New Techniques and Algorithms for Multiobjective and Lexicographic Goal-Based Shortest Path Problems

UNIVERSIDAD
DE MÁLAGA

TESIS DOCTORAL

Francisco Javier Pulido Arrebola

Universidad de Málaga

7 de Julio de 2015

Documento maquetado con TEXIS v.1.0.

New Techniques and Algorithms for Multiobjective and Lexicographic Goal-Based Shortest Path Problems

Memoria que presenta el doctorando

Francisco Javier Pulido Arrebola

para optar al grado académico de Doctor

Dirigida por el Doctor

Lawrence Mandow Andaluz

Programa de Doctorado en Ingeniería del Software e Inteligencia Artificial
Departamento de Lenguajes y Ciencias de la Computación
Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

7 de Julio de 2015

Tribunal de la tesis / Thesis Committee

Dr. Rafael Morales Bueno - Universidad de Málaga
Dra. Amparo Ruíz Sepúlveda - Universidad de Málaga
Dr. Carlos Linares López - Universidad Carlos III de Madrid
Dra. Camino Rodríguez Vela - Universidad de Oviedo
Dra. Lucie Galand - Université Paris Dauphine

Evaluadores externos / External Reviewers

Dra. Andrea Raith - University of Auckland (New Zealand)
Dr. Antonio Iovanella - University of Rome Tor Vergata (Italy)

E-mail: francis@lcc.uma.es
Web: http://www.lcc.uma.es/~francis

This work was partially funded by / Este trabajo fue parcialmente financiado por:
Consejería de Economía, Innovación, Ciencia y Empresa.
Junta de Andalucía (España)
Referencia: P07-TIC-03018

El Dr. D. Lawrence 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. Francisco Javier Pulido Arrebola, 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:

New Techniques and Algorithms for Multiobjective and Lexicographic Goal-Based Shortest Path Problems

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. Lawrence Mandow Andaluz

Málaga, 7 de Julio de 2015

Para mis padres,
por su amor, dedicación y sacrificios.

Agradecimientos

Gracias, en primer lugar y de todo corazón, a mi tutor, el doctor Lawrence Mandow. Su paciencia, dedicación y guía me han hecho llegar hasta donde estoy. Él, como ninguna otra persona, me ha enseñado la pasión por la investigación, el trabajo metódico y la minuciosidad por los detalles. Ha sido un placer y un privilegio poder contar con su dirección y ayuda.

Gracias a José Luis Pérez de la Cruz, catedrático de la Universidad de Málaga y una de las mentes más brillantes que conozco, por su inestimable ayuda y colaboración. Su visión crítica y experiencia han contribuido a enriquecer los resultados de esta tesis.

Thanks to everyone in the University of Maths in Belgrade and the Computer Engineering and Informatics Department of the University of Patras. Dr. Dušan Tošic, Dr. Vladimir Filipović, Dr. Christos Zaroliagis, Nikos Rousias, Dimitris Gkortsilas and everyone else who made my PhD visits so enriching and enjoyable.

Gracias a mis compañeros de laboratorio: Javi, Dani, Martyna, Rafa y todos aquellos con los que he compartido alegrías y penas a lo largo de esta Tesis, y especialmente, a los doctores Enrique Machuca y Jaime Gálvez, por su desinteresada ayuda y su eterna disponibilidad. Esta Tesis Doctoral y mi tiempo en la Universidad no habrían sido lo mismo sin ellos.

Gracias a la doctora Raquel Barco, por brindarme la oportunidad de dar un enfoque más comercial a los resultados de esta tesis. Gracias a Nacho, Juan, y el resto de mis compañeros del proyecto de tráfico inteligente por nuestras entretenidas reuniones diarias de sincronización.

Y por encima de todo, y con todo mi amor, gracias a los míos por estar incondicionalmente conmigo, por apoyarme siempre y sin restricciones. Gracias Mamá, Papá, Mari, Lena y Runi. Os quiero mucho.

Francisco Javier Pulido Arrebola
Málaga, 7 de Julio de 2015

Abstract

Shortest Path Problems (SPP) are one of the most extensively studied problems in the fields of Artificial Intelligence (AI) and Operations Research (OR). It consists in finding the shortest path between two given nodes in a graph such that the sum of the weights of its constituent arcs is minimized. However, real life problems frequently involve the consideration of multiple, and often conflicting, criteria. When multiple objectives must be simultaneously optimized, the concept of a single optimal solution is no longer valid. Instead, a set of efficient or Pareto-optimal solutions define the optimal trade-off between the objectives under consideration.

The Multicriteria Search Problem (MSP), or Multiobjective Shortest Path Problem, is the natural extension to the SPP when more than one criterion are considered. The MSP is computationally harder than the single objective one. The number of label expansions can grow exponentially with solution depth, even for the two objective case (Hansen, 1980). However, with the assumption of bounded integer costs and a fixed number of objectives the problem becomes tractable for polynomially sized graphs (e.g. see (Mandow & Pérez de la Cruz, 2009; Müller-Hannemann & Weihe, 2006)).

A wide variety of practical application in different fields can be identified for the MSP, like robot path planning (Wu et al., 2011), hazardous material transportation (Caramia et al., 2010), route planning (Jozefowiez et al., 2008), optimization of public transportation (Raith, 2009), QoS in networks (Craveirinha et al., 2009), or routing in multimedia networks (Climaco et al., 2003).

Goal programming is one of the most successful Multicriteria Decision Making (MCDM) techniques used in Multicriteria Optimization. In this thesis we explore one of its variants in the MSP. Thus, we aim to solve the Multicriteria Search Problem with lexicographic goal-based preferences. To do so, we build on previous work on algorithm NAMOA*, a successful extension of the A^* algorithm to the multiobjective case. More precisely, we provide a new algorithm called LEXGO*, an exact label-setting algorithm that returns the subset of Pareto optimal paths that satisfy a set of lexicographic goals, or the subset that minimizes deviation from goals if these cannot be fully satisfied. Moreover, LEXGO* is proved to be admissible and expands only a subset of the labels expanded by an optimal algorithm like NAMOA*, which performs a full Multiobjective Search.

Since time rather than memory is the limiting factor in the performance of multicriteria search algorithms, we also propose a new technique called *t-discarding* to

speed up dominance checks in the process of discarding new alternatives during the search. The application of *t-discarding* to the algorithms studied previously, NAMOA* and LEXGO*, leads to the introduction of two new time-efficient algorithms named NAMOA$^*_{\text{dr}}$ and LEXGO$^*_{\text{dr}}$, respectively.

All the algorithmic alternatives are tested in two scenarios, random grids and realistic road maps problems. The experimental evaluation shows the effectiveness of LEXGO* in both benchmarks, as well as the dramatic reductions of time requirements experienced by the t-discarding versions of the algorithms, with respect to the ones with traditional pruning.

Contents

List of Figures

List of Tables

Part I

Motivation and Fundamentals

This first part of this thesis is divided into three chapters. The first one is devoted to introduce its field of study. We enumerate the goals of this dissertation and introduce the contributions that we will develop further in the second part. The second chapter describes our main subject of study and research, Multicriteria Search problems and algorithms. The third chapter is dedicated to the benchmarks used to conduct the experimental evaluation. Thus, in particular:

- Chapter 1 gives an overview of the motivation, scope and goals of this thesis, and enumerates its contributions.
- Chapter 2 defines the Multicriteria Search Problem, gives some examples of application and classifies the approaches to deal with the problem. Formal properties of the lower bounds to apply to the multicriteria search algorithms are also described. Finally, NAMOA* is introduced emphasizing its relevant features and properties to our own work.
- Chapter 3 enumerates relevant benchmarks employed in the literature and describes the test beds used in this thesis, as well as the main parameters followed in the experimental evaluation.

Chapter 1
Introduction

Research is what I'm doing when I don't know what I'm doing

Wernher von Braun (1912-1977)

This doctoral dissertation falls within the scope of the Artificial Intelligence (AI) and Operations Research (OR) fields. Shortest Path Problems (SPP) are one of the oldest and most extensively studied problems in both fields, which consists in finding the shortest path between two given nodes in a graph such that the sum of the weights of its constituent arcs is minimized. The SPP arises naturally in real life, e.g. planning the route path in a road trip, or navigating a mobile robot to avoid obstacles, and can be also used to solve optimally puzzle games like Rubik's cube (Korf, 1997) or the twenty-four puzzle (Korf & Taylor, 1996).

The Multicriteria Search Problem (MSP), or Multiobjective Shortest Path Problem, is the natural extension to the SPP whenever more than one criterion is considered. The MSP is computationally harder than the single objective one. The number of label expansions can grow exponentially with solution depth, even for the two objective case (Hansen, 1980). With the assumption of bounded integer costs and a fixed number of objectives the problem becomes tractable for polynomially sized graphs, but still harder than single objective search (e.g. see (Mandow & Pérez de la Cruz, 2009; Müller-Hannemann & Weihe, 2006)).

Recent experiments on problems like bicriteria route planning have revealed that time, rather than space, is the practical limiting factor in the calculation of the full set of efficient solutions in exact algorithms in Multicriteria Search (Machuca & Mandow, 2012; Machuca et al., 2009). In this thesis we address this problem from the point of view of Goal Programming (GP), which has proven to be a very effective model of decision maker's preferences over multicriteria decision making (MCDM) problems. Goal Programming is a general paradigm which claims that a decision problem can be expressed through a set of goals defined by the decision maker, rather than through the optimization of a set of objectives. Roughly speaking, a goal defines a degree of satisfaction of a given criteria that is deemed satisfactory or acceptable by the decision maker. One of the most commonly used schemes to express goal-based preferences is the lexicographic method of grouping criteria by pre-emptive importance (Charnes &

Cooper, 1977; Romero, 1991).

Our starting hypothesis is that in those cases where user preferences can be initially bounded by a set of goals, specially designed search algorithms could perform more efficiently than searching for the full Pareto frontier. The main goals of this thesis are to explore this hypothesis, and to improve the performance of current multiobjective search algorithms. More precisely, we explore new algorithmic contributions to the MSP with lexicographic goal-based preferences, and also a new technique to speed up multicriteria search algorithms based on labeling techniques.

Section 1.1 introduces the motivation and significance of this work for the AI and OR fields. Section 1.2 presents our scope and orientation. The goals and contributions of this thesis are enumerated in Sections 1.3 and 1.4, respectively. Related publications of this research work are shown in Section 1.5. Finally, Section 1.6 outlines the structure of this thesis.

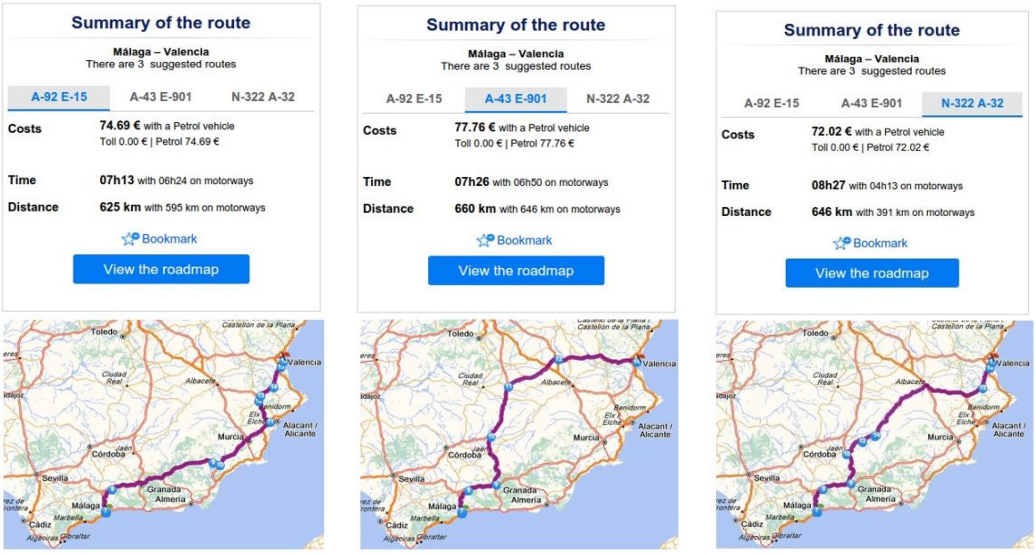
1.1 Motivation

The Shortest Path Problem is a recurrent problem in the AI and OR literature. Dijkstra (1959) proposed the first algorithm to find the minimal cost route between two nodes in a graph. The A* algorithm (Hart et al., 1968) is an important algorithmic reference that exploits specific problem knowledge (the so-called heuristic function in the AI community) to guide the search and improve its efficiency. This problem knowledge is in the form of a distance or cost estimate.

Real life decision problems frequently involve the consideration of multiple criteria simultaneously. For example, Figure 1.1 shows three car routes from Málaga to Valencia (both in Spain) suggested by a sample web application devoted to road route planning. Assume we are concerned with the minimization of three different criteria in this problem: travel cost, time and distance. Route 1.1(a) is the fastest and shortest route and Route 1.1(c) is the cheapest. Whenever there are no other preferences defined, we say that both routes are *efficient solutions* to this problem, i.e. these solutions represent an optimal trade-off between the criteria. These are extreme efficient solutions, but there might well be other interesting trade-offs. The set of all solutions such that none of the objectives can be improved without worsening at least one of the others, is called the Pareto set or Pareto frontier, and they are call efficient or non-dominated solutions.

Among all feasible routes in our example only a subset can be considered non-dominated in terms of all objectives. Thus, route 1.1(b) is called a dominated solution, since there is another route, route 1.1(a), that is lower in all objectives. However, the calculation of the Pareto set is typically a hard problem, hence, we consider the application of Goal Programming (GP) to model the decision maker's preferences and establish a set of goals that restrict the set of solutions to those ones which satisfy those goals, or which minimize the deviation from the goals if they can not be satisfied. Thus, in our example, a user of the road route planner can define their expectations of the maximum economic cost, time and distance for the trip.

In order to deal with the problem described above, two different alternatives can be followed. In the first one, a full Multicriteria Search algorithm can return the full Pareto



(a) Route 1 (b) Route 2 (c) Route 3

Figure 1.1. Screenshots from a sample route planning web application, showing alternative routes from Málaga (Spain) to Valencia (Spain). © `www.viamichelin.com`

set of solutions to the problem. This set can then be used to determine the subset of solutions that satisfy the goals, or in case the goals can not be fully satisfied, the subset which minimizes the deviation from those goals according to a given definition of distance. For this alternative, we consider NAMOA*, which is currently the most efficient exact algorithm to calculate the Pareto frontier.

The second alternative to deal with the MSP with goal-based preferences concentrates the search effort on the subset of Pareto optimal solutions that satisfy the goals, discarding those paths that will not lead to efficient solutions according to the goals. Notice that we shall always seek for efficient or non-dominated solutions for the problem, regardless whether the goals can be satisfied or not. This is not the case of Goal Programming in general, that allows non-dominated solutions to the problem.

For the second alternative, we introduce LEXGO*, a best-first Multicriteria Search algorithm that is also exact and efficient when provided with consistent lower bounds.

Finally, this thesis is also devoted to improve the time performance of exact multicriteria search algorithms with consistent lower bounds and to do so, a new dimensionality reduction technique is introduced. This technique allows discarding non-dominated paths with a dramatic decreased number of dominance checks against permanent labels. Its application to NAMOA* and LEXGO* leads to the introduction of two new algorithms, $\text{NAMOA}^*_{\text{dr}}$ and $\text{LEXGO}^*_{\text{dr}}$. Their effectiveness is also studied from formal and empirical points of view against traditional pruning or discarding.

1.2 Scope and Orientation

We summarize the boundaries of this doctoral dissertation with the following terms:

Multicriteria Search with lower bounds. The Shortest Path Problem consists in finding the shortest path between two nodes in a graph. The natural extension when considering multiple conflicting criteria is the Multicriteria Search Problem. We consider the A^* algorithm as a reference in the solution of this problem and, therefore, analyze multicriteria search algorithms whose performance can improve with the use of lower bounds or distance estimates to restrict the number of label expansions.

Additive costs. Arcs in the graph are labeled with a magnitude for each criterion, or cost, under consideration. In the MSP, the arcs in the graph represent the costs of navigating from the source to the destination node. The cost of the route in the graph to be minimized corresponds to the sum of the costs vectors of its component arcs.

Pareto optimality. When multiple criteria are considered, the concept of minimum no longer applies, and is usually replaced by that of Pareto optimality. Pareto optimal solutions are defined as those that cannot be improved according to one objective without worsening at least one of the others.

Label-setting multicriteria search algorithms. Best-first algorithms are classified in the OR literature as label-setting or label-correcting. The former explores only optimal paths in the graph, and therefore set permanent labels for each scanned node. The latter may also explore suboptimal paths and therefore may establish temporary labels for scanned nodes. When algorithms A* or NAMOA* use consistent lower bounds as cost estimates, they behave as label-setting algorithms. That will also be the case for the algorithms analyzed in this thesis.

Pre-emptive and weighted preferences. Goal Programming models preferences given by a decision maker for a multicriteria problem. We focus on the technique where the criteria are grouped in priority levels sorted in order of decreasing pre-emptive importance. Furthermore, a level comprises a set of one or more attributes. We define targets for each attribute and weights to establish the importance of the criteria within the level.

Theoretical proofs of correctness. We contribute in this thesis several new algorithms. Formal proofs of their admissibility and efficiency are provided. These are complemented with empirical analyses of the algorithms.

1.3 Research Goals

The main goals of this doctoral dissertation can be outlined as follows:

1. **Address the Multicriteria Search Problem with goal-based preferences.** The first goal of this thesis is the description of the Multicriteria Search Problem and specifically, the Multicriteria Search Problem with Preferences Based on Goals. We will describe the two considered approaches to deal with such a problem in detail.

2. **Devise a new algorithm to cope with lexicographic goal-based preferences.** The principle of optimality holds for Multiobjective Shortest Path Problems, but regrettably does not hold for lexicographic goal-based preferences. In order to approach the MSP with a specifically designed goal-based algorithm, we concentrate our efforts to devise a new algorithm based on a label selection policy.

3. **Prove formally the correctness and efficiency of this new algorithm.** Another goal of this thesis is to complement the defined algorithm with formal analyses of its correctness as well as its efficiency over an optimal algorithm that performs a full Multiobjective Search (NAMOA*).

4. **Study possible improvements to multiobjective shortest path algorithms.** Our efforts will also be focused on the improvement of multiobjective shortest path algorithms in general. More specifically, we aim to introduce a new dimensionality reduction technique to improve the runtime performance of label-setting multiobjective shortest path algorithms with consistent lower bounds.

5. **Prove formally the correctness of the dimensionality reduction technique.** We will formally develop the application of the dimensionality reduction technique to the new defined algorithm based on goal preferences, as well as the application to NAMOA*. In particular, we will prove theoretically the correctness of both algorithms when employing the dimensionality reduction technique.

6. **Perform an empirical evaluation of all the algorithmic alternatives.** Finally, the last goal of this dissertation is to provide an extensive evaluation of all the proposed algorithms. We will employ test beds over randomly generated and realistic scenarios.

1.4 Contributions

The contributions of this thesis are enumerated as follows:

1. **Description of the Goal-Based Multicriteria Graph Search Problem.** In the first place, we outline the Goal Programming technique within the Multicriteria Decision Making discipline. We also outline the Multicriteria Graph Search within the Multiobjective Optimization Problem, and given that framework, we will describe and define formally the Goal-Based Multicriteria Graph Search problem and the different algorithmic approaches to deal with it.

2. **A new Multicriteria Search algorithm for the MSP.** We introduce LEXGO* (Lexicographic Goals A*), a new exact label-setting algorithm for Multicriteria Search problems with goal-based preferences. LEXGO* returns the subset of non-dominated optimal paths that satisfy a set of lexicographic goals, or the subset that minimizes deviation from goals if these cannot be fully satisfied.

3. **Formal characterization of the admissibility and efficiency of LEXGO*.** We prove theoretically the admissibility of LEXGO*, i.e. the algorithm is exact

and returns the whole set of solutions to the problem, as well as its efficiency, i.e. the number of labels scanned by the algorithm decreases with better (more informed) lower bounds, over the full Multicriteria Search.

4. **Introduction of *t-discarding*.** We introduce a new dimensionality reduction technique called *t-discarding*. This technique can be applied to the processes of pruning and filtering in order to reduce in one dimension the size of the vectors considered to discard new alternatives. Thus, we will reduce the amount of time needed to check dominance against closed vectors and therefore, the time requirements of the algorithms.

5. **Formal characterization of the t-discarding technique.** We apply t-discarding to NAMOA* and LEXGO*, introduce new algorithms NAMOA$^*_{\text{dr}}$ and LEXGO$^*_{\text{dr}}$, show their correctness, evaluate their effectiveness, and analyze their performance.

6. **Empirical evaluation of the new algorithmic contributions.** Two main scenarios are used to test the effectiveness of our devised algorithms, random grids and realistic road maps problems. We present dramatic reduction in time requirements for three-objective, four-objective and five-objective search problems over random grids. To the best of our knowledge, the results reported over road maps problems represent the largest three-objective search problems solved to date.

1.5 Related Publications

Contributions of this thesis has been presented in international peer-reviewed journals and conferences.

- **Journals:**

  Pulido, F. J., Mandow, L., & Pérez de la Cruz, J. (2014). Multiobjective shortest path problems with lexicographic goal-based preferences. European Journal of Operational Research, 239(1), 89-101. doi:10.1016/j.ejor.2014.05.008

  Pulido, F. J., Mandow, L., & Pérez de la Cruz, J. (2015). Dimensionality reduction in multiobjective shortest path search. Computers & Operations Research. Volume 64, 60-70. doi:10.1016/j.cor.2015.05.007

- **Conferences:**

  Mandow, L., Pulido, F. J., & Pérez de la Cruz, J. L. (2013). Searching Graphs with Lexicographic Goal Preferences. In 22nd International Conference on Multiple Criteria Decision Making - MCDM 2013.

1.6 Outline

This doctoral dissertation is structured in eight chapters grouped in three parts. The first part is devoted to present the motivation of this thesis, its foundations and previous work, and comprises this introductory chapter, and Chapters 2 and 3. Chapter

2 introduces the reader to the Multicriteria Search Problem with lexicographic goal-based preferences. Relevant models of decision maker's preferences are enumerated and several practical applications are described. This chapter also introduces the fundamentals of Multicriteria Search Problems, reviews different kinds of multiobjective search methods, and presents NAMOA*, the reference algorithm we use to evaluate our contributions under a common framework. Chapter 3 briefly reviews previous relevant multiobjective benchmarks. A set of good practices in the experimental evaluation of algorithms as well as the key concepts to measure the performance of Multicriteria Search are also presented. Finally, benchmarks employed in this thesis are described in detail.

The second part of this dissertation groups the main contributions of this thesis. Chapter 4 presents the new algorithms LEXGO*, NAMOA$^*_{\text{dr}}$ and LEXGO$^*_{\text{dr}}$. Chapter 5 gives a formal analysis of the algorithms devised as a result of this thesis, and presents the proofs for their properties of exactness and efficiency, i.e. it is formally proved for each algorithm in this thesis that the full set of efficient solutions is returned and the number of explored labels decreases with more informed lower bounds. Chapters 6 and 7 analyze the performance of all the algorithmic alternatives over random grids and realistic road maps problems, respectively. These reveal the effectiveness of the proposed techniques.

Finally, part three summarizes the conclusions of this thesis and introduces future lines of work in Chapter 8.

Chapter 2
MultiCriteria Graph Search

It is not the task of the University to offer what society asks for, but to give what society needs.

Edsger Dijkstra (1930-2002)

This chapter has a twofold purpose, on one hand, provide an overview of two opposite decision paradigms: optimization and satisfaction. On the other hand, deepen insight into the Multicriteria Search Problem (MSP), present the state of the art in algorithms to deal with such problems, and delimit the frame of the algorithms studied in this thesis.

An optimization problem is the problem of finding the best solution from all feasible solutions. This process is suitable for developing computational algorithms that optimize the decision according to given criteria, although in certain cases cannot be appropriate due to its greater computational requirements to solve problems. Humans, however, do not pursue the optimization of their decisions in general. This is where the concept of satisfaction emerges. A satisfactory solution from this perspective is any feasible solution that fulfills the standards or goals of the decision maker. We will further elaborate on both decision paradigms in this Chapter.

The Multicriteria Decision Making (MCDM) discipline is introduced in the first place in Section 2.1. Both decision paradigms explained above are part of this discipline. Then, the Multiobjective Optimization Problem (MOP) is presented in Section 2.2 along with a classification of the main methods to deal with it.

Goal Programming (GP), see Section 2.3, is a popular method for dealing with multiple objective decision-making problems based on satisfying the goals of a decision maker. GP is a branch of Multicriteria Decision Making that aims to "satisfice" instead of optimize. The most popular variants of GP preference modeling are described in Section 2.3.1. Among them, we focus on Lexicographic GP and introduce relevant definitions concerning this modeling tool in Section 2.3.2.

This thesis tackles the Shortest Path Problem with multiple criteria. Prior to defining the MSP, the Shortest Path Problem is defined in Section 2.4 along with the most well known approaches to solve it. Right after, the MSP is presented in Section 2.5, as well as a wide range of fields where the MSP has been applied successfully.

Sections 2.6 and 2.7 analyze the two main algorithmic approaches to solve a MSP, classified depending on the a priori or a posteriori character of the decision maker's preferences. In the first case, we will further study the class of best-first algorithms, and in particular, the state-of-the-art algorithm NAMOA*. In the second case, we review research on multicriteria search algorithms that can provide compromise solutions or solutions according to some targets.

Finally, Section 2.8, summarizes the key concepts of this given chapter, and introduces the motivation and contributions of this research work.

2.1 Multicriteria Decision Making

Multicriteria Decision Making (MCDM), also called Multicriteria Decision Analysis (MCDA), is a sub-discipline of the field of Operations Research. MCDM is a paradigm that involves the consideration of multiple, and in general conflicting, criteria in decision-making environments. In real life problems, there are typically multiple conflicting criteria that need to be evaluated in making decisions. The purpose of MCDM is to support decision makers (an individual or a group of individuals) to make those decisions. Typically, there does not exist a unique optimal solution for such problems and it is necessary to use the decision maker's (DM) preferences to select between solutions. For instance, when buying a new computer, cost, performance and design may be some of the main criteria a buyer considers. It is unusual to have the cheapest computer to be the best designed and the most powerful.

Solving a MCDM problem can be interpreted in different ways. A decision maker can seek the "most preferred" alternative, i.e. the best alternative from a set of available alternatives. Another interpretation could be choosing a small set of good alternatives, or grouping alternatives into different preference sets. Finally, the last interpretation could be to find all *efficient* or *non-dominated* alternatives.

Let us first introduce some relevant concepts taken from (Romero, 1991, 1993):

Definition 2.1 *Zeleny (Zeleny, 1982) and Romero define* ***attributes*** *as descriptors of an objective reality to represent values of the DMs. These values are measurable properties that can be expressed as a mathematical function* $g(\vec{x}) : X \to \mathbb{R}$*, where* $\vec{x}$ *is the vector of the decision variables and* X *is the set of solutions to the decision problem.*

Definition 2.2 ***Objectives*** *represent the desired improvement of an attribute, i.e. the maximization or the minimization of the mathematical functions corresponding to the attributes under consideration. In short, objectives take the form:* $\max g(\vec{x})$ *or* $\min g(\vec{x})$*.*

Definition 2.3 *A* ***target*** *or aspiration level,* $t \in \mathbb{R}$*, is an acceptable level of achievement for any of the attributes considered by the DM. A* ***goal*** *is a combination of an attribute with a target, stated by the decision maker to define their preference.*

Definition 2.4 *A* ***deviation variable*** *represents the distance between the i-th goal and its associated aspiration level. A formulation model is defined as the minimization*

Table 2.1. Goals and deviation variables (taken from (Romero, 1993).

Type of objective function	Goal type	Deviation to minimize
Minimization	$g_i(\vec{x}) \leq t_i$	d_i
Maximization	$g_i(\vec{x}) \geq t_i$	p_i
Exact achievement	$g_i(\vec{x}) = t_i$	$d_i + p_i$

of the deviation variables to achieve the goals. Three different kinds of goals are defined in this context. In each of them we select different deviation variables to minimize. Table 2.1 shows the concept of unwanted deviation variable, or variable to minimize, depending on the kind of goal.

The distance between the i-th *goal and its associated aspiration level may be negative (represented by d_i) or positive (represented by p_i). A negative value represents the number of units in which the i-th goal falls below with respect to the target defined. The positive value represents just the opposite, i.e. the number of units in which the achievement of the i-th goal has been surpassed regarding the aspiration level proposed. In general, the i-th goal expressed algebraically is,*

$$g_i(x) + d_i - p_i = t_i \qquad d_i, p_i \geq 0 \tag{2.1}$$

Definition 2.5 *The term **criterion** comprises attributes, objectives and goals of a DM relevant to a particular decision-making problem.*

Romero (1993) splits the MCDM framework into two scenarios. The first one corresponds to a decision making situation with a discrete number of feasible solutions to be ranked according to different attributes. In this case a multiattribute utility function represents the preferences of the DM and is used to order the set of finite feasible alternatives. This approach is called Multiattribute Decision Making (MADM) (see for example (Tzeng & Huang, 2011)).

The second scenario corresponds to a decision making situation with an infinite number of decision alternatives where the practical possibility of obtaining a reliable representation of the DM's utility function is very limited. In this case with multiple objectives the Multiobjective Optimization, also known as Multiobjective Programming (MOP), is the approach to consider. In general, a Multiobjective Optimization problem assumes a simple preference structure, the so-called Pareto ordering (Pareto, 1897), to find the set of trade-offs between all objectives considered (Chankong & Haimes, 1983).

Let us now revise a formal definition of the Multiobjective Optimization Problem, introduce definitions relevant to this research work, and classify the most popular techniques to deal with it.

2.2 Multiobjective optimization

Let us first introduce a mathematical formulation of the Multiobjective Optimization Problem,

Definition 2.6 *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 $k \in \{1, 2, \ldots, q\}$ objectives. A multiobjective problem in X can be formulated as a minimization problem,*

$$\min \vec{f}(\vec{x}) = (f^1(\vec{x}), f^2(\vec{x}), \ldots, f^q(\vec{x})) \quad s.t.\ \vec{x} \in X \tag{2.2}$$

Note that there is no loss of generality in considering the objective function's minimization, since the maximization case can be reduced to this one. The criteria to be minimized are the so-called **objectives**. In general, there is not a single solution to the multiobjective problem which is simultaneously optimal for all objective functions. Hence, in this context the concept of optimality is replaced by the concept of Pareto optimality, and the solutions to a multiobjective problem are called **Pareto-optimal**, non-dominated or **efficient solutions**.

