of algorithms).

Theorem 2.6 *(Martins & Santos, 1999) If all cycles in (N, A) are non-negative for at least one of the objectives,*

$$\begin{aligned} \forall P_{cycle} \in \mathbb{P},\ \exists i \in [1 \dots q] \mid c_i(P_{cycle}) > 0 \\ P_{cycle} = \langle n_1, \dots, n_l \rangle \\ c_i(P_{cycle}) = \sum_{j \in [1, \dots l]} c_i(n_j) \end{aligned} \tag{2.19}$$

then every efficient path is a loopless nondominated path and viceversa

The absence of negative cycles is the fundamental condition for the finiteness and boundedness of the problem. Despite that, there is no easy way to identify the set of nondominated paths. However we can take advantage of the Principle of Optimality (Bellman, 1954) when choosing among potential nondominated alternatives. The nonexistence of cycles is also a fundamental condition for optimality,

Theorem 2.7 *(Martins & Santos, 1999, Theorem 6) The MSP problem satisfies the optimality principle for shortest path problems, if there are no negative cycles in (N, A).*

2.3 Blind MultiObjective Shortest Path Algorithms

Decision making is a cognitive process where a choice among several alternatives must be taken. Decision making involves the consideration of some criteria (usually conflicting) and the ranking of alternatives in terms of how attractive is the alternative for the agents responsible for taking decisions. These are called decision makers (DM) and the subfield of OR that considers multiple criteria in decision choices is called Multicriteria Decision Analysis (MDCA) or Multicriteria Decision Making (MDCM). There are several multicriteria decision approaches: besides the consideration of all possible alternatives, there are algorithms returning only best compromise solutions or some based in utility functions for example.

According to the taxonomy of Cohon (1978, chapter 5), and recently suggested also by (Clímaco & Pascoal, 2012), there are three types of multicriteria path algorithms:

1. *Multiobjective Analysis* include those for which there is no preference imposed about any criterion to the DM (approaches for which there is an *a posteriori* aggregation of preferences), i.e. these type of algorithms generate the whole set of nondominated solutions. Labelling, ranking and two-phases methods lie under this category

2. *Interactive approaches* are those for which there is a dialog of preferences with the DM[7].

3. A third category includes those for which there is an utility function with an *a priori* articulation of preferences in the criteria (e.g. in a weighted form). These algorithms can not obtain in general all nondominated solutions in discrete problems (see section 2.2.1 about nonsupported solutions)

[7]Regarding interactive approaches, there is dialog phase, but we refer only to the calculation phase with the term *interactive approach*

A complete and recent survey on the literature of the exact algorithms in the three categories can be found in (Clímaco & Pascoal, 2012). The algorithms described in this thesis fall in the first category: algorithms where the full set of alternatives is considered (a posteriori) using additive metrics and the full set of Pareto-optimal solutions is returned. The next section describes the types of methods under this category that have been applied to MSP problems.

2.3.1 A classification of blind MSP algorithms

Single-objective shortest path *labelling* algorithms are frequently classified into two groups: *label-setting* and *label-correcting.* The same two categories are valid for the multiobjective case. However, Skriver (2000) identifies two additional strategies in the literature: *ranking* approaches (or *K-th shortest paths*) and *two-phases* approaches. Recently Raith (2009); Clímaco & Pascoal (2012) suggested these four categories for BSP problems. Both Clímaco & Pascoal (2012) and Raith (2009) reference to annotated bibliography and give some sample code about each type of method. Three main type of algorithms can be found then in exact *a posteriori* approaches:

Labelling methods MSP labelling algorithms work in a similar way to single-objective counterparts. These algorithms run in an iterative way, and at each step select a *label* for expansion [8] and generate new successor labels until a stopping criterion is fulfilled. When this occurs, the labels at goal nodes represent optimal costs of paths from source to goal nodes. This category includes *label-setting* and *label-correcting.* These are further discussed in section 2.3.2.

Ranking methods These type of algorithms use a ranking method for listing paths by non-decreasing order. In MSP, one of the objective functions to be minimized has to be chosen as we have only a partial order, e.g. for the biobjective case, search begins with the lexicographic shortest path respect to the first criterion, and new paths are generated (and compared to existing solution paths) until the mimimal value for the second criterion is reached, i.e. *k-shortest paths* are generated between the two *extreme* solutions. These are further discussed in section 2.3.3.

Two-phase methods These algorithms work in two differentiated phases. In phase 1, the extreme supported efficient solutions are computed. In phase 2, the remaining non-supported efficient solutions are computed with any enumerative approach (e.g. labelling methods). Search with enumerative methods can be restricted to small areas between duality gaps. A full description of the method can be found in section 2.3.4

[8]Note that different notation is used by the different communities that have approached the shortest path problem. While in the Algorithmic community we can find the terms *settle a node* and *relax an edge* (e.g. (Bauer et al., 2010b)), the usual terms in Operations Research are *scan a label* or *merge/extend labels* (e.g. (Raith, 2009)) for the usual terms in Artificial Intelligence, *node selection* or *node expansion* (e.g. (Mandow & Pérez de la Cruz, 2010a)). The notation also changes vertices-edges ($G = (|V|, |E|)$) for the terms nodes-arcs ($G = (|N|, |A|)$), Labels ($L(i, j)$) or distances ($d(i, j)$) for accrued costs of paths ($g(n)$) or potentials ($\pi_{i,j}$) for heuristic estimates ($h(n)$). We will use in this thesis the Artificial Intelligence notation, that uses arcs and nodes, which are selected and expanded based in an evaluation function $f(n) = g(n) + h(n)$

Some other exact approaches can be found in the literature, like dynamic programming, branch-and-bound, linear programming and other mathematical approaches, or even some specifically designed algorithms. Some of them do not return all the nondominated solutions (e.g. only supported solutions), some of them return only approximate solutions or use a utility function. Only these three categories cited above are discussed here.

2.3.2 MSP Labelling algorithms

In general, labelling algorithms for the MSP problem are generalizations of the single-objective case. The main difference lies in the existence of more than one nondominated path to every node starting in node s, as more than one objective function implies that the optimal cost can not be the same for all of them. The central concept remains the same:

Definition 2.19 *A multiobjective* ***label*** *stands for the cost of a path found to a given node. In multiobjective search algorithms, each node n can be labeled with a set of vector-valued labels, which stand for the nondominated costs of paths currently found to n.*

Like in the single-objective case, there are two types of multiobjective labelling algorithms, *label-setting* and *label-correcting.* Three main issues characterize differences between both type of algorithms:

Label updating Label-setting algorithms follow a best-first selection policy. Therefore, they guarantee that any label selected for expansion is optimal. Label-correcting algorithms follow different selection policies (e.g. FIFO). Therefore, suboptimal labels may be selected for expansion. Later on, if better labels are found to the same node, the suboptimal labels are eliminated or *corrected* by the new better ones.

Convergence to optimal solution(s) Single-objective label-setting algorithms stop once the label of the goal node is selected. In multiobjective problems as soon as the set of selected goal labels dominates all other unexplored labels. Label-correcting algorithms need to explore the graph until no unexplored labels remain.

Successor(s) generation There are two basic selection policies: *label-selection* and *node-selection.* When only one label at a particular node is expanded, we have a *label-selection* method. On the contrary, if all labels found at that node are simultaneously expanded, a *node-selection* method is used. In any case, each path represented by a label is extended following all outgoing arcs of that node. Both in label-setting and in label-correcting, new labels generated must be compared to previous labels stored in the node. The dominance test and/or merge with existing labels is an expensive process of labelling algorithms and must be considered for a careful study in computational comparisons[9]. Obviously, node-selection in label-setting does not guarantee that all expanded labels are Pareto-optimal.

[9]For biobjective cases the dominance test can take advantage of the fact that labels can be totally ordered, reducing the number of necessary operations to determine whether a new label is nondomi-

For the original proposals of multiobjective labelling, see (Hansen, 1979; Martins, 1984b) for label-setting and (Vincke, 1974; Brumbaugh-Smith & Shier, 1989) for label-correcting. Several analyses have tried to determine the best alternative among blind labelling multiobjective algorithms (Skriver & Andersen, 2000; Guerriero & Musmanno, 2001; Paixão & Santos, 2007; Raith & Ehrgott, 2009; Raith, 2009). The results point out to label-correcting with node-selection as the best approach, but not for all instances of problems, where label-setting with label-selection performs better.

2.3.3 MSP Ranking algorithms

Among the first to present multiobjective ranking approaches, were Clímaco & Martins (1981, 1982). Single-objective ranking or k-best algorithms generates solution paths one after another with non-decreasing values for a minimization problem, until the k^{th} solutions is found. Two adaptations are necessary for multiobjective problems:

1. An order must be imposed about costs in a multiobjective space to rank solutions, either with a partial order (e.g. lexicographic order) or with a weighted sum problem. More details about the two approaches can be found on the thesis of Raith (2009, Section 1.3.1)

2. The ranking process can not be stopped in multiobjective problems until all efficient solutions are guaranteed to be found. The value of k is not clear a priori

Whenever the ranking algorithm returns a new solution, it must be compared to existing nondominated solutions, as in the labelling approaches. New solutions can be used to improve bounds on the weighted sum value used for ranking.

A general MSP algorithm was devised by Azevedo & Martins (1991), according to a lexicographic objective. For problems with more than two objectives, the dominance tests must take into account previously ranked solutions (Clímaco & Martins, 1981). The original proposal of this type of method is based on (Martins, 1984a). The ranking process depends on the number of nondominated solutions. In the worst case, an exponential number of paths are listed, although Müller-Hannemann & Weihe (2006) show that in practical applications it is not a problem. According to the classification proposed by Clímaco & Pascoal (2012), ranking methods can be classified into three categories:

Deletion algorithms After computing a shortest path, a new graph is built avoiding the construction of the former path. An iterative procedure lists paths by order of cost. See for instance (Martins et al., 2001)

Labelling algorithms If the graph does not contain negative cycles, the optimality principle can be used to rank paths, storing the k best labels. See for instance (Martins et al., 2000; Guerriero et al., 2001)

nated with existing labels, e.g. with a lexicographic order, we can rank solutions in order of the first objective and we know that a label is dominated if the value of the second objective exceeds that value for the best efficient solution for the first objective. However, for the general MSP case, we have to compare the cost of the label with all efficient paths obtained before. For instance, (3,3,3),(8,1,7),(6,4,2) are all nondominated and each one is minimal for one different objective. A total order like $\prec_{lex}$ can rank vectors using a fixed preference among objectives but it is not possible to obtain bounds on the other objectives like in biobjective case.

Deviation algorithms They are based on the idea that a suboptimal path has a common part with an efficient path plus a deviation arc. Based on the assumption that the part from the tail of a deviation arc up to terminal node t is also common, and the fact that we can obtain in advance shortest paths from all nodes to t, new candidate paths can be generated simply changing the deviation arc. See for instance (Martins et al., 2007)

A review of the literature and a detailed explanation on ranking approaches can be found in (Clímaco & Pascoal, 2012; Martins et al., 2007).

Several analyses have identified multiobjective ranking approaches as not competitive with label-setting and label-correcting methods, e.g. Huarng et al. (1996); Skriver & Andersen (2000); Raith & Ehrgott (2009). However, recent improvements (Paixão & Santos, 2008) have shown that multiobjective deviation algorithms can be significantly faster for some types of problems with few Pareto-optimal solutions.

2.3.4 MSP Two-Phase algorithms

The third alternative approach is the *Two-Phase Method*. It has been applied to BSP problems by Mote et al. (1991); Raith & Ehrgott (2009); Raith (2009). In phase 1, supported solutions can be easily computed with weighted sum problems (see (2.13)). The remaining efficient solutions can be obtained with the enumerative methods described above.

Supported efficient solutions can be quickly obtained in phase 1, as the derived single-objective problem can be solved with well known algorithms, faster than a multiobjective approach. Enumerative methods can be applied in phase 2 in an effective way, because we are restricting their search area to small triangles between supported solutions (see section 2.2.1). The information gathered in phase 1 can be exploited as well to derive bounds. The method may lead to obtain all nondominated solutions faster than a purely enumerative approach. Depending on the approach used in phase 1, a initialization phase can be necessary, to obtain extreme points of the Pareto front, e.g. in the biobjective case the lexicographic optimum regarding the first objective or the second objective.

A complete computational study has been recently presented by (Raith & Ehrgott, 2009; Raith, 2009), where all the known solution approaches for BSP are compared on different testsets, even realistic scenarios. Several alternatives are analyzed for each one of the two phases:

Initialization phase Single-objective label-setting, label-correcting and network simplex algorithms are implemented. FIFO approach seemed the most efficient for label-correcting while DIKBD (Dijkstra with double bucket) for label-setting.

Phase 1 Three different approaches are used: label-setting and label-correcting algorithms (the same from initialization phase) to solve WSP instances arising from a dichotomic approach, and a modified parametric network simplex approach for the multiobjective case.

Phase 2 Three different approaches are applied here: biobjective label-setting and label-correcting[10], and a near-shortest path adapted from Carlyle & Wood (2005) in lieu of ranking approaches[11]

They analyzed the best approach for each phase. The Two-Phase method was found superior than a pure label-setting or label-correcting only for some types of problems (actually with few Pareto-optimal solutions).

2.4 Heuristic MultiObjective Shortest Path Algorithms

This thesis is mainly concerned with heuristic search algorithms for multiobjective problems. Over the years, three different extensions of the label-setting (best-first) heuristic search algorithm A^* to the multiobjective case have been proposed: MOA^* (Stewart & White, 1991), $NAMOA^*$ (Mandow & Pérez de la Cruz, 2005) and TC algorithm (Tung & Chew, 1988, 1992).

Stewart & White (1991) proposed MOA^* (MultiObjective A^*), a direct extension of A^* to heuristic multiobjective search. The operation of the algorithm preserves similar node expansion principles. However, MOA^* does not share all the formal properties of A^*. Therefore, a new algorithm was recently devised (Mandow & Pérez de la Cruz, 2005), namely $NAMOA^*$ (New Approach to MultiObjective A^*), following a single path expansion policy, that preserves the good formal properties of A^*. The algorithm proposed by (Tung & Chew, 1988, 1992) (called hereafter TC) follows a similar strategy to $NAMOA^*$. The authors also proposed several multiobjective heuristic functions, discussed later.

Obviously, these are *best-first label-setting* algorithms. While $NAMOA^*$ and MOA^* accept both blind and heuristic search, TC algorithm was devised only for heuristic search. This section describes the three algorithms in detail, highlighting their differences.

2.4.1 The algorithm $NAMOA^*$

The pseudocode for $NAMOA^*$ is shown in table 2.1, slightly adapted from (Mandow & Pérez de la Cruz, 2010a). The algorithm uses a *label-selection* strategy. Analogously to A^*, it uses a list of $OPEN$ labels to control the search, and builds a search graph SG rooted at the start node s to record all the interesting paths found. New nodes and labelled arcs (pointers) are added to SG at step 5. At each iteration, SG contains the set of all nodes visited by the algorithm, and all nondominated paths found to such nodes with arcs reversed. With this information, given a node and one of its associated costs, it is possible to trace back the path(s) that reach the node from s with that particular cost.

[10]It is important to note that the original formulation from two phase method is modified in the sense that the label-correcting and label-setting algorithms for phase 2 are not run for every triangle as they detected that is quite inefficient. Thus, they run label-correcting or label-setting just once, but discarding labels that are not in any of the areas defined by two consecutive supported solutions. Bounds from phase 1 are also exploited

[11]The justification is made in the sense that "the cost of finding paths in order of their lengths is quite high"(Huarng et al., 1996; Skriver, 2000). Thus, a purely k-shortest path approach seems to be inefficient, and only paths with a maximal deviation from optimal path length are generated

We shall denote by $\vec{g}(P)$ the cost vector of each individual path $P \in \mathbb{P}_{s\Gamma}$ stored in the search graph. For each node n in SG, the sets $G_{cl}(n)$ and $G_{op}(n)$ denote the sets of nondominated cost vectors (labels) of paths reaching n that have and have not been explored yet respectively (i.e. closed and open). Each cost vector in these sets labels one or more arcs in the graph from n to its parents. If a vector is eliminated from this sets (pruned or filtered), so are its associated labelled pointers.

A set $H(n)$ of vector heuristic estimates $\vec{h}_n = \vec{h}(P_{n\gamma})$ is used for each node. These estimate cost vectors of paths from n to each goal node $\gamma \in \Gamma$. Therefore, for each path P_{sn} from s to n with cost $\vec{g}(P_{sn}) = \vec{g}_n$, there will be a set of heuristic evaluation vectors, $F(P_{sn})$. This function is the analogue in $NAMOA^*$ to $f(n)$ in A^*,

$$F(P_{sn}) = F(n, \vec{g}_n) = \text{nd}\{\vec{f}_P \mid \vec{f}_n = \vec{g}_n + \vec{h}_n \ \wedge \ \vec{h}_n \in H(n)\} \tag{2.20}$$

The set $H(n)$ is usually composed of a single element and hence also the set $F(n)$.

The $OPEN$ list consists of a set of tuples, or *partial solution paths*, that can be further explored. For each node n and each label $\vec{g} \in G_{op}(n)$, there is a corresponding tuple $(n, \vec{g}, F(n, \vec{g}))$ in $OPEN$. Therefore, each such tuple stands for a single label of node n. Initially, $(s, \vec{g}_s, F(s, \vec{g}_s))$ is the only tuple/label in $OPEN$, where s is the start node, and $\vec{g}_s = \vec{0}$. At each iteration, the algorithm considers in step 5 the expansion of a tuple from $OPEN$ with a nondominated vector value $\vec{f}_x \in F(x, \vec{g}_x)$ of some node x. This nondominated path is selected in step 3,

PATH SELECTION in heuristic $NAMOA^*$: Select for expansion an alternative $(n, \vec{g}_n, \{\vec{f}\})$ from $OPEN$ such that $\nexists (n', \vec{g}_{n'}, \{\vec{f}'\}) \in OPEN \quad \text{with } \vec{f}' \prec \vec{f}$

This rule is also valid for blind search, assuming that $\vec{h}_x = \vec{0}$ for all $\vec{g}_x \in G_{op}(x)$, but it could be stated also as,

PATH SELECTION in blind $NAMOA^*$: Select for expansion an alternative $(n, \vec{g}_n, \{\vec{g}\})$ from $OPEN$ such that $\nexists (n', \vec{g}_{n'}, \{\vec{g}'\}) \in OPEN \quad \text{with } \vec{g}' \prec \vec{g}$

When several paths reach the same node, dominated ones are discarded. This process (step 5(c)i) is called *pruning* :

- New paths found to a known node n are *pruned* if their cost is dominated by some vector in $G_{op}(n)$ or $G_{cl}(n)$.
- If the new path is not dominated, it is included in $G_{op}(n)$, *pruning* vectors in $G_{op}(n)$ and $G_{cl}(n)$ dominated by the new path's cost [12].

Once a solution is found its cost is kept in a set called $COSTS$, with all nondominated solution costs found to the moment. Each new alternative selected for expansion is checked against $COSTS$. If the estimated cost of an alternative is found to be dominated by the cost of a solution (step 4), then it can be safely discarded (this operation

[12]If the heuristic function $H(n)$ satisfies the so-called *consistency* property, then all labels expanded by $NAMOA^*$ are known to be nondominated. In such cases, all labels in the $G_{cl}(n)$ sets are *permanent* and can never be pruned (see Mandow & Pérez de la Cruz (2010a) for details). All heuristics considered in the following sections are consistent.

```
1. CREATE:
   —An empty search graph $SG$, and place $s$ as its root.
   —Two empty sets $G_{cl}(s)$ and $G_{op}(s)$, and insert $\vec{g}_s = \vec{0}$ in $G_{op}(s)$.
   —A list of alternatives, $OPEN = \{(s, \vec{g}_s, F(s, \vec{g}_s))\ \}$.
   —Two empty sets, $GOALN$, $COSTS$.

2. CHECK TERMINATION.
   — If $OPEN$ is empty, then backtrack in $SG$ from the nodes in $GOALN$ and
   return the set of solution paths with costs in $COSTS$.

3. PATH SELECTION.
   — Select an alternative $(n, \vec{g}_n, F_n)$ from $OPEN$ with $\vec{f}_n \in F_n$ nondominated in
   $OPEN$, i.e. for all $(n', \vec{g}_{n'}, F'_n) \in \mathrm{O}PEN$ it does not exist $\vec{f}'_n \in F'_n$ such that
   $\vec{f}'_n \prec \vec{f}_n$.
   — Delete $(n, \vec{g}_n, F_n)$ from $OPEN$, and move $\vec{g}_n$ from $G_{op}(n)$ to $G_{cl}(n)$.

4. SOLUTION RECORDING. If $n$ is a goal node, then
   —Include $n$ in $GOALN$ and $\vec{g}_n$ in $COSTS$.
   —For all alternatives $(x, g_x, F_x)$ in $OPEN$, eliminate from $F_x$ all vectors domi-
   nated by $\vec{g}_n$ (FILTERING) .
   —Eliminate from $OPEN$ all alternatives $(x, \vec{g}_x, F_x)$ such that $F_x$ is empty.
   —Go back to step 2.

5. PATH EXPANSION: If $n$ is not a goal node, then for all successors nodes $m$ of
   $n$ that do not produce cycles in $SG$ do:
   (a) Calculate the cost of the new path found to $m$: $\vec{g}_m = \vec{g}_n + \vec{c}(n, m)$.
   (b) If $m$ is a new node (i.e. is not already in $SG$)
       i. Calculate $F_m = F(m, \vec{g}_m)$ FILTERING estimates dominated by
          $COSTS$.
       ii. If $F_m$ is not empty, put $(m, \vec{g}_m, F_m)$ in $OPEN$, put $m$ in $SG$ with a
           pointer to $n$ labelled by $\vec{g}_m$, and create $G_{op}(m) = \{\vec{g}_m\}$ and $G_{cl}(m) = \emptyset$.
       iii. Go to step 2.
   (c) else ($m$ is not a new node),
       —If $\vec{g}_m \in G_{op}(m)$ or $\vec{g}_m \in G_{cl}(m)$: create a new pointer in $SG$ from $m$ to $n$
       labelled with $\vec{g}_m$, and go to step 2.
       —If $\vec{g}_m$ is nondominated by any cost vectors in $G_{op}(m) \cup G_{cl}(m)$ (a path to
       $m$ with new cost has been found), then:
       i. PRUNE from $G_{op}(m)$ and $G_{cl}(m)$ vectors dominated by $\vec{g}_m$.
       ii. Calculate $F_m = F(m, \vec{g}_m)$ FILTERING estimates dominated by
           $COSTS$.
       iii. If $F_m$ is empty eliminate stored alternatives $(m, \vec{g}_m, F_m)$ for node m
            from $OPEN$
       iv. If $F_m$ is not empty, put $(m, \vec{g}_m, F_m)$ in $OPEN$, include $\vec{g}_m$ in $G_{op}(m)$,
           and create a new pointer in $SG$ from $m$ to $n$ labelled by $\vec{g}_m$.
       v. Go to step 2.
       —Otherwise: go to step 2.
```

Table 2.1: The $NAMOA^*$ algorithm.

is called *filtering*). The set $COSTS$ is also used to *filter* new dominated alternatives found to a node (steps 5(b)i, 5(c)ii) when generating succesors.

Each time a new goal node is found, it is included in the set $GOALN$. Once the $OPEN$ list becomes empty, the set of all nondominated solution paths can be recovered from the information gathered in the search graph SG, and the sets $COSTS$ and $GOALN$ (step 2).

2.4.2 The algorithm MOA^*

A pseudocode for MOA^* slightly adapted from the original description is shown in table 2.2. Some similarities with $NAMOA^*$ have been highlighted with capitals. An important difference between MOA^* and $NAMOA^*$ is that the former follows a *node-selection* strategy. A single set $G(n)$ stores all nondominated labels found to any node n. Therefore, the set $F(n)$ is defined as,

$$F(n) = nd\{\vec{f}_n \mid \vec{f}_n = \vec{g}_n + \vec{h}_n \ \wedge \vec{g}_n \in G(n) \wedge \vec{h}_n \in H(n)\} \tag{2.21}$$

Multiple paths with the same cost can reach a node. Thus, multiple backpointers to ancestors are needed for each label. A different $LABEL(m, n)$ set is used for each ancestor n of node m. Let $ANCS(m)$ be the set of ancestors of node m. Then, for succesor node m

$$G(m) = nd(\{\cup LABEL(m, n),\ \forall n \in ANCS(m)\}) \tag{2.22}$$

The $OPEN$ set is a list of nodes. At each iteration, an open node n with a vector in $F(n)$ that is not dominated by any vector in the $F(n')$ set of other nodes n' in $OPEN$ is selected for expansion (step 6) and the node is moved to the set $CLOSED$. This node is selected in step 3,

NODE SELECTION in heuristic MOA^*: Select for expansion an alternative (node) $(n, G(n), F(n))$ from $OPEN$ having a vector value $\vec{f}$ such that $\nexists (n', G(n'), F(n')) \in OPEN$ with a $f' \in F(n') \ \mid \ \vec{f'} \prec \vec{f}$

This rule is also valid for blind search as in the case of $NAMOA^*$, assuming that $\vec{h}_x = \vec{0}$ for all $\vec{g}_x \in G(x)$, but it could be stated also as,

NODE SELECTION in blind MOA^*: Select for expansion an alternative (node) $(n, G(n), F(n))$ from $OPEN$ having a vector value $\vec{g}$ such that $\nexists (n', G(n'), G(n')) \in OPEN$ with a $g' \in G(n') \ \mid \ \vec{g'} \prec \vec{g}$

In MOA^* all paths (labels) reaching a node n are expanded simultaneously once it is selected. Therefore, all labels found to a single node at a given time are either simultaneously open or closed. When new nondominated labels are found to a closed node m and included in $G(m)$, the node is put back into $OPEN$, with all its associated labels.

Additional differences with $NAMOA^*$ must be pointed out. The *filtering* and *pruning* processes are slightly different,

```
1. CREATE:
   —A list of alternatives, $OPEN$, with the start node $s$.
   —An empty set, $CLOSED$.
   —Two empty sets, $GOALN$, $COSTS$.
2. CALCULATE the set $ND$ of nodes $n$ in $OPEN$ such that at least one estimate
   $\vec{f} \in F(n)$ is not dominated by the estimates of other open nodes or by any
   solution cost of $COSTS$ (“FILTERING”).
3. CHECK TERMINATION. If $ND$ is empty, then
   —Terminate returning the set of solution paths that reach nodes in $SOLN$ with
   costs in $COSTS$.
   else NODE SELECTION
   —Choose a node $n$ from $ND$ using a domain-specific heuristic, breaking ties in
   favour of goal nodes, and move $n$ from $OPEN$ to $CLOSED$.
4. Do bookkeeping to maintain accrued costs and node selection function values.
5. SOLUTION RECORDING. If $n$ is solution node, then
   —Include $n$ in $GOALN$ and its current costs into $COSTS$.
   —Remove dominated costs from $COSTS$.
   —Go back to step 2.
6. NODE EXPANSION. If $n$ is not solution node, expand $n$ and examine its suc-
   cessors. For all successors nodes $m$ of $n$ do:
   (a) If $m$ is a newly generated node, then
         i. Establish a pointer from $m$ to $n$.
        ii. Set $G(m) = LABEL(m, n)$.
       iii. Compute $F(m)$.
        iv. Add $m$ to $OPEN$.
   (b) Otherwise, $m$ is not new, so do the following,
         i. If any potentially nondominated paths to $m$ have been discovered, then,
            for each one, do the following.
            —Ensure that its cost is in $LABEL(m, n)$, and therefore in $G(m)$.
            —If a new cost was added to $G(m)$ then, PRUNE from $LABEL(m, n)$
            dominated costs, and if $m$ was in $CLOSED$, then move it to $OPEN$.
            —Recompute $F(m)$.
7. Go back to step 2.
```

Table 2.2: The MOA^* algorithm.

- Each time a new nondominated path (label) is found to a node n, MOA^* does not clearly specify that possibly dominated labels should be removed [13] (*pruned*) from $G(n)$ (step 6(b)i).
- In any case, the dominated labels are not removed from $OPEN$ (step 6(b)i). Moreover, in a later step of execution (step 2), the set ND must be calculated because dominated alternatives are not really *filtered* but kept in $OPEN$ list among other nondominated ones.
- The set $COSTS$ is not either adequately used by MOA^* to *filter* unpromising labels (step 6(a)iii) like in $NAMOA^*$ (steps 5(b)i,5(c)ii) . Moreover, the recalculation of $F(m)$ was not indicated in the original description of MOA^* (step 6(b)i).

2.4.3 The algorithm TC

A pseudocode for TC slightly adapted from the original description is shown in table 2.3. Some similarities with $NAMOA^*$ have been highlighted with capitals. TC employs a *label-selection* strategy. Therefore, separate $G_{op}(n)$ and $G_{cl}(n)$ sets of labels are considered[14] like in $NAMOA^*$. The $OPEN$ list consists of all tuples $(n, \vec{g}_n)$ such that $\vec{g}_n \in G_{op}(n)$. These are ranked for selection according to a scalar value

$$f_n = f(n, \vec{g}_n) = \sum_i g_i(P_{sn}) + h_{mix}(n) \tag{2.23}$$

where $g_i(P_{sn})$ is the *i-th* component in $\vec{g}_n$. The scalar heuristic values $h_{mix}(n)$ are precalculated prior to the execution of TC. The precalculation procedure is explained in the section 2.4.4. Notice that, since $h_{mix}(n)$ is not used to discard alternatives, it does not influence the number of paths considered, only the order in which they are selected for expansion.

At each iteration, the algorithm considers in step 5 the expansion of a tuple from $OPEN$ with a nondominated vector value $\vec{f}_x \in F(x, \vec{g}_x)$ of some node x. This nondominated path is selected in step 3 according to the associated scalar value f_x described in (2.23),

PATH SELECTION in heuristic TC: Select for expansion an alternative (scalar) $(n, \vec{g}_n, \{f\})$ from $OPEN$ such that $\nexists (n', \vec{g}_{n'}, \{f'\}) \in OPEN$ with $f' < f$, i.e. with $h_{mix}(n') + \sum_i g_i(P_{sn'}) < h_{mix}(n) + \sum_i g_i(P_{sn})$

TC uses vectorial heuristic estimates $\vec{h}_{TC}(n)$ to calculate the associated vectorial $\vec{f}_n$ values, which are used to discard (*filter*) alternatives. These vector estimates are precalculated also prior to execution of the TC algorithm, according to the procedure explained in section 2.4.4. In TC algorithm, the evaluation function used for *filtering* for each label $\vec{g}_n \in G_{op}(n)$ is just a single vector defined as

$$\vec{f}_n = \vec{f}(n, \vec{g}_n) = \vec{g}_n + \vec{h}_{TC}(n) \tag{2.24}$$

[13]The dominated labels are removed from $LABEL$ set but the removal from $G(n)$ is ambiguously described in MOA^*

[14]In the TC algorithm $G_{op}(n)$ is called $L(n)$, and $G_{cl}(n)$ is referred to as 'permanent labels'

1. CALCULATE $\vec{h}_{TC}$, h_{mix} for each node n, applying Dijkstra's algorithm on a graph with arcs reversed.

2. CREATE:

   —Two empty sets $G_{cl}(s)$ and $G_{op}(s)$, and insert $\vec{g}_s = \vec{0}$ in $G_{op}(s)$ with $F(s, \vec{g}_s)) = \{h_{mix}(s)\}$.

   —A list of alternatives, $OPEN = \{(s, \vec{g}_s, F(s, \vec{g}_s))\ \}$.

   —An empty set $COSTS$.

3. CHECK TERMINATION. If $OPEN$ is empty, then

   —Terminate the execution

   else PATH SELECTION.

   — Select an alternative $(n, \vec{g}_n, F_n)$ from $OPEN$ with $f_n \in F_n$ minimal in $OPEN$, i.e. for all $(n', \vec{g}_{n'}, F'_n) \in$ OPEN it does not exist $f'_n \in F'_n$ such that $f'_n < f_n$, where $\forall x, f_x = \sum_i g_i(P_{sx}) + h_{mix}(x)$. Ties are broken arbitrarily.

   — Delete $(n, \vec{g}_n, F_n)$ from $OPEN$, and move $\vec{g}_n$ from $G_{op}(n)$ to $G_{cl}(n)$.

   — If n is a goal node, go to step 6. Otherwise, go to step 4

4. SUCCESSORS GENERATION. If n is not a goal node, then for all successors nodes m of n that do not produce cycles in SG do:

   — FILTER alternatives from a node m dominated by $COSTS$, i.e. those for which exists some $\vec{g}_{n'} \in COSTS$ such that $\vec{f}_m \prec \vec{g}_{n'}$, where $\vec{f}_m = \vec{g}_m + \vec{c}(n, m) + \vec{h}_{TC}(m)$.

   —If all alternatives are filtered go to step 3. Otherwise, go to step 5

5. PATH EXPANSION: If n is not a goal node, then for all successor nodes m of n not filtered in step 4 do:

   — Calculate the cost of the new path found to m: $\vec{g}_m = \vec{g}_n + \vec{c}(n, m)$.

   — Calculate $F_m = F(m, \vec{g}_m) = \{f_m\}$, where a single scalar value f_m is calculated for each g_m as $f_m = f(m, \vec{g}_m) = \sum_i g_i(P_{sm}) + h_{mix}(m)$

   — Create a new pointer from m to n labelled by $\vec{g}_m$.

   — Put $(m, \vec{g}_m, F_m)$ in $OPEN$, and create $G_{op}(m) = \{\vec{g}_m\}$ and $G_{cl}(m) = \emptyset$.

   — Go to step 3

6. SOLUTION RECORDING. If n is a goal node, then

   —Include $\vec{g}_n$ if it is nondominated in $COSTS$, i.e. for all $\vec{g}_{n'} \in COSTS$ it does not exist such $\vec{g}_{n'}$ that $\vec{g}_{n'} \prec \vec{g}_n$.

   — Trace back from $(n, \vec{g}_n, F_n)$ and return the solution path

   — Go to step 3

Table 2.3: The TC algorithm.

Therefore, for each path P_{sn} from s to n with cost $\vec{g}(P_{sn}) = \vec{g}_n$, there will be a set $F_1(P_{sn})$ with a single scalar value f_n calculated as described in (2.23), used for *selection* from $OPEN$ queue, analogue to $NAMOA^*$, but also a different set $F_2(P_{sn})$ with a single vector value $\vec{f_n}$ calculated as described in (2.24), used only for *filtering*, as $\vec{h}_{TC}$ is used only in this sense.

TC presents two additional differences with $NAMOA^*$,

- Regarding *filtering*, TC discards all newly generated labels whose $\vec{f}(n, \vec{g}_n)$ value is dominated by the current vectors in $COSTS$ (step 4). However, when a new solution cost $\vec{c}^{\,*} = \vec{g}_\gamma$ is discovered, it is not used to *filter* dominated labels already in SG neither in $OPEN$ (step 6).

- Regarding *pruning*, each time a new nondominated path (label) is found to a node n, the algorithm TC does not specify that possibly dominated labels in $G_{op}(n)$ should be removed (step 5) .

Note also that the original description of TC algorithm only considers the *complete set* of Pareto-optimal solutions and traces back the solution path individually for each solution found. However, in this thesis a version of TC returning the whole Pareto set when the algorithm finishes is considered, analogously to MOA^* and $NAMOA^*$.

2.4.4 Heuristic functions

Heuristic functions do not appear explicitly in the pseudocode in tables 2.1, 2.2, 2.3. However, there are several points where heuristic functions are implicitly invoked:

- PATH or NODE SELECTION: Step 3 in $NAMOA^*$, step 2 in MOA^* or step 3 in TC. Usually the node (path) selected from *OPEN* is nondominated according to certain heuristic function.

- SOLUTION RECORDING (filtering): step 4 in $NAMOA^*$. All alternatives $(x, \vec{g}_x, F_x)$, such that all vectors in F_x are dominated by the cost $\vec{g}_n$ of a newly found solution, are eliminated from *OPEN*. These elements $\vec{f_x} \in F_x$ are calculated according to certain heuristic function, as $\vec{f_x} = \vec{g}_x + \vec{h}_x$. Thus, the heuristic function also plays a role at this step.

- PATH EXPANSION (filtering): Step 5(b)i and 5(c)ii in $NAMOA^*$, or step 4 in TC. These are new occurrences of filtering when performing the step PATH EXPANSION in $NAMOA^*$ or the step SUCCESSORS GENERATION in TC. Filtering of new *dominated alternatives* depends on the vectors $\vec{f_x} \in F_x$ and hence on the values of the heuristic function [15].

When no heuristic information is available, the trivial heuristic function is given by $\vec{h}_0(n) = \vec{0}$, for all nodes n. Both MOA^* and $NAMOA^*$ can be used for blind search with this heuristic, while TC was devised only for heuristic search[16].

[15] Notice that the original *algorithm* proposed by Tung & Chew (1992) does not perform all the filtering operations performed by $NAMOA^*$, or performs them in a different way (see section 2.4.3)

[16] A "blind" version of TC algorithm will be further introduced when needed in chapter 5

Tung & Chew (1992) proposed two nontrivial heuristic functions that are well defined for every multiobjective path problem. The first heuristic [17] is defined as, $\vec{h}_{TC}(n) = (c_1^*(n), c_2^*(n), \dots c_q^*(n))$, where $c_i^*(n)$ is the optimal scalar cost of a path from n to the goal, considering only the *i-th* cost component. These values are precalculated for each component by reversing all arcs in the graph, originally labelled with vector cost $\vec{c}(n, n') = (c_1, c_2, \dots c_q)$, labelling them with scalar costs $c_i(n', n) = c_i$ and applying Dijkstra's algorithm to find the shortest path from the goal node to all other nodes in the reversed graph, once for each scalar cost. Ties are broken by a lexicographic order, taking into account the other cost components.

The second heuristic [18] proposed by Tung & Chew (1992) is called in this thesis h_{mix} and is defined in a similar way. Each arc (n, n') in the original graph, labelled with vector cost $\vec{c}(n, n') = (c_1, c_2, \dots c_q)$, is reversed and labelled with a scalar cost $c_{mix}(n', n) = \sum_i c_i(n, n')$. The heuristic values $h_{mix}(n)$ are calculated then applying Dijkstra's algorithm to this reversed graph, as the cost of the shortest path from the goal node to all other nodes.

The experiments described in the next chapters compare the performance of blind and heuristic versions of $NAMOA^*$, MOA^* and TC. In the case of $NAMOA^*$ and MOA^*, blind search amounts to using a zero heuristic, $\forall n \; H_0(n) = \{\vec{h}_0(n)\} = \{\vec{0}\}$, while in TC a "blind" version can be easily derived with $h_{mix}(n) = 0$. Heuristic search is evaluated with $\forall n \; H(n) = \{\vec{h}_{TC}(n)\}$, where $\vec{h}_{TC}$ is as defined in this section.

2.4.5 Formal properties

Let us denote by $H^*(n)$ the set of costs of all nondominated paths from node n to goal nodes, $\vec{h}_n^* = \vec{g}(P_{n\gamma})$ for some $\gamma \in \Gamma$. We say that a heuristic function $H(n)$ is *optimistic* if,

$$\forall \vec{h}_n^* \in H^*(n) \quad \exists \vec{h}_n \in H(n) \quad | \quad \vec{h}_n \preceq \vec{h}_n^* \tag{2.25}$$

Let us denote by C^* the set of all nondominated solution costs $\vec{c}^{\,*}$. Whenever $H(n)$ is **optimistic**, $NAMOA^*$ is guaranteed to terminate with the set of all nondominated solution paths P^* (**admissibility**), with $\vec{g}(P^*) = \vec{c}^{\,*}$ for some $\vec{c}^{\,*} \in C^*$. Additionally, related to the **efficiency** of the algorithm, the use of good heuristic cost estimates can help us to filter candidate paths, reducing the number of alternatives that need to be considered during search.

A well designed systematic procedure to obtain a fully informed optimistic heuristic was proposed by Tung & Chew (1988, 1992). The proposed heuristics are general and can be used by $NAMOA^*$, which accepts any such multiobjective heuristic functions. These two heuristic functions have been described in section 2.4.4. A formal characterization of these heuristic functions can be found in chapter 4.

Formal developments (Mandow & Pérez de la Cruz, 2006, 2010a) show that algorithm $NAMOA^*$ is **optimal** over the class of admissible multiobjective search algorithms when heuristics are monotone. In other words, no algorithm in this class equipped with the same heuristic information can skip the expansion of a path expanded by $NAMOA^*$ without compromising its admissibility.

[17] Tung & Chew (1992) called this heuristic $\vec{q}(n, n')$.

[18] Tung & Chew (1992) called this heuristic with the somewhat confusing name $h^*(n, n')$.

If the heuristic function $H(n)$ satisfies the so-called **consistency** property, then all labels expanded by $NAMOA^*$ are known to be nondominated. In such cases, all labels in the $G_{cl}(n)$ sets are *permanent* and can never be *pruned* (see Mandow & Pérez de la Cruz (2010a) for details).

This is not the case for MOA^*, where an additional set $CLOSED$ is used. As all labels from a node are expanded simultaneously and some of them can be suboptimal, the node can be put back later again in $OPEN$. A family of sample graphs was introduced by Mandow & Pérez de la Cruz (2010b), where this reexpansion of nodes can lead blind MOA^* to perform a number of label expansions that grows exponentially with the size of the problem, while linearly in blind $NAMOA^*$.

The *path expansion*, *filtering* and *pruning* processes have been not formally studied for TC. Chapter 4 takes up this issue.

Most of the experiments and results described in this thesis involve consistent heuristics (see section 2.4.4). Chapter 7 studies the application of inconsistent heuristics to multiobjective search. The first direct consequence is that expanded labels can no more considered as permanent, i.e. an unknown number of suboptimal labels can be expanded. The next section introduces an algorithmic improvement to deal with inconsistent heuristics.

2.4.6 Multiobjective Pathmax and $NAMOA^{**}$

Extensions of A^* to deal with inconsistent heuristics have been studied and analyzed over the years. The pathmax rule (Mérõ, 1984) is one of such improvements. Inconsistency can make the sequence of $f(n)$ values along a path to be not monotonically nondecreasing. This can lead the algorithm to unnecessary node reexpansions, i.e. to the expansion of suboptimal paths for certain nodes. Pathmax uses the information available along nodes in known paths to force consistency at least with the information previously presented by ancestors of a given node n. The single-objective version of pathmax can be stated as follows,

Definition 2.20 *(Mérõ, 1984)* ***Single-objective pathmax****. Replace the evaluation function $f(n) = g(n) + h(n)$ for each node n with*

$$f(n) = \max\{g(n') + h(n') \mid n' \textit{ is on the current path to } n\} \qquad (2.26)$$

Of course, the pathmax rule does not completely prevent node reexpansion (Felner et al., 2011). In the case of A^* the advantage of consistent heuristics is that a node removed from the $OPEN$ queue will never be reopened. With pathmax, the f values never decrease along traversed paths, as the pathmax rule corrects inconsistent values. However, f values can still be non-monotonic for paths that have not been examined yet (Nilsson, 1998, p.153)(Zhou & Hansen, 2002).

The pathmax rule has been successfully extended to the multiobjective case.

Definition 2.21 *(Dasgupta et al., 1995, Section 3.1)* ***Multiobjective pathmax****. Let us assume q objectives. Each heuristic cost $\vec{h}(n)$ belonging to $H(n)$ is a vector $\vec{h}(n) = (h_1(n), \ldots, h_q(n))$, where $h_i(n)$ are domain specific heuristics. When each successor m of a node n is generated, the basic heuristic evaluation function is used to evaluate the* *heuristics at the node m. But before each heuristic is entered in $H(m)$,* ***pathmax*** *is applied as follows:*

- *For each heuristic vector $\vec{h}(n)$ in $H(n)$*
  - *For each heuristic vector $\vec{h}(m)$ evaluated at node m*
    - *Create a new vector $\vec{h}'(m)$ such that for each dimension k:*

$$h'_k(m) = \max\{h_k(m), h_k(n) - c_k(n,m)\} \tag{2.27}$$

      where $\vec{c}(n,m)$ is the cost of the arc from n to m.
    - *Put $\vec{h}'(m)$ in $H(m)$ and remove dominated heuristics, if any.*

Some formal properties were also proven by Dasgupta et al. (1995).

Theorem 2.8 *(Dasgupta et al., 1995, Theorem 3) The set of heuristics generated by* (2.27) *with multiobjective pathmax remain admissible.*

Theorem 2.9 *(Dasgupta et al., 1995, Theorem 4) Any node expanded by an admissible best-first search algorithm A using multiobjective pathmax is also expanded by A without using pathmax.*

Thus, pathmax can be used in the same way as in the scalar case. However, it plays a more significant role in partial order search. Dasgupta et al. (1995) proved that the set of nodes expanded by an admissible best-first algorithm using multiobjective pathmax can be reduced in general.

Theorem 2.10 *(Dasgupta et al., 1995, Observation 5) There are problem instances where an admissible best-first search algorithm that does not use* ***pathmax*** *will have to expand an arbitrarily large number of nodes which will not be expanded if it uses* ***pathmax****.*

The example provided by Dasgupta et al. (1995) in figure 2.3 shows that the subtree under n_5 will be expanded without pathmax. Only one vector in each $H(n)$ was considered for simplicity. Nodes n_2 and n_3 will be necessarily expanded as the single vector in $F(n_3) = \{(2,14)\}$ dominates the single vector in $F(n_4) = \{(3,12)\}$. But the expansion of n_5 depends on whether pathmax is used or not. If pathmax is not used, $F(n_5) = \{(4,9)\}$ whose single vector is nondominated with $F(n_4)$. However, if multiobjective pathmax is used, then $H(n_5) = \{(\max\{2,(1-1)\}, \max\{7,(13-1)\})\} = \{(2,12)\}$ and hence $F(n_5) = \{(4,14)\}$ whose single vector is dominated by the single vector in $F(n_4)$. Thus, the solution cost $F(n_4) = \{(3,12)\}$ is found and this one will filter the unique value in $F(n_5)$, avoiding the expansion of n_5. In this case the solution path to n_4 does not a contain a fully informed non-goal node in any objective.

The theorem show that pathmax rule has a greater significance in multiobjective frameworks. However, it is commonly believed that pathmax definitely corrects problems derived from inconsistency.

Corolary 2.1 *The pathmax rule does not make the F function monotonic.*

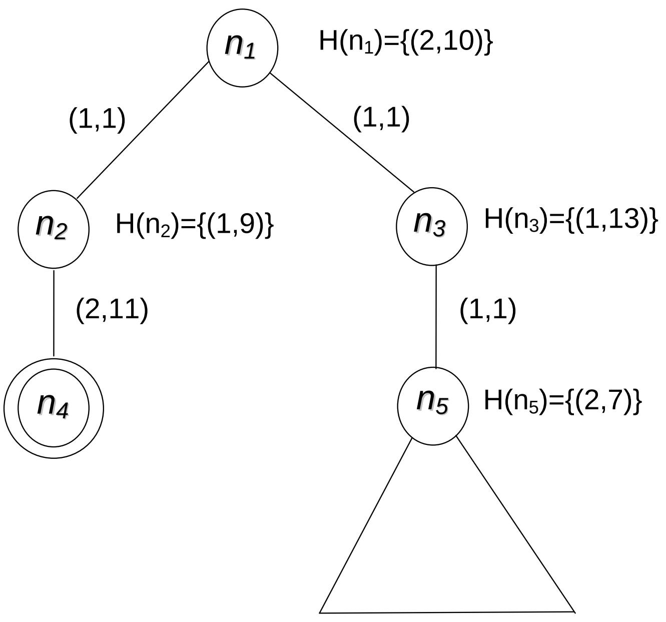


Figure 2.3: The advantage of multiobjective pathmax (Dasgupta et al., 1999)

An analogous example to the one provided by Felner et al. (2011, section 5.2), illustrating that reexpansion can not be avoided by pathmax, can be found for the multiobjective case. In the multiobjective case, no reexpansions are performed with admissible heuristics, but in some cases dominated paths may not be skipped with inconsistent heuristics, as the example shows. For simplicity the example uses $H(n)$ and $F(n)$ sets with a single element. Figure 2.4 shows an isolated part of a bigger graph where the heuristic is admissible but inconsistent. The heuristic values for all nodes are always a lower bound of the cost (50,100) of the unique nondominated solution path in this part of the graph, namely $P^*_{s\gamma} = \langle s, n_1, n_3, \gamma\rangle$. But the heuristic is inconsistent (e.g. $(1,1) \in H(n_3)$, $(48,98) \in H(n_1)$) and therefore the $\vec{f}$ vectors non-monotonic (e.g. $\vec{f} \in F(n_3)$, $\vec{f'} \in F(n_1)$, and $\vec{f} \prec \vec{f'}$).

Without pathmax, the path $P'_{sn_3} = \langle s, n_2, n_3\rangle$ will be expanded and the $\vec{f}$ value of node n_3 will be (11,21). Then, all paths in the subtree under node n_3 with values dominating the cost of the optimal path (50,100) will be expanded, because $(11,21) \prec (49,99)$. The path with $\vec{f}$ value (49,99) of node n_1 will be expanded after. This example shows that a dominated path $P'_{sn_3} = \langle s, n_2, n_3\rangle$ with a $\vec{g}$ cost (10,20) (dominated by another optimal nondominated cost (2,2) to n_3) is inserted in $G_{cl}(n_3)$, namely the path $P''_{sn_3} = \langle s, n_1, n_3\rangle$.

However, with multiobjective pathmax the path P''_{sn_3} will be expanded first than P'_{sn_3} because the $\vec{f} \in F(n_3)$ will be updated to (49,99) as $H(n_3) = \{(\max\{1,(48 - 1)\}, \max\{1,(98-1)\})\} = \{(47,97)\}$. This will cause the expansion of $P^*_{s\gamma}$ before P'_{sn_3}, and then the solution (50,100) will filter the cost of the path P'_{sn_3} with $G_{op} = \{(10,20)\}$ and $H(n_3) = \{(47,97)\}$, and thus $F(n_3) = (57,117)$.

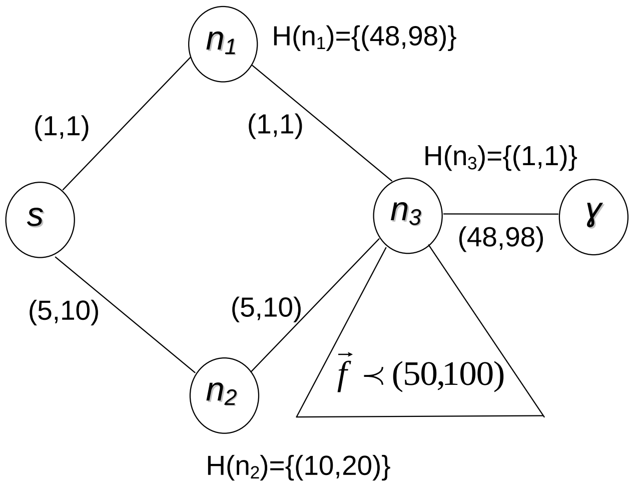


Figure 2.4: Multiobjective pathmax can not avoid the expansion of some dominated paths

Felner et al. (2011) presented BMPX, a bidirectional version of pathmax, where heuristic cost values are propagated even to nodes that are not ancestors, as a partial solution for the single-objective case. The extension of BMPX to the multiobjective case deserves attention in the future.

Dasgupta et al. (1999) presented an extension of MOA^* called MOA^{**} using multiobjective pathmax. In the same context, a pathmax version of $NAMOA^*$, named $NAMOA^{**}$, was described by Mandow & Pérez de la Cruz (2010a). However, this algorithm has never been evaluated in practice. The algorithm $NAMOA^{**}$ is the same algorithm described in table 2.1 of chapter 2, where the heuristic functions $H(n)$ are updated with multiobjective pathmax as indicated by definition 2.21.

2.4.7 Related Work

The general paradigm of multicriteria heuristic search was considered by (Mandow & Pérez de la Cruz, 2003). Multiobjective heuristic search is considered only as one of the possible categories. In this field, other multiobjective heuristic strategies are possible, for example when the calculation of all nondominated solutions is sometimes unnecessary. An adjustment of preferences is made *a priori* resulting in best-compromise solutions.

The method Best Compromise A^* (namely BCA^*) is based in multiobjective A^* search, but the ordering of labels is based in an aggregation function, for example the Tchebycheff norm. BCA^* explores first the labels potentially best according to the actual preferences considered. Futtersack & Perny (2000) proposed a modification of MOA^* while Galand & Perny (2006) of $NAMOA^*$ to handle this situation. The later also compared a kA^* scalarization strategy derived from k-best paths algorithms. Some other possible aggregation functions are studied in the thesis of Galand (2008).

Some improvements over BCA^* include some changes on the aggregation functions used (Sauvanet & Néron, 2010). An application to realistic scenarios can be found in the thesis of Sauvanet (2011). Another alternative ordering of labels, given as $\sum_i w_i v_i$ for $\vec{v} = (v_1, v_2, \ldots, v_q)$ and $\sum_i w_i = 1$, is the operator OWA (Ordered Weight Average) that lets us obtain well-ballanced solutions (Galand & Spanjaard, 2007). The Choquet Integral constitutes a more general method; the Tchebycheff norm or OWA operator are only particular cases. The costs are aggregated using weighting functions defined on every subset of criteria (Galand et al., 2010).

Besides, Dasgupta et al. (1999) developed some modifications of the MOA^* algorithm for extended scenarios, for instance when limited memory is available (algorithm $MOMA^*$) or the heuristics are not consistent (algorithm MOA^{**}). The extension of the *pathmax* rule to multiobjective heuristic search was applied in the later case (Dasgupta et al., 1995).

In the field of depth-first multiobjective best first search, Harikumar & Kumar (1996) presented an iterative deepening approach for multiobjective heuristic search, $IDMOA^*$, extending the single-objective algorithm IDA^* (Korf, 1985), with a run for each single objective. Coego et al. (2009, 2012) recently devised a new algorithm $PIDMOA^*$ which performs a real multiobjective A^* search with a multiobjective threshold, that takes into account solutions already found.

Other related multicriteria approaches include near admissible algorithms (Perny & Spanjaard, 2008), multiobjective frontier search techniques (Mandow & Pérez de la Cruz, 2007, 2008, 2009, 2010c) which allow dramatic reductions on space requirements, the investigation of alternative linear aggregate orderings for multiobjective labelling methods (Iori et al., 2010) or recent computational studies comparing the performance of MOA^* both with several runs of A^* for planning domains (Bryce, 2012) and an approximated version of A^* (namely LDA^*) for grid maps of games (Bayili & Polat, 2011).

2.4.8 Summary

This chapter provides an overview on the field and delimits the frame where the algorithms studied in this thesis work. The three algorithms $NAMOA^*$, MOA^* and TC have been properly described, as well as general well defined multiobjective heuristic functions. The case of inconsistent heuristics is also considered in some detail, with the pathmax rule and algorithm $NAMOA^{**}$.

However, the stage of formal and empirical development varies from one algorithm to another. While formal developments are complete in $NAMOA^*$ (Mandow & Pérez de la Cruz, 2005, 2006, 2010a), some questions still lacked adequate characterization for MOA^* before this thesis. In the case of TC, formal proofs were not even properly completed.

Regarding empirical evaluation, only limited experiments with MOA^* and $NAMOA^*$ were performed for square grids with Manhattan distance (Mandow & Pérez de la Cruz, 2005). Both MOA^* and TC were not empirically tested by their authors.

This thesis completes the formal developments of MOA^* and gives a formal framework to TC. Regarding empirical evaluation, comprehensive analysis will be presented both for blind and heuristic search using these three algorithms.

Chapter 3

Evaluating the Performance of Multiobjective Search

This chapter reviews the relevant literature on the experimental evaluation of multiobjective search algorithms and gives a detailed explanation of the problem sets used in this thesis. Several standard tools and benchmarks have been proposed in the literature (Klingman et al., 1974; Skriver & Andersen, 2000; Schultes, 2005; Santos, 2007b). While some authors test their algorithms on artificial problems (Skriver & Andersen, 2000) it is becoming frequent to provide evaluations also on realistic scenarios (Raith, 2009; Sauvanet, 2011). Both present different advantages.

The chapter is organized as follows: first, a summary of related work and previous test sets used by authors on multiobjective search can be found in section 3.1. Section 3.2 describes artificial and realistic scenarios used for testing the algorithms analyzed in this thesis. Section 3.3 address the problem of how to evaluate the performance of multiobjective search, i.e. which measures are important when comparing different approaches or algorithms.

3.1 Antecedents

Empirical evaluation of algorithms (Johnson, 2002) is a standard practice in Computer Science and Artificial Intelligence. Empirical evaluation allows the comparison of algorithms in extensive and varied cases where formal analyses are not possible. The main methodological principle is reproducibility, which should guarantee that other researchers can perform the same evaluations. In order to make evaluations of new algorithms in an easier way, many fields have developed standard problem benchmarks. This allows quicker comparison of the pros and cons of new alternatives against interesting problem sets.

Empirical evaluation has to be performed with certain care in order to make comparisons as fair as possible. Evaluations can depend on many practical issues like data structures, their implementation, the programming language used, or even the hardware where the programs are finally run. When comparing two similar algorithms it is desirable that they share as much code as possible.

Generally, the development of good widely available benchmarks problems in a particular field is accompanied of rapid development of that field. Many interesting problem sets have been defined for single-objective, e.g. Korf's problem suite for the 15-puzzle (Korf, 1985) or Baldur's Gate game maps [1].

In particular, the "9th DIMACS Implementation Challenge: Shortest Path"[2]) presented an extensive set of maps and problems for route planning. They are usually taken as standard benchmarks for authors who want to test algorithms in a category, e.g. DIMACS or Europe maps by the community involved on route planning algorithms for road maps (Delling et al., 2009; Bauer et al., 2010b; Bast et al., 2007; Delling et al., 2011a).

Formal analyses on single-objective search algorithms generally focus on the number of iterations or nodes expanded as a measure of performance and graph shape as a measure of problem difficulty. Empirical analyses on time performance take into account other implementation aspects like for example the use of heaps to implement $OPEN$ lists.

Regarding multiobjective search, algorithm performance is related to a number of distinct problem factors, like graph shape, solution depth, number of objectives and correlation between objectives. The number of nodes expanded is no longer a conclusive performance measure. The accrued number of individual labels simultaneously stored by the algorithm or the actual account of dominance tests or *merge* operations are more influential factors on the performance of multiobjective algorithms.

Although there is not a clearly established and widely accepted set of problem benchmarks for multiobjective algorithms, many empirical evaluations have been performed over the years in this field.

3.1.1 Multiobjective Blind Search

Korf et al. (2005); Zhou & Hansen (2006) give a good overview of some benchmarks used to evaluate recent techniques. Several authors have proposed different sets of problems for the empirical evaluation of multiobjective search.

A brief summary of different random instances proposed for blind multiobjective search follows.

A Random graphs: NETGEN

One of the earliest methods to generate random graphs found in the literature is NETGEN (Klingman et al., 1974). This program generates first a connected *skeleton*. Then, in a second phase, additional arcs are randomly added. Arc cost values are generated in the range [1,100] but the maximum cost is assigned to a percentage of the skeleton arcs (specified as input parameter) in order to prevent the use of all arcs from the skeleton and have more difficult problems to solve. Several authors have extended the NETGEN generator to multiobjective settings.

Huarng et al. (1996) used for their computational evaluation small random graphs with up to 200 nodes (and no more than 25 Pareto-optimal solutions) that were generated with a modified NETGEN program, with two random costs independently generated in the range [1,200].

[1] `http://www.movingai.com/benchmarks/`

[2] `http://www.dis.uniroma1.it/~challenge9/`

Skriver & Andersen (2000) used an extended NETGEN program to test their algorithm with random graphs and concluded that the networks generated contained few Pareto-optimal solutions (for 100 nodes and 900 arcs, no more than 7.5 on average). The efficient paths seem not to be "spread out through the network". This is attributed to a deterministic generation of a Hamiltonian cycle to ensure connectedness, and as a result the existence of only some few more alternative nondominated subpaths, that share some part of this central path from the source to the goal node.

Guerriero & Musmanno (2001) used also random graphs built with a modified version of NETGEN. The inputs were the number of objectives (2,3, or 4), the number of arcs and nodes (up to 40,000 nodes and 100,000 arcs) and the density of arcs (1.5 to 30). Contrarily to the argument of Skriver & Andersen (2000), the number of Pareto-optimal paths in these problems is very high, from 3,351 for the simplest biobjective network with 500 nodes up to 156,264 and 465,347 efficient paths for the hardest configurations with 40,000 nodes for two and four objectives, respectively.

Guerriero et al. (2001) also generated NETGEN problems with a number of arcs fixed to 1,000,000. The number of nodes goes from 10,000 to 22,360. Arc costs values are integers in the range [1,1000] chosen from an uniform distribution.

Guerriero et al. (2001) also introduced additional NETGEN cases. *Fully dense* problems were built in such a way that all possible arcs are included (i.e. all nodes fully connected). The number of nodes is 200, 400, 600, 800, or 1,000 with a total number of arcs up to 999,000 in the case of graphs with 1,000 nodes. Arc costs values are integers in the range [1,1000] chosen from an uniform distribution as for the other random graphs.

B Random graphs: NETMAKER

The relatively small number of Pareto-optimal paths obtained by the NETGEN program is an argument against this random problem generator given by some authors.

Skriver & Andersen (2000) proposed a new random generator, NETMAKER. Given a number of nodes, the program first builds a Hamiltonian cycle as NETGEN. However, the generation of additional arcs in the second phase is not completely random but based in two input parameters: an interval of random outgoing edges allowed for each node (branching factor) and a second interval that specifies which nodes are allowed to be reached from a particular node with those outgoing edges (interval length). For example, with branching factor in the range [1,3] and interval length $2 * max$, from node 10, it is randomly chosen that 1 to 3 outgoing edges are added, connecting node 10 to nodes in the range $10 \pm (2*3)$, i.e. from node 4 to node 16. This method achieves a wider spread of Pareto-optimal paths along the graph.

They generated networks from 100 to 500 nodes and intervals [1,3], [2,4], and [7,15] were used for the branching factor (i.e. up to $500 * 15 = 7,500$ arcs). Two objectives negatively correlated were used (one cost is in the range [1,33] while the another in [67,100]), in order to obtain more Pareto-optimal solutions as well.

Raith & Ehrgott (2009) argued also that networks generated with NETGEN had a small number of Pareto-optimal solutions and used NETMAKER to generate random graphs. They used as input parameters: a number of nodes of 3,000, 7,000, 14,000, or 21,000, a branching factor interval of [1,20], [5,15], or [10,40], and an interval length I_{node} of 20 or 50. The interval length is defined in the way that only nodes $i \pm \lceil \frac{I_{node}}{2} \rceil$ are allowed to be connected with node i. Several improvements were devised by Raith & Ehrgott (2009), like penalizing arc costs in the Hamiltonian cycle. Nevertheless, these graphs generated with NETMAKER with up to 21,000 nodes and up to $21,000 * 40 = 840,0000$ arcs have only 17 nondominated solutions at most. These same random graphs were used in the thesis of Raith (2009).

C Other random graphs

Clímaco & Martins (1982) used random graphs to test their algorithm. Only the number of nodes is used as input. The result is a graph where each node is the tail of *lambda* arcs with $\lambda \in [1, 10]$. The head of each arc is randomly selected from the set nodes that exclude parallel arcs and arcs from a node to itself. Two costs are independently generated in the range [1,100] for each arc. These random graphs were small (up to 500 nodes and no more than 2,500 arcs) and had a few number of Pareto-optimal paths (no more than 13).

An early work from Nance et al. (1987) uses parametric random graphs where they varied parameters like the number of "links" (branching factor). However, there was no random generator publicly available and the graphs produced were small-sized.

Brumbaugh-Smith & Shier (1989) used random graphs of 100 and 250 nodes where the number of arcs is varied from 300 to 1,900, with less than 700 Pareto-optimal solutions in the worst case. The importance of this experimental study is the introduction of a correlation ρ between randomly generated components of the vector costs, with values for ρ in the range [-1,1]. Some of the chosen values were very near to -1, in order to study difficult cases.

Mote et al. (1991) considered random graphs with 1,000 nodes and several configurations of number of arcs (3,000,5000,or 1,000) and correlation between the two cost values (0.0, 0.5, 0.8). These two positively correlated values are integers in the range [1,200].

Iori et al. (2010) worked also with random graphs, on the basis of the benchmark from Gandibleux et al. (2006). A rooted tree is generated and random arcs are added until a desired density is reached. Random costs in different ranges for min-max problems are generated for nine pairs of combinations of node-density: 50,100 or 200 nodes with a density of 5%,10% or 20%. Three classes of ranges for cost values are used, from [1,100] up to [1,1000000] in some cases.

Random graphs have been also used in other contexts like time-dependent bicriteria scenarios (Hamacher et al., 2006) or tricriteria MinMax problems (Gandibleux et al., 2006; de Lima Pinto et al., 2009). In the first case, extending the NETGEN generator with time horizons, the authors have tested with up to 1000 nodes and branching factors of 2,4,6 and 8. In the second case, the work of de Lima Pinto et al. (2009) introduced random graphs of up to 5,000 nodes and 24,995,000 arcs, varying density (number of arcs) and intervals of the cost integer range values (from [1,5] to [1,70] for two objectives and fixing the third to [1,10000]). However, there were no more than 458 nondominated solutions for the largest size (5,000 nodes and 24,995,000 arcs).

Martins (1984c) also used random graphs with 25 and 50 nodes and different ranges in the costs generation (from [1,10] to [1,5000]) for special bicriterion path problems, where at least one objective is MaxMin.

Also in the context of blind multiobjective search, extensive datasets with random graphs have been used to test ranking algorithms. Martins et al. (2007) proposed a standard benchmark to test labelling algorithms with several classes of problems by the combination of three factors: number of nodes, density of nodes (equal to vary number of arcs) and number of objectives (from 2 up to 10). The general random graphs include the combination of the three factors with up to 15,000 nodes. The largest number of criteria (10) was considered only for random graphs with up to 5,000 nodes and 30,000 arcs.

They also generate fully connected graphs ("complete networks") of up to 120 nodes with density of arcs fixed to the number of nodes minus 1 (i.e. all nodes connected with each other). The number of Pareto-optimal solutions published for each class of graph shows that while in random graphs it is under 200 solutions, complete graphs have several hundreds with far fewer nodes. In the largest number of criteria (8) for this type, more than 1,700 nondominated solutions are found while only a little more than a hundred in general random graphs.

The same set of instances was used in a twin paper (Paixão & Santos, 2007). The testsets are extended with larger instances, and are divided into small and large datasets. The first are used to discard the low performance labelling alternatives, while the second type are used to identify the most efficient alternatives. Costs were in the range [1,1000]. Paixão & Santos (2008) worked with the same set of instances from (Martins et al., 2007).

Caramia et al. (2010) analyzed multiobjective hazardous material transportation problems. They generated random graph instances obtained with the random generator `sprand.exe` from the "9th DIMACS Implementation Challenge: Shortest Paths". Three objective values were randomly generated in the range [1,100] . Node and density are varied, where the bigger problems have 300 nodes and 31,365 arcs with no more than 160 Pareto-optimal solutions.

Galand et al. (2010) also used random graphs in the context of blind labelling Choquet-based optimization. They generated 50 different instances for each size (1,000, 2,000, 3,000, and 4,000 nodes) and number of criteria (2, 3, 5 , and 10).

D Grids: GRIDGEN

Bertsekas (1991) devised for the single-objective case a random grid generator with a connected "skeleton" (guaranteeing problem feasibility) and additional arcs between random starting and ending nodes. These random generator GRIDGEN was extended by Bertsekas et al. (1996), fixing the number of arcs to 1,000,000 and the total number of additional arcs to approximately 2, 3, 4, or 5 times the number of grid arcs.

Guerriero & Musmanno (2001) extended this random generator to the multiobjective case (2, 3, or 4 objectives). Both rectangular and square grids of different shapes, with up to 625 nodes and 2400 arcs were generated, reaching in this case to more than 2 millions of nondominated solutions for some configuration. For biobjective square or rectangular grids, a little more than 20000 Pareto-optimal paths can be found in the hardest configuration.

Square grids problems from Guerriero et al. (2001) are generated with the GRIDGEN generator in the way of Bertsekas et al. (1996). The number of arcs for these grid problems is fixed at 1,000,000, while the number of nodes is set in such a way that the ratio additional/total arcs is an integer in the range [3,7].

However, Guerriero et al. (2001) extended the testset with additional cases, following Bertsekas et al. (1996). Euclidean grids problems are generated in a similar way, but the additional arcs have a cost equal to the euclidean distance between the two nodes. Arc costs values are integers in the range [1,1000] chosen from an uniform distribution for both cases.

E Other Grids

Mote et al. (1991) also used grids of 400 nodes, where two different parameters are varied: different shapes (different side lenghts and three different number of arcs, namely 1,430, 1,500, 1,520) and correlation between cost values (0.0, 0.5,and 0.8). The two correlated values are in the range [1,100] for these grids.

Murthy & Olson (1994) used some of these grid networks with 400 nodes, and some more with other shapes and 900 nodes and up to 3,500 arcs for an interactive multiobjective procedure. The range of cost values is augmented up to [1,1000] and more than a hundred of Pareto-optimal solutions can be found. They stated that "grid networks are particularly hard problems".

Martins et al. (2007) also used their own set of square grids. They varied some parameters as explained before for random graphs, but grids have density fixed to approximately 4 (i.e. square grids with 4 vicinity), therefore varying only the number of nodes (up to 144) and the number of criteria (2 to 10). The largest number of criteria (10) was considered only for grids of up to 100 nodes. Vector costs were randomly generated in the range [1,1000]. The number of Pareto-optimal solutions in these grids is far higher than in the random graphs generated by the same authors, with more than 20,000 Pareto-optimal solutions for the larger instances.

Raith & Ehrgott (2009); Raith (2009) also proposed grid networks of two types: square grids of up to 40,000 nodes and near 160,000 arcs (with no more than 300 hundred nondominated solutions) and grids with different shapes of approximately 4,900 nodes and 19,000 arcs (with more than hundreds of solutions, even more than 1,500 for some cases). It is important to note that the larger number of Pareto-optimal solution paths appears in the narrower grids, where one dimension is far smaller than the other.

Caramia et al. (2010) also used random square grids applied to hazardous material transportation problems. They were obtained with the random generator `spgrid.exe` from the "9th DIMACS Implementation Challenge: Shortest Paths". Three objective values were randomly generated in the range [1,100]. Search was conducted from one corner to the opposite corner and the square grids had no more than 400 nodes. Nevertheless, more than 800 Pareto-optimal solutions can be found.

F Other empirical test sets: realistic scenarios

Several authors have used problems from the route planning domain over the years (Bertsekas et al., 1996; Zhan & Noon, 1998). Benchmarks from the "9th DIMACS Implementation Challenge: Shortest Paths"[3] have been widely used for single-objective shortest path computations (Bauer et al., 2010b). These comprise several sets of road

[3] `http://www.dis.uniroma1.it/~challenge9/download.shtml`

maps of different sizes obtained from the real road network of the U.S.A. Two different cost values are available for each arc: time and distance. DIMACS maps are discussed in detail in section 3.2.3. Little experimentation has been carried out to extend these realistic scenarios to multiobjective cost-valued maps.

Raith & Ehrgott (2009); Raith (2009) also used in their computational studies road maps generated from DIMACS benchmarks [4]. The original maps were modified to include a Hamiltonian cycle and two cost values for each arc (see section 3.2.4). The road networks used are relatively large (more than 300,000 nodes and approximately 1,200,000 arcs for the New Jersey map), but with few Pareto-optimal solution costs (no more than 22). The thesis of Raith (2009) extends these cases to a realistic scenario: biobjective cyclist route choices on small maps of Auckland (New Zealand).

Caramia et al. (2010) used also road networks for the evaluation of their algorithm in the context of hazardous material transportation problems. A map from the italian region of Lazio is used, with 331 nodes and 441 arcs.

An application to realistic scenarios can also be found in the thesis of Sauvanet (2011). Three different maps with sizes in a similar range to the three presented by Raith & Ehrgott (2009); Raith (2009) have been used for multiobjective cyclist route planning. Two maps from Paris and Berlin were obtained from OpenStreetMap [5] while the third is from San Francisco Bay and was obtained from TIGER/line ® data (Topologically Integrated Geographic Encoding and Referencing system) of the U.S. Census Bureau. The first has no more than 200 Pareto-optimal solution paths, the second one no more than 300 and the third from USA with a shaped mesh grid of almost 175,000 nodes and 436,000 arcs has no more than 600 nondominated solutions.

3.1.2 Multiobjective Heuristic Search

Empirical evaluations of multiobjective heuristic labelling algorithms have been relatively scarce. To the author's knowledge, no previous systematic evaluation of the TC algorithm has been performed. The original papers (Tung & Chew, 1988, 1992) tested the algorithm only over hand-made examples.

The same applies to MOA^* (Stewart & White, 1991). Little experimental evaluation related to MOA^* has been found before the description of $NAMOA^*$ (Mandow & Pérez de la Cruz, 2005). The depth-first version of MOA^*, namely $IDMOA^*$ (Harikumar & Kumar, 1996), was evaluated only over simple problems in the circuit partitioning domain (Harikumar & Kumar, 1997). Dasgupta et al. (1999) also applied some of their extensions of MOA^* (e.g limited memory requirements or inconsistent heuristics), like MOA^{**}, only over simple problem instances of different domains, like Operator Scheduling or Log Cutting. Even other recent approaches which are not complete, like the algorithm of Hallam et al. (2001) used only simple problems. More recent multiobjective developments have included more detailed empirical tests. Random graphs, random grids and road networks have been also proposed for multiobjective heuristic search.