Definition 2.7 *A particular solution to a Multiobjective Optimization Problem is said to be **Pareto-optimal** or **Pareto-efficient** if no other solution to the problem can improve according to one objective without worsening at least one of the others.*

Definition 2.8 *The set of solutions to a Multiobjective Optimization Problem, known as **efficient set**, **Pareto frontier** or **Pareto set**, comprises all the feasible Pareto-optimal solutions to the problem.*

The main difference between a Multiobjective Optimization Problem and its scalar counterpart is the use of cost vectors which induce only a partial order relation. We will now reproduce some standard definitions regarding preference relations between q-dimensional cost vectors $\vec{y}, \vec{y'} \in \mathbb{R}^q$.

Definition 2.9 *A partial order relation $\prec$ denominated **dominance** or Pareto-optimal preference is defined as follows:*

$$\vec{y} \prec \vec{y'} \quad \Leftrightarrow \quad \forall i\ (1 \leq\ i \leq\ q) \quad y_i \leq y'_i\ \wedge\ \vec{y} \neq \vec{y'} \tag{2.3}$$

and we define equivalently the preference relation $\preceq$ called *dominance or equality*

$$\forall \vec{y}, \vec{y'} \in \mathbb{R}^q \quad \vec{y} \preceq \vec{y'} \quad \Leftrightarrow \quad \vec{y} \prec \vec{y'}\ \vee\ \vec{y} = \vec{y'} \tag{2.4}$$

where y_i denotes the i-th component of vector $\vec{y}$.

Therefore, given two q-dimensional vectors $\vec{y}$ and $\vec{y'}$ (where $q > 1$), it is not always possible to say that one is preferred to the other. For instance, in a three-dimensional cost space, vector $(2, 3, 1)$ dominates $(5, 6, 3)$ and $(3, 6, 4)$ but no dominance relation exists between $(2, 3, 1)$ and $(1, 7, 3)$ or $(5, 2, 7)$. They are all said to be *non-dominated*.

The Pareto-optimal solutions to the minimization problem can be split into two different types:

Supported They 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

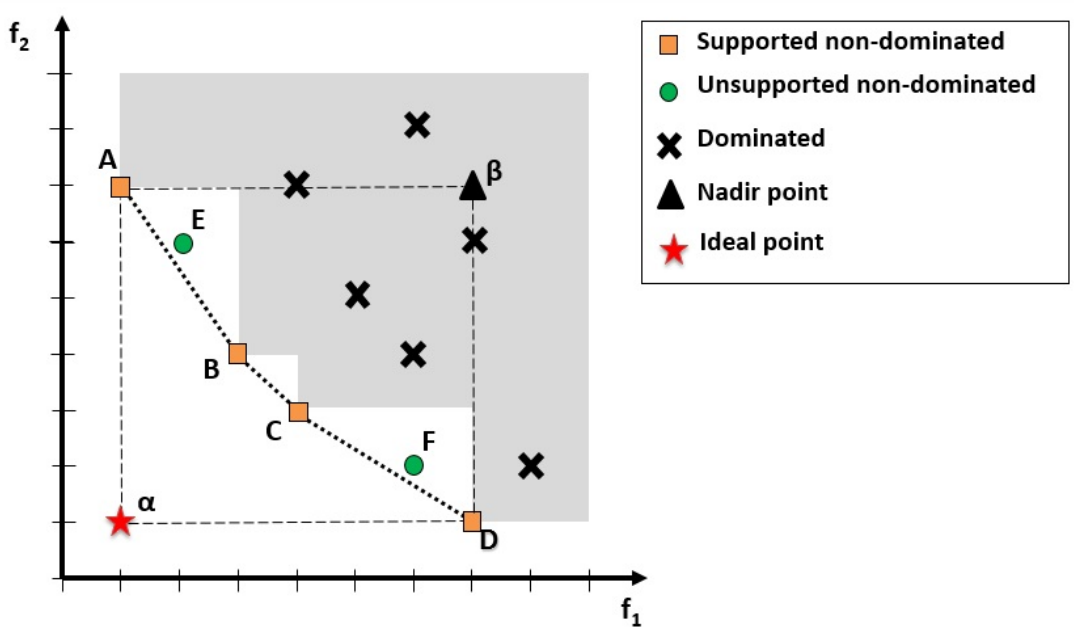


Figure 2.1. Types of solutions and relevant points in a biobjective cost space.

$\vec{x} = (x_1, x_2)$, these include all optima of the following function, for all values of the λ_1 and λ_2 parameters.

$$\min_{x \in X} \lambda_1 x_1 + \lambda_2 x_2 \tag{2.5}$$

The set of all supported efficient solutions is denoted by X_S, and the set of non-dominated image values by F_S.

Unsupported The remaining non-dominated solutions are called *unsupported* solutions. They cannot be obtained as solutions to WSPs. Unsupported solutions are located in the interior of triangles formed by two adjacent supported solutions. These areas are denominated *duality gaps* by some authors.

Let us take a look at Figure 2.1, where a sample bidimensional image space is depicted. Points $A = (1, 7)$, $B = (3, 4)$, $C = (4, 3)$, and $D = (7, 1)$ represent supported solutions. The *extreme* non-dominated solutions are supported solutions that have the minimum possible value in at least one of the objectives (points A and D). Points $E = (3, 5)$ and $F = (6, 2)$ represent unsupported non-dominated solutions.

Definition 2.10 *Given a set of vectors X, we shall define $\mathcal{N}(X)$ as the* ***set of non-dominated vectors*** *in X in the following way:*

$$\mathcal{N}(X) = \{\vec{x} \in X \mid \nexists \vec{y} \in X \quad \vec{y} \prec \vec{x}\} \tag{2.6}$$

Definition 2.11 *We say that a vector $\vec{x}$ is dominated by a set X when there exists $\vec{x'} \in X$ such that $\vec{x'} \prec \vec{x}$. Thus, all points represented as crosses in Figure 2.1 are dominated by $N(X) = \{(1, 7), (3, 4), (4, 3), (7, 1)\}$.*

Definition 2.12 *Let us denote* $\alpha_i = min_{\vec{x} \in \mathcal{N}(X)}\{x_i\}$, *and* $\beta_i = max_{\vec{x} \in \mathcal{N}(X)}\{x_i\}$. *The set* $\mathcal{N}(X)$ *is bounded by the* ***ideal point*** $\vec{\alpha} = (\alpha_1 \dots \alpha_q)$, *and the anti-ideal or* ***nadir point*** $\vec{\beta} = (\beta_1 \dots \beta_q)$. *The ideal point can be calculated optimizing each objective separately. However, for* $q > 2$ *it is difficult to calculate the nadir point without computing the whole set of non-dominated solutions.*

In Figure 2.1 the ideal and nadir points are represented as a red star and a black triangle, respectively. The set $\mathcal{N}(X)$ is bounded by the extreme non-dominated solutions A and B, the ideal point (α) and the nadir point (β).

Since there usually exist multiple Pareto-optimal solutions for Multiobjective Optimization Problems, there are different approaches in the literature to the concept of "solving a Multiobjective Optimization Problem". This concept is often understood as approximating or obtaining all or a representative set of Pareto-optimal solutions (Ehrgott, 2005). Approaches can be also divided into those methods providing approximate solutions, and those providing exact solutions.

There is a great variety of approximate techniques applied to the Multiobjective Optimization Problem e.g. fuzzy, and metaheuristics. Within them we can find a large amount of approaches like genetic algorithms, tabu search, or ant colony systems to name a few. A classification of all types of approaches to solve MOP is out the scope of this thesis. Some classifications can be found in (Ehrgott & Gandibleux, 2000; Ehrgott, 2005; Branke et al., 2008).

Some popular exact approaches combine multiple objectives into one single objective. The most used one is the weighted sum or scalarization method. This method along with Multiobjective Linear Programming (MOLP) is often included within the exact algorithms, although only the set of supported solutions can be found.

To classify exact algorithms that tackle the MOP we follow the general classification of MCDM methods, which can also be extended to MOP, proposed and followed by different authors over the years (Cohon, 1978; Clímaco & Pascoal, 2012). The following categories distinguish the methods according to the role of the decision maker in the resolution process:

1. **A priori methods**. The first category comprises those methods in which the preferences of the DM have been provided a priori. These preferences might be given for instance in the form of an utility function or goals to satisfy. Compromise Programming (CP) and Goal Programming (GP) are the most popular techniques included in this category. Some authors also believe than GP could be included in the next category (Jones & Tamiz, 2010), since it can be correctly used as a part of the interactive process to develop and refine the model accurately to reflect the decision maker preferences.

2. **Interactive decision process**. The decision maker can also be involved directly in the solution process. The methods which lie in this category establish a dialogue between the DM and the algorithmic resolution. If the DM is satisfied with the non-dominated solution provided, the algorithm finishes; otherwise, the calculation phase continues and the DM may set new preferences.

3. **A posteriori methods**. The third category includes methods where a multiobjective analysis is performed. There are no preferences given by the decision

maker and therefore, the whole set of efficient solutions must be calculated. Afterwards, this set is analyzed by the decision maker. It is worth mentioning that returning the whole Pareto frontier can be a computationally costly procedure. Therefore, these methods may have prohibitive runtimes when they are applied to large problems.

The two mentioned a priori approaches are MCDM distance function methods. Actually, both methodological approaches aim to minimize the distance, not in a geometric sense but in a preferential one, between a certain point and the actual achievement for each of the objectives under consideration. This point is represented in GP as a set of targets. Analogously, CP uses the *ideal point* (Definition 2.12) which corresponds to the optimum value of each objective.

Let us now turn our attention to the Goal Programming methodology.

2.3 Goal Programming

Goal programming is one of the oldest and most successful multicriteria decision making techniques (Chankong & Haimes, 1983). GP was born in the beginning of the 1960s thanks to the contributions of Charnes & Cooper (1961). Many variants and an impressive number of applications followed, enumerating hundreds of papers dealing with a wide range of problems and applications (Ignizio, 1976, 1978; Ignizio & Thomas, 1984; Charnes & Cooper, 1977; Zeleny, 1981, 1982, 1984; Zanakis & Gupta, 1985; Romero, 1986, 1991; Schniederjans, 1995; Tamiz et al., 1998; Aouni & Kettani, 2001). Several surveys have also been recently presented to review the improvements in the field (Caballero et al., 2009; Orumie & Ebong, 2014; Aouni et al., 2014).

Currently, GP has evolved from its original form into a powerful methodology that may incorporate techniques from artificial intelligence, such as genetic algorithms (Pal et al., 2012; Deb, 1999) or fuzzy logic (Bankian-Tabrizi et al., 2012; Shahnazari-Shahrezaei et al., 2013; Sen & Pal, 2013; da Silva & Marins, 2014).

Two key concepts serve to distinguish GP from conventional methods of optimization. First, the use of goals, or flexible constraints, as opposed to the rigid constraints of single objective optimization in mathematical programming, and second, the philosophy of "satisficing" as opposed to optimizing. The term satisfice was introduced by Simon (1956) as a combination of the terms satisfy and suffice.

As a consequence of the principle of satisficing, any solution to a goal programming problem is ranked by an achievement function which measures the degree of deviation from the problem goals when these goals can not be satisfied. The specific way in which this deviation is measured characterizes the particular model of the GP approach that is employed.

It must be emphasized that, despite the popularity of the GP model, a weakness have been pointed out from the DM point of view. In general, GP methods do not guarantee that the obtained solution is Pareto-optimal, though some tests of Pareto optimality can be found in Miettinen (1998); Larbani & Aouni (2007).

2.3.1 Variants of goal-based preferences

Section 2.1 introduced the concepts of attribute, objective, target and deviation variables. Following those concepts, we introduce below the most popular GP formulation models. At this point, it is important to mention that inequalities have been traditionally used in mathematical programming models to define the set of feasible solutions to a problem. From a mathematical point of view, goals and constraints share the same syntax. However, their semantics are quite different. Constraints must be satisfied by candidate solutions to be acceptable. If all constraints cannot be simultaneously satisfied, the problem is inconsistent and there is no solution. Goals, on the other hand, are used to represent the decision maker's preferences. They represent their desires or aspirations. Hence, feasible solutions may or may not achieve all goals. A solution to a goal problem is *satisfactory* when *all* the goals can be satisfied. If there are no satisfactory solutions to a problem, GP seeks solutions that minimize deviation from the targets.

Romero (1991) divides GP formulations in two categories. Both GP formulations do not generate the same solution, neither is one method superior to the other, because each variant is designed to satisfy certain decision makers' preferences.

1. **Weighted Goal Programming (WGP)**. The first approach, called weighted goal programming (WGP), considers all goals simultaneously as they are all in a composite objective function. This function aims to minimize the sum of all the deviations between the goals and their aspirational levels. The deviations are weighted according to the relative importance of each goal for the DM.

   In WGP weights are associated to each of the goals to establish the relative importance of deviations from their target. WGP handles several goals simultaneously by adding their relative deviations, hence, a normalization of constants is generally required in this model. Popular choices to normalize deviations are the goal target values (hence turning all deviations into percentages) or the range of the corresponding attribute (between the best and the worst possible values, hence mapping all deviations onto a zero-one range).

2. **Lexicographic (or pre-emptive) Goal Programming (LGP)**. The second approach, called pre-emptive or lexicographic goal programming (LGP), requires ranking all the goals in order of importance. The different goals are divided into several levels of pre-emptive priorities in such a way that if a specific priority level Q_i is preferred to another priority level Q_{i+1}, then the fulfillment of the goals in Q_i is infinitely more important than the fulfillment of the goals in Q_{i+1}.

   This variant models a problem with a combination of both weighted and lexicographic approaches. This model can suit the decision maker preferences when the attributes can be categorized into groups where the attributes within a group k are infinitely more important to satisfy than those of level $k+1$. Each level can have one or more attributes and their importance within the level is established by a certain weight also defined by the DM.

We further analyze lexicographic goal-based preferences in the next section.

2.3.2 Lexicographic goal-based preferences

This section introduces a formal characterization of lexicographic goal-based preferences that will be later used in our formulation. Let us first recall that Goal Programming is a distance function method, that is, it aims to minimize the distance or deviation from the actual solution achievement to the point that represents the goals established by the DM. Therefore, the way to define and measure this deviation is an important component of the GP model. Let us now consider a problem, where goals are always of the form $g_i(\vec{x}) \leq t_i$. Let $\vec{g} = (g_1, g_2, ..., g_q)$ be a vector of $1 \leq i \leq q$ attributes (costs) of a given solution $\vec{x} \in X$. Several methods have been proposed to measure the deviation of a solution vector from a set of goals. The most frequent ones are the following (Romero, 1991, 1993):

- minimization of the weighted sum of deviations:

$$d(\vec{g}) = \sum_{i=1}^{q} w_i \times \max(0, g_i - t_i), \tag{2.7}$$

- minimization of the maximum weighted deviation, also called minmax strategy:

$$d(\vec{g}) = \max_i [w_i \times \max(0, g_i - t_i)]. \tag{2.8}$$

Notice that this definition measures the deviation of the vector g with respect to the set of targets defined by the DM, i.e. corresponds to the aggregation of the non-achievement values of each individual goal $g_i - t_i$ (see Definition 2.4). We do not consider deviation when a goal is satisfied.

In the lexicographic goal-based model formulation we introduce below, we build upon the deviation measured as the minimization of the weighted sum of deviations. Let $\vec{g} = (g_1, g_2, ..., g_q)$ be a vector of $1 \leq i \leq q$ attributes of a given solution $\vec{x} \in X$, where $g_i : X \to \mathbb{R}, 1 \leq i \leq q$ grouped in l priority levels sorted in order of decreasing preemptive importance. Each priority level k comprises a set I_k of one or more attributes. Goals are defined by setting *targets* t_i for each attribute, always in the form $g_i(x) \leq t_i$.

This model considers the minimization of the weighted sum of deviations for each priority level, i.e. we measure the deviation of a solution vector from a set of goals grouped in preemptive priority levels of importance. We can calculate a deviation vector for $\vec{g}$ with one component for each priority level, $\vec{d}(\vec{g}) = (d_1(\vec{g}), d_2(\vec{g}), ..., d_l(\vec{g}))$. For each level k, its deviation d_k can be defined as:

$$d_k(\vec{g}) = \sum_{i \in I_k} w_i \times \max(0, g_i - t_i) \tag{2.9}$$

where w_i is the relative weight of goal i in level k.

Let us now introduce some more relevant preference relations between vectors that will be employed throughout this research work.

Definition 2.13 *Let us consider two q-dimensional vectors $\vec{y}, \vec{y'} \in \mathbb{R}^q$. A total order relation $\prec_L$ denominated* ***lexicographic order*** *is defined as follows:*

$$\vec{y} \prec_L \vec{y'} \quad \Leftrightarrow \quad \exists j \ (1 \leq \ i \leq \ q) \ y_j < y'_j \ \wedge \ \forall i < j \ \ y_i = y'_i. \tag{2.10}$$

and the preference relation $\preceq_L$:

$$\forall \vec{y}, \vec{y'} \in \mathbb{R}^q \quad \vec{y} \preceq_L \vec{y'} \quad \Leftrightarrow \quad \vec{y} \prec_L \vec{y'} \quad \vee \quad \vec{y} = \vec{y'} \tag{2.11}$$

Definition 2.14 *Let us consider two q-dimensional vectors $\vec{y}, \vec{y'} \in \mathbb{R}^q$. A total order relation $\prec_{lin}$ denominated **linear aggregation order** is defined as follows:*

$$\vec{y} \prec_{lin} \vec{y'} \;\Leftrightarrow\; \sum_i y_i \;<\; \sum_i y'_i \;\;,\; 1 \;\leq\; i \;\leq\; q \tag{2.12}$$

where y_i denotes the i-th component of vector $\vec{y}$.

Definition 2.15 *Let us consider two q-dimensional vectors $\vec{y}, \vec{y'} \in \mathbb{R}^q$. A total order relation $\prec_{wlin}$ denominated **weighted linear aggregation order** is defined as follows:*

$$\vec{y} \prec_{wlin} \vec{y'} \;\Leftrightarrow\; \sum_i \lambda_i y_i \;<\; \sum_i \lambda_i y'_i \;\;,\; 1 \;\leq\; i \;\leq\; q \tag{2.13}$$

where $\vec{\lambda}$ stands for the vector of weights and y_i, λ_i denote the i-th component of vectors $\vec{y}, \vec{\lambda}$.

A useful property of the previously mentioned orders (lexicographic, linear aggregation and weighted linear aggregation) is that their optimum in a set of vectors is also a non-dominated vector. The order relations defined by $\prec_L$, $\prec_{lin}$ or $\prec_{wlin}$ are total orders. Then, given two different q-dimensional vectors $\vec{y}$ and $\vec{y'}$ (where $q > 1$), it is always possible to say that one is preferred to another. For instance, in a three-dimensional cost space there is no dominance relation between $(1, 2, 4)$ and $(3, 2, 1)$. However, $(1, 2, 4) \prec_L (3, 2, 1)$ and $(3, 2, 1) \prec_{lin} (1, 2, 4)$, since $3 + 2 + 1 < 1 + 2 + 4$.

Definition 2.16 *We define the **optimum achievement** vector $\vec{d^*} = (d_1^*, d_2^*, ..., d_l^*)$ as the minimum lexicographic deviation vector among all solutions. Thus, the set of satisfactory solutions consists of all feasible solutions with a deviation equal to $\vec{d^*}$. If there is at least a satisfactory solution, then the optimum achievement vector is equal to $\vec{0}$.*

Let us now address the issue of optimality. In general, GP methods do not guarantee that the obtained solution is Pareto-optimal or non-dominated. However, in this thesis we are concerned with methods or algorithms able to ensure that the solution to the GP model is also a non-dominated solution. Therefore, all the new algorithms devised and presented in this thesis lie in this category, defining a solution to the GP model as the set of all non-dominated solutions that satisfy the DM goals, or the set of non-dominated solutions that minimize the deviation from goals if these cannot be satisfied.

Let us now define an order relation based on lexicographic goal preferences to guarantee our GP model only provides Pareto-optimal solutions,

Definition 2.17 *We define **lexicographic goal preferences** ($\prec_G$) as a partial order relation,*

$$\vec{y} \prec_G \vec{y'} \quad \Leftrightarrow \quad \vec{d}(\vec{y}) \prec_L \vec{d}(\vec{y'}) \;\; \vee \;\; (\vec{d}(\vec{y}) = \vec{d}(\vec{y'}) \;\; \wedge \;\; \vec{y} \prec \vec{y'}) \tag{2.14}$$

It is easy to see that $\prec_G$ is a strict partial order (it is irreflexive and transitive).

Definition 2.18 *Given a set of vectors X, we shall define $\mathcal{O}_G(X)$, the* ***set of optimal vectors*** *in X* ***according to lexicographic goal preferences*** *(i.e.* goal-optimal *vectors), as:*

$$\mathcal{O}_G(X) = \{\vec{x} \in X \mid \nexists \vec{y} \in X \quad \vec{y} \prec_G \vec{x}\} \tag{2.15}$$

Notice that an optimal solution according to $\prec_G$ is also a non-dominated solution, i.e. $\mathcal{O}_G(X) \subseteq \mathcal{N}(X)$.

One of the main motivations of this thesis is the application of the GP approach to the Multicriteria Search Problem. In the next sections we introduce the Shortest Path Problem and its variant considering multiple objectives or criteria.

2.4 The Shortest Path Problem

The Shortest Path Problem (SPP) has been extensively studied for many years by the Artificial Intelligence (AI) and Operational Research (OR) communities, e.g. see (Pearl, 1984; Gallo & Pallottino, 1988; Ahuja et al., 1990; Cherkassky et al., 1996). In graph theory, the single-pair shortest path problem is the problem of finding a feasible path between two nodes (or vertices) such that the sum of the weights of its component arcs (or edges) is minimized. Finding the fastest route on a road map from one location to another is an example of the shortest path problem. In this case, the nodes represent locations and the arcs represent roads which are labeled with the cost to traverse them.

Many real problems can be modeled as finding the shortest path between two nodes in a graph. Let us see a formal description of the problem.

Let $G = (N, A, c)$ be a locally finite labeled directed graph, defined by a set of N nodes, and a set of $A = \{(i_i, j_1), ..., (i_m, j_m)\} \subseteq N \times N$ arcs, where positive values $c_{ij} \in \mathbb{R}^+$ are associated with each arc $(i, j) \in A$. Sometimes we will denote c_{ij} as $c(i, j)$.

Definition 2.19 *A* ***path*** *P in G is any sequence of nodes $P = (n_1, n_2, ...n_k)$ such that $n_i \in N$ and for all $i < k$, $(n_i, n_{i+1}) \in A$. The set of all possible paths in G is denoted by $\mathbb{P}$.*

Definition 2.20 *The* ***cost of a path*** *P is defined as the sum of the costs of its component arcs,*

$$c(P) = \sum_{(i,j)\in P} c(i, j) \tag{2.16}$$

Definition 2.21 *The* ***Shortest Path Problem*** *over a graph consists in finding the path $P \in \mathbb{P}$ in G with the minimum cost $c(P)$ from a given a start node $s \in N$ to a destination[1] node t.* [2]

[1] Graph search literature often refers to this node as the *goal* node. However, in this research work we refer to it as destination node to avoid overloading the meaning of the word "goal".

[2] In the literature SPP algorithms can be split into one-to-one and one-to-all, depending if they seek to find the optimal path to a given destination node or to all nodes in the graph. In this thesis, we are concerned with the one-to-one shortest path problem.

Shortest path algorithms can be classified in terms of different properties like the strategy used to explore the search graph, their admissibility, i.e. whether the returned solution is exact or approximate, or the use of distance estimates, among others. A complete classification and description of the different strategies can be found in Korf (2010).

Most works on shortest path algorithms from the AI field distinguish two broad kinds of solution strategies:

Best-first algorithms Best-first strategies are a general approach to solve graph search problems. Their main feature is selecting the best alternative at each step of the algorithm. Whenever several paths reach the same node, only the best one is preserved whereas the rest are discarded (or pruned). Although they are generally fast algorithms, their main drawback is that all promising alternatives must be kept in memory, which can easily exhaust memory resources for difficult problems.

Depth-first algorithms (Korf, 1985) These strategies only consider the best next alternative, not keeping in memory other feasible alternatives which can be part of the optimal path. This leads to backtrack whenever the currently explored path cannot lead to a feasible solution. Depth-first strategies are specially suitable for tree problems and generally applied when the available memory resources are limited.

Best-first search (BFS) algorithms generally expand the fewer nodes among all exact algorithms using the same cost function, but may require exponential space with solution depth. Depth-first search algorithms need space only linear in the maximum search depth, but generally expand more nodes than BFS.

Dijkstra's algorithm (Dijkstra, 1959) is the most popular algorithm in this sense for one-to-one shortest path problems, i.e. finding the shortest path from a source node to a destination node in a graph. However, the reference algorithm in best-first search is A^* (Hart et al., 1968), which can be considered a generalization of Dijkstra's algorithm that uses *heuristic knowledge*, or distance estimates, to focus the search on the most promising alternatives.

The term *heuristic* is employed by the AI community to denote techniques that exploit knowledge about the problem to improve their efficiency. On the other hand, the OR community uses the term heuristic to denote methods used to speed up the process of finding good approximate solutions. In order to avoid confusion, algorithms that may return exact solutions like A^* are frequently categorized as "heuristic search" in the AI literature. In this research work we refer to the terms "heuristic function" or "heuristic estimates" used in the context of A^*-like algorithms simply as "distance estimates".

As stated above, the main feature of best-first algorithms is selecting at each step of the algorithm the most promising node n according to a certain evaluation function. Let $g(n)$ denote the accrued cost of a path from s, the source node, to n. Dijkstra's algorithm uses $g(n)$ as the evaluation function, i.e. it selects the node with the lowest value of $g(n)$ to scan next. The A^* algorithm uses distance estimates to predict how

close is the evaluated node to the destination node. This function is denoted by $h(n)$. Thus, the evaluation function employed by A^* is $f(n) = g(n) + h(n)$.

Best-first algorithms keep all the recorded alternative paths with source in s stored in a search tree. Efficient selection of the current best candidate for expansion is typically implemented using a priority queue ordered by the evaluation function. Each alternative path is labeled with the cost of the path, and additionally a reference to the parent node to recover the solution path at the end of the algorithm. The labels still pending to be explored are typically stored in a queue of $OPEN$ alternatives. A distinct set of $CLOSED$ nodes can be used to store already evaluated alternatives, in order to perform duplicate detection.

Definition 2.22 *A **node 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 value, that stands for the cost of the current best known path to n.*

At each step the label of some node n_i is selected and scanned (or expanded), and all the possible extensions via outgoing arcs of node n_i are generated and compared with actual stored labels in successor nodes. The stored cost for an adjacent node n_j is updated whenever the extension of the selected path through some outgoing arc of n_i represent a least cost path to the node n_j.

In the OR literature, shortest path algorithms are frequently classified as **label-setting** or **label-correcting**. For example, Dijkstra's algorithm (Dijkstra, 1959) is label-setting, since every extended path becomes permanent. This path represents the least cost to the node, i.e. once an alternative is scanned and extended, it is guaranteed that the least cost path to this node has already been found. In label-correcting algorithms, e.g. see (Zhan & Noon, 2000), this is not guaranteed until all nodes have been examined, since the label of an already explored node can be corrected by a new better one.

Label-setting algorithms repeat an iterative node labeling process until the destination node is selected for extension[3]. Then, the optimal cost of the path is found in the node label. The actual optimal path can be recovered tracing back pointers to parents.

In Dijkstra's algorithm, any permanent node label stands for the optimal shortest path distance from s to the node. Therefore, a "closed" node will never be re-opened. In the case of A^*, a permanent label may store the optimal path or not depending on properties related to the distance estimate function $h(n)$. Hence, let us now review the formal properties of $h(n)$ relevant to A^*.

Definition 2.23 *Let $h^*(n)$ be the actual optimal cost of a path from n to a destination node t. A distance estimate function $h(n)$ is **optimistic** when*

$$h(n) \leq h^*(n) \quad \forall n \in N \tag{2.17}$$

Definition 2.24 *Let $k(n, n')$ denote the cost of an optimal path in G from a node n to another node n'. A distance estimate function $h(n)$ is **consistent** when*

$$h(n) + k(n, n') \leq h(n') \quad \forall n, n' \in N \tag{2.18}$$

[3]We refer to the one-to-one version of Dijkstra's algorithm. In the one-to-all version, the algorithm will not finish until all nodes have been visited.

Definition 2.25 *Equivalently, a distance estimate function $h(n)$ is said to be* ***monotone*** *when*

$$h(n) + c(n, n') \leq h(n') \quad \forall (n, n') \in A \tag{2.19}$$

Definition 2.26 *A distance estimate function $h_2(n)$ is said to be* ***more informed*** *than another distance estimate function $h_1(n)$ when both are optimistic and*

$$h_2(n) > h_1(n) \quad \forall n \in N, n \neq t \tag{2.20}$$

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

Property 2.1 (Admissibility) *On finite graphs, when $h(n)$ is a lower bound of the cost of an optimal path from n to t (i.e., it is optimistic) A^* is* ***admissible****, i.e.* ***it is guaranteed to find an optimal solution if this solution exists, i.e. it is an exact algorithm****. A^* is* admissible *even on infinite graphs with some additional assumptions:*

$$\begin{aligned} \forall n \in N,\ h(n) \geq 0 \\ \forall (n, n') \in A,\ c(n, n') \geq \epsilon > 0 \end{aligned} \tag{2.21}$$

Property 2.2 (Efficiency) *When $\forall n \in N,\ h(n) = 0$, A^* is equivalent to the one-to-one version of Dijkstra's algorithm. When $h(n)$ is* ***consistent*** *or* ***monotone****, A^* is a label-setting algorithm, and requires, in the worst case $O(|N|)$ iterations, storing $O(|N|)$ nodes in memory. If the cost of the optimal solution is denoted by $c^* = k(s, t)$, 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 distance estimate function, more actual suboptimal alternatives can be pushed out the search frontier $f(n) = c^*$ with* ***more informed*** *functions, i.e. bigger values of $h(n)$, reducing search effort.*

Property 2.3 (Optimality) *When the distance estimate function $h(n)$ is* ***monotone****, A^* is proven to be* ***optimal*** *among the class of admissible best-first algorithms*[4] *(Dechter & Pearl, 1985), in the number of scanned nodes to find the solution. In other words, any node scanned by A^* must be also scanned by another algorithm in this class to preserve admissibility.*

2.5 The Multicriteria Search Problem

The problem of finding a shortest path from a source node to a destination node has been considered, traditionally, as a single-objective optimization problem. However, real decision problems frequently involve multiple conflicting criteria. Therefore, it seems natural to extend the SPP to the multicriteria case. Multicriteria preferences over paths in a graph can be expressed in the same ways already analyzed in the previous sections. For example, the solution to the Multiobjective Search Problem

[4]Defined as the class of single-pair unidirectional search algorithms.

is the set of the so-called Pareto-optimal solution paths. This was in fact the first Multicriteria Search Problem to be analyzed by Hansen (1980). Multicriteria Search Problems have a wide range of practical applications in different domains, arising naturally in many fields, such as:

- **Robot surveillance** (Delle Fave et al., 2009). The problem consists in how to automate surveillance tasks based on robotic platforms and fixed sensors. This surveillance automation is a complex problem posing many technical challenges, where robots must find a path to visit a set of locations equipped with alarms.

- **Transportation of hazardous materials**. The problem is concerned with finding routes with minimum cost and minimum risk (Erkut et al., 2007; Caramia et al., 2010; Machuca et al., 2011).

- **Satellite scheduling**. Gabrel & Vanderpooten (2002) proposed the problem of scheduling an Earth Observing Satellite as a Multicriteria Search Problem. A label-correcting was presented to take into consideration three objectives: demand satisfaction of shots taken by the satellite, priority of those shots (related to strategic importance), and satellite use (related to the number of instruments on/off).

- **Robot path planning**. The problem consists in finding a navigation path for a mobile robot that must consider multiple costs in navigation, such as the length of the path, traversability of the area, time to travel to the destination location, etc. (Fujimura, 1996). A similar problem deals with autonomous aerial vehicles and involves the motion planning that consists of a sequence of connected linear tracks (or trajectory segments) (Wu et al., 2011).

- **Route planning in different contexts**. In a road network problem several parameters (such as time, distance, economical cost, etc.) can be considered either the problem takes into account a single vehicle or a fleet (Jozefowiez et al., 2008; Delling & Wagner, 2009; Machuca et al., 2012).

- **Public transportation**. Multicriteria problems over an urban transportation network consider the minimization of the overall cost, time and users' discomfort (Modesti & Sciomachen, 1998). Raith (2009) presented the cyclist route choice as a new application of the bi-objective shortest path problem where cyclists aim to reach their destination in minimal travel time, but also along a safe route. Multicriteria search on time dependent public transportation systems is another application recently studied (Wu & Hartley, 2004; Müller-Hannemann & Schnee, 2004; Pyrga et al., 2008; Disser et al., 2008). These problems usually involve the minimization of travel time, bus-train changes, walking distance, etc.

- **QoS in networks**. In these problems the criteria to minimize may be, for example, number of lost packages, maximal delay time of packages, traffic crossing by a link, route length, probability of route unreliability, etc. (Climaco et al., 2003; Craveirinha et al., 2009).

We will introduce some useful definitions before presenting a formal definition of the Multiobjective Shortest Path Problem. Let $G = (N, A, \vec{c})$ be a locally finite labeled directed graph, defined by a set of N nodes, and a set $A = \{(i_i, j_1), ..., (i_m, j_m)\} \subseteq N \times N$ of arcs, where q positive costs $\vec{c}_{ij} = (c^1_{ij}, ..., c^q_{ij}) \in \mathbb{R}^{q+}$ are associated with each arc $(i, j) \in A$. Sometimes we will denote $\vec{c}_{ij}$ as $\vec{c}(i, j)$.

Let a path in G be any sequence of nodes $P = (n_1, n_2, ...n_k)$ such that for all $i < k$, $(n_i, n_{i+1}) \in A$. A problem over a multicriteria graph is defined by a start node $s \in N$, and a destination node t. Each path in the graph of the form $P = (s, ..., n_i, n_{i+1}, ..., t)$ represents a feasible solution to the problem.

Definition 2.27 *The **cost of a path** P in multiobjective shortest path problems is a q-dimensional vector defined as the sum of the costs of its arcs,*

$$\vec{c}(P) = \sum_{(i,j)\in P} \vec{c}(i, j) \tag{2.22}$$

Definition 2.28 *Given a start node $s \in N$ and a destination node $t \in N$, the Multiobjective Shortest Path Problem consists in finding all the non-dominated paths from s to t. More precisely, the problem can be formulated mathematically as the following network flow problem [adapted from Raith & Ehrgott (2009)]*

$$min \quad \vec{f}(x) = \begin{cases} f_1(x) = \sum\limits_{(i,j)\in A} c^1_{ij} x_{ij}, \\ ... \\ f_q(x) = \sum\limits_{(i,j)\in A} c^q_{ij} x_{ij}, \end{cases} \tag{2.23}$$

$$s.t. \quad \sum_{(i,j)\in A} x_{ij} \quad - \quad \sum_{(j,i)\in A} x_{ji} = \begin{cases} 1 & \text{if } i = s, \\ 0 & \text{if } i \neq s, t, \\ -1 & \text{if } i = t \end{cases} \tag{2.24}$$

$$x_{ij} \in \{0, 1\}, \quad \text{for all } (i, j) \in A. \tag{2.25}$$

where x is a vector of flows on the arcs, and the constraints (2.24) represent flow balance at the different nodes.

As stated above, many problems from a wide variety of fields can be formulated as Multicriteria Search Problems. There are not less algorithmic approaches to cope with these problems. As explained in Section 2.2, algorithmic approaches to this problem can be divided into exact and approximate.

Multicriteria Search Problems are indeed a particular case of the MCDM problems, hence, the classification presented in Section 2.2 for the MOP can also be extended to them. A detailed description of all types of approaches to solve MSP is out the scope of this thesis. Some classifications can be found in Ehrgott & Gandibleux (2000); Ehrgott (2005); Tarapata (2007); Clímaco & Pascoal (2012).

Let us now review some of the exact algorithms that have been proposed to deal with MSP categorized according to the role of the decision maker. We omit the interactive methods described in Section 2.2, since they are out of the scope of this thesis. Some proposed methods can be found in Current et al. (1990); Granat & Guerriero (2003).

First, we will review with some detail exact algorithms with a posteriori preferences. These are used when no a priori preferential information is available. Basically, these algorithms find the set of all Pareto-optimal solutions to the problem, so that the most preferred ones can be selected a posteriori. We will pay special attention to the NAMOA* search algorithm, which uses lower bound estimates to efficiently find the set of all Pareto-optimal paths to a problem. On the contrary, a priori algorithms have preferential information from the decision maker. These may be expressed, for example, in the form of a set of goals. A basic approach to solve this problem is to use a posteriori algorithms, and then select the preferred ones among them. However, sometimes it is also possible to devise algorithms that find only the preferred alternatives. In this thesis, we will analyze both approaches in the solution of the shortest path problem with lexicographic goal preferences.

2.6 Exact a posteriori algorithms

Four categories of exact algorithms without a priori preferences from the DM have been identified by different authors (Skriver & Andersen, 2000; Raith, 2009; Clímaco & Pascoal, 2012; Machuca, 2012). Three of them are generalizations from single-objective shortest path problems, these are label-setting, label-correcting and k-th shortest paths; the last one is a two-phase approach.

A **label** (or node label) stands for the cost of a path found from the start to a given node. In single-objective search, a node n is labeled with a single value that stands for the cost of the current best known path to n. In multiobjective search algorithms, multiple non-dominated paths can reach a node n, therefore, n is not labeled with a single cost but with a set of node labels which stand for the non-dominated cost vectors of known paths that reach node n.

Label-setting methods The most classical single-objective shortest path algorithms follow a labeling method. Their name derives from the process of labeling each node with the cost of the best path found so far to reach that node. The multiobjective label-setting algorithms follow the same strategy, with the necessary adaptations to the multicriteria case. An iterative process scans alternatives and generates new successors until the stopping criterion is fulfilled. Then, the labels at the destination node represent efficient costs of solution paths. The first proposals of multicriteria label-setting algorithms can be found in (Hansen, 1980; Martins, 1984b).

Another interesting field of study for label-setting algorithms is to reduce space requirements, e.g. frontier search algorithms (Korf et al., 2005), which have also been extended to the multicriteria case (Mandow & Pérez de la Cruz, 2007, 2008a,b).

Label-correcting methods The main difference between label-correcting and label-setting algorithms is the label selection policy. Label-setting algorithms employ a best-first strategy whereas that label-correcting algorithms follow different strategies, e.g. FIFO. This different strategy does not allow these algorithms to guarantee that any label selected for expansion is optimal, therefore if a better path to some node is found the suboptimal labels will be "corrected" by the new one. An analysis of multicriteria label-correcting methods can be found in (Brumbaugh-Smith & Shier, 1989). Some analysis which compare label-correcting and label-setting approaches appear in (Skriver & Andersen, 2000; Guerriero & Musmanno, 2001; Paixão & Santos, 2013).

Ranking methods In a similar manner to k-best single-objective algorithms, multiobjective ranking algorithms employ a ranking method to list paths by a non-decreasing total order. However, there exist two necessary adaptations in these algorithms. First, an order must be imposed to rank solutions either using a linear combination of the objectives, or a partial order (e.g. lexicographic order). Second, the algorithm can not be stopped until all efficient solutions are guaranteed to be found. Moreover, the value of k is unknown a priori.

A complete literature review of ranking algorithms applied to the MSP can be found in (Martins et al., 2007; Clímaco & Pascoal, 2012). Some ranking algorithms appear in (Martins, 1984c; Azevedo & Martins, 1991; Martins et al., 2007; Paixão & Santos, 2008). Several recent studies, however, have found these algorithms not competitive with label-setting and label-correcting methods (Huarng et al., 1996; Skriver & Andersen, 2000; Raith, 2009).

Two-phase methods Algorithms from this category compute the extreme supported efficient solutions in a first phase. In a second phase the remaining unsupported efficient solutions are calculated with a labeling algorithm. Depending on the method chosen for the first phase an initialization phase may be necessary. They have been applied to the Bicriteria Search Problem (BSP) by Mote et al. (1991); Raith & Ehrgott (2009); Raith (2009). However, they have been found to be generally slower than the label-setting or label-correcting algorithms (Raith & Ehrgott, 2009; Raith, 2009).

In labeling search algorithms two different successor generation strategies have been applied, *node-selection* and *label-selection*. The *node-selection* strategy implies that all labels belonging to a node are expanded simultaneously, on the contrary, if only one label at a time is expanded it is called *label-selection*.

2.6.1 Extensions of A^* to the multiobjective case

Several extensions of A^* to the Multiobjective case have been proposed. The behavior of these algorithms does not easily fit in the usual OR categories, and depends largely on the properties of the distance estimate. Hansen (1980) presented a bi-objective extension of Dijkstra's label setting algorithm. Martins (1984a) proposed a general label-setting algorithm for the multiobjective case. Stewart & White (1991) proposed a multiobjective algorithm based on node-selection, called MOA*(Multiobjective A^*).

Tung & Chew (1992) proposed two multicriteria lower bound functions along with a label-setting multicriteria algorithm. Mandow & Pérez de la Cruz (2010) presented NAMOA*(A New Approach to Multiobjective A^*), a multiobjective algorithm that has been proven to be an exact algorithm when provided with lower bound estimates, and to explore an optimal number of labels.

Recent works studied the influence of lower bounds on multicriteria search algorithms. The thesis of Machuca (2012) evaluated the algorithms of Tung & Chew, MOA*, and NAMOA* with and without lower bound estimates. The main outcome obtained was the improvement, in general, of NAMOA* when provided with lower bounds, and its formal and empirical superiority over the other two algorithms when provided with lower bounds (Machuca, 2012, 2.4.3). A formal explanation for the superiority of NAMOA* over MOA* was presented recently by Pérez de la Cruz et al. (2013). This paper considers the performance of MOA* when provided with lower bounds, and analyzes a pathological behavior observed in MOA*, namely, that the efficiency of the algorithm decreases with more informed distance estimates. It is shown that in certain cases uninformed search is more efficient for MOA* than perfectly informed search, in terms of both node and label expansions. Thus, we consider MOA* obsolete and we will focus our attention in NAMOA*.

We present NAMOA* below, and we will further analyze the use of the ideal point as a multicriteria lower bound function. This was proposed by Tung and Chew and we will apply it to NAMOA*.

2.6.2 Algorithm NAMOA*

NAMOA* (Mandow & Pérez de la Cruz, 2005; Mandow & Pérez de la Cruz, 2010) is a successful generalization of A^* to the multiobjective case. NAMOA* is defined to accept a lower bound function $H(n)$ that returns a set of cost estimates of all non-dominated paths from node n to a node which belongs to a set of destination nodes. Table 2.2 shows the pseudocode of NAMOA* simplified to use a single vector estimate $\vec{h}(n)$ and one destination node t.

NAMOA* is a best-first algorithm that uses a *label-selection* strategy and builds a search graph SG rooted at the start node s to store all non-dominated paths found to each node. The set COSTS stores all known non-dominated solution costs to the destination node t. Each node n in the search graph has two sets of labels: $G_{op}(n)$ denotes the set of cost vectors of paths reaching n that can be further explored, while $G_{cl}(n)$ denotes the set of those that have already been expanded (in fact COSTS is equivalent to $G_{cl}(t)$) and we refer to them as closed or permanent (when the lower bound function satisfies certain conditions). Each cost vector in these sets labels one or more arcs in the graph from n to its parents. NAMOA* uses extended labels $(n, \vec{g}, \vec{f})$. By abuse of language, we refer to them as labels when there is no ambiguity.

Let $\vec{g}(P_{sn})$ be the cost of some path P_{sn} reaching some node n from the start node. Then, $\vec{h}(n)$ is a lower bound of the cost of any extension of such path to the destination node and $\vec{f}(P_{sn}) = \vec{g}(P_{sn}) + \vec{h}(n)$ the evaluation function used.

At each iteration, a label with a non-dominated $\vec{f}$ in OPEN is selected for expansion. If n is the destination node, then $\vec{f}$ is stored in COSTS, the set of non-dominated solutions costs. Otherwise, the non-dominated label $(n, \vec{g}, \vec{f})$ selected from OPEN is

Table 2.2. [Adapted from (Mandow & Pérez de la Cruz, 2010)] Pseudocode of NAMOA* algorithm.

1. CREATE:
   — An empty search graph SG, and place s as its root.
   — Two sets $G_{cl}(s) = \emptyset$ and $G_{op}(s) = \{(\vec{0})\}$.
   — A list of alternatives, OPEN $= \{(s, \vec{0}, \vec{h}(s)\}$.
   — An empty set, COSTS.
2. PATH SELECTION. If OPEN is not empty, then,
   — Select a label $(n, \vec{g}_n, \vec{f}_n)$ from OPEN ("PATH SELECTION STRATEGY") such that $\nexists(n', \vec{g}_{n'}, \vec{f}_{n'}) \in$ OPEN $\mid \vec{f}_{n'} \prec \vec{f}_n$.
   — Delete the selected label from OPEN, and move $\vec{g}_n$ from $G_{op}(n)$ to $G_{cl}(n)$.
   — If $\exists \vec{c^*} \in$ COSTS $\mid \vec{c^*} \prec \vec{f}_n$, then repeat step 2 (lazy filtering)
3. CHECK TERMINATION. If OPEN is empty, then backtrack in SG from t and return the set of solution paths with costs in COSTS.
4. SOLUTION RECORDING. If n is the destination node, then
   — Include $\vec{g}_n$ in COSTS.
   — Go back to step 2.
5. PATH EXPANSION: If n is not the destination node, then for all successor nodes m of n do:
   (a) Calculate the cost of the new path found to m and its lower bound, $\vec{g}_m = \vec{g}_n + \vec{c}(n, m)$, $\vec{f}_m = \vec{g}(m) + \vec{h}(m)$.
   (b) If $\nexists \vec{c^*} \in$ COSTS $\mid \vec{c^*} \prec \vec{f}_m$, ("FILTERING") then:
      i. If $m \notin$ SG:
         - Set $G_{op}(m) = \{(\vec{g}_m)\}$ and add $(m, \vec{g}_m, \vec{f}_m)$ to OPEN.
         - Label with $\vec{g}_m$ a pointer from n to m.
      ii. else if $\vec{g}_m$ equals some cost vector in $G_{op}(m) \cup G_{cl}(m)$ then
         - Label with $\vec{g}_m$ a pointer from n to m.
      iii. else if $\vec{f}_m$ is not dominated by $G_{cl}(m) \cup G_{op}(m)$ ("PRUNING"), then:
         - Eliminate vectors $\vec{g}_{m'} \in G_{op}(m) \mid \vec{g}_m \prec \vec{g}_{m'}$ and its corresponding label $(m, \vec{g}_{m'}, \vec{f}_{m'})$ from OPEN.
         - Add $(m, \vec{g}_m, \vec{f}_m)$ to OPEN, $\vec{g}_m$ to $G_{op}(m)$ and label with $\vec{g}_m$ a pointer from n to m.
   (c) Go back to step 2.

made permanent (i.e., g is moved from $G_{op}(n)$ to $G_{cl}(n)$). For each arc in the graph (n, n') with cost $\vec{c}(n, n')$, a new label is generated for node n' with cost $\vec{g'} = \vec{g} + \vec{c}(n, n')$. Several operations are carried out in NAMOA* for each such label:

1. The algorithm applies the optimality principle, i.e. only non-dominated labels to each node are considered. Therefore, the new label has to be checked for dominance against all labels in $G_{op}(n)$ and $G_{cl}(n)$. If the new label is dominated, then it is discarded. This operation is called *pruning*.

2. For each new label, its cost estimate $\vec{f'} = \vec{g'} + \vec{h}(n')$ is tested for dominance against vectors in COSTS. If the new estimate is dominated, it can never lead to a new non-dominated solution and is discarded. This operation is called *filtering*.

These operations are computationally costly. For the case with two objectives the label sets $G_{op}(n), G_{cl}(n)$ and COSTS can be ordered lexicographically, allowing for efficient dominance checks (Sanders & Mandow, 2013). However, for three or more objectives there is no known efficient way to check the dominance of a vector against a set.

A difference between the original pseudocode and this adaptation in Table 2.2 is the application of *lazy filtering*, as described in (Sanders & Mandow, 2013), i.e. we do not explicitly filter existing labels when a new solution is found. Labels are tested and, if necessary, filtered only after selection. This prevents a costly update operation.

NAMOA* shares important properties with A*. If the estimate function $\vec{h}(n)$ returns an optimistic estimate (lower bound) of the cost of *any* path from n to the destination, then it is guaranteed to terminate with the set of all non-dominated solution paths to the problem. If the lower bound function satisfies the so-called monotone property, then it explores only non-dominated labels and is optimal in the class of admissible algorithms, see Section 5.1 for further details.

2.6.3 The ideal point as lower bound

The use of lower bound functions has been studied for many years as a way to improve the efficiency of combinatorial optimization problem solving. In graph search problems a lower bound function estimates the cost of a solution path from node n to the destination node t. In MSP there is more than one criterion, therefore, this function does not return a single value but a cost vector estimate. Tung & Chew (1992) proposed a multicriteria lower bound function along with a label-setting multicriteria algorithm. The lower bound function represents the ideal point presented in Definition 2.12. This is indeed the most informed and admissible estimate function that can be defined for a MSP[5] and we will name it as $\vec{h}_\alpha$.

The lower bound function is defined as $\vec{h}_\alpha(n) = (c_1^*(n), c_2^*(n), \dots c_q^*(n))$, where $c_i^*(n)$ is the optimal scalar cost of a path from n to the destination node, considering only the *i-th* cost component. These values are precalculated by reversing all arcs in the graph, originally labeled with cost vector $\vec{c}(n, n') = (c_1, c_2, \dots c_q)$ the new labels will be of the form $c_i(n', n) = c_i$. Any single-objective one-to-all shortest path problem can be

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

applied to find the optimal cost from the destination node to all the other nodes in the reversed graph. This process will be executed q times, once for each scalar cost. Ties are broken by a lexicographic order, taking into account the other cost components.

The method proposed by Tung and Chew to calculate the $\vec{h}_\alpha$ function considered originally the estimation for all nodes in the graph. In medium and difficult multiobjective problems this precalculation time is minor in comparison with the runtime of the multiobjective algorithm. However, this time can be proportionally significant when the destination node is "close" to the source node. Machuca & Mandow (2012) presented a bounded calculation procedure for this lower bound function in bicriteria problems that avoids calculating the estimate for all nodes in the graph. They also compared the performance of NAMOA* over road map problems with three different lower bound functions, the corrected great circle distance, the distance estimate $\vec{h}_\alpha$, and bounded $\vec{h}_\alpha$. The latter clearly outperformed the other two.

Let us now review the equivalent formal properties of multicriteria lower bound functions $H(n)$ to the single-valued function. These properties can be briefly review in (Machuca, 2012, 2.1.2.1) or in-depth in (Pearl, 1984).

Definition 2.29 *(Mandow & Pérez de la Cruz, 2005, p. 184) A distance estimate function $H(n)$ is said to be admissible when for all non-dominated solutions $P^* = (s = n_0, n_1, \ldots, n_i, n_{i+1}, \ldots, n_l = t)$, and for all subpaths $P_i^* = (n_0, \ldots, n_i)$ of P^* the following holds:*

$$\exists \vec{h} \in H(n_i) \quad | \quad g(P_i^*) + \vec{h} \preceq g(P^*) \tag{2.26}$$

Definition 2.30 *(Mandow & Pérez de la Cruz, 2010, Definition 5.6) A multicriteria lower bound function $H(n)$ is **consistent** if for all pairs of nodes n, n' in the graph, for all non-dominated path between them $P = (n, \ldots, n')$, and for all lower bound cost vectors $\vec{h}' \in H(n')$, the following condition holds:*

$$\exists \vec{h} \in H(n) \quad | \quad \vec{h} \preceq c(P) + \vec{h}' \tag{2.27}$$

Definition 2.31 *(Mandow & Pérez de la Cruz, 2010, Definition 5.7) Equivalently, a multicriteria lower bound function $H(n)$ is **monotone** when for all arcs (n, n') in the graph, the following condition holds:*

$$\forall \vec{h}' \in H(n') \quad \exists \vec{h} \in H(n) \quad | \quad \vec{h} \preceq c(n, n') + \vec{h}' \tag{2.28}$$

Definition 2.32 *(Mandow & Pérez de la Cruz, 2010, Definition 5.4) A multicriteria lower bound 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,*

$$\forall \vec{h}_2 \in H_2(n) \quad \exists \vec{h}_1 \in H_1(n) \quad | \quad \vec{h}_1 \preceq \vec{h}_2 \tag{2.29}$$

2.7 Exact a priori algorithms

Finally, we will review relevant exact multicriteria algorithms proposed for the case where preferences between alternatives are provided a priori. More precisely, we review

the cases where preferences are expressed in terms of compromise solutions, or a set of goals. Both of these a priori techniques aim to minimize the distance between the DM preferences and the actual achievement for each of the objectives under consideration.

2.7.1 Compromise Search

A well known approach in MSP is the compromise solution method (Yu, 1985) or best compromise search. In this case, the preferences of the DM can be given as their aspiration level on each criterion. Then, a function is defined to evaluate a solution according to its distance to that preference, which is called the reference point. The best compromise solution is then the closest one to the reference point[6].

Achievement scalarizing functions are employed in Compromise Search in order to evaluate solutions. Let us take the following definition from (Machuca et al., 2013):

Definition 2.33 *A scalarizing function s must have the following properties: (1) any non-dominated solution can be optimal with respect to s (with an appropriate choice of parameters); and (2) any optimal solution with respect to s has to be non-dominated. These requirements ensure that, for any rational decision maker, the preferred solution can be reached optimizing some scalarizing function.*

The Tchebycheff norm, the Euclidean, Manhattan or Minkowski's are examples of distance functions to seek for the compromise search solution. However, there does not exist any scalarizing function satisfying simultaneously the two requirements (Wierzbicki, 1986). Even so, Compromise Search applied to MSP has been widely studied, and compared to algorithms where the whole set of non-dominated solutions is returned (Machuca et al., 2013).

Perny & Spanjaard (2005) introduced a general formalism for preference-based combinatorial problems, and proposed several algorithms to obtain the set of preferred solutions. Different measures are used in best compromise search to obtain the set of compromise solutions. The ordered weighted averaging (OWA) operator is a natural manner to express the preferences between solutions. Galand & Spanjaard (2007) proposed a best compromise search algorithm based on OWA, called OWA*. Galand et al. (2010) used the Choquet integral, another preference model used in decision theory for aggregating preferences, as a preference model. The Tchebycheff scalarizing function was employed by other recent studies as Perny & Weng (2010) and Galand et al. (2013). The latter, presented the first bidirectional multiobjective search algorithm, called bidirectional PBMOA*.

Galand & Perny (2006) presented two algorithms to find the best compromise solution paths, BCA* and kA*. BCA* is a variant of NAMOA* with an upper bound λ to store the value of the Tchebychev distance of the best compromise solution path found. Additionally, the algorithm incorporates a specific stopping condition. In the worst case, BCA* performs a similar enumeration of paths as NAMOA*, and finds the best compromise at the last step. On average, however, BCA* stops much earlier than NAMOA*. kA* enumerates the k best paths in a scalarized version of the

[6]The reference point does not need to be necessarily specified by the DM and it can also be defined as the ideal solution point (Zeleny, 1982). In this case, this approach could be included in the a posteriori category.

multiobjective graph. This algorithm relies on a modified version of A^* that works on labels attached to paths instead of labels attached to nodes.

Several differences between both algorithms can be pointed out. First, the use of vectors versus scalar values in BCA* and kA*, respectively. Second, BCA* prunes sub-optimal paths reaching the same node as well as those paths that according to their $\vec{f}$ value cannot lead to the best compromise solutions. kA*, however, cannot prune sub-optimal paths, since they should always be kept for future considerations. Finally, the comparison between both algorithms revealed a better performance of kA* in certain cases (Galand & Perny, 2006; Machuca et al., 2013). Nevertheless, kA* presents an exponential worst case behavior, i.e. it can explicitly explore a combinatorially large number of paths with the same cost vector.

Lastly, Sauvanet & Néron (2010) presented two improvements to speed-up the search of BCA*, applying that development to a bicycle routing application available online and for smartphones[7].

2.7.2 Goal Programming

Goal Programming has also been applied to the MSP. Different approaches to model a GP problem were presented in Section 2.3.1. One of them is Lexicographic Goal Programming. METAL-A*(Mandow & Pérez De La Cruz, 2001) was proposed as a general algorithm for graph search problems with additive lexicographic goals. In a way, it is an extension of A^* to goal-based multicriteria preferences.

Section 2.6.1 analyzed the proposed extensions of A^* to the multiobjective case. As we stated there, MOA* is considered obsolete due to the formal and empirical superiority of NAMOA* over MOA*. METAL-A* is based on MOA*, and subject also to the same faults detected in the work of Pérez de la Cruz et al. (2013) for MOA*, namely, that the performance of the algorithm decreases with better informed distance estimates. Therefore, this algorithm can also be considered deprecated, and will not be further considered in this research work. To our knowledge, there have been no further developments on MSP with Lexicographic Goals. This thesis addresses precisely this question. In Chapter 4 we will present two new algorithms based on NAMOA* to deal with lexicographic goals.

2.8 Summary and motivation

In this chapter we have reviewed the fundamental concepts of Multicriteria Decision Theory, with special attention to goal-based preferences. We have also reviewed the Multicriteria Shortest Path Problem and the main approaches proposed to date in the literature to solve it. All methods developed in this chapter have been mainly divided into approximate and exact techniques. In this thesis we focus on exact algorithmic techniques, i.e. those techniques seeking only non-dominated solutions, either the full set of the Pareto frontier, or the subset of Pareto-optimal solutions that satisfy the DM preferences.

[7]Website: `www.geovelo.fr,AndroidAPP:https://play.google.com/store/apps/details?id=fr.geoveloparis`

Four categories of exact a posteriori algorithms were identified in Section 2.6: labeling (label-setting and label-correcting), ranking, and two phases approaches. This thesis focuses on label-setting algorithms using lower bound estimates. We pay special attention to the Dijkstra and A^* algorithmic extensions to the Multiobjective case. In particular, we describe an improved dominance check method for NAMOA*.

Among a priori methods, we reviewed contributions in Compromise Search and Goal Based Search. Formulations for the WGP and the Lexicographic GP were presented. To our knowledge, the only previous work on goal-based preferences for SPP was based on MOA*, and there are no contributions in this sense based on NAMOA*. We will further describe in this thesis our contributions in the category of label-setting algorithms with lower bounds that employ a label-selection policy, and lexicographic goal programming algorithmic techniques. In particular, one of the main contributions of this thesis relies on an extension of NAMOA* to deal with lexicographic goal techniques. In those cases where the goals cannot be satisfied the distance to the goals is measured with the definition stated in 2.7, which minimizes the weighted sum of deviations. Table 2.3 displays a classification of representative a priori and a posteriori multicriteria shortest path algorithms with additive metrics.

This thesis aims to provide a contribution to the Multicriteria Search Problems with lexicographic goal-based preferences. In other words, the combination of the MSP with GP techniques. We define two alternatives to deal with such a problem. The first alternative approaches the problem from the perspective of an optimization problem. A full search of the Pareto frontier is performed to obtain all the non-dominated solutions, and afterwards extract the subset of satisfactory solutions for the provided goals. The second alternative approaches the problem from the satisfaction perspective. An algorithm which takes advantage of the preferences given by the DM is devised. Its purpose is to restrict the search and discard sooner in the search those paths which will not lead to satisfactory solutions. The first alternative is characterized as a posteriori method, while the second corresponds to the category of a priori methods.

In addition to define a new specifically designed algorithm to cope with MSP with lexicographic goal-based preferences (second alternative presented above), we aim to improve the runtime performance of algorithms based on the extension of A^* to the multiobjective case (label-setting algorithms within the first alternative). More specifically, we will apply the devised techniques to NAMOA*.

Time requirements have been identified as the limiting factor in the size of problems that can be practically solved by multiobjective shortest path algorithms (Machuca et al., 2009, 2012; Mali et al., 2012). Thus, we enumerate below several fundamental aspects to achieve better runtime performance in label-setting algorithms.

1. **Use of lower bounds.** In the same manner than A* outperforms Dijkstra's algorithm by using lower bounds, or estimates of the distance to the destination node, the label-setting multicriteria search algorithms can also take advantage of these bounds to discard sooner suboptimal paths and "guide" the search more efficiently. Recent studies have demonstrated that *node-selection* algorithms as MOA* does not share all the formal properties of A*. However, NAMOA* following a *label-selection* strategy does preserve all formal properties of A* (Mandow & Pérez de la Cruz, 2010), see Section 5.1.

Table 2.3. Classification of some representative a priori and a posteriori multicriteria shortest path algorithms.