[4]Files in DIMACS format are not publicly available over the Internet, but were obtained from the authors

[5]`http://www.openstreetmap.org/`

A Random graphs

Random graphs have been used for OWA^* (Galand & Spanjaard, 2007) with cost values in the range [1,100]. The number of nodes varies from 1,000 (with 190,000 arcs) to 3,000 (with 2,000,000 arcs).

BCA^* has been evaluated by (Galand & Perny, 2006) with randomly created graphs where the inputs are the number of nodes and the number of criteria. The number of nodes ranges from 200 (with 7,500 arcs) to 1,000 (with 200,000 arcs). The number of criteria evaluated are 5, 10, or 20.

An approximated version of MOA^* (Perny & Spanjaard, 2008) has been also evaluated with random graphs. The input parameters were the number of nodes (1,000, 2,000, and 3,000) and the number of objectives (2,5 or 10). Cost values were in the range [1,100].

B Hansen-like graphs

BCA^* was evaluated with graphs similar to those described by Hansen (1979). These graphs have $2 * p + 1$ nodes and 3^p arcs, with $p \in [5, 12]$.

Best compromise solutions with BCA^* were analyzed also by (Galand & Perny, 2006) with Hansen-like graphs containing 21, 25, 29, and 33 nodes.

C Random grids

Random square grids have been used by Mandow and Pérez de la Cruz to evaluate $NAMOA^*$ against frontier search $NAMOA^*$ (Mandow & Pérez de la Cruz, 2007) or MOA^* (Mandow & Pérez de la Cruz, 2005). Search was conducted from one corner of the grid to the opposite corner. Grid sizes varied from 101×101 (Mandow & Pérez de la Cruz, 2007) up to 400×400 in some cases (Mandow & Pérez de la Cruz, 2008). Cost values were in general in the range [1,10] but different ranges between [1,2] and [1,15] were evaluated by (Mandow & Pérez de la Cruz, 2010b). Most instances were bicriteria grids. Three objective cost vectors were used in (Mandow & Pérez de la Cruz, 2005, 2009).

Besides, MOA^* has been recently compared to LDA^*, an approximated version of A^*, on maze-like grid maps with size up to 96×96 for games (Bayili & Polat, 2011).

D Road networks

BCA^* has been also evaluated by Sauvanet & Néron (2010) on a moderated size map from Berlin of 77,940 nodes and 228,462 arcs. Two values are used for the costs: travel distance (in meters) and insecurity ($distance \times insecurity\ of\ the\ road$). They tested 4 groups of 20 instance problems with different ranges of Pareto-optimal solutions. The hardest 20 instances contain from 200 to 600 efficient solutions.

Two additional maps are used by the thesis of Sauvanet (2011) from Paris (29,086 nodes and 64,358 arcs) and San Francisco Bay area (174,975 nodes and 435,959 arcs) to test multiobjective cyclist route planning with BCA^* algorithm. Four categories of difficulty analogous to those presented in (Sauvanet & Néron, 2010) are found in the instances used by Sauvanet (2011). The hardest instances have up to 600 Pareto-optimal solutions for the map of San Francisco and 300 for the map of Paris. Three objectives have been also tested for some instances of the Berlin map, adding as objective the effort made by the cyclist (e.g. due to a slope).

3.1.3 Summary

Three main types of test sets have been proposed in the literature: random graphs, grids and realistic maps. Under the first category, several generators have been used: NETGEN, `sprand` or NETMAKER. But they seem not to be able to generate the difficult problems represented by grid classes. Only fully connected graphs can be roughly compared.

For grids, several shapes and sizes have been investigated. It is important to note that in these sets of problems searching from one corner to the opposite can be considered as the norm. GRIDGEN or `spgrid` are common generators.

In the case of realistic scenarios, only relatively small-sized maps have been used, when compared to single-objective search. The main reason is that multiobjective search takes a long time to solve big road map problems due to the inherent complexity of the problem.

In the following section some empirical benchmarks used in this thesis are presented, inspired in the typical cases found in the literature. Additionaly, new scenarios are investigated.

3.2 Test Sets used in this Thesis

This thesis presents a combination of randomly generated graphs and realistic route planning problems for experimental evaluation. Each one is adequate to evaluate different aspects of the algorithms and heuristics analyzed in this thesis. This section gives a detailed explanation of the characteristics of each one of the datasets used.

Artificially generated environments, like random grids, allow the controlled evaluation of performance with respect to different parameters, like number of nodes or correlation between objectives. On the contrary, realistic scenarios come with an implicit set of fixed parameters, e.g. the number of edges in a road map (like the number of cities connected to Malaga) can not be changed. However, the evaluation of an algorithm with well known datasets (usually large size maps) allows the simulation of performance under realistic conditions.

3.2.1 Artificial Problems: Random Grids

Random graphs are one the most recurrent test sets in the literature. However, gradual increase in problem difficulty is more controversial to calibrate than in grids, as several parameters have to be settled, like branching factor, number of arcs, and the set of nodes allowed to be connected by outgoing arcs to a particular node. Square grids with a fixed vicinity are easier to understand and the difficulty can be gradually increased only by increasing size. Besides, a computational study from Paixão & Santos (2008) reveals that for similar configurations, random grids have a large number of nondominated paths and Pareto-optimal solutions.

Random grids are common in the literature (Raith & Ehrgott, 2009; Guerriero & Musmanno, 2001) and at the same time realistic for certain applications like pathfinding in games (Bayili & Polat, 2011). In this thesis, random bidimensional square grids without obstacles are the main artificial testbed for heuristic search algorithms presented in chapter 2. Nevertheless, a smaller set of random graphs is also considered (see section 3.2.2).

The random grids presented in this section are designed to allow the controlled evaluation of performance with respect to solution depth and correlation between objectives. For this purpose, square grids of varying size have been randomly generated. A vicinity of four neighbours was used. Bidimensional costs (c^1_{ij}, c^2_{ij}) were considered for each arc from i to j. Cost values were integers calculated randomly in the range $[1, 10]$ using the scheme proposed by Mote et al. (1991). A positive association between arc costs is introduced using a correlation multiplier, $0 \leq \rho \leq 1$. The first integer arc cost c^1_{ij} is randomly generated using a uniform distribution in the range specified. The second arc cost is generated in the same range as

$$c^2_{ij} = \rho \times c^1_{ij} + (1 - \rho) \times c^{2*}_{ij} \tag{3.1}$$

where c^{2*}_{ij} is another integer randomly generated using a uniform distribution in the same range. However, this procedure does not cover the cases where there exists a negative association between arc costs, i.e. $-1 \leq \rho \leq 0$. In order to consider these cases, the following additional formula has been applied,

$$c^2_{ij} = 1 + (c_{max} - (|\rho| \times c^1_{ij} + (1 - |\rho|) \times c^{2*}_{ij})) \tag{3.2}$$

where c_{max} is the maximum value in the cost range (in the examples presented $c_{max} = 10$). The values of c^2_{ij} were rounded to the nearest integer in all cases.

The values of ρ considered in the experiments performed in this thesis were 0.8, 0.4, 0, -0.4, and -0.8. Notice that $\rho = 1$ implies that $c^1_{ij} = c^2_{ij}$, i.e. a single objective problem. Decreasing values of ρ yield progressively more difficult problems, where c^1_{ij} differs more and more from c^2_{ij}. When $\rho = -1$ the randomness is partially lost, since both vector components follow a linear rule $c^2_{ij} = 1 + c_{max} - c^1_{ij}$.

Two different classes of problem instances are considered, presenting quite different characteristics and difficulty.

3.2.1.1 Class I Square Grids

Let s be the number of nodes in each of the dimensions in the grid. In the first class (I), problems were generated searching from one corner (the upper left) of the grid $(0, 0)$, to the opposite $(s - 1, s - 1)$ (down right). Solution depth is $d = 2s - 2$ in this case.

For class I, a problem set was generated with sizes s varying from 10 to 100 in steps of 10, and 10 problems for each size. Therefore, solution depth varies from 20 to 200 in steps of 20. The total number of instance problems generated for this case is 500, i.e. 100 for each value of ρ. Nodes are connected in both directions. Thus, the total number of nodes and arcs for the larger size grids (100×100) is 10,000 and 39,600 respectively. The average number of Pareto-optimal solution paths for the ten larger problems (100×100) for each correlation is 8.6, 96.2, 274.7, 435.6 and 772.3 in order of decreasing value of ρ.

3.2.1.2 Class II Square Grids

In the second class (II), the size was set to $s = 2d + 1$, with the start node at (d, d), and the goal node (in the middle) at $(d/2, d/2)$, i.e. placing the goal node at depth d from start node (i.e. at a 1/4 of a corner).

For class II, d varies from 10 to 100 in steps of 10 with 10 problems for each size. The total number of instance problems generated for this case is 500, i.e. 100 for each value of ρ. Nodes are connected in both directions. Thus, the total number of nodes and arcs for the largest-sized grids (200×200) is 40,000 and 159,200 respectively. The average number of Pareto-optimal solution paths for the ten larger problems (200×200) for each correlation is 5, 43.6, 124.8, 192.5 and 376.4 in order of decreasing value of ρ.

Notice that in class II test problems the goal node is not placed as deep as in class I. However, class II problems present their own difficulty since search is generally not constrained by grid boundaries.

3.2.2 Artificial problems: Random graphs

In addition to random grids, this thesis considers also a set of random graph problems with three objectives used in the work of Caramia et al. (2010).

These random graphs[6] allow the evaluation of performance depending on number of nodes n, arc density d and number of objectives k. Different problem sets are considered with a number n of nodes equal to 100, 200 and 300. For each of these sizes, density values d were set to 0.2, 0.5 or 0.7. Each arc is labelled with a vector of three costs. Each one is an integer value in the range [1,100]. These problem sets were originally generated with a random graph generator (`sprand.exe`) from the 9th DIMACS Implementation Challenge on Shortest Paths[7].

For the evaluation of biobjective random graphs, two of these three arc costs were selected by pairs, using the same configuration of node-density. Three additional analogous sets of instances were derived from the original set of problems with three objectives. The computation of Pearson's correlation coefficient over pairs of two objectives is displayed in table 3.1. In general, these three objectives are linearly uncorrelated, resulting in moderately difficult multiobjective problems.

Source and goal nodes were set for all instances to nodes 1 and n, respectively. Ten different random instances are available for each of the nine combinations of n and d. Thus, the total number of problem instances for each category is 90, in total 360 problem instances. These four sets of instance problems let us evaluate the performance of algorithms when considering the simultaneous optimization of two objectives compared to three objectives at the same time. The biggest instances ($n = 300$, $d = 0.7$) have on average 26,423 arcs. The average number of Pareto-optimal paths for these ten hardest problem instances is 71.20 for the three objectives configuration, and 12.10, 12.30, 11.70 for the combinations of objectives (1,2), (1,3), and (2,3), respectively.

[6]The set of random graph files was kindly provided by Antonio Iovanella
[7]`http://www.dis.uniroma1.it/~challenge9/`

Objectives	ρ
1,2	0.01
1,3	0.01
2,3	-0.09

Table 3.1: Correlation between pairs of objectives for random graphs problems.

3.2.3 Realistic Route planning problems: 9th DIMACS challenge maps

In this thesis, route planning has been selected as a realistic domain for the application of multiobjective heuristic search. Route planning is a current research area with great practical importance, where graph search algorithms are put on test. Road maps can be defined as graphs where arcs represent roads, and nodes represent road junctions. Formal and empirical evaluation of search algorithms on road maps has a long tradition (Pearl, 1984, section 5.3) (Zhan & Noon, 1998). However, little work has been carried out on multiobjective route planning.

The main testbed used in this thesis for the domain of route planning comprise problems over maps from the "9th DIMACS Implementation Challenge: Shortest Paths". These are publicly available and include two objectives: time and distance. This scenario has been extended in this thesis, as explained below. Additionally, other realistic maps have been considered (see sections 3.2.4 and 3.2.5).

A set of twelve road maps of increasing size was prepared as part of the Challenge for benchmarking of algorithms [8]. These include arcs representing road segments, and nodes representing road junctions. Coordinates (longitude and latitude) are provided for each node. The data were originally taken from the 2000 U.S. Census Bureau's TIGER/Line ® files (Topologically Integrated Geographic Encoding and Referencing system).

Two sets of realistic multiobjective road map problems were built from the original data:

- The first set of problems includes as objectives the *time* and *distance* values provided in the files of the challenge.
- The another set of problems includes as objectives the *time* and *economic cost.* The former is taken directly from the files of the challenge while the latter is built specifically for this thesis.

Particularly, four maps from the DIMACS map set were selected in the first case (time vs distance): New York City, San Francisco Bay, Colorado and Florida. Some information about these maps is presented in table 3.2. A sample rendering of the New York city map is shown in figure 3.1. A set of 50 problems were generated for each map, selecting random start and goal nodes using a uniform distribution.

According to the information provided by the Challenge organizers, arcs are labelled with two different cost values: *physical distance* (c_1) and *travel times* (c_2). These are

[8]`http://www.dis.uniroma1.it/~challenge9/download.shtml#benchmark`

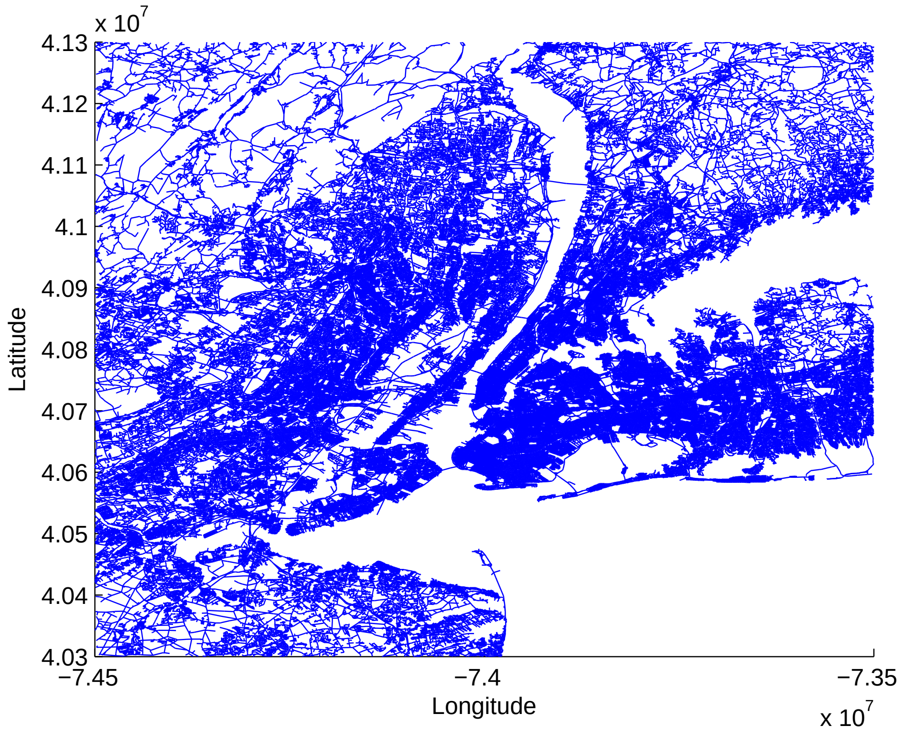


Figure 3.1: Rendering of the New York City map.

the two objectives to be minimized by multiobjective search in the first problem set, and shall be grouped in a vector cost $\vec{c} = (c_1, c_2)$. The cost of a path is calculated as the sum of the costs of its component arcs. Each arc cost is defined as follows,

- Original arc distances were calculated as the so-called *great circle* distance between its nodes. This is the shortest distance taking into account the Earth's curvature and average radius. *Physical distance* values are integers obtained from the original distances using the formula $c_1 = (int)(distance \times 10 + 0.5)$, i.e. the original distance data were multiplied by 10, and truncated after adding 0.5. This means that each arc cost represent decimeters and that each arc can accumulate, in the worst case, an error of 0.5 units.

- *Travel time* values are calculated as the integer part of the original distance divided by an average speed factor that depends on road category, i.e. $c_2 = (int)(distance/factor)$. There are four such categories, obtained from the TIGER/-Line data, with associated factors 1.0 (A - primary highway), 0.8 (B - primary road), 0.6 (C - secondary and connecting road), and 0.4 (D - local, neighborhood, and rural road).

Travel time, distance or economic cost are frequent objectives to be minimized, and most current road planners offer the opportunity to optimize each of them individually. While most current road planners optimize distance or time, it is interesting the opportunity to evaluate other factors. While these two seem highly correlated, both are

Name	Location	Nodes	Arcs	ρ	Avg. $\lvert C^*\rvert$
NY	New York City	264,346	730,100	0.96	198.62
BAY	San Francisco Bay	321,270	794,830	0.98	118.82
COL	Colorado	435,666	1,042,400	0.98	426.62
FL	Florida	1,070,376	2,712,798	0.97	738.76

Table 3.2: Road networks from DIMACS challenge (time/distance).

Name	Location	Nodes	Arcs	ρ	Avg. $\lvert C^*\rvert$
NY_2	New York City	264,346	730,100	0.16	2086.6

Table 3.3: Road network from DIMACS challenge with modified objectives (time/economic cost).

conflicting with economic cost. Thus, an alternative case was also considered. Two objectives are optimized simultaneously: travel time, and economic cost, which includes fuel cost and highway tolls.

The economic cost attribute is not available for DIMACS maps. A realistic cost value was calculated for the problems in this thesis according to the following considerations,

- Travel cost results from the addition of *fuel cost* and *tolls*.
- All roads *type A* are set to pay a toll of 1.86 cents per Kilometer (3 cents per mile). All other roads are toll-free. A rendering of the roads type A of the map is shown in figure 3.2.
- Fuel cost depends on *fuel price*, *fuel efficiency* and *travelled distance*.
- *Road type.* Fuel efficiency depends on road type, and particularly on its allowed maximum speed. We associate current speed limits in the NY City area to every TIGER/Line road type as shown in table 3.4. Road types were calculated using the physical distance, travel time, and time factors information.
- *Fuel efficiency.* Fuel efficiency is usually measured in miles per gallon (mpg) or litres per 100 kilometer (l/100Km). Fuel efficiency depends heavily on travel speed and vehicle type (and also on the particular model and production year). Instead of conducting our experiments for a particular car model, we used general gas mileage values provided by a standard chart from the U.S. Department of Energy [9]. These values are shown in table 3.4. This chart reflects the fact that, in general, maximum fuel efficiency is usually achieved at speeds in the range 80-90 Km/h (50-55 mph).
- *Fuel price.* The price of fuel was set to a current value in the New York City area of 375 cents/gallon (99 cents/litre).

[9]`http://www.fueleconomy.gov/feg/drivehabits.shtml`

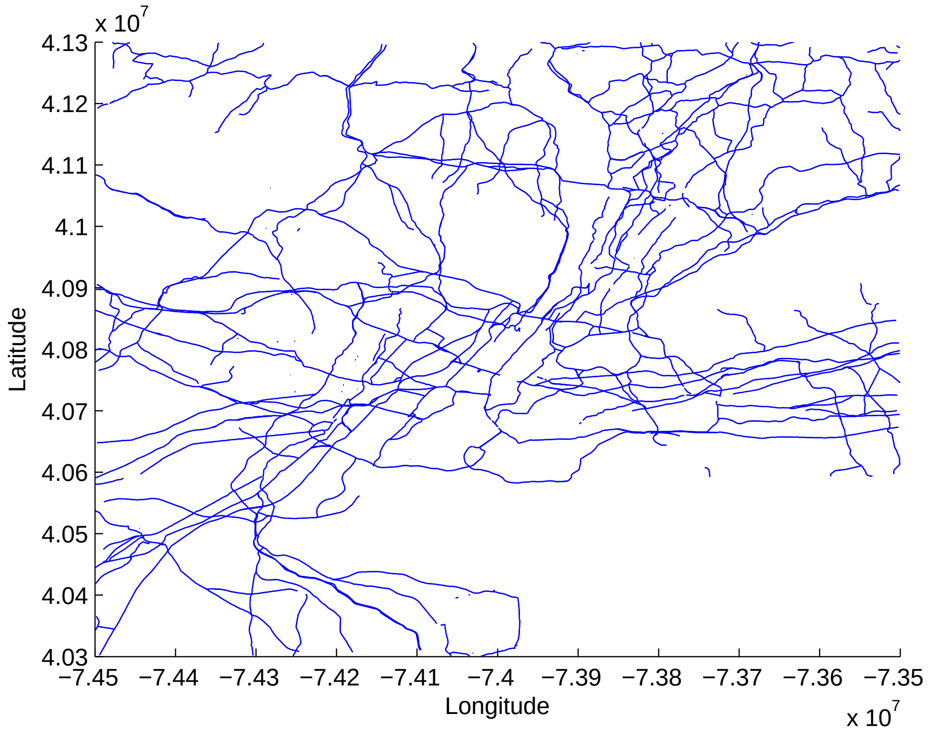


Figure 3.2: Rendering of primary (toll) highways of the New York City map.

Road type	Limit (mph)	Limit (Km/h)	mpg	l/100Km
A	65	105	27	8.71
B	55	89	30	7.84
C	50	80	30	7.84
D	30	48	29	8.11

Table 3.4: Speed limits and fuel efficiency values associated to each road type.

- In summary, fuel cost in cents was calculated for each arc in the graph according to the fuel spent travelling its distance with its assigned general fuel efficiency rate.

While these calculated costs may not reflect precisely actual travel costs, they are sufficiently realistic for experimentation purposes [10]. The computation of Pearson's coefficient over the two objectives gives a value of 0.16, i.e. the objectives are not linearly correlated. This is therefore a moderately difficult multiobjective problem.

A set of 20 different route planning problems were defined over this modified map of New York, selecting random origin and destination nodes using an uniform distribution.

[10]A file in DIMACS format with the calculated travel costs is available from the author

Name	Location	Nodes	Arcs	ρ	Avg. $\lvert C^* \rvert$
DC	Washington D.C.	9,559	39,377	0.99	3.33
RI	Rhode Island	53,658	192,084	0.99	9.44
NJ	New Jersey	330,386	1,202,458	0.99	10.66

Table 3.5: Modified Road networks from Tiger/Line Files of DIMACS challenge.

3.2.4 Realistic Route planning problems: modified DIMACS maps

A set of random route planning problems in road maps was proposed by Raith & Ehrgott (2009) [11] and later used in the thesis of Raith (2009). These comprise nine random problem instances for each one of three different maps: Washington D.C. (DC), with 9,599 nodes; Rhode Island (RI), with 53,658 nodes; and New Jersey (NJ) with 330,386 nodes. Two objectives were considered by Raith & Ehrgott (2009): travel time and physical distance.

The maps files were taken from the assembly of Tiger/Line ® data of (U.S. Census Bureau, 2002) made by Schultes (2005) for the DIMACS challenge [12], and were slightly modified:

- The original data arcs were duplicated to obtain an undirected graph.
- These maps represent cuts of a real U.S. map. Each map corresponds to a different state. Therefore, there is no guarantee that all nodes are connected and some arcs can contain the value (0,0). A Hamiltonian cycle with "high" cost values (10,000) for both time and distance was added in order to ensure connectedness (specially in the case of RI map).

In this thesis, some adaptations were performed over these files prepared by Raith & Ehrgott (2009). In particular, arcs with values (0,0) should be in general avoided. Preliminary tests showed that the arcs with cost (10000,10000) in the Hamiltonian cycle and some of the arcs with cost (0,0) can influence the number of Pareto-optimal solution paths. In order to obtain the same number of efficient solutions reported by Raith & Ehrgott (2009); Raith (2009), we remove only arcs with cost $(0, 0)$ when the origin and destination node is the same[13]. Besides, the additional arcs from the Hamiltonian cycle with values (10000,10000) were also preserved in the data files used in this thesis.

More information about the maps can be found in table 3.5. The computation of Pearson's coefficient over the two objectives gives a value of $\rho = 0.99$ for all maps, i.e. there is a strong linear correlation between both objectives.

[11] The road maps problem set and data files were kindly provided by Andrea Raith

[12] `http://www.dis.uniroma1.it/~challenge9/data/tiger/`

[13] There are some remaining cycles in the maps between two nodes, represented by two reciprocal arcs of cost $(0, 0)$. Therefore, the actual number of arcs in the data files used in this thesis is slightly different from the numbers presented by Raith & Ehrgott (2009) and shown in table 3.5. However, results are not affected by this particular detail.

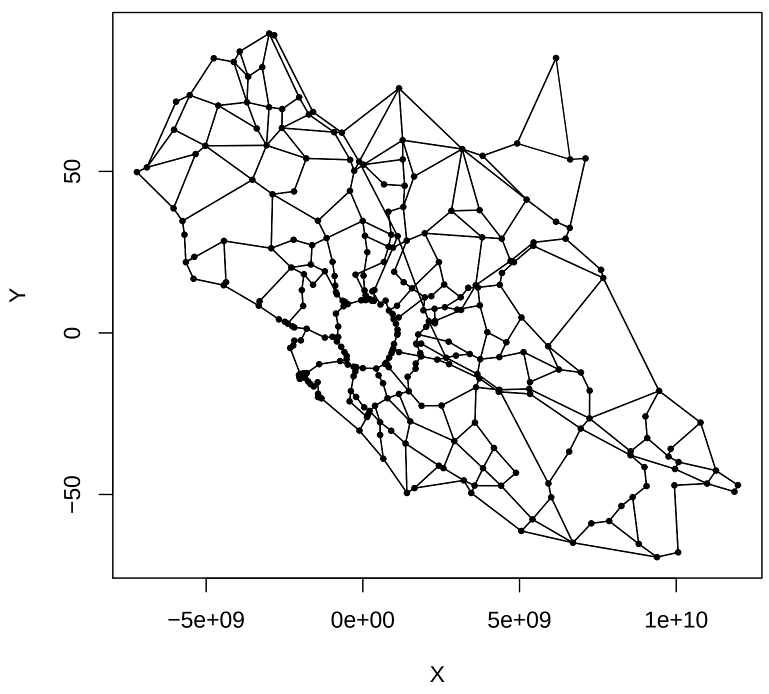


Figure 3.3: Geo-referenced graph of the Lazio region in Italy.

3.2.5 Realistic Hazmat Transportation Problems

Hazardous material transportation (*hazmat*) is another important practical problem recurrently found in the literature on shortest paths. The selection of optimal routes inherently involves the consideration of multiple conflicting objectives. These include the minimization of risk (e.g. the exposure of the population to hazardous substances in case of accident), transportation cost, time, or distance.

In this thesis, a set of 50 problem instances with random source and destination nodes was generated over a real road network of the Italian region of Lazio used by Caramia et al. (2010). Figure 3.3 shows a rendering of the map with its 311 (georeferred) nodes and 879 arcs. Each arc is labelled with a vector of three costs, which represent values of distance (in meters), time (in seconds) and societal risk (defined as the "product between the population inside the impact zone and the incident probability") obtained from data in the original files of Caramia et al. (2010) [14]. Three additional road networks were derived, each one with a different combination of pair of objectives. The same set of source and destination nodes was used. The computation of Pearson's correlation coefficient over pairs of two objectives can be observed in table 3.6.

3.2.6 Significance of the test sets

For grid problems, the grid sizes and number of Pareto-optimal solution paths are in the range of test instances of (Raith & Ehrgott, 2009; Caramia et al., 2010) and (Martins

[14]The data files were kindly provided by Antonio Iovanella

Name	Objectives	Nodes	Arcs	ρ	Avg. $\lvert C^*\rvert$
Lazio	Time, distance & societal risk	311	879		3.96
	Time & distance	311	879	0.99	1.06
	Time & societal risk	311	879	-0.18	3.92
	Distance & societal risk	311	879	-0.18	3.36

Table 3.6: Road networks for hazmat problems.

et al., 2007; Paixão & Santos, 2007) (in the biobjective case) for example. However, larger number of nodes and arcs and a higher number of nondominated solutions have been used by some authors (for example with more than two criteria (Martins et al., 2007; Paixão & Santos, 2007).

For random graphs, the parameters evaluated (number of nodes, density, range of cost values and number of criteria) are only in the range of some papers. There are authors who have used larger random graphs with higher number of nodes, arcs and Pareto-optimal paths (de Lima Pinto et al., 2009; Guerriero et al., 2001) or even a larger number of criteria (Martins et al., 2007; Paixão & Santos, 2007). However, the set of problems is enough for the experimentation purposes: the selected random graphs are used only to compare the impact on the algorithm performance of the number of criteria.

For realistic test sets several different scenarios have been included: large sized multiobjective road maps, difficult multiobjective problems by uncorrelated objectives and hazmat problems with more than two objectives. The combination of all these scenarios and the dimension of parameters evaluated are enough to be significant test cases to the date of presentation of these results.

3.3 Performance Evaluation of Multiobjective Search

In the experiments presented in this thesis we try to characterize the performance of the algorithms with respect to a number of problem dependent factors. These include solution depth, correlation between objectives (Mote et al., 1991; Brumbaugh-Smith & Shier, 1989), or the presence/absence of heuristic information.

It is important to note that in multiobjective search algorithms, the number of nodes considered is no more a significant performance measure. Most analyses concentrate on the number of iterations or distinct labels expanded by the algorithm. However, formal analyses (Stewart & White, 1991; Mandow & Pérez de la Cruz, 2010a) do not characterize whether differences in the number of expansions are significant for time performance in practice.

Thus, recent experimental evaluations in the multiobjective field (Raith & Ehrgott, 2009; Sauvanet & Néron, 2010) include both space and time performance in the comparison of algorithms. Moreover, the nature of the problem may require the observation of some other dimension, as the study from Iori et al. (2010) suggests for the blind case, where the number of comparisons performed by the algorithms are considered as well.

Besides problem dependent factors, there are also other implementation dependent issues, most notably the selection strategy for $OPEN$ alternatives. Among different possibilities, lexicographic order is the most usual one, but other are possible, e.g. linear rules like those explained in section 2.2.1 or the one explained in section 2.4.3 for TC algorithm. The evaluation of alternative orders has been recently considered (Iori et al., 2010) as an important performance issue. Additionally, in the literature, several authors have analyzed different strategies for the management of the $OPEN$ alternatives (Brumbaugh-Smith & Shier, 1989; Guerriero & Musmanno, 2001; Paixão & Santos, 2007).

Some other particular details of the implementation are important as well for the comparison of time performance,

- Architecture of the machine where the test is run, e.g. type of machine/server, multicore, multithread, etc.
- Speed of the processor, and number of simultaneous thread executions
- Amount of physical memory available (in Gigabytes) in the system and the amount of memory available to the program
- Programming language used, e.g. LISP or C++. The 9th DIMACS implementation challenge suggested the use of a public reference code and the publication of the set of instances considered, in order to be able to use the very same set of instances and a public code as benchmarking reference. Some authors have followed this suggestion[15]. The set of instance problems used in this thesis will be then publicly available [16]
- Implementation of the $OPEN$ queue, e.g. whether a binary heap is used or not, or whether only the current best cost estimate of each node was kept in $OPEN$ at each iteration, as suggested by Mandow & Pérez de la Cruz (2005)
- Implementation of the G_{op} (and G_{cl}), e.g. whether sets were ordered (or not) according to their respective linear evaluation functions like in TC algorithm or whether these sets were implemented as unordered lists or another data structure
- The implementation of "merge" and "prune" operations, i.e. the strategies used for the comparison of new alternatives against known labels of the node (Skriver & Andersen, 2000; Raith, 2009; Iori et al., 2010)
- The particular instances and/or source/goal nodes used. Paixão & Santos (2007) remark that a statistical analysis performed by (Santos et al., 2005) suggests that a minimum number of 50 instances should be taken for each class of graph tested

Concerning to the practical implementation of algorithms in this thesis,

- The algorithms $NAMOA^*$, MOA^* and TC were implemented to share as much code as possible. The programming language used is ANSI Common Lisp. Each problem instance was solved using an individual process with a single thread.

[15]For example, a public reference code and the set of test instances used by Martins et al. (2007) are publicly available at `http://www.mat.uc.pt/~zeluis/INVESTIG/MSPP/mspp.htm`

[16] `http://alef.iaia.lcc.uma.es/projects/alef-public/wiki`

- The $OPEN$ lists were implemented as binary heaps but only the current best cost estimate of each node was kept in $OPEN$ at each iteration, as suggested by Mandow & Pérez de la Cruz (2005)
- Unless explicitly noted, lexicographic order was used to choose among nondominated open alternatives in $NAMOA^*$ and MOA^* (see sections 2.4.1 and 2.4.2)
- The G_{op} sets for $NAMOA^*$ and G sets for MOA^* were also lexicographically ordered (unless explicitly noted)
- TC use the heuristic rule described in (2.23), see section 2.4.3
- The G_{op} sets for TC were ordered according to the linear evaluation function described by (2.23) in section 2.4.3

Part II

Formal & Empirical Analyses

This second part comprises both formal and empirical analyses performed through this research work, and contains all the contributions of this thesis. On the one hand, formal analyses sum up the previous theoretical results found for the three algorithms, and completes the characterization of MOA^* and $\vec{h}_{TC}$ heuristic. On the other hand, empirical analyses comprise the evaluation of existing algorithms with known heuristics, and the development of improvements over algorithms and heuristics.

- Chapter 4 analyzes the theoretical performance of $NAMOA^*$, MOA^* and TC. The chapter formally proves a worse behaviour of MOA^* with perfect heuristic information when compared to blind search. The heuristic function $\vec{h}_{TC}$ is also formally shown in this chapter to be consistent and therefore relevant for multiobjective search.
- Chapter 5 empirically tests the actual performance of the three algorithms over several benchmarks previously described in chapter 3, both in the case of blind search and heuristic search, providing a better understanding of the best alternative.
- Chapter 6 provides a bounded calculation method for the $\vec{h}_{TC}$ heuristic which saves precomputation effort. Besides, multiobjective heuristic search is evaluated on potential realistic scenarios, like route planning in road maps and hazmat problems.
- Chapter 7 develops more informed heuristics by using multiple heuristic estimates. A new precalculation method is presented and a comprehensive empirical evaluation is provided.

Chapter 4

Formal Analysis on Multiobjective Algorithms

4.1 Introduction

The A^* algorithm (Hart et al., 1968) is a heuristic shortest path algorithm with important formal properties. Particularly, the algorithm is admissible when provided with optimistic heuristic cost estimates. When these estimates are also consistent, more informed heuristics always result in equally or more efficient search (Pearl, 1984). In the absence of heuristic information (uninformed search), A^* performs like Dijkstra's algorithm.

Moreover, when the heuristics are monotone, the A^* algorithm is optimal in the number of steps to reach the solution. Any algorithm in its class must expand at least the same labels. The properties of A^* are summarized in section 2.1.2.1.

Three multiobjective counterparts to A^* have been presented in section 2.4, namely MOA^*, $NAMOA^*$ and TC algorithms. Their differences mainly lie in their label selection procedure and filtering/pruning processes. While MOA^* maintains a list of open nodes, similar to A^*, TC and $NAMOA^*$ maintain a list of open labels, and select them individually. When a node is selected in MOA^*, all labels associated to that node are considered for expansion. $NAMOA^*$ accept any selection procedure of nondominated alternatives while TC has a particular scalar selection rule.

These three algorithms have reached different degrees of formal characterization,

Admissibility Theoretical proofs on the admissibility of search were presented for MOA^* (Stewart & White, 1991), and recently also for $NAMOA^*$ (Mandow & Pérez de la Cruz, 2006, 2010a). On the contrary, no theoretical proofs on the admissibility of TC were presented by Tung & Chew (1988, 1992), but a simple justification.

Optimality A recent formal analysis argued that a relevant measure in the efficiency of multiobjective search is the number of explored labels, and showed that $NAMOA^*$ is optimal according to this measure when used with consistent heuristics (Mandow & Pérez de la Cruz, 2010a). The same result does not apply to MOA^*, even nor to TC algorithm. TC shares with $NAMOA^*$ a label-selection strategy. However, none of the paths expanded by $NAMOA^*$ can be avoided in an admissible search.

Efficiency In the case of $NAMOA^*$, it has been shown that the use of more informed consistent heuristics results in an equal or smaller number of label expansion operations by the algorithm (Mandow & Pérez de la Cruz, 2010a). This result is analogous to the properties of A^*. However, regarding MOA^* no equivalent result has ever been presented. In fact, section 4.3 shows that this property does *not* hold in general for MOA^*. A class of problem instances is described in section 4.4 where the performance of MOA^* (measured by the number of expanded labels) can degrade considerably with more informed heuristics. A similar result was recently presented for the blind case ($H(n) = \vec{0}, \forall n \in N$) for an analogous family of instance problems (Mandow & Pérez de la Cruz, 2010b). Regarding TC algorithm, the authors did not devise it for blind search. The precalculated heuristic multiobjective function introduced by (Tung & Chew, 1992) is analytically examined in section 4.7. The performance of this heuristic algorithm is expected to be similar to $NAMOA^*$ heuristic version. However, some details in its conception (see section 2.4.3) are revealed to cause a worse performance than $NAMOA^*$.

This chapter is organized as follows. Section 4.2 recalls formal aspects of $NAMOA^*$. The same properties are analyzed for the case of MOA^* in section 4.3 and the formal characterization of this algorithm is completed in the subsequent sections. A class of simple multiobjective search problems is presented in section 4.4. The performance of MOA^* over this class of problems is analyzed for the blind and perfectly informed cases in sections 4.5 and 4.6 respectively. The second part of the chapter, section 4.7, involves the study of TC algorithm and its original heuristic function. The chapter is concluded by a discussion.

4.2 Formal characterization of $NAMOA^*$

4.2.1 Admissibility

The algorithm A^* is said to be admissible under reasonable conditions (see section 2.1.2.1), as it finds the optimal solution whenever a solution exists. For multiobjective search, the concept must be generalized.

Definition 4.1 *(Mandow & Pérez de la Cruz, 2006, p. 184) A heuristic function $H(n)$ is said to be admissible when for all nondominated solutions $P^* = (s = n_0, n_1, \ldots, n_i, n_{i+1}, \ldots, n_l = \gamma_l)$, being $\gamma_l \in \Gamma$, and for all subpaths $P_i^* = (n_0, \ldots, n_i)$ of P^* the following holds,*

$$\exists \vec{h} \in H(n_i) \quad | \quad \vec{g}(P_i^*) + \vec{h} \preceq \vec{g}(P^*) \tag{4.1}$$

Property 4.1 (Admissibility) *When the graph $G = (N, A)$ is locally finite and $H(n)$ is a lower bound (admissible) the search is considered **admissible**, i.e. it is guaranteed* *to find* ***all nondominated*** *optimal solutions, or does not terminate if there are infinite solutions.* $NAMOA^*$ *is admissible even on infinite graphs with some additional assumptions:*

$$\begin{aligned} &\forall n \in N \wedge \forall \vec{h} = (h_1, \ldots, h_q) \in H(n), \quad \forall k \in [1, q],\ h_k(n) \geq 0 \\ &\forall (n, n') \in A, \wedge \forall \vec{c}(n, n') \in \vec{c}, \quad \forall k \in [1, q],\ c_k(n, n')\ \geq \epsilon > 0 \end{aligned} \tag{4.2}$$

These property of $NAMOA^*$ relies on these assumptions and the following theoretical results:

Theorem 4.1 *(Mandow & Pérez de la Cruz, 2010a, Theorem 4.2) For each nondominated solution path* $P^* = (s, n_1, \ldots, n_i, n_{i+1} \ldots \gamma)$ *with cost* $\vec{g}(P^*) = \vec{c}^{\,*}$, *there is always before its discovery a subpath* $P_i^* = (s, n_1, \ldots, n_i)$ *of* P^* *such that:*

a) P_i^* *is recorded in SG*

b) $\vec{g}(P_i^*) \in G_{op}(n_i)$

c) $\exists \vec{f} \in F(P_i^*) \mid \vec{f} \preceq \vec{c}^{\,*}$

Theorem 4.2 *(Mandow & Pérez de la Cruz, 2010a, Theorem 4.3) If there is at least a solution path* P^*, *the algorithm terminates even on infinite graphs.*

Corolary 4.1 *(Mandow & Pérez de la Cruz, 2010a, Corollary 4.4) Whenever there is at least a solution path* P^*, *the set of nondominated solution costs* C^* *is finite.*

Lemma 4.1 *(Mandow & Pérez de la Cruz, 2010a, Lemma 4.5) Each path* $P \in \mathbb{P}_{sn}$ *selected from* $OPEN$ *for expansion satisfies upon selection that,*

$$\exists \vec{h} \in H(n) \quad | \quad \nexists \vec{c}^{\,*} \in C^*, \quad \vec{c}^{\,*} \prec \vec{g}(P) + \vec{h} \tag{4.3}$$

Theorem 4.3 *(Mandow & Pérez de la Cruz, 2010a, Theorem 4.6) A dominated solution can never be selected for expansion.*

Corolary 4.2 *(Mandow & Pérez de la Cruz, 2010a, Corollary 4.7) The set of found solution COSTS is at any time a subset of the set of all nondominated solution costs, i.e.* $COSTS \subseteq C^*$

Corolary 4.3 *(Mandow & Pérez de la Cruz, 2010a, Corollary 4.7) For each path P in SG whose evaluation vectors are dominated by some solution cost* $\vec{c}^{\,*} \in C^*$, *i.e.*

$$\forall \vec{f} \in F(P),\ \exists \vec{c}^{\,*} \in C^* \quad | \quad \vec{c}^{\,*} \prec \vec{f} \tag{4.4}$$

none of its possible extensions will be generated and stored in SG.

Theorem 4.4 *(Mandow & Pérez de la Cruz, 2010a, Theorem 4.9) Since* $NAMOA^*$ *satisfies all the above conditions,* $NAMOA^*$ *is admissible.*

4.2.2 Efficiency of heuristics

For the single objective case, Pearl (1984, pp.79-85) analyzed the effect of heuristic information on the efficiency of A^*. For the multiobjective case, the nature of multiobjective problems must be taken into account in order to make a parallel analysis. Some generalizations must be made,

Definition 4.2 *(Mandow & Pérez de la Cruz, 2010a, Definition 5.1) A path $P = (s = n_0, n_1, n_2, \ldots, n_k)$ is said to be* ***C-bounded*** *with respect to $H(n)$ (or C(H)-bounded) if for all subpaths $P_i = (n_0, n_1, \ldots, n_i)$ of P it holds that,*

$$\exists \vec{h} \in H(n_i) \quad | \quad \nexists \vec{c} \in C, \quad \vec{c} \prec \vec{g}(P_i) + \vec{h} \tag{4.5}$$

By definition, a C^*-bounded path will never be filtered (see Lemma 4.1 and Theorem 4.3). Therefore, such paths will be selected for expansion or pruned.

Definition 4.3 *(Mandow & Pérez de la Cruz, 2010a, Definition 5.6) A multiobjective heuristic function $H(n)$ is* ***consistent*** *if for all pairs of nodes n, n' in the graph, for all nondominated path between them $P = (n, \ldots, n')$, and for all heuristic cost vector $\vec{h}' \in H(n')$, the following condition holds,*

$$\exists \vec{h} \in H(n) \quad | \quad \vec{h} \preceq \vec{c}(P) + \vec{h}' \tag{4.6}$$

Definition 4.4 *(Mandow & Pérez de la Cruz, 2010a, Definition 5.7) A multiobjective heuristic function $H(n)$ is* ***monotone*** *when for all arcs (n, n') in the graph, the following condition holds,*

$$\forall \vec{h}' \in H(n') \quad \exists \vec{h} \in H(n) \quad | \quad \vec{h} \preceq \vec{c}(n, n') + \vec{h}' \tag{4.7}$$

The consistency and monotonicity properties are equivalent conditions for $H(n)$ (Stewart & White, 1991, Lemma 18). When the $H(n)$ function is monotone then it is also admissible (Stewart & White, 1991, Lemma 19).

Theorem 4.5 *(Mandow & Pérez de la Cruz, 2010a, Theorem 5.9) If $H(n)$ is consistent, then a* ***necessary and sufficient*** *condition for NAMOA to select some path $P = (s, \ldots, n)$ for expansion is that:*

a) *P be a nondominated path from s to n*

b) *P be C^*-bounded*

Definition 4.5 *A heuristic function $H_2(n)$ is said to be* ***at least as informed*** *as other $H_1(n)$ when both are admissible and for all nodes n,*

$$\forall \vec{h}_2 \in H_2(n) \quad \exists \vec{h}_1 \in H_1(n) \quad | \quad \vec{h}_1 \preceq \vec{h}_2 \tag{4.8}$$

Theorem 4.6 *(Mandow & Pérez de la Cruz, 2010a, Theorem 5.10) Let $H_1(n)$ and $H_2(n)$ be two admissible heuristics for the same problem. Let $H_2(n)$ be additionally* ***monotone****. Let $NAMOA_1^*$ and $NAMOA_2^*$ be two versions of $NAMOA^*$ that differ only in the use of different heuristic functions $H_1(n)$ and $H_2(n)$ respectively. If $H_2(n)$ is at least as informed as $H_1(n)$, then* ***all*** *paths selected for expansion by $NAMOA_2^*$ will also be selected for expansion by $NAMOA_1^*$.*

Property 4.2 (Efficiency) *When $\forall n \in N$, $H(n) = \{\vec{0}\}$, $NAMOA^*$ is analogous to the blind algorithm of Martins (1984c) or Raith (2009). When $H(n)$ is* ***consistent*** *or* ***monotone****, only the strictly necessary C^*-bounded paths will be expanded, and the pruning of those C^*-bounded paths not belonging to nondominated solutions will be maximal, analogously to the single-objective case. If the costs of some optimal solution is denoted by vectors $\vec{c}^{\,*}$, $NAMOA^*$ will always expand for sure all labels with some $\vec{f}(n) \prec \vec{c}^{\,*}$. Given consistent heuristic functions, more actual suboptimal alternatives can be pushed out the frontiers $\vec{f}(n) = \vec{c}^{\,*}$ for all $\vec{c}^{\,*}$ (i.e. out of $F(n) = C^*$, the Pareto-frontier of efficient optimal solutions) with* ***more informed*** *heuristics, reducing search effort.*

4.2.3 Optimality

For the single objective case, there is no optimal algorithm in the number of expansion operations (Mérõ, 1984) nor in the distinct nodes expanded (Dechter & Pearl, 1985). However, A^* is shown to be optimal over admissible algorithms by both measures when applied with **consistent** heuristics.

For multiobjective problems, the set of expanded nodes is no longer a significant measure. There can be many different nondominated paths thay may reach each node in the graph. Thus, the analysis has to be made in terms of the number of path expansions (Mandow & Pérez de la Cruz, 2010a).

Let M_{ad} be the class of admissible multiobjective search algorithms. These are compared in terms of path expansions of previously generated paths emanating from source node s. This excludes other strategies like bidirectional search. Each instance problem can be defined by a cuadruple $I = (G, s, \Gamma, H)$. Let also $\mathbb{P}^{C^*}$ be the set of paths expanded by $NAMOA^*$.

Definition 4.6 *(Mandow & Pérez de la Cruz, 2010a, Definition 6.1) An algorithm $A \in M_{ad}$ is said to* ***dominate*** *another algorithm $B \in M_{ad}$ when for every instance I, the set $\mathbb{P}_A$ of expanded paths by A is a subset of the set $\mathbb{P}_B$ of expanded paths by B, i.e. $\mathbb{P}_A \subseteq \mathbb{P}_B$.*

Definition 4.7 *(Mandow & Pérez de la Cruz, 2010a, Definition 6.2) An algorithm A is* ***optimal*** *over M_{ad} when dominates every algorithm $B \in M_{ad}$. Algorithm A is* ***nondominated*** *over M_{ad} if no member of M_{ad} dominates A.*

Theorem 4.7 *(Mandow & Pérez de la Cruz, 2010a, Theorem 6.4) Any algorithm $A \in M_{ad}$ expands for each instance I all paths in $\mathbb{P}^{C^*}$, when the heuristic evaluation functions $H(n)$ are* ***consistent****.*

Property 4.3 (Optimality) *When the heuristic evaluation functions $H(n)$ are* ***monotone****, the cost of a path found by the algorithm is known to be optimal and the label (path) found is consider permanent and does not need to be reopened. $NAMOA^*$ is then* ***optimal*** *among the class of admissible best-first algorithms M_{ad}.*

4.3 Formal characterization of MOA^*

4.3.1 Admissibility

Analogous assumptions were made by Stewart & White (1991) to the characteristics of the graph G, the cost vectors $\vec{c}(n, n')$ and the heuristic functions $H(n)$.

Property 4.4 (Admissibility) *When the graph $G = (N, A)$ is locally finite, the costs vectors are positive and bounded, there are no cyclic paths and $H(n)$ is a lower bound (admissible), the search with MOA^* is considered **admissible**, even on infinite graphs.*

This property relies on some analogous theoretical results to those presented for $NAMOA^*$.

Definition 4.8 *(Stewart & White, 1991, p. 789) Let $\mathbb{P}^*{}_{sn}$ be the set of all efficient paths between s and node n. The set of all nondominated path costs between s and node n, $G^*(n)$ is such that*

$$G^*(n) = \{c(P) \mid P \in \mathbb{P}^*{}_{sn}\} = nd\{c(P) \mid P \in \mathbb{P}_{sn}\} \tag{4.9}$$

Lemma 4.2 *(Stewart & White, 1991, Lemma 12) For each non-dominated solution path $P^* = (s, n_1, \ldots, n_i, n_{i+1} \ldots \gamma)$ with cost $\vec{g}(P^*) = \vec{c}\,^*$, there is always before its discovery a subpath $P_i^* = (s, n_1, \ldots, n_i)$ of P^* such that:*

a) *P_i^* is recorded in SG*

b) $\vec{g}(P_i^*) \in G(n_i)$

c) $c(P^*) \in G^*(n)$

Theorem 4.8 *(Stewart & White, 1991, Theorem 2) If there is at least a solution path P^*, the algorithm is complete and returns a solution path.*

Corolary 4.4 *(Stewart & White, 1991, Corollary 1) Whenever each solution path P^* is finite, the set of nondominated solution costs C^* is finite. Moreover, the sets $H^*(n), G^*(n), F^*(n), \mathbb{P}^*{}_{s\Gamma}$ are also finite.*

Lemma 4.3 *(Stewart & White, 1991, Lemma 14) Each node selected from $OPEN$ for expansion by MOA^*satisfies upon selection that,*

$$\exists \vec{f} \in F(n) \quad | \quad \nexists \vec{c}\,^* \in C^*, \quad \vec{c}\,^* \prec \vec{f} \tag{4.10}$$

Theorem 4.9 *(Stewart & White, 1991, Theorem 7) Each node selected from $OPEN$ for expansion by MOA^*satisfies upon selection that,*

$$\exists (\vec{g} \in G^*(n) \wedge \vec{h} \in H(n)) \quad | \quad \nexists \vec{c}\,^* \in C^*, \quad \vec{c}\,^* \prec \vec{g} + \vec{h} \tag{4.11}$$

Lemma 4.4 *(Stewart & White, 1991, Lemma 15) A dominated solution can never be selected for expansion.*

Theorem 4.10 *(Stewart & White, 1991, Theorem 3) MOA^* is admissible whenever the set of heuristics $H(n)$ is an admissible set of heuristics.*

These properties are analogous to those found in $NAMOA^*$. However, some extra operations are carried out by MOA^*, and some properties analogous to those of A^* are not found in MOA^*. In particular, corollaries 4.2 and 4.3 do not apply to MOA^*. This implies that the set of solutions found so far (the set COSTS), and the sets of paths recorded in SG (i.e. the sets $G(n)$) can contain at some particular stage of execution some dominated members.

4.3.2 Optimality

The theorem 4.5 establishes the optimality of $NAMOA^*$ among M_{ad} over problems with consistent heuristics. No algorithm can skip a path expanded by $NAMOA^*$ without compromising admissibility. As the corollaries 4.2 and 4.3 do not apply to MOA^*, the set of paths expanded by $NAMOA^*$ is always a subset of those expanded by MOA^*, which is dominated by $NAMOA^*$.

4.3.3 Efficiency of heuristics

The analysis of Stewart & White (1991) established some properties of MOA^* in analogy to those of A^*. Some definitions are equivalent to those described for $NAMOA^*$ in section 4.2.2.

Definition 4.9 *(Stewart & White, 1991, p. 804) A path $P = (s = n_0, n_1, n_2, \dots, n_k)$ is said to be* ***C-nondominated*** *with respect to $H(n)$ if for all nodes n_i in P exists a path $P_i = (n_0, n_1, \dots, n_i)$ such that,*

$$\exists \vec{h} \in H(n_i) \quad | \quad \nexists \vec{c} \in C, \quad \vec{c} \prec \vec{c}(P_i) + \vec{h} \tag{4.12}$$

Definition 4.10 *(Stewart & White, 1991, p. 806) A set of heuristic functions is said to be* ***consistent*** *with respect to the associated cost and preference structure if for all pairs of nodes n, n' in the graph such that $n' \in \{N \backslash s\}$ and $n \in ANCS(n')$, and for all nondominated path between them $P = (n, \dots, n')$, and for all heuristic cost vector $\vec{h}' \in H(n')$, the following condition holds,*

$$\forall \exists \vec{h} \in H(n) \quad | \quad \vec{h} \preceq \vec{c}(P) + \vec{h}' \tag{4.13}$$

Definition 4.11 *(Stewart & White, 1991, p. 806) A set of heuristic functions is said to be* ***monotone*** *with respect to the associated cost and preference structure when for all pairs of nodes n, n' in the graph such that $n' \in \{N \backslash s\}$ and $n \in ANCS(n')$, the following condition holds,*

$$\forall \vec{h}' \in H(n') \quad \exists \vec{h} \in H(n) \quad | \quad \vec{h} \preceq \vec{c}(n, n') + \vec{h}' \tag{4.14}$$

Lemma 4.5 *(Stewart & White, 1991, Lemma 18) A set of heuristic functions is consistent if and only if it is monotone.*

Lemma 4.6 *(Stewart & White, 1991, Lemma 19) A monotone set of heuristic functions is also admissible.*

Theorem 4.11 *(Stewart & White, 1991, Theorem 7) If $H(n)$ is monotone, then a* ***necessary and sufficient*** *condition for MOA^* to select some node n for expansion is that:*

> **a)** *exists a nondominated path P from s to n, i.e. $P \in G^*(n)$*
>
> **b)** *P be C^*-bounded*

This theorem is analogous to the Theorem 4.5 presented for $NAMOA^*$. The necessary and sufficient condition for *node* expansion in MOA^* is the existence of some C^*-bounded nondominated path to the node. But there can be multiple dominated paths also in the node. These conditions do not prevent that they are expanded by MOA^*, even with monotone heuristics. Moreover, the conditions for $NAMOA^*$ precise which C^*-bounded paths are inevitably expanded and which not.

Definition 4.12 *(Stewart & White, 1991, p. 802) A heuristic function $H_2(n)$ is said to be* ***at least as informed*** *as other $H_1(n)$ when both are admissible and for all nodes n,*

$$\forall \vec{h}_2 \in H_2(n) \quad \exists \vec{h}_1 \in H_1(n) \quad | \quad \vec{h}_1 \preceq \vec{h}_2 \tag{4.15}$$

Theorem 4.12 *(Stewart & White, 1991, Theorem 4) Let MOA_1^* and MOA_2^* be two versions of MOA^* that differ only in the use of different heuristic functions $H_1(n)$ and $H_2(n)$ respectively. If $H_2(n)$ is at least as informed as $H_1(n)$, then MOA_2^* is* ***nondominated*** *with respect to MOA_1^*.*

In such case, when two consistent functions H, H' are considered, if H is at least as informed at H', it was proven that the set of nodes expanded by MOA^* with H is a subset of those expanded with H' (Stewart & White, 1991, Theorem 4, p. 805). However, the authors recognized that nodes may be reopened even when the heuristic function is consistent (Stewart & White, 1991, p. 806), and that the set of expanded nodes is not a significant measure in the analysis of the performance of MOA^*. The theorem 4.12 can not tell *how many times* a node will be expanded with each heuristic.

An analysis on the number of label expansions is more adequate for multiobjective algorithms. Mandow & Pérez de la Cruz (2010a) analyzed the performance of $NAMOA^*$ in this sense and found that when the heuristic functions are consistent, each label is expanded by $NAMOA^*$ only once. Additionally, the set of labels expanded by $NAMOA^*$ with H is a subset of those expanded with H' (Theorem 4.6). The following sections show that this stronger result is *not* true in general for MOA^*.

4.4 A class of multiobjective search problems

Let us consider a family of graphs with a regular structure in levels described by the following rules,

- The graph has n levels.

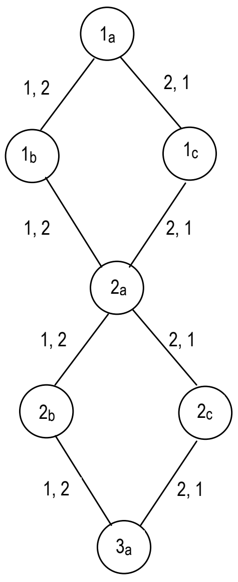


Figure 4.1: Graph $MC(2)$

- Each level i has 3 nodes, named i_a (up), i_b (left), i_c (right).
- The source node is 1_a, and the graph has exactly $3n + 1$ nodes including a goal node $(n + 1)_a$.
- The graph has $4n$ arcs,
  - For each level i, there are two arcs $i_a \rightarrowtail i_b$, $i_b \rightarrowtail (i + 1)_a$ labelled with a cost vector $(1, 2)$.
  - For each level i, there are two arcs $i_a \rightarrowtail i_c$, $i_c \rightarrowtail (i + 1)_a$ labelled with a cost vector $(2, 1)$.

We shall refer to these graphs as MC (multiobjective chain) graphs, and as $MC(n)$ to the MC graph with n levels. Graph $MC(2)$ is shown in figure 4.1.

Notice that the number of distinct paths in a MC graph reaching the goal node grows exponentially with n. However, the number of distinct costs of paths grows only linearly.

Lemma 4.7 *Let the only cost of node* 1_a *be* $(0, 0)$. *For a MC graph, the set of distinct cost vectors of paths reaching node* i_a *is,*

$$2(l + (i - 1), 2(i - 1) - l), \forall l, 0 \leq l < i$$

The combination of two $(1, 2)$ vector costs or two $(2, 1)$ vector costs at each level, i.e. $2i * (1, 2) + 2(n - i) * (2, 1), \forall i, 0 \leq i \leq n$ gives the former linear combination of costs.

Lemma 4.8 *All the solution paths in a MC graph are nondominated.*

In fact, from Lemma 4.7 it is obvious that all possible costs of paths reaching every node are nondominated (see figure 4.2).

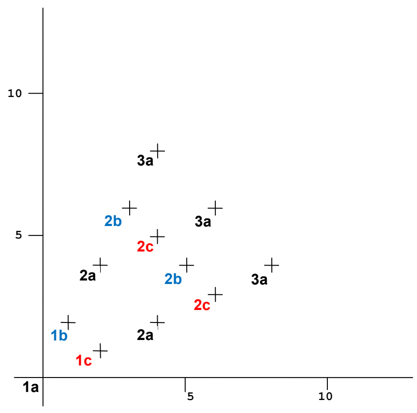


Figure 4.2: Nondominated costs of paths in Graph $MC(2)$

Theorem 4.13 *The minimum number of label expansions performed by any admissible multiobjective algorithm on a $MC(n)$ graph is given by the expression* [1]

$$3 \sum_{1 \leq i \leq n} i = \frac{3n(n+1)}{2} = \Theta(n^2).$$

The result follows from the fact that an admissible algorithm will need to consider all labels of all nodes, since all paths in a MC graph are nondominated by construction.

4.4.1 Examples

Two sample runs of MOA^* for $H(n) = \{\vec{0}\}$ (uninformed case) and $H(n) = H^*(n)$ (perfect information) are provided in tables 4.1 and 4.3 respectively. The node with the best lexicographic nondominated alternative fron ND is selected for expansion at each iteration. It is marked with $\leftarrow$ among all open nodes shown in column n. In the heuristic case, further ties are solved by selecting the node with lower level (breadth-first). Values of $G(n)$ include all nondominated costs from generated paths to the node. New discovered paths are indicated in bold face for $G(n)$. Values of $F(n)$ are the same as $G(n)$ in blind search, while they include the estimations in table 4.2 for the heuristic case. The last column amounts to the number of expanded labels, i.e. the cardinality of $G(n)$ at the moment of the node expansion.

We can observe in the uninformed case that i_a nodes from level $k+1$ are not expanded until nodes i_b, i_c from upper level k have been expanded. However, providing MOA^* with perfect heuristic information lets us see that i_a, i_b nodes are selected for expansion each time a new nondominated path is found to the node, while i_c (right) nodes are selected only when all nodes of upper levels have been expanded. The selection of a i_c (right) node triggers the reexpansion of all left and central nodes of subsequent levels.

In the presented example, breaking ties in favour of a deeper node (upper level) would even lead to a worse performance of heuristic MOA^*, with 12 iterations and

[1]We consider that labels at the goal node are selected but not expanded, i.e. their successor nodes and costs do not need to be calculated. Nevertheless, this does not change the overall result.

It	n	$G(n)$	$F(n)$	Lab exp
1	$1_a \leftarrow$	$((0,0))$	$((0,0))$	1
2	$1_b \leftarrow$	$((\mathbf{1,2}))$	$((1,2))$	1
	1_c	$((\mathbf{2,1}))$	$((2,1))$	
3	$1_c \leftarrow$	$((2,1))$	$((2,1))$	1
	2_a	$((\mathbf{2,4}))$	$((2,4))$	
4	$2_a \leftarrow$	$((2,4)(\mathbf{4,2}))$	$((2,4)(4,2))$	2
5	$2_b \leftarrow$	$((\mathbf{3,6})(\mathbf{5,4}))$	$((3,6)(5,4))$	2
	2_c	$((\mathbf{4,5})(\mathbf{6,3}))$	$((4,5)(6,3))$	
6	$2_c \leftarrow$	$((4,5)(6,3))$	$((4,5)(6,3))$	2
	3_a	$((\mathbf{4,8})(\mathbf{6,6}))$	$((4,8)(6,6))$	
7	$3_a \leftarrow$	$((4,8)(6,6)(\mathbf{8,4}))$	$((4,8)(6,6)(8,4))$	3

Table 4.1: Trace of uninformed MOA^* for the MC(2) graph. Contents of the OPEN list are displayed for each iteration.

i	i_a	i_b	i_c
1	((4,8),(6,6),(8,4))	((3,6),(5,4))	((4,5),(6,3))
2	((2,4),(4,2))	((1,2))	((2,1))
3	((0,0))	-	-

Table 4.2: Sets of heuristic cost values returned by heuristic function $H(n) = H^*(n)$ for all nodes in graph MC(2).

20 label expansions. This simple example suffices to prove that heuristic (informed) MOA^* can perform more label expansions than blind (uninformed) MOA^*. The performance of the algorithm for general $MC(n)$ graphs in the uninformed case is studied in section 4.5. Section 4.6 analyzes the heuristic case showing that, using perfect information, MOA^* can perform orders of magnitude worse than uninformed search.

4.5 Performance of uninformed MOA^*

This section characterizes the number of label expansions performed by uninformed MOA^* on an $MC(n)$ graph. Throughout this section we assume that $\forall n\ H(n) = \{\vec{0}\}$, and that the node with the lexicographic optimum among evaluation values in ND is selected for expansion at each iteration.

Lemma 4.9 *When MOA^* selects for expansion node $(k+1)_a$,*

- *it has already expanded all nondominated labels of all nodes at level k*
- *it has permanently closed all the nodes at all levels $j \leq k$, i.e. they are closed and will never again be put back into $OPEN$.*
- *it is the only open node, and all nondominated labels to $(k+1)_a$ have been generated.*

It	n	$G(n)$	$F(n)$	Lab exp
1	$1_a \leftarrow$	$((0,0))$	$(\underline{(4,8)}(6,6)(8,4)))$	1
2	$1_b \leftarrow$	$(\mathbf{(1,2)})$	$(\underline{(4,8)}(6,6)))$	1
	1_c	$(\mathbf{(2,1)})$	$((6,6)(8,4))$	
3	1_c	$((2,1))$	$((6,6)(8,4))$	
	$2_a \leftarrow$	$(\mathbf{(2,4)})$	$(\underline{(4,8)}(6,6)))$	1
4	1_c	$((2,1))$	$((6,6)(8,4))$	
	$2_b \leftarrow$	$(\mathbf{(3,6)})$	$(\underline{(4,8)})$	1
	2_c	$(\mathbf{(4,5)})$	$((6,6))$	
5	1_c	$((2,1))$	$((6,6)(8,4))$	
	2_c	$((4,5))$	$((6,6))$	
	$3_a \leftarrow$	$(\mathbf{(4,8)})$	$(\underline{(4,8)})$	1
6	$1_c \leftarrow$	$((2,1))$	$(\underline{(6,6)}(8,4))$	1
	2_c	$((4,5))$	$((6,6))$	
7	2_c	$((4,5))$	$((6,6)(8,4))$	
	$2_a \leftarrow$	$((2,4)\mathbf{(4,2)})$	$(\underline{(4,8)}(6,6)(8,4)))$	2
8	2_c	$((4,5)\mathbf{(6,3)})$	$((6,6)(8,4))$	
	$2_b \leftarrow$	$((3,6)\mathbf{(5,4)})$	$(\underline{(4,8)}(6,6))$	2
9	2_c	$((4,5)(6,3))$	$((6,6)(8,4))$	
	$3_a \leftarrow$	$((4,8)\mathbf{(6,6)})$	$(\underline{(4,8)}(6,6))$	2
10	$2_c \leftarrow$	$((4,5)(6,3))$	$(\underline{(6,6)}(8,4))$	2
11	$3_a \leftarrow$	$((4,8)(6,6)\mathbf{(8,4)})$	$(\underline{(4,8)}(6,6)(8,4))$	3

Table 4.3: Sample run of heuristic MOA^* with $H(n) = H^*(n)$ for MC(2) graph. Contents of the OPEN list are displayed for each iteration. The lexicographic optimum among $F(n)$ values responsible for selection is underlined.

Proof. By induction on the number of levels in the graph.
Base case: i=1. The expansion of 1_a generates one label $(1,2)$ to 1_b, and one label $(2,1)$ to 1_c. Node 1_b is lexicographically better than 1_c. Its expansion generates a label with cost $(2,4)$ to 2_a. However, node 1_c is still lexicographically better. It is expanded and generates a second label $(4,2)$ to node 2_a. Now, it is the turn of 2_a for expansion. Since all nodes in level 1 are closed, and no more paths reach these nodes, they will never be put back into $OPEN$.
Hypothesis. Let us assume that it is the turn of $(i+1)_a$ for expansion, that it is the only open node, that all nondominated labels to $(i+1)_a$ have been generated, and that i_a, i_b, i_c are now permanently closed and all their nondominated labels have been expanded.
Induction step. From Lemma 4.7, there are $i+1$ possible nondominated labels reaching $(i+1)_a$, given by $\{2(l+i, 2i-l), \forall l, 0 \leq l < i+1\}$

Upon expansion of $(i+1)_a$, its successors $(i+1)_b$ and $(i+1)_c$ are generated and opened with the following costs,

$$F((i+1)_b) = G((i+1)_b) = \{2(l+i, 2i-l) + (1,2), 0 \leq l \leq i\}$$

$$F((i+1)_c) = G((i+1)_c) = \{2(l+i, 2i-l) + (2,1), 0 \leq l \leq i\}$$

Since $F((i+1)_b)$ is lexicographically better, it is considered before for expansion, generating and opening $(i+2)_a$ with,

$$LABEL((i+2)_a, (i+1)_b) = \{2(l+i, 2i-l) + (2,4), 0 \leq l \leq i\}$$

and

$$F((i+2)_a) = G((i+2)_a) = LABEL((i+2)_a, (i+1)_b)$$

Since the new node is lexicographically worse than $(i+1)_c$, this will be expanded next, generating an additional set of labels to $(i+2)_a$ option,

$$LABEL((i+2)_a, (i+1)_c) = \{2(l+i, 2i-l) + (4,2), 0 \leq l \leq i\}$$

and therefore,

$$F((i+2)_a) = G((i+2)_a) = \{LABEL((i+2)_a, (i+1)_b) \cup LABEL((i+2)_a, (i+1)_c)\}$$

which is from Lemma 4.7 the set of all labels reaching $(i+2)_a$,

$$F((i+2)_a) = \{2(l+(i+1), 2(i+1)-l), \forall l, 0 \leq l < i+2\}$$

Since $(i+2)_a$ is the only open node and there are no paths leading back to nodes in previous levels, the result holds.

Theorem 4.14 *The number of label expansions performed by uninformed MOA^* on a $MC(n)$ graph is*

$$3 \sum_{1 \leq i \leq n} i = \frac{3n(n+1)}{2} = \Theta(n^2)$$

This follows from Lemma 4.9 since each node at level i is expanded only once, and at that time its $i+1$ labels have already been found. Therefore, from Theorem 4.13 it follows that uninformed MOA^* perfoms optimally in $MC(n)$ graphs in terms of label expansions, when nodes are selected lexicographically for expansion.

4.6 Performance of MOA^* with perfect heuristic information

Let us now consider the performance of MOA^* over MC graphs using perfect information. Throughout this section we shall assume that the heuristic function returns for each node the set of actual nondominated costs of paths from that node to the goal, i.e. $\forall n\ H(n) = H^*(n)$. By definition, this heuristic is admissible and consistent. A lexicographic order will be used for selection an expansion and nodes. In case of further ties the node at the lower level i will be selected (breadth-first tie-break rule).

Let us consider now the set of solution costs described in Lemma 4.7 ordered lexicographically for the goal node $(n+1)_a$. We shall say that the lexicographic optimum solution cost corresponds to the first element in the set, the second best to the second element, and the i-th best to the i-th element.

Lemma 4.10 *For MC graphs with perfect heuristic information, when MOA^* expands for the first time the k-th solution label,*

1. *All nodes i_a, i_b are closed.*

2. *All nodes i_c with $i \leq k$ are closed.*

3. *All nodes i_a, i_b with $i \leq k$, and i_c with $i < k$ will not be reopened (are permanently closed).*

4. *All nodes i_c with $i \geq k$ are opened and have not been expanded yet.*

5. *The next node selected for expansion is k_c and will reopen $(k+1)_a$ (except when $k = n+1$, since search is terminated).*

Proof. By induction on the number of solutions.
Base case (k=1) All nodes i_a, i_b are traversed by the solution path with the lexicographic optimum cost. All nodes i_c are traversed, in the best case, by a path including exactly two arcs with cost $(2,1)$ and $2n-2$ arcs with cost $(1,2)$, which is the second best among solution costs in lexicographic order. Let us denote by $\vec{f}^{*1}$ the lexicographic optimum solution cost, and by $\vec{f}^{*2}$ the cost of the second best lexicographic solution cost.

MOA^* will start expanding nodes in the sequence $1_a 1_b 2_a 2_b \dots n_a n_b (n+1)_a$ finding the first solution, and expanding a single label for each node. For all nodes n in this set, $F(n) = \{\vec{f}^{*1}\}$. All nodes i_c are now open, and $\forall i$ $F(i_c) = \{\vec{f}^{*2}\}$. No new paths reach 1_a or 1_b, so they are permanently closed. Since, by assumption, ties are broken in favor of shallower nodes, node 1_c will be the next to be selected for expansion. This will generate a path with a new cost $(2,4)$ to node 2_a, which will be added to $G(2_a)$ causing this node to be reopened with two labels.
Hypothesis Let us assume the property holds after the k-th solution has been found.
Induction step Node k_c will be the next to be selected for expansion. Since all nodes above k_c are permanently closed, all nondominated labels to k_c have been found. Its expansion will generate a new label to node $(k+1)_a$, in particular the one corresponding to a path that reaches that node through the sequence $1_a 1_c 2_a 2_c \dots k_c (k+1)_a$, i.e. the addition of $2k$ arcs with cost $(2,1)$. This will be added to $G((k+1)_a)$ which will be placed back into OPEN with *all* its labels. Now, all nondominated labels to node $(k+1)_a$ have been found. Since $\vec{f}^{*1} \in F((k+1)_a)$, this will trigger the reexpansion of the sequence of nodes $(k+1)_a (k+1)_b (k+2)_a (k+2)_b \dots (n+1)_a$, each one with $n+1$ labels. Note that the new label is unavoidably propagated through this sequence, reopening nodes in turn. Each time a node in this sequence is selected for expansion, the next is reopened and selected, since for all nodes n in this sequence $\vec{f}^{*1} \in F(n)$. At this point, the (k+1)-th solution is found, and the lemma holds.

Lemma 4.11 *When MOA* is applied to a MC graph with perfect heuristic information, nodes are selected lexicographically, and additional ties are broken using the breadth-first rule, each node i_a, i_b is expanded exactly i times, while nodes i_c are expanded exactly one time.*

This follows from Lemma 4.10, since each node i_a i_b is expanded to find the first solution, and then reopened and expanded once more after the expansion of each j_c with $j < i$. On the other hand, the i_c nodes are expanded only once, since at the time of their expansion all shallower nodes are permanently closed.

Theorem 4.15 *When MOA^* is applied to a MC graph with perfect heuristic information, nodes are selected lexicographically, and additional ties are broken using the breadth-first rule, the total number of label expansions is*

$$\frac{n(n+1)(2n+7)}{6} = \Theta(n^3)$$

Proof. The total number of individual labels expanded by MOA^* sums up,

1. expansions corresponding to i_c (right) nodes, i.e.,

$$\sum_{1\le i\le n} i = \frac{n(n+1)}{2}. \tag{4.16}$$

2. expansions corresponding to i_a (up) and i_b (left) nodes, i.e,

$$2\sum_{1\le i\le n}\sum_{1\le j\le i} j = 2\sum_{1\le i\le n}\frac{i(i+1)}{2} =$$

$$= \sum_{1\le i\le n} i(i+1) = \sum_{1\le i\le n} i^2 + \sum_{1\le i\le n} i$$

that is,

$$\frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2} \tag{4.17}$$

Finally, the sum of equations (4.16) and (4.17) gives the desired result,

$$\frac{n(n+1)(2n+7)}{6} = \Theta(n^3)$$

A comparison of this result with Theorem 4.14 indicates that the number of label expansions performed by MOA^* on MC graphs is clearly worse with perfect heuristic information than with uninformed search.

The formal analysis presented above covers two extreme cases: uninformed search, and search with perfect heuristic information. Figure 4.3 displays the number of label expansions in a $MC(10)$ graph with different degrees of heuristic precision. In particular, vector estimates in $H^*(n)$ were multiplied by a constant $k \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. Results show that rather than improve, the number of label expansions actually grows with improved heuristic precision. The figure additionally shows the results obtained with a "depth-first" tie breaking rule. As noted in section 4.4.1, performance is even worse with that rule.

4.6.1 Summary

The first part of the chapter considers the theoretical performance of multiobjective heuristic best-first algorithms and specially concentrates on the performance of the MOA^* multiobjective heuristic search algorithm. Results show that, in general, performance does not improve with heuristic information (a key result of A^* or $NAMOA^*$).

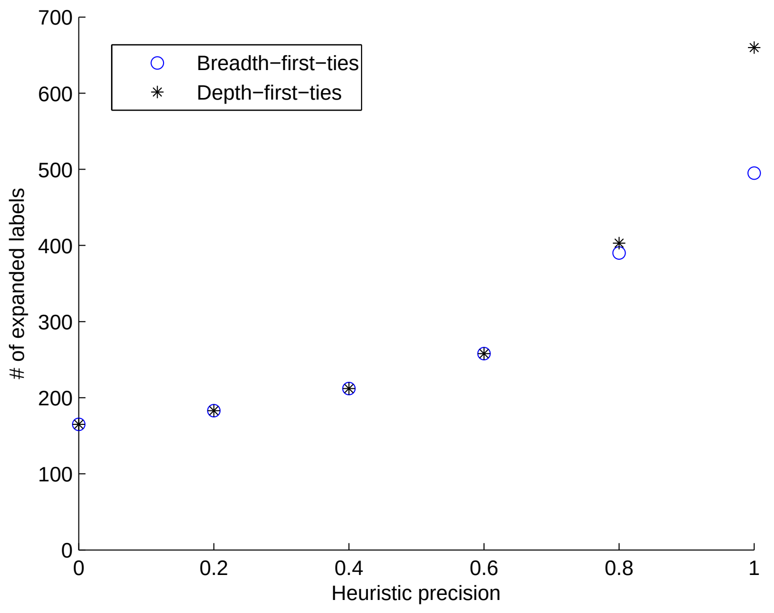


Figure 4.3: Label expansions of MOA^* in graph $MC(10)$ for different degrees of heuristic precision. Nodes are selected lexicographically, and values are shown for two different tie breaking rules.

A class of problems is presented (multiobjective chain graphs) where the use of perfect heuristic information (a trivially consistent informed heuristic) does not result in a reduction in the number of label expansions performed by the algorithm. In fact, the number of such expansions grows substantially as heuristic information improves, when compared to uninformed search.

Multiobjective chain graphs (MC) formalize a common situation in practical multiobjective search, a sequence of nodes traversed by at least two conflicting paths. Nodes can be closed and remain far behind the search frontier awaiting reexpansion for a long time. Consider the case of some solution path $n_1, n_2, ...n_k$ quickly found by the algorithm. If a new nondominated label is found to some node n_i, then the node would be reopened and automatically selected for expansion, since it was already found best among $OPEN$ nodes in previous iterations. This produces a cascading effect prompting the reopening and expansion of all nodes in the sequence $n_i, n_{i+1}, ...n_k$. In fact, the better the heuristic estimates, the deeper a node can remain behind the search frontier before it is reexpanded and, in consequence, the worse the cascading effect.

The formal analysis has revealed that the number of nondominated labels in MC graphs grows quadratically with graph size, and that uninformed MOA^* expands each label exactly once. However, when MOA^* is combined with H^*, the best possible heuristic, the number of such expansions can grow cubically.

Previous results (Mandow & Pérez de la Cruz, 2010a) showed $NAMOA^*$ to dominate MOA^* in terms of label expansions. A simple example was presented where MOA^* performed more label expansions than $NAMOA^*$. The analysis presented in this chapter further shows that the performance of MOA^* is not in general asymptotically similar to that of $NAMOA^*$, even in simple common situations.

4.7 Formal characterization of TC

The algorithm presented by Tung & Chew (1988, 1992) is a best-first label-setting heuristic algorithm similar to $NAMOA^*$. Differences can be found in the filtering/pruning processes, as explained in section 2.4.3. Besides, a particular rule is followed for the selection of promising paths at each iteration. However, this does not affect the set of expanded labels (but only the order in which they are selected) and TC shares with $NAMOA^*$ the same *path expansion* policy. Therefore, some properties are also common to TC algorithm and do not need to be recalled. There is no need to reformulate theorems in terms of the TC algorithm. Nevertheless, it is important to note that $NAMOA^*$ has been shown to be optimal along its class (Theorem 4.7). This means that the expected performance for TC will be always lower than in $NAMOA^*$.

While the scalar heuristic h_{mix} of the particular selection rule of TC can not be used in general (e.g. for pruning/filtering with COSTS), the application of the $\vec{h}_{TC}$ heuristic functions can be introduced in sets $H(n)$ without problems and deserve special investigation. These are well designed general heuristics that can be used by multiobjective heuristic algorithms like $NAMOA^*$ in their $H(n)$ functions in order to improve performance.

However, Tung & Chew (1992) did not specified whether these heuristics were either consistent or monotone. The second part of this chapter is devoted to the study of these heuristics functions and their formal properties, in order to proof the actual interest of their application to real problems.

4.8 Monotonicity of $\vec{h}_{TC}$

A formal proof on the admissibility and monotonicity of $\vec{h}_{TC}$ heuristic should guarantee that results obtained with its application are expected to be correct and all nondominated optimal solution paths are guaranteed to be found with $NAMOA^*$. Monotonicity is a stronger constraint than admissibility. Besides, we know by Theorem 4.10 that if the $\vec{h}_{TC}$ heuristic is proven to be monotone, then it is trivially admissible. Therefore, only proofs about monotonicity of the $\vec{h}_{TC}$ heuristic function are presented in this section.

The formal proofs presented for the heuristic $\vec{h}_{TC}$ in this section are based on some basic assumptions/conditions:

- The graph G is locally finite
- It is possible to reverse each arc of the graph and search from a terminal node γ to node s following the arcs in the optimal paths on inverse direction, i.e. arcs are bidirectional
- The costs of an arc remain the same when the graph is reversed, i.e. the cost of a

path $P = \{n, \dots, n'\}$ in G is the same than the cost of the path $P^{rev} = \{n', \dots, n\}$ in a reversed graph G^{rev}.

The simple biobjective case ($q = 2$) is managed on these formal proofs for costs structure, i.e. the cost of an arc between nodes n, n' is a vector with two components $\vec{c}(n, n') = (c_1(n, n'), c_2(n, n'))$. The general case for more than two objectives can be easily extrapolated.

Definition 4.13 *(Definition (10) Tung & Chew, 1992) Let $q_i(n, n')$ be the minimum cost for the objective i among all paths joining nodes n and n',*

$$q_i(n, n') = \min c_i(\mathbb{P}_{nn'}) \tag{4.18}$$

The minimum cost $q_i(n, \gamma)$ has been informally presented in section 2.4.4 as $c_i^*(n)$. The TC algorithm uses a single vector heuristic $\vec{h}_{TC}$, i.e. this means that $\forall n \; H(n) = \{\vec{h}_{TC}(n)\}$. This vectorial heuristic can be defined for the bicriteria case as

$$\vec{h}_{TC}(n) = (q_1(n, \gamma), q_2(n, \gamma)) = (\min c_1(\mathbb{P}_{n\gamma}), \min c_2(\mathbb{P}_{n\gamma})) \tag{4.19}$$

Theorem 4.16 *The heuristic function $\vec{h}_{TC}$ is monotone.*

Proof. Monotonicity implies that the heuristic function $\vec{h}_{TC}$ must satisfy that

$$\forall (n, n'), \quad \vec{h}_{TC}(n) \preceq \vec{c}(n, n') + \vec{h}_{TC}(n') \tag{4.20}$$

Let us assume first that node n' is a direct successor of node n and that node n' belongs to an optimal path from n to γ regarding the first objective. Let us denote this path by $P_{n\gamma}^{*1} = (n, n', n'', \dots, n^k, \gamma)$. By definition (2.15), the cost of this path is

$$c(P_{n\gamma}^{*1}) = \vec{c}(n, n') + \vec{c}(n', n'') + \ldots + \vec{c}(n^k, \gamma)$$

and regarding the first objective,

$$c_1(P_{n\gamma}^{*1}) = c_1(n, n') + c_1(n', n'') + \ldots + c_1(n^k, \gamma)$$

By the definition of optimal path (regarding the first objective) the cost of $P_{n\gamma}^{*1}$ is

$$c_1(P_{n\gamma}^{*1}) = \min c_1(\mathbb{P}_{n\gamma})$$

and that is by the definition of $\vec{h}_{TC}$ the first objective in (4.19)

$$\min c_1(\mathbb{P}_{n\gamma}) = q_1(n, \gamma)$$

Thus, $q_1(n, \gamma) = c_1(P_{n\gamma}^{*1})$, that is,

$$q_1(n, \gamma) = c_1(n, n') + c_1(n', n'') + \ldots + c_1(n^k, \gamma)$$

and as we assume that node n' belongs to an optimal path from n to γ regarding the first objective,

$$q_1(n,\gamma) = c_1(n,n') + \min c_1(\mathbb{P}_{n\gamma})$$

that is, the result proofs the theorem for the first objective,

$$q_1(n,\gamma) = c_1(n,n') + q_1(n',\gamma)$$

Analogously, for the second objective the proof is similar and

$$q_2(n,\gamma) = c_2(n,n') + q_2(n',\gamma)$$

Let us assume now that node n' is a direct successor of node n and that the node n' does not belong to any optimal path from n to γ regarding the first objective. Let us denote again by $P^{*1}_{n\gamma} = (n, n'^{*}, n''^{*}, \dots, n^{k*}, \gamma)$ this optimal path regarding the first objective. However, we can find a path $P'_{n'\gamma} = (n', n'', \dots, n^{k}, \gamma)$ that is optimal for the first objective from n' to γ.

By definition of $\vec{h}_{TC}$ the first objective of the heuristic vector is

$$q_1(n,\gamma) = \min c_1(\mathbb{P}_{n\gamma}) = c_1(P^{*1}_{n\gamma})$$

and by definition, this path has the minimal cost among all paths in the graph (regarding the first objective),

$$c_1(P^{*1}_{n\gamma}) \leq c_1(\mathbb{P}_{n\gamma}), \quad \forall P_{n\gamma} \in \mathbb{P}_{n\gamma},\ P_{n\gamma} \neq P^{*1}_{n\gamma}$$

It is true also for $P'_{n'\gamma}$,

$$c_1(P^{*1}_{n\gamma}) \leq c_1(n,n') + c_1(P'_{n'\gamma})$$

We assume that path $P'_{n'\gamma}$ is the minimal regarding the first objective from n' to γ,

$$c_1(P^{*1}_{n\gamma}) \leq c_1(n,n') + \min c_1(\mathbb{P}_{n',\gamma})$$

and that means that this result proves the theorem for the first objective also,

$$q_1(n,\gamma) = c_1(P^{*1}_{n\gamma}) \leq c_1(n,n') + q_1(n',\gamma)$$

Analogously, for the second objective the proof is similar and

$$q_2(n,\gamma) \leq c_2(n,n') + q_2(n',\gamma)$$

Therefore, in both cases

$$\forall (n,n'), \quad \vec{q}(n,\gamma) \ \preceq\ \vec{c}(n,n') + \vec{q}(n',\gamma) \tag{4.21}$$

q.e.d.

4.9 Discussion

The results presented in sections 4.5, 4.6 complete the formal study of MOA^*. Important properties were presented for this algorithm by Stewart & White (1991), similar to those found in A^*. However, this chapter confirms what the authors recognize, that monotonicity and consistency are not as important for MOA^* as in the single-objective case. This chapter has shown that more informed heuristics may not lead MOA^* to an improved performance, as it happens to $NAMOA^*$.

These important theoretical results do not completely settle the question of which algorithm is better in practice. Regarding uninformed (blind) search, the adequacy of node selection strategies has been investigated by other authors for the case of uninformed label correcting algorithms (Guerriero & Musmanno, 2001; Raith & Ehrgott, 2009). Despite theoretical expected performance is lower in MOA^* than in $NAMOA^*$, a empirical study should confirm whether in fact this occurs in all situations. Apparently, the node-selection strategy in MOA^*, can compensate in some particular situations for the additional number of labels considered.

Besides, the TC heuristics have not been evaluated in practice. It is expected that the performance of $NAMOA^*$ may benefit from the use of these consistent well informed heuristics. The chapter 5 presents a detailed empirical analysis of the three algorithms with both blind search and using the TC heuristics for the heuristic case.

Chapter 5

Empirical Analysis on Multiobjective Algorithms

This chapter takes up the task of evaluating TC, MOA^* and $NAMOA^*$ algorithms over artificially generated grid problems as described in section 3.2.1. This allows the evaluation of the algorithms in terms of solution depth and correlation between objectives. Two different scenarios were considered for grid problems. In the first one, search is constrained by grid boundaries (class I grid problems) and in the second one no such constraint exists (class II grid problems). Performance is also evaluated over a realistic scenario, modified DIMACS road maps described in section 3.2.4.

Prior to this research no systematic evaluation and comparison of multiobjective heuristic search algorithms had been performed. This is one of the contributions of this thesis.

The empirical evaluation draws a clear picture on the performance of each algorithm, resulting in $NAMOA^*$ as the best choice in general. Several phenomena previously unnoticed in the literature are also described and analyzed in this chapter:

- The performance of MOA^* is found to degrade considerably with better heuristic information, which confirms in practice the bad results for this algorithm obtained in the formal analysis of chapter 4
- Finding solutions in early stages of multiobjective search is found to degrade the performance of all algorithms, but specially in the case of TC, which uses a particular heuristic in label selection (see section 2.4.3)
- The use of heuristic information does not always result in improved performance in certain classes of problems (class I grid problems)

These results have been reported at the European Journal of Operational Research (Machuca et al., 2012) and at the national portuguese and german AI conferences (namely EPIA and KI conferences, respectively). Preliminary results of the evaluation of $NAMOA^*$ and TC algorithm over class I grids with $\rho = 0$ were presented in EPIA'2009 (Machuca et al., 2009). The results presented in this chapter for blind MOA^* and $NAMOA^*$ with class I and class II grids can be found also in the proceedings of KI'2010 (Machuca et al., 2010).

The chapter is organized as follows. Section 5.1 describes the application of blind multiobjective search techniques on grid problems. Section 5.2 is devoted to the application of heuristic multiobjective algorithms on grid problems. Pair-by-pair comparisons of the three algorithms can be found in this section, as well as a *blind vs heuristic search* analysis. The phenomena related above are adequately explained in sections 5.3 and 5.4, respectively where the time performance of heuristic MOA^* on the one hand and heuristic $NAMOA^*$ and TC on the other hand are analyzed. An analysis of these heuristic algorithms on modified road maps is also contained in this chapter in section 5.5. Finally some conclusions about this empirical study can be found at the end of the chapter.

5.1 Blind multiobjective search on grid problems

This section deals with the two algorithms MOA^* and $NAMOA^*$. Algorithm TC was not devised to work in absence of heuristic information. Performance of both algorithms is compared over sets of randomly generated bicriterion grids, which allows to control difficulty through solution depth and correlation between objectives.

A detailed description of algorithms MOA^* and $NAMOA^*$ was presented in chapter 2 (sections 2.4.2 and 2.4.1 respectively). The analysis in this section allows the comparison of the main difference between both algorithms: MOA^* is built around the idea of *node* selection and expansion, while $NAMOA^*$ is built around the idea of *path* selection and expansion. The selection and expansion policy of MOA^* may seem more efficient at first sight, however it has a major drawback, since each time a new nondominated path is found to a closed node, the whole node needs to be put back into the open list.

Square grids of varying size were randomly generated as described in chapter 3. A vicinity of four neighbours was used. Only two costs (c^1_{ij}, c^2_{ij}) were considered for each arc from i to j. The values of these objectives were calculated in the range $[1, 10]$ using various correlation values, as described in section 3.2.1: $\rho \in \{$ 0.8, 0.4, 0, -0.4, -0.8$\}$.

Two different classes of problem instances (class I and class II) were used (see section 3.2.1 for further description). The algorithms were implemented in Lisp using LispWorks Professional 5.01, and run on a HP Proliant D160 G5 server with 2 Intel Xeon QuadCore 4572 @ 3GHz processors and 18 Gb of RAM under Windows Server 2008 (32-bit).

5.1.1 Results

Let t_{MOA} and t_{NAMOA} be the time taken to solve a problem by MOA^* and $NAMOA^*$ respectively. Let also v_{MOA} and v_{NAMOA} be the maximum number of cost vectors stored by both algorithms when solving a problem.

For class I problems, no significant difference was found regarding memory requirements, regardless of correlation values. The hardests problems took about $3,35 \times 10^6$ cost vectors. The relative performance in time t_{MOA}/t_{NAMOA} is presented in figure 5.1(a) as a function of solution depth d. The hardest problems took about 2303 s (MOA^* with $d = 160$) and 1590 s ($NAMOA^*$ with $d = 200$). A time-out of 3600 s (1 h) was set for the resolution of each problem.

Values are shown for all correlation ratios and averaged over ten problems generated for each depth, in all reported results in this analysis.

Figure 5.1(b) presents the relative space requirements v_{MOA}/v_{NAMOA} against solution depth d, for class II problems. The hardests problems took about $2,99 \times 10^6$ cost vectors ($NAMOA^*$) and $3,84 \times 10^6$ cost vectors (MOA^*).

Relative time performance t_{MOA}/t_{NAMOA} of the algorithms is presented in figure 5.1(c) as a function of solution depth d, for class II problems. The hardest problems took about 2531 s (MOA^* with $d = 90$) and 1496 s ($NAMOA^*$ with $d = 100$). The same time-out of 3600 s was set for the resolution of each problem.

5.1.2 Analysis on Class I problems

In this class, memory requirements of both algorithms were very similar, in fact indistinguishable in a graphical plot. This is due to the fact that both algorithms need to search virtually all nodes in this class. However, important differences in time requirements can be appreciated in figure 5.1(a).

In order to characterize the evolution of the relative performance of the algorithms, a statistical regression analysis has been carried out. Let us hypothesize by the observation of figure 5.1(a), that the relative difference of performance decreases following the rule [1] , $r_p = \alpha * s^{\beta}$, where $r_p = t_{MOA}/t_{NAMOA}$ is the dependent variable to adjust, denoting the relative performance of the algorithms, and s is the independent variable, representing the size of the problem [2].

Applying logarithms on both sides, the formula would be transformed into $\log r_p = \log \alpha + \beta * \log s$. With this model we can calculate a lineal regression adjustment, assuming by dependent/independent variables the logarithms of the original variables.

Several indicators can be presented to test the precision of this model. Table 5.1(a) summarizes some of these indicators for each correlation. The R values represent the gain that can be obtained when predicting the dependent variable from the knowledge of the value from the independent one. Values of R close to 1 indicate the model is a good approximation. The Durbin-Watson (D-W) indicator measures the independence of consecutive errors in the adjustment. Values close to 2 indicate errors are uncorrelated, and therefore, a good adjustment of the model.

Values presented in table 5.1(a) indicate that the adjustment of the model is good enough to explain the behaviour, specially for correlation 0 and below. The more complex the problem (i.e. negative correlation) the better the adjustment is.

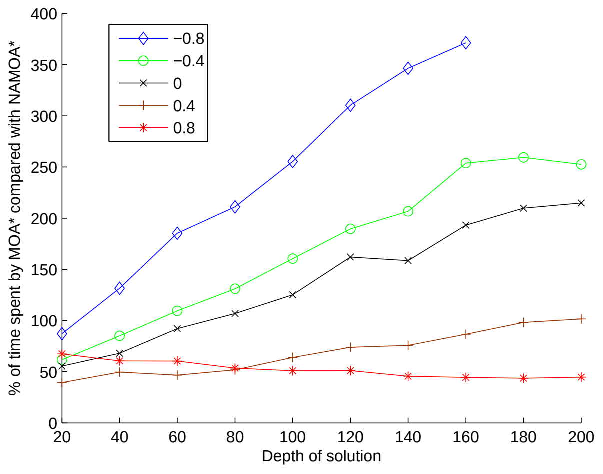
Bad results on correlation 0.4 are related to the change of tendency in the difference in relative performance. MOA^* is clearly faster for problems with correlations 0.8 and 0.4. However, for problems with lower correlation MOA^* is faster only with smaller solution depth. As solution depth increases, $NAMOA^*$ is clearly better than MOA^*.

The estimated parameter values α and β are shown [3] in figure 5.2(a), for different

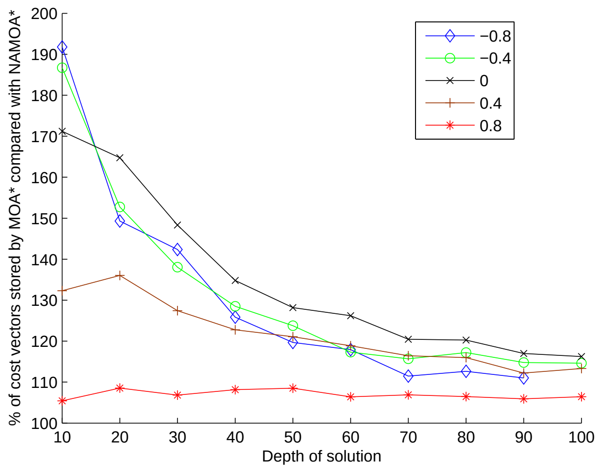
[1]Several rules have been tested, but only results concerned with this rule are presented, as it was the best fit

[2]Notice that depth of the solution could have been used as independent variable, and the results would be only affected by a constant factor

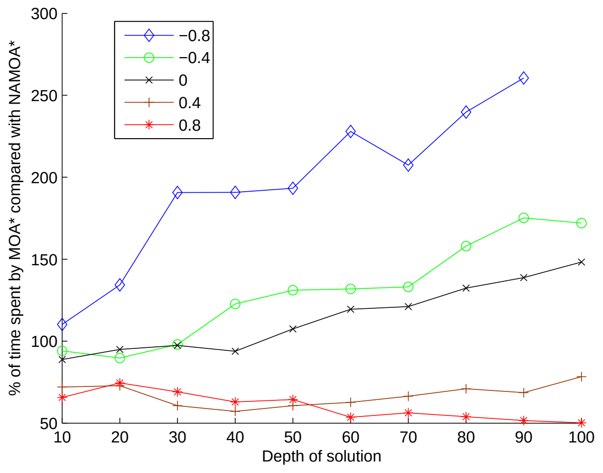
[3]Logarithm of α should be applied over all values reported. However, results would only be affected by a constant factor



(a) Relative time performance in class I problems



(b) Relative memory requirements in class II problems



(c) Relative time performance in class II problems

Figure 5.1: Relative performance between MOA^* and $NAMOA^*$ on class I/II grid problems with different correlation values (blind search)

Table 5.1: Estimated parameter values for the model $r_p = \alpha * s^{\beta}$ and indicators in the statistical analysis of the relative performance between MOA^* and $NAMOA^*$ on class I/II grid problems with different correlation values (blind search)

(a) Indicators for time, class I

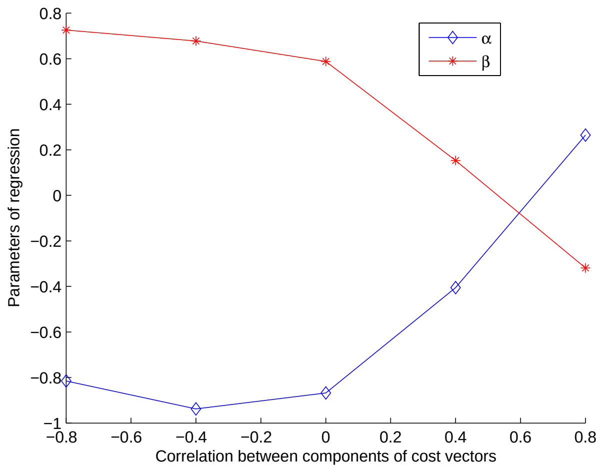
Correlation	α	β	R	R^2	D-W
0.8	0.265	-0.319	0.705	0.497	1.784
0.4	-0.405	0.153	0.352	0.124	0.430
0	-0.868	0.588	0.913	0.833	1.522
-0.4	-0.938	0.678	0.947	0.897	2.199
-0.8	-0.815	0.726	0.973	0.947	2.068

(b) Indicators for memory, class II

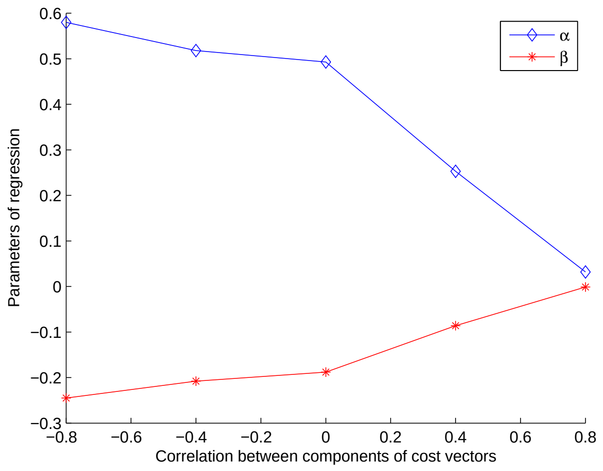
Correlation	α	β	R	R^2	D-W
0.8	0.032	-0.001	0.035	0.001	1.948
0.4	0.253	-0.086	0.833	0.694	1.762
0	0.493	-0.188	0.936	0.877	1.963
-0.4	0.518	-0.208	0.930	0.864	1.316
-0.8	0.580	-0.245	0.922	0.849	2.065

(c) Indicators for time, class II

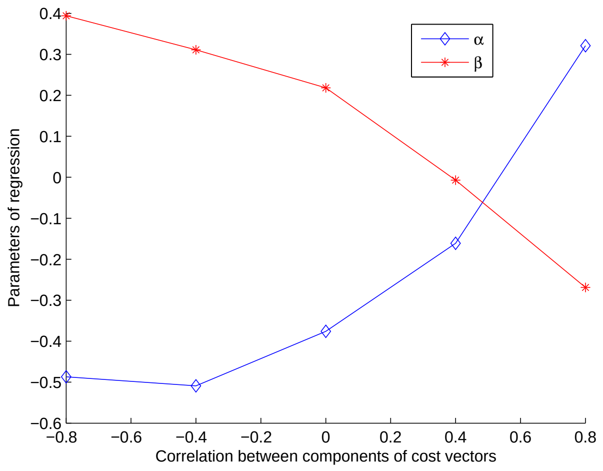
Correlation	α	β	R	R^2	D-W
0.8	0.321	-0.269	0.822	0.676	2.230
0.4	-0.161	-0.007	0.033	0.001	1.114
0	-0.376	0.218	0.692	0.479	1.777
-0.4	-0.509	0.311	0.801	0.642	1.361
-0.8	-0.487	0.394	0.804	0.647	2.108



(a) Parameters for time, class I



(b) Parameters for memory, class II



(c) Parameters for time, class II

Figure 5.2: Estimated parameter values for the model $r_p = \alpha * s^\beta$ in the statistical analysis of the relative performance between MOA^* and $NAMOA^*$ on class I/II grid problems with different correlation values (blind search)

correlations. Notice that the horizontal axis presents correlation values from lower to higher, i.e. the average number of nondominated paths to node (difficulty) follow an inverse rule. The lower the correlation, the more complex the problem is and more the value of α increases and β decreases. This means that MOA^* behaves better for positive correlations, though $NAMOA^*$ is better for 0 or negative correlations.

5.1.3 Analysis on Class II problems

For class II problems, a different result can be observed. Figure 5.1(b) shows that relative space requirements v_{MOA}/v_{NAMOA} are reduced as solution depth d increases. For easier problems MOA^* behaves even nearly two times worse. But additional effort tends to only a 15%-25% overhead approximately in the more difficult problems. This is actually smaller than originally expected, given that MOA^* can consider for expansion many dominated paths during search.

An analogous statistical analysis has been carried out for this case. The parameters and indicators can be found in table 5.1(b). The adjustment is good enough to say that the relative difference in space requirements of MOA^* with respect to $NAMOA^*$, decreases in an inverse rule to that shown before. The value for R in the case $\rho = 0.8$ can not be representative because differences between algorithms are minimal due to the simplicity of problems. The evolution of parameters α and β can be found in figure 5.2(b). Now, the value of α increases with problem difficulty (correlation) and the value of β decreases. This means that the relative space overhead of MOA^* approaches a constant ratio for difficult problems (i.e. higher solution depth) in all correlations considered.

In the case of time requirements, the analysis is similar to that found for class I problems. Figure 5.1(c) shows the same behaviour, though the difference is not as big for complex problems. Notice that the results presented include problems only half as deep as those from class I. Search in class II problems is more complex since it is not limited by grid boundaries. The adjustment to the same rule used in section 5.1.2 is not as good as that observed for class I problems (see table 5.1(c)). However, it is sufficient to draw the same conclusions as in the corner to corner grids. The same observation can be applied to R from correlation 0.4 as in that case.

5.1.4 Summary

This section has analyzed the relative performance of MOA^* and $NAMOA^*$. Blind search was considered over sets of randomly generated square grids with two objectives and bounded integer cost values.

As expected from previous formal analyses, $NAMOA^*$ is never beaten by MOA^* in space requirements. In class I problems both algorithms have very similar space requirements, while in class II problems, the overhead of MOA^* is found (counterintuitively) to be relatively small, and is well approximated by a polynomial law for uncorrelated objectives.

In problems with high correlation between objectives the selection and expansion scheme of MOA^* results in a faster algorithm. In class II problems there is a speed/memory trade-off between MOA^* and $NAMOA^*$ for high correlations. In the most difficult problems (uncorrelated objectives and deep solutions) $NAMOA^*$ clearly outperforms MOA^* in time though there is not as much difference in memory. The time overhead of MOA* for uncorrelated objectives is also well approximated by a polynomial law.

5.2 Heuristic multiobjective search on grid problems

This section considers the three multiobjective counterparts [4] of A^*, namely $NAMOA^*$, MOA^*, and TC algorithm. A detailed description of these algorithms was presented in section 2.4. Some details about the implementation can be found in section 3.3.

The experiments presented in this section analyze the impact of heuristic information over the sets of random class I/II grid problems previously used. In the case of $NAMOA^*$ and MOA^*, blind search amounts to using a zero heuristic, $\forall n\ H_0(n) = \{\vec{h}_0(n)\} = \{\vec{0}\}$, while heuristic search is evaluated with, $\forall n\ H_{TC}(n) = \{\vec{h}_{TC}(n)\}$. Algorithm TC always runs with the precalculated heuristics h_{mix} and $\vec{h}_{TC}$. These two heuristic functions were described in section 2.4.4.

Results show that, contrary to what was previously believed, use of heuristic information does not always result in improved performance. An additional analysis is carried out to determine which algorithm performs better in practice. Results show $NAMOA^*$ to be faster than MOA^* for harder problems and, regrettably, an important degradation in time performance in MOA^* with heuristic information, in accordance with the formal analysis presented in chapter 4. Finally, the analysis is concluded with a comparison of the relative performance of TC and heuristic $NAMOA^*$.

The algorithms tested in this section were implemented in Common Lisp using LispWorks Professional 5.01, and run on a HP Proliant DL160 G5 server with 2 Intel Xeon QuadCore 4572 @ 3GHz processors and 18 Gb of RAM, under Windows Server 2008 (32-bit).

5.2.1 Efficiency of Heuristic search

The experiments in this section try to analyze whether the use of heuristic information has a real impact in the space and time efficiency of $NAMOA^*$ and MOA^*. In this analysis, blind search is compared to heuristic search. Algorithm TC is not considered in this analysis, since it is designed to operate only with heuristic information.

5.2.1.1 Results

Regarding class I problems, space and time requirements for various correlations of blind and heuristic $NAMOA^*$ are shown in figures 5.3(a) and 5.3(b) respectively. Likewise, space and time requirements for blind and heuristic MOA^* are shown in figures 5.4(a) and 5.4(b).

Regarding class II problems, space and time requirements of blind and heuristic $NAMOA^*$ for various correlations are shown in figures 5.5(a) and 5.5(b) respectively. Likewise, space and time requirements of MOA^* are shown in figures 5.6(a) and 5.6(b).

[4]Recall that $NAMOA^*$ and MOA^* accept both blind and heuristic search, while TC algorithm was devised only for heuristic search.

Figures 5.3(a), 5.3(b), 5.4(a), 5.4(b), 5.5(a), 5.5(b), 5.6(a), and 5.6(b) show absolute performance values for the algorithms. Results are displayed only for certain correlation values. These were selected to show trends in behavior and at the same time allow clear simultaneous display at the given graphic scales of each figure. In section 5.2.2 the relative performance of the heuristic algorithms is analyzed and figures displaying values for all correlations considered are presented.

Values are averaged over ten problems generated for each depth, in all reported results hereafter. Notice that averages for certain depths, where at least one problem exceeded a one-hour runtime limit, are not displayed.

5.2.1.2 Analysis on Class I problems

Regarding space requirements, heuristic $NAMOA^*$ presents some savings over its blind counterpart. However, this advantage decreases with lower correlation, and space requirements are almost the same for $\rho = -0.8$. The same behavior is observed when comparing blind and heuristic MOA^*.

Regarding time requirements, the analysis of $NAMOA^*$ shows that with high correlation ($\rho = 0.4$) heuristic search is somewhat faster than blind search. The advantage is lost with $\rho = 0$ and, surprisingly, blind search is clearly faster with $\rho = -0.4$. This phenomenon is analyzed and discussed in 5.4.

The same behavior is found in the time performance of blind and heuristic MOA^*. However, heuristic MOA^* performs worse than blind MOA^* in *all* correlation values analyzed, in accordance to the formal analysis presented in chapter 4.

This results indicate that, for certain classes of problems, heuristic search can be of no advantage or even worse than blind search when time performance is considered.

5.2.1.3 Analysis on Class II problems

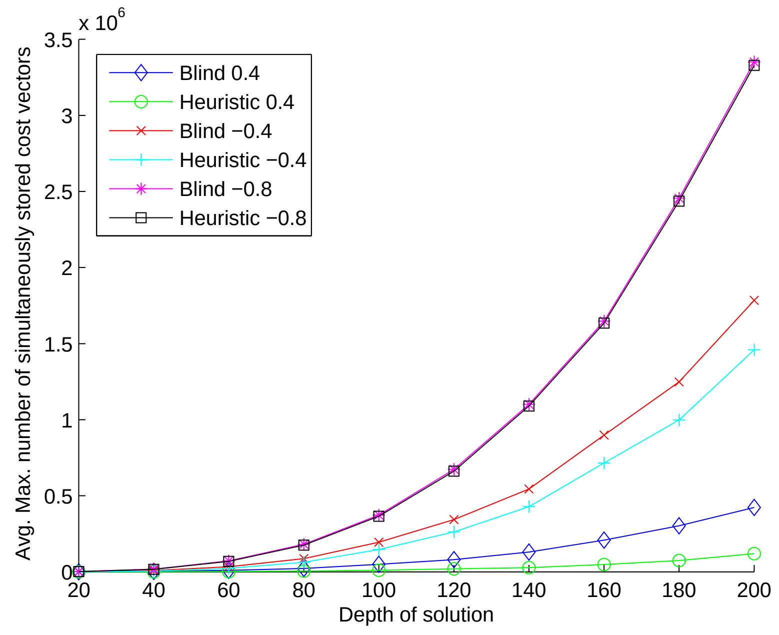
Regarding space requirements, heuristic $NAMOA^*$ shows almost an order of magnitude in savings compared to blind search. Savings are important in all correlations analyzed. The same behavior can be observed when comparing heuristic and blind MOA^*.

Regarding time requirements, heuristic $NAMOA^*$ also shows a very important saving compared to blind $NAMOA^*$ with both positive and negative correlations. Surprisingly, the behavior of heuristic MOA^* does not show much improvement with values of $\rho = 0.8$ or $\rho = 0.4$. For values of $\rho = 0$ or smaller, the situation gets worse, and heuristic MOA^* is clearly outperformed by blind MOA^*. This phenomenon is further analyzed and discussed in section 5.3.

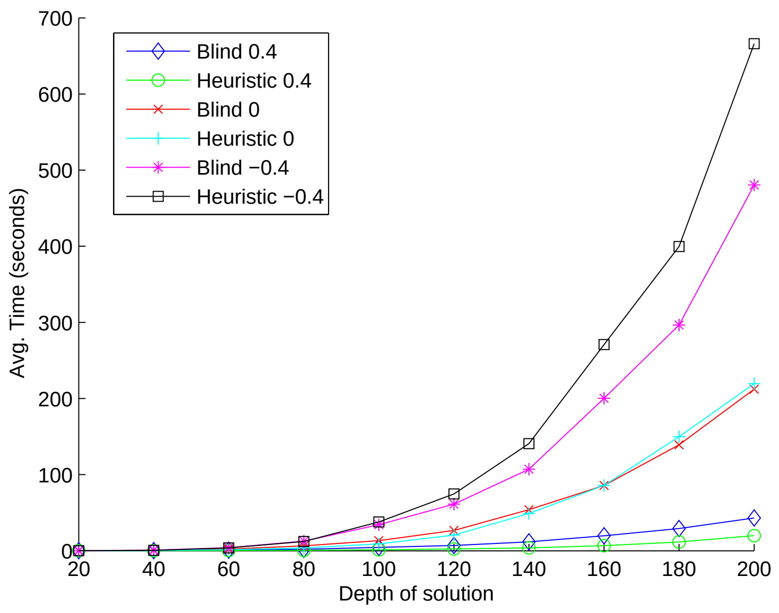
The time performance of heuristic MOA^* in both class I and class II problems clearly questions the use of accurate heuristic information to improve its efficiency. To the author's knowledge, this counter-intuitive result has not been previously reported in the literature.

5.2.2 Heuristic search, MOA^* vs $NAMOA^*$

The results presented in section 5.1 showed blind MOA^* to be faster than blind $NAMOA^*$ only under limited circumstances. The results presented in section 5.2.1

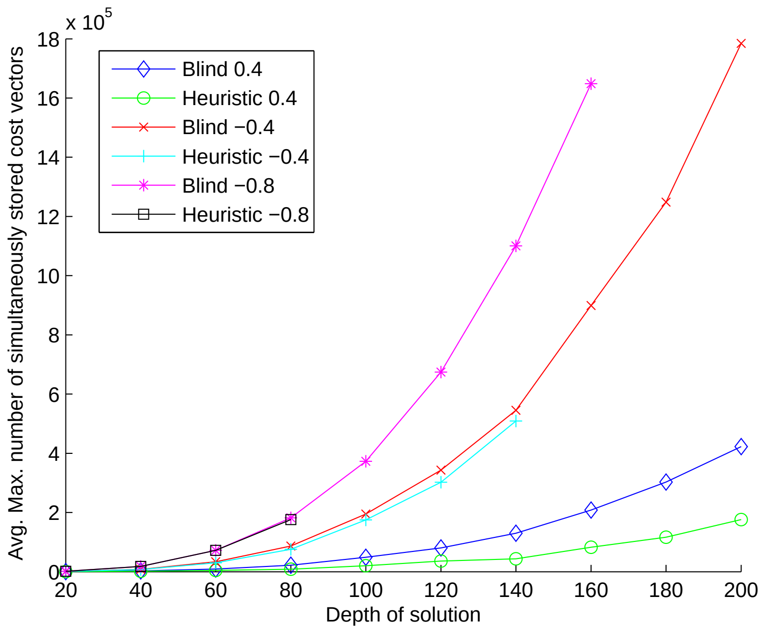


(a) Space requirements for correlation values 0.4,-0.4,-0.8

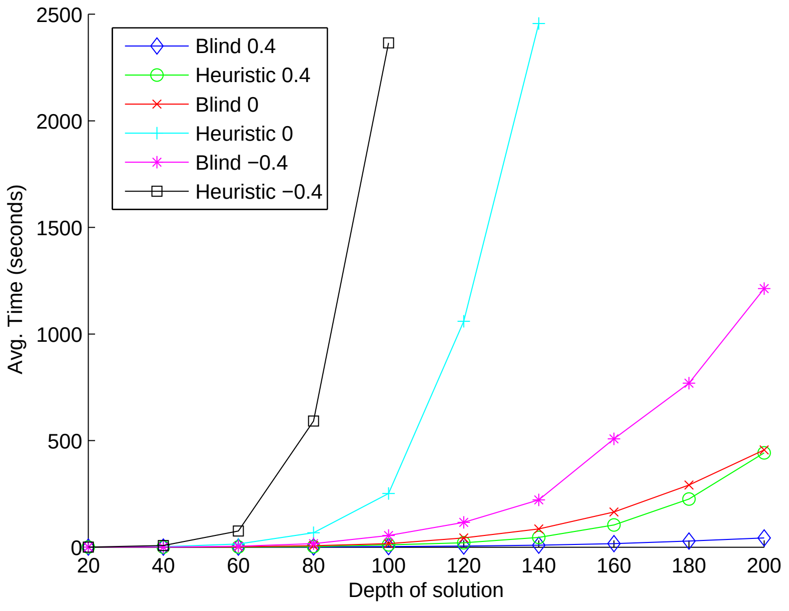


(b) Time requirements for correlation values 0.4,0,-0.4

Figure 5.3: Performance of blind and heuristic $NAMOA^*$ search on class I grid problems. Average values over ten problems generated for each depth.

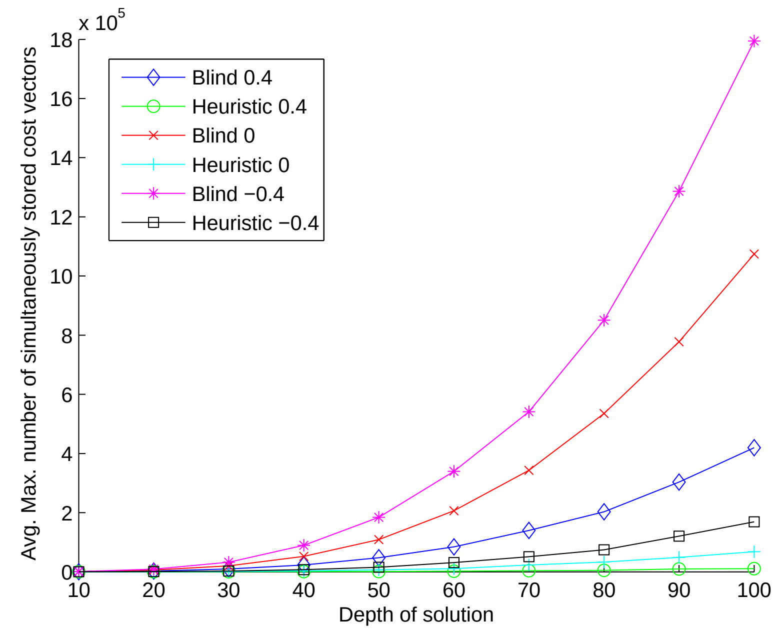


(a) Space requirements for correlation values 0.4,-0.4,-0.8

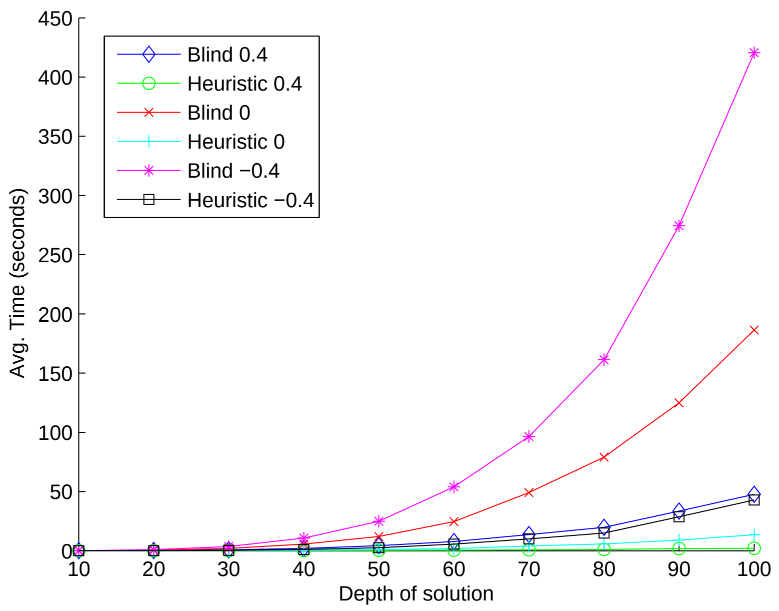


(b) Time requirements for correlation values 0.4,0,-0.4

Figure 5.4: Performance of blind and heuristic MOA^* search on class I grid problems. Average values over ten problems generated for each depth. Values are not displayed when at least one problem exceeded a 1h runtime limit.

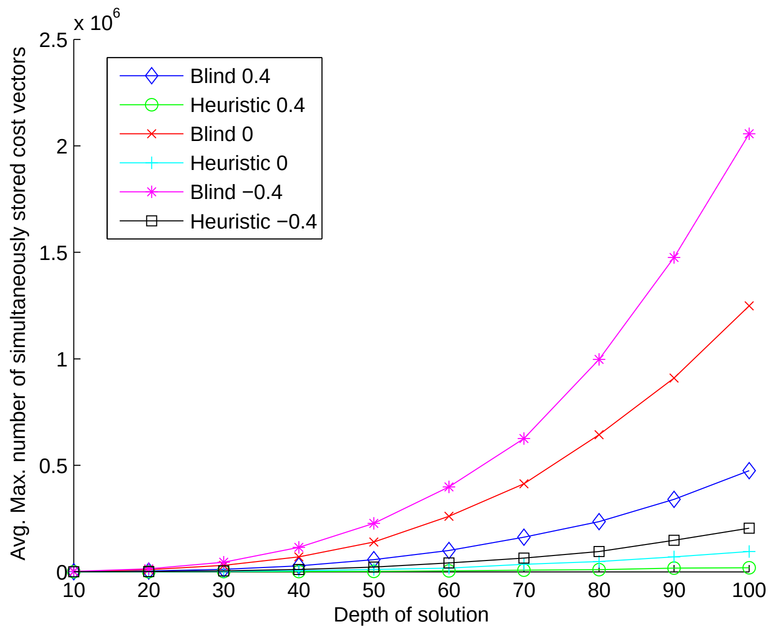


(a) Space requirements for correlation values 0.4,0,-0.4

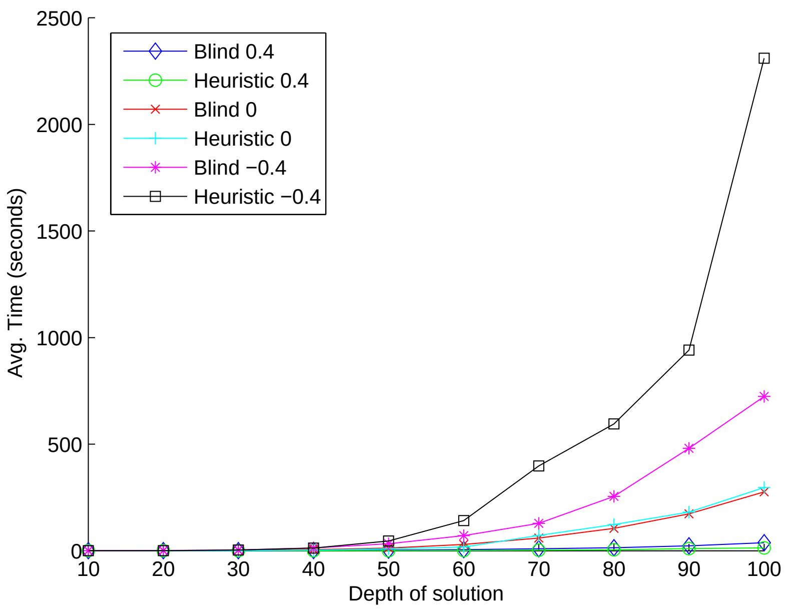


(b) Time requirements for correlation values 0.4,0,-0.4

Figure 5.5: Performance of blind and heuristic $NAMOA^*$ search on class II grid problems. Average values over ten problems generated for each depth.



(a) Space requirements for correlation values 0.4,0,-0.4



(b) Time requirements for correlation values 0.4,0,-0.4

Figure 5.6: Performance of blind and heuristic MOA^* search on class II grid problems. Average values over ten problems generated for each depth.

have shown heuristic MOA^* to perform worse than blind MOA^* in time requirements for both class I and class II problems. This section analyzes the relative performance of both MOA^* and $NAMOA^*$ using the heuristic H_{TC} to see if MOA^* is still competitive for some problem set.

5.2.2.1 Results

Let t_{MOA} and t_{NAMOA} be the time taken to solve a problem by MOA^* and $NAMOA^*$ respectively. Let also v_{MOA} and v_{NAMOA} be the maximum number of cost vectors stored by both algorithms when solving a problem. We shall denote by $r_v = v_{\text{MOA}}/v_{\text{NAMOA}}$ and $r_t = t_{\text{MOA}}/t_{\text{NAMOA}}$ the ratios of performance of MOA^* against $NAMOA^*$ in space and time respectively.

Regarding the resolution of class I problems with heuristic algorithms, relative space $(100 \times r_v)$ and time $(100 \times r_t)$ requirements as a function of solution depth d are shown for both algorithms in figures 5.7(a) and 5.7(b) respectively. Notice that all figures in this section show the ratios of performance as percentages.

Figure 5.8(a) presents the relative space requirements $100 \times r_v$ against solution depth d, for class II problems. Relative time performance $100 \times r_t$ of the algorithms is presented in figure 5.8(b) as a function of solution depth d.

5.2.2.2 Analysis on Class I problems

Regarding space requirements, heuristic MOA^* is clearly more demanding than heuristic $NAMOA^*$. Differences tend to reduce as problem difficulty increases (more negative correlation ratio and larger d). Only in the case of $\rho = 0.8$ (easier problems), MOA^* is increasingly worse.

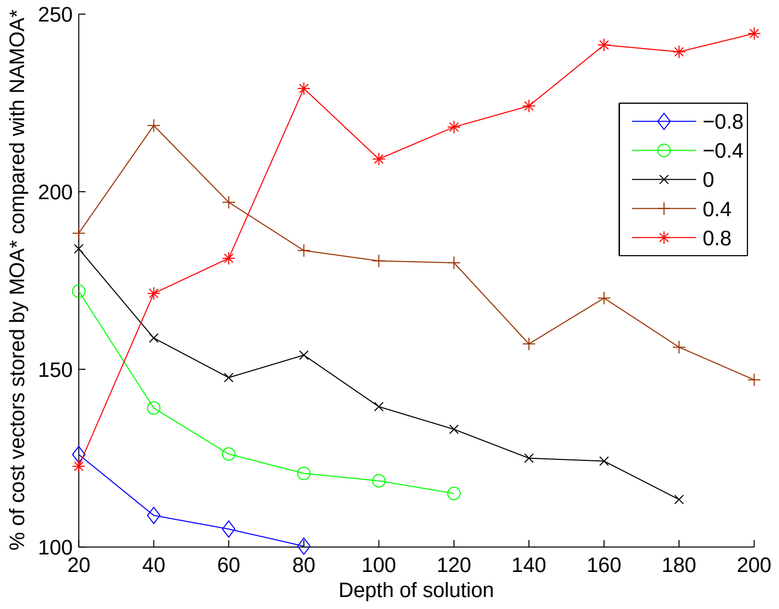
Regarding time requirements, important differences can be appreciated in the performance of both algorithms. The performance of MOA^* is similar to that of $NAMOA^*$ only for $\rho = 0.8$. For lower values of ρ, heuristic MOA^* is grossly outperformed by heuristic $NAMOA^*$ reaching overheads of more than 5000% in problems with solution depth 60 and $\rho = -0.8$. Recall that problems with deeper solutions in the smaller correlations are not shown, since the runtime of MOA^* exceeds the one-hour time limit established in the experiments.

5.2.2.3 Analysis on Class II problems

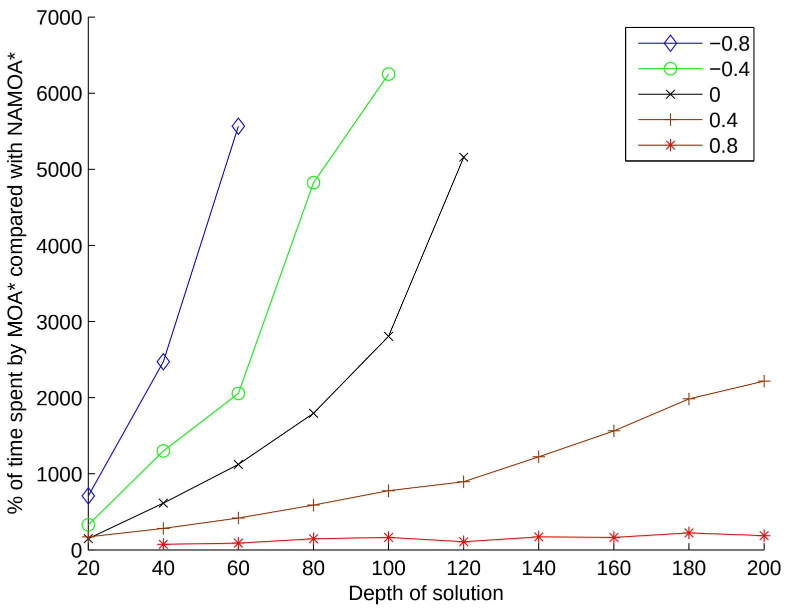
The relative performance of the heuristic algorithms for class II problems is quite similar to the one observed for class I ones.

MOA^* demands more space, but approaches the performance of $NAMOA^*$ for $\rho \leq 0.4$ as solution depth d increases. The relative space overhead of MOA^* approaches a constant ratio for difficult problems (i.e. higher solution depth and negative correlation).

Regarding time requirements, the analysis is similar to that found for class I problems, though the difference is somewhat smaller for complex problems.

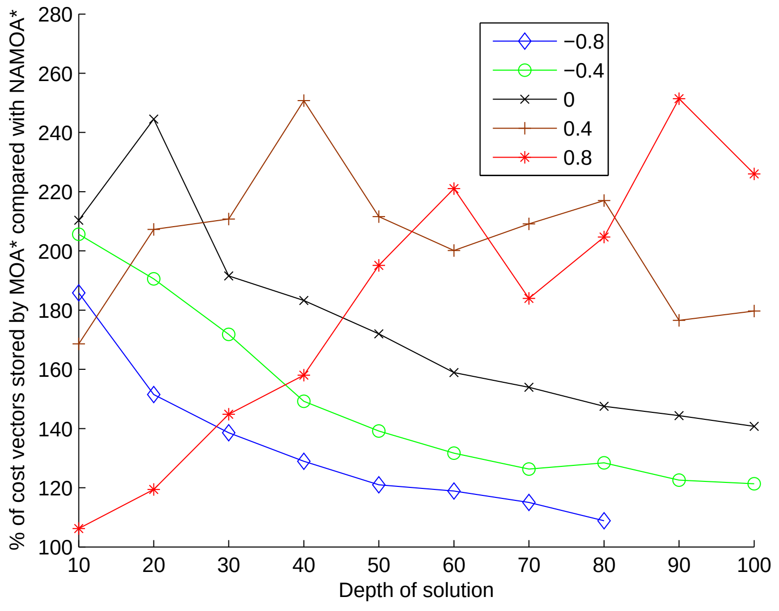


(a) Relative space requirements on class I problems

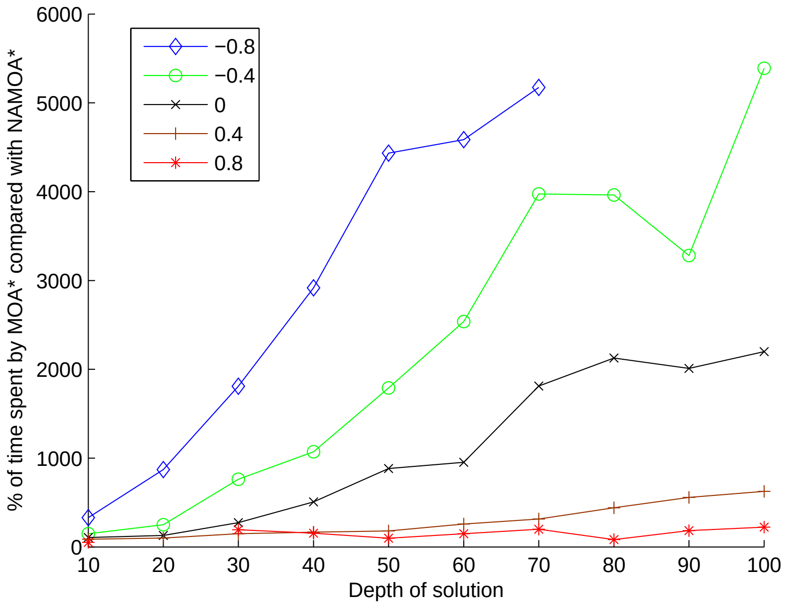


(b) Relative time performance on class I problems

Figure 5.7: Relative performance between MOA^* and $NAMOA^*$ on class I grid problems with different correlation values (heuristic search)



(a) Relative space requirements on class II problems



(b) Relative time performance on class II problems

Figure 5.8: Relative performance between MOA^* and $NAMOA^*$ on class II grid problems with different correlation values (heuristic search)

5.2.3 Heuristic search, TC vs $NAMOA^*$

The results presented in section 5.2.2 show that heuristic MOA^* is not competitive with heuristic $NAMOA^*$ in most cases. This section compares the relative performance of algorithm TC (which by definition uses the heuristics H_{TC} and h_{mix}) when compared to $NAMOA^*$ with heuristic H_{TC}.

5.2.3.1 Results

The performance of algorithms TC and heuristic $NAMOA^*$ is quite similar for both classes of problems. Results for space and time in class I problems are presented in figures 5.9(a) and 5.9(b) respectively. Analogous results for class II problems are shown in figures 5.10(a) and 5.10(b). Since performance measures are very similar, absolute values are presented in all cases, i.e. maximum number of cost vectors simultaneously stored in the search graph as a measure of space requirements, and runtime in seconds as a measure of time requirements.

5.2.3.2 Analysis

Regarding space requirements, figures 5.9(a) and 5.10(a) show that the performance of both algorithms is very similar for both classes of problems and for all correlations. This is quite reasonable to expect, since both algorithms follow a path (label) selection procedure, and use the same heuristic H_{TC} for filtering. However, as explained in section 2.4.3, a slightly different filtering policy is applied in TC when a new solution cost is discovered.

Regarding time requirements, the same effects can be observed for both classes of problems. Performance is indistinguishable for $\rho = 0.4$. However, for lower correlations $NAMOA^*$ performs clearly better. Both algorithms explore virtually the same set of paths (except for those not pruned by TC as explained in section 2.4.3) and therefore perform virtually the same number of iterations. One would expect a better performance of TC respect to $NAMOA^*$, as the accurate h_{mix} heuristic used by TC directs search more quickly to solutions. However, TC is found to be slower than heuristic $NAMOA^*$. This phenomenon is further analyzed and discussed in 5.4.

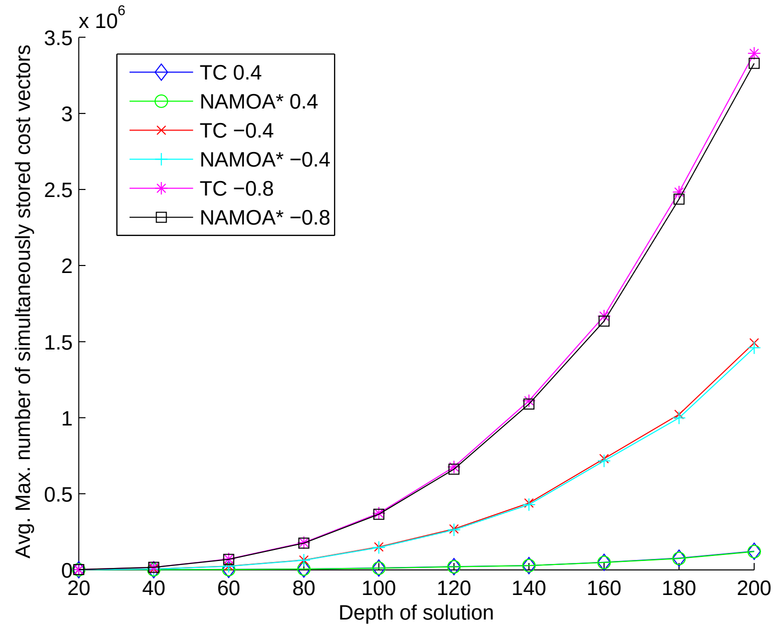
5.2.4 Time devoted to precalculation of heuristics

It is important to note that time results shown in this section do not include time devoted to the precalculation of heuristics. Several reasons justify this decision:

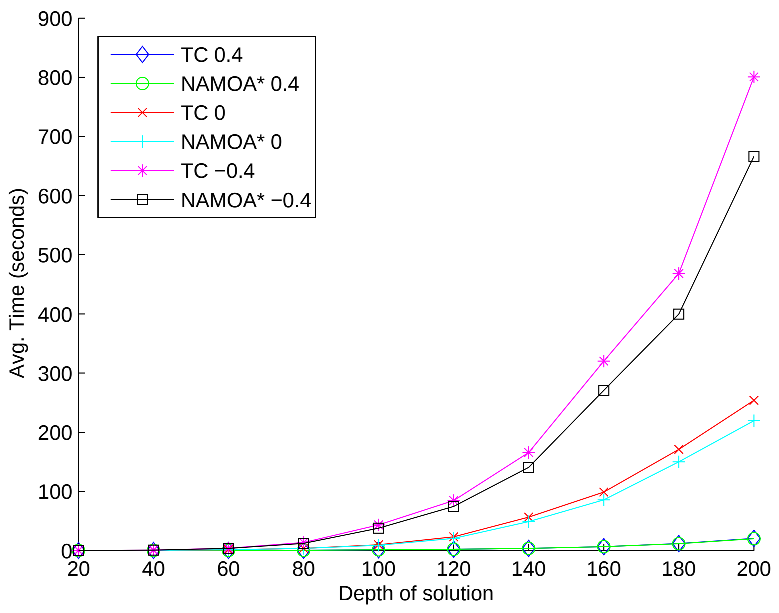
- When heuristic search is compared, the same time requirements apply for the precalculation of $\vec{h}_{TC}$ to MOA^* and $NAMOA^*$. TC takes a little more time as it must calculate in addition h_{mix} values. This additional time is not significant for class I, but it can represent 2-3 s more in the largest problems for class II[5]

- These time requirements for heuristics precalculation are not significant compared to total execution time in grid problems, and do not change the results of relative

[5]Note that the size of the grid is half depth in class I than in class II, and hence precalculation times are smaller for class I.

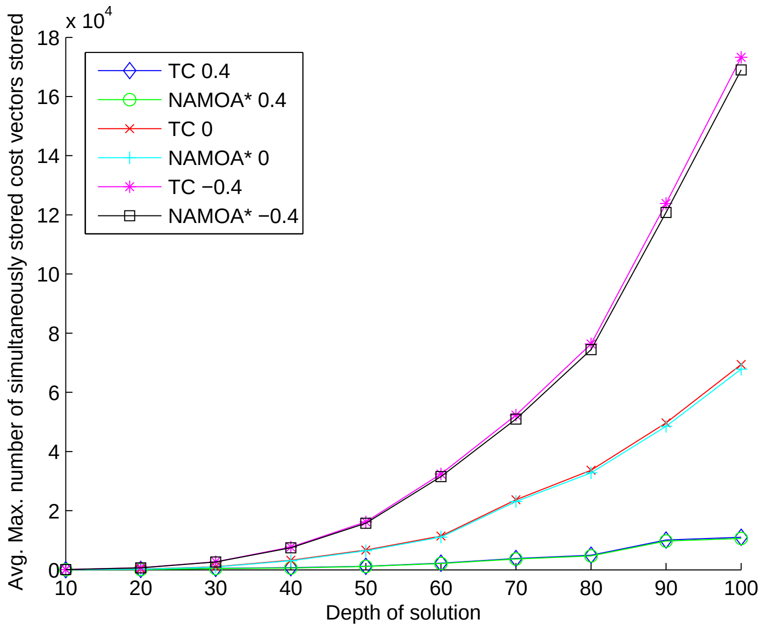


(a) Space requirements for correlation values 0.4,-0.4,-0.8

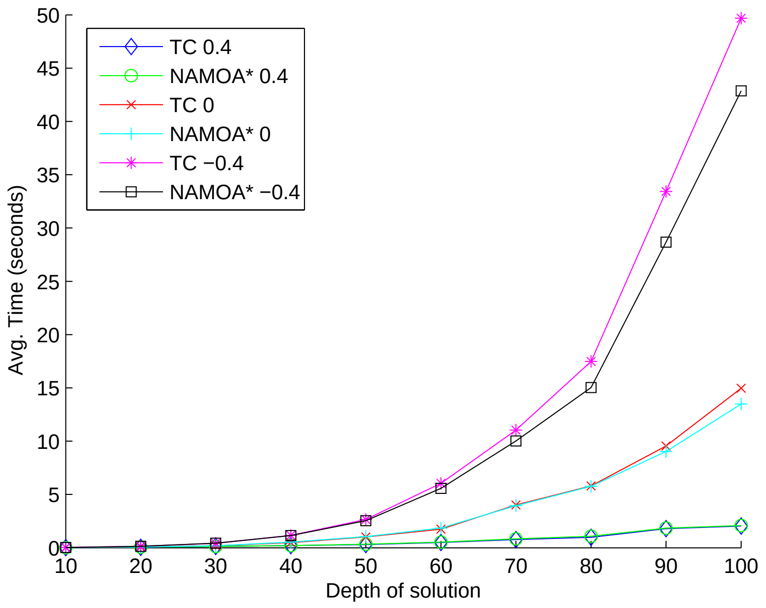


(b) Time requirements for correlation values 0.4,0,-0.4

Figure 5.9: Performance of TC and heuristic $NAMOA^*$ search on class I grid problems. Average values over ten problems generated for each depth.



(a) Space requirements for correlation values 0.4,0,-0.4



(b) Time requirements for correlation values 0.4,0,-0.4

Figure 5.10: Performance of TC and heuristic $NAMOA^*$ search on class II grid problems. Average values over ten problems generated for each depth.

performance when comparing two algorithms

- Moreover, precalculation time is not influenced by the correlation between objectives, as only one objective is optimized on each Dijkstra's run. Thus, heuristics precalculation time is the same regardless of the difficulty of the problems due to ρ

- Finally, the calculation method can be optimized, as it is shown in chapter 6.

Average time requirements for precalculation of heuristics are shown for each size in class I and class II problems on figures 5.11(a) and 5.11(b), respectively. These results indicate that the single objective search runs used to precompute the heuristic values have much smaller time requirements than bicriterion search. As the difficulty of the problem increases, the precalculation time becomes less significant compared to the time invested by the multiobjective search stage. However, it can be significant in some cases for class II problems, and must be improved. It is not worth calculating some heuristic values, as many nodes in class II problems will not be examined. This is further explained in chapter 6.

5.2.5 Summary

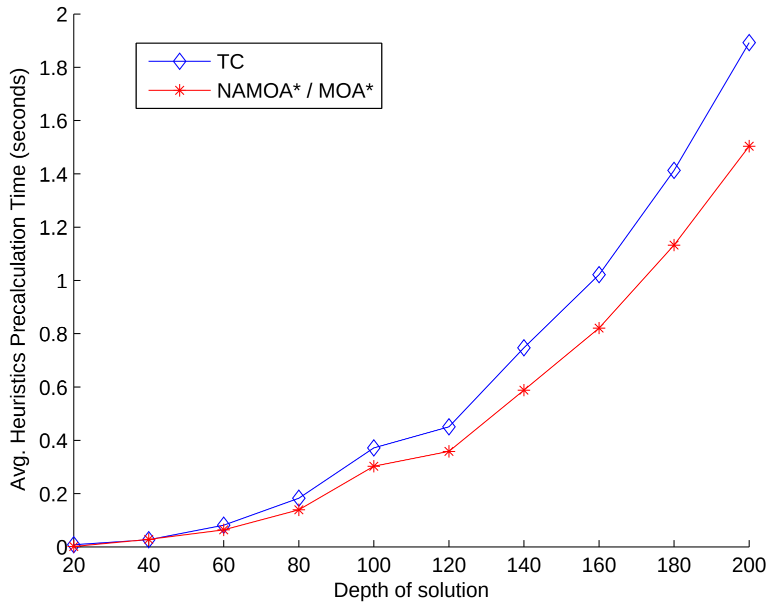
Table 5.2 summarizes the main results and ideas discussed in previous sections on the performance of the three algorithms analyzed.

The first round of experiments considered the impact of heuristic information in $NAMOA^*$ and MOA^*. Results show that heuristic search can save space in both classes of problems and for all correlations, except for $\rho = -0.8$ and class I problems, where only a very small difference is appreciated. Savings were specially significant for class II problems. Relative performance showed $NAMOA^*$ to perform better than MOA^*, though the space overhead in MOA^* seems to approach a constant ratio with higher solution depth.

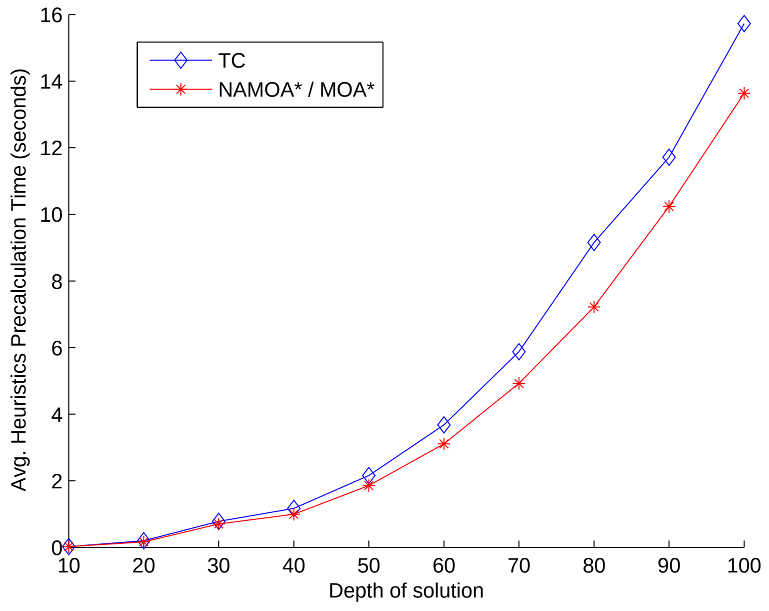
The convenience of heuristic search in terms of time requirements is not so clear. MOA^* was found to perform consistently worse with heuristic information, except for class II problems and $\rho = 0.8$. $NAMOA^*$ performed clearly better with heuristic information for class II problems, but worse for class I problems with $\rho < 0$. We can conclude that the use of the H_{TC} heuristic in $NAMOA^*$ always offers the best advantages in space savings, but time performance depends on the characteristics of the problem at hand.

The comparison of search with H_{TC} in $NAMOA^*$ and MOA^* showed that, as could be expected from previous theoretical results, the former has lower space requirements. MOA^* was found to approach a similar performance for lower correlations and increasing solution depth. However, time performance of heuristic MOA^* was found to be surprisingly bad, even when compared to blind MOA^*. Only for values of high correlation ($\rho = 0.8$) MOA^* shows a competitive performance with $NAMOA^*$.

Finally, the comparison between TC (using H_{TC} and h_{mix}) and $NAMOA^*$ (using H_{TC}) found small differences in space requirements, with some advantage for $NAMOA^*$. TC performed worse in time requirements for decreasing values of ρ and increasing solution depth. In particular, the use of the accurate h_{mix} heuristic in TC seems to work against the time performance of the algorithm.



(a) Time requirements for heuristics precalculation on class I problems.



(b) Time requirements for heuristics precalculation on class II problems.

Figure 5.11: Time devoted to heuristics precalculation by *TC* and heuristic *NAMOA*/ MOA** on class I/II grid problems. Average values over ten problems generated for each depth.