Classification					Some representative algorithms
A posteriori	Labeling	Label-setting	MO Dijkstra		Hansen (1980); Martins (1984a)
			MO A^*	*Node-selection*	MOA*
				Label-selection	Tung & Chew (1992); NAMOA*; Vector frontier Search
		Label correcting			Brumbaugh-Smith & Shier (1989); Guerriero & Musmanno (2001)
	Ranking				Martins et al. (2007); Paixão & Santos (2008)
	Two phases				Mote et al. (1991); Raith & Ehrgott (2009)
A priori	CP	Tchebycheff norm			BCA*, kA*; Sauvanet & Néron (2010)
		OWA operator			OWA*, Bidirectional PBMOA*
		Choquet integral			Galand et al. (2010)
	GP	LGP			METAL-A*

2. **More efficient dominance checks.** Several adaptations are necessary to generalize algorithms to the multicriteria case. The most important factor to consider is the fact that a set of non-dominated labels need to be kept at a node, instead of a single label as in the single-objective case. Therefore, a great amount of dominance checks will be necessary to check if a new generated path must be discarded[8]. Thus, we propose in this thesis a new dimensionality reduction technique to speed up dominance checks in label-setting multicriteria search algorithms, see Section 4.2.

3. **Label selection policy in best-first algorithms.** Best-first algorithms employ a selection policy to choose between all non-dominated alternatives. We presented several total order relations in Section 2.3.2 which can be used as label selection policies, being the lexicographic and linear addition orders the most commonly used. Different studies have been made to compare the performance of label-setting algorithms employing these orders (Iori et al., 2010; Machuca et al., 2012). In the experiments presented in this thesis, label selection policies are also analyzed as an important parameter in the algorithm time performance.

4. **Use of more efficient data structures.** The use of efficient data structures for the SPP has always been a hot topic (Ahuja et al., 1990; Zhan, 1997; Cherkassky et al., 1999; Cazenave, 2006; Mehlhorn & Sanders, 2008). Nevertheless, the use of different data structures to sort the list of open alternatives in best-first MSP still deserves more research. We will further analyze this and other future improvements in Chapter 8.

[8]For bi-objective cases labels can be totally ordered by using the lexicographic order, reducing time requirements to determine whether a new label is non-dominated with existing labels, for instance, with a lexicographic order all labels in the set are sorted by the first objective, hence, a single comparison of the second objective of the new label and the best value for any label in the set will be enough to check the dominance of the new label against the whole set. However, for the general MSP case, all labels must be compared against the new one to check whether is dominated or not.

Chapter 3
Benchmarks

A goal is not always meant to be reached, it often serves simply as something to aim at.

Bruce Lee (1940-1973)

This chapter introduces relevant literature on the experimental evaluation of multiobjective search algorithms and describes the test sets used to assess the performance of the algorithms studied in this thesis. We survey some of the tools and test beds proposed in the past to test algorithmic improvements in the field, review the importance of connecting the theoretical analysis with empirical evaluation, and enumerate important factors to consider in the latter.

The chapter is organized as follows. First, a summary of related benchmarks 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 employed to test the algorithms presented in this thesis. Section 3.3 addresses the different factors involved in the evaluation of performance from two points of view: what variables must be measured and what implementation and external factors are desirable to control.

3.1 Multiobjective Search benchmarks

The empirical evaluation of algorithms is a main tool of research in Computer Science (Johnson, 2002). Theoretical studies can provide formal proofs of correctness and indicate the superiority of one algorithm over another, however, the point of research is, and has always been, applying those algorithms to real life problems. To do so, an empirical evaluation, either on simulated or realistic scenarios, is needed to confirm the results already provided in a theoretical manner. Wherever formal studies can not be provided, an empirical evaluation is the best way to approach the problem.

An empirical evaluation is based on two principles. First, reproducibility, i.e. guaranteeing that other researchers may obtain equivalent results when they use the same parameters to reproduce the experiments; and second, fairness, i.e. algorithms should share as much code as possible, define clearly implementation and execution parameters, as well as employ benchmarks or problems available online, see for example the 9th DIMACS Implementation challenge described below.

In this thesis we tackle the Multicriteria Search Problem and consider goals provided by a decision maker that define the subset of Pareto optimal solutions to be returned. On one hand, Multicriteria Search performance can be related to a number of distinct factors relative to the benchmark, such as graph size and shape, number of arcs, solution depth, costs range, number of criteria or correlation between objectives. On the other hand, Goal Programming is also concerned with the efficiency, which can be attributed to the form that the preferences are given, as well as the amount of solutions that satisfy them. Therefore, we define our empirical evaluation based on parameters relative to the graph, problem, and preferences given by the DM.

Many test beds have been proposed over the years for Multiobjective Search problems, for instance, randomly generated graphs, grids or road maps, see Section 3.1 (Machuca, 2012) for a recent survey of the literature in Multiobjective Search benchmarks. For instance, Klingman et al. (1974) published NETGEN, a random graph generator that first generates a connected skeleton and then adds arcs randomly. Skriver & Andersen (2000) used this tool in their research, however, they presented a new random graph generator, NETMAKER, arguing that graphs generated with NETGEN only had a small number of efficient solutions, even for large scale graphs.

In order to generate graphs, NETMAKER needs two input parameters: the branching factor and the interval length. The latter specifies which nodes are allowed to be reached from a particular node. Other random graphs have been proposed over the years (Hansen, 1980; Climaco & Martins, 1982; Nance et al., 1987; Brumbaugh-Smith & Shier, 1989; Mote et al., 1991; Gandibleux et al., 2006; Martins et al., 2007; Iori et al., 2010; Caramia et al., 2010; Galand et al., 2010).

Euclidean, rectangular, and square grids are another alternative to test multiobjective search algorithms performance. They have also been extensively used in the literature, e.g. (Mote et al., 1991; Guerriero & Musmanno, 2001; Guerriero et al., 2001; Martins et al., 2007; Caramia et al., 2010).

Several road maps scenarios have been provided for experimentation purposes, like the 9th DIMACS implementation challenge, see Section 3.2.2 for further details, maps obtained from OpenStreetMap [1] (Raith & Ehrgott, 2009; Raith, 2009), road networks from the Italian region of Lazio (Caramia et al., 2010), or small maps of Auckland (New Zealand) (Raith, 2009) for the bi-objective cyclist route problem.

To our knowledge, there are no specific benchmarks developed to test goal-based graph search algorithms.

3.2 Benchmarks used in this thesis

A combination of artificial and realistic scenarios are used in this thesis to assess the performance of the proposed algorithms. Artificially generated environments, like random grids, allow to control different parameters, such as number of nodes, branching factor, correlation between objectives or solution depth. Therefore, these are suitable to analyze tendencies over increasingly difficult problem instances.

On the other hand, realistic scenarios come with a fixed set of parameters, which can not be modified at will. However, these provide first-hand information on the

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

applicability of the proposed algorithms.

The evaluation strategy in this thesis combines both artificial and realistic scenarios. First, we test the performance over artificially designed environments, and then over realistic scenarios.

In addition to these scenarios, we aim to evaluate the performance of Multicriteria Search Problems according to different preferences, or sets of goals grouped in priority levels. Thus, we propose several classes of targets that emulate the possible preferences of a decision maker. These targets split the experiments into two classes of problems (see Section 6.1 for a detailed description of the experimental setup). Thus, we analyze the efficiency: (1) according to the satisfiability of the goals (whether they can be satisfied or not) (2) when goals can be satisfied; the size of the set of efficient solutions with respect to the size of the Pareto set. Let us now review the benchmarks.

3.2.1 Random grids

Artificially generated graphs and grids are the most extensively test sets used in the literature. In order to adequate the problem difficulty in graphs, several parameters have to be defined, such as branching factor, number of arcs, and mainly, the topology of the graph. Square grids have a fixed vicinity, are easy to create, and their size can be gradually increased.

Grids can also be considered a realistic scenario compatible with real-file applications, e.g. pathfinding in computer games (Bayili & Polat, 2011). Moreover, a recent study from Paixão & Santos (2008) revealed that problems defined over randomly generated graphs have a number of non-dominated solutions larger than graphs with similar configurations.

The random grids used in this thesis are designed to allow the controlled evaluation of performance with respect to solution depth. In particular, we have generated square bi-dimensional grids of 100×100, bidirectional arcs and a vicinity of four neighbors, i.e. grids with 10,000 nodes and 39,800 arcs. The start node is placed at the grid center $(50, 50)$ and a single destination node is placed in the diagonal from the center to the bottom right corner.

Different solution depths are considered, varying from 20 to 100, i.e. for solution depth d, the destination node is at coordinates $(50 + d/2, 50 + d/2)$. A set of five different problems is generated for each solution depth. For each arc q integer scalar costs $\vec{c}(i, j) = (c_1, c_2, ..., c_q)$ are randomly generated in the range [1,10] using an uniform distribution, i.e. leading to uncorrelated objectives. Table 3.1 shows the average number of Pareto-optimal solutions for each solution depth and number of objectives considered in our random grids experiments.

3.2.2 Road maps

Among the multiple realistic domains where Multicriteria Search arises (see Section 2.5 for a detailed classification of several application domains for MSP) we have selected route planing. Route planing is currently a hot research topic driven partly by the boom of GPS navigation devices and on-line route planners.

Single-objective search on road maps has been intensively studied over the last

Table 3.1. Average number of Pareto-optimal cost vectors relative to solution depth and number of objectives (q) in our grid problems.

q	Sol. depth	Avg. $\lvert C^* \rvert$
3	20	122
3	30	302
3	40	694
3	50	1,599
3	60	2,007
3	70	2,561
3	80	5,423
3	90	5,912
3	100	8,307
4	20	493
4	30	2,230
4	40	7,826
4	50	24,942
5	20	1,819
5	30	10,830
5	40	49,634

decades (Pearl, 1984; Zhan & Noon, 1998; Klunder & Post, 2006; Geisberger et al., 2008; Schultes, 2008; Delling et al., 2009). However, research on multiobjective route planning is a recent area of interest (Delling & Wagner, 2009; Raith & Ehrgott, 2009; Machuca & Mandow, 2011, 2012; Mali et al., 2012) and it is expected to become even more popular with the progressive introduction of electrical vehicles (Baum et al., 2014; Goodrich & Pszona, 2014) to optimize the trade-off between attributes like energy consumption or travel time.

The experiments presented in this thesis involve the use of realistic road maps from the 9th DIMACS Implementation Challenge: Shortest Path. Road maps are defined as graphs where arcs represent roads and nodes represent road junctions. Coordinates (longitude and latitude) are also provided for each node. The test bed comprises a set of twelve road maps of increasing size[2]. These data were provided by the 2000 U.S. Census Bureau's TIGER/Line R database (Topologically Integrated Geographic Encoding and Referencing system).

The original DIMACS maps provide two different criteria: physical distance and travel time. An additional criterion was introduced in Machuca & Mandow (2011) by estimating economic cost. This was obtained combining certain values for tolls and fuel consumption according to road category. The resulting values are not linearly correlated to those of the other cost values. The experiments reported in Chapter 7 consider the simultaneous minimization of these three attributes, physical distance (c_1), travel time (c_2), and economic cost (c_3).

Machuca & Mandow (2012) generated fifty problem instances for each of the four

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

smallest maps from the DIMACS Challenge: New York, San Francisco Bay Area, Colorado and Florida, to minimize simultaneously attributes (c_1) and (c_2). Given that our experimental evaluation considers the minimization of three attributes instead of two, our problems have much higher difficulty and therefore, we only selected the first twenty problems of the New York city map. Additionally, due to the fact that only fourteen out of the twenty problems of New York could be solved within the runtime limit, we employed a second road map from the Vermont State[3] and generated twenty random problems in a similar way. The Vermont State road map represents an interesting benchmark to observe performance trends, since all algorithms solve all problem instances within the runtime limit.

3.2.3 Significance of the test sets

Regarding random grid problems, the majority of past studies focus on the bi-criterion case (Raith, 2009; Machuca, 2012). Among those considering the multicriteria case, Caramia et al. (2010) use grids of 20×20 nodes and 167.5 efficient paths in average, Martins et al. (2007); Paixão & Santos (2013) also use grids with 20×20 nodes but a greater number of efficient paths, since they generated arcs cost in the range [1,1000]). Our experimental test on Multicriteria Search consider in general larger instances than those studies.

Regarding road map experiments, Machuca (2012) presented recent experiments over two sets of problems:

1. A first set of problems minimizes *distance* (c_1) and *time* (c_2) values provided by the DIMACS challenge maps.

2. A second set of problems minimizes *time* (c_2) and *economic cost* (c_3).

Table 3.2 illustrates the average number of Pareto-optimal solutions for each set of our problems. Pearson's correlation coefficient for costs c_1 and c_2 is 0.96 while that for c_2 and c_3 is 0.16, i.e. there is a strong linear correlation between time and distance, while there is no such a correlation between travel time and economic cost.

The sets of experiments with two objectives proposed by Machuca (2012) were composed by 50 problem instances, while our test set with three objectives comprises only twenty instances. We decided to reduce the number of problems in the test set due to the greater difficulty of the three objectives search compared to the bi-objective. Table 3.2 shows the average number of Pareto-optimal solution vectors for experiments with (c_1,c_2), (c_2,c_3), and (c_1, c_2, c_3) cost functions. For New York city road map, the average number of Pareto-optimal solution vectors of (c_1, c_2, c_3) experiments shows an asterisk. This indicates that 47,738.8 represents the average considering only the fourteen solvable problems of the set. Yet this number is approximately 240 and 22 times greater than the average Pareto-optimal solution vectors in experiments with (c_1,c_2) and (c_2,c_3) cost functions, respectively. In consequence, we expect our test sets to be in line with the hardest ones proposed to date for multiobjective search.

[3] Available on `http://www.dis.uniroma1.it/challenge9/data/tiger/`

Table 3.2. Average Pareto-optimal cost vectors for three sets of experiments. ($^+$) represents the average of the fourteen problems solved.

New York city road map	
Set of problems	Average $\|C^*\|$
(c_1, c_2)	198.6
(c_2, c_3)	2,086.6
(c_1, c_2, c_3)	$^+$47,738.8

Vermont road map	
Set of problems	Average $\|C^*\|$
(c_1, c_2)	81.6
(c_2, c_3)	247.1
(c_1, c_2, c_3)	3,334.6

3.2.4 Evaluation of preferences based on goals

In order to assess the impact of the goals satisfiability on the efficiency of the goal-based algorithms, we employ a lexicographic goal programming model that considers three goals grouped in two priority levels:

Level 1	$g_1 \leq t_1$,	$w_1 = 0.5$
	$g_2 \leq t_2$,	$w_2 = 0.5$
Level 2	$g_3 \leq t_3$,	$w_3 = 1$

Sets of target values for each problem are defined in terms of the ideal $\vec{\alpha} = (\alpha_1, \alpha_2, \alpha_3)$, and nadir points $\vec{\beta} = (\beta_1, \beta_2, \beta_3)$. These were previously calculated from the full Pareto sets obtained with NAMOA*. The nadir point is generally unknown in practice, but we take advantage of it in these experiments to obtain targets with different degrees of satisfaction for the purpose of experimentation. In a practical situation the ideal point is known thanks to the lower bound precalculations (Tung & Chew, 1992). These also provide the nadir point for two objectives, and at least an approximation for three or more objectives.

Two different classes of experiments were carried out. For the first class, five different target sets were calculated as follows:

$$t_i = \alpha_i + (\beta_i - \alpha_i) \times k_1, \qquad k_1 \in \{0, 0.25, 0.5, 0.75, 1\} \tag{3.1}$$

For example, for $k_1 = 1$ all Pareto-optimal solutions will satisfy all goals, and for $k_1 = 0$ no Pareto solution will likely satisfy them.

For the second class, targets of the first level were fixed for $k_1 = 0.75$ and $k_1 = 0.5$, which were found to provide satisfactory solutions. We then measured efficiency setting stricter targets for the third goal:

$$t_3 = \alpha_3 + (\beta_3 - \alpha_3) \times k_2 \qquad k_2 = k_1 \times k', \quad \text{where } k' \in \{0.25, 0.5, 0.75, 1\} \tag{3.2}$$

.

These values of t_3 allow us to evaluate the performance when some goals are satisfied and some not.

3.3 Evaluation of performance in Multicriteria Search

In the following chapters we introduce three new algorithms. The first one, called LEXGO*, is a goal-based algorithm; the second one, NAMOA$^*_{\text{dr}}$, represents a specialization of NAMOA*, and the last, LEXGO$^*_{\text{dr}}$, is devised as a specialization of LEXGO*. We use these algorithms to evaluate two alternatives to deal with MSP with goal preferences. In the first one, NAMOA* and NAMOA$^*_{\text{dr}}$ return the full Pareto set of non-dominated solutions to the problem and determine the subset of solutions that satisfy the goals from that set. In the second one, LEXGO* and LEXGO$^*_{\text{dr}}$ are used to search only for goal-optimal solutions.

Our purpose with these new algorithms can be summarized with two statements. In the first place, improve the performance of both alternatives, and in the second place, assess their performance.

This evaluation of the performance is usually characterized with respect to solution depth, explored labels, correlation between objectives (Brumbaugh-Smith & Shier, 1989; Mote et al., 1991; Machuca et al., 2010) or the presence/absence of lower bounds.

All proposed algorithms are based on NAMOA*. It has been proved that the more informed a consistent lower bound function is, the smaller the number of labels explored by NAMOA* with this function (Mandow & Pérez de la Cruz, 2010). The impact on efficiency has also been empirically confirmed by several studies (Machuca et al., 2012; Machuca, 2012). Hence, we will employ a lower bound as informed as possible in all analyzed algorithms regardless they find the full Pareto set or only the solutions to the problem that satisfy the goals.

Traditionally experimental evaluation of algorithms, e.g. see (Raith, 2009; Sauvanet & Néron, 2010), considers both space and time performance. Space requirements can be measured by the number of expanded or permanent labels, while the number of dominance checks has been pointed out as an important limiting factor in the time performance, e.g. see (Iori et al., 2010; Machuca & Mandow, 2011).

Once the variables to measure algorithm performance are identified, we turn our attention to the factors concerning the implementation of the algorithms. In Multicriteria Search two aspects must be specified whenever an algorithm is evaluated. Firstly, the label selection policy is used to choose non-dominated labels from OPEN. Lexicographic order is a frequent choice (e.g. see (Martins, 1984a)). A recent study showed the linear aggregation order can have better performance than the lexicographic one (Iori et al., 2010). NAMOA* will be tested with lexicographic and linear selection orders. In this thesis, we will show that the lexicographic order combined with the t-discarding technique can clearly outperform the linear aggregation order for the case with three objectives. The second aspect in the implementation of a multicriteria search algorithm (similarly in SPP) is the data structure to keep sorted the OPEN queue of alternatives (Paixão & Santos, 2013). Our algorithms are implemented using a binary heap.

Some other implementation details which can influence time performance are as follows:

- The particular machine where the experimental evaluation is run, e.g. architecture, number or processors, processor speed, physical memory available, etc.

- Process execution, e.g. number of simultaneous thread executions, amount of memory available to the process, Operating System, programming language used, compiler version, compiler optimization level, etc.
- In the OPEN queue two policies can be employed, depending on whether only the current best cost estimate of each node is kept in OPEN at each iteration Mandow & Pérez de la Cruz (2005), or all alternatives are stored in OPEN (Mali et al., 2012).
- Implementation of the G_{op} (and G_{cl}) sets of non-dominated labels, e.g. whether these sets, which consist of unordered items by definition, are ordered (or not) according to the label selection policy employed by the algorithm, and the data structure used to sort them.
- An optimized graph structure to store nodes and arcs in memory. This structure provides dynamic memory management of the graph and can optimize the expansion of consecutive nodes and edges (Mali et al., 2013).
- 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)

In particular, the decisions taken in the practical implementation of algorithms in this thesis are as follows:

- All the algorithms were run on an Intel Core i7 3612QM at 2.1 Ghz, 4GB of DDR3 RAM under Windows 7 (64-bit), and on a Sun Fire X4140 server with 2 six-core AMD Opteron 2435 at 2.6 GHz processors and 64 Gb of DDR2 RAM under Windows Server 2008 R2 (64-bit).
- The algorithms NAMOA*, NAMOA$^*_{\text{dr}}$, LEXGO* and LEXGO$^*_{\text{dr}}$ were implemented to share as much code as possible. The programming language used was ANSI Common Lisp. Each problem instance was solved using an individual process with a single thread.
- The lexicographic and linear selection orders were used to choose among non-dominated open alternatives in NAMOA* and LEXGO*. NAMOA$^*_{\text{dr}}$ and LEXGO$^*_{\text{dr}}$ use only the lexicographic order.
- The OPEN queue was implemented as a binary heap but only the current best label of each node is kept in OPEN at each iteration.
- The G_{op} and G_{cl} sets were ordered according to the label selection policy employed by the algorithm.

Part II

Contributions

The second part of this thesis describes our contributions to the field of multiobjective graph search algorithms with preferences based on goals. This comprises the formal and empirical analyses performed through this research work. This part is organized as follows:

- Chapter 4 introduces some of the contributions of this dissertation. Prior to presenting LEXGO*, our new label-setting multicriteria search algorithm with goal-based preferences, we introduce new definitions concerning lexicographic preferences and a new pruning rule devised for lexicographic preferences. These are presented along with LEXGO* in Section 4.1.

  A new dimensionality reduction technique that speeds up the time performance of exact multicriteria search algorithms is described in Section 4.2. The new algorithms, NAMOA$^*_{\text{dr}}$ and LEXGO$^*_{\text{dr}}$, based on the application of this technique, are presented in Section 4.3 and 4.4, respectively.

- Chapter 5 gives a formal analysis of the multiobjective algorithms considered in this thesis. Section 5.1 reminds the formal properties of NAMOA*. The formal properties of LEXGO* are introduced in Section 5.2 and it is formally proved that labels expanded by LEXGO* are always a subset of the labels expanded by NAMOA*. The t-discarding method is proved to be theoretically correct and that any admissible multiobjective search algorithm will remain to be admissible when this technique is applied. More precisely, the formal properties of NAMOA$^*_{\text{dr}}$ and LEXGO$^*_{\text{dr}}$, the versions of NAMOA* and LEXGO* applying t-discarding, are presented in Sections 5.3 and 5.4, respectively.

- Chapters 6 and 7 describe the empirical evaluation of the algorithms introduced in this thesis. These are conducted over random grids and realistic road map problems, respectively. First, the space and runtime performance of LEXGO* over NAMOA* is evaluated, and second, the time requirements of algorithms with t-discarding, i.e. NAMOA$^*_{\text{dr}}$ and LEXGO$^*_{\text{dr}}$ over their counterparts which use the standard dominance checks. Finally, a summary for each experiment analyzes the relative performance of all tested algorithms.

Chapter 4

New techniques for multiobjective and goal-based search

An algorithm must be seen to be believed.
Donald Knuth (1938-)

This chapter describes the algorithmic contributions of this thesis. These are broadly aimed at an efficient solution of goal-based search problems. In the first place, we introduce a new algorithm that specifically searches for lexicographic goal solutions in Section 4.1. The algorithm is called LEXGO*. We start noting that the optimality principle does not hold in general for lexicographic goal based preferences, and develop a specific pruning criterion for them. In the second place, we introduce in Section 4.2 a dimensionality reduction technique that speeds up the time performance of Label-setting multicriteria search algorithms. We review the application of this technique to NAMOA* and LEXGO*, in Sections 4.3 and 4.4, respectively. These algorithmic contributions are the base for the different alternatives to goal-based search analyzed later in this thesis.

4.1 Algorithm LEXGO*

This section introduces LEXGO*, an algorithm for lexicographic goal-based search problems. More precisely, we will address goal-based search according to the preferences already introduced in Section 2.3.2, and that will be used throughout this chapter. Multiobjective search algorithms benefit from the principle of optimality, i.e. an optimal path is made up of optimal subpaths. This property allows to drastically reduce the number of paths to be explored during search, pruning dominated alternatives at each node. Regrettably, the principle of optimality does not hold for lexicographic goal-based preferences. Let us see this by way of a simple example.

Example 1 *Let us consider a search problem in a multicriteria graph with start node s, a destination node t, and the following preferences over three different attributes:*

$$\begin{array}{lll} \textit{Level 1:} & g_1 \leq 20, & w_1 = 1.0 \\ \textit{Level 2:} & g_2 \leq 20, & w_2 = 0.5 \\ & g_3 \leq 20, & w_3 = 0.5 \end{array} \tag{4.1}$$

Let us further assume the graph has two different paths P_1 and P_2 reaching some node n, where $\forall n\ \vec{h}(n) = (0,0,0)$, and the following costs and associated deviations:

$$\begin{array}{l} \vec{g}(P_1) = \vec{f}(P_1) = (15,16,22) \ \Rightarrow\ \vec{d}(P_1) = (0,1) \\ \vec{g}(P_2) = \vec{f}(P_2) = (20,12,16) \ \Rightarrow\ \vec{d}(P_2) = (0,0) \end{array}$$

We observe that $\vec{f}(P_2) \prec_G \vec{f}(P_1)$, since $\vec{d}(P_2) \prec_L \vec{d}(P_1)$. However, we cannot discard P_1 in favor of P_2. Let us now consider there is only one additional path $P_3 = (n, \ldots, t)$ from n to the destination node with cost $\vec{g}(P_3) = (4,4,4)$. It is easy to show now that the concatenation P_1P_3 is the only goal-optimal solution:

$$\begin{array}{l} \vec{g}(P_1P_3) = (19,20,26) \ \Rightarrow\ \vec{d}(P_1P_3) = (0,3) \\ \vec{g}(P_2P_3) = (24,16,20) \ \Rightarrow\ \vec{d}(P_2P_3) = (4,0) \end{array}$$

This is the main difficulty in the development of a specific goal-based search algorithm. According to our definition of goal-based preferences, goal optima are among Pareto optima. Therefore, pruning dominated paths will lead to a correct algorithm, although the number of explored paths is likely to be very similar to that of full Pareto search. Our aim is to introduce a specific pruning condition that allows us to reduce the number of paths explored in goal-based search and, more specifically, that guarantees that the set of paths explored in such search is a subset of the set of paths explored by a full Pareto search. Let us start introducing the fundamentals of this pruning preference.

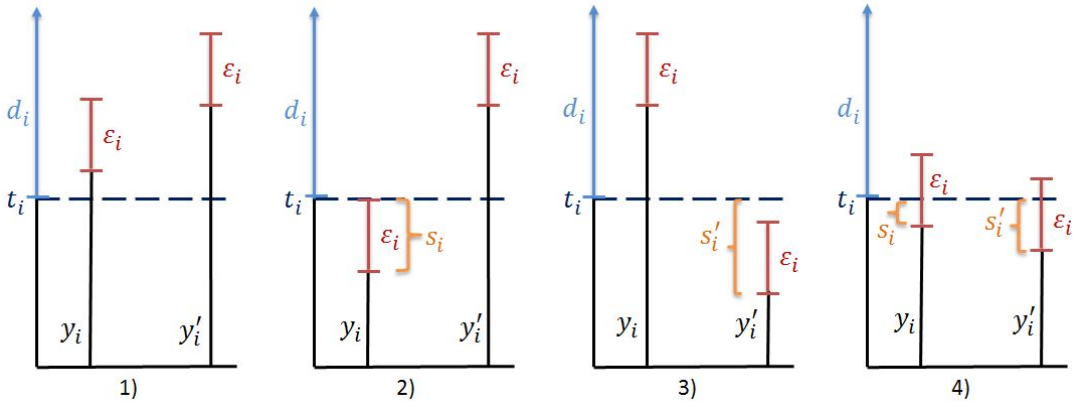


Figure 4.1. a) Graphic representation of slack variables for several scenarios where (1) $y_i, y_i' \geq t_i$, (2) $y_i \leq t_i < y_i'$, (3) $y_i' \leq t_i < y_i$ and (4) $y_i, y_i' < t_i$, adding ϵ_i to both y_i and y_i'.

Definition 4.1 *Let us consider a goal $y_k \leq t_k$ over some measurable attribute y_k. The* ***slack variable*** *s_k for this goal is defined as*

$$s_k = \max(0, t_k - y_k) \tag{4.2}$$

Let us assume two vectors $\vec{y}, \vec{y'} \in \mathbb{R}^q$ *and a level* j *such that* $d_j(\vec{y}) < d_j(\vec{y'})$*. Let us denote* $\Delta_j(\vec{y}, \vec{\epsilon}) = d_j(\vec{y} + \vec{\epsilon}) - d_j(\vec{y})$*. Obviously, if* $\vec{\epsilon} \succeq \vec{0}$*,* $\Delta_j(\vec{y}, \vec{\epsilon}) \geq 0$*. We define the* ***cross-slack*** $\delta_j(\vec{y}, \vec{y'}) = \max_{\vec{\epsilon} \in \mathbb{R}^{+q}}(\Delta_j(\vec{y}, \vec{\epsilon}) - \Delta_j(\vec{y'}, \vec{\epsilon}))$*, i.e. the greatest relative increment of the deviations of* $\vec{y}$ *and* $\vec{y'}$ *at level* j *when adding any* $\vec{\epsilon} \in \mathbb{R}^{+q}$*. Notice that* $\delta_j(\vec{y}, \vec{y'}) \geq 0$ *and generally* $\delta_j(\vec{y}, \vec{y'}) \neq \delta_j(\vec{y'}, \vec{y})$*.*

Each priority level j *comprises a set* I_j *of one or more attributes* i*. Figure 4.1 shows that for each* $i \in I_j$ *four different cases can arise: (1)* $y'_i, y_i \geq t_i$*; (2)* $y'_i \geq t_i$ *and* $y_i < t_i$*; (3)* $y'_i < t_i$ *and* $y_i \geq t_i$*; (4)* $y'_i, y_i < t_i$*. It is straightforward that the greatest relative increment in cases 1 and 2 is 0, since the slack variable* s'_i *equals* 0*, while in cases 3 and 4, the greatest relative increment is* $w_i \times (s'_i - s_i) = w_i \times (t_i - y'_i - (t_i - y_i)) = w_i \times (y_i - y'_i)$*. Therefore, an operative way of calculating the* cross-slack $\delta_j(\vec{y}, \vec{y'})$ *of* $\vec{y}, \vec{y'}$ *at level* j *is*

$$\delta_j(\vec{y}, \vec{y'}) = \sum_{k \in I_j} w_k \times \max(0, s'_k - s_k) \tag{4.3}$$

Definition 4.2 *We define the* ***pruning preference*** $\prec_P$ *by imposing on the lexicographical goal preference additional conditions concerning cross-slacks:*

$$\vec{y} \prec_P \vec{y'} \quad \Leftrightarrow \quad \exists j \ (d_j(\vec{y}) < d_j(\vec{y'}) \quad \wedge \quad \delta_j(\vec{y}, \vec{y'}) < d_j(\vec{y'}) - d_j(\vec{y}) \\ \wedge \quad \forall i < j \ \ (d_i(\vec{y}) = d_i(\vec{y'}) \quad \wedge \quad \delta_i(\vec{y}, \vec{y'}) = 0)) \tag{4.4}$$

i.e., $\vec{y} \prec_P \vec{y'}$ *when (i)* $\vec{y} \prec_G \vec{y'}$*; (ii) the cross-slacks of* $\vec{y}$ *and* $\vec{y'}$ *are zero for the first levels (where deviations are the same); and (iii) for the first level where deviations differ, the cross-slack of* $\vec{y}$ *and* $\vec{y'}$ *is strictly smaller than the difference between deviations.*

It can be easily checked that $\prec_P$ *is irreflexive and transitive. Therefore* $\prec_P$ *is a partial order relation. We read* $\vec{y} \prec_P \vec{y'}$ *as «*$\vec{y}$ *allows to prune* $\vec{y'}$*».*

Table 4.1 introduces LEXGO*, an exact label-setting multicriteria search algorithm for lexicographic goal preferences with lower bound estimates. The inputs are a multiobjective graph G, a start node s, a destination node t, a set of weighted goals grouped in pre-emptive priority levels, and a monotone distance estimate function. LEXGO* outputs the solution subgraph with the set of all goal-optimal solution paths between s and t. In a similar manner than most of the multicriteria search algorithms presented in Section 2.5, the following data structures are managed by the algorithm:

- **SG**: A search graph that records partial solution paths emanating from s and their costs. Each node n in SG stores the following information:
  - $G_{op}(n)$: Set of cost vectors (labels) $\vec{g}_n$ of paths reaching node n which have not been explored yet.
  - $G_{cl}(n)$: Set of labels reaching node n which have already been explored.
- **OPEN**: A priority queue of unexplored labels. For each node n in SG and each cost vector $\vec{g_n} \in G_{op}(n)$, there is a label $(n, \vec{g_n})$ in OPEN. In fact, labels are extended to include also evaluation vectors and their deviation from goals. Each extended label $(n, \vec{d}_n, \vec{f}_n, \vec{g}_n)$ denotes that node n is reached by a path with cost $\vec{g}_n$, deviation vector $\vec{d}_n$, and evaluation vector $\vec{f}_n$. We define $\vec{f}_n = \vec{g}_n + \vec{h}(n)$.