	Memory		Time	
Comparison	Class I	Class II	Class I	Class II
Blind/Heuristic Search	MOA^*& $NAMOA^*$ heuristic are better, but differences decrease as ρ decreases For $\rho = -0.8$, no significant differences	MOA^*& $NAMOA^*$ heuristic clearly better for all values of ρ, reductions up to 10-20 times	MOA^* heuristic much worse than blind MOA^* with all values of ρ $NAMOA^*$ heuristic faster for $\rho \geq 0$, and blind faster for $\rho \leq 0$	MOA^* heuristic worse than blind MOA^* for $\rho \leq 0$ $NAMOA^*$ heuristic always clearly better than blind $NAMOA^*$
Heuristic $MOA^*/NAMOA^*$	MOA^* becomes decreasingly worse for $\rho < 0.8$, tending to reduce difference with decreasing ρ and increasing depth $NAMOA^*$ clearly better for $\rho = 0.8$		MOA^* worse for $\rho < 0.8$, tends to rapidly increase difference with decreasing ρ and increasing depth, specially for $\rho \leq 0$ and class I problems	
Heuristic $NAMOA^*/TC$	TC worse, but little differences with $NAMOA^*$		TC worse, increasing differences with decreasing ρ and increasing depth, specially in class I problems up to 20-25% with $\rho \leq -0.4$ and depth 200	

Table 5.2: Summary of results from the empirical analyses on the performance of $NAMOA^*$, MOA^* and TC

In summary, there are classes of problems in which heuristic $NAMOA^*$ can provide important reductions in time and space requirements over blind search. This is the case of class II problems. In these problems search spans from the center of a grid outwards. Most paths are unlikely to be constrained by grid boundaries. Therefore, the use of heuristic estimates helps to filter large portions of the search space that are not aimed in the direction of the goal node.

The experiments revealed also two cases where the time performance of heuristic search was found worse than that of blind search,

- Heuristic $NAMOA^*$ and TC clearly take more time than blind search in many class I problems. Section 5.4 analyzes the time performance of the algorithms in terms of the number of dominance checks, following a methodology similar to the one proposed by Iori et al. (2010). The use of good heuristic cost estimates directs search quickly towards the goal. In the scalar optimization performed by A^*, this helps to find an optimal solution earlier, considerably reducing search effort (Pearl, 1984). However, multiobjective search must find *all* nondominated solutions, and terminates only when the $OPEN$ list is empty, i.e. when all alternatives have been either selected, pruned or filtered. Finding solutions at the earlier stages of search has the undesired side effect of increasing the effort needed for filtering checks. The effect is worse for TC, which is guided earlier to solutions by an extra heuristic h_{mix}.

  Therefore, there are situations in which the number of paths filtered with the help of heuristics is not enough to compensate for the extra effort in filtering checks. This is the case of class I problems, where search is performed from one corner of a grid to the opposite. Paths are constrained by grid boundaries, and most aim approximately in the direction of the goal node. Therefore, the use of heuristics filters only a small amount of them. As it turns out, the number of filtered paths decreases rapidly with decreasing correlation.

- The same phenomenon described above for $NAMOA^*$ and TC applies to MOA^* as well. However, an additional phenomenon degraded substantially the performance of MOA^* in our experiments for all but the simpler problems. This is discussed in section 5.3. The analysis reveals that reexpansions of already expanded labels are clearly responsible for the very bad behavior observed for heuristic MOA^*, in accordance to the formal analysis presented in chapter 4. These were triggered by the combination of accurate heuristics with the node selection policy.

5.3 Analysis on the time performance of heuristic MOA^*

The time performance results presented in sections 5.2.1 and 5.2.2 clearly question the idea that heuristic information improves search efficiency in MOA^*. This section analyzes in detail a particular problem instance in order to provide better insight into the performance of the algorithm and explain its behavior. This empirical result confirms the formal analysis presented in chapter 4. The instance chosen is a 80×80 class II grid problem with $\rho = 0$.

Recall that no admissible algorithm equipped with the same consistent heuristic information can skip the expansion of a label expanded by $NAMOA^*$. When compared to $NAMOA^*$, MOA^* can be shown to perform three kinds of label expansions (Mandow & Pérez de la Cruz, 2005),

1. Necessary label expansions, i.e. at least the labels expanded by $NAMOA^*$ on the same problems will be necessarily expanded once.

2. Redundant label expansions, i.e. reexpansions of necessary labels due to node reopenings.

3. Superfluous label expansions, i.e. expansions or reexpansions of labels that will never be considered by $NAMOA^*$ when solving the same problem. These generally correspond to nondominated labels that are nevertheless filtered by $NAMOA^*$ (i.e. dominated by some solution path).

Figures 5.12(a) and 5.12(b) show the total number of label expansions for each node in the sample grid of blind and heuristic $NAMOA^*$ respectively. Recall that the start node is $(40, 40)$, and the goal node $(20, 20)$. These figures provide a graphical explanation for the good performance of $NAMOA^*$ in class II problems. While the explored portion of the grid grows approximately circular in blind search, the use of the H_{TC} heuristic aims search precisely towards the goal, constraining the explored portion to approximately a square defined by the start and goal nodes.

Notice also that, as could be expected, the number of (nondominated) labels considered per node grows with node depth.

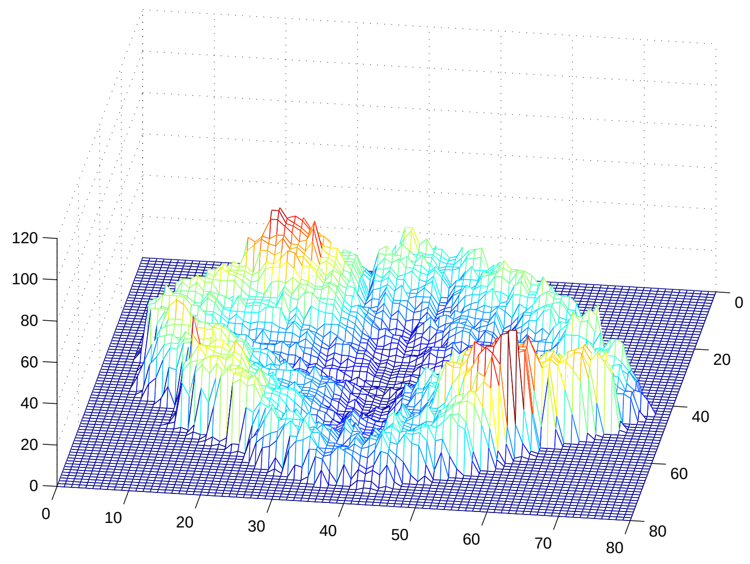
Figures 5.13(a) and 5.13(b) show the total number of labels expanded for each node by MOA^* with blind and heuristic search respectively. Notice that while the explored portion of the graph is the same as with $NAMOA^*$ equipped with the same heuristic information, the number of redundant and superfluous label expansions makes the total number of expansions per node much higher in MOA^*.

5.3.1 Performance of blind MOA^*

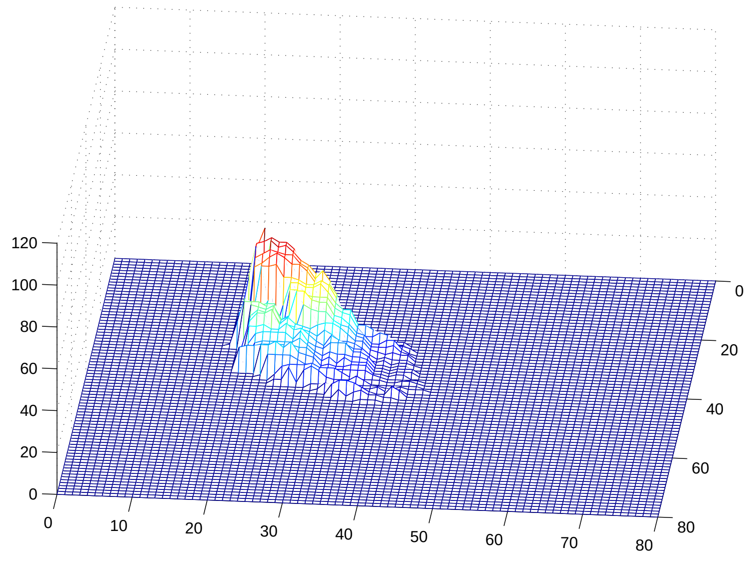
Let us analyze first the case of blind MOA^*. Figures 5.12(a) and 5.13(a) show the total number of label expansions per node of $NAMOA^*$ and MOA^*respectively. Notice that vertical scales are different (the range in figure 5.12(a) is 120 and in figure 5.13(a) is 2000).

The hypothesis proposed here is that blind MOA^* incurs in a time overhead over blind $NAMOA^*$ mainly due to the expansion of superfluous labels in the deeper nodes considered during search.

Figure 5.14 displays the number of superfluous labels expansions performed by MOA^* for each node (vertical scale is 120). This number is summarized and compared to the total number of label expansions of MOA^* in figure 5.15(a). Expansions are averaged for all nodes at the same depth and displayed as a function of depth. This clearly shows that almost all label expansions performed by MOA^* in the search frontier are superfluous, and account for much of the search effort performed by the algorithm. This effect is accentuated by the fact that the number of nodes considered at

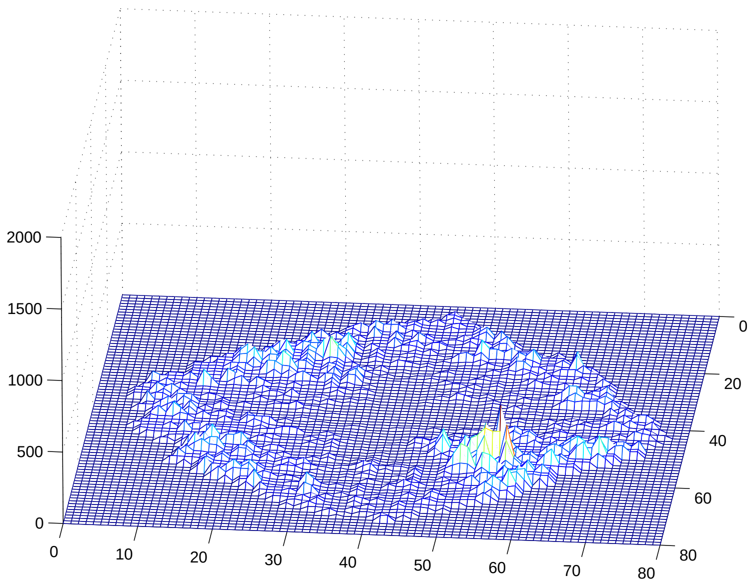


(a) Blind $NAMOA^*$

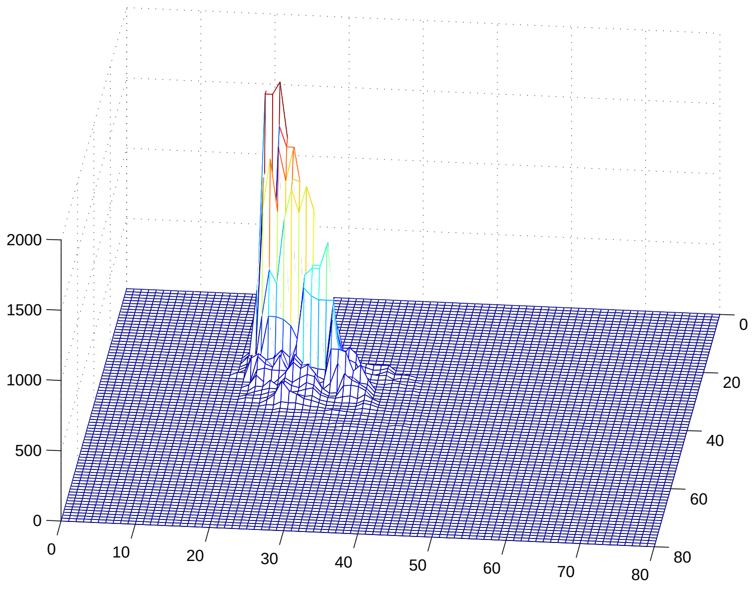


(b) Heuristic $NAMOA^*$

Figure 5.12: Number of label expansions per node performed by blind and heuristic $NAMOA^*$ in a sample 80×80 class II grid problem with $\rho = 0$. Start node is $(40, 40)$ and goal node is $(20, 20)$.



(a) Blind MOA^*



(b) Heuristic MOA^*

Figure 5.13: Number of label expansions per node performed by blind and heuristic MOA^* in a sample 80×80 class II grid problem with $\rho = 0$. Start node is $(40, 40)$ and goal node is $(20, 20)$.

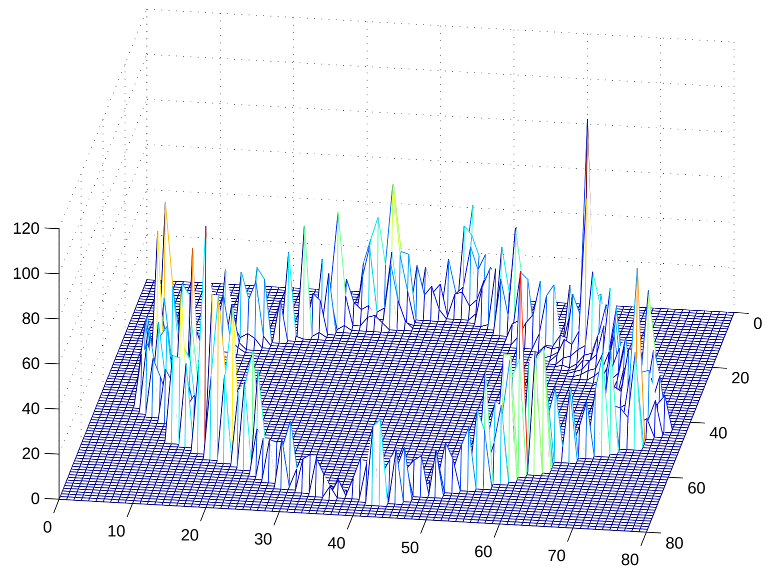


Figure 5.14: Number of superfluous label expansions per node performed by blind MOA^* in a sample 80×80 class II grid problem with $\rho = 0$.

a given depth grows steadily with depth. Figure 5.15(b) compares the average number of necessary and non-superfluous (necessary plus redundant) expansions performed by MOA^* for each node depth. This clearly suggests also that redundant expansions (i.e. reexpansion of necessary labels), while significant, are not fundamental for the time overhead incurred by blind MOA^*. In fact, these sharply decrease as the goal depth (40) is reached.

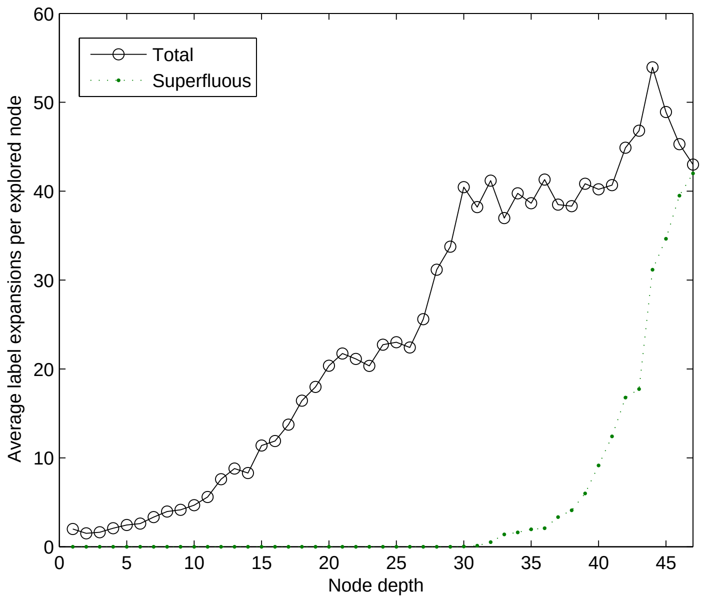
These results support the hypothesis that the node selection and expansion strategy performed by MOA^* leads the algorithm to an unnecessary expansion of many labels in frontier nodes that will later be filtered by the algorithm. The same effect was observed in other randomly selected problem instances.

5.3.2 Performance of heuristic MOA^*

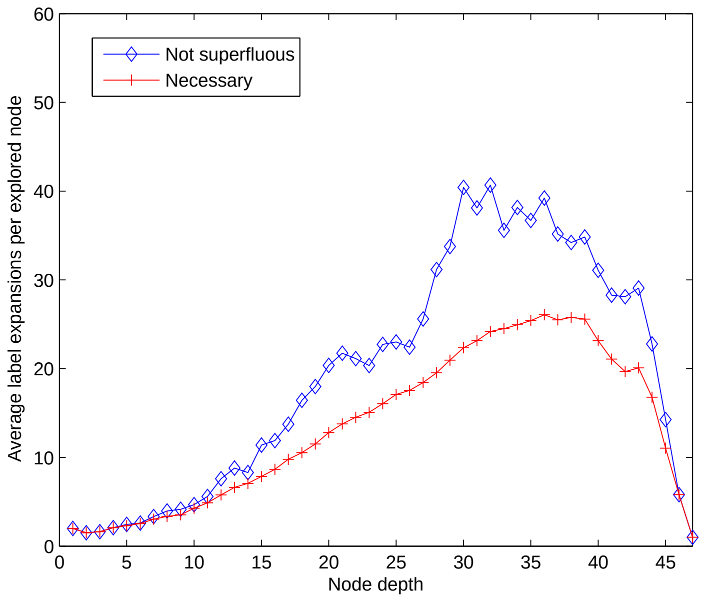
A comparison of figures 5.12(b) and 5.13(b) reveals that a new additional effect must be boosting the number of label expansions performed by heuristic MOA^*. Notice that vertical scales are different (the range in figure 5.12(b) is 120 and in figure 5.13(b) is 2000).

Figure 5.16 shows the number of superfluous label expansions by heuristic MOA^*. While there is a similar effect to that observed in the frontier of blind MOA^*, the largest counts of label expansions occur in nodes deep inside the explored area of the graph. Notice also that, in this case, only a few deep nodes are considered, and that most are mid-distance from the start to the goal.

The hypothesis in this case is that, as search is constrained by accurate heuristics



(a) Total and superfluous label expansions



(b) Non-superfluous and necessary label expansions

Figure 5.15: Averages of different classes of label expansions per node depth performed by blind MOA^* in a sample 80×80 class II grid problem with $\rho = 0$.

to narrow areas of the graph, the same nodes are explored and reopened again and again, producing a large number of redundant label expansions, as in the case formally analyzed in chapter 4 (see specially section 4.6).

Figure 5.17(a) compares the average number of superfluous expansions per node with the total number of label expansions. While the first magnitude is significant, it accounts only for half of the expansions even at the deeper nodes. Figure 5.17(b) compares the average number of necessary and non-superfluous (necessary plus redundant) expansions performed by MOA^* for each node depth. This clearly supports the hypothesis that redundant expansions are responsible for the important time overhead of MOA^*.

To gain further insight into the performance of blind and heuristic MOA^* it is interesting to recall the effects of $G(n)$ and $H(n)$ on heuristic search [6]. The former adds a *breadth-first* component to search. Without $H(n)$, blind MOA^* reduces to a purely breadth-first (uniform cost) search. Although MOA^* cannot guarantee that upon expansion all nondominated labels to a given node have been found, it is unlikely that any node will remain for a long time far behind the search frontier awaiting reexpansion. Thus, the search frontier advances more or less uniformly closing nodes definitively shortly after their first expansion.

The use of $H(n)$ introduces some sort of *depth-first* behaviour in the operation of MOA^*. In fact, the sole use of $H(n)$, an estimate of the proximity to the goal, could be misleading in many cases for the algorithm. It is the balance of $G(n)$ and $H(n)$ that guarantees at the same time that search is aimed in the right direction, and that all nondominated solutions are found.

However, the experiments performed with heuristic MOA^* confirm that the combination of the node selection strategy with the depth-first component of $H(n)$ plays against the time efficiency of the algorithm, as formally shown for a particular case in chapter 4.

5.4 Analysis on the time performance of heuristic $NAMOA^*$ and TC

This section gives an explanation to the bad time performance of heuristic search in $NAMOA^*$ and TC in two different scenarios,

1. Those cases where the time performance of heuristic $NAMOA^*$ and TC was found to be worse than that of blind $NAMOA^*$ (see section 5.2.1.2)
2. Those where TC performed worse than heuristic $NAMOA^*$ (see section 5.2.3.2)

Our hypothesis is that better informed heuristics lead the algorithms more quickly to solutions. Since each new candidate path needs to be checked for dominance against the cost of already found solutions (*filtering*), this slows down the algorithms as more and more (filtering) dominance checks need to be performed.

[6]see (Pearl, 1984, sec 3.2.1) for a similar description pertaining to A^*. However, the adverse effect described here for MOA^* does not appear in A^*, since the latter terminates as soon as a first solution is found.

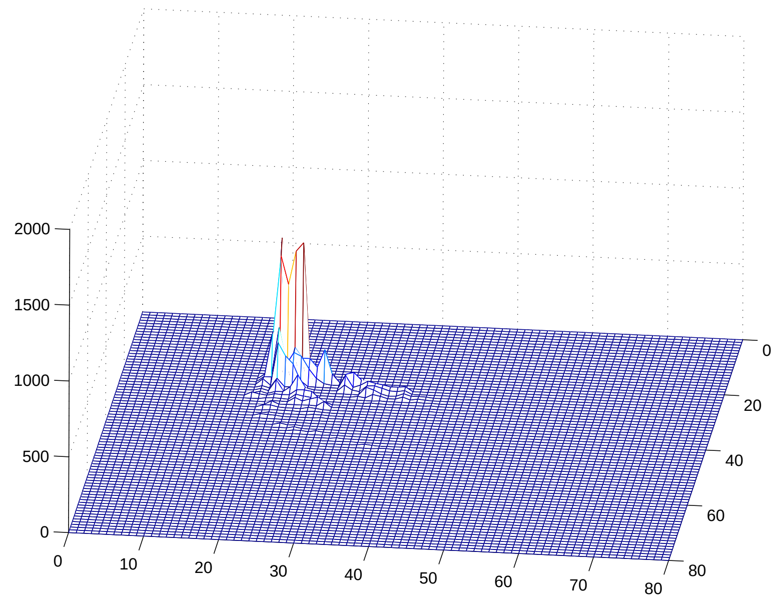


Figure 5.16: Number of superfluous label expansions per node performed by heuristic MOA^* in a sample 80×80 class II grid problem with $\rho = 0$.

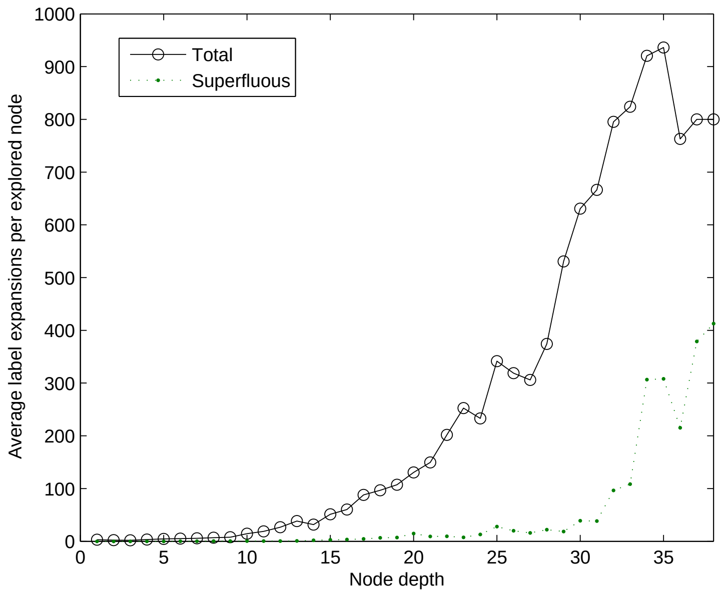
5.4.1 Analysis on the number of dominance checks

This section describes the experiments conducted to confirm this hypothesis. In particular, the number of dominance checks due to pruning and filtering was counted for all problem classes, correlations, and depths reported in previous sections.

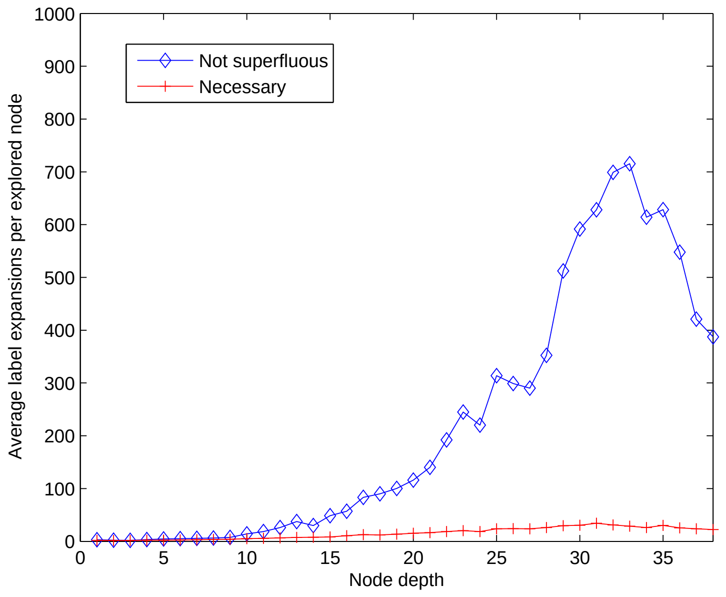
Regarding time requirements of $NAMOA^*$ in class I problems, figure 5.18(b) shows that, for any given correlation, the number of pruning dominance checks of blind search is always larger than those of heuristic search. This is in accordance with the larger number of paths explored by blind search. However, figure 5.18(a) clearly shows that, for any given correlation, the number of filtering dominance checks is larger in heuristic search than in blind search. Since heuristic $NAMOA^*$ explores less paths than blind $NAMOA^*$, the additional number of filtering dominance checks must be due to the fact that solutions are found much earlier in the former case. Notice also that, in absolute terms, the number of filtering dominance checks is always much larger than the number of pruning dominance checks in the heuristic searches.

This confirms the hypothesis that finding solutions early has the undesired side effect of boosting the number of dominance checks needed for filtering, slowing down the speed of search. In summary, in class I problems the reduction in the number of iterations (alternatives considered) achieved by heuristic search does not always compensate the increase in dominance checks per iteration when compared to blind search.

Figure 5.19 compares the number of dominance checks of TC and heuristic $NAMOA^*$



(a) Total and superfluous label expansions



(b) Non-superfluous and necessary label expansions

Figure 5.17: Averages of different classes of label expansions per node depth performed by heuristic MOA^* in a sample 80×80 class II grid problem with $\rho = 0$.

for class I problems. Figure 5.19(a) clearly shows that the number of filtering dominance check is even greater for TC than for heuristic $NAMOA^*$. This can be attributed to the accurate heuristic used for label selection in TC, which guides the algorithm to solutions even at earlier stages of the search. Pruning dominance checks also grow, but to a lesser extent. This can be attributed to the particular filtering strategy of TC (see section 2.4.3).

Figure 5.20 shows the percentage of filtering dominance checks among all (pruning and filtering) dominance checks for class I problems and $\rho = 0$. While filtering checks account in average only for 42.84% of the total in blind $NAMOA^*$, these raise to 81.12% in heuristic $NAMOA^*$, and 87.44% in TC. Values of $\rho = 0.4$ and $\rho = -0.4$ change the figures to 18.27% and 52.55% for blind $NAMOA^*$, but the percentages in the heuristic searches vary less than 2% (not shown for clarity).

Again, similar percentages were found for class II problems, though in these problems the extra effort due to the increase in filtering dominance checks is more than compensated by the reduction in the number of alternatives considered.

Regarding the operation of MOA^* in class I problems, the number of filtering dominance checks also increases with heuristic search, but so does even in larger proportion the number of pruning checks (see figures 5.21(a), 5.21(b)). Therefore, the general bad performance of heuristic MOA^*, which extends even to class II problems, must be due to a different cause. This have been analyzed in section 5.3.

5.4.2 Disadvantages of more informed heuristics

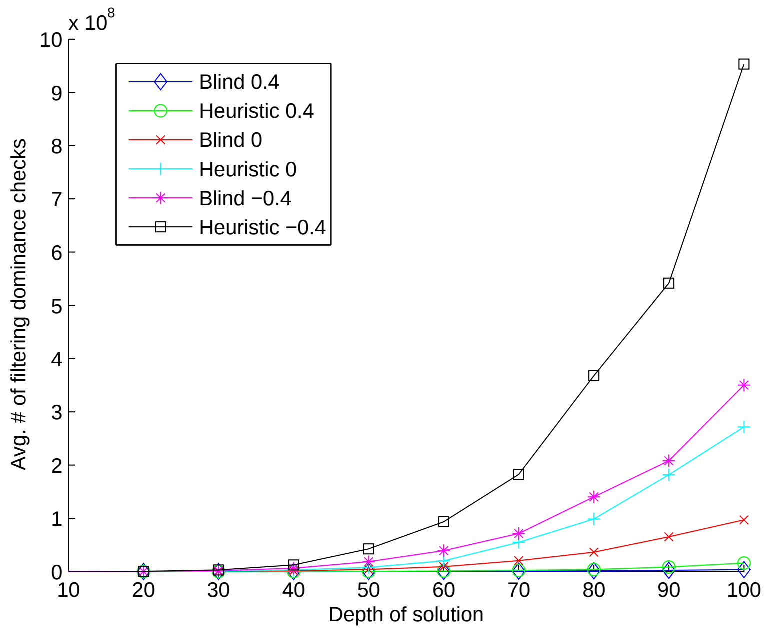
Search time cannot be easily extrapolated from the number of iterations. This is due to the fact that in bicriteria search, label expansion is by no means an atomic constant-time operation. Bicriteria search algorithms share with scalar ones a variability in time per iteration due to the need to sort open alternatives. In bicriteria search the number of open alternatives is low at the beginning and end of the search, and high some time in between. Additionally, each new label selected for expansion generates a number of successor labels that need to be checked for dominance with the labels in the $G_{op}(n)$ and $G_{cl}(n)$ sets of the destination node n, as well as the set of nondominated labels found to the goal nodes (i.e. those kept in the $COSTS$ set). Particularly, the size of $COSTS$ can grow rapidly with solution depth. To further validate our hypothesis, this was analyzed for one particular class I problem of size 100×100 for the case of $\rho = 0$.

In order to evaluate the impact of the h_{mix} heuristic, we shall consider an additional "blind" selection rule, in addition to those rules used in previous sections:

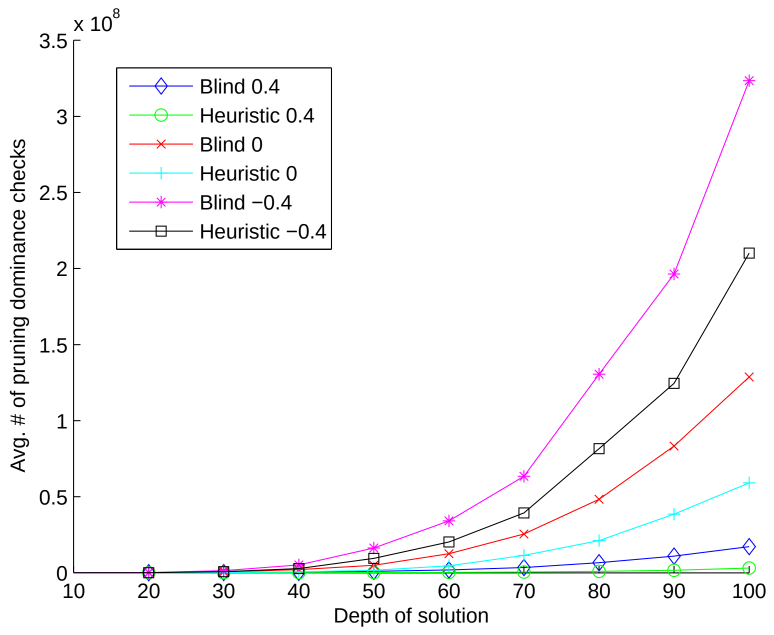
PATH SELECTION in TC with blind selection: Select for expansion an alternative label $(n, \vec{g}_n, \{g\})$ from *OPEN* such that $\nexists (n', \vec{g}_{n'}, \{g'\}) \in OPEN$ with $\sum_i g_i(P_{sn'}) < \sum_i g_i(P_{sn})$

It is easy to prove that minimizing $\sum_i g_i(P_{sn})$ yields a nondominated $\vec{f}$ (just consider that $\vec{h}_0(n)$ is used for selection). The G_{op} sets are ordered in this case according to this blind linear evaluation function. Table 5.3 sums up the features of all the algorithms evaluated in this chapter.

The algorithms were implemented in this case using the LispWorks Professional

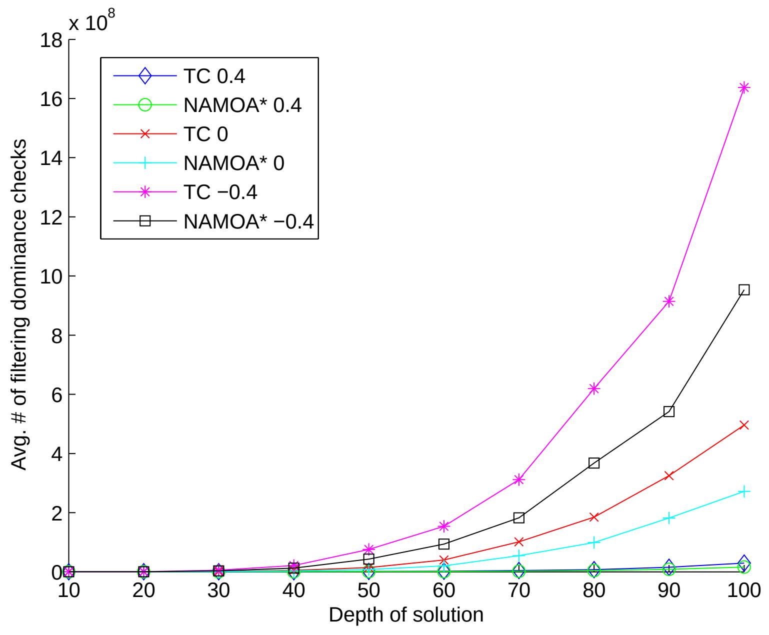


(a) Filtering dominance checks for correlation values 0.4,0,-0.4

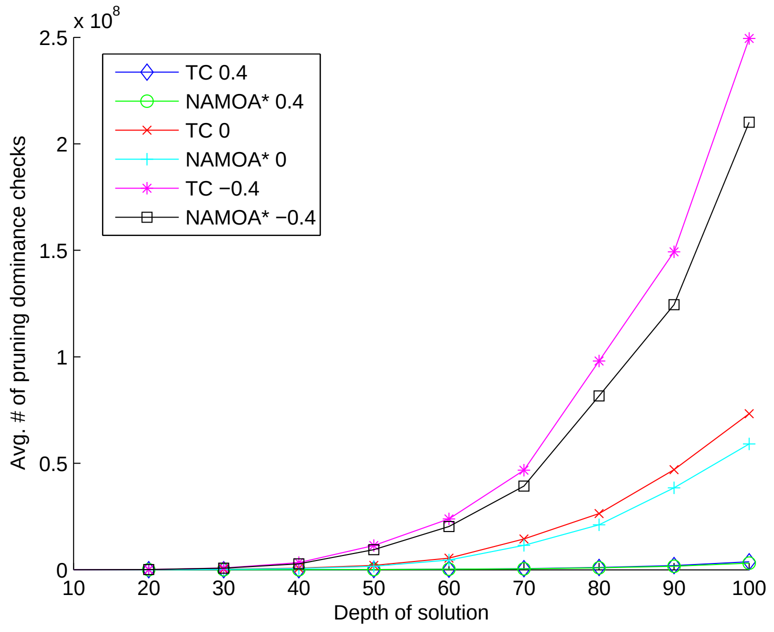


(b) Pruning dominance checks for correlation values 0.4,0,-0.4

Figure 5.18: Dominance checks performed by blind and heuristic $NAMOA^*$ search for class I grid problems. Average values over ten problems generated for each depth.



(a) Filtering dominance checks for correlation values 0.4,0,-0.4



(b) Pruning dominance checks for correlation values 0.4,0,-0.4

Figure 5.19: Dominance checks performed by TC and heuristic $NAMOA^*$ search for class I grid problems. Average values over ten problems generated for each depth.

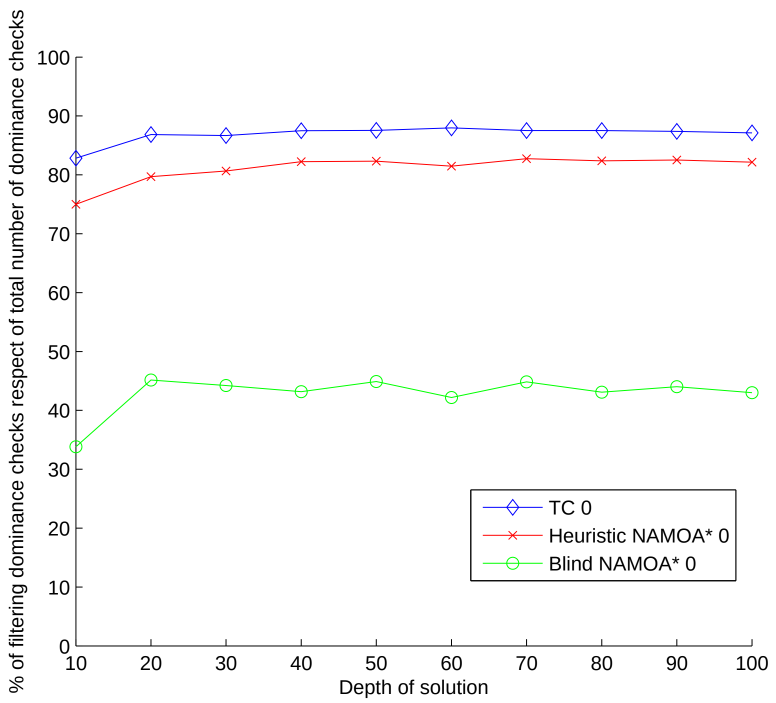
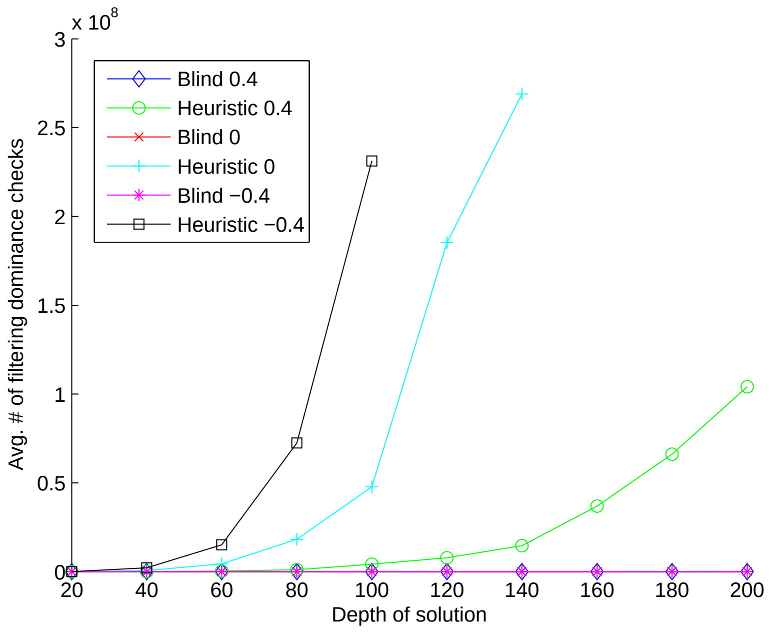
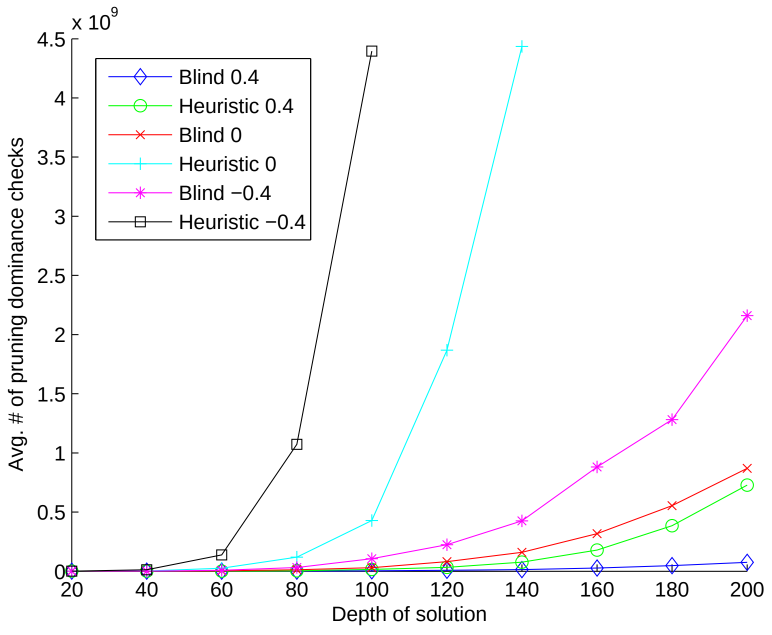


Figure 5.20: Percentage of filtering dominance checks among total (pruning and filtering) dominance checks for class I grid problems with $\rho = 0$.



(a) Filtering dominance checks for correlation values 0.4,0,-0.4



(b) Pruning dominance checks for correlation values 0.4,0,-0.4

Figure 5.21: Dominance checks performed by blind and heuristic MOA^* search for class I grid problems. Average values over ten problems generated for each depth. Values are not displayed when at least one problem exceeded a 1h runtime limit.

Algorithm	Selection rule	Additional selection rule	Filtering criteria
Blind $NAMOA^*$	best $\vec{g}$	lex order	$\vec{g}$ dominated
Heuristic $NAMOA^*$	best $\vec{g}+\vec{h}$	lex order	$\vec{g}+\vec{h}$ dominated
Blind MOA^*	node with best $\vec{g}$	lex order	$\{\vec{g}\}$ dominated
Heuristic MOA^*	node with best $\vec{g}+\vec{h}$	lex order	$\{\vec{g}+\vec{h}\}$ dominated
TC with blind selection	best $\sum_i g_i$		$\vec{g}+\vec{h}$ dominated
TC with heuristic selection	best $h_{mix}+\sum_i g_i$		$\vec{g}+\vec{h}$ dominated

Table 5.3: Algorithms evaluated in the empirical analyses performed on chapter 5.

5.01 programming environment, and were run on a Windows XP 32-bit platform, with an Intel Core2 Quad Q9550 at 2.8Ghz, and 4Gb of RAM.

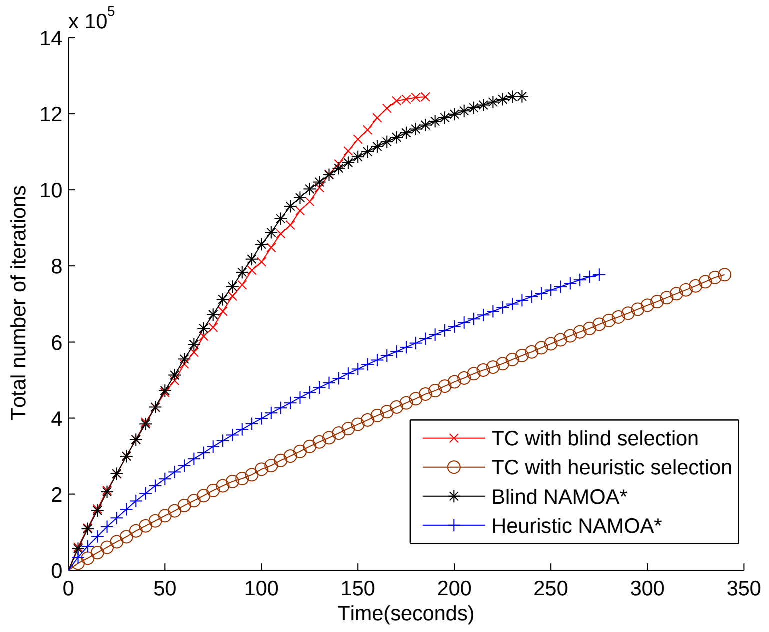
Figure 5.22(a) shows that both TC with blind selection and blind $NAMOA^*$ performed the same number of iterations in this problem. However, TC with blind selection performed a roughly constant number of iterations per second and abruptly slowed down at the end, while blind $NAMOA^*$ slowed down more gradually and finally took more time to finish. Notice that TC with heuristic selection and heuristic $NAMOA^*$ performed less iterations than the previous algorithms, but required more time per iteration from the beginning, specially TC with heuristic selection. The result is that these algorithms were slower while solving the same problem instance. Similar behaviour was found in other cases analyzed.

The explanation of this behavior can be found in figure 5.22(b). All algorithms found the same number of solutions. However, TC with blind selection found the solutions in the final seconds of search, coincidentally with the abrupt descent of iterations per second observed in figure 5.22(a). In a similar way, blind $NAMOA^*$ found solutions more gradually but also in the final search stage, resulting in the second fastest alternative. Heuristic $NAMOA^*$ found solutions even faster and was therefore slower. Finally, TC with heuristic selection found solutions very quickly and was the slowest algorithm.

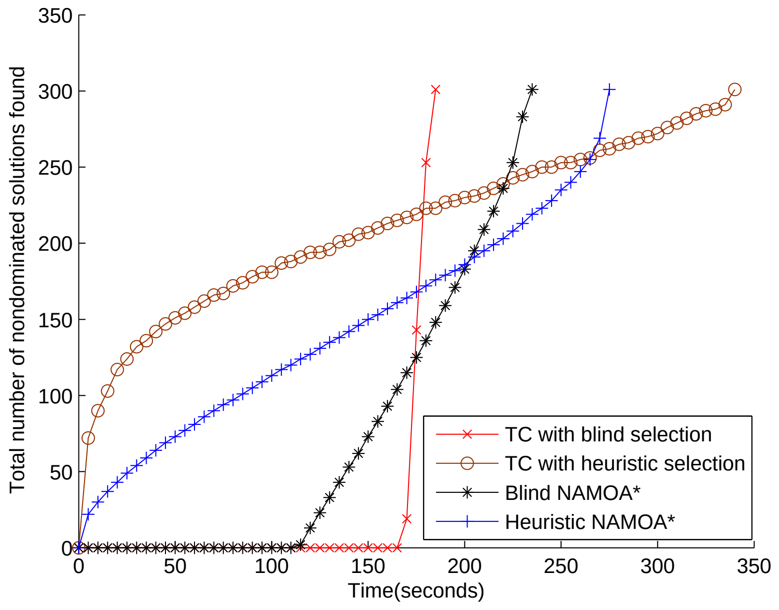
Figure 5.23 shows the time taken by each algorithm to find the *first* solution as a function of solution depth averaged for all problem sets. Again, TC with blind selection is the algorithm that starts to find solutions later, followed by blind $NAMOA^*$. Both heuristic $NAMOA^*$ and TC with heuristic selection find solutions very early and appear undistinguishable at this scale.

5.5 Heuristic multiobjective search on modified DIMACS road maps

This chapter is concluded with an analysis of the three algorithms over a set of realistic route planning problems in road maps proposed by Raith & Ehrgott (2009). These three road maps (DC, RI and NJ) have been previously described in section 3.2.4. Two objectives linearly correlated ($\rho = 0.99$) were considered: travel time and physical distance.



(a) Number of iterations vs. time of $NAMOA^*$ and TC for a particular search problem.



(b) Number of nondominated solutions found vs. time of $NAMOA^*$ and TC for a particular search problem.

Figure 5.22: Analysis on the time performance of TC and $NAMOA^*$ on a sample 100×100 class I grid problem with $\rho = 0$.

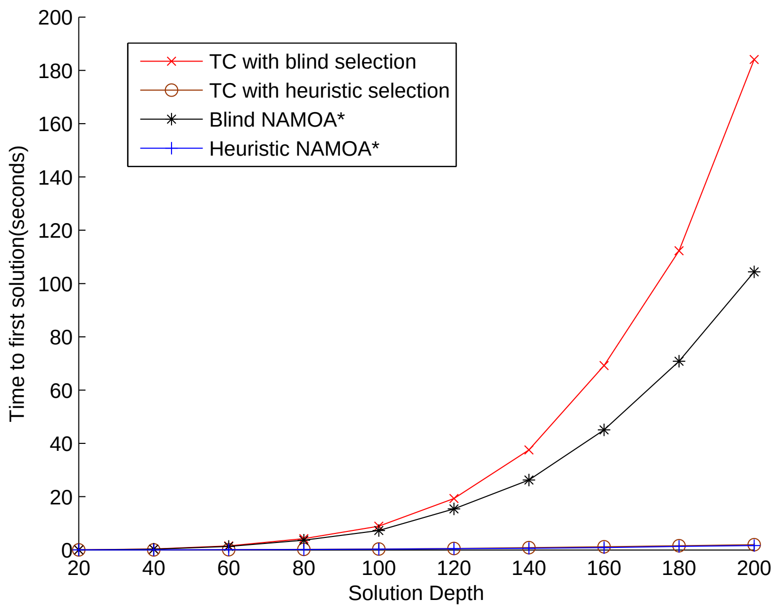


Figure 5.23: Time requirements for $NAMOA^*$ and TC to reach the first solution on class I problems with $\rho = 0$, averaged over ten problem generated for each depth.

5.5.1 Results

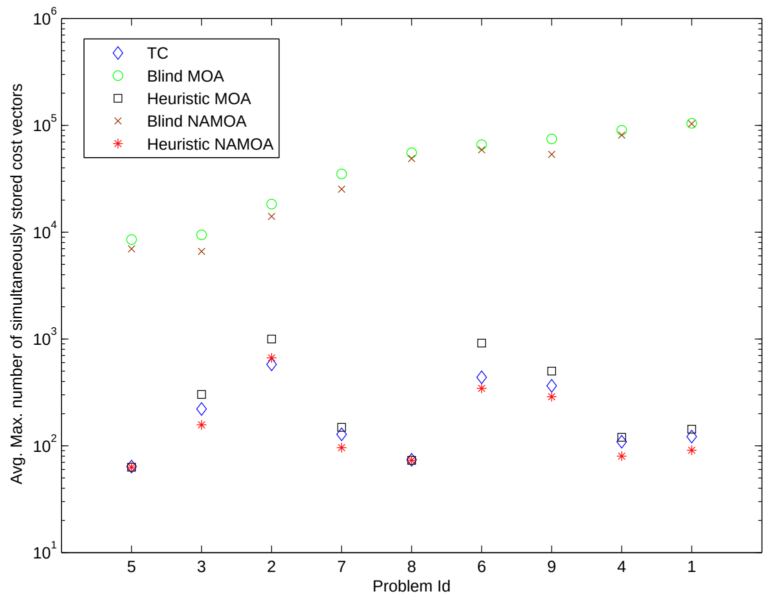
Results presented in this section are the average values of ten different runs for each problem. A runtime limit of one hour was set. Figures 5.24(a), 5.24(b), 5.25(a), 5.25(b), 5.26(a), 5.26(b) show only results for problem instances which do not exceed the runtime limit of 1-h. Values are presented in a logarithmic scale for the ordinate axis. Problem instances in the abscissa axis are ordered by increasing value of ordinate for blind MOA^* (the slowest algorithm).

5.5.2 Analysis

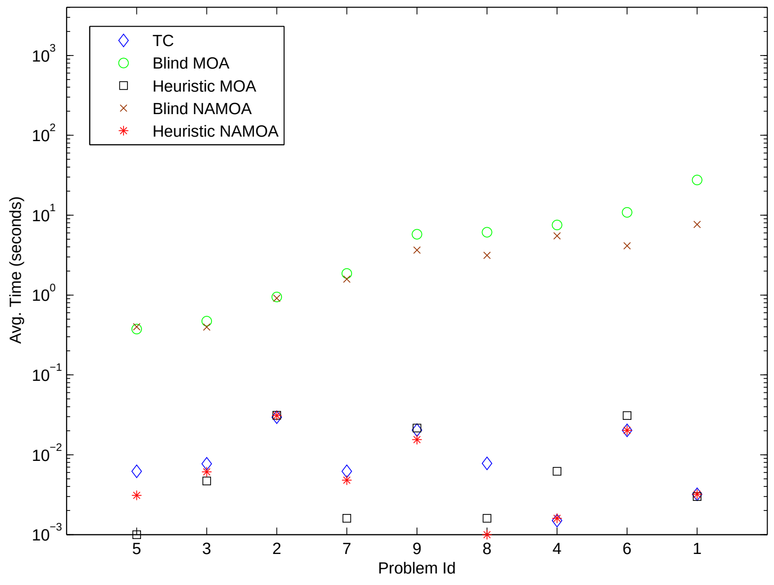
The performance of the algorithms over this problem set is in general consistent with the results presented for class II problems and high correlation between objectives (see section 5.2.5).

The analysis of space requirements shows that,

- Heuristic MOA^* and heuristic $NAMOA^*$ achieve important reductions in space requirements over all problem instances when compared to blind searches.
- A comparison of heuristic $NAMOA^*$ with heuristic MOA^* confirms that space requirements are always smaller with the former.
- The requirements of TC are in general between those of the other heuristic algorithms, and closer to the figures of $NAMOA^*$.

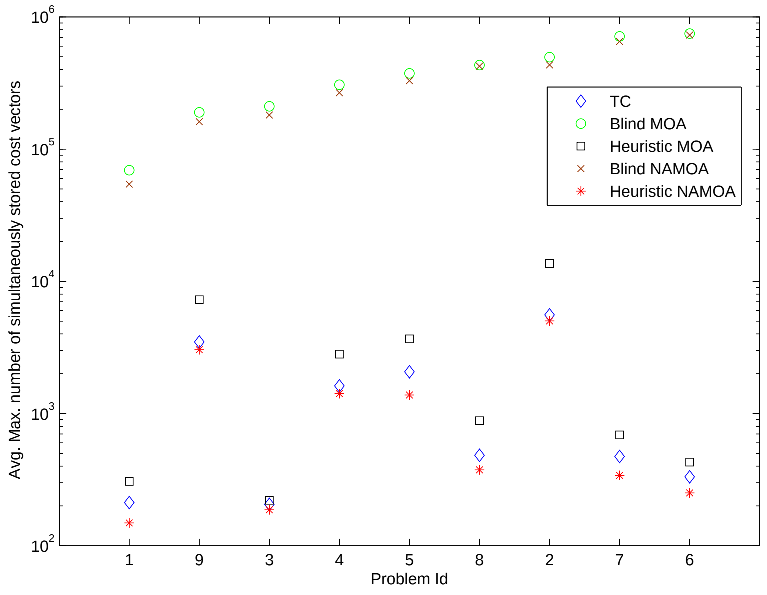


(a) Space requirements

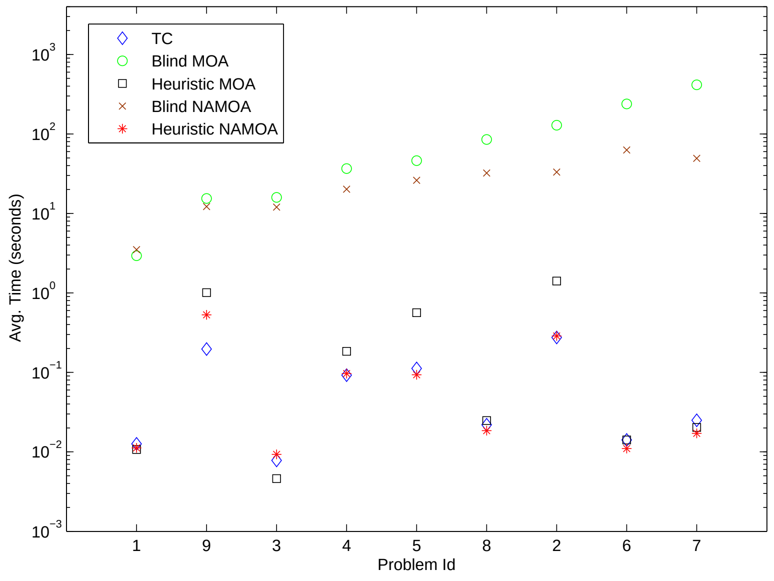


(b) Time requirements

Figure 5.24: Performance of blind and heuristic search on DC road map problems. Average values over ten runs for each problem instance.

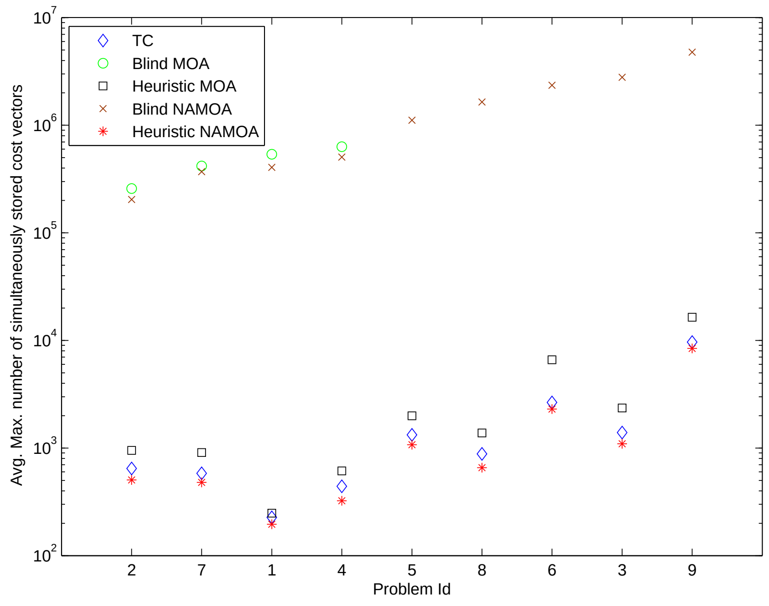


(a) Space requirements

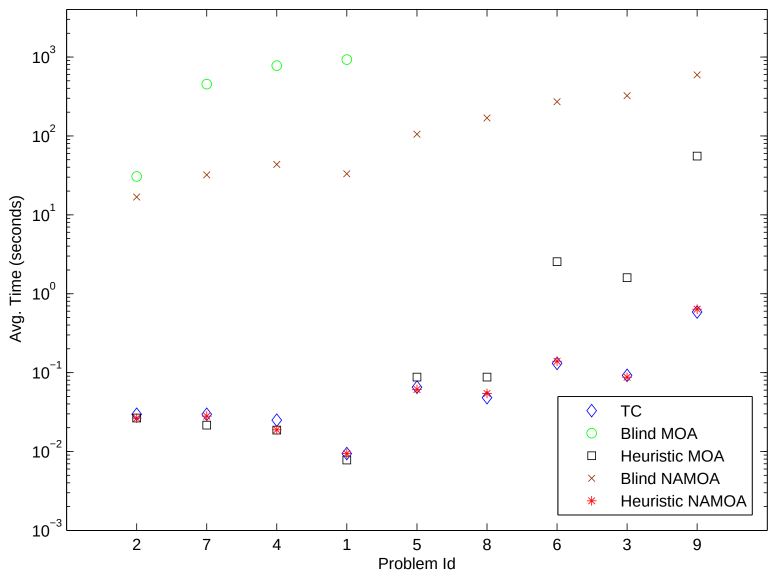


(b) Time requirements

Figure 5.25: Performance of blind and heuristic search on RI road map problems. Average values over ten runs for each problem instance.



(a) Space requirements



(b) Time requirements

Figure 5.26: Performance of blind and heuristic search on NJ road map problems. Average values over ten runs for each problem instance. Values exceeding a 1h time limit are not displayed.

Map	$NAMOA^*$ / MOA^*	TC
DC	0.55	0.75
RI	3.27	4.77
NJ	24.95	37.48

Table 5.4: Average heuristics precalculation time (in seconds) for modified DIMACS road networks.

The analysis of time requirements shows that,

- Heuristic MOA^* and heuristic $NAMOA^*$ perform faster than their blind counterparts respectively.
- Heuristic $NAMOA^*$ was faster on average than heuristic MOA^*. This is specially true on the problems in the larger NJ map, mainly due to the poor performance of MOA^* in instances NJ3, NJ6 and NJ9.
- The difference between heuristic $NAMOA^*$ and TC is very small. On average, $NAMOA^*$ was slightly faster in DC, TC slightly faster in RI, and virtually the same in NJ. In general, the problems are relatively easy for heuristic search, and each instance is typically solved in less than a second by these algorithms.

5.5.3 Time devoted to precalculation of heuristics

It is important to note also that, as in section 5.2, time results shown in this section do not include time devoted to precalculation of heuristics. Precalculation time requirements observed for all instance problems are very similar for the three maps. Average values for the nine problems on each set are presented in table 5.4.

These values are larger in most cases than biobjetive stage search time requirements, as these road map problems correspond to the same range of easy grid problems, e.g. those with small solution depth. Thus, precalculation time was not added to results. In the case of TC, the h_{mix} values must be calculated in addition, resulting in a *worse total execution time than heuristic $NAMOA^*$ in all the road map problems analyzed.*

Besides, as it was previously pointed out for grids, the precalculation procedure can be greatly improved because the heuristic values of many nodes are not used and do not need to be precalculated. The chapter 6 presents a bounded calculation method for $\vec{h}_{TC}$ heuristic.

5.6 Conclusions

This chapter presents the first systematic comparison of the space and time performance of the three different multiobjective heuristic best-first (label setting) search algorithms $NAMOA^*$, MOA^* and TC, using lexicographic order in the selection of nondominated alternatives, and applying the accurate general heuristic H_{TC} proposed by Tung & Chew (1992) also to MOA^* and $NAMOA^*$. Pair-by-pair comparisons through extensive evaluations over different test domains have been performed.

As expected from previous theoretical results, $NAMOA^*$ presents the best space performance. Additionally, several interesting phenomena were observed. In the first place, the use of heuristic information in selection strategies can deteriorate time performance. This effect was found to increase with decreasing correlation. In particular the very accurate heuristic selection strategy used by TC was found to work against its time performance. The node selection strategy of MOA^* was also found to be specially penalized in time performance by the use of the H_{TC} heuristic. Therefore, we can conclude that heuristic $NAMOA^*$ performs better than the other algorithms in most cases, and specially for the hardest problems, i.e. those with deeper solutions and negative correlation between objectives.

In summary, the following conclusions can be drawn:

- Blind MOA^* can be faster than blind $NAMOA^*$ in certain cases (very high correlation or very small solution depth). Otherwise, blind $NAMOA^*$ outperforms blind MOA^*. The memory overhead in MOA^* was found to stabilize around 15-25% in the worst cases analyzed.

- The use of heuristic information can result in important savings in time and space in $NAMOA^*$. However, in particular cases like class I grid problems with low correlation, the advantage in space can be minimal and time performance can degrade. In those cases the reduction in the number of labels explored does not seem to compensate the time overhead of finding solutions earlier.

- The use of heuristic information saves space in MOA^*, but regrettably degrades considerably time performance in all cases. To the author's knowledge, this result had never been reported in the literature. This empirical finding is in accordance with the formal analysis presented in chapter 4.

- $NAMOA^*$ performs somewhat better than TC. Space requirements are similar but the h_{mix} heuristic penalyzes the time performance in TC.

- Heuristic $NAMOA^*$ performs better than the other algorithms in most cases, specially for the hardest problems (i.e. deeper solutions and negative correlation between objectives).

- The speed of the algorithms was found to be related to the discovery of nondominated solutions. Those algorithms that found solutions later in the search performed consistently faster. In this sense, the use of a specialized selection heuristic in TC with heuristic selection was found to work against the time efficiency of the algorithm.

- The results indicate that the order of selection among nondominated open labels is an important element in time efficiency and suggest that the investigation of alternative orderings that combine heuristic search and delayed expansion of solutions could lead to more efficient algorithms. This is analyzed in the next chapter, where linear order is evaluated for $NAMOA^*$, and compared to lexicographic order.

- The experiments confirmed that the time needed to calculate the heuristics is not significant compared to total execution time for class I problems. However, it must be improved for class II problems as it can represent a significant time for some instance problems.

- The road map problems analyzed in this chapter have similar time requirements to grid problems with smaller solution depth. Large size realistic road maps, analogous to difficult class II problems, are used in chapter 6, where an improved calculation method for heuristics is also tested.

Chapter 6

Multiobjective Heuristic Search in Road Maps

The heuristic function $\vec{h}_{TC}$ proposed by Tung & Chew (1992) has been shown to pay off during the multiobjective search stage in many cases. In particular, heuristic search seems to be very effective on the modified road maps proposed by Raith & Ehrgott (2009). However, the total execution time including the precalculation of heuristics questions the application of this heuristic for some road map problems. The heuristic values are precalculated for all nodes in the map, even for those that are never explored in multiobjective search. Thus, the heuristic precalculation procedure can be improved for many cases. In this chapter a bounded precalculation procedure is presented. The heuristic is successfully applied to realistic large sized road maps.

Besides, the empirical study has confirmed that $NAMOA^*$ has the lowest memory requirements among the algorithms analyzed. The formal analysis performed in chapter 4 shows a degradation in time for MOA^*, and the empirical analysis performed in chapter 5 confirms that MOA^* is not competitive in time. TC shows a somewhat worse performance than $NAMOA^*$ for many problems, as expected from formal analyses. But for the modified road maps tested in the last chapter, it is unclear which algorithm has the best time performance in practice. Moreover, TC presents a linear selection rule while $NAMOA^*$ has been applied with lexicographic selection. A fair comparison between both algorithms should consider the application of linear aggregate rules in the selection of nondominated open alternatives. This is analyzed also in this chapter.

An additional related scenario is analyzed as well: hazardous material transportation (hazmat) problems. Random graphs and road maps with three objectives allow the evaluation of the relative performance compared to two objectives.

The chapter is organized as follows: first, a summary of related work and previous results in single and multiobjective search on route planning problems can be found in section 6.1. Later, section 6.2 describes some heuristics useful for multiobjective route planning. An improved procedure for the calculation of the $\vec{h}_{TC}$ heuristic is described. Sections 6.3 and 6.4 show the application of these heuristics to large size road map problems from the "9th DIMACS Implementation Challenge: Shortest Paths". The former experiments minimize both time and physical distance. The latter present a more realistic and difficult scenario where time and economical cost are simultaneously optimized. Moreover, the application of linear orderings is also evaluated in section 6.4. The chapter is concluded with the consideration of another realistic road map scenario in section 6.5: hazmat problems. Some conclusions are drawn at the end of the chapter.

The results presented in this chapter have appeared in Expert Systems with Applications: An International Journal (Machuca & Mandow, 2012), the national portuguese AI conference EPIA'2011 (Machuca & Mandow, 2011) and the national spanish AI conference CAEPIA'2011 (Machuca et al., 2011).

6.1 Multiobjective Route Planning

Some objections to the application of exact heuristic search techniques to large size realistic problems have been made in the past. Several authors have pointed out that particular classes of multiobjective search problems do not exhibit exponential worst-case behavior (Müller-Hannemann & Weihe, 2006; Mandow & Pérez de la Cruz, 2009). In particular, in polynomial state spaces with bounded integer costs and a fixed number of objectives, the number of nondominated solutions can be shown to grow only polynomially with goal depth in the worst case (Mandow & Pérez de la Cruz, 2009). Since road maps are generally defined over a surface, the number of nodes typically grows quadratically with node depth in the worst case, assuming node density is locally upper bounded by some value. Therefore, as long as the number of objectives is fixed and arc costs are bounded and integer, the problem becomes formally tractable. The empirical evaluation of the last chapter has shown that the precalculated multiobjective heuristic $\vec{h}_{TC}$ can be very beneficial in practice, justifying the use of precalculated heuristics.

Applications of multiobjective search in road maps range from off-line planning of routes for hazardous material (hazmat) transportation (Erkut et al., 2007; Dell'Olmo et al., 2005; Caramia et al., 2010) to route search in car navigation systems (Kanoh, 2007; Kanoh & Hara, 2008; Kim et al., 2009; Schultes, 2008; Delling et al., 2009). An overview of multiobjective routing problems can be found in (Jozefowiez et al., 2008). However, most previous applications of multiobjective search have involved approximation schemes or relatively small sized problems. This chapter presents the successful application of multiobjective heuristic search to large size realistic multiobjective road map search problems.

6.1.1 Single-objective Route Planning Algorithms

Route planning in road maps is a current research topic among transportation problems. Current techniques generally rely on exhaustive precalculations that can be later used to speed up route queries. Travel time, distance or economic cost are frequent objectives to be minimized, and most current road planners offer the opportunity to optimize each of them individually. Most recent contributions concentrate on single objective problem formulations (Schultes, 2008; Delling et al., 2009).

Pearl (Pearl, 1984, sec 5.3) presented a formal analysis on the relative performance of Dijkstra's and the A^* algorithm (using an Euclidean distance heuristic) on road maps with randomly distributed cities and uniform density. Some other empirical studies were performed with real road networks using A^* in the past (Zhan & Noon, 1998). New techniques have been developed in the recent years to further improve search efficiency, like precomputing distance bounds (Goldberg & Harrelson, 2005) or the use of bidirectional A^* (Nannicini et al., 2008).

But most of the recent works concentrate on a variety of speed up techniques for Dijkstra's algorithm. A brief description of these techniques can be found in section 2.1.3. In general, speed up techniques exploit information gathered in previous extensive searches of the road map. The challenge is to achieve fast shortest-path queries with practical preprocessing time and memory. A recent work established that the optimal adjustment in many recent techniques (for example, the assignment of landmarks to a graph in the ALT technique cited above) is NP-hard (Bauer et al., 2010a). In practice, these adjustments are frequently settled experimentally.

6.1.2 Multiobjective Route Planning Algorithms

Multiobjective search in road maps is a relatively unexplored area when compared to single-objective search. Limited experiments with multiobjective genetic algorithms in dynamic networks for car navigation are described in (Kanoh, 2007; Kanoh & Hara, 2008; Kim et al., 2009). These involve search in networks with hundreds or a few thousand nodes and three or four objectives. Martins' multiobjective search algorithm has been applied to route planning of hazardous materials in networks with a few hundreds of nodes with two (Dell'Olmo et al., 2005) and three objectives (Caramia et al., 2010). A typical objective in these applications is the minimization of risk, frequently complemented with the minimization of time and/or distance. An overview of the application of multiobjective route planning to hazardous material transportation can be found in (Erkut et al., 2007). Some attempts have been made to extend speed up techniques typically used with single-objective route planning to the multiobjective case. Delling and Wagner (Delling & Wagner, 2009) extended the single-objective SHARC (Shortcuts + ARC-flags) technique to be used with Martins blind multiobjective label-setting algorithm. Reports on the exact solution of problems with two objectives and networks of up to 77,740 nodes are presented. An approximation technique is successfully used to search much larger networks, where the extensive multiobjective preprocessing searches are not practical (Delling & Wagner, 2009).

Finally, Raith & Ehrgott (2009) compared a number of blind multiobjective search algorithms on different test cases, including modified road networks of up to 330,386 nodes and two objectives (time and distance). The analysis concluded that a two-phase method combining two multiobjective label-setting searches provided the best results in these networks.

Regarding multiobjective heuristic search, a partial analysis with modified DIMACS road maps was presented in section 5.5. To the author's knowledge, before this thesis no previous results on the application of heuristic multiobjective search to road maps had been reported. The combination of $NAMOA^*$ with the $\vec{h}_{TC}$ heuristic on a set of random problems obtained from realistic road networks of up to 1,070,376 nodes is tested in the next section. A more efficient calculation method for the $\vec{h}_{TC}$ heuristic is also described.

6.2 Heuristics for Multiobjective Route Planning

The experiments described in the following sections use the $NAMOA^*$ and TC algorithms to solve problems over the road networks described in section 3.2.3. The efficiency of these algorithms can be improved using an *optimistic* and *well informed* heuristic function $H(n)$ to estimate path costs (see section 2.4.4). If the estimates are lower bounds, then the algorithms are guaranteed to terminate with the set of all nondominated solutions. In general, concerning to consistent heuristics, more precise estimates should result in more efficient search. Therefore, finding accurate heuristic estimates is a central issue in multiobjective heuristic search.

Several cost functions will be considered in this chapter: time, distance, economic cost, and societal risk. Different lower bounds can be devised to estimate physical distances and travel times. All the multiobjective heuristic functions considered in this section return a single vector estimate that lower bounds *all* nondominated solution paths emanating from a given node, i.e. $H(n) = \{\vec{h}(n)\}$. Therefore, we shall refer to the multiobjective heuristic function for simplicity as $\vec{h}(n)$. The basic approach is to use $\vec{h}_0(n) = (0,0)$ which amounts to *uninformed* or *blind* search. Two well informed heuristic functions are used as well: a classical distance heuristic, namely $\vec{h}_{cd}(n)$, and the same heuristic function used in the previous chapter, namely $\vec{h}_{TC}(n)$.

6.2.1 Corrected great circle distance heuristic

Let us consider first an estimate for physical distance. Euclidean distance is a simple way to estimate distances between nodes in a planar surface. However, this measure presents several drawbacks for the maps considered in this study. In the first place, node positions are described through longitude and latitude, and not by coordinates in a plane. Secondly, arc distances are calculated in the maps using the *great circle* distance as explained in section 3.2.3. Finally, arc distances in the maps are truncated and paths frequently accumulate errors (up to 0.5 units per arc). Therefore, the straightforward calculation of Euclidean or great circle distances between nodes can easily overestimate optimal distances calculated using map data. This yields both measures inadmissible and inconsistent in general.

Nevertheless, great circle distances can be *corrected* subtracting the maximum possible error to obtain an admissible heuristic. Let n be some node, and γ the destination node. Let $d(n)$ be the great circle distance between n and γ, and $d^*(n)$ the actual optimal physical distance of a path joining both nodes. Let us further assume this optimal path actually joins both nodes following a circular line over the Earth's surface. Ideally, the equality $d(n) = d^*(n)$ should hold. However, due to truncations in physical distance, $d(n)$ will in general overestimate $d^*(n)$. The worst overestimate would occur if the path were made up of a large number of cocircular arcs, each one accumulating a maximum truncation error of 0.5 units.

The cumulative error in this case can be upper bounded as follows. Let us group all the arcs in the graph in disjoint categories such that each category groups all arcs with the same physical distance, and there are N_i arcs of physical distance L_i for each category i. Let us further assume that $\forall i < j \; L_i < L_j$. The truncation error $e(n)$ of a straight path joining n with γ is upper bounded as $e(n) = 0.5 \times N(n)$ where $N(n)$

is the maximum number of map arcs that can be concatenated to reach a distance of $d(n)$, i.e. N_1 arcs of length L_1 plus N_2 arcs of length L_2, etc. More precisely, let us define a_i as the cumulative physical distance of all arcs in the graph up to category i,

$$a_i = \sum_{j=0}^{i} N_j \times L_j \tag{6.1}$$

and let k be a value such that,

$$a_k \leq d(n) < a_{k+1} \tag{6.2}$$

then,

$$N(n) = \sum_{j=0}^{k} N_j + \lceil \frac{d(n) - a_k}{L_{k+1}} \rceil \tag{6.3}$$

The corrected distance value $h_{cd}(n) = d(n) - e(n)$ is always a lower bound of the optimal *physical distance* calculated in the maps from n to the goal. A test over 10000 randomly chosen pairs of nodes in each map revealed that the average error $e(n)$ subtracted in the heuristic calculations is just 0.67% of the great circle distance. In consequence, the estimate preserves most of the precision of great circle distances, while guaranteeing at the same time the admissible (optimistic) values needed by $NAMOA^*$.

Notice that the a_i, N_i and L_i values can be easily precalculated for each map and are problem independent. Therefore, the calculation of heuristic estimates is quite efficient.

Regarding travel time, the optimistic estimate is to assume that the corrected great circle distance can be traversed at the maximum speed, i.e. a time estimate is calculated dividing by the largest factor, which is 1.0 in the DIMACS maps. Since $h_{cd}(n)$ is a lower bound on the physical distance of optimal paths from n to the goal, then $h_{cd}(n)/1.0$ is also a lower bound on travel time.

In consequence, the *corrected distances* multiobjective heuristic estimate is defined as $\vec{h}_{cd}(n) = (h_{cd}(n), h_{cd}(n))$. This heuristic function has been defined for obtaining a (distance,time) estimate, as it will be applied only in that scenario. In the case of economic cost, similar assumptions could be made to obtain a lower bound, but only the $\vec{h}_{TC}$ heuristic will be used in that scenario in this chapter.

6.2.2 Bounded calculation for the TC heuristic

The TC heuristic is defined as $\vec{h}_{tc}(n) = (c_1^*(n), c_2^*(n))$ for the two objectives case, where $c_i^*(n)$ is the optimal cost of a path from n to the goal, considering only the i-th cost component. These values are precalculated for each component by reversing all arcs in the graph, and applying Dijkstra's algorithm to find the shortest path from the goal node to all other nodes in the graph.

Notice that the TC heuristic estimates are trivially more accurate than those described in section 6.2.1. However, TC estimates depend on the goal node and generally need to be precalculated for each problem instance. The calculation procedure is also much more costly computationally.

The preprocessing described above calculates estimates for *all* nodes in the graph (see section 2.4.4). It is obvious that many problem instances require the examination of fewer nodes.

Let C^* be the set of nondominated solution costs for a given problem. It can be shown that $NAMOA^*$ will never consider for expansion paths whose estimates are dominated by some vector in C^* (Lemma 4.1). This property can be conveniently used to limit the number of nodes examined in the precalculation of the heuristic values. Let us denote by c_1^*, c_2^* the optimal solutions for the single objective under consideration. Let c_2' be the minimum value of the second objective among all solution paths attaining c_1^*, and c_1' the minimum value in the first objective among those paths attaining c_2^* in the second one.

Figure 6.1 displays these values in a biobjective cost space. All nondominated solution costs to a problem $\vec{c}^{\,*} \in C^*$ lie by definition in rectangle A defined by the two extreme points [1] (c_1^*, c_2') and (c_1', c_2^*) .

Lemma 4.1 implies that $NAMOA^*$ will never consider for expansion paths in rectangle C, with costs dominated by (c_1', c_2'), since these can never lead to nondominated solutions. Therefore, heuristic estimates for nodes n with optimal costs $c_1^*(n) > c_1'$ and $c_2^*(n) > c_2'$ do not need to be calculated (i.e. they could be set to infinity).

The following three-stage calculation procedure is proposed,

1. Reverse all arcs in the graph, and apply a modified Dijkstra's algorithm to find the shortest path from the goal node to other nodes in the graph. The modified Dijkstra's algorithm minimizes c_1 values, but in case of ties, prefers paths with smaller value of c_2, i.e. optimizes using a total lexicographic order [2] defined by $(c_1(n), c_2(n))$. Pause search as soon as the start node is selected for expansion. This will provide the optimal lexicographic cost (c_1^*, c_2').

2. Reverse all arcs in the graph, and apply a modified Dijkstra's algorithm to find the shortest path from the goal node to other nodes in the graph using a lexicographic order defined by $(c_2(n), c_1(n))$. Stop search as soon as a node n selected for expansion satisfies $c_2(n) > c_2'$. Notice that the start node will be selected at some time during this search, and labelled with the optimal lexicographic cost (c_1', c_2^*).

3. Resume the search paused in stage 1, and stop as soon as a node n selected for expansion satisfies $c_1(n) > c_1'$.

Notice that this procedure calculates exactly the same heuristic values of the TC heuristic, except for those nodes that will never be considered by $NAMOA^*$ (i.e. those with estimates outside rectangle A figure 6.1). These missing values can be set to infinity.

[1] Note that the TC heuristic estimate for the start node is (c_1^*, c_2^*), which trivially dominates all points in rectangle A

[2] A detailed analysis of such algorithm can be found elsewhere (Mandow & Pérez de la Cruz, 2003)

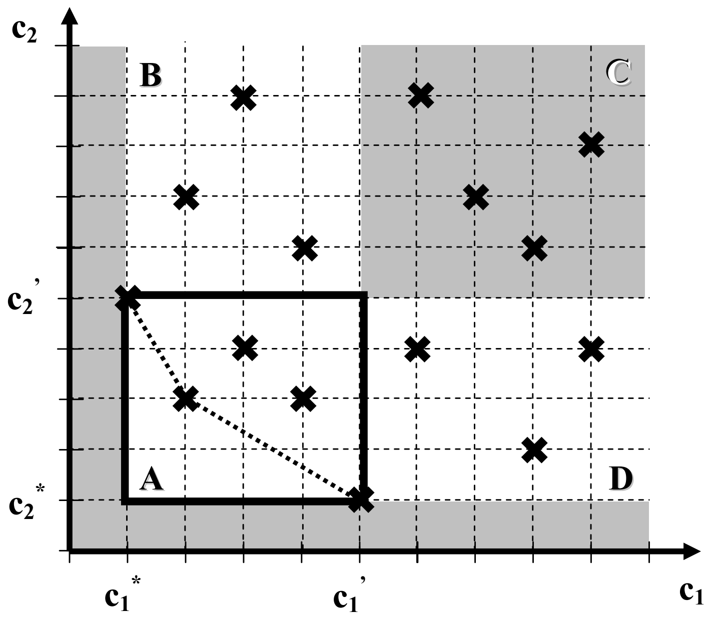


Figure 6.1: A typical Pareto-front in cost space

6.3 Route Planning: Time vs. Distance

The experiments described in this section have been performed over the DIMACS road map problems described in section 3.2.3. The performance of $NAMOA^*$ with different heuristic functions has been analyzed over 200 random problems in four road maps of increasing size with two objectives: time and physical distance. More information about these maps can be found in table 3.2. Fifty random problem instances were generated for each one of the road maps. Each problem instance was solved 5 times with each heuristic. Average time values are presented in the results.

The heuristics under evaluation are blind search ($\vec{h}_0$), corrected great circle distance ($\vec{h}_{cd}$), and Tung & Chew heuristic ($\vec{h}_{TC}$) with both the original method and the improved calculation method. Time values presented for $NAMOA^*$ with $\vec{h}_{TC}$ include heuristic precalculation time plus multiobjective search time. For each problem instance the Tung & Chew heuristic was precalculated 10 times using the method of choice (see section 6.2.2). Average precalculation values are added in the results to $NAMOA^*$ as heuristic precalculation time when these heuristics were applied.

The algorithms were implemented in ANSI Common Lisp using LispWorks 6.0 Enterprise 64 bits, and run on a Sun Fire X4140 server with 2 six-core AMD Opteron 2435 @ 2.60GHz processors and 64 Gb of DDR2 RAM, under Windows Server 2008 R2 Enterprise (64-bits).

The time requirements of $NAMOA^*$ for the NY, BAY, COL and FLA maps are presented in figures 6.2, 6.3, 6.4, 6.5 respectively. The abscissa axis x shows problem indexes for each instance in the 50 problem set for each map. Instances are ordered according to increasing time taken by $NAMOA^*$ with heuristic $\vec{h}_{TC}$ calculated using the bounded method. The value for ordinate y (time in seconds) is shown in a logarithmic scale.

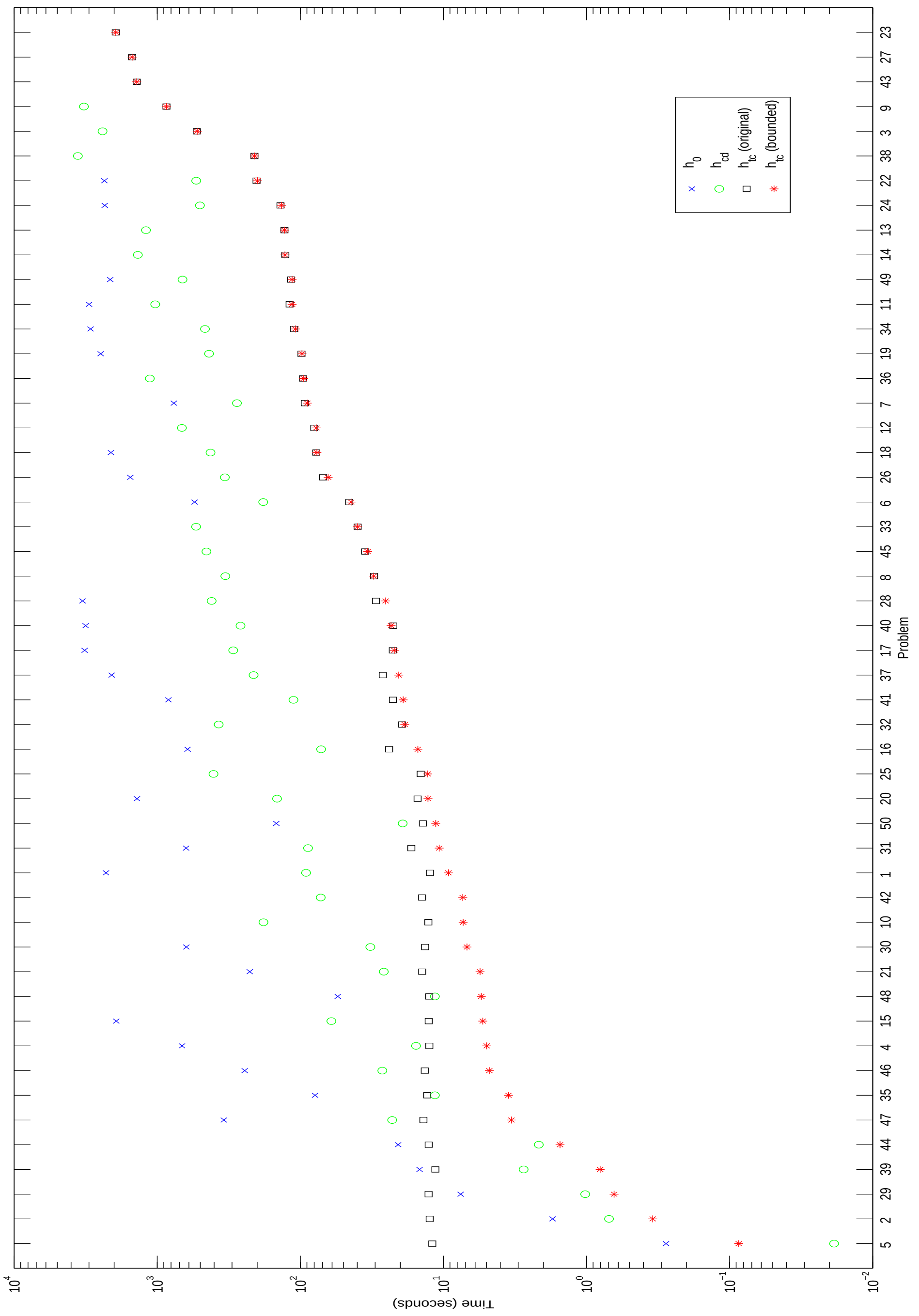


Figure 6.2: Time requirements for $NAMOA^*$ with blind and heuristic search in NY map

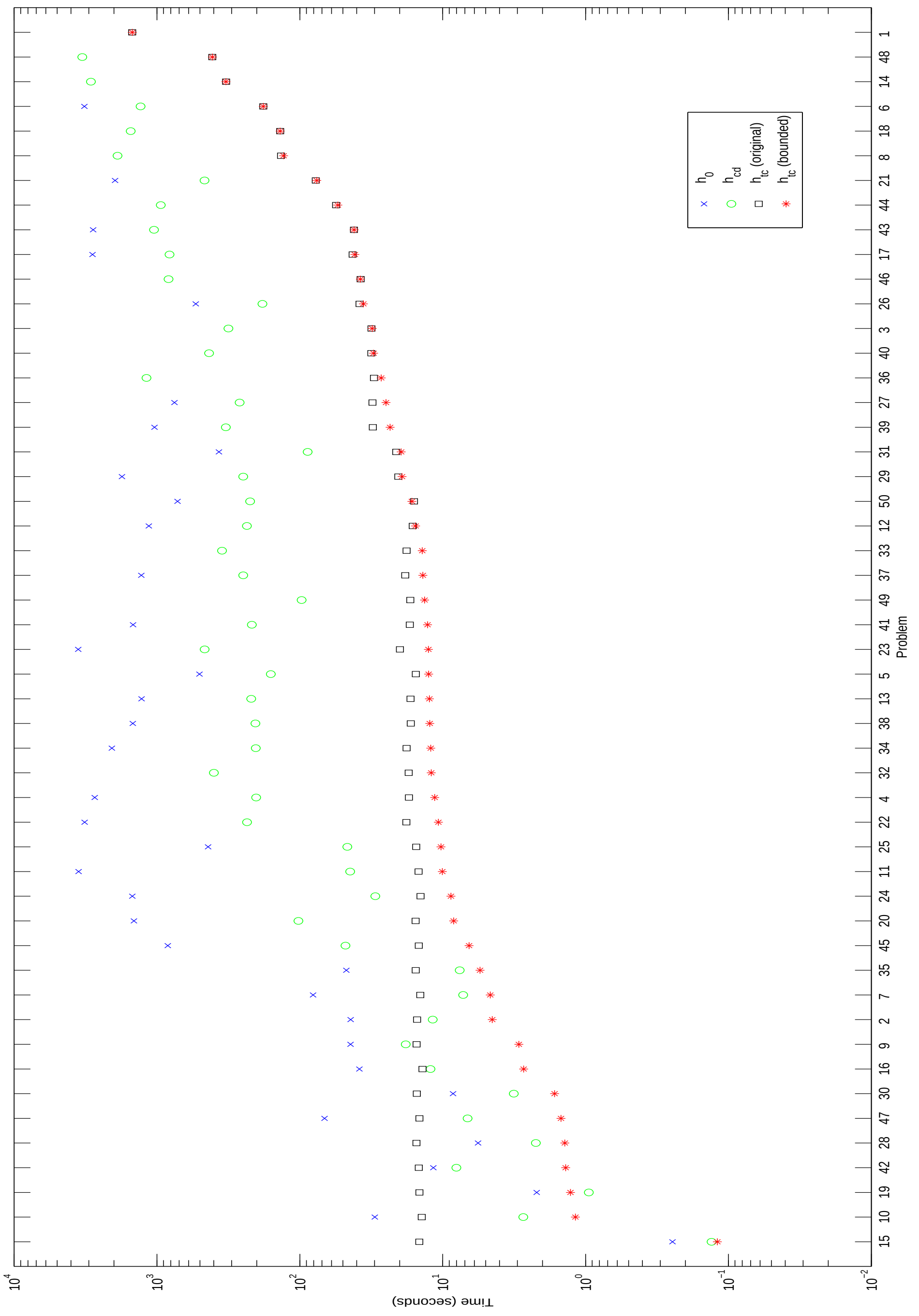


Figure 6.3: Time requirements for $NAMOA^*$ with blind and heuristic search in BAY map

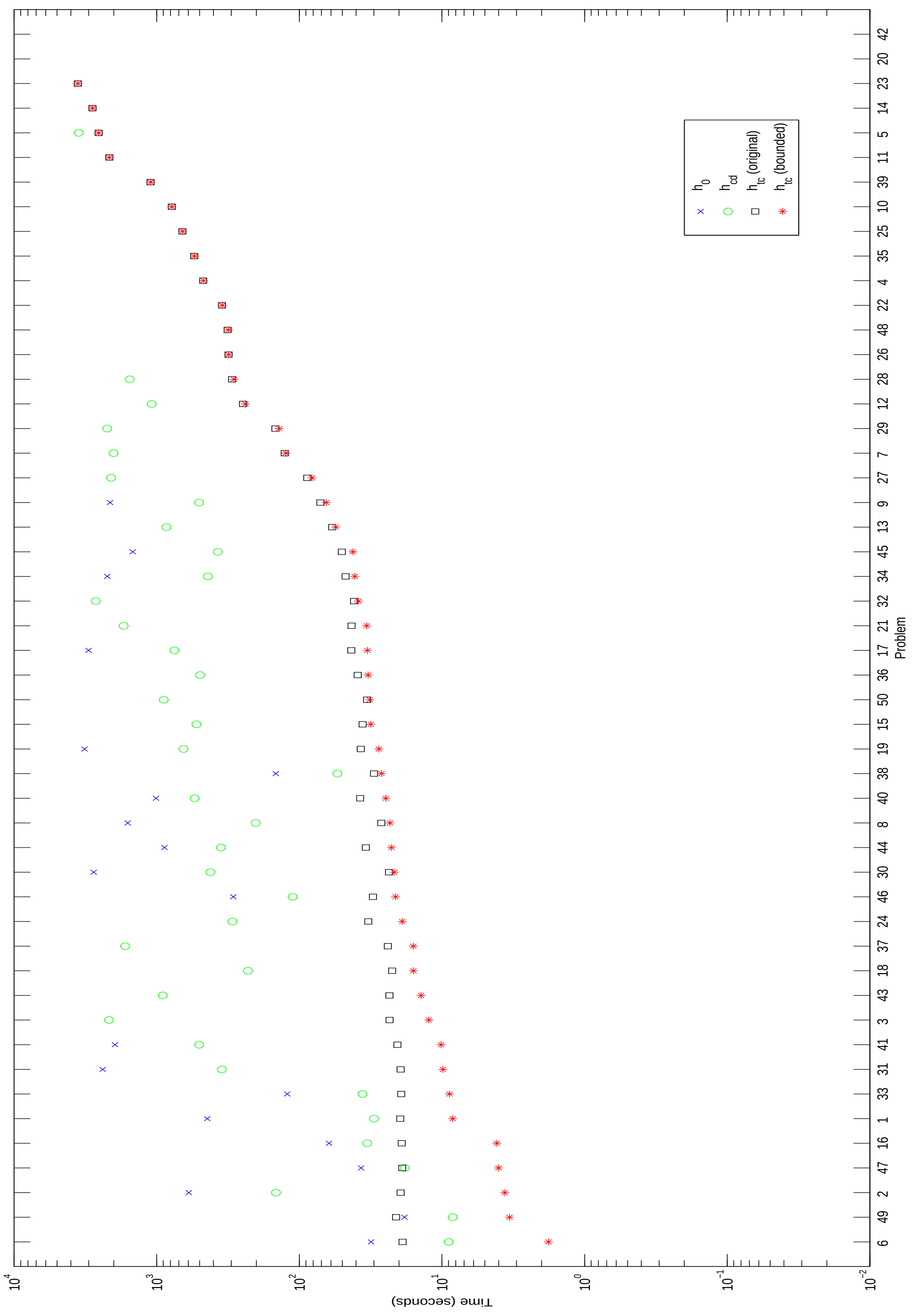


Figure 6.4: Time requirements for $NAMOA^*$ with blind and heuristic search in COL map

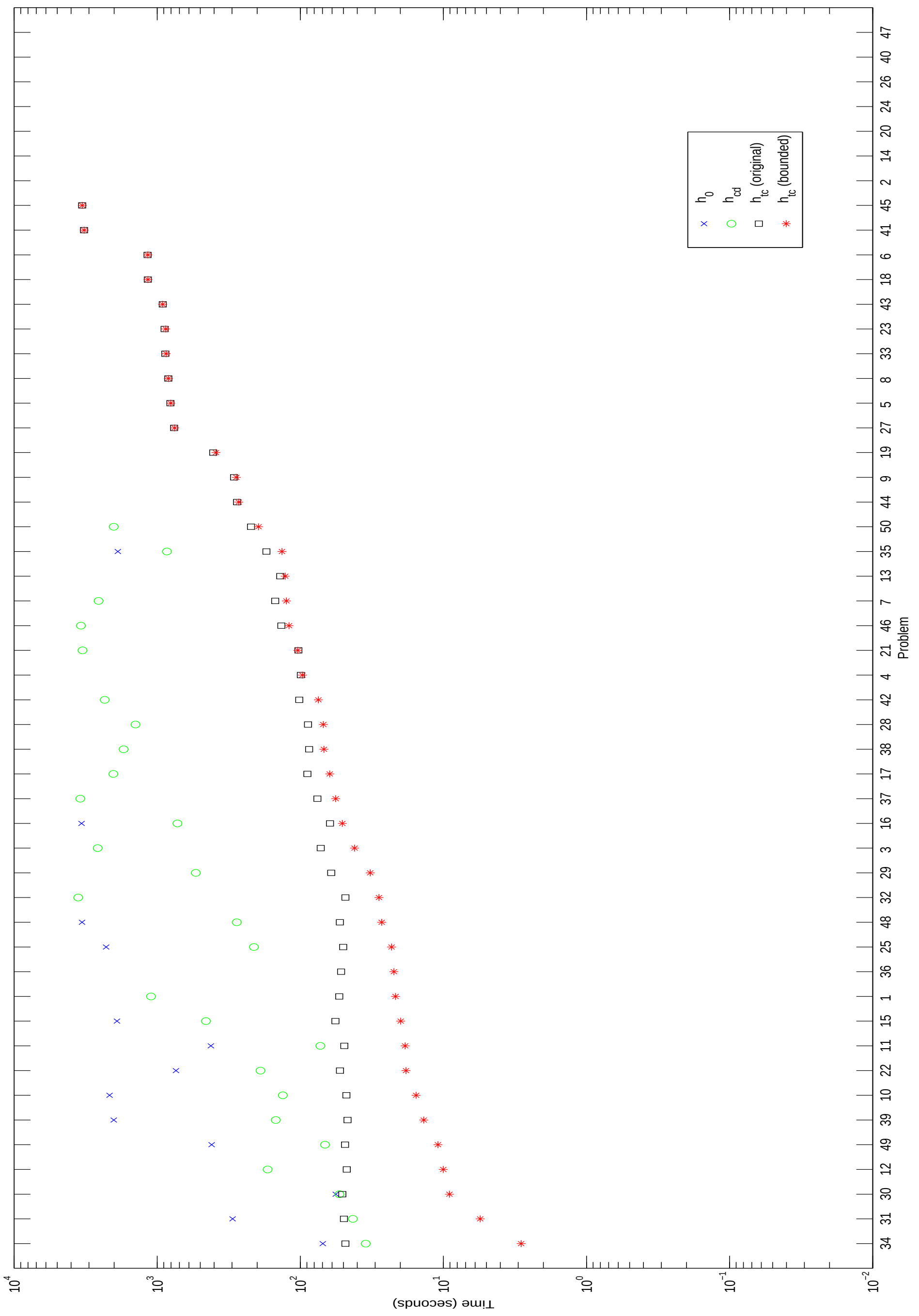


Figure 6.5: Time requirements for $NAMOA^*$ with blind and heuristic search in FLA map

Heuristic	NY	BAY	COL	FLA
h_0	17	12	30	37
h_{cd}	3	1	13	23
h_{tc}	0	0	2	7

Table 6.1: Number of road map problem instances that could not be solved in 1h by $NAMOA^*$ with lexicographic selection

ID #	NY38	BAY6	BAY23	BAY48	COL5	COL23	FLA32	FLA41,45	FLA48
$\vec{h}_0$	0	3	2	0	0	0	0	0	4
$\vec{h}_{cd}$	1	5	5	1	1	0	3	0	5
$\vec{h}_{tc}$	5	5	5	5	5	1	5	3	5

Table 6.2: Number of successful executions for road map problem instances that exceeded at least once, but not always, the time limit with $NAMOA^*$ and lexicographic selection

For practical reasons, instances were solved each time with a one-hour time limit. For certain instances, none of the five executions performed with a given heuristic where able to solve the problem within the time limit. In such cases, values are not displayed in the figures for that particular heuristic. Table 6.1 shows the number of problem instances that were unsolvable within the time limit for each map and heuristic. In a few instance-heuristic combinations only some of the five executions exceeded the time limit. Therefore, these seem to require an average time very close to the limit. Averages presented for these instance-heuristic combinations take into account only successful executions. Table 6.2 lists these particular instances, and how many successful executions were run within the time limit for each heuristic.

6.3.1 Problem data

Tables 6.3, 6.4, 6.5, 6.6 display the experimental data shown in figures 6.2, 6.3, 6.4, 6.5 respectively. Additional information is presented for each problem instance, like source and destination identifiers in the DIMACS challenge maps, multiobjective search time of $NAMOA^*$ (without precalculation time) for the $\vec{h}_{TC}$ heuristic, and the number of different nondominated solution cost vectors $|C^*|$. Average search times for the different heuristics are in seconds. Times for problem instances that could not complete all 5 runs within the one-hour time limit are averaged only over successful runs and are marked with underlined text.

6.3.2 Blind search vs. Heuristic search

The results presented in section 6.3 show that the heuristic approaches ($\vec{h}_{cd}$, $\vec{h}_{TC}$) clearly outperform blind search ($\vec{h}_0$), both in the number of instances solved (see table 6.1), and in run time (see figures 6.2, 6.3, 6.4, 6.5). Blind search was not able to solve the full instance set for any of the maps within the time limit, and performed rather poorly failing in 30 and 37 cases out of 50 in the COL and FLA maps respectively.

Prob	Source	Goal	t_{h_0}	$t_{h_{cd}}$	$t_{h_{tc}}(o)$	$t_{h_{tc}}(b)$	t_{namoa}	|C*|
1	33502	163335	2279.89	91.22	12.43	9.22	0.85	45
2	198561	195430	1.73	0.70	12.47	0.34	0.04	12
3	40851	4310	-	2408.30	528.29	525.87	515.72	344
4	19103	95503	670.30	15.53	12.53	4.98	0.24	24
5	65190	57030	0.28	0.02	11.95	0.09	0.02	1
6	172882	189944	548.33	181.62	45.52	44.17	32.87	163
7	181176	151910	764.42	277.22	92.99	89.46	80.06	308
8	177414	103345	-	334.00	30.56	30.72	18.66	122
9	186166	71968	-	3252.33	862.23	862.22	850.69	487
10	50616	76333	-	180.79	12.75	7.27	1.04	31
11	56699	159358	2992.14	1031.99	118.74	114.02	106.95	401
12	103987	175817	-	672.40	79.80	77.33	67.92	213
13	75533	165171	-	1198.00	128.96	129.05	117.45	245
14	191865	72103	-	1364.85	127.17	128.17	115.90	346
15	35170	237017	1932.06	60.71	12.66	5.31	0.82	26
16	207442	156433	613.34	71.63	23.96	15.08	12.44	69
17	62306	134007	3211.47	294.02	22.60	22.05	10.22	78
18	58427	135252	2104.84	424.20	77.26	76.63	64.46	242
19	91985	200812	2484.05	433.97	98.17	97.36	85.71	241
20	242644	163590	1385.60	145.47	15.16	12.83	3.78	156
21	40180	100359	225.49	26.11	14.07	5.54	2.32	77
22	38497	207344	2340.13	534.32	202.15	199.53	190.24	465
23	180834	83150	-	-	1947.98	1948.15	1936.36	814
24	129948	7003	2324.38	503.12	137.47	135.24	125.34	234
25	259195	173121	-	404.64	14.40	12.87	2.09	72
26	147806	136543	1541.13	337.06	69.30	63.93	55.57	371
27	179874	57536	-	-	1495.45	1495.73	1483.94	643
28	189934	31336	3327.36	416.25	29.56	25.34	17.02	169
29	138263	253856	7.58	1.02	12.69	0.64	0.04	11
30	246144	166336	625.33	32.43	13.43	6.84	1.00	65
31	25610	143842	627.76	88.28	16.76	10.71	4.21	86
32	228779	167251	-	372.19	19.49	18.60	8.03	162
33	78936	34136	-	535.16	39.74	39.86	27.58	111
34	124173	138439	2924.85	463.34	110.40	108.12	96.66	295
35	260563	233292	78.90	11.46	12.98	3.51	0.36	36
36	193168	66816	-	1127.19	95.75	94.74	83.40	280
37	29432	29834	2080.89	212.23	26.54	20.50	13.57	131
38	193241	144927	-	<u>3586.53</u>	208.97	209.09	196.63	787
39	161522	171446	14.66	2.75	11.37	0.80	0.02	1
40	176910	109129	3160.91	261.96	22.42	23.21	10.45	164
41	251416	53900	835.19	111.63	22.52	19.09	9.11	106
42	201505	262626	-	71.97	14.11	7.32	1.61	48
43	86937	190907	-	-	1389.32	1389.98	1377.66	632
44	35252	18638	20.72	2.16	12.66	1.53	0.03	4
45	92562	65120	-	452.98	35.23	33.85	23.14	202
46	230423	2724	244.50	26.75	13.50	4.78	1.11	66
47	17285	92411	342.28	22.80	13.79	3.35	0.90	17
48	177037	199832	54.72	11.47	12.55	5.44	0.07	8
49	68330	206280	2130.97	665.56	115.93	114.12	102.93	270
50	61414	50367	147.27	19.25	13.89	11.32	1.47	50

Table 6.3: Data from NY map problems

Prob	Source	Goal	t_{h_0}	$t_{h_{cd}}$	$t_{h_{tc}}(o)$	$t_{h_{tc}}(b)$	t_{namoa}	|C*|
1	217950	116998	-	-	1489.11	1489.03	1474.76	812
2	251602	34430	44.11	11.74	15.12	4.50	0.29	9
3	98882	122447	-	315.91	31.53	31.16	17.05	207
4	52642	224747	2725.53	202.15	17.23	11.38	3.59	43
5	139849	299950	504.02	159.59	15.41	12.53	1.06	46
6	227292	149896	3219.98	1302.36	180.20	180.24	166.23	374
7	184605	142630	80.72	7.19	14.34	4.65	0.98	27
8	61991	285138	-	1887.31	135.24	129.96	120.68	436
9	157468	111871	44.19	18.13	15.25	2.93	0.46	18
10	8057	53146	29.85	2.73	14.02	1.18	0.04	9
11	63481	317962	3535.25	44.44	14.75	10.07	1.37	34
12	100852	32440	1139.09	234.63	16.21	15.48	2.74	45
13	88253	223600	1281.66	219.04	16.80	12.39	2.85	51
14	256221	137112	-	2899.57	328.21	329.05	314.01	445
15	296236	87228	0.25	0.13	14.58	0.12	0.00	1
16	79044	118604	38.27	12.16	13.84	2.71	0.03	3
17	101931	260608	2823.59	817.16	42.73	41.23	29.33	137
18	151143	33304	-	1528.02	137.36	137.21	123.21	353
19	252289	254898	2.20	0.95	14.55	1.27	0.04	9
20	168761	275306	1450.15	102.30	15.48	8.38	0.75	42
21	151157	220839	1965.79	464.95	77.32	76.09	62.89	203
22	50213	139654	3205.05	234.38	17.97	10.72	3.10	27
23	6574	319340	3556.67	463.61	19.93	12.59	5.34	104
24	299482	198914	1487.94	29.64	14.28	8.76	0.27	16
25	319552	169293	438.47	46.49	15.33	10.27	0.63	25
26	214934	40905	534.58	182.76	38.23	35.90	22.60	61
27	41820	197615	754.82	263.92	30.99	24.91	16.85	175
28	88417	292062	5.65	2.23	15.27	1.40	0.02	2
29	190939	185572	1756.89	249.12	20.43	19.25	6.12	64
30	132185	126599	8.46	3.18	15.17	1.65	0.10	16
31	307539	72237	368.20	88.17	21.12	19.52	5.79	53
32	167018	129836	-	399.18	17.31	12.02	2.90	77
33	199994	273242	-	350.16	17.90	13.91	3.74	89
34	76122	222337	2068.02	203.48	17.89	12.13	4.03	35
35	60060	15213	47.29	7.60	15.46	5.48	0.77	42
36	35111	110152	-	1184.39	30.28	26.94	15.36	196
37	250483	268079	1285.62	249.11	18.30	13.77	3.00	145
38	177519	38511	1477.15	204.66	16.73	12.33	2.08	82
39	50330	193990	1040.35	329.55	30.82	23.37	16.44	144
40	132178	166402	-	432.11	31.56	30.44	16.94	133
41	193656	269296	1470.89	216.88	17.01	12.78	2.47	64
42	174698	73496	11.64	8.02	14.70	1.38	0.07	7
43	224414	285621	2796.62	1048.87	41.75	41.72	27.35	164
44	304900	212465	-	940.88	55.73	54.15	41.57	167
45	103114	109425	840.32	47.95	14.71	6.54	0.56	32
46	226489	314957	-	830.48	37.56	37.67	23.42	87
47	183973	310416	67.22	6.70	14.56	1.49	0.03	3
48	261308	150051	-	3329.43	409.74	409.49	395.36	499
49	70155	317172	-	97.18	16.86	13.39	2.68	75
50	118218	48399	717.68	222.91	15.90	16.43	1.27	53

Table 6.4: Data from BAY map problems

Prob	Source	Goal	t_{h_0}	$t_{h_{cd}}$	$t_{h_{tc}}(o)$	$t_{h_{tc}}(b)$	t_{namoa}	|C*|
1	186399	206453	442.16	30.01	19.62	8.41	0.92	59
2	106474	399484	595.76	146.04	19.48	3.62	0.12	8
3	219775	41597	-	2166.09	23.30	12.33	3.98	134
4	240731	182571	-	-	471.60	470.38	451.03	655
5	417012	345347	-	$\underline{3530.80}$	2555.49	2552.26	2536.21	544
6	218133	101499	31.42	8.99	18.89	1.79	0.50	33
7	82209	119557	-	2008.11	126.56	124.35	107.79	241
8	93458	188506	1598.14	202.76	26.64	23.17	7.08	106
9	342828	9384	2124.14	505.43	71.07	64.82	52.32	324
10	173475	148741	-	-	783.07	782.61	764.60	1214
11	368975	307038	-	-	2149.43	2146.99	2130.25	1332
12	266403	288429	-	1084.79	247.95	239.11	229.84	477
13	39969	357717	-	856.05	58.79	55.57	40.22	506
14	311623	141945	-	-	2817.74	2817.19	2798.52	2028
15	192685	230403	-	526.20	36.01	31.58	16.36	221
16	242487	387257	61.86	33.47	19.14	4.12	0.46	29
17	260867	187773	3007.22	753.24	43.19	33.29	24.96	194
18	345397	57539	-	229.62	22.35	15.84	2.75	125
19	98052	233707	3206.37	650.37	37.10	27.72	17.92	81
20	16974	272085	-	-	-	-	-	-
21	373200	393176	-	1706.76	43.06	33.73	23.16	200
22	82255	231704	-	-	348.15	346.69	329.30	295
23	170699	374715	-	-	$\underline{3570.71}$	$\underline{3568.19}$	$\underline{3551.32}$	1831
24	344226	41837	-	295.06	32.87	18.98	14.69	253
25	311543	79737	-	-	659.86	658.47	639.73	1119
26	233022	342755	-	-	313.39	313.15	293.43	715
27	287214	273946	-	2090.95	87.99	81.49	69.55	394
28	105391	434721	-	1546.68	295.80	287.43	275.79	407
29	261648	309214	-	2225.72	147.10	138.76	128.19	368
30	277313	107050	2764.01	420.47	23.47	21.63	4.70	70
31	38771	199544	2389.27	349.98	19.50	9.85	0.66	30
32	253430	282880	-	2680.23	41.35	38.61	22.66	178
33	175642	183753	121.57	36.06	19.31	8.86	0.11	6
34	313912	290255	2220.55	438.62	47.34	40.90	28.13	374
35	32506	97065	-	-	546.07	543.17	527.08	388
36	384664	320926	-	497.01	38.95	32.90	20.51	158
37	220613	381762	-	1665.09	23.98	15.88	5.00	169
38	276063	158585	146.34	54.17	29.91	26.57	10.75	89
39	104682	393819	-	-	1105.07	1102.20	1086.65	1272
40	6425	183698	1012.25	542.74	37.47	24.73	17.67	164
41	185170	419257	1959.44	503.84	20.50	10.15	1.12	49
42	299734	236575	-	-	-	-	-	-
43	6364	172422	-	907.95	23.35	14.01	4.83	105
44	108080	330816	882.63	355.54	34.19	22.60	15.38	119
45	60885	371450	1473.93	373.38	50.37	42.11	32.20	80
46	355708	233614	290.35	111.46	30.46	21.14	11.23	384
47	114038	221082	36.88	18.25	18.99	4.01	0.08	18
48	258459	178122	-	-	317.94	314.05	298.18	546
49	431418	72100	18.34	8.40	20.98	3.36	0.34	42
50	296685	11132	-	892.32	33.53	32.07	14.66	245

Table 6.5: Data from COL map problems

Prob	Source	Goal	t_{h_0}	$t_{h_{cd}}$	$t_{h_{tc}}(o)$	$t_{h_{tc}}(b)$	t_{namoa}	|C*|
1	361739	698672	-	1103.68	53.47	21.61	7.95	186
2	686602	27291	-	-	-	-	-	-
3	546667	1044042	-	2604.01	72.01	41.77	27.97	305
4	115105	421966	-	-	99.17	96.59	54.27	298
5	742805	335320	-	-	808.83	804.18	764.49	1187
6	88673	333047	-	-	1168.53	1163.91	1121.30	803
7	766579	263017	-	2567.95	149.87	125.11	103.42	280
8	28100	848660	-	-	835.81	835.32	790.38	779
9	134765	11866	-	-	290.13	279.88	243.91	612
10	158576	949455	2151.40	132.54	47.58	15.50	1.90	13
11	659282	327441	421.83	72.46	49.25	18.53	3.25	83
12	539004	639594	-	169.07	47.31	10.02	1.52	34
13	489044	463492	-	-	137.97	127.43	91.18	290
14	192295	127144	-	-	-	-	-	-
15	481860	1046443	1912.12	456.51	56.81	19.87	10.29	145
16	273776	154436	3375.94	721.89	62.01	50.86	13.97	117
17	946451	513773	-	2023.97	89.21	62.32	43.00	269
18	90921	359195	-	-	1162.02	1160.10	1115.85	848
19	783218	996886	-	-	406.23	388.91	360.44	517
20	1052751	190896	-	-	-	-	-	-
21	646797	149214	-	3321.75	103.10	104.12	58.34	182
22	398569	982263	739.69	189.77	52.85	18.25	5.94	222
23	809772	870827	-	-	887.27	871.37	843.37	605
24	635036	38956	-	-	-	-	-	-
25	234560	955775	2275.86	210.93	50.14	23.02	2.94	56
26	274945	143720	-	-	-	-	-	-
27	716892	344531	-	-	761.94	756.30	715.94	1063
28	516174	154020	-	1419.68	88.40	68.99	41.82	247
29	129998	118211	-	536.97	60.84	32.44	16.36	109
30	905861	756883	56.46	53.19	50.82	9.08	3.87	144
31	41614	404340	296.99	42.90	49.53	5.53	1.60	9
32	933700	561390	-	3566.71	48.41	28.18	1.42	8
33	237886	310889	-	-	877.19	864.33	829.91	824
34	257739	652062	69.65	34.88	48.35	2.86	1.53	21
35	478200	1062969	1885.46	855.62	172.31	134.38	125.32	656
36	173720	246425	-	-	51.84	22.18	7.63	161
37	43974	803673	-	3449.18	75.92	56.61	29.53	214
38	382275	1044332	-	1715.21	86.79	68.37	41.86	310
39	462808	85391	2012.71	148.33	46.82	13.72	2.46	67
40	643063	593489	-	-	-	-	-	-
41	310505	612278	-	-	3248.15	3237.24	3203.14	1168
42	818016	667330	-	2327.15	101.68	74.78	55.41	264
43	257389	17759	-	-	915.11	916.32	870.24	570
44	16738	751658	-	-	276.51	269.02	232.90	371
45	401809	616933	-	-	3341.79	3341.38	3298.90	1217
46	364903	404709	-	3411.00	135.65	119.92	89.59	188
47	624617	364615	-	-	-	-	-	-
48	472495	193187	3353.50	277.91	52.87	26.99	9.56	94
49	131865	294161	416.87	67.16	48.70	10.91	3.87	120
50	363001	263258	-	2011.66	221.10	195.72	175.66	216

Table 6.6: Data from FLA map problems

	Heuristic		
Map	$\vec{h}_{cd}$	$\vec{h}_{TC}(orig)$	$\vec{h}_{TC}(bounded)$
NY	6.20	29.19	33.50
BAY	6.07	49.77	74.37
COL	4.40	40.13	60.52
FLA	5.41	18.18	45.05
Overall	5.52	34.32	53.36

Table 6.7: Speedup of heuristic $NAMOA^*$ with lexicographic selection with respect to $NAMOA^*$ with $\vec{h}_0$

Regarding run time, blind search was beaten in all instances by $\vec{h}_{cd}$ and $\vec{h}_{TC}$ with the improved calculation method. However, blind search was able to beat $\vec{h}_{TC}$ with the original method in 9 of the easiest instances (3 in NY, 5 in BAY, and 1 in COL). This was due to the constant overhead in the heuristic precalculation for this method, regardless of problem difficulty.

For the subset of problem instances that could be solved by blind search [3], the overall average speedup of heuristic search was 5.52 for $\vec{h}_{cd}$, and 34.32 and 53.36 for $\vec{h}_{TC}$ with the original and improved methods respectively. In the case of $\vec{h}_{TC}$ the speedup was much larger for BAY and COL maps than for the NY and FLA ones (see table 6.7).

6.3.3 Performance of heuristic search

Regarding the relative performance of the heuristic approaches, table 6.1 shows that $\vec{h}_{cd}$, although consistently better than $\vec{h}_0$, is clearly outperformed by $\vec{h}_{TC}$ in terms of the number of solved instances. In fact, $\vec{h}_{TC}$ could solve the complete sets for NY and BAY within the time limit, and failed only for 2 and 7 problems in COL and FLA respectively.

Time performance with the improved method for $\vec{h}_{TC}$ outperformed $\vec{h}_{cd}$ in all cases, except for two simple instances (NY 5 and BAY 19), where the overhead of precalculation did not compensate for the very small search effort needed. In fact, even for these instances, $NAMOA^*$ performed less iterations with $\vec{h}_{TC}$ than with $\vec{h}_{cd}$. However, $\vec{h}_{cd}$ was able to beat $\vec{h}_{TC}$ with the original calculation method in 22 of the simpler problem instances. Once again, this was due to the constant overhead in the heuristic precalculation for this method, regardless of problem difficulty.

For the set of problem instances [4] that could be solved by $\vec{h}_{cd}$, the overall average speedup of $\vec{h}_{TC}$ was 11.20 and 14.49 with the original and improved methods respectively (see table 6.8). More precisely, the speedup of $\vec{h}_{TC}$ (improved method) over $\vec{h}_{cd}$ is of 5.94, 12.44, 17.09, and 22.48 for the NY, BAY, COL, and FLA maps, indicating that the Tung-Chew heuristic becomes increasingly better than corrected great circle distance for larger maps. This is probably related not only with graph size, but with the fact that the larger graphs have also larger physical extension.

[3]Instances that could not be solved in all five runs within the 1h time limit by $\vec{h}_0$ are not considered.

[4]Instances that could not be solved in all five runs within the 1h time limit by $\vec{h}_{cd}$ are not considered.

	$NAMOA^*$ with	
Map	$\vec{h}_{TC}(orig)$	$\vec{h}_{TC}(bounded)$
NY	5.59	5.94
BAY	10.53	12.44
COL	14.24	17.09
FLA	14.43	22.48
Overall	11.20	14.49

Table 6.8: Speedup of heuristic $NAMOA^*$ with lexicographic selection and heuristic $\vec{h}_{TC}$ with respect to $\vec{h}_{cd}$

These results confirm that more accurate heuristic estimates result in improved time performance with $\vec{h}_{TC}$, since $\vec{h}_{TC}$ is trivially more informed than $\vec{h}_{cd}$.

6.3.4 Effectiveness of the Tung & Chew heuristic

The results presented in section 6.3.3 clearly indicate that $\vec{h}_{TC}$ is by far the best heuristic among the alternatives considered. Search time taken by $NAMOA^*$ with this heuristic is much smaller than with h_{cd}, the second best alternative (see tables 6.3, 6.4, 6.5, 6.6).

The experiments also clarify the adequacy of heuristics precalculated through search in multiobjective problems. Even when the time taken in the precalculations is added to the search time of $NAMOA^*$, this alternative remains very competitive. The original calculation method calls for a one-to-all search of shortest paths for each objective under consideration. This implies an overhead of about 10 seconds in the NY and BAY maps, and of 11 and 14 seconds in COL and FLA respectively. However, the formal properties of $NAMOA^*$ allow us to bound the number of nodes whose heuristic needs to be precalculated. With this bounded calculation method $\vec{h}_{TC}$ was beaten by h_{cd} only in two simple cases, NY 5 and BAY 19. In NY 5 the precalculations took 2,729 iterations, while $NAMOA^*$ took only 21 and 61 iterations with $\vec{h}_{TC}$ and h_{cd} respectively. In BAY 19, the precalculations took 40,790 iterations, while $NAMOA^*$ performed 613 and 6,446 iterations with $\vec{h}_{TC}$ and h_{cd} respectively.

Figure 6.6 shows the total number of iterations performed by the original TC heuristic calculation and the new bounded method in the case of NY map for the 50 randomly chosen problems. Similar values are obtained for the other maps. The original TC method made exactly the same effort for all problems (i.e. all the nodes in the graph were explored twice). The bounded method can improve efficiency in many cases, though in difficult problems, where source and destination nodes are far apart, it is necessary to calculate heuristic values for all nodes in the graph regardless of the calculation method.

Among the alternatives considered in this analysis, the $\vec{h}_{TC}$ heuristic with the bounded calculation method is clearly the option of choice for multiobjective search in road maps.

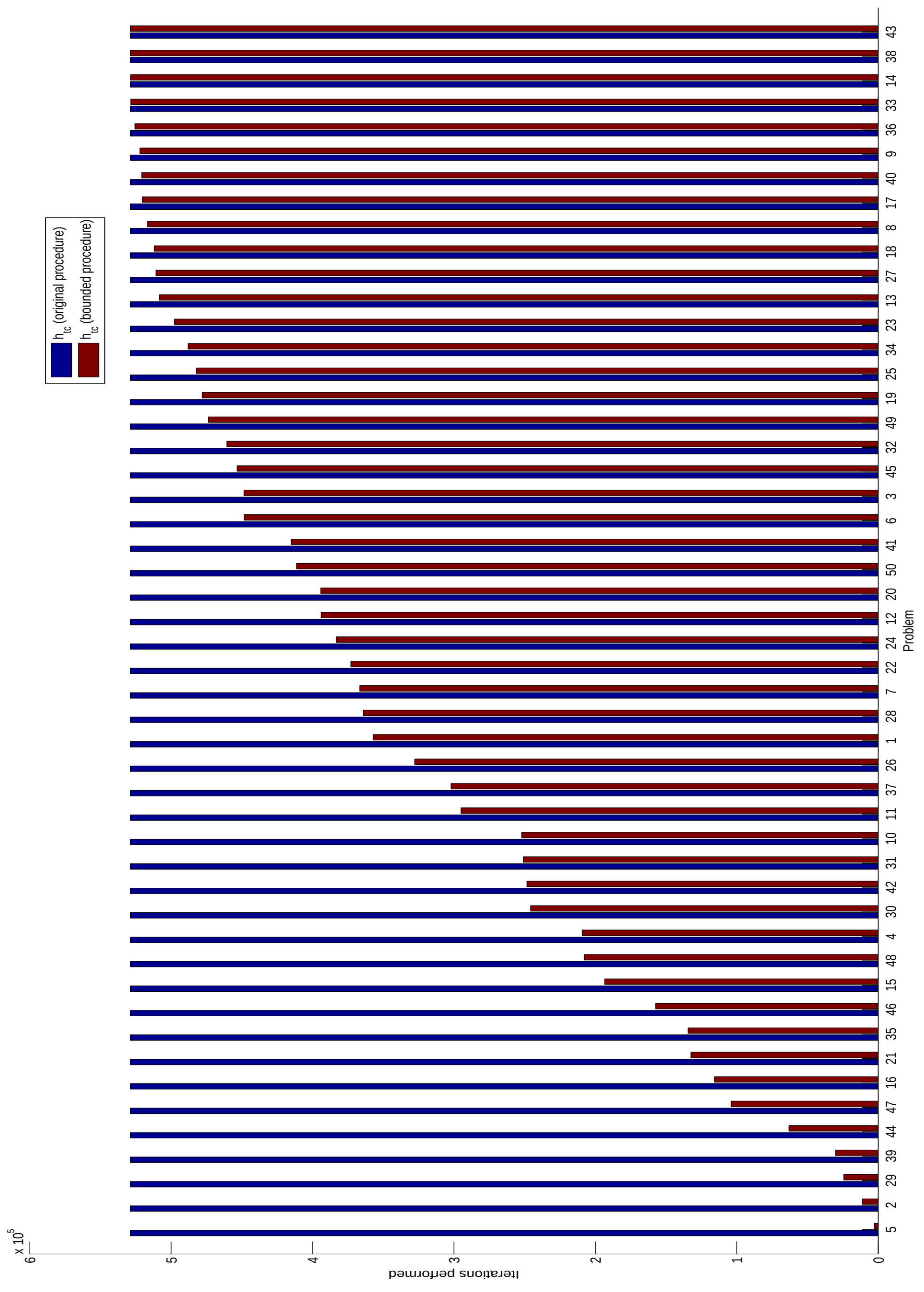


Figure 6.6: Iterations performed by $NAMOA^*$ with the original and bounded methods for the TC heuristic on NY problem instances.

ID #	COL20	COL42	FLA2	FLA14	FLA20	FLA24	FLA26	FLA40	FLA47
$\vec{h}_{TC}$(bounded)	4,026	12,783	12,578	140,212	58,246	78,422	13,531	144,834	19,504

Table 6.9: Run time (in seconds) for problem instances run without time limit with $NAMOA^*$ and bounded $\vec{h}_{TC}$

6.3.5 Difficult problem instances

A set of nine problem instances could not be solved within the time limit regardless of the heuristic function. These problem instances were solved once without time limits with the $\vec{h}_{TC}$ heuristic with bounded calculation, which is identified in the analyses as the most effective alternative. All of them could be solved with the available resources. Table 6.9 shows the time taken to solve each instance. The hardest one, FLA 40, was solved in about 40 hours and 14 minutes.

6.3.6 Summary

This section has reported the successful application of exact multiobjective techniques to search in large size realistic road maps. Road map data were obtained from the 9th DIMACS implementation challenge on shortest paths, as explained in section 3.2.3.

The $NAMOA^*$ multiobjective heuristic search algorithm has been tested with a heuristic precalculated with search ($\vec{h}_{TC}$), a classical distance heuristic ($\vec{h}_{cd}$, corrected great circle distance) and without heuristic information ($\vec{h}_0$, blind search). In addition to the original calculation method for the $\vec{h}_{TC}$ heuristic, a new bounded calculation method is proposed. This method takes advantage of formal properties of the $NAMOA^*$ algorithm to reduce the precalculation effort. All the heuristics and precalculation methods considered were run on a set of 200 random problems defined over four different maps.

Results show that the combination of $NAMOA^*$ with the $\vec{h}_{TC}$ heuristic is a competitive approach in the solution of biobjective search problems. Furthermore, the time devoted to heuristic precalculations has been found to pay off, greatly reducing multiobjective search effort. Additionally, the new bounded calculation method for the $\vec{h}_{TC}$ heuristic has been found to be more efficient in practice than the original method, significantly reducing precalculations in many problem instances.

The $\vec{h}_{TC}$ heuristic with the new bounded calculation method was found to clearly outperform a classical distance heuristic and blind search, both in the number of solved problems and in time performance. The overall speedup of the $\vec{h}_{TC}$ heuristic with bounded calculation in the experiments is 53.56 and 14.49 when compared to blind search and the distance heuristic respectively.

The approach described in this section could solve random problem instances in realistic road maps of up to 1,070,376 nodes and 2,712,798 arcs. Most instances could be solved within a one-hour time limit and the hardest one was solved in over 40 hours, which is reasonable for off-line route planning applications.

The road maps used in this section are not comparable in size to those used for single-objective testbeds, but enough large for multiobjective search to represent difficult problem instances in many cases. However, the high positive correlation presented between time and physical distance (ρ near 1) gives us an analogous scenario to that of the modified road maps shown in section 5.5, i.e. the performance is comparable to the case of class II grids with $\rho = 0.8$ (see table 5.2). More realistic scenarios (e.g. uncorrelated objectives) should be investigated despite its difficulty for multiobjective search. This is analyzed in the next section.

Moreover, different orderings among nondominated open alternatives should be investigated as suggested by Iori et al. (2010) (see section 5.4.2). Besides, TC presents a linear heuristic selection rule described in section 2.4.3. The next section makes a fair comparison on the performance of $NAMOA^*$ (with linear selection) and TC over these realistic maps with uncorrelated objectives.

6.4 Route Planning: Time vs. Economic Cost

This section evaluates the application of multiobjective heuristic search to the road map of the New York City area with 264,346 nodes and 730,100 arcs (see table 3.3). Two objectives are optimized simultaneously: travel time, and economic cost, which includes fuel cost and highway tolls. The travel cost attribute was calculated from available information of DIMACS challenge, as described in section 3.2.3. Unlike the problems analyzed in section 6.3, the objectives are not linearly correlated ($\rho = 0.16$). This is therefore a difficult multiobjective problem. Figures 3.2 and 3.1 display the set of toll highways, and all roads in the map respectively. A set of 20 route planning problems were generated selecting random origin and destination nodes using an uniform distribution.

Regarding the algorithms, two heuristic multiobjective search algorithms are evaluated, $NAMOA^*$ and TC. In order to guarantee that all nondominated solutions are found, multiobjective algorithms need to select at each iteration an open label with a nondominated heuristic evaluation. $NAMOA^*$ accepts any of such schemes, like lexicographic or linear order (see section 2.2.1). TC uses a special linear selection rule that adds the particular precalculated heuristic estimate h_{mix} to the components of each label evaluation vector (see section 2.4.3 for details).

The efficiency of multiobjective search algorithms is known to depend on the particular label selection strategy. A recent analysis on uninformed multiobjective search has suggested that a linear aggregation rule is more effective than a lexicographic one (Iori et al., 2010). To the author's knowledge, no analogous study had been reported on heuristic multiobjective algorithms.

In order to evaluate the impact of a linear evaluation rule, we shall consider two additional selection rules for $NAMOA^*$, in addition to the rules used in previous sections and chapters:

PATH SELECTION in $NAMOA^*$ with heuristic linear selection:
Select for expansion an alternative $(n, \vec{g}_n, \{f\})$ with minimal scalar value f from *OPEN*, i.e. a label such that $\nexists(n', \vec{g}_{n'}, \{f'\}) \in OPEN$ with $f' < f$, namely

$$\sum_i h_i(n') + \sum_i g_i(P_{sn'}) < \sum_i h_i(n) + \sum_i g_i(P_{sn}) \qquad (6.4)$$

This rule is also valid for blind search, assuming that $\vec{h}_0$ is used for all $\vec{g}_x \in G_{op}(x)$, but it could be stated also as,

PATH SELECTION in $NAMOA^*$ with blind linear selection:

Select for expansion an alternative $(n, \vec{g}_n, \{g\})$ with minimal scalar value g from *OPEN*, i.e. a label such that $\nexists (n', \vec{g}_{n'}, \{g'\}) \in OPEN$ with $\sum_i g_i(P_{sn'}) < \sum_i g_i(P_{sn})$

$NAMOA^*$ is evaluated with both a lexicographic and a linear selection rules. The TC algorithm was run using its special linear selection rule, as required in the original description of the algorithm. The G_{op} sets were ordered according to the same particular rule used for selection. Additionally, the precalculated multiobjective heuristic $\vec{h}_{TC}$ proposed by Tung & Chew (1992) is used in both algorithms, and its performance compared to blind search ($\vec{h}_0$). A classical distance heuristic like $\vec{h}_{cd}$ is not considered, as $\vec{h}_{TC}$ has been shown to be significantly better.

The algorithms were implemented in ANSI Common Lisp using LispWorks 6.0 Enterprise 64 bits, and run on a Sun Fire X4140 server with 2 six-core AMD Opteron 2435 @ 2.60GHz processors and 64 Gb of DDR2 RAM, under Windows Server 2008 R2 Enterprise (64-bits). Table 6.10 sums up the features of all the algorithms evaluated in this section.

Algorithm	Selection rule	Additional selection rule	Filtering criteria
Blind NAMOA-LEX	best $\vec{g}$	lex order	$\vec{g}$ dominated
NAMOA-LEX	best $\vec{g} + \vec{h}$	lex order	$\vec{g} + \vec{h}$ dominated
Blind NAMOA-LIN	best $\sum_i g_i$		$\vec{g}$ dominated
NAMOA-LIN	best $\sum_i g_i + \sum_i h_i$		$\vec{g} + \vec{h}$ dominated
TC with heuristic selection	best $h_{mix} + \sum_i g_i$		$\vec{g} + \vec{h}$ dominated

Table 6.10: Algorithms evaluated for road map problem instances with time vs. economic cost (NY_2 map).

6.4.1 Results

Figure 6.7 shows the time requirements for the five algorithmic instances under consideration, i.e. blind and heuristic $NAMOA^*$ with lexicographic selection (*NAMOA-LEX*), blind and heuristic $NAMOA^*$ with linear selection (*NAMOA-LIN*), and TC algorithm. The abscissa axis x shows problem indexes for each instance in the 20 problem set. Instances are ordered according to increasing time taken by *NAMOA-LIN* with heuristic. The value for ordinate y (time in seconds) is shown in a logarithmic scale.

Time values for each algorithmic instance and the heuristic precomputation time, for both $NAMOA^*$ and TC approaches, are presented also in table 6.11. Additional information, like source and destination nodes for each problem instance, along with the number of distinct Pareto-optimal solutions $|C^*|$ for each instance is also presented in this table.

For practical reasons, instances were solved with a twelve-hour time limit. In the case that an algorithm could not solve an instance, values are not displayed in the figure (and as a result in table 6.11) for that particular heuristic-algorithm combination. In addition, values under 1 second obtained by the heuristic algorithms *TC*, *NAMOA-LIN* and *NAMOA-LEX* in problem instances NY$_2$ 15 and NY$_2$ 3 are not displayed for the sake of clarity in figure 6.7.

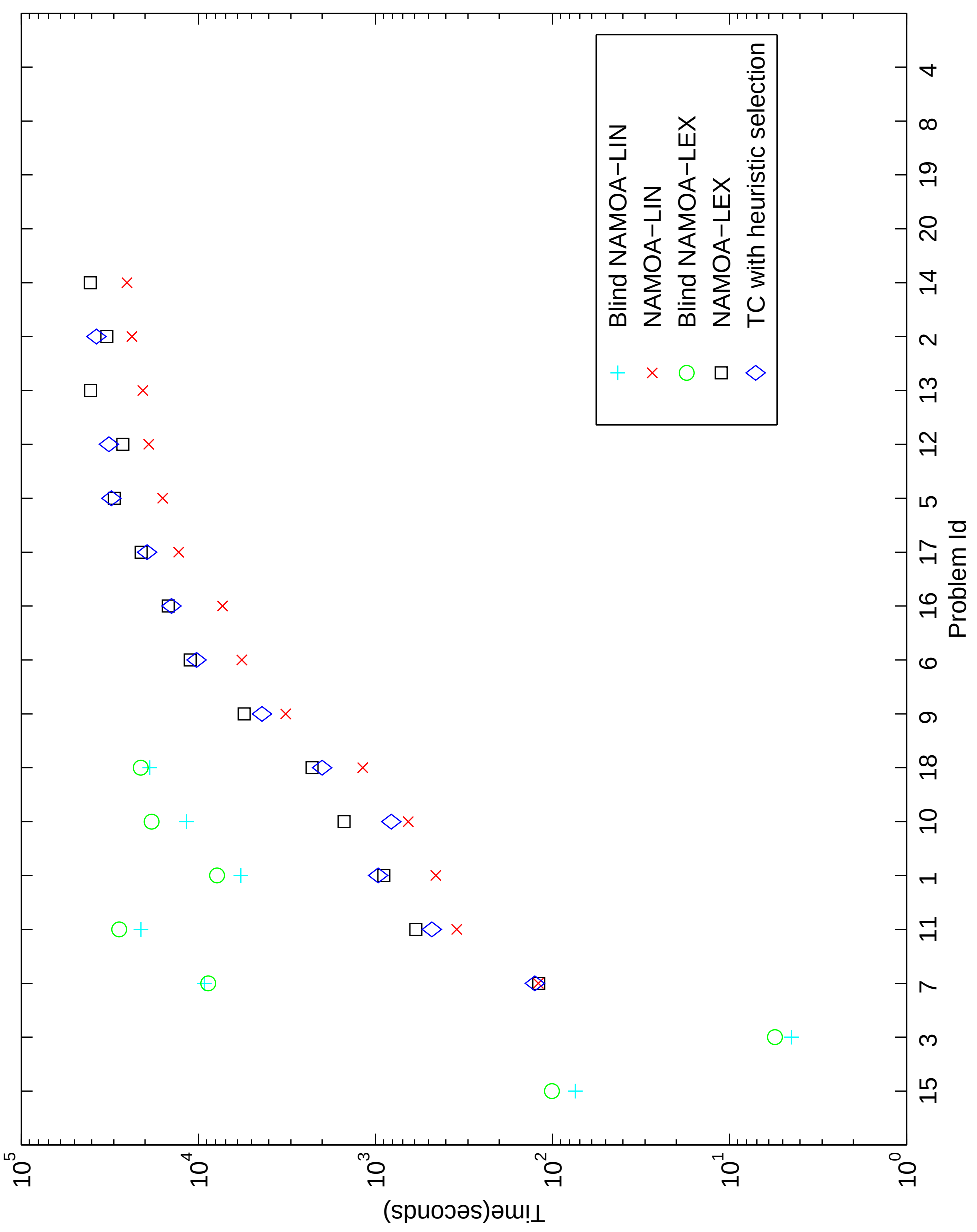


Figure 6.7: Time requirements for $NAMOA^*$ with blind and heuristic search and TC with heuristic selection in NY_2 map

Problem Information				Heuristic algorithms			Blind algorithms		Heu. precomp.	
Id	Source	Goal	|C*|	t_{LIN}	t_{LEX}	t_{TC}	t_{LIN}	t_{LEX}	t_{NAMOA}	t_{TC}
1	256042	202263	1089	455.54	894.03	964.43	5752.88	7827.09	14.52	16.40
2	124324	261253	1469	23733.43	32828.91	37572.79	-	-	14.79	16.43
3	182612	181891	16	0.14	0.16	0.22	4.48	5.54	13.93	17.94
4	121393	176780	5121	-	-	-	-	-	-	-
5	231474	180640	2451	15874.99	29829.76	30969.30	-	-	15.52	17.74
6	114871	76041	1502	5676.02	11097.60	10240.98	-	-	14.60	16.99
7	245379	233193	272	119.97	119.31	125.58	9242.08	8777.86	12.93	16.25
8	175518	82191	7391	-	-	-	-	-	-	-
9	188151	178586	919	3202.78	5511.78	4366.34	-	-	11.31	17.83
10	101251	25656	774	650.82	1500.74	813.86	11668.67	18349.15	14.98	17.94
11	17791	251420	631	347.29	590.54	479.17	21075.36	27970.64	12.54	17.11
12	53355	242716	1573	19029.05	26662.83	31949.90	-	-	12.42	18.11
13	31657	208961	3046	20565.00	40542.49	-	-	-	13.35	18.22
14	162004	68204	2957	25284.55	40720.64	-	-	-	14.18	16.41
15	176111	178359	1	0.02	0.02	0.01	74.22	100.75	14.02	16.80
16	102756	261577	2034	7285.45	14800.00	14170.77	-	-	13.20	18.63
17	128414	202886	1724	12887.40	21050.46	19440.64	-	-	13.45	18.35
18	65899	51303	1276	1177.79	2279.47	2000.18	18809.62	21156.51	14.71	19.27
19	70307	181121	4224	-	-	-	-	-	-	-
20	208578	106079	3262	-	-	-	-	-	-	-

Table 6.11: Data from NY_2 map problems

Algorithm	# of instances
Blind *NAMOA-LEX*	13
NAMOA-LEX	4
Blind *NAMOA-LIN*	13
NAMOA-LIN	4
TC algorithm	6

Table 6.12: Number of instances from NY_2 map not solved in 12h

Heuristic $NAMOA^*$ with a linear selection rule is clearly the best option. It is worth noting that the special selection heuristic used by the TC algorithm does not seem to provide any practical advantage. This confirms our previous analysis which suggested that using additional heuristic information in label selection can in fact degrade time performance (see section 5.4.2).

Table 6.12 shows the number of problem instances that were unsolvable within the time limit for each algorithm. A set of 4 problem instances could not be solved within the time limit regardless of the algorithm. These problem instances were solved without time limit with heuristic *NAMOA-LIN*, which is identified in the analyses as the most effective alternative. All of them could be solved with the available resources. Table 6.13 shows the time taken to solve each of these instances. The hardest one, NY_2 4, was solved in about 127 hours.

Table 6.14 shows in the two columns the average speedup of $NAMOA^*$ with respect to TC algorithm and blind search, for each one of the two strategies used for selection, *LEX* and *LIN*. The results presented in figure 6.7 indicate that the linear strategy is the most suitable one for all cases, with a speedup of 1.55 over TC algorithm, and of 693

Prob. ID	Time (seconds)	Time (hours)
NY_2 20	47,640.3	13.2
NY_2 19	128,984.3	35.8
NY_2 8	421,416.6	117.1
NY_2 4	458,006.3	127.2

Table 6.13: Time requirements for heuristic *NAMOA-LIN* on NY_2 problem instances run without time limit

	NAMOA-LEX	*NAMOA-LIN*
TC	0.9767	1.5579
Blind	926.1803	693.5811

Table 6.14: Average speedup of heuristic $NAMOA^*$ over TC and blind $NAMOA^*$ for NY_2 problem instances solved within runtime limit

over blind $NAMOA^*$ with linear selection. Additionally, with heuristic $NAMOA^*$ the speedup of linear versus lexicographic selection was 1.66. With uninformed $NAMOA^*$, the speedup of linear versus lexicographic was 1.27. Only the problems that could be solved within the runtime limit were taken into account for the average speedups.

6.4.2 Summary

This section presents a realistic analysis of heuristic search algorithms for multiobjective route planning problems. The analysis involves the consideration of two linearly uncorrelated objectives (travel time and travel cost) defined over a relatively large road map, with 264,346 nodes and 730,100 arcs.

The analysis confirms that uninformed search algorithms are in general not practical for this kind of problems. This is a serious objection for approaches that rely on extensive uninformed precalculations.

Regarding heuristic search approaches, two algorithms were analyzed, TC and $NAMOA^*$. The latter was tested with lexicographic and linear selection rules. The analysis confirms that the linear selection rule is more efficient than the lexicographic one. However, the special heuristic linear selection used by TC algorithm was found to perform worse. In fact, this strategy is even worse than heuristic lexicographic $NAMOA^*$ for the hardest problems, confirming the analysis in chapter 5.

While heuristic $NAMOA^*$ with linear selection was able to find the exact solutions to all tested problems with the available resources, current time requirements of this kind of search are only reasonable for off-line route planning applications.

Average precalculation values are not added to time results of $NAMOA^*$ or TC when the heuristic $\vec{h}_{TC}$ was applied. While these times are not significant compared to total execution time, there can be some easy problems where the total execution time of blind search is lower, like NY_2 3. This is not a serious problem as the bounded calculation method proposed in section 6.2.2 can be used to reduce heuristics precomputation time. The heuristics precomputation is shown to pay-off in most cases with dramatic time reductions.

The next section further investigates multiobjective search on hazmat transportation problems. The development of more efficient heuristics is analyzed in chapter 7.

6.5 Hazmat Transportation Problems

The problem of hazardous material (*hazmat*) transportation is currently an active research topic (Erkut et al., 2007). The search for alternative routes that minimize the risk of exposure of the population to hazardous substances can avoid bigger disasters in case of an accident. This involve the consideration of several aspects at the same time, like the transportation time, distance and cost besides risk. Multiobjective analysis (Ehrgott, 2005) becomes then an important tool in hazmat tranportation decision making.

In the literature, the performance of blind search multiobjective techniques has been widely analyzed (Caramia et al., 2010). In this section, multiobjective heuristic search algorithms have been applied to the hazmat tranportation problem. The use of an informed multiobjective heuristic function can significantly improve the efficiency of these problems. In order to evaluate this, test problems with two and three objectives were defined over random graphs and over a real road map. The experiments performed in this section report a substantial improvement over blind multiobjective search.

6.5.1 Hazmat route planning

The majority of the study on hazmat transportation deals with two related subjects: the evaluation of the risk for the population and the environment affected by the hazmat shipments, and the selection of a set of alternative paths to service the hazmat shipments. Erkut et al. (2007) in their survey on hazmat transportation classify routing hazmat shipments into *local* and *global* route planning problems. In the local route planning problems, one is concerned with finding route(s) between a given origin-destination pair for a given hazmat, transport mode, and vehicle type. In the global route planning problem, in general, we have to find a set of paths to route hazmat shipments from distinct origins to different destinations.

There are many papers in the open literature addressing the hazmat local route planning problem (Abkowitz & Cheng, 1988; Kara et al., 2003; Erkut & Verter, 1998; Erkut & Ingolfsson, 2005). Even if hazmat route planning is intrinsically a multi-objective problem only few papers address it by means of multi-objective optimization approaches, e.g. (Cox, 1984; Wijeratne et al., 1993). Recently, Caramia et al. (2010) proposed an algorithm for hazmat shipments that selects k representative paths among the set of efficient paths, with respect to the minimization of length, time (cost) and risk; in particular, the selection is made by choosing paths with high spatial dissimilarity. The approach followed in our analysis is similar to the first phase of Caramia et al. (2010), consisting of finding the whole set of Pareto-optimal paths that minimize distance, time and societal risk.

6.5.2 Results

Multiobjective heuristic search is evaluated in this section on two different sets of problems. The first is a set of problems over random graphs with three objectives used in the work of Caramia et al. (2010), that were described in section 3.2.2. The second is a set of randomly generated pairs of nodes over a real road network with three objectives from the region of Lazio in Italy. This road network was also used by Caramia et al. (2010), and was described in section 3.2.5. The experiments reported in this section evaluate the performance of blind and heuristic $NAMOA^*$ with two or three objectives and lexicographic order. Two objectives are derived as described in sections 3.2.2 and 3.2.5.

Regarding the algorithms, $NAMOA^*$ was run twice for each problem instance: once without heuristic information ($\vec{h}_0$) and the second one using the precalculated $\vec{h}_{TC}$ heuristic. The algorithms were implemented using LispWorks Professional. The random graph problems were run on a Windows 64-bit platform, with an Intel Core2 Quad Q9550 at 2.8Ghz, and 4Gb of RAM, and the simpler Lazio map problems on a Windows 32-bit platform, with an Intel Pentium IV and 256Mb of RAM.

6.5.3 Random graphs

Minimum, maximum and average time in seconds over the ten problems for each set of random graphs with 3 objectives were calculated. These are shown in table 6.15 for blind and heuristic search. The table also reports for each set the minimum, maximum and average cardinality of the set of Pareto-optimal solution costs $|C^*|$.

Problem class		$\lvert C^*\rvert$			Time Blind (Seconds)			Time Heu. (Seconds)		
n	d	Min.	Avg.	Max.	Min.	Avg.	Max.	Min.	Avg.	Max.
100	0.20	4	13.80	24	0.02	0.05	0.09	0.00	0.01	0.02
100	0.50	10	26.90	69	0.14	0.20	0.45	0.01	0.05	0.17
100	0.70	17	34.50	71	0.13	0.36	0.64	0.01	0.11	0.36
200	0.20	13	26.50	51	0.16	0.35	0.75	0.02	0.07	0.17
200	0.50	16	40.20	52	0.58	1.18	1.53	0.17	0.32	0.42
200	0.70	23	46.60	72	0.97	1.85	2.70	0.11	0.57	0.94
300	0.20	21	34.20	58	0.64	0.93	1.73	0.08	0.18	0.42
300	0.50	42	62.60	87	2.75	3.76	4.95	0.83	1.07	1.53
300	0.70	37	71.20	105	4.15	6.14	8.85	1.19	1.92	3.14

Table 6.15: Average results on random graphs with 3 objectives for blind and heuristic search with $NAMOA^*$

Analogously, two-objective search has been evaluated for all pairs of objectives. Results for blind and heuristic search with the combination (1,2) can be found in table 6.16. The combination (1,3) is summarized for blind and heuristic search in table 6.17, while the results for blind and heuristic search with the last combination (2,3) can be found in table 6.18.