For the sake of simplicity, we will denote $\vec{d}(\vec{f}_n)$ as $\vec{d}_n$. Initially, $(s, \vec{d}_s, \vec{f}_s, \vec{g}_s)$ is the only label in OPEN. Labels in OPEN are sorted lexicographically according to deviation vectors. In case of ties they are ordered lexicographically according to evaluation vectors $\vec{f}$. This ensures that the first element in the queue has a goal-optimal evaluation among all f_n in OPEN.

- **COSTS**: The set of cost vectors of solution paths found to the destination node.
- **Best achievement** vector $\vec{d}_B$ among all solutions already found.

The structure of LEXGO* is similar to previous label-setting multicriteria algorithms with label expansion, but incorporating elements of lexicographic goal preferences to guarantee that only a subset of the labels explored by a full multicriteria search will need to be explored.

The algorithm has five main steps. The first one is devoted to data structure initialization. The second one is devoted to label selection from OPEN. At each iteration, the algorithm selects the first label $(n, \vec{d}_n, \vec{f}_n, \vec{g}_n)$ from OPEN, which has a goal-optimal evaluation vector f_n. The label is removed from OPEN, and moved from $G_{op}(n)$ to $G_{cl}(n)$. The third step recovers and returns the solution subgraph whenever some termination condition is satisfied. The fourth step records the solution whenever a destination node is selected. COSTS and $\vec{d}_B$ are updated accordingly. Finally, the selected label is expanded in step 5, i.e. all the extensions of the selected label are considered for inclusion in the search graph and the OPEN set.

The algorithm iterates over steps 2, 3 , 4 and 5 until OPEN is empty, or $\vec{d}_B \prec_L \vec{d}_n$, i.e. all potential goal-optimal solutions have been examined. In such case, the algorithm terminates returning a solution subgraph, made up of all goal-optimal solution paths. COSTS stores the set of distinct goal-optimal costs.

During path expansion two different conditions may prevent an extension from consideration: filtering and pruning. These are described in detail below.

4.1.1 Pruning conditions

As already explained in Example 1, the optimality principle does not hold for lexicographic goal preferences. Therefore, pruning and filtering using goal preferences would not yield an admissible label setting algorithm in this case. Nevertheless, LEXGO* includes two pruning conditions that improve search efficiency and, at the same time, guarantee that no goal-optimal solution will be pruned, (see Theorem 5.8 in Section 5.2.2):

- Pareto pruning. As in other Pareto search algorithms like NAMOA*, we prune any dominated path to any node. A new label $(m, \vec{d}_m, \vec{f}_m, \vec{g}_m)$ to node m is pruned whenever

$$\exists \vec{g} \in G_{op}(m) \cup G_{cl}(m) \mid \vec{g} \prec \vec{g}_m \tag{4.5}$$

- Deviation-based pruning. We propose an additional specific pruning condition based in the pruning preference defined by equation 4.4. We prune a new label $(m, \vec{d}_m, \vec{f}_m, \vec{g}_m)$ to node m whenever

Table 4.1. Pseudocode of LEXGO* algorithm

1. CREATE:
   —An empty search graph SG, and set s as its root.
   —Two sets $G_{cl}(s) = \emptyset$ and $G_{op}(s) = \{\vec{0}\}$.
   —A list of alternatives, OPEN $= \{(s, \vec{d}(\vec{h}(s)), \vec{h}(s), \vec{0}\}$.
   —An empty set, COSTS.
   —$\vec{d}_B = \vec{\infty}$, optimum achievement vector for solutions found.
2. PATH SELECTION. If OPEN is not empty, then,
   —Select a label $(n, \vec{d}_n, \vec{f}_n, \vec{g}_n)$ from OPEN such that $\nexists (n', \vec{d}_{n'}, \vec{f}_{n'}, \vec{g}_{n'}) \in OPEN$ such that $\vec{f}_{n'} \prec_G \vec{f}_n$.
   —Delete the selected label from OPEN, and move $\vec{g}_n$ from $G_{op}(n)$ to $G_{cl}(n)$.
   —If $\exists \vec{c^*} \in COSTS$ such that $\vec{c^*} \prec \vec{f}_n$, then repeat step 2 (lazy filtering)
3. CHECK TERMINATION. If OPEN is empty, or $\vec{d}_B \prec_L \vec{d}_n$, then backtrack in SG from t and return the set of solution paths with costs in COSTS.
4. SOLUTION RECORDING. If n is a destination node, then
   —Include $\vec{g}_n$ in COSTS.
   —$\vec{d}_B \longleftarrow \vec{d}_n$
   —Go back to step 2.
5. PATH EXPANSION: If n is not a destination node, then for all successor nodes m of n do:
   (a) Calculate the cost of the new path found to m, its evaluation vector and deviation, $\vec{g}_m = \vec{g}_n + \vec{c}(n, m)$, $\vec{f}_m = \vec{g}_m + \vec{h}(m)$, $\vec{d}_m = \vec{d}(\vec{f}_m)$.
   (b) If no Pareto or deviation filtering (equations 4.7 and 4.8), then:
   - If $m \notin SG$:
     – Add $(m, \vec{d}_m, \vec{f}_m, \vec{g}_m)$ to OPEN
     – Set $G_{op}(m) = \{(\vec{g}_m)\}$.
     – Label with $\vec{g}_m$ a pointer from m to n.
   - else if $\vec{g}_m$ equals some cost vector in $G_{op}(m) \cup G_{cl}(m)$ then
     – Label with $\vec{g}_m$ a pointer from m to n.
   - else if no Pareto or deviation pruning (equations 4.5 and 4.6), then:
     i. Eliminate vectors $\vec{g'}_m \in G_{op}(m)$ such that $\vec{g}_m \prec \vec{g'}_m \vee \vec{f}_m \prec_P \vec{g'}_m + \vec{h}(m)$, and their corresponding labels $(m, \vec{d'}_m, \vec{f'}_m, \vec{g'}_m)$ from OPEN.
     ii. Add $(m, \vec{d}_m, \vec{f}_m, \vec{g}_m)$ to OPEN, $\vec{g}_m$ to $G_{op}(m)$ and label with $\vec{g}_m$ a pointer from m to n.
   (c) Go back to step 2.

$$\exists \vec{g} \in G_{op}(m) \cup G_{cl}(m) \mid \vec{g} + \vec{h}(m) \prec_P \vec{f}_m \tag{4.6}$$

Example 2 *Let us assume the same preference as in Example 1 and two paths P and P' reaching the same node n from s with the following evaluation vectors:*

$$\vec{f} = \vec{f}(P) = (22, 22, 12) \Rightarrow \ \vec{d}(P) = (2, 1)$$
$$\vec{f'} = \vec{f}(P') = (22, 18, 26) \Rightarrow \ \vec{d}(P') = (2, 3)$$

We observe that $\vec{d}(P) \prec_L \vec{d}(P')$. We can also easily check that the extra conditions for pruning, $\delta_1(\vec{f}, \vec{f'}) = 0$ and $\delta_2(\vec{f}, \vec{f'}) = 1 < 3 - 1 = 2$, also hold,

$$\begin{aligned}
\delta_1(\vec{f}, \vec{f'}) &= 1 \times \max(0, s'_1 - s_1) = \max(0, 0 - 0) = 0 \\
\delta_2(\vec{f}, \vec{f'}) &= 0.5 \times \max(0, s'_2 - s_2) + 0.5 \times \max(0, s'_3 - s_3) \\
&= 0.5 \times \max(0, 2 - 0) + 0.5 \times \max(0, 0 - 8) \\
&= 1
\end{aligned}$$

Therefore, path P' will never lead to a better solution than P and can be safely pruned.

4.1.2 Filtering conditions

Filtering is the process of discarding labels that will never lead to a solution better than one already found. Two different conditions allow a label $(n, \vec{d}_n, \vec{f}_n, \vec{g}_n)$ to be filtered:

- Pareto filtering. This is the standard dominance filtering in Pareto search algorithms:
$$\exists \vec{c^*} \in COSTS \mid \vec{c^*} \prec \vec{f}_n \tag{4.7}$$
- Deviation based filtering. We introduce a specific filtering condition for goal-based preferences when a known solution has better goal satisfaction:
$$\vec{d}_B \prec_L \vec{d}_n \tag{4.8}$$

When a new solution is found, or the best achievement vector is updated, no new label satisfying the above conditions will be allowed to enter OPEN, however, those labels already in OPEN will not be straightforwardly discarded. This is due to LEXGO* applies *lazy filtering*.

4.1.3 Example

Let us now illustrate the algorithm with a simple example. Let us assume that the decision maker's preference involves two levels of goals:

Level 1	$cost_1(P) \leq 10,$	$w_1 = 0.5$
	$cost_2(P) \leq 10,$	$w_2 = 0.5$

Level 2 $\quad cost_3(P) \leq 10, \quad w_3 = 1$

Let us consider the sample graph in Figure 4.2, where s is the start node, and t the destination node. A lower bound function $\vec{h}(n)$ has been calculated using the method proposed by Tung and Chew (Tung & Chew, 1992) and is presented in Table 4.2. A trace of the OPEN list is shown in Table 4.3. At each iteration the selected label is indicated with an arrow and pruned labels are crossed out.

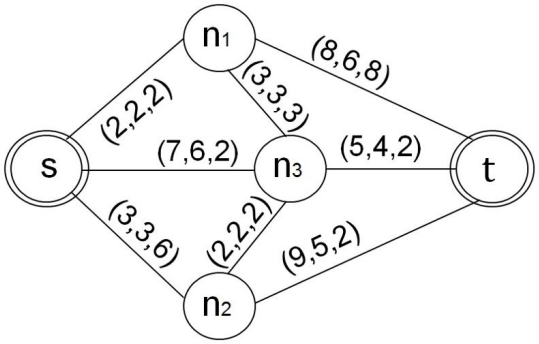


Figure 4.2. Sample graph with satisfiable goals

At iteration 1, SG has only node s as its root and its corresponding label is selected from OPEN. Labels for the three descendants n_1, n_2 and n_3 of s are added to OPEN. At iteration 2, two labels in OPEN have the same deviation vector, so the best lexicographic $\vec{f}$ is used to break the tie. Hence, the label to n_1 is selected. Its two successors n_3 and t are added to OPEN. Addition of n_3 to $G_{op}(n_3)$ prunes the alternative already stored for n_3, since $(10, 9, 7) \prec_P (12, 10, 4)$. Notice that both evaluation vectors are non-dominated, however, the pruning condition presented in equation 4.6 is applied, since $\delta_1((10, 9, 7), (12, 10, 4)) < d_1(12, 10, 4) - d_1(10, 9, 7) \;\Rightarrow\; 0 < 1 - 0$. At iteration 3, the label to n_2 is selected and expanded, generating new paths to the successors n_3 and t. The extension to n_3 is pruned, since the cost vector $(5, 5, 8)$ from path (s, n_2, n_3) is dominated by the cost $(5, 5, 5)$ from path (s, n_1, n_3). The second successor, t, is also pruned due to the existence of another label in $G_{op}(t)$ such that $(10, 8, 10) \prec_P (12, 8, 8)$. At iteration 4, the first path to a destination node is selected, the corresponding cost vector is added to COSTS $= \{(10, 8, 10)\}$ and $\vec{d}_B$ is updated to $(0, 0)$. This means that there is at least one path which satisfies all the goals provided.

At iteration 5, n_3 is selected, and a new path to t is generated and added to OPEN. At iteration 6, the only label in OPEN is selected. The cost vector $(10, 9, 7)$ represents another solution since t is the destination node, its cost is not dominated by any vector in COSTS and it can also satisfy all goals. Finally, in the next iteration OPEN is empty and the algorithm would search backward from t returning the solution subgraph with the two paths with costs (10,8,10) and (10,9,7).

4.2 A dimensionality reduction technique for MSP

This section describes a dimensionality reduction technique that speeds up the time performance of exact multicriteria search algorithms. Dimensionality reduction was

Table 4.2. Lower bounds table with distance estimates of an example of LEXGO* with satisfiable goals

n	$\vec{h}(n)$
s	(10,8,4)
n_1	(8,6,5)
n_2	(7,5,2)
n_3	(5,4,2)
t	(0,0,0)

Table 4.3. Execution trace of an example of LEXGO* with feasible goals (graph in Figure 4.2).

It	OPEN $(n, \vec{d}, \vec{f}, \vec{g})$
1	$(s,(0,0),(10,8,4),(0,0,0))$ ⟵
	$(n_1,(0,0),(10,8,7),(2,2,2))$ ⟵
2	$(n_2,(0,0),(10,8,8),(3,3,6))$
	$(n_3,(1,0),(12,10,4),(7,6,2))$
	$(n_2,(0,0),(10,8,8),(3,3,6))$ ⟵
3	$(t,(0,0),(10,8,10),(10,8,10))$
	$(n_3,(0,0),(10,9,7),(5,5,5))$
	~~$(n_3,(1,0),(12,10,4),(7,6,2))$~~
	$(t,(0,0),(10,8,10),(10,8,10))$ ⟵
4	$(n_3,(0,0),(10,9,7),(5,5,5))$
	~~$(n_3,(0,0),(10,9,10),(5,5,8))$~~
	~~$(t,(1,0),(12,8,8),(12,8,8))$~~
5	$(n_3,(0,0),(10,9,7),(5,5,5))$ ⟵
6	$(t,(0,0),(10,9,7),(10,9,7))$ ⟵
7	EMPTY SET

first proposed as a space saving technique in the development of *vector frontier search* (Mandow & Pérez de la Cruz, 2008a, 2009), a blind multiobjective search algorithm that achieved impressive reductions in space requirements at the expense of increasing time requirements. This technique is used in this thesis showing that it can also be applied under reasonable assumptions to exact multicriteria search (and to NAMOA* and LEXGO* in particular) to achieve important improvements in time performance.

The key idea in dimensionality reduction is that, under a lexicographic order of expansion, dominance checks performed during filtering and certain pruning operations can be greatly simplified. We will start introducing the following useful terminology. We will say that a vector v is dominated by a set X when there exists $v' \in X$ such that $v' \prec v$. Dominance checks performed on labels by NAMOA*(see Table 2.2) can be stated as follows:

- Filtering. Discard $(n, \vec{g}, \vec{f})$ if f is dominated by COSTS.
- Op-pruning. Discard $(n, \vec{g}, \vec{f})$ if $\vec{g}$ is dominated by $G_{op}(n)$.
- Cl-pruning. Discard $(n, \vec{g}, \vec{f})$ if $\vec{g}$ is dominated by $G_{cl}(n)$.

First we reproduce two definitions from Mandow & Pérez de la Cruz (2009):

Definition 4.3 *Given a vector $\vec{v} = (v_1, v_2, \dots v_n)$, its **truncated vector** $t(\vec{v})$ is that vector v without its first component, i.e. $t(\vec{v}) = (v_2, \dots v_n)$.*

Definition 4.4 *Given a set of vectors X, its associated set of truncated vectors is $T(X) = \mathcal{N}(X)(\{t(\vec{x}) |\ \vec{x} \in X\})$.*

We call our reduction technique "t-discarding" (or *truncated discarding*), since it is based on discarding cost vectors by their truncated cost vectors.

Definition 4.5 *Let X be a set of vectors. A vector $\vec{v}$ is t-discarded by X when for all $\vec{v'} \in X$ $v'_1 \leq v_1$ and there is $\vec{v''} \in X$ such that $t(\vec{v''}) \in T(X)$ and one of the following conditions holds:*
a) $v''_1 < v_1$ and $t(\vec{v''}) \preceq t(\vec{v})$; or
b) $v''_1 = v_1$ and $t(\vec{v''}) \prec t(\vec{v})$.

Example 3 Figure 4.3 displays three cost vectors $\vec{x} = (6, 2, 4)$, $\vec{y} = (4, 4, 5)$, and $\vec{z} = (2, 3, 6)$, and shows also their truncated vectors $t(\vec{x})$, $t(\vec{y})$, and $t(\vec{z})$. Notice that none of $\vec{x}, \vec{y}, \vec{z}$ dominates any of the others; so $\mathcal{N}(X) = \{\vec{x}, \vec{y}, \vec{z}\}$. On the other hand the truncated vectors are $t(\vec{x}) = (2, 4)$, $t(\vec{y}) = (4, 5)$ and $t(\vec{z}) = (3, 6)$. Since $t(\vec{y})$ and $t(\vec{z})$ are dominated by $t(\vec{x})$ we have $T(X) = \{(2, 4)\}$.

Let us consider now a vector $\vec{w} = (7, 2, 4)$. The standard dominance test would imply three vector comparisons in the worst case, namely those of $\vec{w}$ against $\vec{x}, \vec{y}$ and $\vec{z}$. However, t-discarding implies just one vector comparison, that of $t(\vec{w})$ against $t(\vec{x})$. Since $t(\vec{x}) \preceq t(\vec{w})$ and $x_1 < w_1$, w will be t-discarded by X.

Let us consider now vector $w' = (6, 2, 4)$. Let us check if $\vec{w'}$ is t-discarded by X. We have $t(\vec{x}) \preceq t(\vec{w'})$, but $x_1 \not< w'_1$, so condition (a) does not hold. Again, $t(\vec{x}) \not\prec t(\vec{w'})$, so condition (b) does not hold either, and in consequence $\vec{w'}$ is not t-discarded by X.

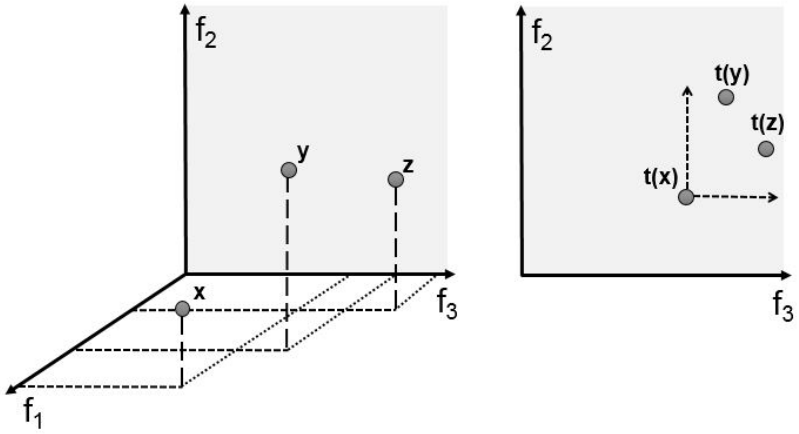


Figure 4.3. A set of vectors $X = \{\vec{x}, \vec{y}, \vec{z}\}$ and its truncated vectors $t(\vec{x}), t(\vec{y}), t(\vec{z})$.

We propose a modification of filtering and/or cl-pruning checks and prove that the new version of the checks is equivalent to the original one, given some assumptions, see the formal proof in Section 5.3.1. An example concerning the impossibility of applying t-discarding to the op-pruning is shown in Section 4.3, while the relative ratio of op-pruning, cl-pruning and filtering operations over the total number of discarded labels in our experiments over random grids and road maps can be seen in Figures 6.9 and 7.4, respectively.

We present below the application of the t-discarding technique to NAMOA*. Moreover, let us emphasize that not only a priori multicriteria algorithms may benefit from this technique, we also review the application to LEXGO* in Section 4.4.

4.3 Algorithm NAMOA$^*_{\mathbf{dr}}$

Table 4.5 shows a slightly modified pseudocode of NAMOA* combined with the dimensionality reduction technique, i.e. NAMOA$^*_{\text{dr}}$. The newly added or modified pseudocode lines have been emphasized with a right arrow. Although NAMOA* uses a general lower bound function $H(n)$ that returns a set of vector estimates, NAMOA$^*_{\text{dr}}$ employs a lower bound function $h(n)$, limited to a single vector estimate per node and that satisfies the monotone property (see Definition 2.31). NAMOA$^*_{\text{dr}}$ uses the same data structures employed by NAMOA*, with the only addition of the sets T(COSTS) and $T(G_{cl}(n))$.

The new cl-pruning and filtering operations in NAMOA$^*_{\text{dr}}$ employ the truncated set of vectors, instead of the original ones, to discard new labels. In addition, the truncated sets must be updated after each insertion of a new truncated vector. The equations corresponding to these operations are shown in Table 4.4. The update operation will be relatively costly depending on the size of the sets of truncated vectors. On the other hand, the size of the sets of truncated vectors over the size of the original ones defines the savings that can be achieved. This along with the frequency that op-pruning occurs shall define the efficiency of NAMOA$^*_{\text{dr}}$. We will analyze empirically the performance of NAMOA$^*_{\text{dr}}$ in Sections 6.3 and 7.2.

Table 4.4. New operations over truncated sets of vectors.

1. Pareto pruning (dr). A label $(m, \vec{f}_m, \vec{g}_m)$ is pruned whenever

$$\exists \vec{v} \in T(G_{cl}(m)) \mid v \prec t(\vec{g}_m) \qquad \vee \qquad \exists \vec{g} \in G_{op}(m) \mid \vec{g} \prec \vec{g}_m \tag{4.9}$$

2. Pareto filtering (dr). A label $(m, \vec{f}_m, \vec{g}_m)$ is filtered whenever

$$\exists \vec{v} \in T(COSTS) \quad | \quad \vec{v} \prec t(\vec{f}_m) \tag{4.10}$$

3. Update set of truncated solution cost vectors. After inserting vector $\vec{v}$ in T(COSTS) do

$$\text{Remove} \quad \vec{v'} \in T(COSTS) \quad | \quad \vec{v} \prec \vec{v'} \tag{4.11}$$

4. Update set of truncated permanent vectors. After inserting vector $\vec{v}$ in $T(G_{cl}(m))$ do

$$\text{Remove} \quad \vec{v'} \in T(G_{cl}(m)) \quad | \quad \vec{v} \prec \vec{v'} \tag{4.12}$$

Two important changes in the adaptation of NAMOA* should be noted. First, the lexicographic order is the mandatory label selection policy in NAMOA$^*_{\text{dr}}$ to select between all non-dominated alternatives in OPEN, whilst NAMOA* can work out with any policy that assures the selected label is non-dominated. Moreover, NAMOA* has been previously reported to perform better under a linear aggregation policy than under a lexicographic one (Machuca & Mandow, 2011). Second, in step 5(b)iii, the original pseudocode of NAMOA* does not establish which pruning operation should be performed in the first place, over open or permanent labels, however, we plainly define the cl-pruning as the first operation to be made, to take advantage of the t-discarding technique before applying op-pruning. Concerning to op-pruning, let us now present an example to show why the new technique cannot be applied to the op-pruning operation.

Example 4 *Figure 4.4 displays a sample graph with three objectives. Let us assume for the sake of clarity that* $\vec{h}(n) = \vec{0}$. *At iteration 1, s is expanded and its two extended paths* $s \to n_1$ *and* $s \to n_2$ *stored in OPEN. At iteration 2, label* $(n_1, (1,1,1), (1,1,1))$ *is expanded and label* $(n_3, (4,4,4), (4,4,4))$ *recorded in* $G_{op}(n_3)$. *In case t-discarding were applied to op-pruning,* $T(G_{op}(n_3)) = \{(4,4)\}$. *At the next iteration label* $(n_2, (2,2,2), (2,2,2))$ *is expanded according to the lexicographic selection order.*

The new generated path, $s \to n_2 \to n_3$, *with label* $(n_3, (3,5,5), (3,5,5))$ *is non-dominated with respect to* $(n_3, (4,4,4), (4,4,4))$, *however, if we apply op-pruning the truncated vector* $(5,5)$ *corresponding to label* $(n_3, (3,5,5), (3,5,5))$ *would be t-discarded by truncated vector* $(4,4)$ *corresponding to label* $(n_3, (4,4,4), (4,4,4))$. *Logically, the first component of the new generated label does not necessarily have its first component equal or greater than all the open labels reaching the node (on the contrary, a lexicographic order and a consistent lower bound function can guarantee that for the*

Table 4.5. Pseudocode of NAMOA$^*_{dr}$ algorithm.

1. CREATE:
   — An empty search graph SG, and place s as its root.
   — Two sets $G_{cl}(s) = \emptyset$ and $G_{op}(s) = \{(\vec{0})\}$.
   — A list of alternatives, OPEN $= \{(s, \vec{0}, \vec{h}(s)\}$.
   — An empty set, COSTS.
   → Two empty sets of truncated vectors $T(G_{cl}(s))$ and T(COSTS).
2. PATH SELECTION. If OPEN is not empty, then,
   → Select a label $(n, \vec{g}_n, \vec{f}_n)$ from OPEN such that $\nexists (n', \vec{g}_{n'}, \vec{f}_{n'}) \in$ OPEN such that $\vec{f}_{n'} \prec_L \vec{f}_n$.
   — Delete the selected label from OPEN, and move $\vec{g}_n$ from $G_{op}(n)$ to $G_{cl}(n)$.
   → Add $t(\vec{g}_n)$ to $T(G_{cl}(n))$ and update $T(G_{cl}(n))$ (equation 4.12).
   → If Pareto filtering (dr) $\vec{f}_n$ (equation 4.10), then repeat step 2
3. CHECK TERMINATION. If OPEN is empty, then backtrack in SG from γ and return the set of solution paths with costs in COSTS.
4. SOLUTION RECORDING. If n is a destination node, then
   Include $\vec{g}_n$ in COSTS and
   → Add $t(\vec{g}_n)$ to T(COSTS) and update T(COSTS) (equation 4.11).
   — Go back to step 2.
5. PATH EXPANSION: If n is not a destination node, then for all successor nodes m of n do:
   (a) Calculate the cost of the new path found to m and its lower bound, $\vec{g}_m = \vec{g}_n + \vec{c}(n, m)$ and $\vec{f}_m = \vec{g}(m) + \vec{h}(m)$.
   (b) → Unless Pareto filtering (dr) $\vec{f}_m$ (equation 4.10):
      i. If $m \notin$ SG:
         - Set $G_{op}(m) = \{(\vec{g}_m)\}$ and add $(m, \vec{g}_m, \vec{f}_m)$ to OPEN.
         - Label with $\vec{g}_m$ a pointer from n to m.
      ii. else if $\vec{g}_m$ equals some cost vector in $G_{op}(m) \cup G_{cl}(m)$ then
         - Label with $\vec{g}_m$ a pointer from n to m.
      iii. → else unless Pareto pruning (dr) (eq 4.11):
         - Eliminate vectors $\vec{g}_{m'} \in G_{op}(m)$ such that $g_m \prec \vec{g}_{m'}$ and its corresponding label $(m, \vec{g}_{m'}, \vec{f}_{m'})$ from OPEN.
         - Add $(m, \vec{g}_m, \vec{f}_m)$ to OPEN, $\vec{g}_m$ to $G_{op}(m)$ and label with $\vec{g}_m$ a pointer from n to m.
   (c) Go back to step 2.

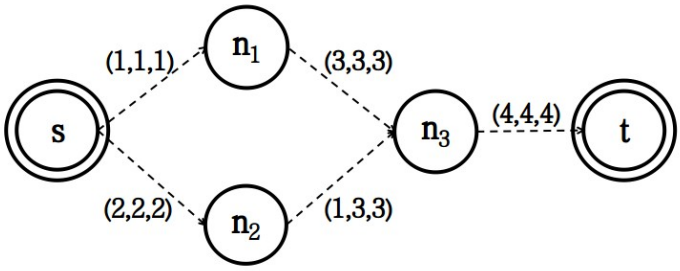


Figure 4.4. Sample graph one with 3 objectives.

permanent labels), therefore, t-discarding cannot be applied to op-pruning.

4.4 Algorithm LEXGO$^*_{\text{dr}}$

Table 4.6 displays the pseudocode of LEXGO* adapted to incorporate the t-discarding procedure, i.e. the version that employs the dimensionality reduction technique, called LEXGO$^*_{\text{dr}}$. A right arrow emphasizes the added or modified lines. The algorithm also uses the same monotone lower bound function $\vec{h}(n)$ and data structures employed by LEXGO*, with the addition of T(COSTS) and $T(G_{cl}(n))$. An example of use of LEXGO* was presented in Section 4.1.3 (the behavior of LEXGO$^*_{\text{dr}}$ does not differ from the behavior of LEXGO*).

LEXGO*, like NAMOA*, can use any label selection policy as long as the selected labels are always guaranteed to be non-dominated. Nevertheless, LEXGO$^*_{\text{dr}}$ must employ the lexicographic order. This restricts the scenarios where cl-pruning and filtering operations can be applied to those where all permanent paths reaching a node have zero deviation for all priority levels, i.e. we define the scenario where t-discarding technique can be applied to LEXGO$^*_{\text{dr}}$ as follows:

$$\forall n \quad \forall l_n = (n, \vec{d}_n, \vec{f}_n, \vec{g}_n) \in (G_{cl}(n)) \quad \vec{d}_n = \vec{0} \quad \wedge \quad \forall \vec{c^*} \in COSTS \quad \vec{d}(\vec{c^*}) = \vec{0} \quad (4.13)$$

In this case, the set of labels expanded by LEXGO* and LEXGO$^*_{\text{dr}}$ will be equivalent and they will be expanded exactly in the same order. Therefore, we will apply t-discarding to filtering or cl-pruning whenever all the labels in the set satisfy the condition to have a null deviation from goals. Since labels are expanded in lexicographic order, as soon as the first label with a positive deviation is selected, LEXGO$^*_{\text{dr}}$ will change its filtering and cl-pruning to the regular one. We present an example of this phenomenon below. The analyses concerning the time performance of LEXGO$^*_{\text{dr}}$ over the other presented algorithms will be further studied empirically and formally in Section 5.4.