Problem class		$\|C^*\|$			Time Blind (Seconds)			Time Heu. (Seconds)		
n	d	Min.	Avg.	Max.	Min.	Avg.	Max.	Min.	Avg.	Max.
100	0.20	2	4.80	7	0.00	0.02	0.03	0.00	0.00	0.02
100	0.50	5	8.60	15	0.03	0.06	0.13	0.00	0.02	0.03
100	0.70	4	9.20	13	0.03	0.09	0.14	0.02	0.03	0.06
200	0.20	3	8.60	16	0.03	0.11	0.17	0.00	0.02	0.05
200	0.50	5	8.70	13	0.16	0.24	0.33	0.01	0.05	0.08
200	0.70	7	10.30	15	0.27	0.35	0.42	0.05	0.08	0.14
300	0.20	6	10.70	15	0.09	0.25	0.41	0.01	0.04	0.09
300	0.50	6	12.30	20	0.44	0.66	1.03	0.06	0.15	0.30
300	0.70	7	12.10	16	0.56	0.90	1.17	0.08	0.19	0.41

Table 6.16: Average results on random graphs with objectives 1,2 for blind and heuristic search with $NAMOA^*$

Problem class		$\|C^*\|$			Time Blind (Seconds)			Time Heu. (Seconds)		
n	d	Min.	Avg.	Max.	Min.	Avg.	Max.	Min.	Avg.	Max.
100	0.20	1	4.80	7	0.00	0.02	0.05	0.00	0.01	0.02
100	0.50	4	8.00	16	0.02	0.05	0.09	0.00	0.01	0.03
100	0.70	5	9.10	13	0.03	0.08	0.13	0.00	0.02	0.05
200	0.20	5	7.80	13	0.03	0.09	0.16	0.00	0.01	0.03
200	0.50	5	9.10	19	0.13	0.22	0.39	0.01	0.04	0.13
200	0.70	7	9.10	13	0.22	0.31	0.45	0.03	0.06	0.09
300	0.20	6	9.90	17	0.09	0.22	0.33	0.01	0.05	0.11
300	0.50	9	11.80	16	0.33	0.62	1.01	0.05	0.14	0.25
300	0.70	7	12.30	16	0.59	0.92	1.25	0.11	0.21	0.44

Table 6.17: Average results on random graphs with objectives 1,3 for blind and heuristic search with $NAMOA^*$

Problem class		$\|C^*\|$			Time Blind (Seconds)			Time Heu. (Seconds)		
n	d	Min.	Avg.	Max.	Min.	Avg.	Max.	Min.	Avg.	Max.
100	0.20	3	5.70	9	0.00	0.02	0.03	0.00	0.01	0.02
100	0.50	3	8.30	12	0.03	0.06	0.08	0.00	0.02	0.03
100	0.70	7	10.40	15	0.06	0.09	0.13	0.00	0.03	0.08
200	0.20	3	7.80	11	0.05	0.09	0.16	0.00	0.02	0.03
200	0.50	6	10.00	17	0.11	0.25	0.47	0.02	0.07	0.14
200	0.70	4	9.80	21	0.11	0.35	0.62	0.01	0.09	0.20
300	0.20	5	10.00	14	0.14	0.23	0.33	0.00	0.04	0.06
300	0.50	5	11.70	17	0.30	0.64	0.98	0.05	0.13	0.30
300	0.70	8	11.70	20	0.61	0.89	1.59	0.08	0.18	0.33

Table 6.18: Average results on random graphs with objectives 2,3 for blind and heuristic search with $NAMOA^*$

6.5.4 Lazio map

Minimum, maximum and average time in seconds over the 50 problems generated for the Lazio map can be found in table 6.19 for blind and heuristic search. The table reports also for each of the combination of objectives (i.e. time, distance & societal risk;

time & distance; time & societal risk; distance & societal risk) the minimum, maximum and average cardinality of the $|C^*|$ set of Pareto-optimal solution costs reported by the algorithm, as done with the random graphs.

Problem class	$\|C^*\|$			Time Blind (Seconds)			Time Heu. (Seconds)		
Obj.	Min.	Avg.	Max.	Min.	Avg.	Max.	Min.	Avg.	Max.
Time, distance & societal risk	1	3.96	20	0.00	0.17	0.77	0.00	0.02	0.17
Time & distance	1	1.06	2	0.00	0.03	0.07	0.00	0.01	0.02
Time & societal risk	1	3.92	18	0.00	0.17	0.73	0.00	0.03	0.15
Distance & societal risk	1	3.36	14	0.00	0.14	0.63	0.00	0.02	0.11

Table 6.19: Average results on Lazio map for blind and heuristic search with $NAMOA^*$

6.5.5 Discussion

The observation of tables 6.15, 6.16, 6.17, 6.18 and 6.19 shows that heuristic estimates led the search more quickly to optimal solutions in all cases presented in this section. In the class of random graphs, $NAMOA^*$ combined with $\vec{h}_{TC}$ heuristic is always several times faster on average than blind search (from 3.95 times in biobjective (1,3) to 4.87 times in (2,3) combination).

The tables also show that time devoted to find a shortest path in a graph increases with the density and the number of nodes of the graph.

Considering random graphs, all the biobjective pairs (1,2), (1,3) and (2,3) (tables 6.16, 6.17, and 6.18) present very similar results. On the other hand, time is greater for three objective problems (table 6.15).

Regarding the Lazio map problems, the analysis shows a different behaviour. As seen in table 6.19 biobjective problems for the pair (time & distance) are easier, while biobjective problems for the pairs (time & societal risk) and (distance & societal risk) are more difficult. Perhaps surprisingly, three-objective problems have no additional difficulty over biobjective problems (time & societal risk) and (distance & societal risk).

The explanation of this phenomenon can be found in the analysis of correlation between costs. In the case of random costs, the correlation is very low for every pair of objectives; therefore, for the same density and size, difficulty is similar for any two objectives, and greater for the set of three objectives.

However, correlation between objectives in Lazio maps depends on the pair considered (table 3.6). Time and distance are highly correlated ($\rho \approx 1$), but societal risk is not linearly correlated with any of them ($\rho \approx 0$). Therefore the number of Pareto-optimal solutions is not affected by considering distance objective if time objective has been considered, and vice-versa.

In the work of Caramia et al. (2010) similar information is shown only for blind search (Martins' algorithm) over the three-objective case in the random graph set. Solution times are faster in the results presented in this section even for blind $NAMOA^*$ search. These differences can be attributed to different pruning and implementation schemes.

6.5.6 Summary

This section has presented an analysis of blind and heuristic search for multiobjective hazmat transportation problems. The analysis involves the consideration of two and three objectives over two classes of problems: random graphs and hazmat transportation problems defined over a real map from the Lazio region in Italy (in this case, the objectives involved are distance, travel time and societal risk).

Concerning the use of heuristics, heuristic estimates allowed faster searches for all cases presented. As expected, problem difficulty increases with graph size and node density. But these results also show the importance of correlation between objectives in the cases considered. The number of Pareto-optimal paths falls as the correlation between objectives increases. Thus, the time needed to solve a multiobjective problem depends on the specific nature of arc costs. In problems with relatively uncorrelated objectives the number of Pareto-optimal paths increases with the number of objectives under consideration, while the addition of highly correlated objectives does not degrade the performance of the search.

6.6 Conclusions

A number of techniques have proven very effective in real-time single-objective route planning applications, like search in car navigation systems. The extension of these techniques to multiobjective settings is an important area of research. However, the results presented in this chapter suggest that techniques based in extensive blind multiobjective precalculations are likely to experience practical difficulties in large maps, as already noted by Delling & Wagner (2009).

From the systematic evaluation of several parameters performed in this chapter, some conclusions about multiobjective heuristic search algorithms can be drawn,

- The TC algorithm is shown to be consistently worse in time performance under similar conditions to $NAMOA^*$ for realistic problems, confirming the results over grid problems. Besides, it is important to recall that algorithm TC may not remove some dominated members from $OPEN$ and introduce some dominated members in COSTS (see section 2.4.3). Therefore, $NAMOA^*$ emerges as the state-of-art multiobjective heuristic search algorithm.

- This chapter has reported the successful application of $NAMOA^*$ to random problem instances in large sized realistic road maps. In addition, difficult multiobjective problems over realistic road maps with uncorrelated objectives have been also considered. However, time requirements are only reasonable for off-line route planning applications.

- An improved method of calculation for the $\vec{h}_{TC}$ heuristic has been presented in this chapter. This method takes advantage of formal properties of the $NAMOA^*$ algorithm to reduce the precalculation effort, significantly reducing precalculations in many problem instances. The combination of $NAMOA^*$ and $\vec{h}_{TC}$ heuristic with the new bounded calculation method was found to clearly outperform a classical distance heuristic and blind search, both in the number of solved problems and in time performance.

- Linear ordering in $NAMOA^*$ seems to combine heuristic search and delayed expansion of solutions, leading to a more efficient algorithm. Only the linear selection rule will be considered for $NAMOA^*$ in the next chapter.

- The impact of the $\vec{h}_{TC}$ heuristic in other practical problem domains has been also investigated. The application of the $\vec{h}_{TC}$ heuristic to hazardous material transportation problems has outperformed previous blind approaches.

- Finally, the impact of the number of objectives in the performance of the algorithm is evaluated with a comparison between $NAMOA^*$ with two and three objectives. The number of Pareto-optimal paths falls as the correlation between objectives increases. Thus, the time needed to solve a multiobjective problem depends on the specific nature of arc costs. In problems with relatively uncorrelated objectives the number of Pareto-optimal paths increases with the number of objectives under consideration, while the addition of highly correlated objectives does not degrade the performance of the search.

The use of precalculated heuristics in multiobjective search is a rather unexplored area that deserves formal and practical developments. The applications described in this chapter could clearly benefit from more informed precalculated heuristics. Chapter 7 explores this possibility, with more efficient heuristics for multiobjective search.

Chapter 7

Multivalued Heuristics for Multiobjective Heuristic Search

The results presented in previous chapters reveal that $NAMOA^*$ is the algorithm with better formal properties and better empirical behaviour in general. The experimental analyses also confirms that a linear selection rule is better than a lexicographic one.

The multiobjective heuristic function $\vec{h}_{TC}$ has been shown to be very effective for all domains tested in this thesis, reducing time as well as space requirements in most cases. Furthermore, a new improved calculation method has been presented in the previous chapter. In realistic scenarios, the use of this heuristic in $NAMOA^*$ achieved significant improvements in performance over a classical distance heuristic and was also found to outperform blind search in difficult instances. However, even with an informed heuristic like $\vec{h}_{TC}$, the multiobjective search stage can take a long time.

Moreover, the full potential of multiobjective heuristic search has not been completely explored, since $\vec{h}_{TC}$ provides only a single heuristic vector for each node. The heuristic algorithm $NAMOA^*$ is not limited to single-valued heuristics. To the author's knowledge, using $H(n)$ sets with *multiple heuristic vectors* in multiobjective search algorithms has not been evaluated to date.

This chapter investigates more informed heuristics. From a formal point of view, a more informed heuristic will reduce the number of explored alternatives. This chapter develops a precalculation method for more precise heuristic evaluations, and analyzes the cost-effectiveness of precalculations in terms of space and time. The informedness of $H(n)$ reflects the amount and quality of available heuristic information.

Section 7.1 gives an informal example of how to improve the $\vec{h}_{TC}$ heuristic. A precalculation method for an improved heuristic function is proposed in section 7.2. The generation process is shown over a small sample graph. The properties of this improved heuristic function are also analyzed. This section also shows that these improved estimates can be inconsistent in general. The results of the application of this new heuristic to $NAMOA^{**}$ with linear selection can be found in the section 7.3. The analysis shows that the new heuristic achieves dramatic reductions in the number of alternatives considered and as a result in the number of cost vectors stored in the search graph, improving space requirements. Nevertheless, time performance is shown to be improved only under certain circumstances, which are analyzed in section 7.4.

Some conclusions of this study are drawn at the end of the chapter, suggesting future improvements.

7.1 Improving the $\vec{h}_{TC}$ heuristic

A common framework to obtain a heuristic function for a particular problem is the relaxation of some constraints related to that problem (Pearl, 1984; Felner et al., 2011). The accurate heuristic proposed by Tung & Chew (1992) is based on precalculations. These reduce the problem to a set of simpler single objective searches. Let us recall that $\vec{h}_{TC}$ obtains for each node lower bounds c_i^* for each objective i with single-objective Dijkstra's searches from γ to all other nodes in a reversed graph (see section 2.4.4). In the biobjective case, two values are obtained for each node, representing the two lexicographic optimal solutions priorizing first objective (c_1^*, c_2') and second objective (c_1', c_2^*) respectively (points B and C in the figure 7.1). The bounded procedure presented in chapter 6 lets us discard the calculation of some values for those nodes that will surely not be examined in the multiobjective search stage, while providing the same heuristic information for the others.

For example, vector (c_1^*, c_2^*) for some node n (point A in figure 7.1) is derived in $\vec{h}_{TC}$ as lower bound for the cost of all nondominated paths reaching γ from n. More details about $\vec{h}_{TC}$ can be found in sections 2.4.4 and 4.8. Let us assume an additional Pareto-optimal solution cost is known. An heuristic function $H(n)$ more informed than $H_{TC}(n) = \{\vec{h}_{TC}(n)\}$ can be obtained.

Given that a new optimal cost is available, e.g. point D with cost (c_1'', c_2''), the same strategy followed by Tung & Chew can be applied to points B,D and to points D,C. Lower bounds on each objective are used again to build an optimistic heuristic value. Therefore, two new heuristic vectors (c_1^*, c_2'') and (c_1'', c_2^*) can be derived (points E,F). These two heuristic vector values are also optimistic and can replace the $\vec{h}_{TC}$ estimate (point A), as they are more informed. Given additional optimal solution costs, heuristic estimates could be further improved. The next section presents a procedure that systematically applies this idea.

7.2 $KDLS$, a new precalculation method for multiobjective heuristics

In this section, we generalize the procedure informally described in the previous section to obtain new heuristic estimates from known Pareto-optimal costs. The procedure obtains these costs through one-to-all [1] single-objective searches on a reversed graph, where the start node is γ. For each single objective search, the original costs of arcs are replaced by some linear weighted combination. Then Dijkstra's algorithm is run on the new graph.

In discrete problems, only supported solutions can be obtained optimizing a linear weighted combination of costs, regardless of the slope used (see section 2.2.1). Figure 7.2 reflects this situation. A different scheme (e.g. a multiobjective stage) must be

[1] The nodes visited by this procedure can be bounded as explained in Chapter 6.

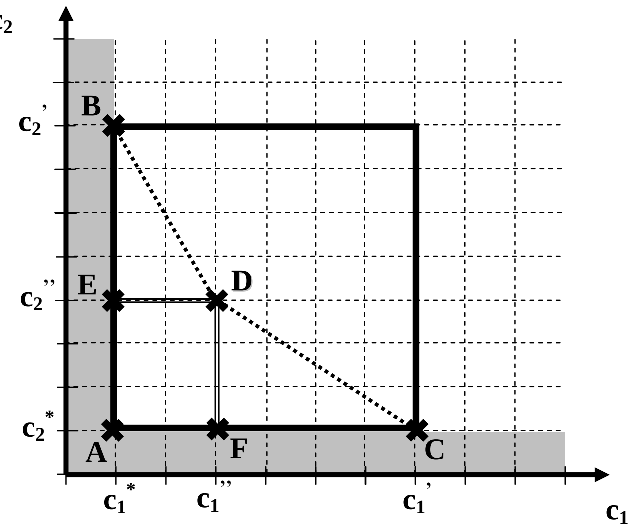


Figure 7.1: Improving $\vec{h}_{TC}$ with two more informed estimates for some node n

used to obtain solutions lying inside the nondominated triangles formed by adjacent supported solutions. Thus, there is a limitation in the number of Pareto-optimal costs and hence in the number of heuristic values that can be obtained for each particular problem with this procedure.

A dichotomic search procedure has been devised to obtain these new heuristic estimates. This is similar to the systematic dichotomic search method described for the first phase in the blind two-phase multiobjective search method suggested by Raith (2009, pp. 33-39), that eventually obtains all supported solutions to the multiobjective problem. In our case, the procedure is used to obtain heuristic values, and hence it is not restricted to one-to-one searches. The procedure is called $KDLS$, standing for "k-dichotomic linear searches", as k levels of splitting processes are performed. Let us call H^k_{KDLS} the heuristic functions obtained by this procedure with parameter k.

Definition 7.1 *Let us denote by r_n the number of supported solutions to the biobjective problem of finding a shortest path from node γ to node n in a reversed graph. Let us assume solutions are ordered lexicographically, priorizing the first objective. The cost of the i-th supported solution to this problem is denoted*[2] *by $\vec{q}^{\,i}(n)$, where $\vec{q}^{\,i}(n) \prec_{LEX} \vec{q}^{\,j}(n), \forall i < j$. Let us denote $\vec{q}^{\,i}(n) = ({q_1}^i(n), {q_2}^i(n))$, or where there is no ambiguity regarding node n as $\vec{q}^{\,i} = ({q_1}^i, {q_2}^i)$*

This means that we have $\vec{q}^{\,1}(n), \vec{q}^{\,2}(n), \ldots, \vec{q}^{\,r_n}(n)$ supported Pareto-optimal solution costs for each node n. They are stored for each node n by $KDLS$ procedure in sets $SUPS(n)$. The extreme solution costs $\vec{q}^{\,1}(n)$, $\vec{q}^{\,r_n}(n)$ are those calculated by the precalculation procedure of Tung & Chew (1992) and previously denoted by

[2]Let us recall that $\vec{q}_i(n, n')$ was defined by Tung & Chew (1992) as the minimal cost for objective i among paths between n and n', see definition 4.13. In the terminology used in this chapter, we will omit the second parameter as all Pareto-optimal costs from each node are referred to goal node γ.

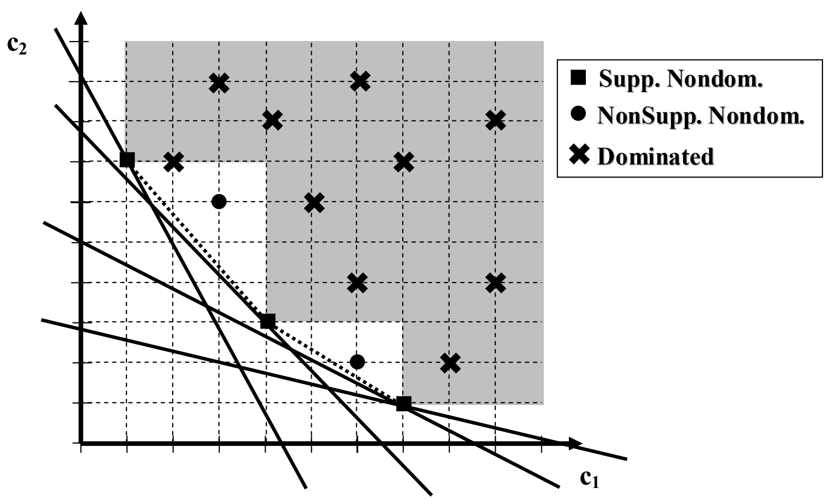


Figure 7.2: Non-supported solutions can not be found solving weighted sum problems

$(c_1^*(n), c_2'(n))$, and $(c_1'(n), c_2^*(n))$ in chapter 6. Thus, $\vec{h}_{TC}(n) = (q_1^1(n), q_2^{r_n}(n))$, as it can be seen in figure 7.3(a). These two extreme supported solutions are the initial members of sets $SUPS$ before running the dichotomic procedure, i.e. initially $SUPS(n) = \{q_1^1(n), q_2^{r_n}(n)\}$.

Definition 7.2 *Let $\vec{c}_1(n, n')$ and $\vec{c}_2(n, n')$ be the two cost functions under consideration. Let $\vec{q}^{\,i}(s)$ and $\vec{q}^{\,j}(s)$ be two known supported solution costs of paths from γ to s, such that $\vec{q}^{\,i}(s) \prec_{LEX} \vec{q}^{\,j}(s)$, and no intermediate solution cost is known.*

Let us define G^{ij} as a graph with the same nodes as G, but all arcs reversed. For all arcs (n, n') of the original graph G, the vector costs $\vec{c}(n, n') = (c_1(n, n'), c_2(n, n'))$ are replaced by a scalar linear combination of both objectives $c^{\,ij}(n', n) = \lambda_1^{ij} c_1(n, n') + \lambda_2^{ij} c_2(n, n')$.

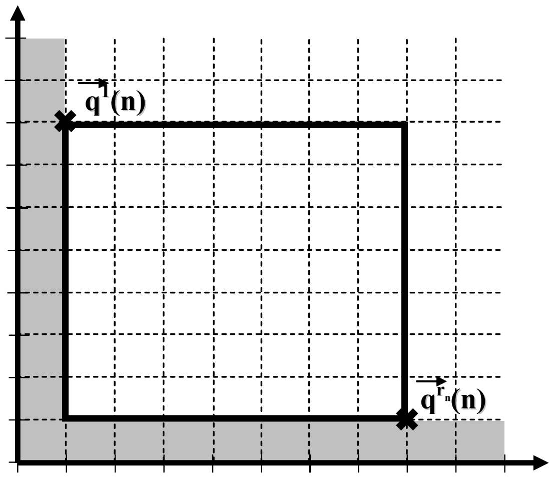
The weights $\vec{\lambda}_{ij} = (\lambda_1^{ij}, \lambda_2^{ij})$ used by KDLS for searching in interval $[\vec{q}^{\,i}(s), \vec{q}^{\,j}(s)]$ are given by the linear combinations

$$\begin{aligned} \lambda_1^{ij} &= q_2^i - q_2^j \\ \lambda_2^{ij} &= q_1^j - q_1^i \end{aligned} \tag{7.1}$$

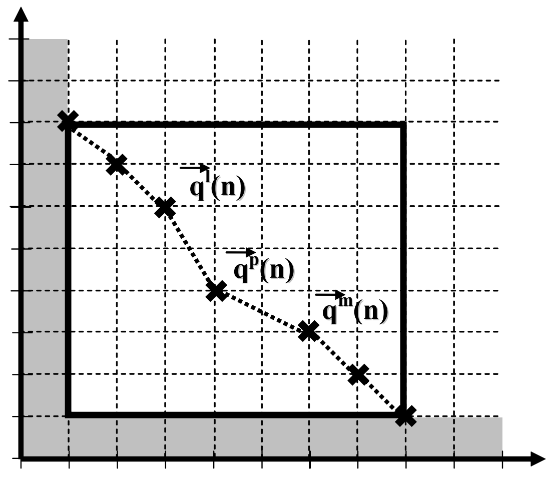
These weights are devised to obtain the supported solution cost with maximal distance to the line joining $\vec{q}^{\,i}(s)$ and $\vec{q}^{\,j}(s)$ (Raith, 2009, pp. 33-39), i.e. the slope is chosen in order to obtain the nearest (supported) point from the Pareto front of node s that can be obtained with linear combinations, as shown in figure 7.4.

The procedure performs on each general step of a dichotomic level k a (bounded) Dijkstra search from γ to s in the reversed graph G^{ij}, trying to find a new supported solution in an interval $[\vec{q}^{\,i}(s), \vec{q}^{\,j}(s)]$, with $i < j$. This also gives us minimal distances $\vec{q}^{\,1}(n), \vec{q}^{\,2}(n), \ldots, \vec{q}^{\,r_n}(n)$ to all interesting nodes, which can be used to construct more informed heuristic values for every node.

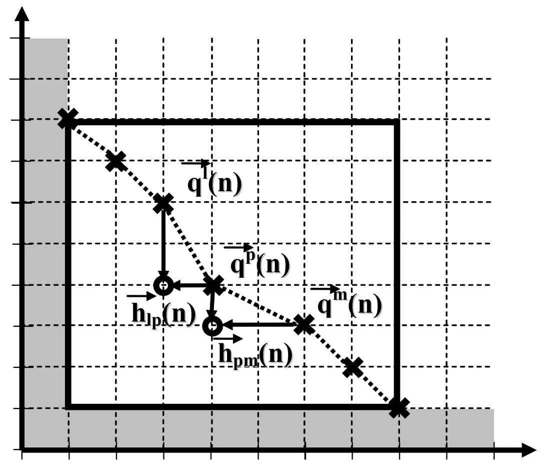
Let n be a node selected for expansion and $g^{ij}(n) = \lambda_1^{ij} g_1(n) + \lambda_2^{ij} g_2(n)$ be the scalar cost of a path $P_{\gamma n}$ when running Dijkstra's algorithm on graph G^{ij}, representing a



(a) Initially $SUPS(n) = \{\vec{q}^{\,1}(n), \vec{q}^{\,r_n}(n)\}$



(b) A new solution cost $\vec{q}^{\,p}(n)$ is added to $SUPS(n) = \{\ldots, \vec{q}^{\,l}(n), \Downarrow, \vec{q}^{\,m}(n), \ldots\}$



(c) Two new heuristic values are obtained

Figure 7.3: Visual description of KDLS procedure

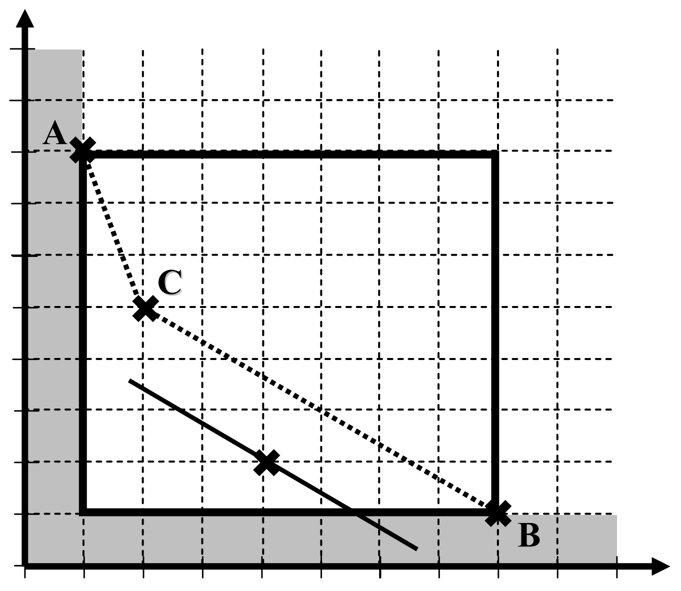


Figure 7.4: A new solution is found in interval [C,B] while no more supported solutions are found in interval [A,C]

Pareto-optimal supported solution reaching n, with vectorial cost $\vec{g}(n) = (g_1(n), g_2(n))$ (see figure 7.3(b)).

Let us denote by $\vec{q}^{\,p}(n) = \vec{g}(n)$ the cost of this new nondominated solution cost to n, and let $\vec{q}^{\,l}(n)$, $\vec{q}^{\,m}(n)$ be the costs immediately before and after in lexicographical order in $SUPS(n)$.

When this cost $\vec{q}^{\,p}(n)$ is new, it can be added to $SUPS(n)$, as shown in figure 7.3(b). Let us consider now the analogous situation for node s. Let $\vec{q}^{\,h}(s)$ be the new solution cost. When $\vec{q}^{\,h}(s) = \vec{q}^{\,i}(s)$ or $\vec{q}^{\,h}(s) = \vec{q}^{\,j}(s)$, no new supported solutions can be found the interval $[\vec{q}^{\,i}(s), \vec{q}^{\,j}(s)]$. Otherwise, it is possible to further continue the search for new supported solutions in the intervals $[\vec{q}^{\,i}(s), \vec{q}^{\,h}(s)]$ and $[\vec{q}^{\,h}(s), \vec{q}^{\,j}(s)]$ (i.e. with a higher dichotomic level $k+1$). Initially, the level $k = 1$ corresponds to the search in the initial interval $[\vec{q}^{\,1}(s), \vec{q}^{\,r_s}(s)]$.

The heuristic proposed by Tung & Chew (1992) is only a particular case, where in the biobjective case $\lambda_1^{ij} = 1$ and $\lambda_2^{ij} = 0$ (and viceversa) are used to optimize the first objective (or respectively the second). The precalculation terminates as soon as these extreme supported solutions are found. Thus, level $k = 0$ denotes the initial search for the two extreme solutions $\vec{q}^{\,1}(n)$, $\vec{q}^{\,r_n}(n)$ in $KDLS$ (i.e. the usual construction for the $\vec{h}_{TC}$ heuristic, with no dichotomic linear searches).

$KDLS$ terminates as soon as all searches up to level k have been completed or no more intervals remain to be examined. Once all relevant intervals have been examined, the sets $SUPS(n)$ are processed to obtain the heuristic functions $H^k_{KDLS}(n)$ for each node (see figure 7.3(c)).

The detailed procedure for two objectives is shown in table 7.1. As for the bounded procedure described in section 6.2.2, some improvements over the calculation method are proposed:

- INITIALIZATION:

  — Create $I = \{\}$

  — Perform two reverse Dijsktra's searches from γ to determine the subset of nodes that will be considered ($SN \subseteq N$), as well as extreme solutions to them $\vec{q}^{\,1}(n)$ and $\vec{q}^{\,r_n}(n)$, $\forall n \in SN$

  — Create $\forall n \in SN,\ SUPS(n) = \{\vec{q}^{\,1}(n), \vec{q}^{\,r_n}(n)\},\ H^k_{KDLS}(n) = \{\}$

  — Invoke KDLS with initial interval $I = \{[\vec{q}^{\,1}(s), \vec{q}^{\,r_s}(s)]\}$ and parameter k

- $KDLS(I, k)$

  1. CHECK TERMINATION:
     — IF $I = \{\}$ goto HEURISTIC CONSTRUCTION
  2. INTERVAL SELECTION:
     — SELECT an interval $[\vec{q}^{\,i}(s), \vec{q}^{\,j}(s)]$ from I with level k_{ij}
  3. CALCULATE WEIGHTS $\vec{\lambda}_{ij} = (\lambda_1^{ij}, \lambda_2^{ij})$ as
     $$\begin{aligned} \lambda_1^{ij} &= q_2^i - q_2^j \\ \lambda_2^{ij} &= q_1^j - q_1^i \end{aligned} \tag{7.2}$$
  4. CREATE REVERSED GRAPH G^{ij}: arcs (n, n') are reversed and original costs $\vec{c}(n, n') = (c_1(n, n'), c_2(n, n'))$ are replaced by
     $$c^{\,ij}(n', n) = \lambda_1^{ij} c_1(n, n') + \lambda_2^{ij} c_2(n, n') \tag{7.3}$$
  5. RUN BOUNDED DIJKSTRA search on G^{ij} from γ to all nodes in SN
  6. STORING NEW SUPPORTED SOLUTIONS:
     — $\forall n \in SN$, IF a new supported solution $\vec{q}^{\,p}(n) = (q_1^p, q_2^p)$ is found, STORE $\vec{q}^{\,p}(n)$ in $SUPS(n)$
     — IF (a new supported solution $\vec{q}^{\,h}(s)$ is found for s) and ($k_{ij} < k \vee k = \infty$), THEN STORE in set I two new subintervals $[\vec{q}^{\,i}(s), \vec{q}^{\,h}(s)]$, $[\vec{q}^{\,h}(s), \vec{q}^{\,j}(s)]$ with level $k_{ij} + 1$
     — Go to step 1

- HEURISTIC CONSTRUCTION:

  — $\forall n \in SN$, process each two consecutive supported solutions $\vec{q}^{\,i}(n) = ({q_1}^i(n), {q_2}^i(n))$, and $\vec{q}^{\,j}(n) = ({q_1}^j(n), {q_2}^j(n))$ in $SUPS(n)$ to obtain a new candidate estimate $\vec{h}_{ij}(n) = (h_1^{ij}(n), h_2^{ij}(n))$ as follows,
  $$h_1^{ij}(n) = q_1^i \qquad h_2^{ij}(n) = q_2^j \tag{7.4}$$

  — IF $\vec{h}_{ij}(n) \notin H^k_{KDLS}(n)$ THEN store it in $H^k_{KDLS}(n)$

  — Return the sets H^k_{KDLS}

Table 7.1: Precalculation of H_{KDLS} for bicriteria search by $KDLS$ procedure.

1. Labels in Dijkstra's searches can be discarded in step 5 if any of the following holds,

$$\begin{aligned} \lambda_1^{ij} g_1(n) &> q_1^j \\ \lambda_2^{ij} g_2(n) &> q_2^i \end{aligned} \tag{7.5}$$

   This result is a direct consequence from the formal properties of $NAMOA^*$, which have been previously presented, as it will surely not expand labels that are not C^*-bounded.

2. Step 5 can be improved as suggested by Raith (2009, pp.29-30). An upper bound based in the nadir point (q_1^j, q_2^i) can be used, i.e. we can discard labels with,

$$\lambda_1^{ij} g_1(n) + \lambda_2^{ij} g_2(n) > \ q_1^j + q_2^i \tag{7.6}$$

7.2.1 Example

Let us consider the sample graph of figure 7.5(a). The first step is to reverse all arcs for the initialization phase (figure 7.5(b)). The two extreme solutions between γ and all other nodes are obtained. For example, regarding node s, extreme solutions are given by path $\langle \gamma, 2, s \rangle$ with cost $\vec{q}^{\,1}(s) = (2, 10)$ (see figure 7.6(a)), and path $\langle \gamma, 3, s \rangle$ with cost $\vec{q}^{\,r_s}(s) = (10, 2)$ (see figure 7.6(b)). Numbers above nodes represent the $g(n)$ value obtained by Dijkstra's algorithm.

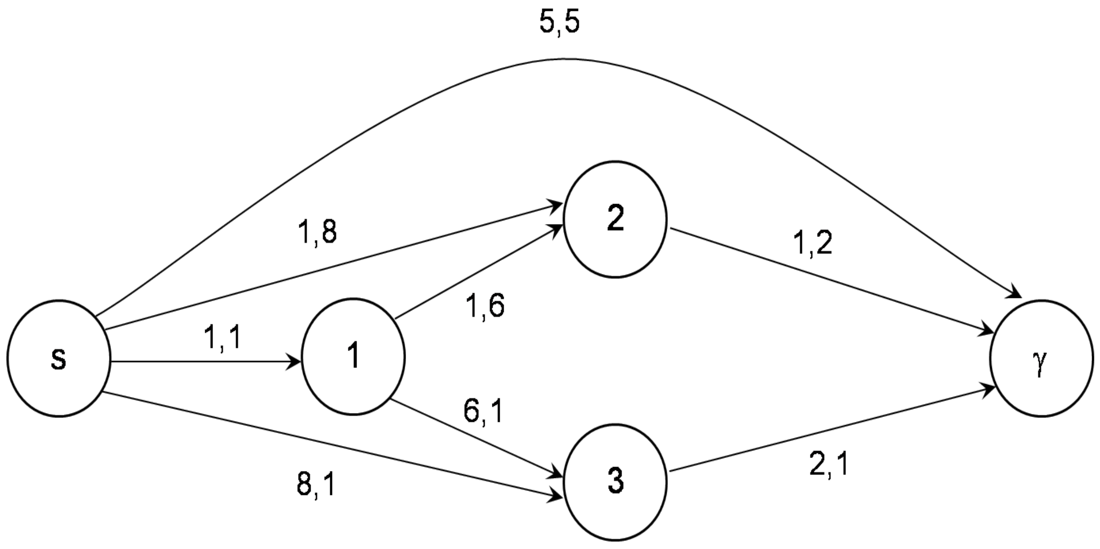
Figure 7.8 displays the set of nondominated solutions from γ to s. The extreme solution costs are $\vec{q}^{\,1}(s)$ and $\vec{q}^{\,3}(s)$. In this case, $r_s = 3$, there is only an additional supported solution from γ to s, the path $\langle \gamma, s \rangle$, with cost $\vec{q}^{\,2}(s) = (5, 5)$. There are two additional nondominated paths, $\langle \gamma, 2, 1, s \rangle$ and $\langle \gamma, 3, 1, s \rangle$, with costs (3,9) and (9,3), paths but they are not supported points.

Procedure $KDLS$ is called with $I = \{[(2, 10), (10, 2)]\}$. Let us assume also that $k = \infty$. The weights $\vec{\lambda}_{13}$ for the first linear search can be calculated as

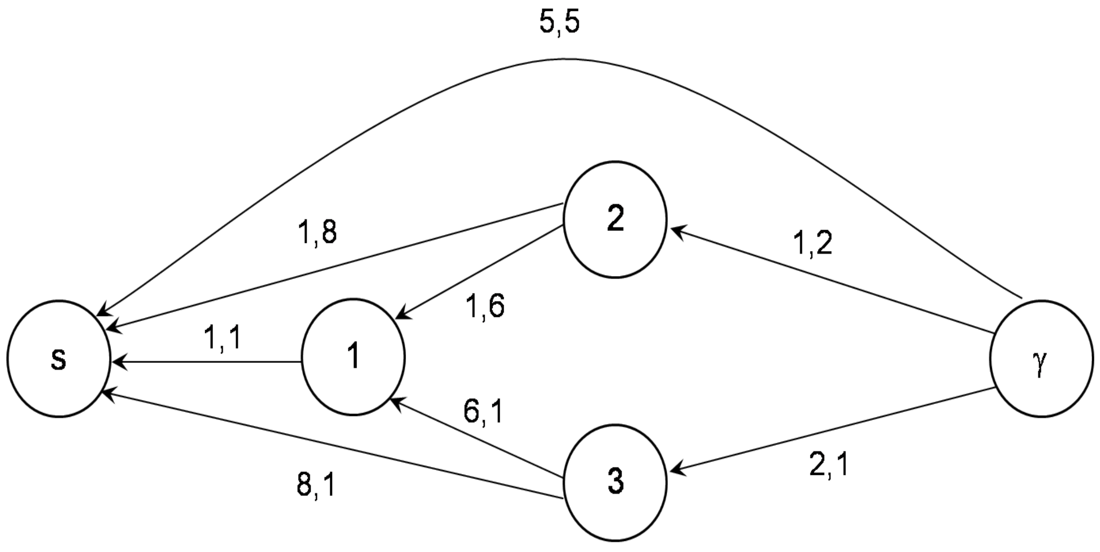
$$\begin{aligned} \lambda_1^{13} &= q_2^1 - q_2^3 = 10 - 2 = 8 \\ \lambda_2^{13} &= q_1^3 - q_1^1 = 10 - 2 = 8 \end{aligned} \tag{7.7}$$

The original graph is transformed into the graph of figure 7.7(a). The result of applying Dijkstra's algorithm on this graph G^{13} is shown in figure 7.7(b). Let us assume that we solve ties on scalar values with lexicographic order on the original costs of arcs. The path $\langle \gamma, s \rangle$, gives an additional supported solution cost in the interval $I_{13} = [(2, 10), (10, 2)]$. The cost of this path $\vec{q}^{\,2}(s) = (5, 5)$ is the unique additional supported solution cost in the graph, as its linear weighted combination value (80) is lower than the cost of any other sum of costs of arcs.

The interval I_{13} is then split, i.e. $I = \{[(2, 10), (5, 5)], [(5, 5), (10, 2)]\}$. Two additional linear searches in intervals $I_{12} = [(2, 10), (5, 5)]$, $I_{23} = [(5, 5), (10, 2)]$ must be performed, but none of them provide new additional supported points, as shown in figure 7.8 and table 7.2. The corresponding figures of Dijkstra's execution are omitted. The last phase of $KDLS$ calculates heuristic values H_{KDLS}^{∞} from sets $SUPS(n)$, which are shown in table 7.2. New supported points are shown in bold face.

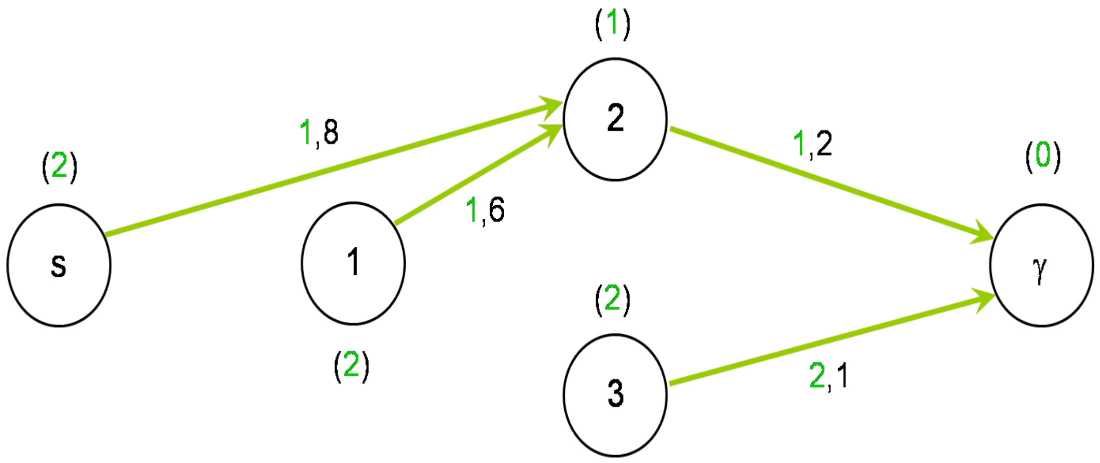


(a) Sample biobjective graph

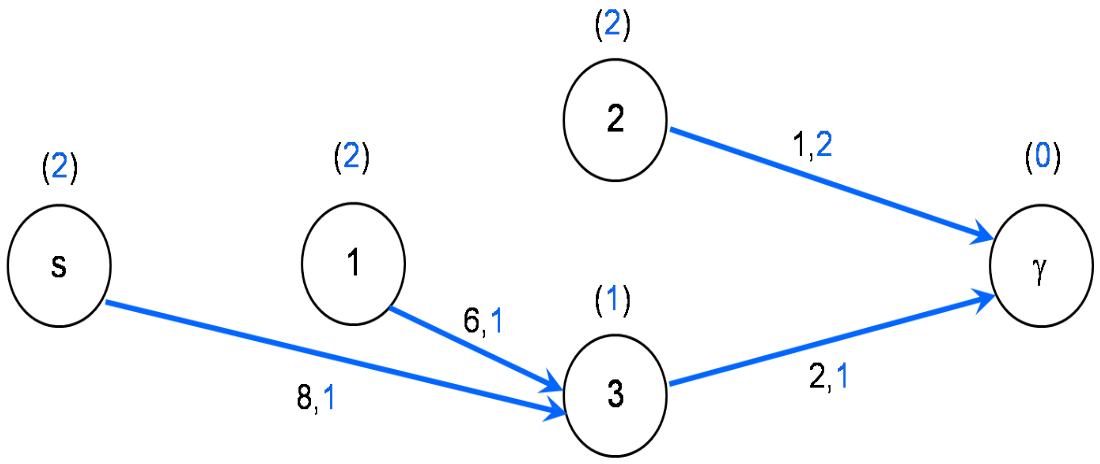


(b) Sample graph with reversed arcs

Figure 7.5: Sample graph for $KDLS$ procedure: initialization

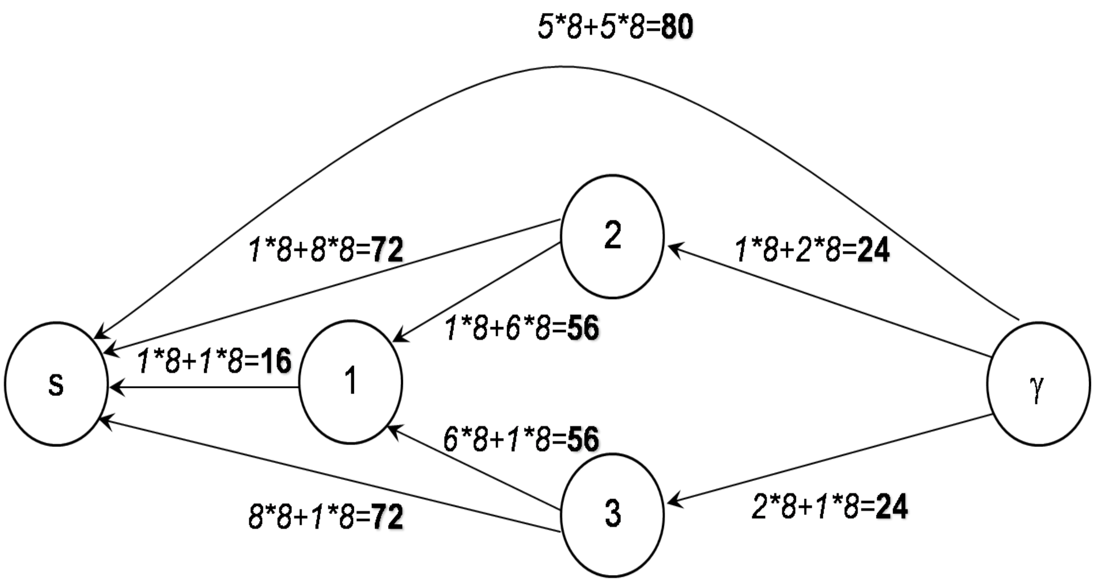


(a) Search tree for the first extreme solution

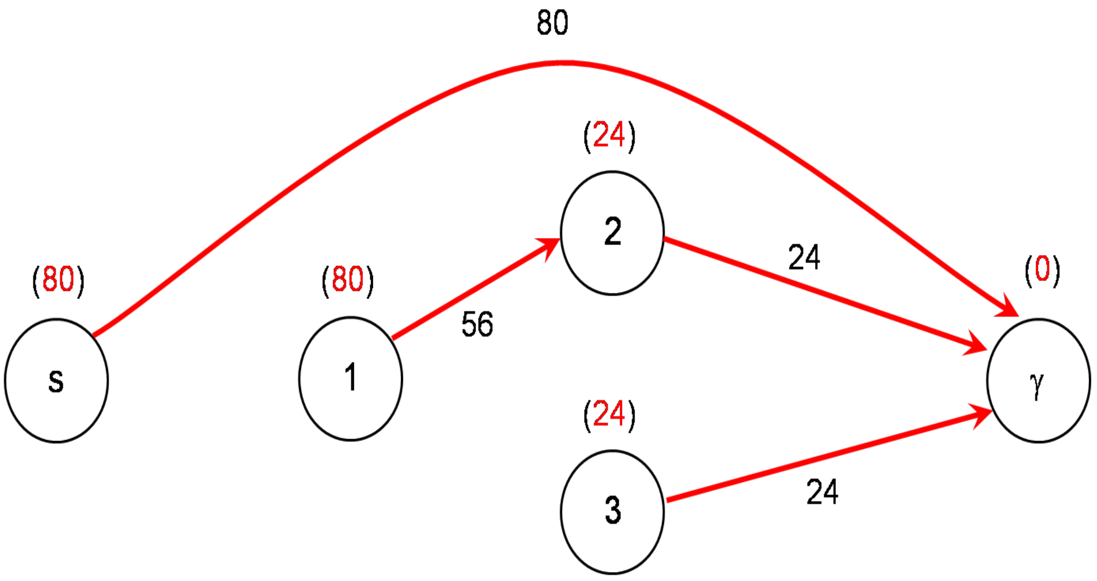


(b) Search tree for the second extreme solution

Figure 7.6: Sample graph for $KDLS$ procedure: searching for extreme solution costs



(a) Reversed graph G_{13}



(b) Search tree for G^{13}

Figure 7.7: Sample graph for *KDLS* procedure: searching for supported solution costs

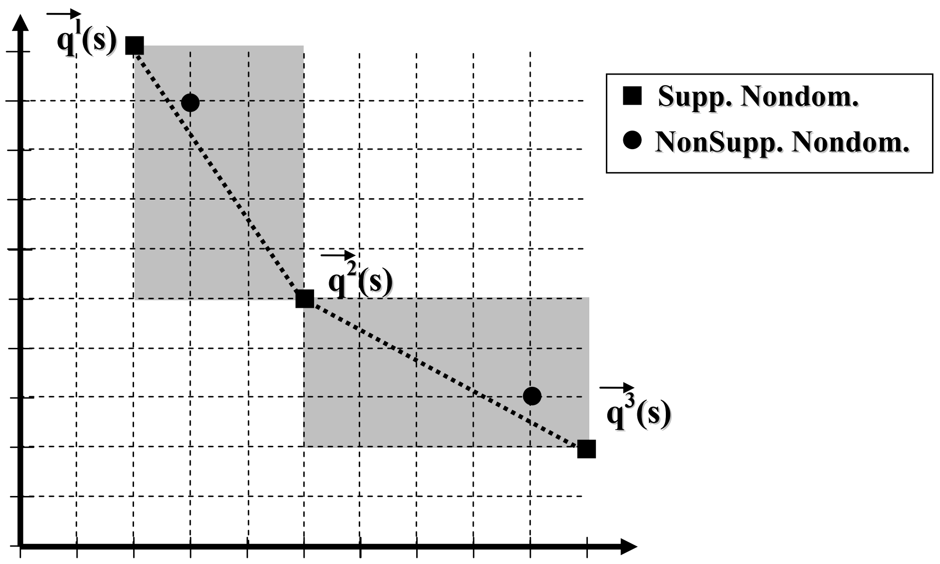


Figure 7.8: Sample graph for $KDLS$ procedure: supported and non-supported solution costs

	Node 3	Node 2	Node 1	Node s
$SUPS(n)$ search lex. 1	(2,1)	(1,2)	(2,8)	(2,10))
$SUPS(n)$ search lex. 2	(2,1)	(1,2)	(2,8),**(8,2)**	(2,10),**(10,2)**
$SUPS(n)$ search lin. I_{13}	(2,1)	(1,2)	(2,8),(8,2)	(2,10),**(5,5)**,(10,2)
$SUPS(n)$ search lin. I_{12}	(2,1)	(1,2)	(2,8),(8,2)	(2,10),(5,5),(10,2)
$SUPS(n)$ search lin. I_{23}	(2,1)	(1,2)	(2,8),(8,2)	(2,10),(5,5),(10,2)
$H^{\infty}_{KDLS}(n)$	(2,1)	(1,2)	(2,2)	(2,5),(5,2)

Table 7.2: Sample graph for $KDLS$ procedure: sets of stored supported solutions and heuristic values H^{∞}_{KDLS} returned

7.2.2 Properties

When $k = \infty$, the termination criterion for $KDLS$ uses the source node s as reference. Maybe more heuristic values could be found for other nodes, but we propose to stop at the moment that all supported solutions from γ to s have been found.

Theorem 7.1 *The total number of single-objective searches performed by $KDLS$ procedure, with parameter k, is at most $2^k + 1$. Thus, the maximum number of heuristic vector values obtained with $KDLS$ for node s is at most 2^k.*

Each time a new supported solution is found, two new subproblems arise. The total number of searches performed in the dichotomic process is then at most $2^k - 1$, as each time 2 new unexplored subintervals can be generated. The extreme solutions require two additional single-objective searches, and hence the total number of searches performed is at most $2^k + 1$. Regarding the number of heuristic vector values obtained by $KDLS$, each time a new supported solution is found two new heuristic values can be obtained (see figure 7.1). Concatenating the split intervals, it is easy to see that a maximum of 2^k possible heuristic values can be obtained with (a maximum of) $2^k - 1$ supported solutions (see figure 7.3(c)).

When the process is carried to the limit (i.e. $k = \infty$), a maximal subset of H^* set is found, constructed from all supported solutions. For $k = \infty$, the procedure stops when all supported solutions from γ to s have been found. Thus, the precalculation procedure in that case can be computationally intensive as the number of supported solutions depends on the nature of the problem and it can be high for large or complex problem instances. The determination of a good value for k is analyzed in section 7.4.

Corolary 7.1 *$KDLS$ terminates in a finite number of steps.*

It is obvious from theorem 7.1 that, in the case $k \neq \infty$, the number of searches (and hence the number of heuristic vectors obtained) is bounded by an integer. In the case $k = \infty$, the number of solutions that can be found with linear combinations is also bounded, as only supported solutions can be reached. Thus, the procedure terminates at some point returning the sets H^k_{KDLS}.

Theorem 7.2 *The heuristic function H^k_{KDLS} obtained with $KDLS$ is admissible.*

The proof is quite simple, since for any two supported solutions consecutive in lexicographic order $\vec{q}^{\,i}(n)$, $\vec{q}^{\,j}(n)$, the heuristic vector $\vec{h}_{ij}(n)$ obtained from them dominates all vectors dominated by $\vec{q}^{\,i}(n)$, $\vec{q}^{\,j}(n)$, as well as the rectangle defined by them, including the *duality gap*. This can be easily seen in figures 7.3(c) or 7.8, where the heuristic values generated are in the lower left corner of a rectangle, dominating all supported and non-supported solutions that will be found later.

Theorem 7.3 *Let $H^k_{KDLS}(n)$ be the heuristic function obtained for node n by procedure $KDLS$ with parameter k. Let $k1>k2>0$ be two values for this parameter in the precalculation performed by $KDLS$ for the same node. Then, the heuristic function $H^{k_1}_{KDLS}(n)$ is **at least as informed** than $H^{k_2}_{KDLS}(n)$.*

It is obvious, that with larger values of k, at least the same number of supported solutions are found, and hence at least the same or a higher number of possible heuristic values are available for each node. By the construction procedure of $KDLS$, new triangles result in heuristic values with a higher value for at least one of the objectives, and thus the heuristic function $H^k_{KDLS}(n)$ is more informed with larger values for k.

Corolary 7.2 *Increasing values of k lead to at least equally or more informed heuristics.*

This is quite straightforward, since further searches could find new supported solutions, and this would replace one heuristic vector by two others dominated by the original one (see figure 7.1).

Theorem 7.4 *The heuristic function H^k_{KDLS} obtained with $KDLS$ can be in general inconsistent.*

It is enough to present a counterexample to the (equivalent) monotonicity property (see definition 4.11), i.e. some case where for some arc (n, n') between two adjacent nodes n, n',

$$\exists \vec{h}' \in H(n') \quad \forall \vec{h} \in H(n) \quad | \quad \vec{h} \not\preceq \vec{c}(n, n') + \vec{h}' \tag{7.8}$$

Let us consider the example presented in section 7.2.1 and heuristic H^∞_{KDLS}. We can observe the heuristic values obtained for nodes s and 1 in table 7.2. Given that $H^\infty_{KDLS}(s) = \{(2, 5), (5, 2)\}$, $H^\infty_{KDLS}(1) = \{(2, 2)\}$, and $\vec{c}(s, 1) = (1, 1)$, we have that for (2,2)

$$(2, 5) \not\preceq (2, 2) + (1, 1)$$
$$(5, 2) \not\preceq (2, 2) + (1, 1)$$

Theorem 7.4 is of great formal importance. When heuristics are inconsistent, there is no guarantee that a label expanded by $NAMOA^*$ is permanent and will not be later pruned. Theorem 4.5 (see section 4.2.2) states that $NAMOA^*$ with consistent heuristics will only expand C^*-bounded paths, but also states that, as a necessary condition, these paths must be nondominated. Let us recall the following important theorems that apply to this situation:

Theorem 7.5 *(Mandow & Pérez de la Cruz, 2010a, Theorem 5.2) A sufficient condition for NAMOA* to select a path $P = (s, \ldots, n)$ for expansion is that:*

a) *P be a nondominated path from s to n*

b) *P be C^*-bounded*

Theorem 7.6 *(Mandow & Pérez de la Cruz, 2010a, Theorem 5.3) A necessary condition for $NAMOA^*$ to select a path $P = (s, \ldots, n)$ for expansion is that P be C^*-bounded.*

This means that all paths bounded by C^* could be expanded by $NAMOA^*$ whether they are nondominated or not. The sequence of heuristics calculated with $KDLS$ procedure is increasingly more informed (see definition 4.5), for larger values of k. However, more informed but inconsistent heuristics do not guarantee a reduction in search effort. All we can expect is stated by the next theorem.

Theorem 7.7 *(Mandow & Pérez de la Cruz, 2010a, Theorem 5.5) Let $H_1(n)$ and $H_2(n)$ be two admissible heuristics for the same problem. Let $NAMOA^*_1$ and $NAMOA^*_2$ be two versions of algorithm $NAMOA^*$ that differ only in the use of heuristic functions $H_1(n)$ and $H_2(n)$ respectively. If $H_2(n)$ is at least as informed as $H_1(n)$, then all nondominated and $C^*(H_2)$-bounded paths selected for expansion by $NAMOA^*_2$ will also be selected by $NAMOA^*_1$.*

As explained in section 2.4.6, the number of dominated labels explored can be reduced using multiobjective pathmax. Therefore, all experiments with H_{KDLS} have been run with algorithm $NAMOA^{**}$.

7.3 Results

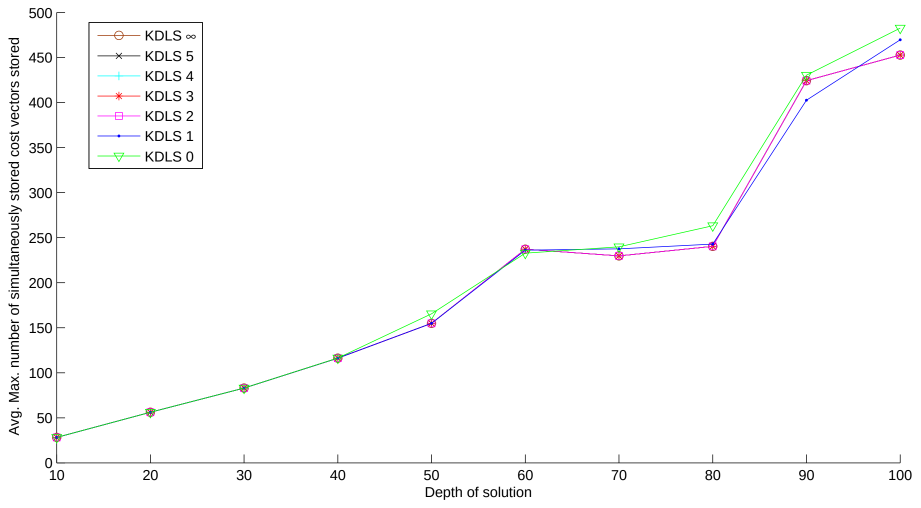
This section reports experimental results on the evaluation of $NAMOA^{**}$, the pathmax version of $NAMOA^*$, with H_{KDLS}. Two problem domains with two objectives are evaluated: class II grids and route planning in road networks. In the case of class II grids (see section 3.2.1.2), graph size (or equivalently, solution depth) is gradually incremented. Additionally, correlation between objectives is set to -0.8, -0.4, 0, 0.4, or 0.8. The route planning problem set is the one defined in section 3.2.3 over the NY_2 map. The objectives in this map have an overall correlation ratio $\rho = 0.16$.

The purpose of these experiments is to evaluate the impact of H_{KDLS} in performance as a function of k, i.e. $H(n) = H^k_{KDLS}(n)$ with $k \in [1, 5]$. Two additional alternatives are considered: $H(n) = H^0_{KDLS}(n)$ (i.e. $H(n) = \{\vec{h}_{TC}(n)\}$, $k = 0$) and $H(n) = H^{\infty}_{KDLS}(n)$ (i.e. $k = \infty$). The bounding improvements in the precalculation of heuristics described by (7.5) and (7.6) have been also applied. The heuristic values are also strengthened in run time through the application of the multiobjective pathmax rule, described in definition 2.21.

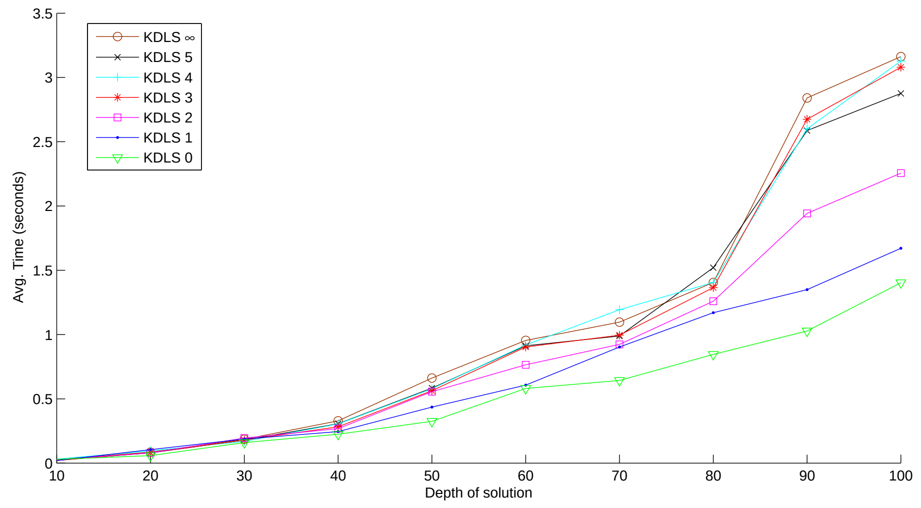
In these experiments $NAMOA^{**}$ is run with a linear selection rule, which provided the best results with consistent heuristics. The same order was applied to G_{op} sets. The algorithm was implemented in ANSI Common Lisp using LispWorks 6.0 Enterprise 64 bits, and run on a Sun Fire X4140 server with 2 six-core AMD Opteron 2435 @ 2.60GHz processors and 64 Gb of DDR2 RAM, under Windows Server 2008 R2 Enterprise (64-bits).

7.3.1 Results on grids

The space and time requirements of $NAMOA^{**}$ with H_{KDLS}, $k \in [0, 5]$ and $k = \infty$, for the case of class II problem instances are presented in figures 7.9(a), 7.9(b) (for the case of ρ=0.8), figures 7.10(a), 7.10(b) (for the case of ρ=0.4), figures 7.11(a) and 7.11(b) (for the case of $\rho = 0$), figures 7.12(a), 7.12(b) (for the case of ρ=-0.4), and figures 7.13(a), 7.13(b) (for the case of ρ=-0.8), respectively. Values are averaged over ten problems generated for each depth, in all reported results hereafter. Time results presented in this section show total time including heuristic precalculations.

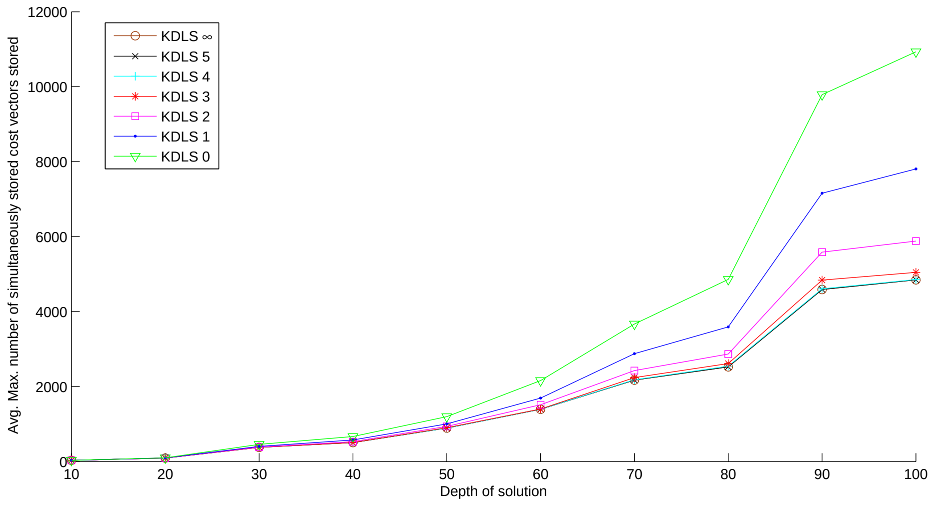


(a) Space requirements for correlation value 0.8

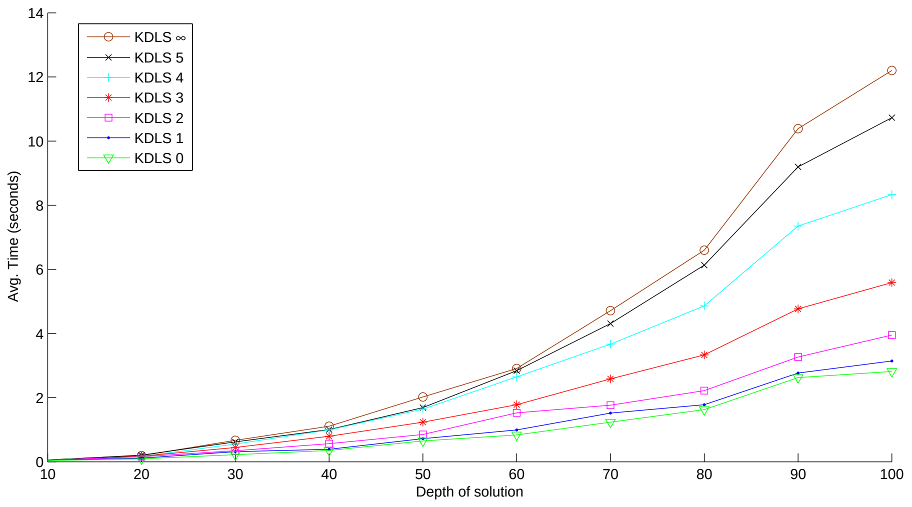


(b) Time requirements for correlation value 0.8

Figure 7.9: Space and time requirements for $NAMOA^{**}$ with H^k_{KDLS} on class II grid problems with ρ =0.8. Average values over ten random problems generated for each depth.

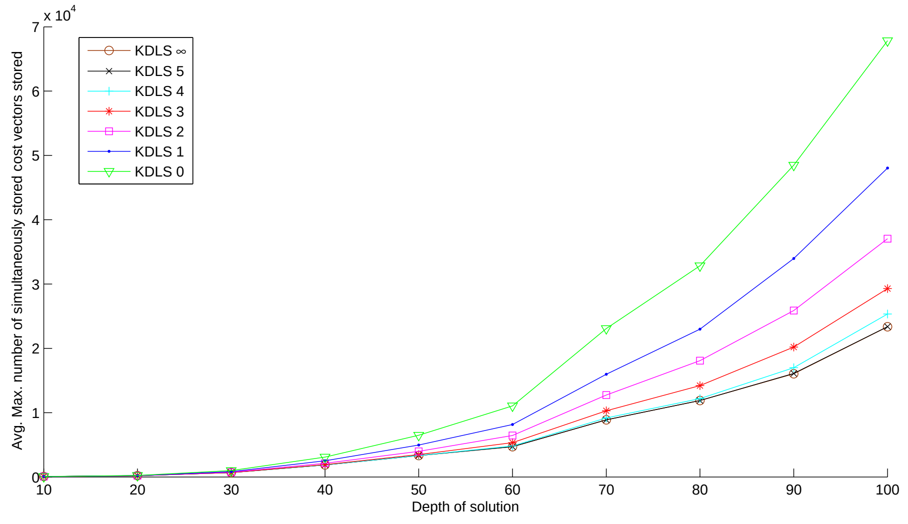


(a) Space requirements for correlation value 0.4

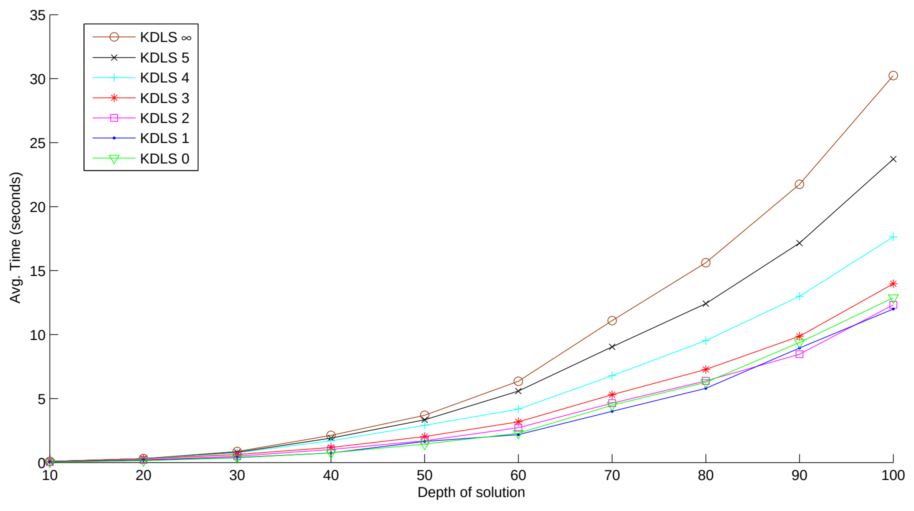


(b) Time requirements for correlation value 0.4

Figure 7.10: Space and time requirements for $NAMOA^{**}$ with H^k_{KDLS} on class II grid problems with ρ =0.4. Average values over ten random problems generated for each depth.

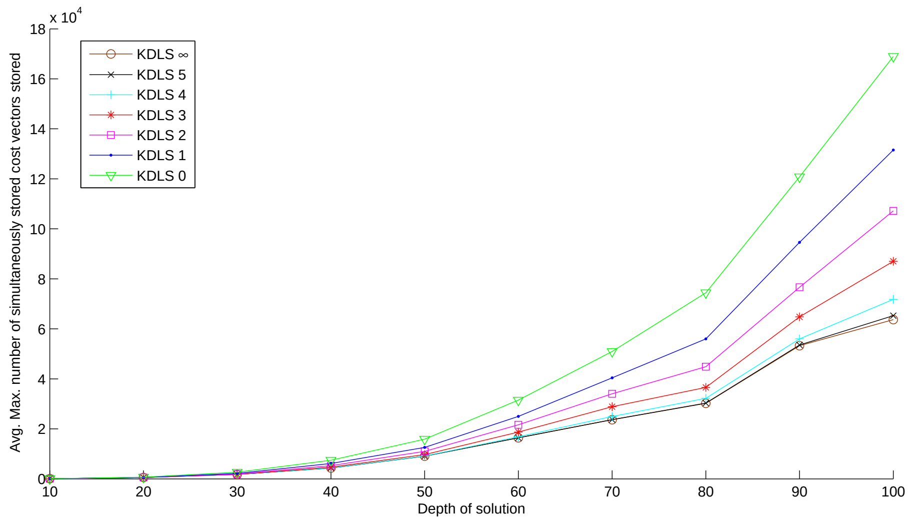


(a) Space requirements for correlation value 0

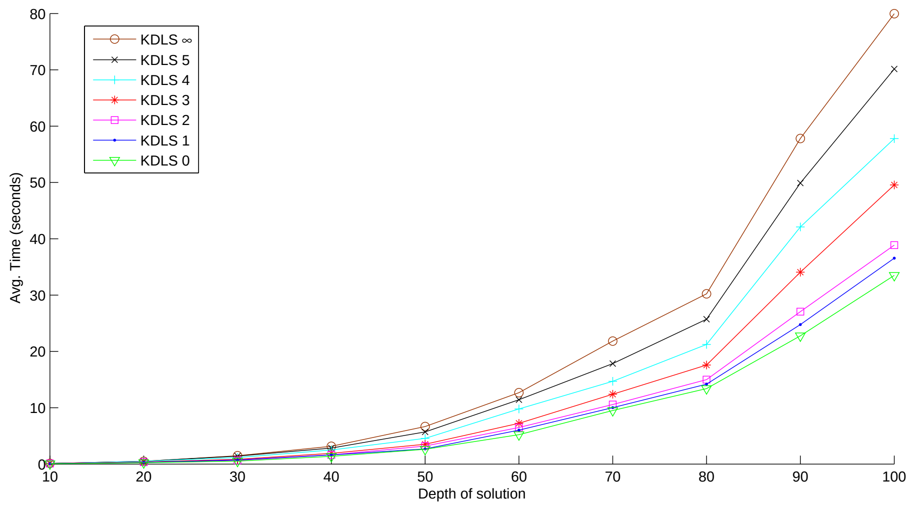


(b) Time requirements for correlation value 0

Figure 7.11: Space and time requirements for $NAMOA^{**}$ with H^k_{KDLS} on class II grid problems with $\rho = 0$. Average values over ten random problems generated for each depth.

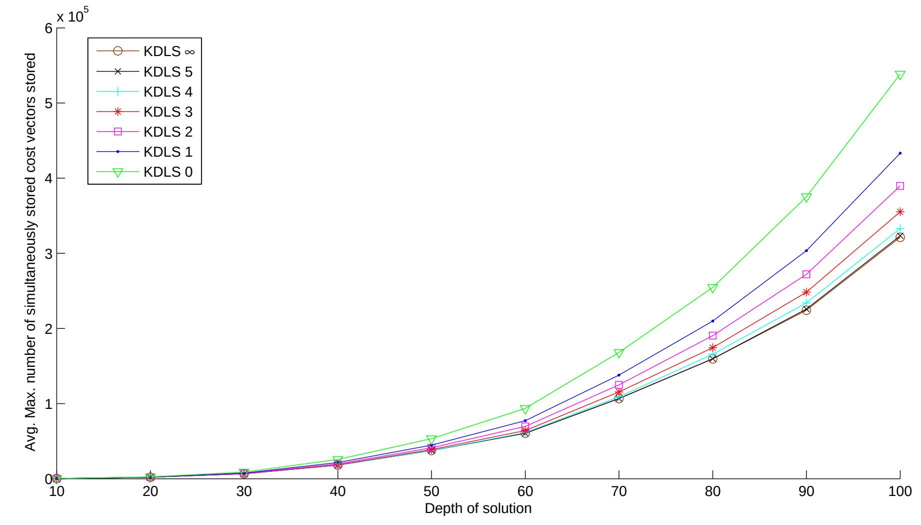


(a) Space requirements for correlation value -0.4

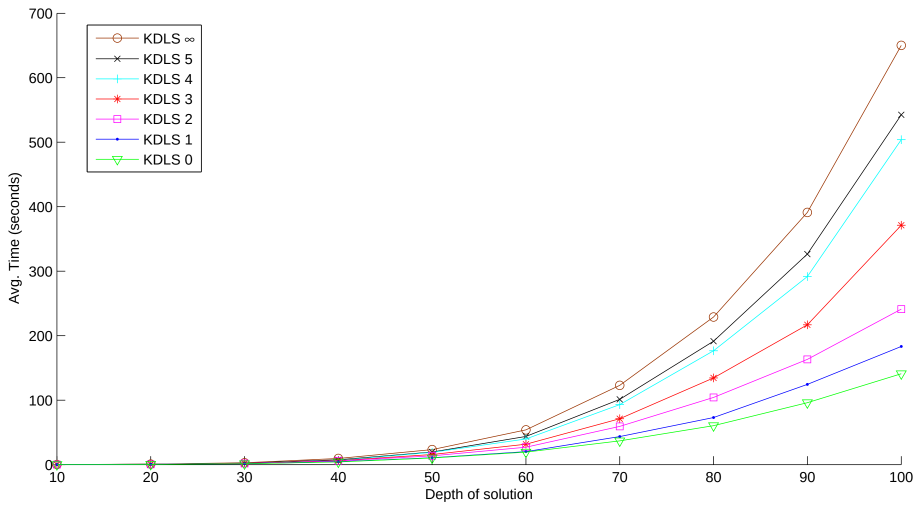


(b) Time requirements for correlation value -0.4

Figure 7.12: Space and time requirements for $NAMOA^{**}$ with H^k_{KDLS} on class II grid problems with ρ =-0.4. Average values over ten random problems generated for each depth.



(a) Space requirements for correlation value -0.8



(b) Time requirements for correlation value -0.8

Figure 7.13: Space and time requirements for $NAMOA^{**}$ with H^k_{KDLS} on class II grid problems with ρ =-0.8. Average values over ten random problems generated for each depth.

7.3.2 Results on road maps

The algorithm $NAMOA^{**}$ was also evaluated on a realistic scenario. Space requirements for the 20 problem instances of NY_2 map are shown in figure 7.14. The abscissa axis x shows problem indexes for each instance in the problem set. Instances are ordered according to increasing number of stored cost vectors needed by $NAMOA^{**}$ with heuristic H^{∞}_{KDLS}. The value for ordinate y (the number of simultaneously stored cost vectors) is shown in a logarithmic scale.

Time requirements for the same problem instances are shown in figure 7.15. The value for ordinate y (time in seconds) is also shown in a logarithmic scale. This time, instances are ordered by increasing time needed by $NAMOA^{**}$ with heuristic H^{0}_{KDLS} (i.e. $\vec{h}_{TC}$), which was identified as the fastest alternative in general over class II grid problems. For practical reasons, instances were solved with a twelve-hour time limit. In those cases where the heuristic could not solve an instance, values are not displayed for that particular heuristic. In addition, values obtained for problem instances NY_2 15 and NY_2 3 were very small and outside the displayed scale.

7.4 Analysis

7.4.1 Analysis on space requirements

A Grids

A great reduction in space requirements can be achieved with H^{k}_{KDLS} for all values of $k > 0$, when compared to $\vec{h}_{TC}$. This can be observed for all values of ρ (figures 7.10(a), 7.11(a), 7.12(a), and 7.13(a)), except in the simpler case of ρ=0.8, (figure 7.9(a)). Results for $k = 5$ and $k = \infty$ appear indistinguishable at this scale for all correlations. The first conclusion is that there are diminishing returns to precalculations beyond $k = 5$, i.e. at most 32 heuristic cost vectors in $H(n)$. The relative gain between $k = 5$ and $k = \infty$ is really small in general.

Let us analyze in some detail the ratio of space requirements with H^{k}_{KDLS} for different values of k respect to the optimal space requirements. These are measured as the space required to store a completelly labeled solution graph (i.e. one in which every node belonging to a nondominated solution is labeled with optimal costs to the source and goal nodes). Let $v_{\mathrm{NAMOA_k}}$ be the maximum number of cost vectors simultaneously stored by $NAMOA^{**}$ with H^{k}_{KDLS}, and let $v_{\mathrm{NAMOA_{sol}}}$ be the strictly necessary cost vectors belonging to solution paths to the problem instance. We shall denote by $rv^{sol}_{k} = v_{\mathrm{NAMOA_k}}/v_{\mathrm{NAMOA_{sol}}}$ the ratio of cost vectors stored by $NAMOA^{**}$ with H^{k}_{KDLS} compared to size of the solution graph. The relative ratio rv^{sol}_{k} is shown for the different values of k in figures 7.16(a), 7.17(a), 7.18(a), 7.19(a), and 7.20(a).

For example, let us consider figure 7.18(a) in some detail (i.e. ρ=0). The ratio for $k = 0$ (i.e. $\vec{h}_{TC}$) is on average somewhat below 12 for the most difficult problems (i.e. deeper solution), while that for $k = \infty$ is nearly 4.

The second conclusion is that the ratio rv^{sol}_{k} grows more slowly with solution depth for higher values of k.

From figure 7.18(a), it is also obvious that the different values of k considerably reduce the space requirements over $\vec{h}_{TC}$ (i.e. $k = 0$). The reduction steadily increases

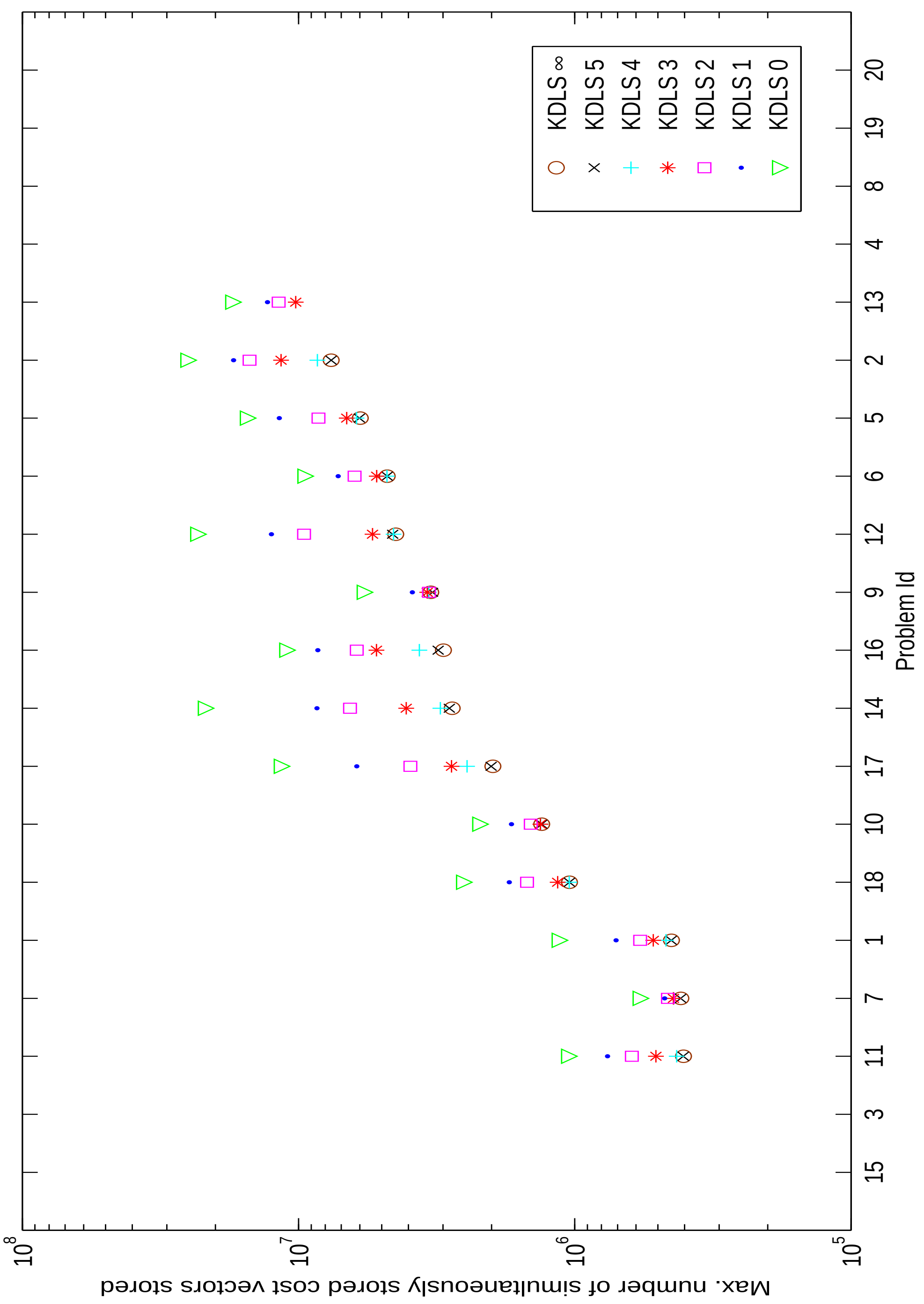


Figure 7.14: Space requirements for $NAMOA^{**}$ with H^{k}_{KDLS} in NY_2 map. Problem instances are ordered by increasing number of stored cost vectors needed by $NAMOA^{**}$ with heuristic H^{∞}_{KDLS}

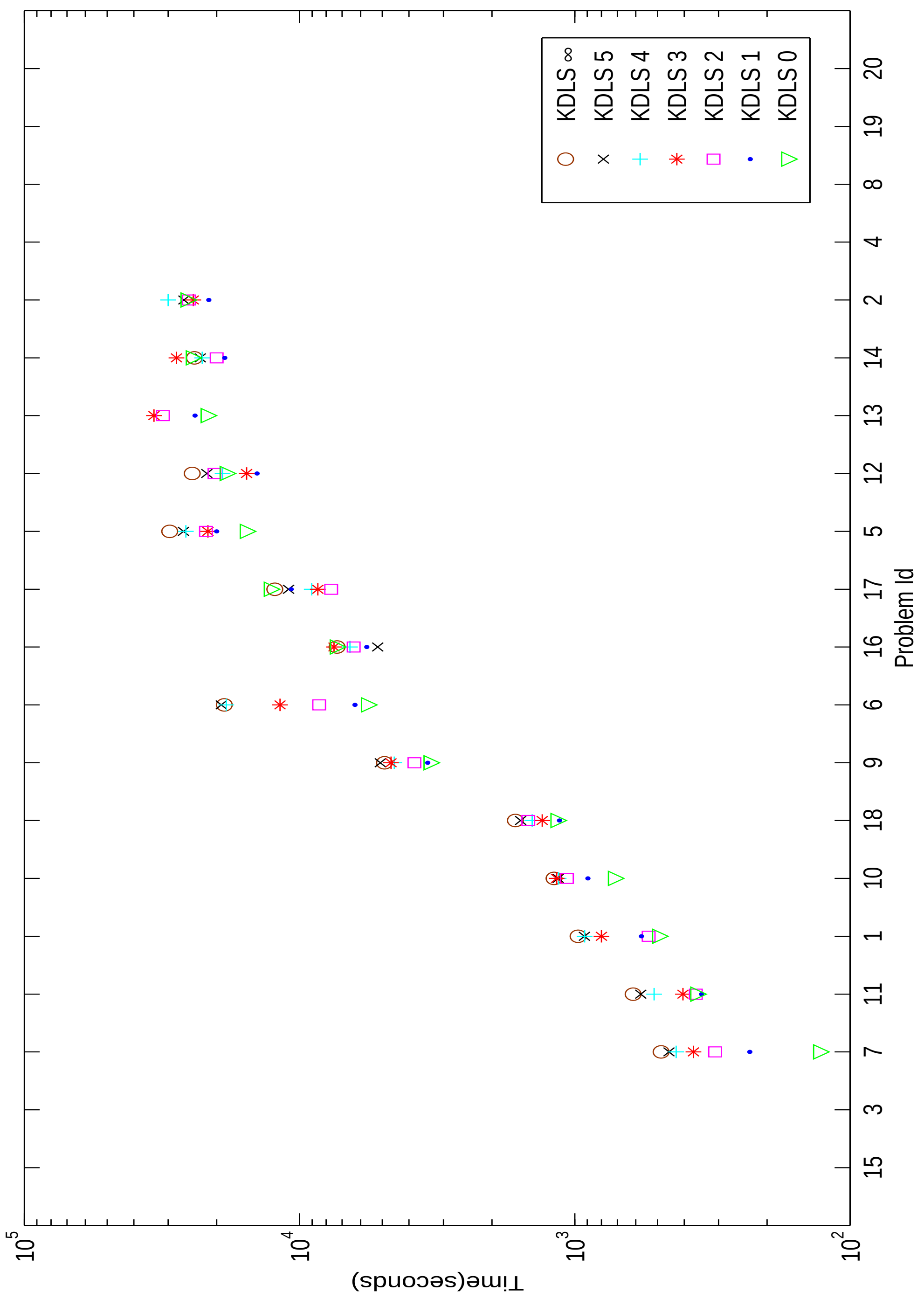


Figure 7.15: Time requirements for $NAMOA^{**}$ with H^k_{KDLS} in NY_2 map. Problem instances are ordered by increasing time needed by $NAMOA^{**}$ with heuristic H^0_{KDLS}

with solution depth. For example, for $k = \infty$ the ratio of reduction over $k = 0$ is approximately 3 for the problems with deeper solution.

B Road maps

Considering the number of label expansions, these problems are around two orders of magnitude more difficult than the random grid problems. The biggest grids have 40,000 nodes and 159,200 arcs and can have on average 124.8 Pareto-optimal solution costs for ρ=0, while the NY_2 map has 264,346 nodes and 730,100 arcs with an average of 2086.6 Pareto-optimal solution costs per problem instance.

A great reduction in space requirements can be achieved with H^k_{KDLS} for all values of $k > 0$, when compared to $\vec{h}_{TC}$ (figure 7.14). Results for $k = 5$ and $k = \infty$ are very similar again for almost all problems. The conclusion that there are diminishing returns to precalculations beyond $k = 5$ still holds for these problems. The relative gain between $k = 5$ and $k = \infty$ is again small in general.

Figure 7.21 shows the relative space requirements rv^{sol}_k for each value of k compared to solution graph size for the route planning problem instances. The abscissa axis x shows problem indexes for each instance in the problem set. Instances are ordered by increasing size of solution graph.

In this domain, the ratios vary widely for small values of k in some instances. In general, for higher values of k (i.e. $k > 3$) the ratio is below 30, while for $k = 0$, it can reach much higher values. For example, intance NY_2 12 presents a ratio close to 140 for $k = 0$. The explanation is that the solution graph size is relatively small for this problem, compared to the search graph size. This can be attributed to the irregularities in the internal structure of road maps.

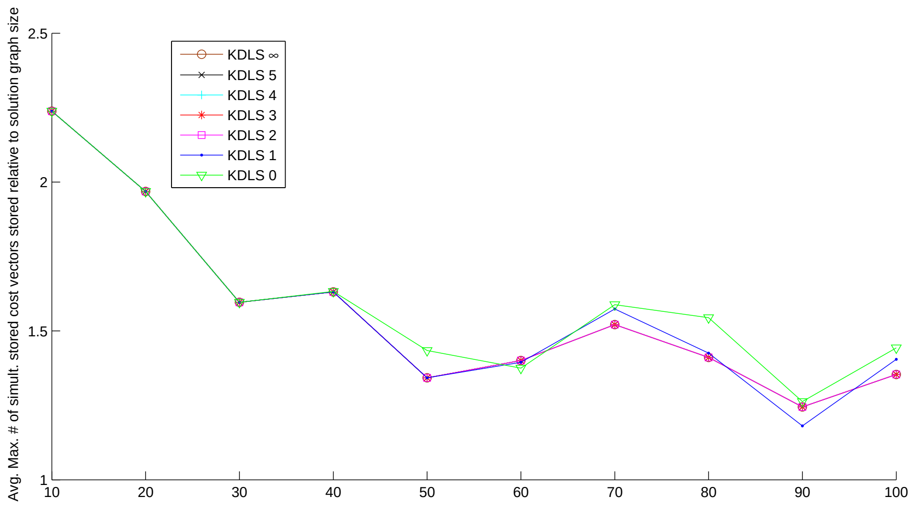
Regarding the comparison between multivalued heuristics with $\vec{h}_{TC}$ some instances exhibit a reduction similar to that obtained for difficult random grid instances with $\rho = 0$, e.g. for $k = \infty$, instance NY_2 11 improves over $\vec{h}_{TC}$ by a factor of 2.71. Other instances show higher improvements, like NY_2 2 (with 3.5), NY_2 16 (with 3.9), NY_2 12 (with 5.25) and NY_2 17 (with 5.92). The conclusion is that multivalued heuristics can provide significant savings in space requirements over $\vec{h}_{TC}$.

7.4.2 Analysis on time requirements

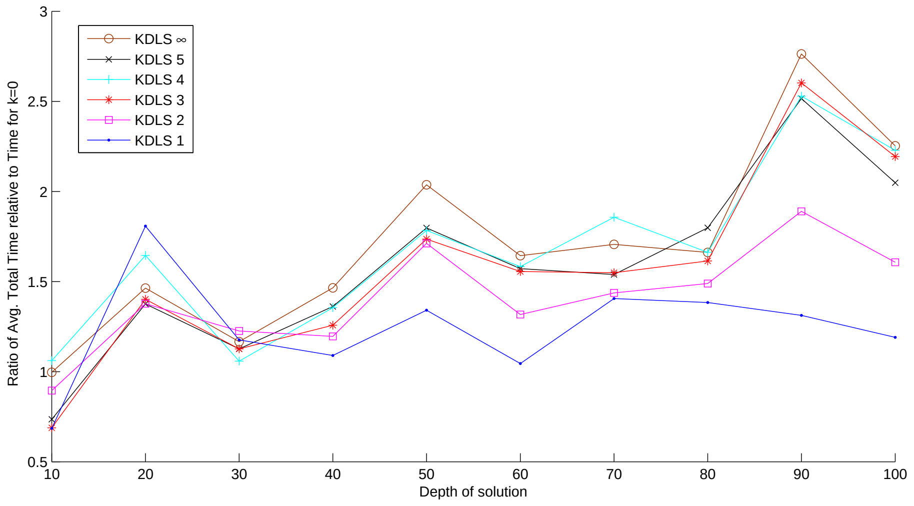
A Grids

Figures 7.9(b), 7.10(b), 7.11(b), 7.12(b), and 7.13(b) show that the absolute time requirements of $NAMOA^{**}$ with H^k_{KDLS} increase with k in grid problems. This indicates that the reduction in space requirements come with a price in terms of run time. Nevertheless, the worst time requirements achieved by multivalued heuristics are still much smaller than those of blind search.

We can analyze the ratio of time taken by multivalued heuristics compared to $\vec{h}_{TC}$. Let $t_{\mathrm{NAMOA_k}}$ be the total time (including heuristics precalculation) taken to solve a problem by $NAMOA^{**}$ with H^k_{KDLS}. We shall denote by $rt^0_k = t_{\mathrm{NAMOA_k}}/t_{\mathrm{NAMOA_0}}$ the ratios of performance in time from $NAMOA^{**}$ with H^k_{KDLS} against H^0_{KDLS} (i.e. equivalent to $\vec{h}_{TC}$). The relative ratio rt^0_k is shown for the different values of ρ in figures 7.16(b), 7.17(b), 7.18(b), 7.19(b), and 7.20(b).

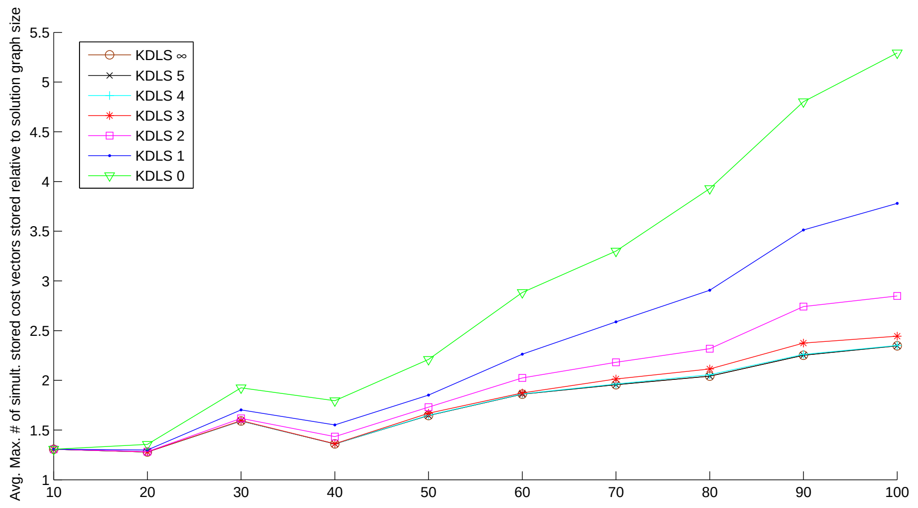


(a) Relative space requirements rv_k^{sol} for correlation value 0.8

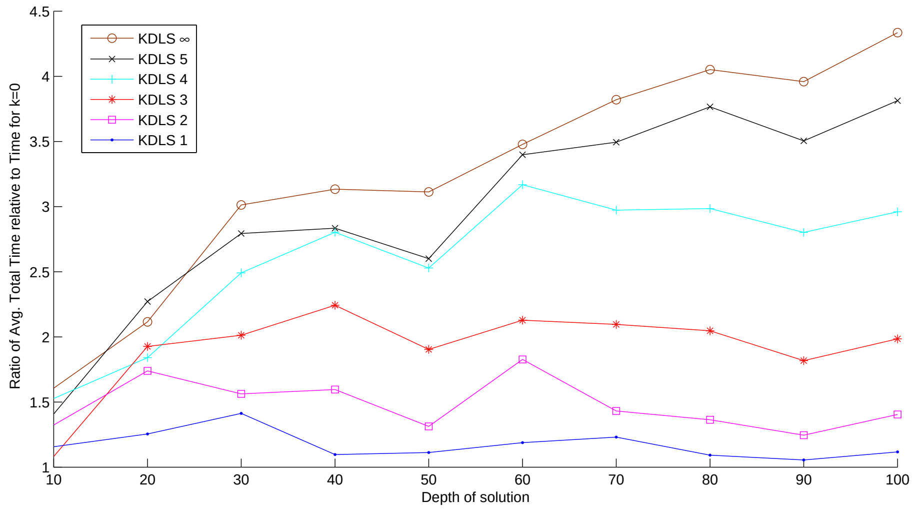


(b) Relative time requirements rt_k^0 for correlation value 0.8

Figure 7.16: Class II grid problems with ρ =0.8. Average values over ten problems generated for each depth. (a) Space requirements for $NAMOA^{**}$ with H_{KDLS}^k relative to graph solution size. (b) Time requirements for $NAMOA^{**}$ with H_{KDLS}^k relative to $NAMOA^{**}$ with H_{KDLS}^0.

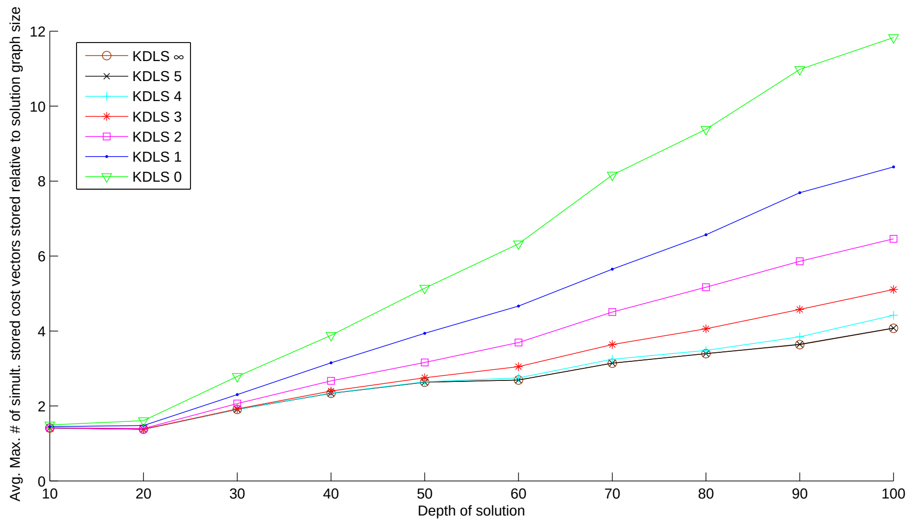


(a) Relative space requirements rv_k^{sol} for correlation value 0.4

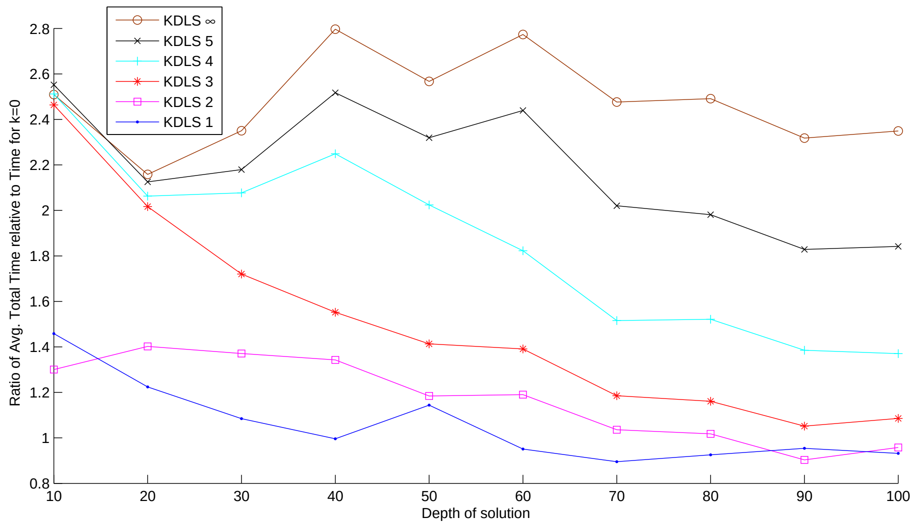


(b) Relative time requirements rt_k^0 for correlation value 0.4

Figure 7.17: Class II grid problems with ρ =0.4. Average values over ten problems generated for each depth. (a) Space requirements for $NAMOA^{**}$ with H_{KDLS}^k relative to graph solution size. (b) Time requirements for $NAMOA^{**}$ with H_{KDLS}^k relative to $NAMOA^{**}$ with H_{KDLS}^0.

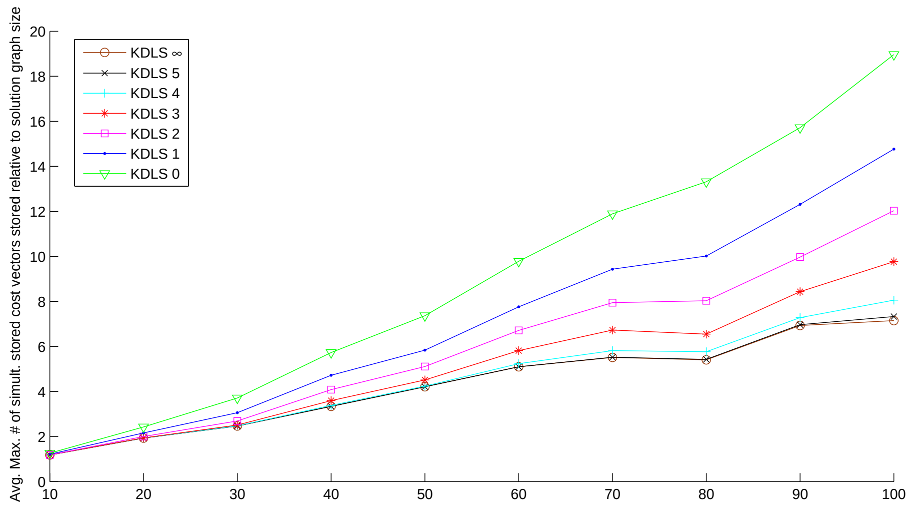


(a) Relative space requirements rv_k^{sol} for correlation value 0

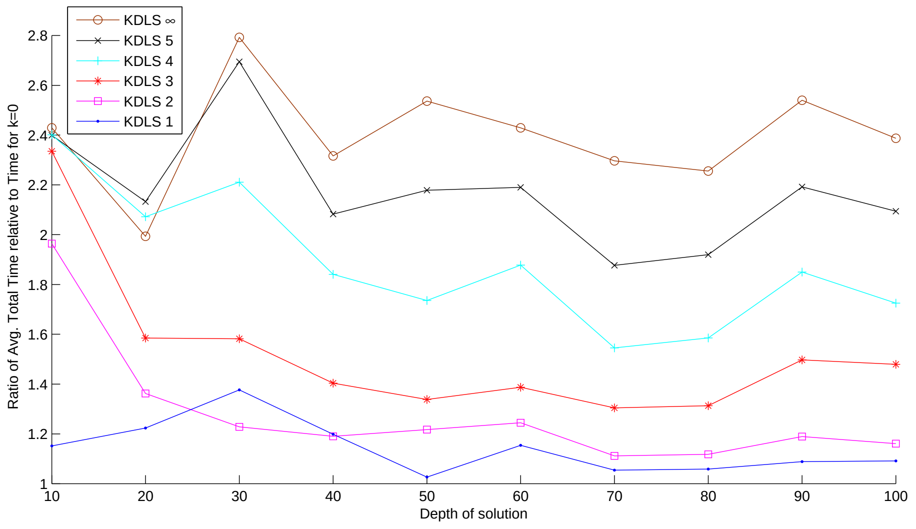


(b) Relative time requirements rt_k^0 for correlation value 0

Figure 7.18: Class II grid problems with ρ =0. Average values over ten problems generated for each depth. (a) Space requirements for $NAMOA^{**}$ with H^k_{KDLS} relative to graph solution size. (b) Time requirements for $NAMOA^{**}$ with H^k_{KDLS} relative to $NAMOA^{**}$ with H^0_{KDLS}.

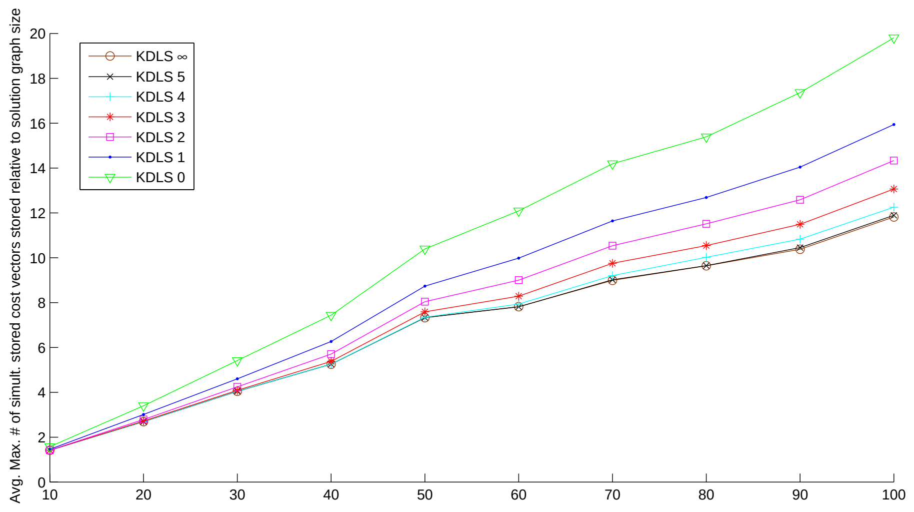


(a) Relative space requirements rv_k^{sol} for correlation value -0.4

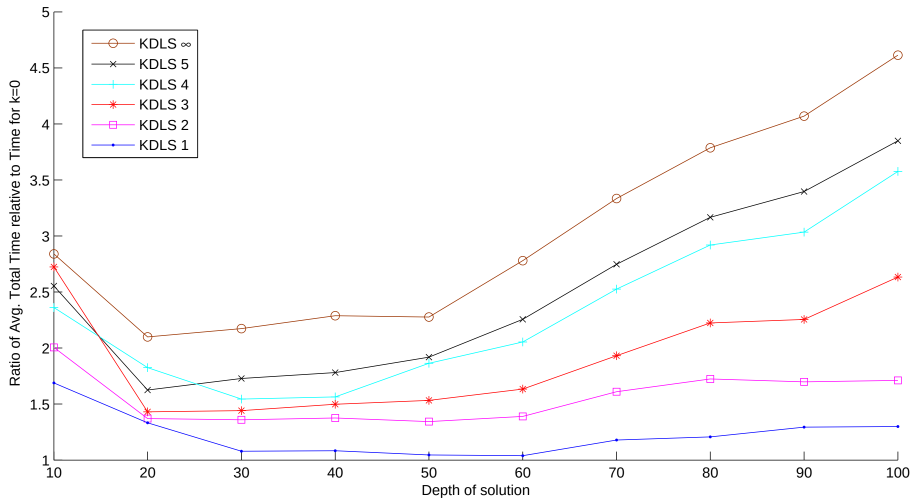


(b) Relative time requirements rt_k^0 for correlation value -0.4

Figure 7.19: Class II grid problems with ρ =-0.4. Average values over ten problems generated for each depth. (a) Space requirements for $NAMOA^{**}$ with H_{KDLS}^k relative to graph solution size. (b) Time requirements for $NAMOA^{**}$ with H_{KDLS}^k relative to $NAMOA^{**}$ with H_{KDLS}^0.



(a) Relative space requirements rv_k^{sol} for correlation value -0.8



(b) Relative time requirements rt_k^0 for correlation value -0.8

Figure 7.20: Class II grid problems with ρ =-0.8. Average values over ten problems generated for each depth. (a) Space requirements for $NAMOA^{**}$ with H^k_{KDLS} relative to graph solution size. (b) Time requirements for $NAMOA^{**}$ with H^k_{KDLS} relative to $NAMOA^{**}$ with H^0_{KDLS}.

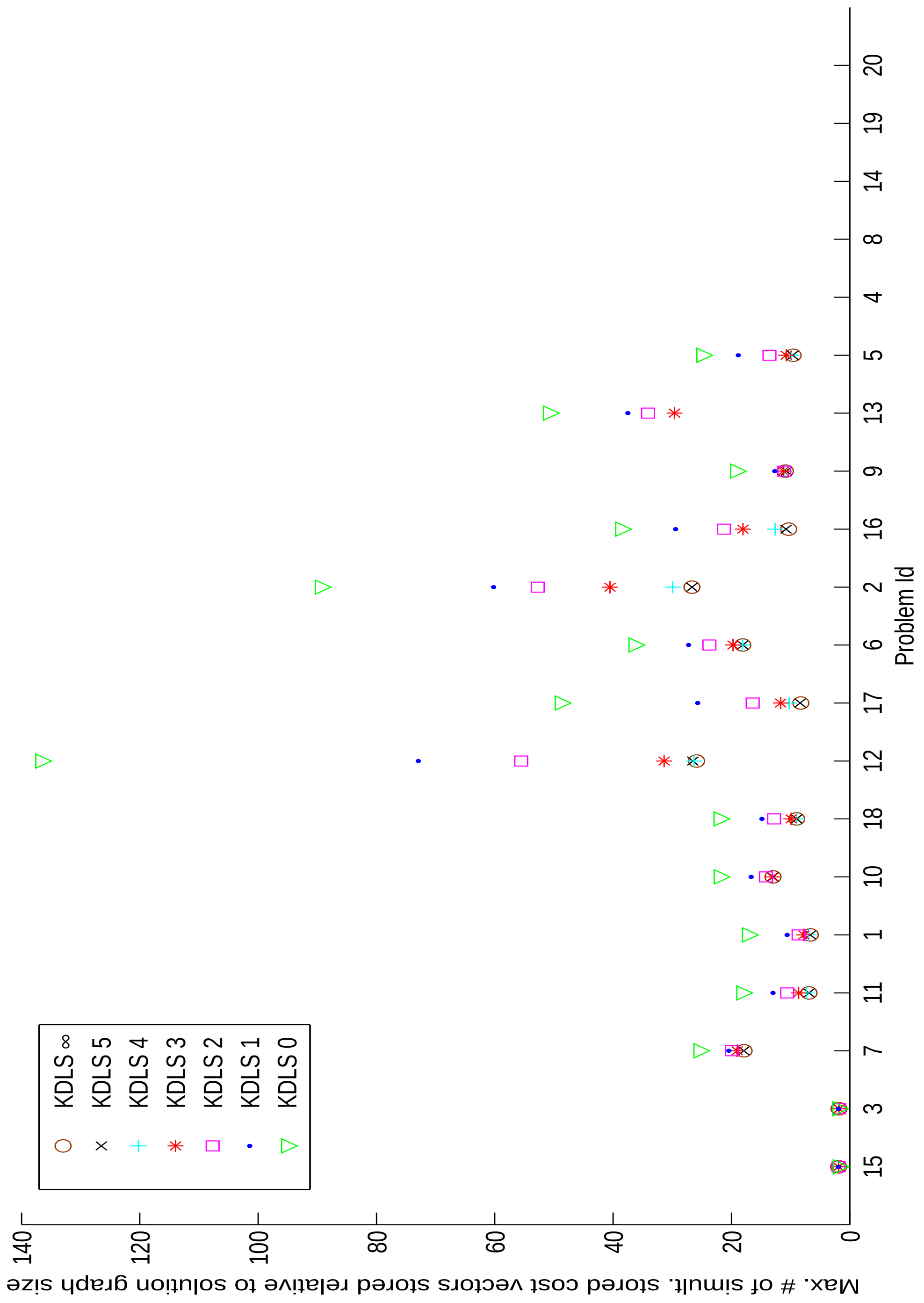


Figure 7.21: Ratio rv_k^{sol} of space requirements for $NAMOA^{**}$ in NY_2 map, relative to solution graph size. Problem instances are ordered by increasing size of solution graph.

For example, let us consider in detail the case $\rho = 0$ (figure 7.18(b)). Only $k = 1$ or $k = 2$ are competitive in time for difficult problems with $\vec{h}_{TC}$ offering significative space reductions for a small increase in run time.

A similar scenario appears for other correlation values, except the extreme values 0.8 and -0.8. For $\rho = 0.8$ the precalculation of heuristics does not compensate, since it does not provide a significant space reduction. For $\rho = -0.8$, $k = 2$ already requires 50% more time than $\vec{h}_{TC}$, though still offering some reduction in space.

It is interesting to analyze the source of this increase in time requirements for multivalued heuristics. Let us consider first the ratio of time due to the precalculation procedure compared to total time invested by $NAMOA^{**}$. Figure 7.22(a) shows the absolute time invested in the precalculation of the H_{KDLS} heuristics. Let $t_{\mathrm{KDLS_k}}$ be the time taken by $KDLS$ procedure to precompute the heuristic H_{KDLS} for each node. We shall denote by $rt^h_k = t_{\mathrm{KDLS_k}}/t_{\mathrm{NAMOA_k}}$ the ratio of precalculation time compared to overall run time of $NAMOA^{**}$ with H^k_{KDLS} (including both precalculation, and multiobjective search). The values of rt^h_k as a function of solution depth d for class II problems with $\rho = 0$ are shown in figure 7.22(b). Notice that this figure shows the ratios of performance as percentages.

For shallower solutions, the ratio is very high, and gradually decreases with solution depth. This figure also shows that while $\vec{h}_{TC}$ $(k = 0)$ represents only about 20% of time for the problems with deeper solutions, H^{∞}_{KDLS} can take about 70% of the time requirements for $NAMOA^{**}$.

B Road maps

In the case of NY_2 map, figure 7.15 shows that in some cases, larger values of k increase run time, while in others, some reduction for certain values of k can be achieved.

Figure 7.23 shows the ratios rt^0_k of time performance for $NAMOA^{**}$ with H^k_{KDLS} against H^0_{KDLS} (equivalent to $\vec{h}_{TC}$). Instances are ordered by increasing time needed by $NAMOA^{**}$ with heuristic H^0_{KDLS} (i.e. $\vec{h}_{TC}$).

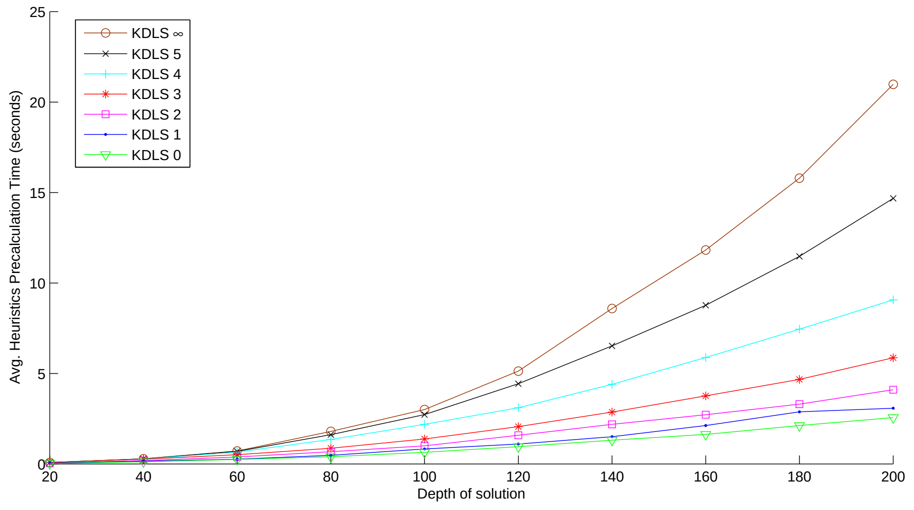
The time behavior of multivalued heuristics varies widely for different instances:

- In relatively easy instances for $\vec{h}_{TC}$, like NY_2 7, NY_2 11, NY_2 1, or even NY_2 6, time requirements increase considerably with k.
- For relatively difficult instances, like NY_2 16, NY_2 17 or NY_2 14, almost all values of k reduce time requirements.
- However, for other hard instances, like NY_2 5, or NY_2 13, higher values of k increase run time.

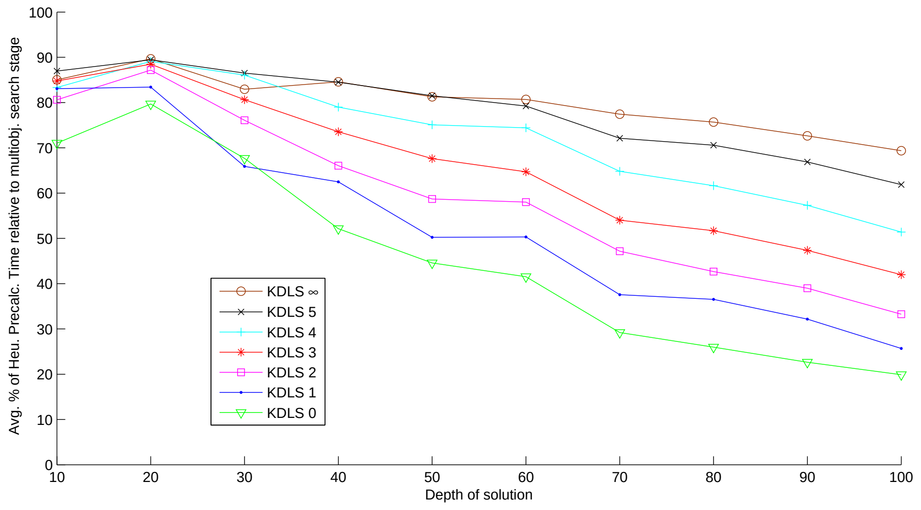
The conclusion is that multivalued heuristics can provide savings in both space and time for some hard problem instances. However, the relative benefits are difficult to anticipate. Once again, this can be attributed to the irregularities in the internal structure of road maps.

7.4.3 Influence of k in heuristic performance

In this section, we analyze the influence of multivalued heuristics in search performance, with special attention to time requirements. Previous analyses have shown that mul-



(a) Time requirements for H^k_{KDLS} precalculation



(b) Relative time requirements rt^h_k for correlation value 0

Figure 7.22: Time requirements for heuristics precalculation with $KDLS$ procedure on class II grid problems with $\rho = 0$. Average values over ten problems generated for each depth.

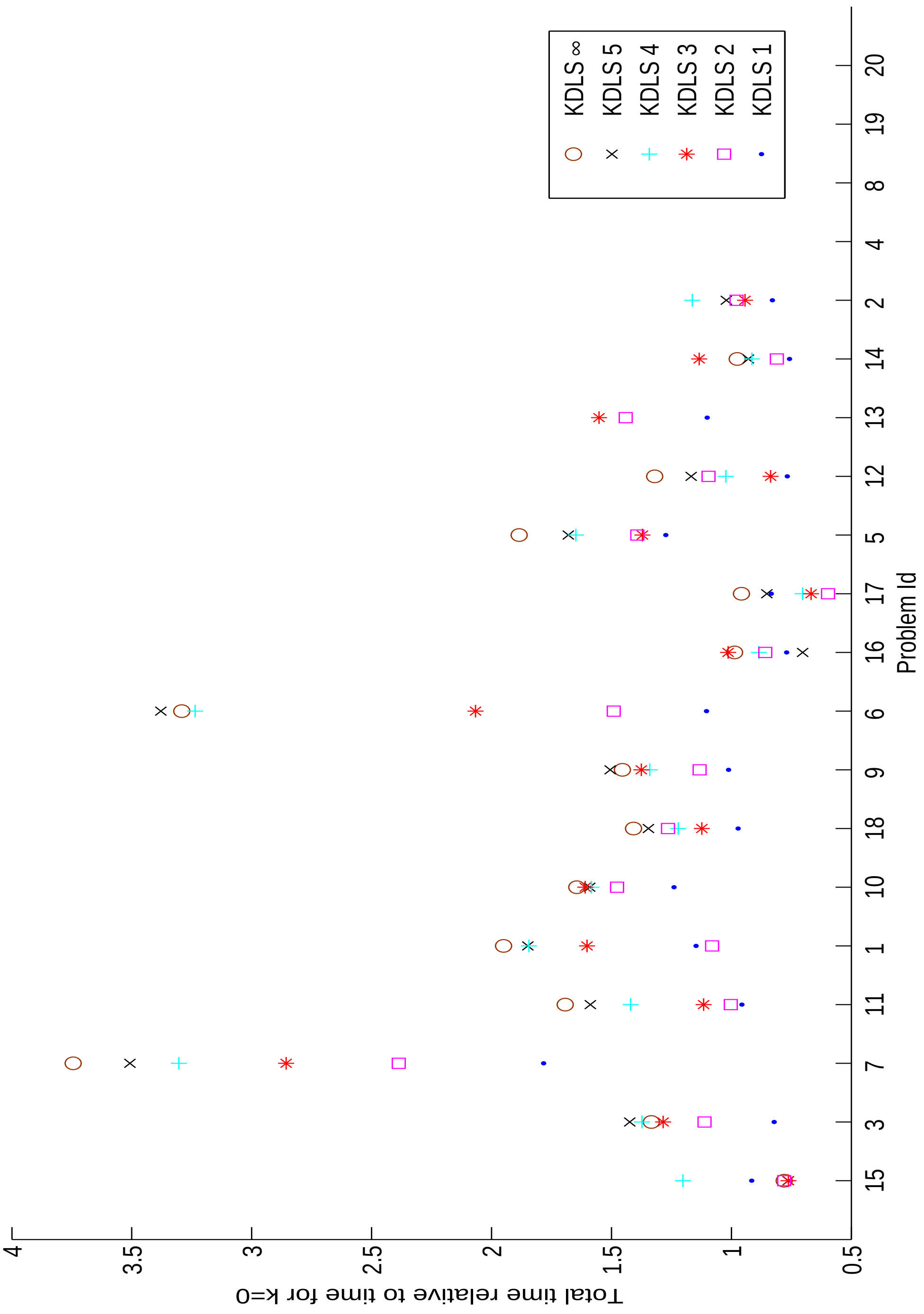


Figure 7.23: Ratio rt_k^0 of time requirements for $NAMOA^{**}$ with H_{KDLS}^k in NY_2 map, relative to time requirements for $NAMOA^{**}$ with $\vec{h}_{TC}$

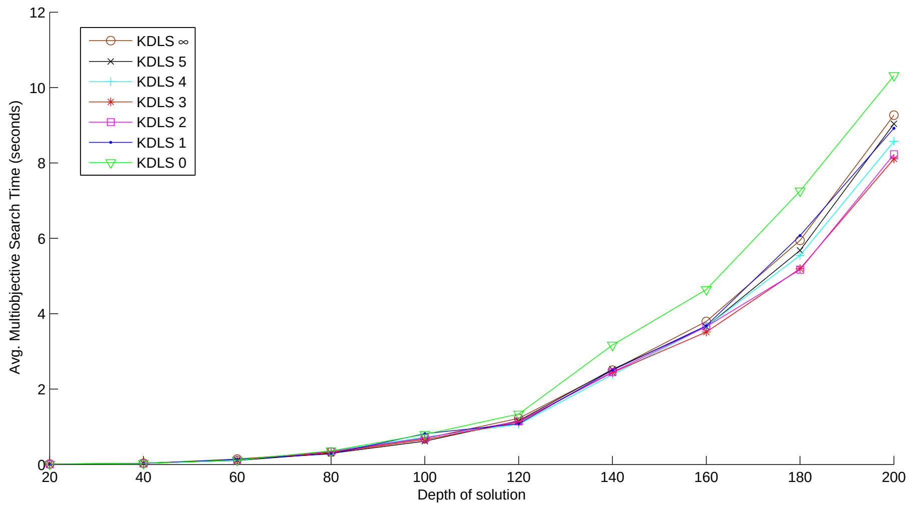


Figure 7.24: Time requirements for the multiobjective search phase of $NAMOA^{**}$ with H^k_{KDLS} in class II grid problems with $\rho = 0$

tivalued heuristics can increase time requirements in random grids and certain route planning instances. This could be attributed to a number of causes:

- A strong search effort in heuristic precalculation.
- A stronger search effort in the multiobjective search stage, in turn due to:
  - The expansion of a large number of dominated labels due to heuristic inconsistency.
  - A time overhead due to pruning dominance checks.
  - A time overhead due to filtering dominance checks.

Our hypothesis is that both precalculation time and multiobjective search time are partly responsible for the increase in run time. Let us consider these in turn.

Regarding precalculation effort, absolute times for $\rho = 0$ (figure 7.22(a)) reveal that for $k \leq 4$ precalculation time is below the overall time of search with $\vec{h}_{TC}$ in figure 7.11(b) (including precalculation and multiobjective search). This means that for the grid problems considered, we can not expect time improvements for larger values of k. However, the heuristic precision gained with values of $1 \leq k \leq 4$ is not paying off with a significative reduction in multiobjective run time. However, this could be a lesser evil, since precalculation times could be partially reduced running precalculation searches in parallel.

Regarding multiobjective search effort, figure 7.24 shows time taken by the multiobjective search phase for different values of k in the same set of problems (i.e. $\rho = 0$). All values of $k > 0$ require less time than $\vec{h}_{TC}$. However, the trend is not strictly decreasing with higher k, but slightly increases after reaching a minimum for $k = 3$.

	It.	Exp. Lab.	Dom. lab.	Filt. op.	Pruning Op.
$KDLS$ ∞	20,032	19,910	995	10,852,007	607,446
$KDLS$ 5	20,473	20,351	1,442	9,909,402	624,844
$KDLS$ 4	21,410	21,288	692	9,180,564	665,090
$KDLS$ 3	24,397	24,275	827	7,785,417	763,354
$KDLS$ 2	30,116	29,994	1,040	6,487,609	1,015,308
$KDLS$ 1	44,099	43,977	2,002	3,561,514	1,636,976
$KDLS$ 0	62,055	61,933	0	2,136,028	2,524,910

Table 7.3: Analysis of dominance checks on a sample 200×200 class II grid problem instance with $\rho = 0$. Start node is (100,100) and goal node is (50,50).

A first hypothesis to the increase in overall time requirements could be that a large number of dominated labels is being expanded due to heuristic inconsistency. Table 7.3 displays relevant information regarding the resolution of one of the hardest class II grid problems with $\rho = 0$ for different values of k. The first column displays the number of iterations, and the second the number of label expansions (excluding the iterations due to goal labels). Both decrease steadily for higher values of k.

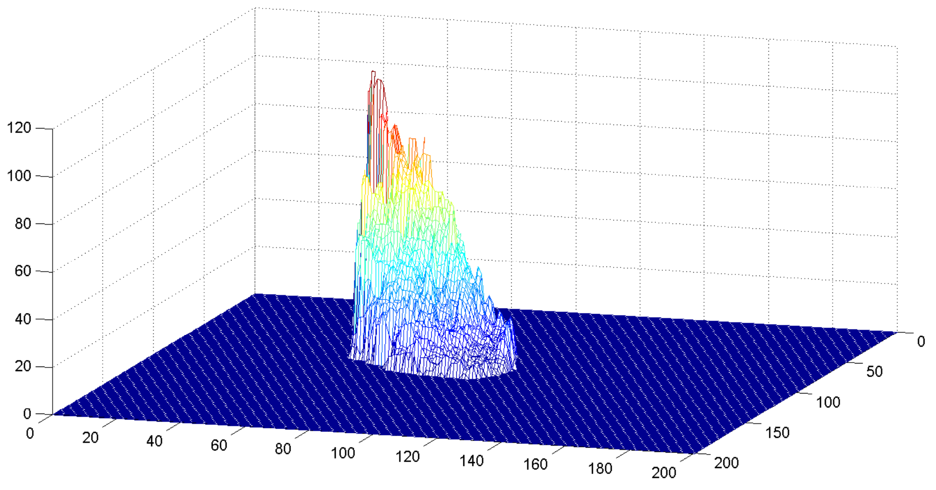
Figure 7.25(a) shows a visual representation of the search space explored by running $NAMOA^{**}$ with H^0_{KDLS}, while figure 7.25(b) shows the behaviour with H^5_{KDLS}. The number of label expansions per node are depicted for our sample problem. The search with $\vec{h}_{TC}$ is directed to the solution but the heuristic information available with $k = 0$ is not able to discard many labels. Nevertheless, providing a multivalued heuristic like H^5_{KDLS} allows to discard much more labels.

The third column of table 7.3 further displays the total number of dominated label expansions, which can be attributed exclusively to the inconsistency of H^k_{KDLS}. All these considerations clearly confirm that heuristic inconsistency is not an important concern in this case.

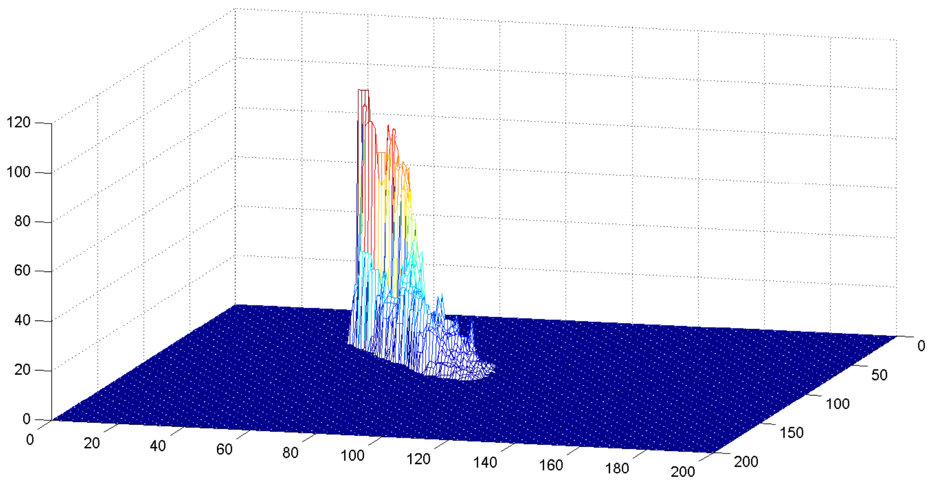
A second hypothesis could be that the extra dominated labels are increasing the number of pruning dominance checks. Again, the last column of table 7.3 suggests this is not a real cause of concern.

Finally, we could expect a time overhead due to an increase in filtering dominance checks. This hypothesis is supported by the great increase in filtering checks reported in the fourth column of table 7.3 for increasing values of k. This could be due to a double case. On one hand, the increase in the cardinality of $H(n)$ would result in an increase in the cardinality of the $F(n)$ sets, that need to be filtered each time a new solution is found. On the other hand, the improved heuristics could be leading the algorithm much quicker to solutions, boosting an effect similar to the one described in previous chapters.

Let us now consider the cardinality of H^k_{KDLS}. Figure 7.26 shows the set of Pareto-optimal costs for the same sample problem. This would be the value of $H^*(s) = C^*$. The figure also shows the different heuristic vectors obtained for H^k_{KDLS} for this node. Several phenomena can be observed. In the first place, the heuristic generated for $k = 5$ is almost the same as the one for $k = \infty$. In the second place, the number of supported solutions obtained with $k = \infty$ (23) is quite low compared to the actual cardinality of the set C^* (121). Nevertheless, the approximations obtained with $k \geq 4$ are pretty



(a) Number of label expansions for $NAMOA^{**}$ with H^0_{KDLS}



(b) Number of label expansions for $NAMOA^{**}$ with H^5_{KDLS}

Figure 7.25: Number of label expansions of $NAMOA^{**}$ using H_{KDLS} heuristics on a sample 200×200 class II grid problem instance with $\rho = 0$. Start node is (100,100) and goal node is (50,50).

	C^*	$KDLS$ ∞	$KDLS$ 5	$KDLS$ 4	$KDLS$ 3	$KDLS$ 2	$KDLS$ 1	$KDLS$ 0
Prob. 91	105	25	25	15	8	4	2	1
Prob. 92	143	20	20	15	8	4	2	1
Prob. 93	147	24	23	16	8	4	2	1
Prob. 94	100	23	23	16	8	4	2	1
Prob. 95	140	27	26	16	8	4	2	1
Prob. 96	106	18	18	15	8	4	2	1
Prob. 97	132	22	22	15	8	4	2	1
Prob. 98	129	23	23	16	8	4	2	1
Prob. 99	125	29	27	16	8	4	2	1
Prob. 100	121	23	22	15	8	4	2	1

Table 7.4: Size of C^* set and number of heuristic vectors obtained for start node (100,100) by $KDLS$ for the ten largest class II grid problem instances of size 200×200.

good. This is confirmed also by the results in table 7.4.

Figure 7.27(a) shows the distribution of the number of heuristic values in $H(n)$ for all nodes in this sample problem for $k = 5$. While this value can be high, search is constrained to a small strip of nodes aimed to the goal (see figure 7.25(b)). For these relevant nodes, the cardinality of $H(n)$ quickly decreases.

Figure 7.27(b) displays the product of the number of nondominated labels considered by $k = 5$ for each node by the cardinality of the heuristic set for that node. This is an upper bound on the cardinality of the $F(n)$ sets. This reveals an increase in the size of this sets, though only moderately significant (over 300 up to approx. 600) only for nodes at mid distance from source to goal.

Finally, figure 7.28 also confirms that in general $NAMOA^{**}$ with H^k_{KDLS} finds solutions faster for higher values of k.

In conclusion, the increase in the time performance of $NAMOA^{**}$ can be attributed to a combination of all these factors.

7.5 Conclusions

In this chapter the use of multiple heuristic vector values in $H(n)$ has been investigated. A heuristic precalculation procedure named $KDLS$ has been proposed. The procedure is parameterized with a value k and generates the $\vec{h}_{TC}$ heuristic as a base case ($k = 0$). Larger values of k generate equally or more informed heuristics. The procedure performs single-objective searches in a reversed graph to determine a subset of supported nondominated solutions from the goal node to other nodes. Heuristic vectors are then generated from these nondominated solutions. In the best case ($k = \infty$) the procedure terminates when all supported nondominated solutions to the start node have been generated.

Several properties of the heuristics H^k_{KDLS} generated by $KDLS$ have been analyzed. The heuristics are admissible, but can be inconsistent in general. This leads us to use $NAMOA^{**}$, the multiobjective pathmax version of $NAMOA^*$, as the multiobjective algorithm of choice.

The effectiveness of $NAMOA^{**}$ with H^k_{KDLS} has been investigated for two different domains: random grids and route planning problems in road maps.

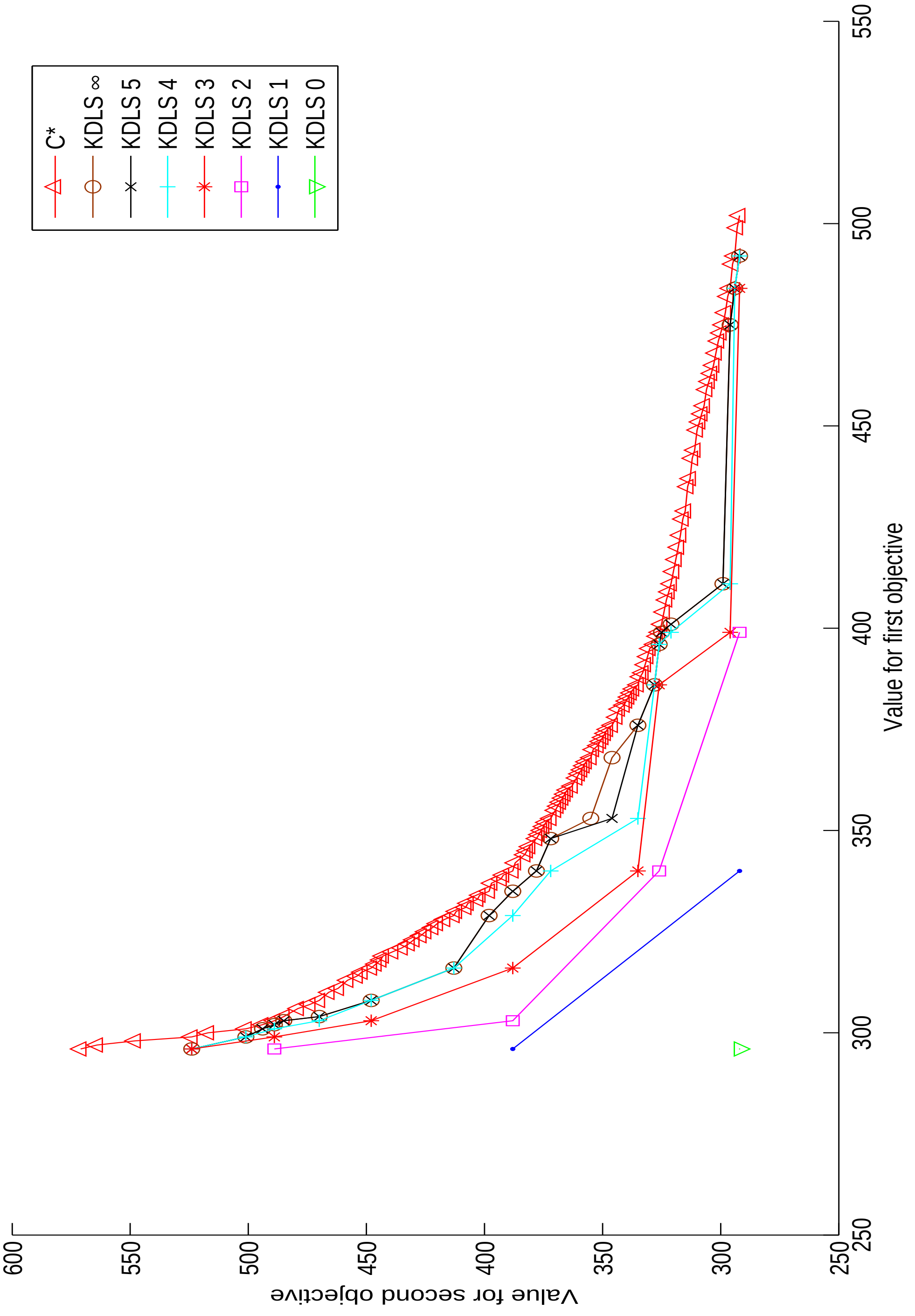
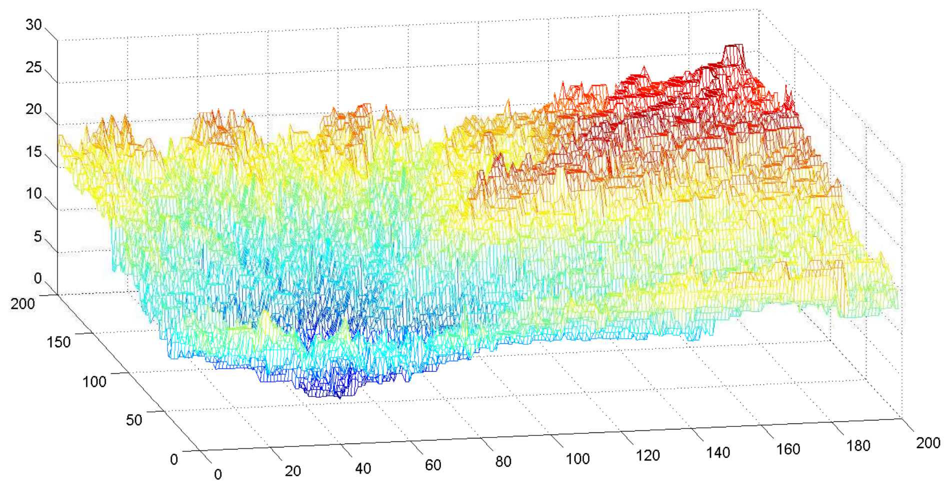
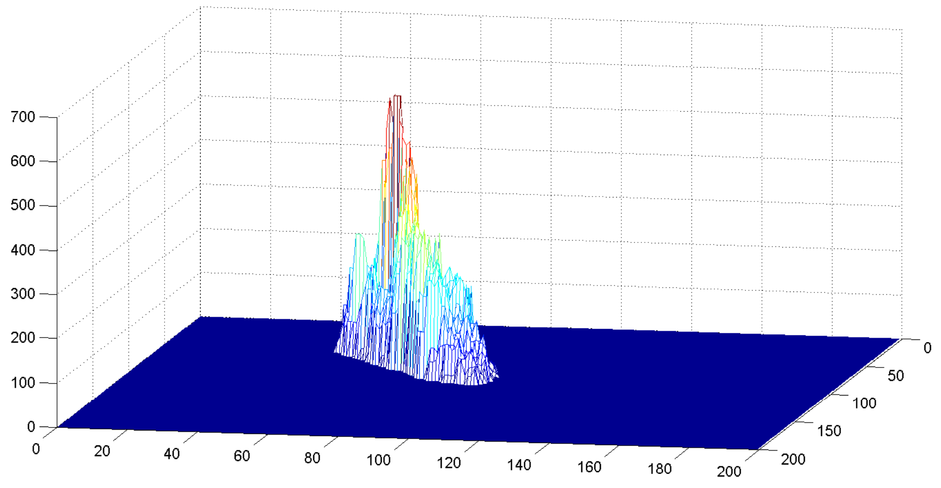


Figure 7.26: Heuristic vectors obtained for start node (100,100) by $KDLS$ on a sample class II grid problem instance of size 200 × 200



(a) Heuristic vectors obtained by $KDLS$ with $k = 5$ for all nodes



(b) Number of vectors in $F(n)$ for $KDLS$ with $k = 5$ for all nodes

Figure 7.27: Number of vectors in $H(n)$ and $F(n)$ for $NAMOA^{**}$ using H_{KDLS} heuristics on a sample 200×200 class II grid problem instance with $\rho = 0$. Start node is (100,100) and goal node is (50,50).

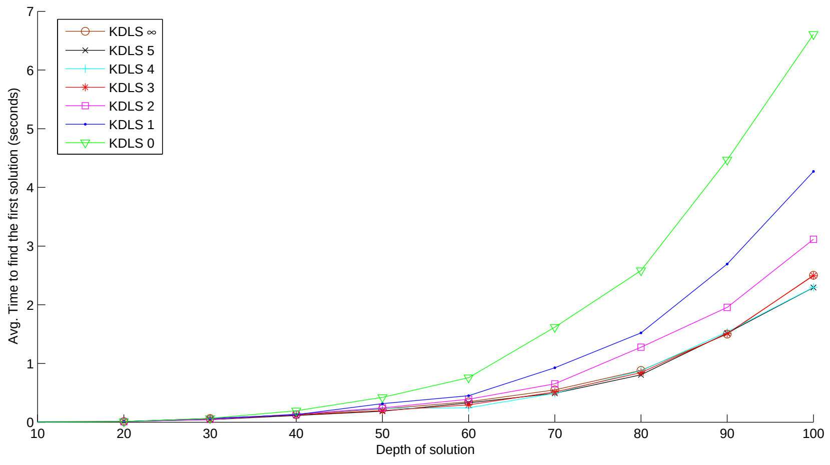


Figure 7.28: Time to find the first solution for $NAMOA^{**}$ with H^k_{KDLS} in class II grid problems with $\rho = 0$

Results indicate that important space savings can be achieved by H^k_{KDLS} over $\vec{h}_{TC}$. The improvement is higher as the difficulty of the problem increases in both domains. However, it is observed that in the route planning domain, more difficult problems do not necessarily result in larger solution graphs. This is attributed to the particular structure of the road map. In most cases analyzed, results for values of $k = 5$ and $k = \infty$ are very similar.

Regarding time requirements, run time increases in general for larger values of k in random grids, while results are mixed for route planning instances. This can be attributed again to the particular topology of road maps. In general, values of $k = 1$ or $k = 2$ offer a significant reduction in space requirements with low time overhead (or even a certain speedup in some route planning instances).

A deeper analysis on the time requirements of $NAMOA^{**}$ with H^k_{KDLS} reveal that the increase in time can be due to a number of causes. Interestingly, the inconsistency of heuristic information does not seem to result in a significant number of dominated label expansions. However, though multiobjective search time can be reduced for any value of k, this is not proportional to the reduction observed in the number of iterations or label expansions. This is due to an increase in the number of filtering dominance checks. The main causes for this increase point to a faster determination of nondominated solutions, and to an increase in the number of vectors in the $F(n)$ sets that need to be checked. Due to this reasons, the time devoted by $KDLS$ to increase the precision of heuristics does not seem to be compensated by a significant reduction in multiobjective search time. However, this opens up the interesting possibility of paralellizing the independent single-objective searches performed by $KDLS$.

At the same time, the effectiveness observed in H^k_{KDLS} to obtain initial solutions suggests that combining precalculated heuristics with other multicriteria approaches that search for a single nondominated solution, like compromise search, could be a very fruitful line of future research.

Part III

Conclusions

This third part reviews the goals of this thesis, summarizes the conclusions drawn from formal and empirical analyses and collects the original contributions of this research work. Some lines for future improvements are also suggested.

- Chapter 8 sums up the conclusions gathered from this research work and suggests future lines of research

Chapter 8

Conclusions and Future Work

Many important optimization problems involve the consideration of multiple objectives at the same time. In this thesis we have considered the Multiobjective Shortest Path Problem, which is inherently a difficult problem to solve. The development of efficient techniques is of practical importance for current research in Artificial Intelligence and Operational Research.

Several algorithms for solving this type of problems have been proposed in the past thirty years. This thesis is concerned with exact techniques. In particular, we deal with best-first graph search algorithms that incorporate heuristic information to improve performance. This has been a very fruitful alternative in single-objective search, and can be reasonably expected to provide also good results in multiobjective search. The three main algorithms proposed in this sense are $NAMOA^*$, MOA^*, and the one by Tung and Chew (TC).

The general goal of this thesis is to deepen our understanding of multiobjective search and to complete an empirical and formal characterization of these algorithms. Empirical analyses have been performed over sets of randomly generated grid problems, as well as on realistic route planning scenarios. The former allow the evaluation of the algorithms under controlled conditions (like solution depth, or correlation between objectives), while the latter provides a necessary link to potential practical applications.

In summary, this thesis contributes the first systematic evaluation of heuristic multiobjective graph search algorithms. A first important conclusion is that MOA^* is not, in general, a reliable heuristic search algorithm, since its time performance can degrade considerably with more informed heuristic information. A second conclusion is that, except in particular cases, both heuristic algorithms $NAMOA^*$ and TC can outperform blind search in terms of space and time requirements. In particular, the precalculated heuristic $\vec{h}_{TC}$ proposed for TC can be also applied to $NAMOA^*$. Between both algorithms, $NAMOA^*$ seems to perform better in spite of the additional specialized label selection heuristic used by TC. In general, finding solutions early is found to work against the time efficiency of multiobjective heuristic search. Two improvements have been proposed regarding the application of precalculated heuristics to $NAMOA^*$. The first is a bounded calculation procedure for $\vec{h}_{TC}$, that reduces the search effort of the precalculation stage. The second one is a procedure that calculates more informed precalculated heuristics. Our analysis reveals that these always reduce significantly the space needs, but the management of additional heuristic estimates can increase search time over $\vec{h}_{TC}$ in certain cases, although remaining still very competitive with blind search. Finally, the evaluations performed in this thesis reveal that multiobjective heuristic search can be currently applied to offline route planning applications over relatively large road maps.

The rest of this chapter presents a more detailed description of the conclusions, and points out future lines of improvement.

8.1 Conclusions

The main conclusions of this research can be summarized as follows:

1. **Analysis of MOA^*.**

   Prior to this research, only a very limited empirical comparison between algorithms MOA^* and $NAMOA^*$ had been performed. This suggested MOA^* to be slightly faster at the expense of more space needs in simple problems. However, the systematic empirical evaluation presented in previous chapters reveals a quite different situation for much harder problems. When equipped with the powerful $\vec{h}_{TC}$ heuristic, the space requirements of MOA^* appear to be bounded by a constant factor over those of $NAMOA^*$. However, time requirements are clearly worse than even those of blind MOA^*. To our knowledge, this unexpected result had never been previously reported in the literature. A deep empirical and formal analysis has been performed to explain this behavior. In particular, a class of biobjective problems has been presented were the number of label expansions performed by MOA^* with *perfect* heuristic information grows cubically with solution depth, compared to a quadratic performance of $NAMOA^*$ under analogous conditions. This behavior has been formally related to the node-selection strategy used by MOA^*. In general, and contrary to what could be expected, the time requirements of MOA^* can become increasingly worse with more informed heuristics. Our analyses found MOA^* to be competitive only for simple *blind* search problems.