Example 5 *Figure 4.5 displays a sample partial graph with three objectives. Let us assume that* $\vec{h}(n) = \vec{0}$ *and the lexicographic goals are the ones presented in Section 4.1.3. At iteration 1, s is expanded and its two extended paths* $s \rightarrow n_1$ *and* $s \rightarrow n_2$ *stored in OPEN and labels* (n_1, *(0,0), (3,3,3), (3,3,3))* *and* (n_2, *(0,0), (10,4,4), (10,4,4)), stored in* $G_{op}(n_1)$ *and* $G_{op}(n_2)$, *respectively.*

Table 4.6. Pseudocode of LEXGO$^*_{\text{dr}}$ algorithm.

1. CREATE:

   —An empty search graph SG, and set s as its root.

   —Two empty sets $G_{cl}(s)$ and COSTS, and $G_{op}(s) = \{\vec{0}\}$.

   —A list of alternatives, OPEN $= \{(s, \vec{d}(\vec{h}(s)), \vec{h}(s), \vec{0}\}$.

   → Two empty sets of truncated vectors $T(G_{cl}(s))$ and T(COSTS).

   —$\vec{d}_B = \vec{\infty}$, optimum achievement vector for solutions found.

   → A variable $sat = TRUE$ to record whether goals are satisfied or not.

2. PATH SELECTION. If OPEN is not empty, then,

   Select a label $(n, \vec{d}_n, \vec{f}_n, \vec{g}_n)$ from OPEN s.t.
   $\nexists(n', \vec{d}_{n'}, \vec{f}_{n'}, \vec{g}_{n'}) \in OPEN \mid \vec{f}_{n'} \prec_G \vec{f}_n$.

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

   → If $\vec{d} \neq \vec{0}$ then $sat = FALSE$.

   If $sat == TRUE$ then $t(\vec{g}_n)$ to $T(G_{cl}(n))$ and update $T(G_{cl}(n))$ (equation 4.12).

   - If Pareto filtering (dr) f_n (equation 4.10), then repeat step 2

   → Else if Pareto filtering (equation 4.7) then repeat step 2

3. CHECK TERMINATION. If OPEN is empty, or $\vec{d}_B \prec_L \vec{d}_n$, then backtrack in SG from t and return the set of solution paths with costs in COSTS.

4. SOLUTION RECORDING. If n is a destination node, then

   → Include $\vec{g}_n$ in COSTS and $\vec{d}_B \longleftarrow \vec{d}_n$

   → If sat is TRUE add $t(\vec{g}_n)$ to T(COSTS) and update it (equation 4.11).

   —Go back to step 2.

5. PATH EXPANSION: If n is not a destination node, then for all successor nodes m of n do:

   (a) Calculate the cost of the new path found to m, its evaluation vector and deviation, $\vec{g}_m = \vec{g}_n + \vec{c}(n, m)$, $\vec{f}_m = \vec{g}_m + \vec{h}(m)$, $\vec{d}_m = \vec{d}(\vec{f}_m)$.

   (b) If no Pareto or deviation filtering (equations 4.7 (or 4.10 when $sat == TRUE$) and 4.8), then:

   - If $m \notin SG$:
     - Add $(m, \vec{d}_m, \vec{f}_m, \vec{g}_m)$ to OPEN
     - Set $G_{op}(m) = \{(\vec{g}_m)\}$.
     - Label with $\vec{g}_m$ a pointer from m to n.
   - else if $\vec{g}_m$ equals some cost vector in $G_{op}(m) \cup G_{cl}(m)$ then
     - Label with $\vec{g}_m$ a pointer from m to n.
   - else if no Pareto or deviation pruning (equations 4.5 (or 4.10 when $sat == TRUE$) and 4.6), then:
     i. Eliminate vectors $\vec{g'}_m \in G_{op}(m) \mid \vec{g}_m \prec \vec{g'}_m \vee \vec{f}_m \prec_P \vec{g'}_m + \vec{h}(m)$, and their corresponding labels $(m, \vec{d'}_m, \vec{f'}_m, \vec{g'}_m)$ from OPEN.
     ii. Add $(m, \vec{d}_m, \vec{f}_m, \vec{g}_m)$ to OPEN, $\vec{g}_m$ to $G_{op}(m)$ and label with $\vec{g}_m$ a pointer from m to n.

   (c) Go back to step 2.

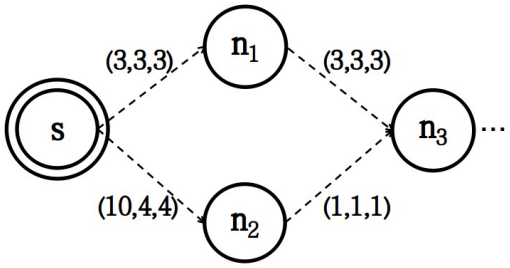


Figure 4.5. Sample graph two with 3 objectives.

At iteration 2, label (n_1, (0,0), (3,3,3), (3,3,3)) is expanded and label (n_2, (0,0), (6,6,6), (6,6,6)) recorded in $G_{op}(n_3)$. Thus, At iteration 3, label (n_3, (0,0), (6,6,6), (6,6,6)) is expanded, since the concatenation of its deviation and evaluation vectors ($\vec{d}\cdot\vec{f}$) is lexicographically better than the other alternative in OPEN, (n_2, (0,0), (10,4,4), (10,4,4)), and added to $G_{cl}(n_3)$.

At iteration 4, label (n_2, (0,0), (10,4,4), (10,4,4)) is expanded and its extension (n_2, (0.5,0), (11,5,5), (11,5,5)) is checked for a cl-pruning operation against $G_{cl}(n_3) = \{((0,0),(6,6,6),(6,6,6))\}$. If we do not pay attention to the fact that not all deviation vectors of labels in n_3 are zero, the condition to t-discard this path applies, i.e. $(5,5) \prec (6,6)$, but it is obvious that $(11,5,5) \not\prec (6,6,6)$.

Chapter 5

Formal Analysis of Multicriteria Algorithms

Make everything as simple as possible, but not simpler.

Albert Einstein (1879-1955)

In this chapter we aim to give a formal analysis of the multiobjective algorithms introduced in Chapter 4. We look for the best algorithmic alternative when goal-based preferences are given by an user to characterize the optimal solution subset. Two possibilities arise here. The first, obvious one, is to calculate the whole Pareto set using a multiobjective search algorithm like NAMOA* and extract a posteriori the subset of Pareto solutions which satisfy the goals. A second alternative is to concentrate the search effort only in the goal-optimal solutions, i.e. discard from the first stages of the search those paths which will not lead to satisfy the goals or minimize the deviation from them. This thesis has introduced one such algorithm, called LEXGO*.

Both algorithms, NAMOA* and LEXGO* may share the same label selection procedure and filtering/pruning processes, though LEXGO* introduces extra processes of pruning and filtering considering the deviation from goals, which will be formally studied further in this section.

Another algorithmic contribution introduced in Chapter 4 is the introduction of a new method, t-discarding, to speed up dominance checks. This method can be applied to the pruning and filtering checks keeping the admissibility of the algorithm. This technique has been applied to multiobjective and goal-based search, yielding two new algorithms: NAMOA$^*_{\text{dr}}$ and LEXGO$^*_{\text{dr}}$. Their properties are also analyzed in this section.

We organize the relevant formal properties for a multicriteria search algorithm as follows:

Admissibility All algorithms introduced in this thesis are theoretically proven to be admissible. The formal proofs on the admissibility of NAMOA* were presented in (Mandow & Pérez de la Cruz, 2005; Mandow & Pérez de la Cruz, 2010). LEXGO*

properties of admissibility will be presented further in Section 5.2. Finally, the t-discarding technique which is employed in NAMOA$^*_{\text{dr}}$ and LEXGO$^*_{\text{dr}}$ is analyzed in Section 5.3.

Efficiency A standard in formal analyses to measure the efficiency of multicriteria search algorithms is the number of explored labels. A recent study showed that NAMOA* is optimal according to this measure when used with consistent lower bounds (Mandow & Pérez de la Cruz, 2010). NAMOA* has been shown to expand an equal or smaller number of labels when using more informed consistent lower bound functions (Mandow & Pérez de la Cruz, 2010). LEXGO*, in particular, always expands a subset of the labels expanded by NAMOA*, see Section 5.2.1 for further details.

The number of explored labels can be a good measure of the space requirements of the algorithms. However, in multicriteria search the time requirements are influenced by other factors as well. The algorithms that use t-discarding expand the same set of labels as the traditional ones. However, the experimental evaluation reveals that the time performance is greatly improved. Therefore, we also analyze another important measures for the time performance of multicriteria search, the number of dominance comparisons and the cardinality of the non-dominated sets used to check dominance. These are the key of the impressive performance of the t-discarding technique.

This chapter is organized as follows. Section 5.1 summarizes formal properties of NAMOA*. The same aspects are comparatively analyzed for LEXGO* in Section 5.2. Finally, the t-discarding technique is formally proved to be time efficient and characterized employing NAMOA$^*_{\text{dr}}$ and LEXGO$^*_{\text{dr}}$ as examples of use. Finally, a brief discussion is presented.

5.1 Formal characterization of NAMOA*

The formal properties of NAMOA* have been recently introduced by Mandow & Pérez de la Cruz (2010) and further studied by Machuca (2012). In this section, we briefly review previous theoretical properties of NAMOA*, since it is the basis on which LEXGO* has been devised. This will allow us to analyze only the new features brought by LEXGO*, the t-discarding technique and their implications on the admissibility and efficiency.

5.1.1 Admissibility

The results that follows are taken from Mandow & Pérez de la Cruz (2010) and Machuca (2012).

Property 5.1 (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 non-dominated*** *optimal solutions, or does not terminate if there are infinite solutions. NAMOA* is admissible even on infinite graphs with some additional assumptions:*

$$\textbf{a)}\ \forall n \in N \wedge \forall \vec{h} = (h_1, \ldots, h_q) \in H(n), \quad \forall k \in [1, q],\ h_k(n) \geq 0$$

$$\textbf{b)}\ \forall (n, n') \in A, \wedge \forall \vec{c}(n, n') \in \vec{c}, \quad \forall k \in [1, q],\ c_k(n, n')\ \geq \epsilon > 0$$

Several important properties of NAMOA* follow. These are analogous to those of the single objective A* algorithm.

Theorem 5.1 *(Mandow & Pérez de la Cruz, 2010, Theorem 4.2) 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) *$\vec{g}(P_i^*) \in G_{op}(n_i)$*

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

That is to say the algorithm never discards Pareto optimal solutions.

Theorem 5.2 *(Mandow & Pérez de la Cruz, 2010, Theorem 4.3) If there is at least a solution path P^*, the algorithm terminates even on infinite graphs.*

Corolary 5.1 *(Mandow & Pérez de la Cruz, 2010, Corollary 4.4) Whenever there is at least a solution path P^*, the set of non-dominated solution costs C^* is finite.*

Lemma 5.1 *(Mandow & Pérez de la Cruz, 2010, Lemma 4.5) Each path $P \in \mathbb{P}_{sn}$ selected from OPEN for expansion satisfies upon selection that,*

$$\exists \vec{h} \in H(n) \quad | \quad \nexists \vec{c}^{\,*} \in C^*, \quad \vec{c}^{\,*} \prec \vec{g}(P) + \vec{h} \tag{5.1}$$

Theorem 5.3 *(Mandow & Pérez de la Cruz, 2010, Theorem 4.6) A dominated solution can never be selected for expansion.*

Corolary 5.2 *(Mandow & Pérez de la Cruz, 2010, Corollary 4.7) The set of found solution vectors, COSTS, is at any time a subset of the set of all non-dominated solution cost vectors, i.e. $COSTS \subseteq C^*$.*

Theorem 5.4 *(Mandow & Pérez de la Cruz, 2010, Theorem 4.9) Since NAMOA* satisfies all the above conditions, NAMOA* is admissible.*

Admissibility (i.e. the algorithm is exact and returns the whole set of solutions to the problem) is an important property. However, it is also important to prove that the efficiency of the algorithm improves with more precise lower bounds.

5.1.2 Efficiency of lower bounds and optimality

Pathological behavior has been observed in another multiobjective search algorithm called MOA* (Stewart & White, 1991). Although MOA* is admissible, it has been proven to decrease performance in certain cases with more informed lower bounds (Pérez de la Cruz et al., 2013). Fortunately, NAMOA* has been proven to enjoy efficiency properties analogous to those of A^*.

Definition 5.1 *(Mandow & Pérez de la Cruz, 2010, 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:*

$$\exists \vec{h} \in H(n_i) \quad | \quad \nexists \vec{c} \in C, \quad \vec{c} \prec \vec{g}(P_i) + \vec{h} \tag{5.2}$$

By definition, a C^*-bounded path will never be filtered (see Lemma 5.1 and Theorem 5.3). Therefore, such paths will be either selected for expansion or pruned.

Theorem 5.5 *(Mandow & Pérez de la Cruz, 2010, 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 non-dominated path from s to n*

b) *P be C^*-bounded*

Theorem 5.6 *(Mandow & Pérez de la Cruz, 2010, Theorem 5.10) Let $H_1(n)$ and $H_2(n)$ be two admissible lower bounds 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 lower bound 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 5.2 (Efficiency) *When $\forall n \in N,\ H(n) = \{\vec{0}\}$, NAMOA* is analogous to the blind algorithm of Martins (1984a) 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 non-dominated 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 lower bound functions, more actual suboptimal alternatives can be pushed out of the Pareto frontier with **more informed** lower bounds, reducing search effort.*

Property 5.3 (Optimality) *NAMOA* has been proved to be optimal in the number of path expansions among the class of exact best-first algorithms when using consistent distance estimates (Mandow & Pérez de la Cruz, 2010), i.e. no algorithm in this class provided only with the same information could avoid exploring a single label explored by NAMOA* without compromising admissibility.*

5.2 Formal characterization of LEXGO*

This section proves some relevant properties of LEXGO*. First, we will show that LEXGO* is at least as *efficient* as NAMOA* in terms of label expansions, i.e. it always expands a subset of the labels expanded by NAMOA*. Then, we will show that it is *admissible*, i.e. it always returns the set of all goal-optimal solutions. Again, these properties are analogous to those of the single objective A^* search algorithm.

The proofs presented in this section rely on a set of reasonable assumptions, analogous to those presented in Section 5.1.1 to prove the admissibility of NAMOA* and other multiobjective label-setting algorithms:

Assumption 5.1 *The graph $G = (N, A)$ to be searched is locally finite, i.e. only a finite number of arcs emanate from each node.*

Assumption 5.2 *The lower bound function $\vec{h}(n)$ is consistent.*

5.2.1 Efficiency

Let us consider first the question of efficiency. LEXGO* is essentially a version of NAMOA* with additional pruning and filtering rules. However, simply adding additional discarding rules to NAMOA* does not necessarily guarantee that the resulting algorithm will explore a subset of the labels expanded by NAMOA*. The example in Figure 5.1 illustrates the case for an arbitrary pruning rule. Let us assume $\forall n \ \vec{h}(n) = \vec{0}$. There are two non-dominated paths from s to n_1 with costs (8,6) and (9,1), respectively. Let us assume that by a certain arbitrary rule the path with cost (8,6) prunes the one with cost (9,1). There are two paths from s to n_2 through n_1 with costs (9, 14) and (10, 9). The latter dominates the path from s to n_2 with cost (10,10). However, due to the pruning rule, it will never be generated, and the dominated path with cost (10,10) will need to be expanded. In other words, the inclusion of an arbitrary pruning rule may lead to the exploration of labels never considered by NAMOA*.

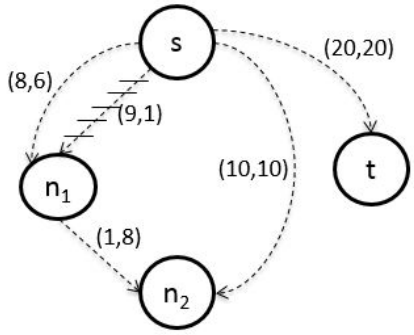


Figure 5.1. A pruning rule prunes a path to n_1 with cost (9,1) leading to the expansion of the dominated label (10,10) in n_2.

So we must formally show that the additional rules of LEXGO* guarantee that only a subset of the labels expanded by NAMOA* are actually considered. To do so, Lemma 5.2 analyzes the relation between pruning, goal, and Pareto preferences. Then, Theorem 5.7 proves the desired efficiency of LEXGO*, and finally, Theorem 5.8 establishes its admissibility.

Lemma 5.2 *Assume $\vec{\epsilon} \succeq \vec{0}$. Then*
a) If $\vec{y} \prec_P \vec{y'}$ then $\vec{y} + \vec{\epsilon} \prec_G \vec{y'} + \vec{\epsilon}$.
b) If $\vec{y} \prec_P \vec{y'}$ then $\vec{y} + \vec{\epsilon} \prec_P \vec{y'} + \vec{\epsilon}$.
c) If $\vec{y} \prec_P \vec{y'}$ and $\vec{y'} \prec \vec{y''}$, then $\vec{y} \prec_P \vec{y''}$.

Proof. Notice that, by definition,

$$\delta_i(\vec{y}, \vec{y'}) = 0 \quad \Rightarrow \quad \forall k \in I_i \quad s_k \geq s'_k \tag{5.3}$$

Additionally, assume $d_i(\vec{y}) = d_i(\vec{y'})$. Since $\vec{y}$ has greater or equal slack than $\vec{y'}$ for all goals in level i, then it is straightforward that,

$$\forall \vec{\epsilon} \succeq \vec{0}, \quad d_i(\vec{y'} + \vec{\epsilon}) \geq d_i(\vec{y} + \vec{\epsilon}) \tag{5.4}$$

Notice again that, by definition,

$$\delta_j(\vec{y}, \vec{y'}) < d_j(\vec{y'}) - d_j(\vec{y}) \quad \Rightarrow \quad \forall \vec{\epsilon} \succeq \vec{0}, \quad d_j(\vec{y'} + \vec{\epsilon}) > d_j(\vec{y} + \vec{\epsilon}) \tag{5.5}$$

Property (a) follows then from the definition of goal preferences. Assume $\vec{y} \prec_P \vec{y'}$, and that $\exists j \quad d_j(\vec{y}) < d_j(\vec{y'}) \wedge (\forall i < j \quad d_i(\vec{y}) = d_i(\vec{y'}))$. Then, from equations 5.4 and 5.5, $\vec{y'} + \vec{\epsilon}$ will not have better deviation over $\vec{y} + \vec{\epsilon}$ for any of the first j levels, and will have strictly worse deviation for at least one of them, i.e. $\vec{y} + \vec{\epsilon} \prec_G \vec{y'} + \vec{\epsilon}$.

For part (b) we still have to prove the additional constraints imposed on cross-slacks. Let us denote by s''_k and s'''_k the slack for goal k of vectors $\vec{y} + \vec{\epsilon}$ and $\vec{y'} + \vec{\epsilon}$ respectively. For all levels $i < j$ we have,

$$\delta_i(\vec{y}, \vec{y'}) = 0 \quad \Rightarrow \quad \forall k \in I_i \; s_k \geq s'_k \quad \Rightarrow \quad \forall k \in I_i \quad s''_k \geq s'''_k \quad \Rightarrow \quad \delta_i(\vec{y} + \vec{\epsilon}, \vec{y'} + \vec{\epsilon}) = 0$$

and also $d_i(\vec{y'} + \vec{\epsilon}) \geq d_i(\vec{y} + \vec{\epsilon})$.

If for some $m < j \quad d_m(\vec{y'} + \vec{\epsilon}) > d_m(\vec{y} + \vec{\epsilon})$, then $\delta_m(\vec{y} + \vec{\epsilon}, \vec{y'} + \vec{\epsilon}) = 0 < d_m(\vec{y'} + \vec{\epsilon}) - d_m(\vec{y} + \vec{\epsilon})$, and the property holds. Otherwise, we need to prove that the condition on cross-slacks still holds for level j. Let us define $\delta^i(\vec{y}, \vec{y'}) = w_i \times \max(0, s'_i - s_i)$. The following is an alternate definition of formula 4.3,

$$\delta_j(\vec{y}, \vec{y'}) = \sum_{m \in I_j} \delta^m(\vec{y}, \vec{y'})$$

Analogously, let us define $d^i(\vec{y}) = w_i \times \max(0, y_i - t_i)$. Then,

$$d_j(\vec{y}) = \sum_{m \in I_j} d^m(\vec{y})$$

Now, we analyze for each goal $m \in I_j$ its influence in deviations and cross-slack. We have three cases to consider:

- When $s_m = s'_m$, deviations increase in the same amount (i.e. their relative difference does not change) and $\delta^m(\vec{y} + \vec{\epsilon}, \vec{y'} + \vec{\epsilon}) = \delta^m(\vec{y}, \vec{y'}) = 0$.
- If $s_m > s'_m$, then $d^m(\vec{y'}) - d^m(\vec{y}) \leq d^m(\vec{y'} + \vec{\epsilon}) - d^m(\vec{y} + \vec{\epsilon})$, i.e. the relative difference between deviations can never decrease. Since $\delta^m(\vec{y}, \vec{y'}) = \delta^m(\vec{y} + \vec{\epsilon}, \vec{y'} + \vec{\epsilon})$, the condition will hold for the goal.
- If $s_m < s'_m$, then we have to consider three distinct cases:
  - When $0 \leq \epsilon_m \leq s_m < s'_m$, both deviations are zero, their relative difference remains zero and $\delta^m(\vec{y}, \vec{y'})$ does not change.

- When $s_m < \epsilon_m \leq s'_m$, we have $[d^m(\vec{y'}) - d^m(\vec{y})] - [d^m(\vec{y'} + \vec{\epsilon}) - d^m(\vec{y} + \vec{\epsilon})] = w_m \times (\epsilon_m - s_m)$. However, we also have $\delta^m(\vec{y}, \vec{y'}) - \delta^m(\vec{y} + \vec{\epsilon}, \vec{y'} + \vec{\epsilon}) = w_m \times (\epsilon_m - s_m)$, i.e. it decreases in the same amount as before, and the inequality still holds for goal m.
- When $s_m < s'_m < \epsilon_m$, we have $[d^m(\vec{y'}) - d^m(\vec{y})] - [d^m(\vec{y'} + \epsilon) - d^m(\vec{y} + \epsilon)] = w_m \times (s'_m - s_m)$. However, $\delta^m(\vec{y}, \vec{y'}) - \delta^m(\vec{y} + \vec{\epsilon}, \vec{y'} + \vec{\epsilon}) = w_m \times (s'_m - s_m)$, i.e. it also decreases in the same amount as before, and the inequality still holds for goal m.

Part (c) is quite straightforward. Notice that,

$$\vec{y'} \prec \vec{y''} \Rightarrow \forall l \forall k \in I_l \quad s'_k \geq s''_k \tag{5.6}$$

If $\vec{y} \prec_P \vec{y'}$, then we have that for all levels $i < j$, $\delta_i(\vec{y}, \vec{y'}) = 0$, $\delta_i(\vec{y}, \vec{y''}) = 0$, and $d_i(\vec{y''}) \geq d_i(\vec{y'}) = d_i(\vec{y})$.

Let us examine level j. From equation 5.6 it follows that $\delta_j(\vec{y}, \vec{y'}) \geq \delta_j(\vec{y}, \vec{y''})$ and from dominance $d_j(\vec{y''}) \geq d_j(\vec{y'})$. In consequence,

$$d_j(\vec{y''}) - d_j(\vec{y}) \;\geq\; d_j(\vec{y'}) - d_j(\vec{y}) \;>\; \delta_j(\vec{y}, \vec{y'}) \;\geq\; \delta_j(\vec{y}, \vec{y''}) \tag{5.7}$$

and therefore $\vec{y} \prec_P \vec{y''}$. □

Theorem 5.7 *When the lower bound function is monotone LEXGO* explores a subset of the labels explored by NAMOA*, i.e. if NAMOA* does not explore a label $(n, \vec{g})$, LEXGO* will not explore it either.*

Proof. A label $(n, \vec{g}, \vec{f})$ is not explored by NAMOA* if (a) $\exists \vec{c^*} \in C^*$ such that $\vec{c^*} \prec \vec{f}$, or (b) $\vec{g}$ is dominated in n.

It is straightforward that LEXGO* never explores a label discarded by NAMOA* by condition (a). Since $C^*_G \subseteq C^*$, for all $c^* \in C^*$, either $c^* \in C^*_G$, or $\vec{d}_B = \vec{d^*} \prec_L \vec{d}(\vec{c^*})$. In the latter case, if for some $\vec{f}$, $\vec{c^*} \prec \vec{f}$, then $\vec{d^*} \prec_L \vec{d}(\vec{c^*}) \preceq_L \vec{d}(\vec{f})$. Therefore, LEXGO* filters the labels with equations 4.7 and 4.8.

Let us consider now labels discarded by NAMOA* by condition (b). Let us assume a non-dominated path $P = (s, n, \ldots, n_i, \ldots, n_k)$ to n_k represented by label $(n_k, \vec{g}, \vec{f})$, and its two subpaths $P_1 = (s, n, \ldots, n_i)$ and $P_2 = (n_{i+1}, \ldots, n_k)$. Let us also assume a dominated path $P' = (s, \ldots, n_k)$ to n_k in OPEN with label $(n_k, \vec{g'}, \vec{f'})$. Finally, lets assume that P_1 is the largest subpath of P to enter OPEN, with label $(n_i, \vec{g}_1, \vec{f}_1)$. This situation is depicted in Figure 5.2.

Let us assume label $(n_i, \vec{g}_1, \vec{f}_1)$ is in OPEN. Since the lower bound function is monotone, as defined in the assumptions, $\vec{g}_1 + \vec{h}_i \preceq \vec{g} + \vec{h}_k \prec \vec{g'} + \vec{h}_k$ and P' can never be selected by LEXGO*. If eventually, $n_i = n_k$ P' is dominated and pruned by P.

On the other hand, if $(n_i, \vec{g}_1, \vec{f}_1)$ is never selected and not in OPEN, then there must be some other path P_3 that pruned P_1, i.e. $\vec{f}(P_3) \prec_P \vec{f}(P_1)$. By Lemma 5.2(b), we have $\vec{f}(P_3P_2) \prec_P \vec{f}(P_1P_2)$. This fact, together with the fact that $\vec{f}(P_1P_2) \prec \vec{f}(P')$, leads us to conclude by virtue of Lemma 5.2(c), that $\vec{f}(P_3P_2) \prec_P \vec{f}(P')$, i.e. if a path prunes some other non-dominated path, then the extensions of the former will also prune those that would be pruned by the latter. Therefore, the property holds.□

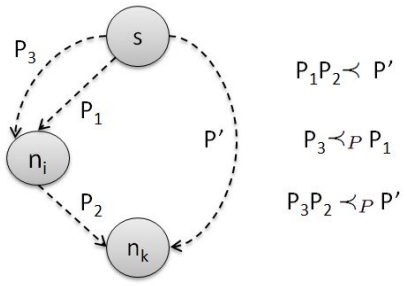


Figure 5.2. Scenario where a dominated path P' is pruned either by P_1P_2 or P_3P_2.

5.2.2 Admissibility

Once the efficiency of LEXGO* has been established, we turn our attention to admissibility, i.e. to prove that the subset of labels explored by LEXGO* still includes all goal-optimal solutions. A scalar algorithm is said to be *admissible* if it is guaranteed to return an optimal solution whenever a solution exists. We extend the definition as follows: a multiobjective search algorithm with goal-based preferences is *admissible* if it terminates with the set of *all* goal-optimal solutions to the problem.

Theorem 5.8 *Algorithm LEXGO* is admissible.*

Proof. LEXGO* is a label-setting algorithm that generates partial paths from the start node to the destination. Each partial path is either expanded, filtered, or pruned. A goal-optimal solution could be pruned by pruning conditions 4.5 or 4.6 as described in Section 4.1.1, or could be filtered by filtering conditions 4.7 or 4.8 from Section 4.1.2. By definition, a goal-optimal solution has a non-dominated cost. Since the optimality principle holds for dominated costs, neither pruning condition 4.5 nor filtering condition 4.7 will ever discard a goal-optimal solution.

The proof for condition 4.6 follows. Let us assume two paths $P^1 = (s, \ldots, n)$ and $P^2 = (s, \ldots, n)$, leading to the same node n, and an additional Pareto-optimal path $P^3 = (n, \ldots, t)$ leading from n to a destination node. Let us call $\vec{f^1} = \vec{f}(P^1) = \vec{g}(P^1) + \vec{h}(n)$, $\vec{f^2} = \vec{f}(P^2) = \vec{g}(P^2) + \vec{h}(n)$. Since the lower bound is optimistic, we know that $\vec{h}(n) \preceq \vec{g}(P^3)$. Let us call $\vec{e} = \vec{g}(P^3) - \vec{h}(n) \succeq \vec{0}$. Extending P^1 and P^2 with P^3, the costs of both solutions are respectively $\vec{f^{13}} = \vec{g}(P^1) + \vec{g}(P^3) = \vec{g}(P^1) + \vec{h}(n) + \vec{e} = \vec{f^1} + \vec{e}$ and $\vec{f^{23}} = \vec{g}(P^2) + \vec{g}(P^3) = \vec{g}(P^2) + \vec{h}(n) + \vec{e} = \vec{f^2} + \vec{e}$.

Let us assume that P^1 prunes P^2 in virtue of condition 4.6. Then $\vec{f^1} \prec_P \vec{f^2}$ and, by Lemma 5.2(a), $\vec{f^{13}} = \vec{f^1} + \vec{e} \prec_G \vec{f^2} + \vec{e} = \vec{f^{23}}$, so by this expansion P^2 does not lead to a better solution than P^1. Since no assumptions were made about P^3, the result holds for every expansion of n, therefore P^2 does not lead to a better solution than P^1 and the pruning is correct.

Finally, let us consider filtering condition 4.8, due to the lexicographic selection policy, the deviation of the first solution found $\vec{d_B} = \vec{d_t}$ is trivially equal or lexicographically better than the deviation of any other label in OPEN. No goal-optimal solution cost $\vec{c^*} \in C^*_G$ with deviation $\vec{d}(\vec{c^*})$ can have worse lexicographical deviation than $\vec{d_B}$ Therefore, filtering condition 4.8 never filters goal-optimal solutions.