2. **Analysis of TC and the $\vec{h}_{TC}$ precalculated heuristic.**

   This thesis provides also the first systematic analysis of the TC algorithm, as well as of its companion precalculated heuristic $\vec{h}_{TC}$. This heuristic includes a single vector estimate for each node, and can be also applied to algorithms MOA^* and $NAMOA^*$. The first conclusion is that $\vec{h}_{TC}$ is a really powerful heuristic that drastically reduces the number of labels examined by TC. In general, the precalculation effort of $\vec{h}_{TC}$ is more than compensated by the reduction in multiobjective search effort, except in the simpler problems. This behavior over simple problems is due to the fact that $\vec{h}_{TC}$ has to be computed over the whole graph, regardless of the choice of initial and goal nodes.

   At the same time, the second specialized heuristic for label selection used by TC was found to lead the algorithm quickly to solutions. However, this was found to have the undesirable effect of slowing down algorithm performance, since each new label selected for expansion needs to be checked for dominance against the set of already found solutions.

Finally, a simple proof was presented to show that the $\vec{h}_{TC}$ heuristic is consistent.

3. **Analysis of $NAMOA^*$.**

   The analyses in this thesis reveal $NAMOA^*$ to be the most space efficient algorithm among those considered. This is in accordance with previous formal results. In fact, the use of $NAMOA^*$ in combination with the $\vec{h}_{TC}$ heuristic has been shown to beat TC for a small but significant margin in almost all situations considered, both in terms of space and time.

4. **Label selection strategies.**

   In the first place, as mentioned above, our analysis lead to the conclusion that label selection is a superior strategy for heuristic multiobjective search when compared to node selection.

   Regarding label selection, our analyses additionally reveal that the order of label selection in multiobjective search can have important influence in time performance. In particular, the heuristic label selection rule of algorithm TC was found to make it slower. Regarding $NAMOA^*$ a linear aggregate selection rule was found to be faster than the usual lexicographic selection rule. This is in accordance to similar tests performed over blind search. Our analysis traces down the cause of this behavior to the speed with which the algorithm finds the set of Pareto-optimal solutions. Finding solutions early slows down the algorithm.

5. **Improved calculation of $\vec{h}_{TC}$.**

   The formal properties of $NAMOA^*$ and, more precisely, the bounds on the set of nodes explored by the algorithm, allow us to devise a more efficient precalculation procedure for the $\vec{h}_{TC}$ heuristic. The new procedure avoids the need to examine all nodes in the graph in the precalculation stage. This makes the technique potentially useful for infinite graphs, and renders heuristic search generally more efficient than blind search also in the simpler problem instances.

6. **Better informed heuristics: procedure $KDLS$.**

   The successful application of $NAMOA^*$ with $\vec{h}_{TC}$ to randomly generated problems as well as to realistic road maps lead us to investigate the possibility of more informed multivalued precalculated heuristics.

   A precalculation procedure ($KDLS$) has been proposed. This is parameterized with a value k, such that the larger k, the more informed the resulting heuristic. The procedure calculates a subset (in the extreme, the full set) of nondominated supported solutions through a set of single-objective searches. These are then used to generate an admissible set of heuristic estimates for all nodes that will be potentially visited later by $NAMOA^*$.

   Our analysis reveals in the first place that such precalculated heuristic will be admissible but inconsistent in general. This appears to be a handicap, since consistency is an important formal property for algorithm efficiency in $NAMOA^*$ just as it is for A^*. However, our experience reveals that the search overhead induced by inconsistency is quite small, and is in fact compensated by the reduction in label expansions achieved by the heuristic. The goodness of the heuristic function calculated by $KDLS$ has been evaluated with algorithm $NAMOA^{**}$, which incorporates the pathmax strategy to $NAMOA^{*}$.

The new heuristic can significantly reduce the number of labels explored by $NAMOA^{**}$ and, in consequence, the space requirements of the algorithm. Regrettably, the management of multiple heuristic evaluations for each label can result in a time overhead for multiobjective search when compared to $\vec{h}_{TC}$, although still very competitive with blind search. In random grids, only for small values of k the algorithm is competitive in time with the new heuristic when compared to $\vec{h}_{TC}$, while providing important additional space savings. In route planning problems in road maps, analogous space reductions are obtained. Larger values of k increase over the run time with $\vec{h}_{TC}$ only in some cases, while in others, some reduction for certain values of k can be achieved in this scenario.

7. **Significance of multiobjective heuristic search.**

   Finally, the evaluations performed in this thesis lead us to conclude that heuristic search (in particular, the combination of $NAMOA^{*}$ with precalculated heuristics) makes multiobjective analysis practical for offline route planning applications. Random problems from road maps of increasing size up to 1,070,376 nodes and 2,712,798 arcs have been solved with reasonable resources. On the contrary, blind search techniques were clearly outperformed, and could solve only a subset of the easier problem instances.

8.2 Future Work

This research has clarified several pending issues in multiobjective heuristic search but, at the same time, has raised new questions and future lines of research. In particular, we believe the following issues deserve further investigation:

- This research has highlighted the importance of the label selection rule in time performance. The determination of an optimal label selection rule, or at least, when a given rule is better than others is therefore an important issue for further research.

- This research has also shown that using more precise heuristics will lead faster to solutions, increasing the number of dominance checks. This is even more important in the case of the heuristic functions obtained with $KDLS$. The development of efficient specialized data structures and procedures for dominance test could lead to important improvements in algorithm efficiency. In a similar way, the development of a lazy strategy in the calculation of the $F(n)$ sets could also lead to performance improvements when multiple vector estimates are available.

- With the rapid proliferation of multicore machines, research in parallelization is gaining much interest. Regarding this thesis, the heuristic precalculation method proposed in the $KDLS$ procedure seems a good candidate for parallelization. Interesting works in this sense are (Di Stefano et al., 2006; Tsaggouris & Zaroliagis, 2009).

- The performance of $NAMOA^{**}$ with H^k_{KDLS} heuristic functions should be compared against multiobjective vector frontier search, which has achieved very low space requirements at the cost of increasing time requirements. One important disadvantage of vector frontier search is that it is currently a blind search strategy. The combination of vector frontier search with heuristic search could result in a very powerful multiobjective search algorithm.

- This thesis deals with the multiobjective decision problem. Finding the set of *all* nondominated solutions gives rise to a number of efficiency problems, since any new label selected for expansion need to be checked against the set of all previously found solutions. However, there are other multicriteria techniques that seek a single Pareto-optimal solution, like compromise search (Galand, 2008; Sauvanet, 2011). The research in multiobjective heuristic search performed in this thesis could be extended to other multicriteria decision rules.

- Regarding the application of multiobjective heuristic search to route planning problems, there are also many possibilities for further improvements. For example, hierarchical single-objective techniques could be generalized to the multiobjective case in order to reduce the size of the graph to be searched. Multilevel graphs (Schulz et al., 2002) represent an interesting approach in this sense. Another example are contraction hierarchies (Geisberger et al., 2008), which could be also extended to multiobjective settings. This technique requires bidirectional search. Thus, in order to combine $NAMOA^*$ with contraction hierarchies, multiobjective bidirectional search should be additionally investigated.

- The extension of bidirectional multiobjective pathmax techniques to multiobjective search could be also investigated, as in the case of a recent single-objective search study (Felner et al., 2011).

- Finally, the identification of other potential domains of application for multiobjective search algorithms is also an important line of research.

Part IV

Appendix

Appendix A

Resumen

Muchos problemas reales requieren examinar un número exponencial de alternativas para encontrar la elección óptima. A este tipo de problemas se les llama de optimización combinatoria. Además, en problemas reales normalmente se evalúan múltiples magnitudes que presentan conflicto entre ellas. Cuando se optimizan múltiples *objetivos* simultáneamente, generalmente no existe un valor óptimo que satisfaga al mismo tiempo los requisitos para todos los criterios. Solucionar estos problemas combinatorios multiobjetivo deriva comúnmente en un gran conjunto de soluciones Pareto-óptimas, que definen los balances óptimos entre los objetivos considerados.

En esta tesis se considera uno de los problemas multiobjetivo más recurrentes: la búsqueda de *caminos más cortos* en un grafo, teniendo en cuenta múltiples objetivos al mismo tiempo. Se pueden señalar muchas aplicaciones prácticas de la búsqueda multiobjetivo en diferentes dominios: enrutamiento en redes multimedia (Clímaco et al., 2003), programación de satélites (Gabrel & Vanderpooten, 2002), problemas de transporte (Pallottino & Scutellà, 1998), enrutamiento en redes de ferrocarril (Müller-Hannemann & Weihe, 2006), planificación de rutas en redes de carreteras (Jozefowiez et al., 2008), vigilancia con robots (delle Fave et al., 2009) o planificación independiente del dominio (Refanidis & Vlahavas, 2003).

La planificación de rutas multiobjetivo sobre mapas de carretera realistas ha sido considerada como un escenario de aplicación potencial para los algoritmos y heurísticos multiobjetivo considerados en esta tesis. El transporte de materias peligrosas (Erkut et al., 2007), otro problema de enrutamiento multiobjetivo relacionado, ha sido también considerado como un escenario de aplicación potencial interesante.

Los métodos de optimización de un solo criterio son bien conocidos y han sido ampliamente estudiados. La Búsqueda Heurística permite la reducción de los requisitos de espacio y tiempo de estos métodos, explotando el uso de estimaciones de la distancia real al objetivo. Los problemas multiobjetivo son bastante más complejos que sus equivalentes de un solo objetivo y requieren métodos específicos. Éstos, van desde técnicas de solución exactas a otras aproximadas, que incluyen los métodos metaheurísticos aproximados comúnmente encontrados en la literatura. Esta tesis se ocupa de algoritmos exactos primero-el-mejor y, en particular, del uso de información heurística para mejorar su rendimiento.

Esta tesis contribuye análisis tanto formales como empíricos de algoritmos y heurísticos para búsqueda multiobjetivo. La caracterización formal de estos algoritmos es importante para el campo. Sin embargo, la evaluación empírica es también de gran importancia para la aplicación real de estos métodos. Se han utilizado diversas clases de problemas bien conocidos para probar su rendimiento, incluyendo escenarios realistas como los descritos más arriba.

Los resultados de esta tesis proporcionan una mejor comprensión de qué métodos de los disponibles son mejores en situaciones prácticas. Se presentan explicaciones formales y empíricas acerca de su comportamiento. Se muestra que la búsqueda heurística reduce considerablemente los requisitos de espacio y tiempo en la mayoría de las ocasiones. En particular, se presentan los primeros resultados sistemáticos mostrando las ventajas de la aplicación de heurísticos multiobjetivo precalculados. Esta tesis también aporta un método mejorado para el precálculo de los heurísticos, y explora la conveniencia de heurísticos precalculados más informados.

Los objetivos de la tesis se presentan en la sección A.1. Las contribuciones están resumidas en la sección A.2. Se puede encontrar un resumen de los capítulos de la tesis en la sección A.3. La sección A.4 resume las conclusiones extraídas de este trabajo de investigación. Finalmente, se proponen líneas de investigación futuras en la sección A.5.

A.1 Objetivos

La investigación realizada en esta tesis persigue varios objetivos:

Formalización teórica Uno de los objetivos de esta tesis es completar los análisis formales del rendimiento heurístico de MOA^*, siguiendo los desarrollos formales recientes obtenidos para $NAMOA^*$. Además, se proporciona una caracterización de los heurísticos TC.

Evaluación empírica Un segundo objetivo es realizar una comparación sistemática de los algoritmos en orden a determinar cuál se comporta mejor de acuerdo a distintos parámetros del problema, como la profundidad de la solución o la correlación entre objetivos. Además, se evalúa el rendimiento de la búsqueda multiobjetivo en dominios realistas de planificación de rutas. En cualquiera de los casos, se busca un entendimiento profundo de las causas de los comportamientos observados.

Efectividad de la búsqueda heurística Un tercer objetivo es establecer bajo qué condiciones la búsqueda heurística puede realmente mejorar el rendimiento de algoritmos primer-el-mejor multiobjetivo desde un punto de vista práctico.

Mejora de las técnicas conocidas Finalmente, otro objetivo consiste en el uso del conocimiento adquirido a través de los análisis formales y empíricos para explorar nuevas vías de mejora en el rendimiento de los algoritmos. Se presta especial atención a ciertas alternativas en la implementación del algoritmo, como el orden de selección de alternativas para la exploración.

A.2 Contribuciones

Análisis de MOA^* Esta tesis completa los análisis formales previos de MOA^*. Los análisis previos fueron incapaces de establecer claramente la importancia de heurísticos informados en el rendimiento del algoritmo. Se muestra que el número de expansiones de etiquetas de MOA^* puede ser mucho mayor con búsqueda heurística que con búsqueda ciega. De hecho, el rendimiento puede llegar a ser peor cuando se usan heurísticos consistentes más informados. Este fenómeno está relacionado formalmente con la estrategia de selección de nodos usada por MOA^*. Además, los resultados empíricos muestran que la situación puede aparecer fácilmente en la práctica. Como consecuencia, MOA^* puede ser descartado en general como alternativa adecuada para la búsqueda heurística multiobjetivo.

Análisis de TC Se proporciona una caracterización simple para mostrar que el heurístico precalculado ideado por Tung & Chew (1992) es consistente. La consistencia se ha identificado recientemente como una importante propiedad formal para la búsqueda heurística multiobjetivo (Mandow & Pérez de la Cruz, 2010a). Análisis empíricos sistemáticos muestran por primera vez que este heurístico puede mejorar considerablemente el rendimiento de ambos algoritmos TC y $NAMOA^*$ sobre la búsqueda ciega. Sin embargo, TC se comporta algo peor que $NAMOA^*$ a pesar del uso de un heurístico adicional para la selección de alternativas. Este fenómeno ha sido adecuadamente explicado: encontrar las soluciones pronto puede penalizar en tiempo a la búsqueda heurística.

Análisis de $NAMOA^*$ Se ha encontrado que el algoritmo $NAMOA^*$ es la aproximación más efectiva en la mayoría de las situaciones. Sin embargo, en ciertas situaciones se ha identificado que la búsqueda heurística puede representar una sobrecarga en el rendimiento en términos de tiempo. Son los casos en que la reducción del número de alternativas consideradas no compensa la penalización de tiempo asociada a encontrar pronto las soluciones, inherente a la búsqueda heurística. Más concretamente, esto se debe al número de chequeos de dominancia necesitados por el algoritmo. Además, las evaluaciones empíricas sugieren que una regla de selección lineal mejora significativamente el rendimiento en tiempo de $NAMOA^*$ cuando se compara con la tradicional lexicográfica.

Heurísticos precalculados El método de precálculo originalmente propuesto por Tung & Chew (1992) puede ser mejorado en varias maneras. El método original requiere el cálculo de una búsqueda uno-a-todos de un solo objetivo para cada objetivo considerado. Mostramos cómo las propiedades formales de $NAMOA^*$ permiten acotar los nodos que serán visitados por $NAMOA^*$. Esto elimina la necesidad de considerar en la fase de precálculo aquellos que nunca van a ser alcanzados en la fase de búsqueda multiobjetivo.

El heurístico precalculado TC original proporciona una sola estimación vectorial para cada nodo. $NAMOA^*$ acepta funciones heurísticas $H(n)$ generales con múltiples estimaciones vectoriales. Se presenta en esta tesis un nuevo método de calculo, llamado $KDLS$. Se pueden precalcular funciones heurísticas más informadas, con múltiples vectores, con $KDLS$. La precisión de este nuevo heurístico está determinada por un parámetro k. Valores de k más grandes producen heurísticos más informados pero, al mismo tiempo, requieren más esfuerzo de precálculo. En general, mejores heurísticos reducen considerablemente los requisitos de espacio de $NAMOA^*$. Sin embargo, los requisitos de tiempo se incrementan de forma uniforme en mallas aleatorias, y sólo para los valores de k más pequeños son competitivos con el enfoque original en este sentido. No obstante, en problemas de planificación de rutas los heurísticos multivaluados pueden ofrecer ahorro tanto en tiempo como en espacio para algunas instancias.

Aplicaciones en escenarios realistas La aplicación de $NAMOA^*$ con heurísticos informados ha demostrado ser competitiva con las aproximaciones del estado del arte en búsqueda multiobjetivo. Se han considerado diferentes bancos de prueba para planificación de rutas en mapas de carreteras, un área de aplicación potencial. Se han resuelto de forma exacta con los recursos disponibles las combinaciones de objetivos tiempo/distancia y la más compleja tiempo/coste económico sobre mapas de carreteras grandes.

A.3 Resumen de los capítulos de la Tesis

A.3.1 Búsqueda Multiobjetivo en Grafos: Problemas y Algoritmos

El capítulo 2 proporciona una visión general del campo y delimita el marco en el que se mueven los algoritmos estudiados en esta tesis. Posteriormente, se describen adecuadamente los tres algoritmos $NAMOA^*$, MOA^* y TC, así como funciones heurísticas multiobjetivo generales bien definidas, propuestas por Tung & Chew (1992). El caso de heurísticos inconsistentes es también tratado con cierto detalle, con la regla pathmax y el algoritmo $NAMOA^{**}$.

El estado de desarrollo formal y empírico varía de uno a otro algoritmo. Mientras el desarrollo formal de $NAMOA^*$ se había completado (Mandow & Pérez de la Cruz, 2005, 2006, 2010a), algunas cuestiones todavía requerían adecuada caracterización en el caso de MOA^* antes de esta tesis. En el caso de TC, no se completaron nunca demostraciones formales adecuadas.

Con respecto a la evaluación empírica, sólo se habían realizado previamente experimentos limitados sobre mallas cuadradas usando la distancia de Manhattan con MOA^* y $NAMOA^*$ (Mandow & Pérez de la Cruz, 2005). Los algoritmos MOA^* y TC nunca fueron probados empíricamente por sus respectivos autores.

Esta tesis completa los desarrollos formales de MOA^* y proporciona un marco formal también para TC. Con respecto a la evaluación empírica, se presentan análisis exhaustivos en capítulos posteriores, para los tres algoritmos tanto con búsqueda ciega como con búsqueda heurística.

A.3.2 Evaluación del Rendimiento de la Búsqueda Multiobjetivo

La primera parte del capítulo corresponde a una revisión de la literatura sobre bancos de prueba multiobjetivo. Tres tipos de bancos de pruebas han sido propuestos en la literatura: grafos aleatorios, mallas y mapas realistas.

La segunda parte del capítulo presenta los bancos de pruebas empíricos que se han usado en esta tesis, inspirados en los casos típicos encontrados en la literatura. Además, se han investigado nuevos escenarios.

Esta tesis usa una combinación de grafos generados aleatoriamente y de problemas realistas de planificación de rutas en la evaluación empírica. Cada uno es adecuado para evaluar diferentes aspectos de los algoritmos y heurísticos analizados en esta tesis. Esta parte del capítulo da una explicación detallada de las características de cada uno de los conjuntos de datos usados.

Los entornos generados artificialmente, como las mallas aleatorias, permiten la evaluación controlada del rendimiento con respecto a diferentes parámetros, como el número de nodos o la correlación entre objetivos. Se han usado dos clases de mallas: las del primer tipo son similares a las descritas en la literatura donde se busca de esquina a esquina, mientras que en el segundo tipo la búsqueda se comienza en el centro de la malla, por lo que presentan su propia dificultad al no estar la búsqueda restringida por los límites de la malla.

Para los conjuntos de datos realistas, se han incluido diferentes escenarios: mapas de carreteras multiobjetivo de gran tamaño, problemas multiobjetivo difíciles mediante objetivos no correlacionados y problemas de transporte de materias peligrosas (hazmat) con más de dos objetivos. La combinación de todos estos escenarios y la dimensión de los parámetros evaluados son suficientes para considerar los casos de prueba como signicativos entre los disponibles hasta el momento de presentación de estos resultados.

Además de factores dependientes del problema, hay otros factores dependientes de la implementación, destacando entre ellos la estrategia de selección de alternativas abiertas. Entre las diferentes posibilidades, el orden lexicográfico es el más usual, pero existen otras alternativas, por ejemplo reglas lineales como la explicada en la sección 2.2.1 o la usada por el algoritmo TC (ver sección 2.4.3). La evaluación de órdenes alternativos ha sido considerada recientemente por Iori et al. (2010) como un detalle importante que afecta al rendimiento. Además, en la literatura, algunos otros autores han analizado diferentes estrategias para el manejo de las alternativas abiertas (Brumbaugh-Smith & Shier, 1989; Guerriero & Musmanno, 2001; Paixão & Santos, 2007). También se comentan en este capítulo otros detalles singulares con respecto a la implementación que son considerados importantes a la hora de comparar el rendimiento en tiempo de los algoritmos.

A.3.3 Análisis Formal de Algoritmos Multiobjetivo

Este capítulo aborda el análisis formal de los tres algoritmos heurísticos multiobjetivo presentados en el capítulo 2. Los resultados presentados en las secciones 4.5, 4.6 completan el estudio formal del algoritmo MOA^*. Stewart & White (1991) presentaron importantes propiedades para este algoritmo, similares a las encontradas para A^*. Sin embargo, este capítulo confirma algo que los autores mismos reconocieron, que la monotonicidad y consistencia no son tan importantes para MOA^* como en el caso de un solo objetivo. Este capítulo muestra que heurísticos más informados no llevan a un rendimiento mejor en MOA^*, aunque sean consistentes, tal como sí ocurre en $NAMOA^*$.

Estos resultados teóricos son importantes pero no responden completamente a la cuestión de qué algoritmo es mejor en la práctica. Conforme a la búsqueda ciega, se ha estudiado por otros autores la conveniencia de estrategias de selección de nodos en el caso de algoritmos de corrección de etiquetas (label-correcting) ciegos (Guerriero & Musmanno, 2001; Raith & Ehrgott, 2009). A pesar de que téoricamente se espera que el rendimiento obtenido en MOA^* debe ser menor que el obtenido para $NAMOA^*$, este punto tiene que ser confirmado por un estudio empírico que confirme en efecto si esto ocurre en todas las situaciones. Aparentemente, la estrategia de selección de nodos de MOA^* puede compensar en algunas situaciones particulares con respecto al número mayor de etiquetas consideradas.

Además, los heurísticos TC no han sido evaluados en la práctica. Este capítulo ha demostrado formalmente que el heurístico $\vec{h}_{TC}$ presentado por Tung & Chew (1992) es consistente, por lo que se espera que el rendimiento de $NAMOA^*$ pueda verse beneficiado por el uso de estos heurísticos consistentes que están bien informados. El capítulo 5 presenta un análisis empírico detallado de los tres algoritmos, tanto con búsqueda ciega como con los heurísticos TC en el caso heurístico.

A.3.4 Análisis Empírico de Algoritmos Multiobjetivo

Este capítulo presenta la primera comparación sistemática en términos de espacio y tiempo de los tres algoritmos de búsqueda heurística multiobjetivo primero el mejor (label setting) $NAMOA^*$, MOA^* y TC, usando orden lexicográfico en la selección de alternativas no dominadas, y aplicando el preciso heurístico H_{TC} propuesto por Tung & Chew (1992) también a MOA^* y $NAMOA^*$. Se han realizado comparaciones por pares a través de amplias evaluaciones sobre diferentes dominios de prueba.

Tal como se esperaba de los análisis téoricos, $NAMOA^*$ presenta mejor rendimiento en términos de espacio. Además, se han observado diversos fenómenos que no habían sido previamente señalados en la literatura. En primer lugar, el uso de información heurística en las estrategias de selección puede deteriorar el rendimiento en términos de tiempo. Este efecto incrementa con el grado de correlación entre objetivos. En particular, la precisa estrategia heurística usada por TC parece jugar en contra de su rendimiento en tiempo. La estrategia de selección de nodos de MOA^* se ve especialmente penalizada en tiempo por el uso del heurístico H_{TC}. Por tanto, podemos concluir que $NAMOA^*$ se comporta mejor que los otros algoritmos en la mayoría de los casos, especialmente para los problemas más difíciles, aquellos con soluciones más profundas y correlación negativa entre los objetivos.

En resumen, pueden extraerse algunas conclusiones del análisis empírico:

- MOA^* ciego puede ser más rápido que $NAMOA^*$ ciego en algunos casos (alta correlación o profundidad de solución pequeña). En otro caso, $NAMOA^*$ ciego supera ampliamente a MOA^* ciego. La sobrecarga en memoria de MOA^* se estabiliza en torno a un 15-25% en los peores casos analizados.

- El uso de información heurística puede derivar en ahorros importantes en cuanto a espacio y tiempo en $NAMOA^*$. Sin embargo, en algunos casos concretos como problemas de mallas clase I (búsqueda de esquina a esquina) con baja correlación, la ventaja en espacio puede ser mínima y el rendimiento en tiempo puede degradarse. En esos casos, la reducción en el número de etiquetas exploradas no parece compensar la sobrecarga en tiempo asociada a encontrar las soluciones antes.

- El uso de información heurística ahorra espacio en MOA^*, pero degrada considerablemente el rendimiento en tiempo en todos los casos. Según el conocimiento del autor, este resultado no había sido señalado nunca antes en la literatura, pero está de acuerdo con el análisis formal presentado en el capítulo 4.

- $NAMOA^*$ se comporta algo mejor que TC. Los requisitos de espacio son similares pero el heurístico particular h_{mix} penaliza el rendimiento en tiempo del algoritmo TC.

- $NAMOA^*$ heurístico se comporta mejor que los otros algoritmos en la mayoría de los casos, especialmente para los problemas más complicados (aquellos con soluciones más profundas y correlación negativa entre objetivos)

- Se ha descubierto que la velocidad de los algoritmos está relacionada con el ritmo de descubrimiento de soluciones no dominadas. Los algoritmos que encuentran las soluciones más tarde se comportan sistemáticamente de forma más rápida. En este sentido, el uso de una estrategia heurística especializada en el algoritmo TC juega en contra de la eficiencia temporal del algoritmo.

- Los resultados indican que el orden de selección entre etiquetas no dominadas abiertas es un elemento importante en la eficiencia temporal, y sugieren que la investigación de ordenes alternativos que combinen la búsqueda heúristica y la expansión retardada de soluciones podría llevar a algoritmos más eficientes. Esto se analiza en el siguiente capítulo, donde el orden lineal es evaluado en $NAMOA^*$, y comparado con el orden lexicográfico.

- Los experimentos confirman que el tiempo necesario para calcular los heurísticos no es significativo comparado con el tiempo de ejecución total en el caso de problemas clase I. Sin embargo, debe ser mejorado en el caso de problemas clase II, ya que puede representar un tiempo significativo para algunas instancias de problemas.

- Los problemas sobre mapas de carreteras analizados en este capítulo tienen requisitos de tiempo similares a problemas sobre mallas con menor profundidad de solución. En el capítulo 6 se usan mapas de carreteras de gran tamaño, análogos en dificultad a los problemas clase II. Para la búsqueda multiobjetivo en estos mapas, se hace necesaria una mejora en el método de cálculo de los heurísticos.

A.3.5 Búsqueda Heurística Multiobjetivo en Mapas de Carreteras

Una serie de técnicas han demostrado ser muy efectivas en aplicaciones de planificación de rutas en tiempo real para el caso de un solo objetivo, como la búsqueda en sistemas de navegación para coches. La extensión de estas técnicas a entornos multiobjetivo es un área importante de investigación. Sin embargo, los resultados presentados en este capítulo sugieren que las técnicas multiobjetivo ciegas basadas en extensos precálculos es probable que experimenten dificultades prácticas en grandes mapas, como ya fue indicado por Delling & Wagner (2009).

De la evaluación sistemática de varios parámetros realizada en este capítulo, pueden extraerse algunas conclusiones sobre los algoritmos de búsqueda heurística multiobjetivo,

- El algoritmo TC demuestra ser constantemente peor en tiempo bajo similares condiciones que $NAMOA^*$ en problemas realistas, confirmando los resultados sobre problemas en mallas. Además, es importante recalcar que el algoritmo TC puede que no elimine algunos miembros dominados de OPEN (ABIERTOS) e introducir algunos en COSTS (ver sección 2.4.3). Por tanto, $NAMOA^*$ emerge como el algoritmo heurístico multiobjetivo del estado del arte.

- Este capítulo informe sobre la aplicación con éxito de $NAMOA^*$ a instancias de problemas aleatorias en mapas de carreteras realistas de gran tamaño. Además, se han considerado problemas multiobjetivo difíciles sobre mapas de carreteras realistas con objetivos no correlacionados. Sin embargo, los requisitos de tiempo son sólo razonables para aplicaciones de planificación de rutas offline.

- Se ha presentado en este capítulo un método mejorado para el cálculo del heurístico $\vec{h}_{TC}$. Este método aprovecha las propiedades formales de $NAMOA^*$ para reducir el esfuerzo del precálculo, reduciendo significativamente los precálculos en muchas de las instancias de problema. La combinación de $NAMOA^*$ y el heurístico $\vec{h}_{TC}$ con el nuevo método de cálculo acotado, claramente supera de forma amplia a un heurístico de distancia clásico y a la búsqueda ciega, tanto en el número de problemas resueltos como en el rendimiento obtenido en tiempo.

- El orden lineal en $NAMOA^*$ parece combinar la búsqueda heurística y la expansión retardada de soluciones, llevando a un algoritmo más eficiente. La regla de selección lineal es considerada en los sucesivos capítulos para $NAMOA^*$.

- Se ha investigado el impacto del heurístico $\vec{h}_{TC}$ en otros dominios de problemas prácticos. La aplicación del heurístico $\vec{h}_{TC}$ a los problemas de tranporte de materias peligrosas ha superado ampliamente a aproximaciones previas ciegas.

- Finalmente, el impacto en el número de objetivos en el rendimiento de los algoritmos se ha evaluado con una comparación entre $NAMOA^*$ con dos objetivos y tres objetivos. El número de caminos Pareto-óptimos cae a medida que la correlación entre objetivos aumenta. Por tanto, el tiempo necesario para resolver un problema multiobjetivo depende de la naturaleza específica de los costes en los arcos. En problemas con objetivos relativamente no correlacionados, el número de caminos Pareto-óptimos incrementa con el número de objetivos considerados, mientras que añadir objetivos altamente correlacionados no degrada el rendimiento de la búsqueda.

El uso de heurísticos precalculados en la búsqueda multiobjetivo es un área no suficientemente explorada que necesita desarollos formales y prácticos. Las aplicaciones descritas en este capítulo podrían beneficiarse de heurísticos precalculados más informados. El capítulo 7 explora esta posibilidad, con heurísticos más eficientes para búsqueda multiobjetivo.

A.3.6 Heurísticos Multivaluados para Búsqueda Heurística Multiobjetivo

En este capítulo, se ha investigado el uso de múltiples valores vectoriales heurísticos en $H(n)$. Se ha propuesto un procedimiento de precálculo de heurísticos llamado $KDLS$. El procedimiento está parametrizado con un valor k y genera el heurístico $\vec{h}_{TC}$ como caso base ($k = 0$). Valores más grandes de k generan heurísticos igual o mejor informados. El procedimiento realiza búsquedas de un solo objetivo en un grafo invertido para determinar un subconjunto de las soluciones soportadas no dominadas del nodo objetivo a los otros nodos. Los vectores heurísticos se generan a partir de estas soluciones no dominadas. En el mejor caso ($k = \infty$), el procedimiento termina cuando se han generado todas las soluciones soportadas no dominadas al nodo de inicio.

Varias propiedades de los heurísticos H^k_{KDLS} generados por $KDLS$ han sido analizadas. Los heurísticos son admisibles pero, en general, pueden ser inconsistentes. Esto nos lleva a usar $NAMOA^{**}$, la version de $NAMOA^{*}$ con pathmax multiobjetivo, como el algoritmo multiobjetivo a elegir.

Se ha investigado la efectividad de $NAMOA^{**}$ con H^k_{KDLS} para diferentes dominios: mallas aleatorias y problemas de planificación de rutas en mapas de carreteras.

Los resultados indican que se pueden conseguir importantes ahorros de espacio con H^k_{KDLS} con respecto a $\vec{h}_{TC}$. La mejora es mayor a medida que la dificultad del problema incrementa en ambos dominios. Sin embargo, se observa que, en el dominio de planificación de rutas, problemas más difíciles no necesariamente producen grafos solución más grandes. Esto es atribuido a la estructura particular del mapa de carreteras. En la mayoría de casos analizados, los resultados son muy similares para valores de $k = 5$ y $k = \infty$.

Respecto a los requisitos de tiempo, el tiempo de ejecución aumenta en general para valores más grandes de k en mallas aleatorias, mientras que los resultados están mezclados para instancias de planificación de rutas. Esto es atribuido de nuevo a la topología particular de los mapas de carreteras. En general, los valores $k = 1$ o $k = 2$ ofrecen una reducción significativa en los requisitos de espacio con una sobrecarga de tiempo pequeña (o incluso una cierta aceleración en algunas instancias de planificación de rutas).

Un análisis más profundo en los requisitos de tiempo de $NAMOA^{**}$ con H^k_{KDLS} revela que el incremento en tiempo puede ser debido a una serie de causas. En referencia a esto, la inconsistencia en la información heurística no parece producir un número significativo de expansiones de etiquetas dominadas. Sin embargo, aunque el tiempo de búsqueda multiobjetivo puede ser reducido para cualquier valor de k, no es proporcional a la reducción observada en el número de iteraciones o expansiones de etiquetas. Esto es debido a un incremento en el número de chequeos de dominancia debidos a filtrado. Las causas principales de este incremento apuntan a una determinación más rápida de las soluciones no dominadas, y a un incremento en el número de vectores en los conjuntos $F(n)$ que necesitan ser chequeados. Debido a estas razones, el tiempo dedicado por $KDLS$ a incrementar la precisión de los heurísticos no parece compensar con una reducción significativa en el tiempo de búsqueda multiobjetivo. Sin embargo, esto abre la interesante posibilidad de paralelizar las búsquedas de un solo objetivo independientes realizadas por $KDLS$.

Al mismo tiempo, la efectividad observada en H^k_{KDLS} para obtener las soluciones iniciales sugiere que combinar heurísticos precalculados con otras aproximaciones multicriterio que buscan una sola solución no dominada, como la búsqueda compromiso, puede constituir una línea futura de investigación muy fructífera.

A.4 Conclusiones

De la investigación realizada en esta tesis se pueden extraer una serie de conclusiones:

1. **Análisis de MOA^*.**

   Antes de este trabajo de investigación, sólo se había realizado una comparación empírica muy limitada de los algoritmos MOA^* y $NAMOA^*$. Ésta, sugería que MOA^* era ligeramente más rápido en problemas simples a cambio de mayores requisitos de espacio. Sin embargo, la evaluación empírica sistemática, llevada a cabo en capítulos anteriores, revela una situación muy diferente para problemas más difíciles. Cuando está equipado con el potente heurístico $\vec{h}_{TC}$, los requisitos de MOA^* parecen estar acotados por un factor constante respecto a aquellos de $NAMOA^*$. Sin embargo, los requisitos de tiempo son claramente peores incluso que los de la búsqueda ciega con MOA^*. Según nuestro conocimiento, estos resultados inesperados no habían sido señalados nunca previamente en la literatura. Se ha realizado un análisis profundo, tanto formal como empírico, para explicar este comportamiento. En particular, se presenta una clase de problemas biobjetivo donde el número de expansiones de etiquetas realizadas por MOA^* con información heurística *perfecta* crece de forma cúbica con la profundidad de la solución cuando, en condiciones análogas, crece sólo de forma cuadrática en $NAMOA^*$. Este comportamiento ha sido relacionado formalmente con la estrategia de selección de nodos usada por MOA^*. En general, y contrariamente a lo que podría esperarse, los requisitos de tiempo de MOA^* pueden ser cada vez peores con heurísticos más informados. Los análisis encuentran que MOA^* sólo es competitivo para problemas de búsqueda simples con búsqueda ciega.

2. **Análisis de TC y el heurístico precalculado $\vec{h}_{TC}$.**

   Esta tesis proporciona también el primer análisis sistemático del algoritmo TC, así como de su heurístico compañero precalculado $\vec{h}_{TC}$. Este heurístico incluye una estimación de un sólo vector para cada nodo, y puede ser aplicado también a los algoritmos MOA^* y $NAMOA^*$. La primera conclusión es que $\vec{h}_{TC}$ es realmente un heurístico potente que reduce drásticamente el número de etiquetas examinadas por TC. En general, el esfuerzo de precálculo de $\vec{h}_{TC}$ está más que compensado por la reducción en el esfuerzo de la búsqueda multiobjetivo, excepto para problemas simples. Este comportamiento sobre problemas simples, es debido al hecho de que $\vec{h}_{TC}$ debe ser calculado sobre el grafo entero, independientemente de la elección de nodos inicial y final.

Al mismo tiempo, se ha encontrado que el segundo heurístico usado por TC, especializado para selección de etiquetas, lleva al algoritmo rápidamente a las soluciones. Sin embargo, esto tiene el efecto no deseado de ralentizar el rendimiento del algoritmo, ya que cada nueva etiqueta seleccionada para expansión debe ser chequeada para ver si es dominada por alguna de las soluciones ya encontradas. Finalmente, se presenta una simple demostración de que el heurístico $\vec{h}_{TC}$ es consistente.

3. **Análisis de $NAMOA^*$.**

   Los análisis de esta tesis revelan que $NAMOA^*$ es el algoritmo más eficiente en términos de espacio entre los considerados. Esto está en concordancia con resultados formales previos. De hecho, el uso de $NAMOA^*$ en combinación con el heurístico $\vec{h}_{TC}$ derrota al algoritmo TC por un pequeño pero significativo margen en casi todas las situaciones consideradas, tanto en términos de espacio como de tiempo.

4. **Estrategias de selección de etiquetas.**

   En primer lugar, como se menciona más arriba, los análisis llevan a la conclusión de que la selección de etiquetas es una estrategia mejor que la selección de nodos para la búsqueda multiobjetivo heurística.

   Con respecto a la selección de etiquetas, nuestros análisis revelan además que el orden de selección en la búsqueda multiobjetivo puede tener una influencia importante en el rendimiento en términos de tiempo. En particular, la regla de selección de etiquetas del algoritmo TC lo hace más lento. Con respecto a $NAMOA^*$, se ha encontrado que una regla de selección lineal es más rápida que la usual regla de selección lexicográfica. Esto está de acuerdo con tests similares realizados sobre búsqueda ciega. Nuestros análisis indican que la causa de este comportamiento está relacionada con la velocidad con que el algoritmo encuentra el conjunto de soluciones Pareto-óptimas. Encontrar pronto las soluciones hace más lento al algoritmo.

5. **Cálculo mejorado de $\vec{h}_{TC}$.**

   Las propiedades formales de $NAMOA^*$ y, más concretamente, las cotas a los nodos explorados por el algoritmo, nos permiten idear un procedimiento de precálculo del heurístico $\vec{h}_{TC}$ más eficiente. El nuevo procedimiento evita la necesidad de examinar todos los nodos del grafo en la fase de precálculo. Esto hace la técnica potencialmente útil para grafos infinitos, y hace la búsqueda heurística generalmente más eficiente que la búsqueda ciega también en las instancias de problema más sencillas.

6. **Mejores heurísticos precalculados: procedimiento $KDLS$.**

   La aplicación con éxito de $NAMOA^*$ con $\vec{h}_{TC}$ a problemas generados aleatoriamente así como mapas de carreteras realistas, nos lleva a investigar la posibilidad de heurísticos precalculados multivaluados más informados.

   Se ha propuesto un procedimiento de precálculo ($KDLS$). Esta parámetrizado con un valor k, tal que a mayor k, más informado es el heurístico resultante.

El procedimiento calcula un conjunto (en el extremo, el conjunto completo) de soluciones soportadas no dominadas a través de un conjunto de búsquedas de un solo objetivo. Éstas, son usadas para generar un conjunto de estimaciones heurísticas admisibles para todos los nodos que serán potencialmente visitados después por $NAMOA^*$.

Nuestros análisis revelan, en primer lugar, que tal heurístico precalculado será admisible pero inconsistente en general. Esto parece un handicap, pues la consistencia es una importante propiedad formal para la eficiencia del algoritmo $NAMOA^*$, como lo es en el caso de A^*. Sin embargo, la experiencia revela que la sobrecarga en la búsqueda inducida por la inconsistencia es realmente pequeña, y de hecho es compensada por la reducción en el número de expansiones conseguida por el heurístico. La bondad de las funciones heurísticas calculadas por $KDLS$ ha sido evaluada con el algoritmo $NAMOA^{**}$, que incorpora la estrategia pathmax al algoritmo $NAMOA^*$.

Este nuevo heurístico puede reducir significativamente el número de etiquetas exploradas por $NAMOA^{**}$ y, en consecuencia, los requisitos de espacio del algoritmo. Desafortunadamente, el manejo de múltiples evaluaciones heurísticas para cada etiqueta puede resultar en una sobrecarga de tiempo para la búsqueda multiobjetivo cuando se compara con $\vec{h}_{TC}$, aunque todavía muy competitiva con respecto a la búsqueda ciega. En mallas aleatorias, sólo para valores de k pequeños el algoritmo es competitivo en tiempo con el nuevo heurístico comparado con $\vec{h}_{TC}$, aunque proporciona importantes ahorros adicionales de espacio. En problemas de planificación de rutas en mapas de carreteras, se obtienen reducciones de espacio similares. Mayores valores de k incrementan el tiempo de ejecución con respecto a $\vec{h}_{TC}$ en algunos casos, aunque en otros se puede obtener reducción en tiempo también para ciertos valores de k en este escenario.

7. **Importancia de la búsqueda heurística multiobjetivo.**

   Finalmente, las evaluaciones realizadas en esta tesis llevan a la conclusión de que la búsqueda heurística (en particular la combinación de $NAMOA^*$ con heurísticos precalculados) hace el análisis multiobjetivo práctico para aplicaciones de planificación de rutas offline. Se han resuelto, con recursos razonables, problemas aleatorios sobre mapas de carreteras de hasta 1,070,376 de nodos y 2,712,798 de arcos. Por el contrario, las técnicas de búsqueda ciega son claramente sobrepasadas, y pudieron resolver sólamente un subconjunto de las instancias de problemas más fáciles.

A.5 Trabajo Futuro

Esta investigación ha clarificado varios problemas pendientes en la búsqueda heurística multiobjetivo pero, al mismo tiempo, ha planteado nuevas preguntas y futuras líneas de investigación. En particular, creemos que los siguientes asuntos merecen ser investigados después de esta tesis:

- Esta investigación ha señalado la importancia de la regla de selección de etiquetas en el rendimiento del algoritmo en términos de tiempo. La determinación de una regla de selección de etiquetas óptima, o al menos, de cuándo una regla dada es mejor que las otras es por tanto un tema importante de investigación futura.

- Esta investigación también ha mostrado que el uso de heurísticos más precisos puede llevar más rápidamente a las soluciones, incrementando el número de chequeos de dominancia. Esto es incluso más importante en el caso de las funciones heurísticas obtenidas con $KDLS$. El desarrollo de estructuras de datos especializadas y procedimientos para chequeo de dominancia eficientes puede llevar a importantes mejoras en el rendimiento del algoritmo. De manera similar, el desarrollo de una estrategia perezosa en el cálculo de los conjuntos $F(n)$ puede llevar a mejoras en el rendimiento, cuando hay disponibles múltiples estimaciones heurísticas.

- Con la rápida proliferación de máquinas multiprocesador, la investigación en paralelización está ganando interés. Con respecto a esta tesis, el método de precálculo de heurísticos propuesto en el procedimiento $KDLS$ parece un buen candidato para paralelización. Existen trabajos interesantes en este sentido como (Di Stefano et al., 2006; Tsaggouris & Zaroliagis, 2009).

- El rendimiento de $NAMOA^{**}$ con las funciones heurísticas H^k_{KDLS} debería ser comparado con el de la búsqueda frontera vectorial multiobjetivo, que consigue requisitos de espacio muy pequeños a costa de un incremento en los requisitos de tiempo. Una desventaja importante de la búsqueda frontera vectorial es que actualmente es una estrategia de búsqueda ciega. La combinación de búsqueda frontera vectorial con búsqueda heurística podría resultar en un algoritmo muy potente.

- Esta tesis trata con problemas de decisión multiobjetivo. Encontrar el conjunto de *todas* las soluciones no dominadas da lugar a problemas de eficiencia, dado que cada nueva etiqueta seleccionada para expansión necesita ser chequeada contra el conjunto de todas las soluciones previamente encontradas. Sin embargo, existen otras técnicas multicriterio que buscan una sola solución Pareto-óptima, como la búsqueda de soluciones compromiso (Galand, 2008; Sauvanet, 2011). La investigación en búsqueda heurística multiobjetivo realizada en esta tesis podría ser extendida a otras reglas de decisión multicriterio.

- Con respecto a la aplicación de búsqueda heurística multiobjetivo a problemas de planificación de rutas, existen también muchas posibilidades de mejoras posteriores. Por ejemplo, se podrían generalizar las ténicas jerárquicas de un solo objetivo al caso multiobjetivo para reducir el tamaño del grafo en el que se busca. Los grafos multinivel (Schulz et al., 2002) representan un aproximación interesante en este sentido. Otro ejemplo son las jerarquías de contracción (Geisberger et al., 2008), que también podrían ser extendidas a entornos multiobjetivo. Esta técnica requiere búsqueda bidireccional. Por tanto, para combinar $NAMOA^*$ con las jerarquías de contracción, además debería investigarse la búsqueda heurística bidireccional multiobjetivo.

- La extensión de técnicas pathmax bidireccionales al caso de búsqueda multiobjetivo podría ser también investigada, como en el caso de un reciente estudio para el caso de un solo objetivo (Felner et al., 2011).

- Finalmente, la identificación de otros dominios potenciales de aplicación para algoritmos de búsqueda multiobjetivo es una importante línea de investigación.

Bibliography

Abkowitz, M. & Cheng, P. D. (1988). Developing a risk/cost framework for routing truck movements of hazardous materials. *Accident Analysis & Prevention*, 20(1), 39–51.

Azevedo, J. & Martins, E. (1991). An algorithm for the multiobjective shortest path problem on acyclic networks. *Investigação Operacional*, 11(1), 52–69.

Bast, H., Funke, S., Matijevic, D., Sanders, P., & Schultes, D. (2007). In transit to constant time shortest-path queries in road networks. In *9th Workshop on Algorithm Engineering and Experiments, ALENEX 2007*: SIAM.

Batz, G. V., Delling, D., Sanders, P., & Vetter, C. (2009). Time-dependent contraction hierarchies. In *11th Workshop on Algorithm Engineering and Experiments, ALENEX 2009* (pp. 97–105).: SIAM.

Bauer, R., Columbus, T., Katz, B., Krug, M., & Wagner, D. (2010a). Preprocessing speed-up techniques is hard. In T. Calamoneri & J. Diaz (Eds.), *Algorithms and Complexity*, volume 6078 of *Lecture Notes in Computer Science* (pp. 359–370). Springer Berlin / Heidelberg.

Bauer, R. & Delling, D. (2009). SHARC: Fast and robust unidirectional routing. *Journal of Experimental Algorithmics*, 14, 4:2.4–4:2.29.

Bauer, R., Delling, D., Sanders, P., Schieferdecker, D., Schultes, D., & Wagner, D. (2010b). Combining hierarchical and goal-directed speed-up techniques for Dijkstra's algorithm. *Journal of Experimental Algorithmics*, 15, 2.3:2.1–2.3:2.31.

Bayili, S. & Polat, F. (2011). Limited-damage A*: A path search algorithm that considers damage as a feasibility criterion. *Knowledge-Based Systems*, 24(4), 501 – 512.

Bellman, R. (1954). The theory of dynamic programming. *Bulletin of American Mathematical Society*, 60, 503–515.

Bertsekas, D. P. (1991). *Linear network optimization: algorithms and codes*. Cambridge, MA, USA: MIT Press.

Bertsekas, D. P., Guerriero, F., & Musmanno, R. (1996). Parallel asynchronous label-correcting methods for shortest paths. *Journal of Optimization Theory and Applications*, 88(2), 297–320.

Brumbaugh-Smith, J. & Shier, D. (1989). An empirical investigation of some bicriterion shortest path problems. *European Journal of Operational Research*, 43, 216–224.

Bryce, D. (2012). Planning for multiple preferences versus planning with no preference. *ISRN Artificial Intelligence*, vol. 2012, Article ID 714245, 9 pages. DOI: 10.5402/2012/714245.

Caramia, M., Giordani, S., & Iovanella, A. (2010). On the selection of k routes in multiobjective hazmat route planning. *IMA Journal of Management Mathematics*, 21, 239–251.

Carlyle, W. M. & Wood, R. K. (2005). Near-shortest and k-shortest simple paths. *Networks*, 46(2), 98–109.

Cherkassky, B. V., Goldberg, A. V., & Radzik, T. (1996). Shortest paths algorithms: Theory and experimental evaluation. *Mathematical Programming*, 73(2), 129–174.

Clímaco, J. & Martins, E. (1981). On the determination of the nondominated paths in a multiobjective network problem. *Methods in Operations Research*, 40, 255–258.

Clímaco, J. & Martins, E. (1982). A bicriterion shortest path algorithm. *European Journal of Operational Research*, 11(4), 399 – 404.

Clímaco, J. C. N., Craveirinha, J. M. F., & Pascoal, M. M. B. (2003). A bicriterion approach for routing problems in multimedia networks. *Networks*, 41(4), 206–220.

Clímaco, J. C. N. & Pascoal, M. M. B. (2012). Multicriteria path and tree problems: discussion on exact algorithms and applications. *International Transactions in Operational Research*, 19(1-2), 63–98.

Coego, J., Mandow, L., & Pérez de la Cruz, J. (2012). A comparison of multiobjective depth-first algorithms. *Journal of Intelligent Manufacturing*, (pp. 1–9).

Coego, J., Mandow, L., & Pérez de la Cruz, J. L. (2009). A new approach to iterative deepening multiobjective A*. In *AI*IA 2009: Emergent Perspectives in Artificial Intelligence*, volume 5883 of *Lecture Notes in Computer Science* (pp. 264–273). Berlin, Heidelberg: Springer-Verlag.

Cohon, J. L. (1978). *Multiobjective programming and planning*, volume 140 of *Mathematics in Science and Engineering*. Academic Press, New York, (Dover Publications Inc.), Dover edition.

Cox, R. G. (1984). *Routing and scheduling of hazardous materials shipments: algorithmic approaches to managing spent nuclear fuel transport*. PhD thesis, Cornell University, Ithaca, NY.

Dasgupta, P., Chakrabarti, P., & DeSarkar, S. (1995). Utility of pathmax in partial order heuristic search. *Information Processing Letters*, 55, 317–322.

Dasgupta, P., Chakrabarti, P., & DeSarkar, S. (1999). *Multiobjective Heuristic Search*. Braunschweig/Wiesbaden: Vieweg.

de Lima Pinto, L., Bornstein, C. T., & Maculan, N. (2009). The tricriterion shortest path problem with at least two bottleneck objective functions. *European Journal of Operational Research*, 198(2), 387 – 391.

Dechter, R. & Pearl, J. (1985). Generalized best-first search strategies and the optimality of A*. *Journal of the ACM*, 32(3), 505–536.

delle Fave, F., Canu, S., Iocchi, L., Nardi, D., & Ziparo, V. (2009). Multi-objective multi-robot surveillance. In *4th International Conference on Autonomous Robots and Agents, ICARA 2009.* (pp. 68–73).

Delling, D., Goldberg, A., Pajor, T., & Werneck, R. (2011a). Customizable route planning. In P. Pardalos & S. Rebennack (Eds.), *10th International Symposium on Experimental Algorithms, SEA 2011*, volume 6630 of *Lecture Notes in Computer Science* (pp. 376–387). Springer Berlin / Heidelberg.

Delling, D., Goldberg, A. V., Razenshteyn, I., & Werneck, R. F. (2011b). Graph partitioning with natural cuts. In *25th IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2011* (pp. 1135–1146). Los Alamitos, CA, USA: IEEE Computer Society.

Delling, D., Sanders, P., Schultes, D., & Wagner, D. (2009). Engineering route planning algorithms. In *Algorithmics*, volume 5515 of *Lecture Notes in Computer Science* (pp. 117–139). Springer.

Delling, D. & Wagner, D. (2009). Pareto paths with SHARC. In *8th International Symposium on Experimental Algorithms, SEA'09*, volume 5526 of *Lecture Notes in Computer Science* (pp. 125–136). Berlin, Heidelberg: Springer-Verlag.

Dell'Olmo, P., Gentili, M., & Scozzari, A. (2005). On finding dissimilar Pareto-optimal paths. *European Journal of Operational Research*, 162, 70–82.

Di Stefano, G., Petricola, A., & Zaroliagis, C. (2006). On the implementation of parallel shortest path algorithms on a supercomputer. In M. Guo, L. Yang, B. Di Martino, H. Zima, J. Dongarra, & F. Tang (Eds.), *4th International Conference on Parallel and Distributed Processing and Applications, ISPA 2006*, volume 4330 of *Lecture Notes in Computer Science* (pp. 406–417). Springer Berlin / Heidelberg.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1, 269–271.

Ehrgott, M. (2005). *Multicriteria Optimization.* Springer, 2nd edition.

Ehrgott, M. & Gandibleux, X. (2000). A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum*, 22, 425–460.

Ehrgott, M. & Gandibleux, X. (2004). Approximative solution methods for multiobjective combinatorial optimization. *TOP*, 12, 1–63.

Erkut, E. & Ingolfsson, A. (2005). Transport risk models for hazardous materials: revisited. *Operations Research Letters*, 33(1), 81–89.

Erkut, E., Tjandra, S. A., & Verter, V. (2007). Hazardous materials transportation. In C. Barnhart & G. Laporte (Eds.), *Transportation*, volume 14 of *Handbooks in Operations Research and Management Science,* chapter 9, (pp. 539–621). Elsevier.

Erkut, E. & Verter, V. (1998). Modeling of transport risk for hazardous materials. *Operations Research*, 46(5), 625–642.

Felner, A., Zahavi, U., Holte, R., Schaeffer, J., Sturtevant, N. R., & Zhang, Z. (2011). Inconsistent heuristics in theory and practice. *Artificial Intelligence*, 175(9-10), 1570–1603.

Futtersack, M. & Perny, P. (2000). BCA*, une généralisation d'A* pour la recherche de solutions de compromis dans des problèmes de recherche multiobjectifs. In *Proceedings of the 12th conference Reconnaissance des Formes et Intelligence Artificielle*, volume 3 (pp. 377–386).

Gabrel, V. & Vanderpooten, D. (2002). Enumeration and interactive selection of efficient paths in a multiple criteria graph for scheduling an earth observing satellite. *European Journal of Operational Research*, 139, 533–542.

Galand, L. (2008). *Méthodes exactes pour l'optimisation multicritère dans les graphes: recherche de solutions de compromis.* PhD thesis, Université Pierre et Marie Curie (UPMC).

Galand, L. & Perny, P. (2006). Search for compromise solutions in multiobjective state space graphs. In *17th European Conference on Artificial Intelligence, ECAI'2006* (pp. 93–97).

Galand, L., Perny, P., & Spanjaard, O. (2010). Choquet-based optimisation in multiobjective shortest path and spanning tree problems. *European Journal of Operational Research*, 204(2), 303–315.

Galand, L. & Spanjaard, O. (2007). OWA-Based search in state space graphs with multiple cost functions. In *20th International Florida Artificial Intelligence Research Society Conference, FLAIRS 2007* (pp. 86–91).

Gallo, G. & Pallottino, S. (1988). Shortest path algorithms. *Annals of Operations Research*, 13(1), 1–79.

Gandibleux, X., Beugnies, F., & Randriamasy, S. (2006). Martins' algorithm revisited for multi-objective shortest path problems with a maxmin cost function. *4OR: A Quarterly Journal of Operations Research*, 4(1), 47–59.

Garey, M. R. & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York, NY, USA: W. H. Freeman & Co.

Geisberger, R., Kobitzsch, M., & Sanders, P. (2010). Route planning with flexible objective functions. In *12th Workshop on Algorithm Engineering and Experiments, ALENEX 2010* (pp. 124–137).: SIAM.

Geisberger, R., Sanders, P., Schultes, D., & Delling, D. (2008). Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In C. McGeoch (Ed.), *7th International Workshop on Experimental Algorithms, WEA 2008*, volume 5038 of *Lecture Notes in Computer Science* (pp. 319–333). Springer Berlin / Heidelberg.

Goldberg, A. V. & Harrelson, C. (2005). Computing the shortest path: A* search meets graph theory. In *16th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '05* (pp. 156–165). Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

Guerriero, F. & Musmanno, R. (2001). Label correcting methods to solve multicriteria shortest path problems. *Journal of Optimization Theory and Applications*, 111(3), 589–613.

Guerriero, F., Musmanno, R., Lacagnina, V., & Pecorella, A. (2001). A class of label-correcting methods for the k shortest paths problem. *Operations Research*, 49(3), 423–429.

Hallam, C., Harrison, K., & Ward, J. (2001). A multiobjective optimal path algorithm. *Digital Signal Processing*, 11(2), 133 – 143.

Hamacher, H. W., Ruzika, S., & Tjandra, S. A. (2006). Algorithms for time-dependent bicriteria shortest path problems. *Discrete Optimization*, 3(3), 238 – 254.

Hansen, P. (1979). Bicriterion path problems. In *Lecture Notes in Economics and Mathematical Systems*, volume 177 (pp. 109–127).: Springer.

Harikumar, S. & Kumar, S. (1996). Iterative deepening multiobjective A*. *Information Processing Letters*, 58, 11–15.

Harikumar, S. & Kumar, S. (1997). Multiobjective search based algorithms for circuit partitioning problem for acceleration of logic simulation. In *10th International Conference on VLSI Design* (pp. 239–242).

Hart, P., Nilsson, N., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 100–107.

Huarng, F., Pulat, P., & Shih, L. (1996). A computational comparison of some bicriterion shortest path algorithms. *Journal of the Chinese Institute of Industrial Engineers*, 13(2), 121–125.

Iori, M., Martello, S., & Pretolani, D. (2010). An aggregate label setting policy for the multi-objective shortest path problem. *European Journal of Operational Research*, 207(3), 1489–1496.

Johnson, D. S. (2002). A theoretician‘s guide to the experimental analysis of algorithms. In *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges* (pp. 215–250).: American Mathematical Society.

Jozefowiez, N., Semet, F., & Talbi, E.-G. (2008). Multi-objective vehicle routing problems. *European Journal of Operational Research*, 189, 293–309.

Kanoh, H. (2007). Dynamic route planning for car navigation systems using virus genetic algorithms. *International Journal of Knowledge-based and Intelligent Engineering Systems*, 11, 65–78.

Kanoh, H. & Hara, K. (2008). Hybrid genetic algorithm for dynamic multi-objective route planning with predicted traffic in a real-world road network. In *10th annual conference on Genetic and evolutionary computation, GECCO'08* (pp. 657–664). New York, NY, USA: ACM.

Kara, B. Y., Erkut, E., & Verter, V. (2003). Accurate calculation of hazardous materials transport risks. *Operations Research Letters*, 31(4), 285–292.

Kim, B.-K., Jo, J.-B., Kim, J.-R., & Gen, M. (2009). Optimal route search in car navigation systems by multi-objective genetic algorithms. *International Journal of Information Systems for Logistics and Management*, 4(2), 9–18.

Klingman, D., Napier, A., & Stutz, J. (1974). Netgen: A program for generating large scale capacitated assignment, transportation, and minimum cost flow network problems. *Management Science*, 20(5), 814 –821.

Klunder, G. & Post, H. (2006). The shortest path problem on large-scale real-road networks. *Networks*, 48(4), 182–194.

Korf, R., Zhang, W., Thayer, I., & Hohwald, H. (2005). Frontier search. *Journal of the ACM*, 52(5), 715–748.

Korf, R. E. (1985). Depth-first iterative-deepening : An optimal admissible tree search. *Artificial Intelligence*, 27(1), 97–109.

Korf, R. E. (2010). Artificial intelligence search algorithms. In M. J. Atallah & M. Blanton (Eds.), *Algorithms and theory of computation handbook,* chapter 22, (pp. 22.1–22.23). Chapman & Hall/CRC.

Machuca, E. & Mandow, L. (2011). Multiobjective route planning with precalculated heuristics. In L. Antunes, H. Pinto, R. Prada, & P. Trigo (Eds.), *15th Portuguese Conference on Artificial Intelligence, EPIA 2011* (pp. 98–107).

Machuca, E. & Mandow, L. (2012). Multiobjective heuristic search in road maps. *Expert Systems with Applications*, 39, 6435–6445.

Machuca, E., Mandow, L., Pérez de la Cruz, J., & Ruiz-Sepulveda, A. (2012). A comparison of heuristic best-first algorithms for bicriterion shortest path problems. *European Journal of Operational Research*, 217(1), 44–53.

Machuca, E., Mandow, L., & Pérez de la Cruz, J. L. (2009). An evaluation of heuristic functions for bicriterion shortest path problems. In L. Seabra Lopes, N. Lau, P. Mariano, & L. Rocha (Eds.), *New Trends in Artificial Intelligence, Proc. of 14th Portuguese Conference on Artificial Intelligence, EPIA'09* (pp. 205–216).

Machuca, E., Mandow, L., Pérez De La Cruz, J. L., & Iovanella, A. (2011). Heuristic multiobjective search for hazmat transportation problems. In J. Lozano, J. Gómez, & J. Moreno (Eds.), *14th international conference of the Spanish association for artificial intelligence, CAEPIA'11*, volume 7023 of *Lecture Notes in Computer Science* (pp. 243–252). Berlin, Heidelberg: Springer-Verlag.

Machuca, E., Mandow, L., Pérez de la Cruz, J. L., & Ruiz-Sepúlveda, A. (2010). An empirical comparison of some multiobjective graph search algorithms. In R. Dillmann, J. Beyerer, U. D. Hanebeck, & T. Schultz (Eds.), *33rd Annual German Conference on AI, KI'2010*, volume 6359 of *Lecture Notes in Computer Science* (pp. 238–245). Springer.

Mandow, L. & Pérez de la Cruz, J. L. (2003). Multicriteria heuristic search. *European Journal of Operational Research*, 150, 253–280.

Mandow, L. & Pérez de la Cruz, J. L. (2005). A new approach to multiobjective A* search. In *19th International Joint Conference on Artificial Intelligence, IJCAI'05* (pp. 218–223). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Mandow, L. & Pérez de la Cruz, J. L. (2006). Comparison of heuristics in multiobjective A* search. In R. Marín, E. Onaindía, A. Bugarín, & J. Santos (Eds.), *11th international conference of the Spanish Association for Artificial Intelligence, CAEPIA'05*, volume 4177 of *Lecture Notes in Computer Science* (pp. 180–189). Springer.

Mandow, L. & Pérez de la Cruz, J. L. (2007). A multiobjective frontier search algorithm. In M. M. Veloso (Ed.), *20th International Joint Conference on Artificial Intelligence, IJCAI'07* (pp. 2340–2345). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Mandow, L. & Pérez de la Cruz, J. L. (2008). Frontier search for bicriterion shortest path problems. In *18th European Conference on Artificial Intelligence, ECAI 2008* (pp. 480–484).

Mandow, L. & Pérez de la Cruz, J. L. (2009). A Memory-Efficient search strategy for multiobjective shortest path problems. In *32nd Annual German Conference on AI, KI'2009*, volume 5803 of *Lecture Notes in Computer Science* (pp. 25–32). Springer-Verlag.

Mandow, L. & Pérez de la Cruz, J. L. (2010a). Multiobjective A* search with consistent heuristics. *Journal of the ACM*, 57(5), 27:1–25.

Mandow, L. & Pérez de la Cruz, J. L. (2010b). A note on the complexity of some multiobjective A* search algorithms. In *19th European Conference on Artificial Intelligence, ECAI 2010* (pp. 727–731).

Mandow, L. & Pérez de la Cruz, J. L. (2010c). Path recovery in frontier search for multiobjective shortest path problems. *Journal of Intelligent Manufacturing*, 21(1), 89–99.

Martins, E., Paixão, J.M. Rosa, M., & Santos, J. L. E. (2007). *Ranking multiobjective shortest paths.* Technical Report 2007/011, Centre for Mathematics, University of Coimbra.

Martins, E., Pascoal, M. M. B., & Santos, J. L. E. (2000). *Labeling algorithms for ranking shortest paths.* Technical Report 2000/001, Centre for Mathematics, University of Coimbra.

Martins, E., Pascoal, M. M. B., & Santos, J. L. E. (2001). A new improvement for a k shortest paths algorithm. *Investigação Operacional*, 21, 47–60.

Martins, E. Q. V. (1984a). An algorithm for ranking paths that may contain cycles. *European Journal of Operational Research*, 18(1), 123 – 130.

Martins, E. Q. V. (1984b). On a multicriteria shortest path problem. *European Journal of Operational Research*, 16, 236–245.

Martins, E. Q. V. (1984c). On a special class of bicriterion path problems. *European Journal of Operational Research*, 17(1), 85 – 94.

Martins, E. Q. V. & Santos, J. L. E. (1999). *The Labeling Algorithm for the Multiobjective Shortest Path Problem.* Technical Report 1999/005, Centre for Mathematics, University of Coimbra.

Mérõ, L. (1984). A heuristic search algorithm with modifiable estimate. *Artificial Intelligence*, 23(1), 13–27.

Mote, J., Murthy, I., & Olson, D. L. (1991). A parametric approach to solving bicriterion shortest path problems. *European Journal of Operational Research*, 53(1), 81–92.

Müller-Hannemann, M. & Weihe, K. (2006). On the cardinality of the Pareto set in bicriteria shortest path problems. *Annals of Operations Research*, 147(1), 269–286.

Murthy, I. & Olson, D. L. (1994). An interactive procedure using domination cones for bicriterion shortest path problems. *European Journal of Operational Research*, 72(2), 417 – 431.

Nance, R., Moose, R., & Foutz, R. (1987). A statistical technique for comparing heuristics: an example from capacity assignment strategies in computer network design. *Communications of the ACM*, 30(5), 430–442.

Nannicini, G., Delling, D., Liberti, L., & Schultes, D. (2008). Bidirectional A* search for time-dependent fast paths. In *7th international conference on Experimental algorithms, WEA'08* (pp. 334–346). Berlin, Heidelberg: Springer-Verlag.

Nilsson, N. J. (1998). *Artificial Intelligence: A New Synthesis.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1st edition.

Paixão, J. & Santos, J. L. E. (2007). *Labelling methods for the general case of the multiobjective shortest path problem - A computational study.* Technical Report 2007/042, Centre for Mathematics, University of Coimbra.

Paixão, J. & Santos, J. L. E. (2008). *A new ranking path algorithm for the multi-objective shortest path problem.* Technical Report 2008/027, Centre for Mathematics, University of Coimbra.

Pallottino, S. & Scutellà, M. G. (1998). Shortest path algorithms in transportation models: classical and innovative aspects. In P. Marcotte & S. Nguyen (Eds.), *Equilibrium and advanced transportation modelling* (pp. 245–281). Dordrecht: Kluwer Academic Publishers.

Pearl, J. (1984). *Heuristics.* Reading, Massachusetts: Addison-Wesley.

Perny, P. & Spanjaard, O. (2008). Near admissible algorithms for multiobjective search. In *18th European Conference on Artificial Intelligence, ECAI 2008* (pp. 490–494). Amsterdam, The Netherlands: IOS Press.

Raith, A. (2009). *Multiobjective Routing and Transportation Problems.* PhD thesis, University of Auckland.

Raith, A. & Ehrgott, M. (2009). A comparison of solution strategies for biobjective shortest path problems. *Computers & Operations Research*, 36(4), 1299–1331.

Refanidis, I. & Vlahavas, I. (2003). Multiobjective heuristic state-space search. *Artificial Intelligence*, 145, 1–32.

Sanders, P., Schultes, D., & Vetter, C. (2008). Mobile route planning. In *16th annual European symposium on Algorithms, ESA'08* (pp. 732–743). Berlin, Heidelberg: Springer-Verlag.

Santos, J. L., Paixão, J. P., & Rosa, M. S. (2005). A statistical analysis on the number of non-dominated paths in the multiobjective shortest path problem. In *International Network Optimization Conference, INOC'05.*

Santos, J. L. E. (2007b). Multiobjective shortest path problems (Assembled by José Luis Esteves dos Santos <zeluis@mat.uc.pt>, Departamento de Matemática, Universidade de Coimbra). `http://www.mat.uc.pt/~zeluis/INVESTIG/MSPP/mspp.htm`.

Sauvanet, G. (2011). *Recherche de chemins multiobjectifs pour la conception et la réalisation d'une centrale de mobilité destinée aux cyclistes.* PhD thesis, Université François Rabelais - Tours.

Sauvanet, G. & Néron, E. (2010). Search for the best compromise solution on multiobjective shortest path problem. *Electronic Notes in Discrete Mathematics*, 36(1), 615 – 622.

Schultes, D. (2005). United States Road Networks (TIGER/Line) for 9th DIMACS implementation challenge - Shortest Paths (Assembled by Dominik Schultes <mail@dominik-schultes.de>, Universität Karlsruhe (TH), Germany). `http://www.dis.uniroma1.it/~challenge9/data/tiger/`.

Schultes, D. (2008). *Route Planning in Road Networks.* PhD thesis, Fakultät für Informatik der Universität Fridericiana zu Karlsruhe (KIT).

Schulz, F. (2005). *Timetable information and shortest paths.* PhD thesis, Fakultät für Informatik der Universität Fridericiana zu Karlsruhe (KIT).

Schulz, F., Wagner, D., & Weihe, K. (2000). Dijkstra's algorithm on-line: An empirical case study from public railroad transport. *ACM Journal of Experimental Algorithmics*, 5, 12.

Schulz, F., Wagner, D., & Zaroliagis, C. (2002). Using multi-level graphs for timetable information in railway systems. In D. M. Mount & C. Stein (Eds.), *4th International Workshop on Algorithm Engineering and Experiments, ALENEX 2002*, volume 2409 of *Lecture Notes in Computer Science* (pp. 43–59). Springer Berlin / Heidelberg.

Serafini, P. (1986). Some considerations about computational complexity for multiobjective combinatorial problems. In J. Jahn & W. Krabs (Eds.), *Recent advances and historical development of vector optimization*, volume 294 of *Lecture Notes in Economics and Mathematical Systems.* Berlin: Springer-Verlag.

Skriver, A. J. V. (2000). A classification of bicriterion shortest path (bsp) algorithms. *Asia-Pacific Journal of Operational Research*, 17, 199–212.

Skriver, A. J. V. & Andersen, K. A. (2000). A label correcting approach for solving bicriterion shortest-path problems. *Computers & Operations Research*, 27(6), 507–524.

Stewart, B. S. & White, C. C. (1991). Multiobjective A*. *Journal of the ACM*, 38(4), 775–814.

Tarapata, Z. (2007). Selected multicriteria shortest path problems: an analysis of complexity, models and adaptation of standard algorithms. *International Journal of Applied Mathematics and Computer Science*, 17(2), 269–287.

Tsaggouris, G. & Zaroliagis, C. D. (2009). Multiobjective optimization: Improved fptas for shortest paths and non-linear objectives with applications. *Theory of Computing Systems*, 45(1), 162–186.

Tung, C. T. & Chew, K. L. (1988). A bicriterion Pareto-optimal path algorithm. *Asia-Pacific Journal of Operational Research*, 5, 166–172.

Tung, C. T. & Chew, K. L. (1992). A multicriteria Pareto-optimal path algorithm. *European Journal of Operational Research*, 62, 203–209.

U.S. Census Bureau (2002). UA Census 2000 TIGER/Line Files. `http://www.census.gov/geo/www/tiger/tigerua/ua_tgr2k.html`.

Vincke, P. (1974). Problèmes multicritères. *Cahiers du Centre d' Etudes et de Recherche Opérationnelle*, 16(4), 425–439.

Wijeratne, A. B., Turnquist, M. A., & Mirchandani, P. B. (1993). Multiobjective routing of hazardous materials in stochastic networks. *European Journal of Operational Research*, 65(1), 33–43.

Zeng, W. & Church, R. L. (2009). Finding shortest paths on real road networks: the case for A*. *International Journal of Geographical Information Science*, 23(4), 531.

Zhan, F. B. & Noon, C. E. (1998). Shortest path algorithms: An evaluation using real road networks. *Transportation Science*, 32(1), 65–73.

Zhan, F. B. & Noon, C. E. (2000). A comparison between label-setting and label-correcting algorithms for computing one-to-one shortest paths. *Journal of Geographic Information and Decision Analysis*, 4(2), 1–11.

Zhou, R. & Hansen, E. A. (2002). Memory-bounded A* graph search. In *15th International Florida Artificial Intelligence Research Society Conference, FLAIRS 2002* (pp. 203–209).

Zhou, R. & Hansen, E. A. (2006). Breadth-first heuristic search. *Artificial Intelligence*, 170(4-5), 385–408.

Bibliography Index

Y por fin...
«Bien está lo que bien acaba»

Palabras que dan nombre a una obra
«All's Well That Ends Well»
que en castellano sería
«A buen fin no hay mal principio»
(William Shakespeare, 1623)

«El Séptimo día Dios tuvo terminado su trabajo,
y descansó en ese día de todo lo que había hecho.
Bendijo Dios el Séptimo día y lo hizo santo,
porque ese día descansó de sus trabajos
después de toda esta creación que había hecho»
(Génesis 2,2-3)