Since LEXGO* never prunes nor filters goal-optimal solutions, the only remaining possibility is that they are all selected and found before termination, i.e. LEXGO* is admissible.□

5.3 Formal characterization of NAMOA$^*_{\text{dr}}$

This section proves some relevant properties of NAMOA$^*_{\text{dr}}$ under reasonable assumptions. The t-discarding method, see Section 4.2, introduces a new technique to check dominance of new alternatives against the solutions already found (filtering) and against partial paths to each node already explored (cl-pruning). We will show that NAMOA* is not affected when the t-discarding technique is applied instead of the standard dominance tests.

5.3.1 Admissibility

In order to demonstrate the admissibility of NAMOA$^*_{\text{dr}}$, we will make the following assumptions:

Assumption 5.3 *The lower bound function returns a single vector estimate for each node, i.e.* $H(n) = \{\vec{h}(n)\}$*, and satisfies the monotone property.*

Assumption 5.3 is satisfied by the precalculated lower bound proposed by Tung and Chew (Tung & Chew, 1992), which corresponds to the ideal point. This single-valued multiobjective lower bound has been shown to be very effective in practice (Machuca & Mandow, 2012).

Assumption 5.4 *The lexicographic order is applied to select non-dominated labels from OPEN, that is, if* $(n, \vec{g}, \vec{f})$ *is selected, then for all* $(n', \vec{g'}, \vec{f'}) \in$ *OPEN,* $\vec{f} \preceq_L \vec{f'}$.

Lexicographic order is a common choice in multiobjective search algorithms, since the lexicographic optimum is also non-dominated. Notice that dominance always implies lexicographical preference, that is, if $\vec{x} \prec \vec{y}$ then $\vec{x} \prec_L \vec{y}$, and if $\vec{x} \preceq \vec{y}$ then $\vec{x} \preceq_L \vec{y}$ (obviously the opposite is not true).

We will prove (Theorem 5.9) that under Assumptions 5.3 and 5.4 NAMOA$^*_{\text{dr}}$ is not affected when filtering and cl-pruning use t-discarding checks instead of standard dominance checks. We will prove first two auxiliary results (Lemmas 5.3 and 5.4).

Let $\vec{v}$ be a vector and X a set. We will write $X \preceq_L \vec{v}$ when $\vec{v}$ is an upper lexicographic bound of X, that is, when for every $\vec{v'} \in X$, $\vec{v'} \preceq_L \vec{v}$.

Lemma 5.3 *Given Assumptions 5.3 and 5.4, when a label* $l = (n, \vec{g}, \vec{f})$ *is selected from OPEN, then (i)* $G_{cl}(n) \preceq_L \vec{g}$*; and (ii) COSTS* $\preceq_L \vec{f}$.

Proof. Let us define OPEN$_t$ as the set of open labels at iteration t and assume $(m, \vec{g'}, \vec{f'})$ was selected at iteration t of the algorithm. Then, at that moment, for all $(n'', \vec{g''}, \vec{f''}) \in$ OPEN$_t$ we have (by Assumption 5.4) $\vec{f'} \preceq_L \vec{f''}$. Let us consider now the label $(n, \vec{g}, \vec{f})$ selected from OPEN$_{t+1}$ at the next iteration $t+1$. If it was already $(n, \vec{g}, \vec{f}) \in$ OPEN$_t$, then obviously $\vec{f'} \preceq_L \vec{f}$. If $(n, \vec{g}, \vec{f})$ is a new label, then n is a child

of m and $\vec{g} = \vec{g'} + \vec{c}(m,n)$. By Assumption 5.3, $\vec{h}(m) \preceq \vec{c}(m,n) + \vec{h}(n)$ and hence $\vec{f'} = \vec{g'} + \vec{h}(m) \preceq \vec{g'} + \vec{c}(m,n) + \vec{h}(n) = \vec{g} + \vec{h}(n) = \vec{f}$ and hence $\vec{f'} \preceq_L \vec{f}$. We have proved that if $(m, \vec{g'}, \vec{f'})$ is selected for expansion at iteration t and $(n, \vec{g}, \vec{f})$ at iteration $t+1$ then $\vec{f'} \preceq_L \vec{f}$; and it trivially follows by induction that if $(m, \vec{g'}, \vec{f'})$ is selected for expansion before $(n, \vec{g}, \vec{f})$ then $\vec{f'} \preceq_L \vec{f}$.

Now we can prove part (i) of the lemma. Let us consider the set $G_{cl}(n)$. If $\vec{g'} \in G_{cl}(n)$, then there is a closed label $(n, \vec{g'}, \vec{f'})$ at n that was selected before $(n, \vec{g}, \vec{f})$, so $\vec{f'} \preceq_L \vec{f}$. Since $\vec{f} = \vec{g} + \vec{h}(n)$ and $\vec{f'} = \vec{g'} + \vec{h}(n)$, then $\vec{g'} \preceq_L \vec{g}$, so $G_{cl}(n) \preceq_L \vec{g}$ as stated in part (i).

Let us consider now the set COSTS. If $\vec{f'} \in$ COSTS, it belongs to a label $(\gamma, \vec{f'}, \vec{f'})$ (where γ is the destination node) that has been already selected, so $\vec{f'} \preceq_L \vec{f}$, as stated in part (ii). □

Lemma 5.4 *Let $\vec{v}$ be a vector and X a set of vectors such that $X \preceq_L \vec{v}$. Then $\vec{v}$ is dominated by X iff $\vec{v}$ is t-discarded by X.*

Proof. The "if" direction is trivial: if $\vec{v}$ is t-discarded by X, both condition (a) and condition (b) in Definition 4.5 imply that there exists $\vec{v'} \in X$ such that $\vec{v'} \prec v$, hence $\vec{v}$ is dominated by X.

The "only if" direction is also immediate: Suppose $\vec{v}$ is dominated by X. Then there exists at least a $\vec{v'} \in X$ such that $\vec{v'} \prec v$. Suppose first that $t(\vec{v'}) \in T(X)$ and $v'_1 < v_1$. Then it must be $t(v'_1) \preceq t(v_1)$. So $\vec{v'}$ satisfies conditions imposed in Definition 4.5 (option a) to t-discard $\vec{v}$. On the other hand, if $v'_1 = v_1$ it must be $t(\vec{v'}) \prec t(\vec{v})$ and v' also satisfies conditions imposed in Definition 4.5 (option b). Let us suppose now $t(\vec{v'}) \notin T(X)$. Then $t(\vec{v'})$ is dominated by a certain $t(\vec{v''})$, that is, $t(\vec{v''}) \prec t(\vec{v'}) \preceq t(v)$. And, since $\vec{v''} \in X \preceq_L \vec{v}$, it must be $v''_1 \leq v_1$. But if $v''_1 < v_1$, $\vec{v''}$ allows to t-discard $\vec{v}$ (option a); and if $v''_1 = v_1$, $\vec{v''}$ allows to t-discard $\vec{v}$ (option b). So, in any case, if $\vec{v}$ is dominated by X then $\vec{v}$ is t-discarded by X. □

Theorem 5.9 *Under Assumptions 5.3 and 5.4, the workings of NAMOA* are unaffected if filtering and/or cl-pruning are defined as follows:*

- *Filtering. Discard $(n, \vec{g}, \vec{f})$ if $\vec{f}$ is t-discarded by COSTS.*
- *Cl-pruning. Discard $(n, \vec{g}, \vec{f})$ if $\vec{g}$ is t-discarded by $G_{cl}(n)$.*

Proof. First note that by Lemma 5.3 when a label $l = (n, \vec{g}, \vec{f})$ is selected from OPEN, then $G_{cl}(n) \preceq_L \vec{g}$ and COSTS $\preceq_L \vec{f}$. Lemma 5.4 guarantees in that case that dominance and t-discarding by a set X are equivalent. So, a new label will be cl-pruned and/or filtered exactly in the same cases. □

5.3.2 Efficiency

The real advantage of t-discarding is time performance. A recent work (Mandow & Pérez de la Cruz, 2009) shows that when arc costs are integer in the interval $[c_i, c_a]$, $c_i, c_a > 0$, then the worst-case number of Pareto optimal costs reaching a node at depth d with q objectives is $O((dr)^{q-1})$, where $r = c_a - c_i + 1$. This is, therefore, a worst-case

bound on the size of the COSTS and $G_{cl}(n)$ sets, and in the number of dominance checks against those sets.

The use of t-discarding implies checking only against the $T(\text{COSTS})$ and $T(G_{cl}(n))$ sets. The size of vectors in these sets is $q' = q - 1$, hence the worst-case size of the $T(\text{COSTS})$ and $T(G_{cl}(n))$ sets is only $O((dr)^{q-2})$.

Let us consider the class of problems with three objectives ($q = 3$). Then, the worst-case size of the $G_{cl}(n)$ and COSTS sets grows quadratically with depth and range of costs (and so does the worst-case number of dominance comparisons). However, with t-dominance this growth is limited to a linear case. The next chapter presents a set of experiments to evaluate the effectiveness of this method in practice.

5.4 Formal characterization of LEXGO$^*_{\text{dr}}$

This section analyzes the applicability of the t-discarding technique to LEXGO*. t-discarding requires evaluation vectors $\vec{f}$ to be selected in lexicographical order by the algorithm. This is easy to achieve for multiobjective search with a lexicographical policy, but cannot be straightforwardly applied in goal-based search, due to goal-based search with a lexicographical selection order selects the lexicographical optimal deviation vector, not the lexicographical optimal non-dominated f-vector (like in multiobjective search). An scenario where t-discarding can be applied to LEXGO* preserving its admissibility follows.

5.4.1 Admissibility

Assumption 5.5 *The lexicographic order is applied to select goal-optimal labels from OPEN, that is, if $(n, \vec{g}, \vec{f}, \vec{d})$ is selected, then for all $(n', \vec{g'}, \vec{f'}, \vec{d'}) \in$ OPEN, $\vec{d} \preceq_L \vec{d'}$ and if $\vec{d} = \vec{d'}$ then $\vec{f} \preceq_L \vec{f'}$.*

By definition of a goal-optimal solution, the lexicographically optimal deviation vector according to the concatenation of deviation and estimate vectors, $\vec{d} \cdot \vec{f}$, is a goal-optimal alternative, i.e. it has a non-dominated deviation and it is a non-dominated vector.

Lemma 5.5 *Given Assumptions 5.3 and 5.5, when a label $l = (n, \vec{g}, \vec{f}, \vec{d})$ is selected from OPEN and $\vec{d} = \vec{0}$ then for all $(\vec{g'}, \vec{f'}, \vec{d'}) \in$ OPEN, $\vec{d} \cdot \vec{f} \preceq_L \vec{d'} \cdot \vec{f'}$ and (i) $G_{cl}(n) \preceq_L \vec{g}$; and (ii) COSTS $\preceq_L \vec{f}$.*

Proof. On one hand, when $\vec{d} = \vec{0}$, obviously, for all $(n', \vec{g'}, \vec{f'}, \vec{d'}) \in G_{cl}(n)$, $\vec{d'} = \vec{0}$ and in a similar way to Lemma 5.3, we can remove d-vectors and achieve the same result, i.e. $G_{cl}(n) \preceq_L \vec{g}$.

On the other hand, if $\vec{d} = \vec{0}$ and l is selected from OPEN, if there exists any $\vec{c^*} \in$ COSTS, by definition, $d(\vec{c^*}) = \vec{0}$. Therefore, in an equivalent way to Lemma 5.3, we can remove d-vectors and achieve the same result, i.e. COSTS $\preceq_L \vec{f}$. □

Theorem 5.10 *Under Assumptions 5.3 and 5.5, the workings of LEXGO$^*_{dr}$ are unaffected if filtering and/or cl-pruning are defined as follows:*

- *Filtering. Discard* $(n, \vec{g}, \vec{f}, \vec{d})$ *if* $\vec{f}$ *is t-discarded by COSTS.*
- *Cl-pruning. Discard* $(n, \vec{g}, \vec{f}, \vec{d})$ *if* $\vec{g}$ *is t-discarded by* $G_{cl}(n)$*.*

Proof. The same argument used in Theorem 5.9 applies to prove that dominance and t-discarding by a set X are equivalent in $\text{LEXGO}^*_{\text{dr}}$ when a label $l = (n, \vec{g}, \vec{f}, \vec{d})$ with $\vec{d} = \vec{0}$ is selected from OPEN. Thus, this new label will be cl-pruned and/or filtered exactly in the same cases for $\text{LEXGO}^*_{\text{dr}}$ as it was in LEXGO^*. □

5.5 Discussion

This section has reviewed the formal properties of NAMOA^*, and presented some analogous properties for the three algorithms introduced in this thesis. More precisely, the admissibility (i.e. the exactness of the algorithms) is formally proved for all of them and their relative efficiency is also examined.

NAMOA^* is known to be optimal in the number of path expansions among the class of admissible best-first algorithms when using consistent lower bounds. LEXGO^* has also been proven to be admissible.

The worst case scenario for LEXGO^* is when the whole Pareto set of solutions satisfies the goals. The important theoretical results showed in this chapter indicate that (a) provided with the same lower bounds, LEXGO^* always expands a subset of the labels expanded by NAMOA^*; (b) the superiority in runtime of $\text{NAMOA}^*_{\text{dr}}$ over NAMOA^*; and (c) the superiority in runtime of $\text{LEXGO}^*_{\text{dr}}$ over LEXGO^*. These important theoretical results do not completely settle the question of which algorithm is better in practice. We address a comprehensive empirical analysis of this question in Chapters 6 and 7.

Chapter 6

Empirical Analysis on Grid Problems

It doesn't matter how beautiful your theory is,
it doesn't matter how smart you are. If it
doesn't agree with experiment, it's wrong.

Richard P. Feynman (1918-1988)

One of the main goals of this thesis is to develop efficient procedures for the goal-based search problem. In Chapter 4 we introduced LEXGO*, a new search algorithm for lexicographic goal problems, as well as an improved procedure for efficient dominance checks in multicriteria search algorithms. This procedure can be applied to LEXGO* and NAMOA*, yielding algorithms LEXGO$^*_{\text{dr}}$ and NAMOA$^*_{\text{dr}}$. Formal properties of these algorithms were presented in Chapter 5. LEXGO* and LEXGO$^*_{\text{dr}}$ always return the set of all goal-optimal solutions to a lexicographic goal search problem, while NAMOA$^*_{\text{dr}}$ returns the set of all Pareto-optimal solutions to a Multiobjective Search Problem.

In this chapter we evaluate experimentally the performance of two different strategies for lexicographic goal problems: (a) use a multiobjective search algorithm (like NAMOA* or NAMOA$^*_{\text{dr}}$), and then select the goal-optimal solutions among those in the Pareto set, (b) use a specific goal-based algorithm (like LEXGO* or LEXGO$^*_{\text{dr}}$).

First, we shall describe the experimental setup in Section 6.1, which is based on the use of randomly generated grid problems with three to five criteria. We analyze time and space performance of the algorithms for problems of increasing depth. We also evaluate performance for different kinds of goal preferences. More precisely, we set up goals that range from unsatisfiability to satisfability by increasingly larger subsets of the Pareto set. Chapter 7 will present additional experiments over route planning problems in realistic road maps.

We perform systematic comparisons between the algorithms. First we compare LEXGO* against full Pareto search with NAMOA* (using both lexicographic and linear aggregation selection policies in both algorithms) in Section 6.2. Then, we evaluate the algorithms that employ the dimensionality reduction technique against the standard ones. First, we compare in Section 6.3 NAMOA* with a linear aggregation selection

policy against our new version NAMOA$^*_{\text{dr}}$. After that we compare LEXGO$^*_{\text{dr}}$ against LEXGO* in Section 6.4. Finally, we evaluate both versions that employ t-discarding over problems with goals. A summary and conclusions are presented at the end of this chapter.

6.1 Experimental setup

This chapter evaluates the performance of algorithms over sets of randomly generated grids. Furthermore, we have generated three uncorrelated criteria in order to evaluate search with goal-based preferences and three to five uncorrelated criteria when we evaluate multiobjective search without preferences (NAMOA* versus NAMOA$^*_{\text{dr}}$). The solution depth is used as an indicator of problem difficulty. All multiobjective search algorithms analyzed in this thesis employ the ideal point as lower bound (described in Section 2.6.3).

We use randomly generated grids which are a standard test bed in the evaluation of multicriteria search algorithms (Machuca et al., 2012) (Raith & Ehrgott, 2009). Square bi-dimensional grids of 100×100 nodes with a vicinity of four neighbors were generated as described in Section 3.2.1. The start node is placed at the grid center and a single destination node is placed in the diagonal from the center to the bottom right corner. Different solution depths are considered, varying from 20 to 100. A set of five different problems was generated for each solution depth. For each arc a vector $\vec{c}(i,j) = (c_1, c_2, ..., c_q)$ of q integer scalar costs was randomly generated in the range [1,10] using an uniform distribution, i.e. leading to uncorrelated objectives.

In order to assess search with goal-based preferences two different classes of experiments were carried out. Three goals grouped in two priority levels were considered, where target values are defined in terms of the ideal and nadir points. The first class of experiments defines five different sets of targets using the constant $k_1 \in \{0, 0.25, 0.5, 0.75, 1\}$ for both priority levels (see Equation 3.1). For the second class, targets of the first level are fixed for $k_1 = 0.75$ and $k_1 = 0.5$, using the constant $k_2 = k_1 \times k'$ to set the targets of the second priority level, with $k' \in \{0.25, 0.5, 0.75, 1\}$ (see Equation 3.2). The complete description regarding the generation of these classes of experiments can be seen in Section 3.2.4.

Regarding multiobjective search performance, the experiments over NAMOA* and NAMOA$^*_{\text{dr}}$ are carried out with three, four and five uncorrelated criteria. All solution depths for each number of criteria range from $d = 20$ to the maximum solution depth by increments of 10. The maximum solutions depths considered are $d = 100$, $d = 50$, and $d = 40$ for $q = 3$, $q = 4$ and $q = 5$, respectively.

This study analyzes for NAMOA* and LEXGO* the following aspects as a function of solution depth:

1. The percentage of goal-optimal solutions returned by LEXGO* relative to the size of the full Pareto set returned by NAMOA*.

2. The percentage of scanned labels by LEXGO* relative to the number of scanned labels by NAMOA*.

3. The relative runtime requirements of LEXGO* to NAMOA*.

Additionally, in order to analyze the performance of the t-discarding technique the following aspects are analyzed as a function of solution depth:

1. The size of the truncated sets of closed labels ($T(G_{cl})$) and solutions (T(COSTS)) relative to the complete sets of NAMOA*.
2. The percentage of cl-pruned and filtered labels over the total number of discarded labels.
3. Time requirements of t-discarding algorithms ($\text{NAMOA}^*_{\text{dr}}$ or $\text{LEXGO}^*_{\text{dr}}$) relative to their reference algorithms (NAMOA* or LEXGO*).

The algorithms were implemented to share as much code as possible. The programming language used was ANSI Common Lisp using LispWorks Professional 6.01 (64-bit). The OPEN queue of alternatives was implemented as a binary heap but only the current best estimate of each node is kept in OPEN at each iteration. The G_{op} and G_{cl} sets are ordered according to the label selection policy employed by the algorithm. Lexicographic and linear orders were used to choose among non-dominated open alternatives in NAMOA* and LEXGO*. Their counterparts with the t-discarding technique $\text{NAMOA}^*_{\text{dr}}$ and $\text{LEXGO}^*_{\text{dr}}$ employ the lexicographic order.

Grid problems were run on an Intel Core i7 3612QM at 2.1 GHz and 4 GB of DDR3 RAM. All experiments were run on a single thread.

6.2 LEXGO* vs NAMOA*

This section presents the comparison between NAMOA* and LEXGO*. The study tested both the lexicographic and linear selection orders (see Definitions 2.13 and 2.14 for further description of the OPEN selection orders employed in this thesis). Regarding NAMOA*, the first variant is called $\text{NAMOA}^*_{\text{lex}}$, and uses a lexicographic order of selection. The second, called $\text{NAMOA}^*_{\text{lin}}$, uses an order of selection based on a linear aggregation of vector components. Similarly, the first variant of LEXGO* is called $\text{LEXGO}^*_{\text{lex}}$ and the second $\text{LEXGO}^*_{\text{lin}}$. These four alternatives are evaluated over the two classes of experiments described in Section 6.1. Additionally, the performance of LEXGO* was evaluated with and without the deviation-based pruning (see Equation 4.6), in order to evaluate the effectiveness of this newly introduced pruning condition.

Notice that both algorithms, NAMOA* and LEXGO*, return exactly the same set of solutions regardless the label selection policy, i.e. both $\text{NAMOA}^*_{\text{lex}}$ and $\text{NAMOA}^*_{\text{lin}}$ return the full Pareto set of efficient solutions to the problem, that is referred to as C^*, whereas $\text{LEXGO}^*_{\text{lex}}$ and $\text{LEXGO}^*_{\text{lin}}$ return the same set of goal-optimal solutions. Thus, the first comparison between them is devoted to analyze the percentage of goal-optimal solutions regarding the full Pareto set. Then, the space and runtime performance are analyzed. The former is measured by the relative number of scanned labels, and the latter by a runtime performance comparison when the lexicographic or the linear aggregation selection orders are employed.

These experiments were partly published by the author and reported in (Pulido et al., 2014). The results in this thesis go beyond those experiments and also consider the linear selection order and $k_1 = 0.75$ in class II experiments.

Table 6.1. Class I experiments on grids, average percentage of goal-optimal solution vectors returned by LEXGO* relative to average $|C^*|$ as a function of solution depth. An asterisk (*) indicates that the goals could not be satisfied.

		LEXGO*					
	NAMOA*	1	0.75	0.5	0.25	0	k_1
d	Avg. $\|C^*\|$	%	%	%	%	%	
20	122	100	74.3	20.5	0.98*	0.82*	
30	302	100	77.6	22.2	0.33*	0.33*	
40	694	100	78.7	20.8	0.14*	0.17*	
50	1,599	100	78.2	16.8	0.06*	0.06*	
60	2,007	100	83.0	24.6	0.06*	0.05*	
70	2,561	100	82.4	24.6	0.04*	0.04*	
80	5,423	100	82.3	20.3	0.02*	0.02*	
90	5,912	100	77.7	21.0	0.02*	0.03*	
100	8,307	100	77.9	17.0	0.01*	0.01*	

6.2.1 Analysis on class I experiments

Table 6.1 shows reductions in the number of Pareto-optimal solution vectors returned by LEXGO* relative to the full Pareto set C^*. It can be observed that for large values of k_1 the subset of Pareto-optimal solution vectors returned by LEXGO* is somewhat reduced. However, for $k_1 = 0.5$ only about 20% of the Pareto set is returned. For $k_1 = 0.25$ or $k_1 = 0$, no solution satisfies all goals, and a subset of only one or two Pareto-optimal solution cost vectors minimizing deviation is returned. This is also beneficial for the efficiency of LEXGO* since the number of computationally costly filtering checks is greatly reduced when compared to NAMOA*. The portion of the Pareto set returned for varying k_1 appears graphically illustrated in Figure 6.1 for a sample problem with solution depth $d = 100$.

This figure displays all Pareto-optimal solution vectors in cost (or attribute) space. The figure displays a box enclosing all Pareto-optimal solution vectors, delimited by the ideal and nadir points ($k_1 = 1$). Boxes delimiting the regions of cost space that would satisfy the goals established by parameter k_1 equal to 0.75, 0.5 and 0.25 are also displayed. In the case $k_1 = 0$ only the ideal point would satisfy the goals.

Figures 6.2 and 6.3 display the average number of scanned labels and average runtimes of LEXGO*and NAMOA*, respectively, both as a function of solution depth. In these graphics, we observe that LEXGO$^*_{\text{lex}}$ (6.3(a)) achieves important reductions in time of almost one order of magnitude for $k_1 = 0.5$, two orders of magnitude for $k_1 = 0.25$, and up to four orders of magnitude for $k_1 = 0$. These are explained in large part by the reduction observed in the number of labels scanned, i.e. half the number of labels for $k_1 = 0.5$, around one order of magnitude less for $k_1 = 0.25$ and three orders of magnitude less for $k_1 = 0$.

Figure 6.3(b) shows the average runtimes of LEXGO$^*_{\text{lin}}$ and NAMOA$^*_{\text{lin}}$. A small time overhead is also found for $k_1 = 0.75$ in the linear case, as well as poorer results when $k_1 = 0.5$. Notice that NAMOA$^*_{\text{lin}}$ is shown to be approximately two times faster

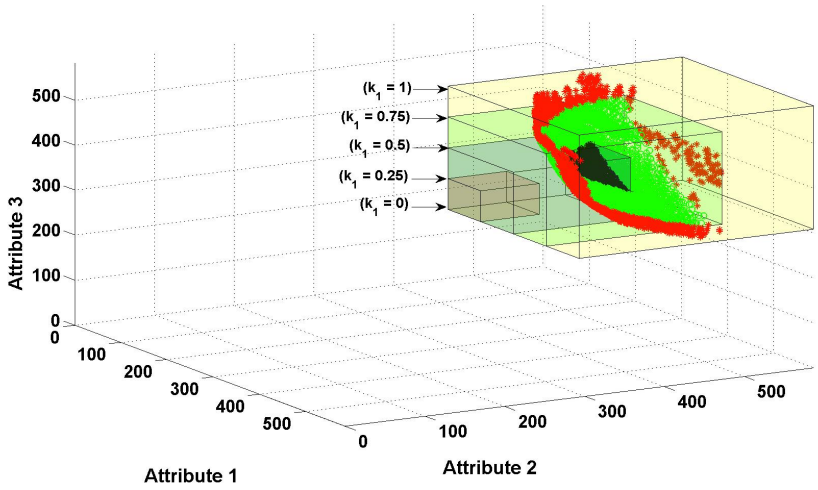


Figure 6.1. Three-dimensional Pareto frontier divided according to goal satisfiability for a sample problem with solution depth $d = 100$.

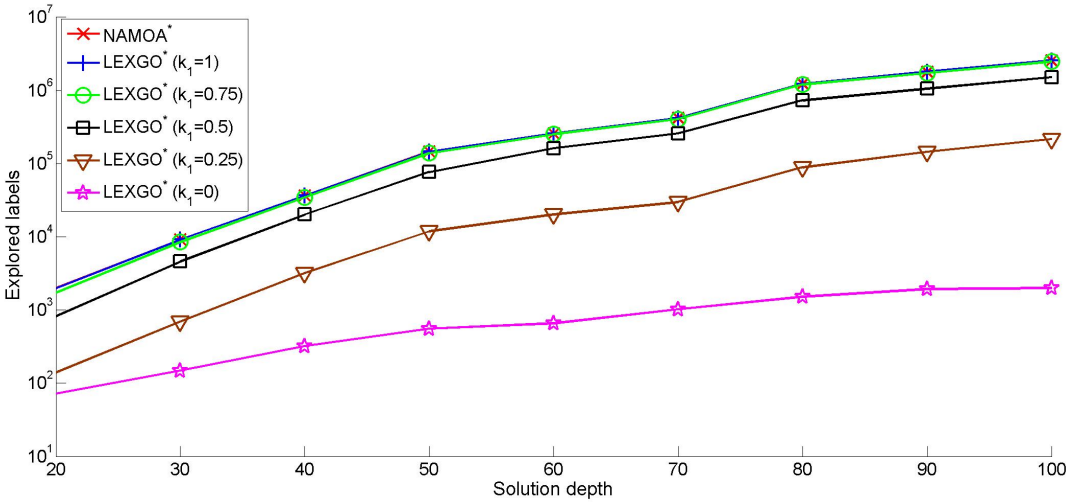


Figure 6.2. Class I experiments on grids, average number of scanned (explored) labels per solution depth for lexicographic selection order.

than $\text{NAMOA}^*_{\text{lex}}$. These results regarding the importance of the selection order are consistent with other recent studies (Machuca & Mandow, 2011; Iori et al., 2010). However, $\text{LEXGO}^*_{\text{lin}}$ is not always more efficient than $\text{LEXGO}^*_{\text{lex}}$. For $k_1 = 1$ and $k_1 = 0.75$ the linear aggregation order guides the search to find the solutions later than the lexicographic order and speeds up the runtime performance of algorithms that use the linear order whenever a big amount of solutions exist, due to a smaller number of filtering comparisons are needed. We further analyze in Section 6.3 the impact of the number of pruning and filtering comparisons in runtime performance.

Table 6.2 summarizes the space and runtime performance of LEXGO* relative to NAMOA* for $d = 100$ with the lexicographic and linear aggregation selection orders, respectively. The space performance is measured in scanned labels ($\sum G_{cl}$) and the runtime performance in seconds.

A small time overhead can be observed for $\text{LEXGO}^*_{\text{lex}}$ with $k_1 = 1$ when compared with $\text{NAMOA}^*_{\text{lex}}$. This time overhead is greater for $\text{LEXGO}^*_{\text{lin}}$, due to the comparison with the more efficient version of NAMOA* with the linear aggregation order. The time difference can be attributed to the extra calculations of deviation from targets needed by LEXGO* for all labels, and the extra checks for pruning and filtering that do not provide any advantage in this situation.

No significant difference is found regarding the scanned labels by $\text{NAMOA}^*_{\text{lex}}$ and $\text{NAMOA}^*_{\text{lin}}$, or the relative number to $\text{LEXGO}^*_{\text{lex}}$ and $\text{LEXGO}^*_{\text{lin}}$. In fact, NAMOA* should expand the same labels regardless of the selection order policy. However, the lazy filtering technique introduces a slight variation which is found not to be significant in any case (see Sanders & Mandow (2013) for a more detailed explanation of the lazy filtering technique).

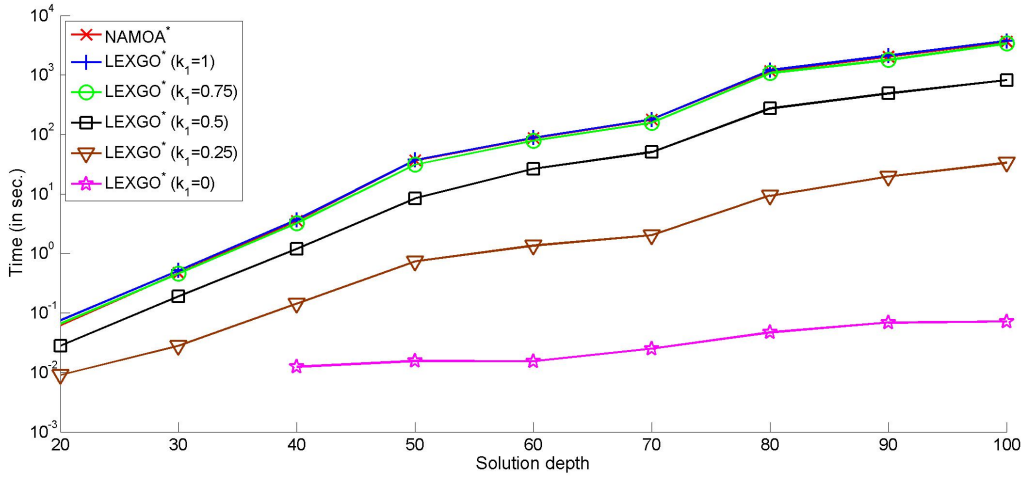
6.2.2 Analysis on class II experiments

Target values in the second class of experiments were defined using $k_1 = \{0.75, 0.5\}$, where k_2 is defined as in Equation 3.2. This allows us to analyze the case where targets for one goal are proportionally stricter, and the extreme case where some goals are satisfied and some not.

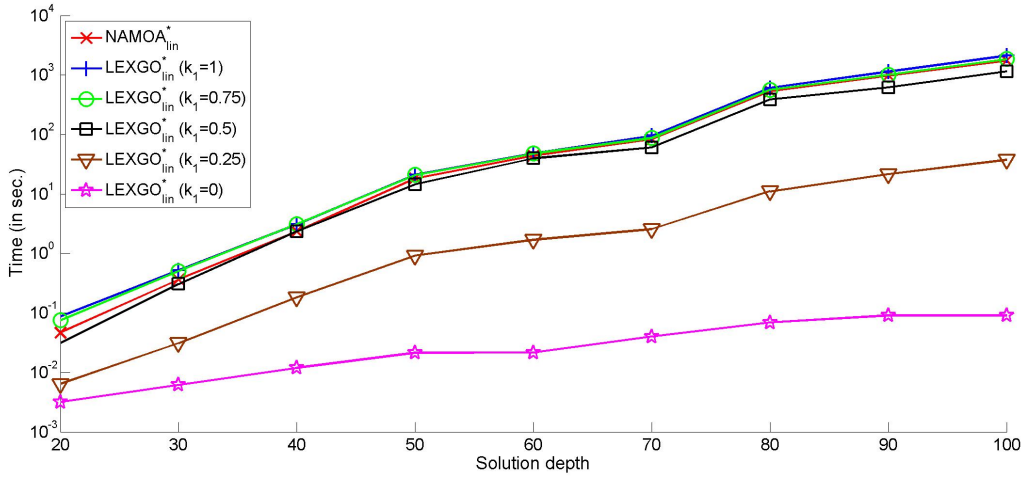
Tables 6.3 and 6.4 show percentages of Pareto goal-optimal solution costs and scanned labels of all values of k_2 in LEXGO* relative to NAMOA*. Figure 6.4 displays the average number of scanned labels as a function of solution depth for $k_1 = 0.75$ and $k_1 = 0.5$.

Table 6.5 displays the relative percentage of LEXGO* runtimes to NAMOA* employing lexicographic and linear selection orders. In a graphical manner, Figures 6.5 and 6.6 show, respectively, average runtimes for $k_1 = 0.75$ and $k_1 = 0.5$ with lexicographic and linear selection orders.

A progressive reduction in scanned labels and runtimes is observed as the value of k_2 decreases. For $d = 100$ and $k_2 = 0.75$, LEXGO* explores around 97% of labels explored by NAMOA*, but for $k_2 = 0.1875$ this value drops to only around 17.5%. Similarly, when $k_2 = 0.5$ LEXGO*explores around 60% of labels explored by NAMOA*, but for $k_2 = 0.125$ this value falls to around 12%. The percentage of Pareto goal-optimal solution costs returned also drops sharply as k_2 decreases. For $k_1 = 0.5$ and $k_2 = 0.125$, some problem instances could not satisfy all goals.



(a) Lexicographic selection order



(b) linear aggregation selection order

Figure 6.3. Class I experiments on grids, average runtime in seconds per solution depth for NAMOA* and LEXGO*.

Table 6.2. Class I experiments on grids, summary of the relative space and runtime performance of LEXGO* over NAMOA* for $d = 100$ experiments.

(a) Relative space performance of $\text{LEXGO}^*_{\text{lex}}$ over $\text{NAMOA}^*_{\text{lex}}$

	$\text{LEXGO}^*_{\text{lex}}$					
$\text{NAMOA}^*_{\text{lex}}$	1	0.75	0.5	0.25	0	k_1
$\sum G_{cl}$	%	%	%	%	%	
2,550,354	99.9	96.9	59.2	8.5	0.08	

(b) Relative space performance of $\text{LEXGO}^*_{\text{lin}}$ over $\text{NAMOA}^*_{\text{lin}}$

	$\text{LEXGO}^*_{\text{lin}}$					
$\text{NAMOA}^*_{\text{lin}}$	1	0.75	0.5	0.25	0	k_1
$\sum G_{cl}$	%	%	%	%	%	
2,598,427	99.9	97.2	59.1	8.3	0.08	

(c) Relative runtime performance of $\text{LEXGO}^*_{\text{lex}}$ over $\text{NAMOA}^*_{\text{lex}}$

	$\text{LEXGO}^*_{\text{lex}}$					
$\text{NAMOA}^*_{\text{lex}}$	1	0.75	0.5	0.25	0	k_1
Runtime (s)	%	%	%	%	%	
3,662.9	102.7	92.3	22.3	0.9	0.001	

(d) Relative runtime performance of $\text{LEXGO}^*_{\text{lin}}$ over $\text{NAMOA}^*_{\text{lin}}$

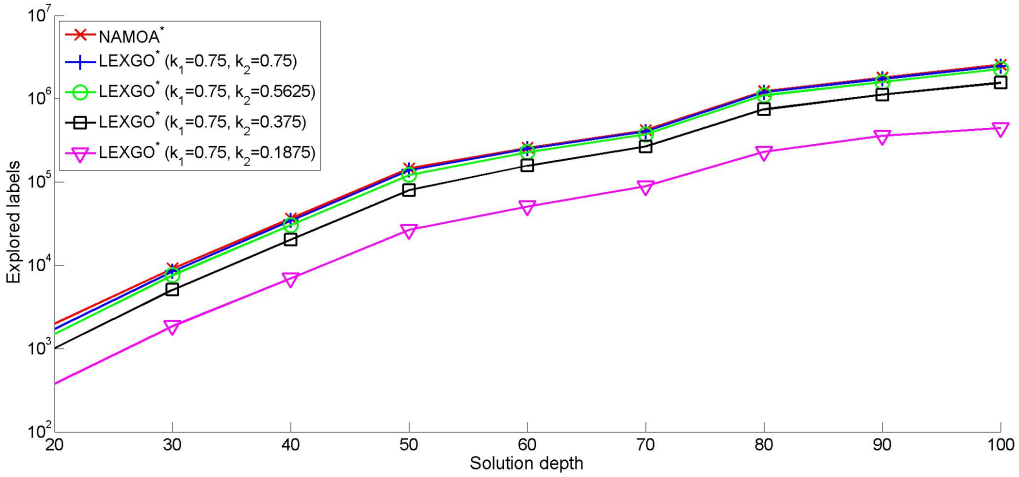
	$\text{LEXGO}^*_{\text{lin}}$					
$\text{NAMOA}^*_{\text{lin}}$	1	0.75	0.5	0.25	0	k_1
Runtime (s)	%	%	%	%	%	
1,754.4	120.5	105.3	65.2	2.1	0.005	

Table 6.3. Class II experiments on grids, LEXGO* average percentage of goal-optimal solution costs relative to C^*. An asterisk (*) indicates some of the five instances could not satisfy all goals, and two asterisks (**) that none of the five instances could satisfy all goals.

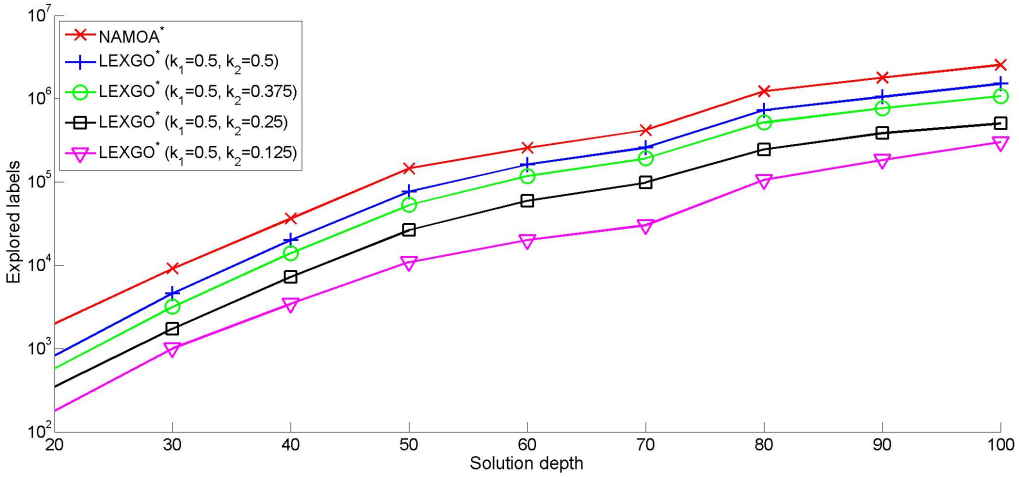
		LEXGO*								
		0.75				0.5				k_1
	NAMOA*	0.75	0.5625	0.375	0.1875	0.5	0.375	0.25	0.125	k_2
d	Avg. $\lvert C^*\rvert$	%	%	%	%	%	%	%	%	
20	122	74.3	59.1	35.2	10.6	20.5	11.5	3.3	*0.82	
30	302	77.6	63.4	38.5	15.5	22.2	12.3	5.6	*1.00	
40	694	78.7	59.3	34.0	12.2	20.8	10.5	3.3	**0.14	
50	1,599	78.2	57.2	33.6	12.3	16.8	8.4	2.6	*0.06	
60	2,007	83.0	62.6	35.9	12.6	24.6	13.1	3.8	*0.25	
70	2,561	82.4	66.0	41.9	14.6	24.6	13.6	4.2	*0.12	
80	5,423	82.3	64.9	36.1	11.6	20.3	9.1	1.7	**0.02	
90	5,912	77.7	63.6	38.2	10.7	21.0	10.5	2.1	**0.02	
100	8,307	77.9	60.5	35.7	10.6	17.0	8.1	1.1	**0.01	

Table 6.4. Class II experiments on grids, $\text{LEXGO}^*_{\text{lex}}$ average percentage of scanned labels ($\sum G_{cl}$) compared to $\text{NAMOA}^*_{\text{lex}}$.

		$\text{LEXGO}^*_{\text{lex}}$								
		0.75				0.5				k_1
	$\text{NAMOA}^*_{\text{lex}}$	0.75	0.5625	0.375	0.1875	0.5	0.375	0.25	0.125	k_2
d	$\sum G_{cl}$	%	%	%	%	%	%	%	%	
20	1,985	86.2	75.1	50.9	19.0	41.4	28.9	17.5	8.9	
30	9,164	92.4	83.1	55.5	20.3	50.3	34.8	18.9	11.0	
40	36,557	94.5	83.2	55.6	19.1	55.1	38.4	19.9	9.5	
50	145,823	95.5	83.4	54.7	18.3	52.7	36.4	18.2	7.5	
60	257,935	97.5	88.9	60.9	19.8	62.9	45.9	23.1	7.8	
70	420,056	96.6	88.9	63.5	21.2	61.5	45.7	23.5	7.2	
80	1,231,565	97.3	89.1	60.4	18.8	59.2	42.0	20.0	8.6	
90	1,789,607	96.7	88.8	62.9	20.0	59.1	43.1	21.6	10.3	
100	2,550,354	97.0	89.3	61.1	17.5	59.3	42.2	19.8	11.8	



(a) $k_1 = 0.75$



(b) $k_1 = 0.5$

Figure 6.4. Class II experiments on grids, average scanned labels per solution depth for NAMOA* and LEXGO* with lexicographic selection order.

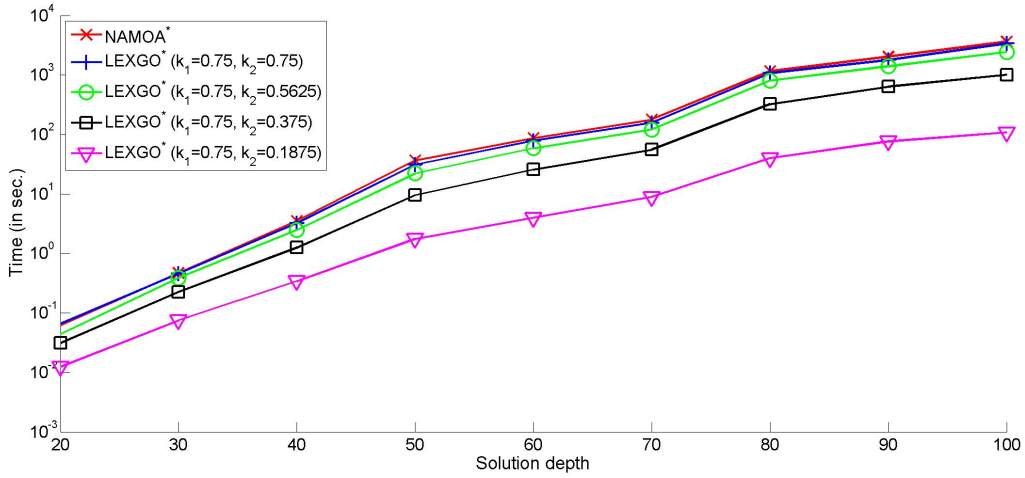
Table 6.5. Class II experiments on grids, LEXGO* runtimes (in seconds) percentage relative to NAMOA*.

(a) Lexicographic selection order

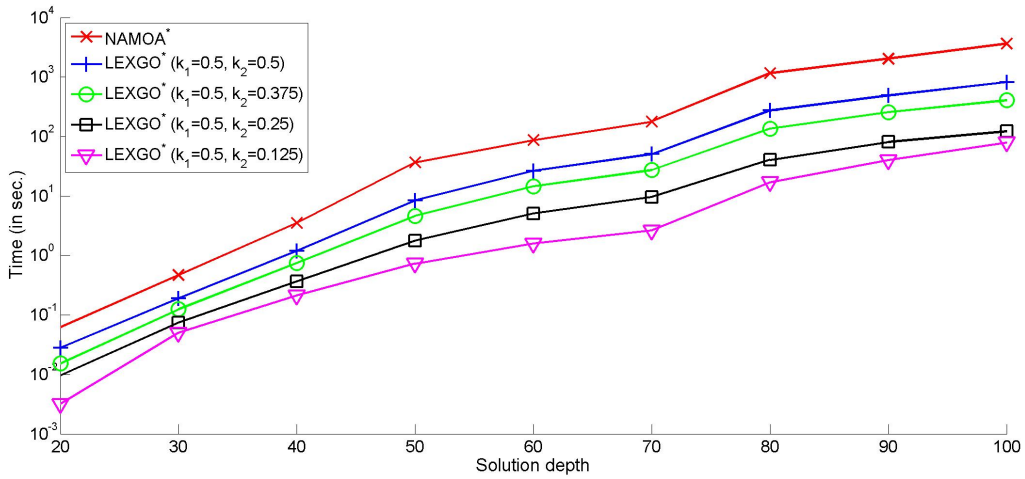
		LEXGO$^*_{\text{lex}}$								
		0.75				0.5				k_1
	NAMOA$^*_{\text{lex}}$	0.75	0.5625	0.375	0.1875	0.5	0.375	0.25	0.125	k_2
d	Runtime (s)	%	%	%	%	%	%	%	%	
20	0.06	100.0	66.6	50.0	16.6	33.3	16.6	1.6	1.6	
30	0.4	112.5	95.0	55.0	17.5	47.5	30.0	17.5	12.5	
40	3.5	91.4	68.5	34.2	8.5	34.2	20.0	8.5	5.7	
50	36.8	84.7	60.0	25.8	4.6	22.8	12.5	4.8	1.9	
60	86.9	90.6	67.4	29.4	4.4	30.4	16.6	5.8	1.8	
70	178.5	88.4	67.5	31.2	4.9	28.1	15.1	5.3	1.4	
80	1,164.1	92.2	69.5	27.8	3.4	23.5	11.6	3.4	1.4	
90	2,030.0	87.7	69.0	31.4	3.7	24.1	12.6	3.9	1.9	
100	3,662.9	92.3	66.7	27.6	2.9	22.3	11.1	3.3	2.1	

(b) linear aggregation selection order

		LEXGO$^*_{\text{lin}}$								
		0.75				0.5				k_1
	NAMOA$^*_{\text{lin}}$	0.75	0.5625	0.375	0.1875	0.5	0.375	0.25	0.125	k_2
d	Runtime (s)	%	%	%	%	%	%	%	%	
20	0.04	160.3	146.6	100.4	40.2	66.2	46.6	32.9	19.7	
30	0.3	138.4	149.5	99.9	30.8	83.8	58.1	28.2	16.3	
40	2.3	131.3	174.6	104.2	25.4	99.7	62.7	26.3	11.5	
50	18.3	113.6	144.0	92.5	17.6	79.2	50.4	18.1	5.3	
60	43.8	108.8	138.6	95.8	16.1	90.2	61.9	22.6	4.8	
70	83.3	104.4	117.4	101.3	19.5	72.6	56.9	21.2	4.2	
80	533.8	105.5	138.3	96.4	13.8	72.6	49.6	14.6	4.0	
90	981.3	102.6	121.4	92.4	12.8	63.3	48.2	15.5	4.8	
100	1,754.4	105.3	130.2	98.0	11.1	65.3	46.1	13.1	5.3	

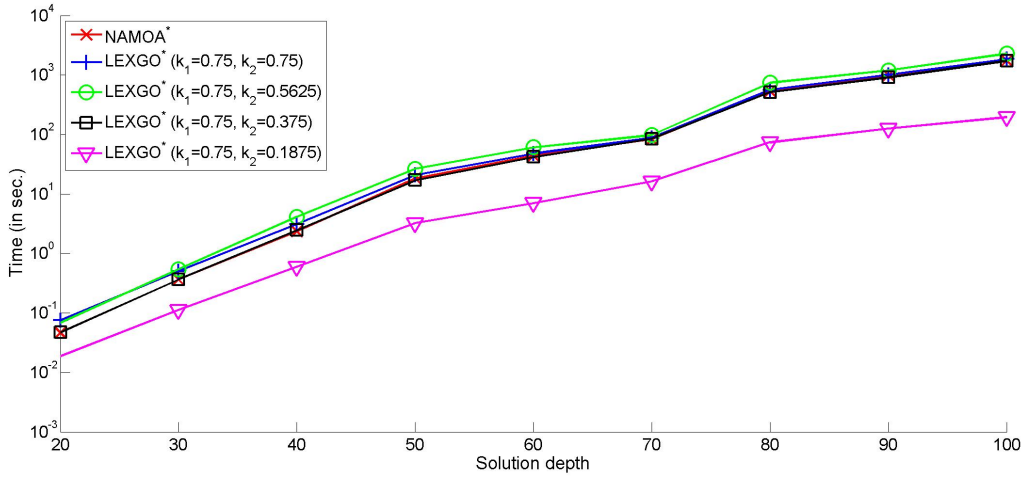


(a) $k_1 = 0.75$

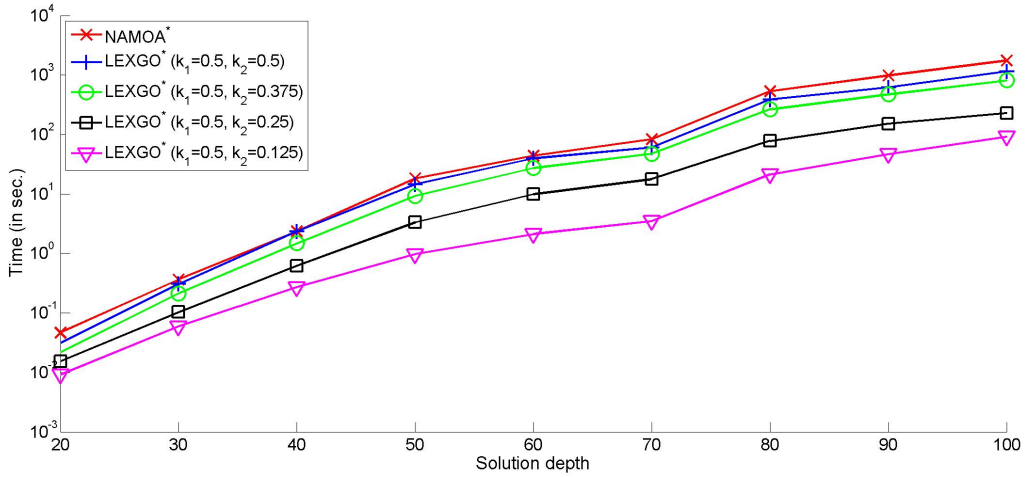


(b) $k_1 = 0.5$

Figure 6.5. Class II experiments on grids, average runtime (in seconds) per solution depth for LEXGO* and NAMOA* with lexicographic selection order.



(a) $k_1 = 0.75$



(b) $k_1 = 0.5$

Figure 6.6. Class II experiments on grids, average runtime (in seconds) per solution depth for LEXGO* and NAMOA* with linear aggregation selection order.

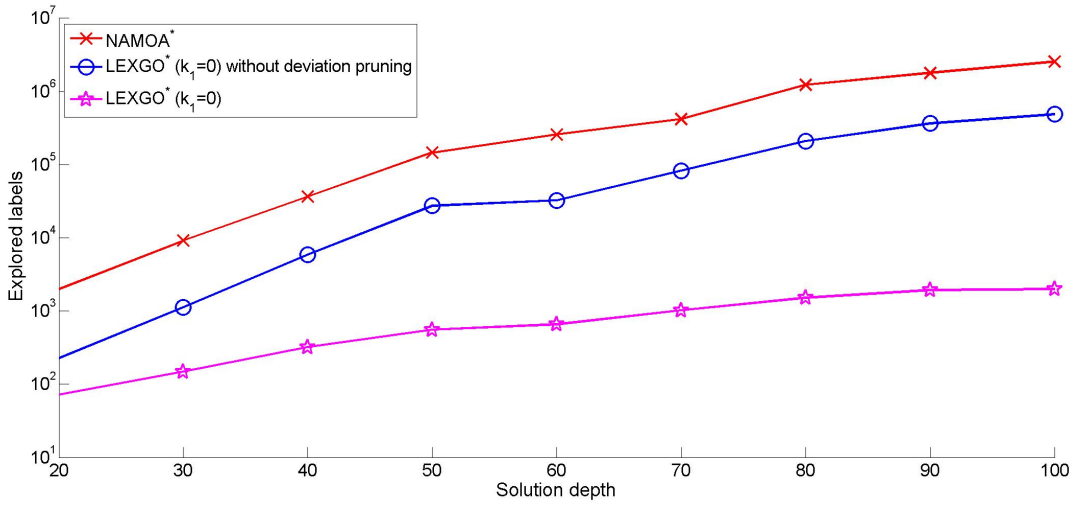


Figure 6.7. Class I experiments on grids, average explored labels per solution depth to LEXGO* ($k_1 = 0$) with and without deviation pruning.

Regarding the differences between the lexicographic and linear selection orders, we observed a better comparative performance of $\text{LEXGO}^*_{\text{lex}}$ when compared to $\text{NAMOA}^*_{\text{lex}}$ than $\text{LEXGO}^*_{\text{lin}}$ to $\text{NAMOA}^*_{\text{lin}}$. The linear order introduces a particular case when $k_1 = 0.75$ and $k_2 = 0.5625$, since its runtime performance is worse than $k_1 = 0.75$ and $k_2 = 0.75$. The former performs a smaller number of label expansions but a much higher number of deviation pruning comparisons than the latter.

6.2.3 Analysis on the pruning condition

Figures 6.7 and 6.8 compare average runtimes and scanned labels by LEXGO*, with lexicographic selection order, with and without deviation pruning (see Equation 4.4), respectively. Only the results with the lexicographic order are shown, since the results with the linear order are practically identical. These are results for the first set of experiments and $k_1 = 0$, where goals are not satisfied and deviation pruning is most effective. Values for NAMOA* are also displayed as reference. As soon as goals are satisfied, deviation pruning loses pruning power. For $k_1 = 0.25$ a smaller advantage is achieved. For larger values of k_1, deviation pruning does not offer practical advantage.

These results show where goals could be satisfied, i.e. $k_1 = \{0.5, 0.75, 1\}$, deviation pruning does not improve performance in practice. However, for those values of k_1, i.e. $k_1 = \{0, 0.25\}$, where goals cannot be satisfied, deviation pruning can make a difference, specially for $k_1 = 0$. Figure 6.8 shows up to three orders of magnitude of improvement in runtime, that can be attributed to a reduction of two orders of magnitude in scanned labels (see Figure 6.7). This can be explained by the fact that when goals are satisfied, values of deviation vectors of expanded labels are $\vec{0}$ and deviation pruning is barely triggered. On the other hand, unsatisfied goals cause greater deviation values and hence, a greater number of pruning opportunities. Therefore, the higher deviation from goals, the more effective deviation pruning.

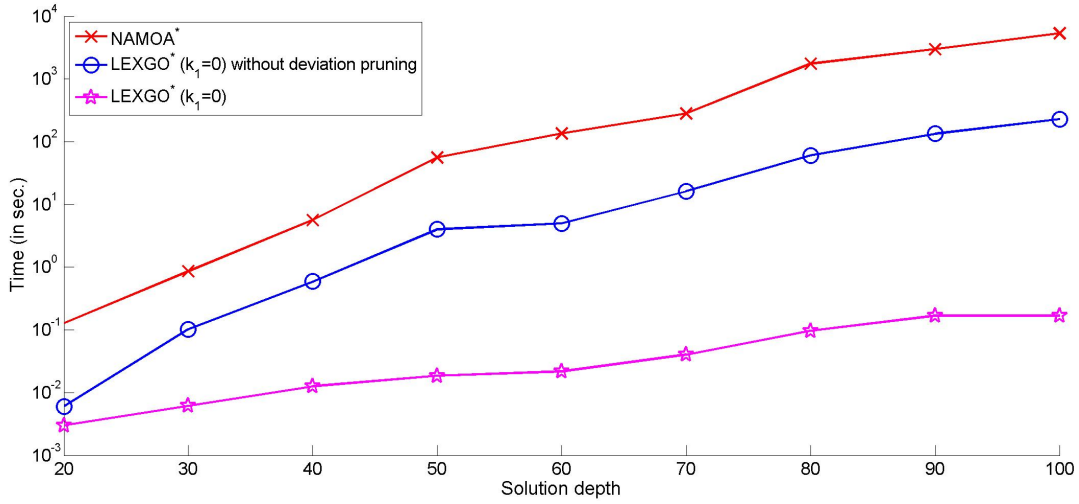


Figure 6.8. Class I experiments on grids, average runtimes in seconds per solution depth to LEXGO* ($k_1 = 0$) with and without deviation pruning.

6.2.4 Summary

We present a deeper study than the experiments already presented by the author in Pulido et al. (2014). We have also added to the experimental set the linear aggregation order to both algorithms, NAMOA* and LEXGO*, as well as $k_1 = 0.75$ for class II experiments. NAMOA$^*_{\text{lin}}$ outperforms NAMOA$^*_{\text{lex}}$ in all cases, by approximately a factor of two. Several studies have reported the same results for NAMOA* on the lexicographic and linear selection orders.

Regarding LEXGO$^*_{\text{lex}}$ and LEXGO$^*_{\text{lin}}$, the latter performs faster when the number of goal-optimal solution costs is bigger, i.e. when $k_1 = \{1, 0.75\}$, by an approximate factor of 1.8, due to the fact that linear version finds the solutions at a later stage and therefore, a smaller number of filtering comparisons is needed. However, LEXGO$^*_{\text{lex}}$ outperforms LEXGO$^*_{\text{lin}}$ when the number of goal-optimal solution vectors is smaller, i.e. when $k_1 = 0.5$. Moreover, LEXGO$^*_{\text{lex}}$ has a slightly better performance when goals cannot be satisfied, i.e. when $k_1 = \{0.25, 0\}$. In the latter case, the relative performance is problem-dependent.

The comparative performance between LEXGO$^*_{\text{lex}}$ vs NAMOA$^*_{\text{lex}}$, and LEXGO$^*_{\text{lin}}$ vs NAMOA$^*_{\text{lin}}$ turns out to be very similar in explored labels. However, the lexicographic order is comparatively more advantageous in runtime for LEXGO* than the linear, having a smaller time overhead and better comparative performance. Even so, both LEXGO* alternatives run faster than NAMOA* for $k_1 = 0.5$, they run several orders of magnitude faster for $k_1 = 0$ and around two orders of magnitude when $k_1 = 0.25$, regardless the selection order.

6.3 NAMOA$^*_{\text{dr}}$ vs NAMOA*

In this section, we analyze the runtime performance of two different versions of the NAMOA* algorithm: the standard version, and the newly introduced one. A descrip-

tion of these algorithms can be found in Sections 2.6.2 and 4.3, respectively. NAMOA* and NAMOA$^*_{\text{dr}}$ were presented in Section 2.6.2 and Section 4.3, respectively.

The experiments presented in this section analyze the impact of the dimensionality reduction technique on the sets of random grid problems previously used with $q = 3$. Extra problem sets with $q = 4$ and $q = 5$ objectives have been added to the experimental evaluation of these algorithms. The solution depths considered for these new experiments are $d = \{20, 30, 40, 50\}$ and $d = \{20, 30, 40\}$ for $q = 4$ and $q = 5$, respectively.

These two versions of NAMOA* differ in the order of selection of OPEN labels and in the way dominance is checked in filtering and cl-pruning operations. The first analyzed variants are NAMOA$^*_{\text{lex}}$ and NAMOA$^*_{\text{lin}}$, which use, to the best of our knowledge, the usual dominance pruning and filtering techniques in previously reported experimental evaluations of multiobjective search algorithms. The second algorithm analyzed, NAMOA$^*_{\text{dr}}$ uses a lexicographic order of selection and the t-discarding technique for filtering and cl-pruning, as described in Section 4.2.

6.3.1 Analysis

Table 6.6 shows the average size of relevant sets of labels for the execution of NAMOA$^*_{\text{dr}}$ on random grid problems. The first column (q) indicates the number of objectives. The second column (d) displays solution depth. The third column (Max OPEN) displays the maximum cardinality of the set of open labels. The fourth column, $\sum G_{cl}$, displays the total number of closed (permanent) labels at termination, calculated as the sum of the number of labels at the G_{cl} sets of all visited nodes. This is also the total number of labels expanded by the algorithm. The fifth column shows for comparison the sum of sizes of the corresponding sets of truncated labels, i.e. $\sum T(G_{cl})$. The sixth column displays the percentage ratio between columns four and five. For example, for $d = 20$, the average number of label expansions by NAMOA$^*_{\text{dr}}$ was 1,985, while the average number of labels in the truncated sets was only 476, this results in a percentage ratio of 23.98%.

Column 7 in Table 6.6 shows the average number of different non-dominated solution vectors in COSTS (C*). The eighth column displays the size of the corresponding truncated set T(C*). The last column displays the percentage ratio between columns six and seven.

Table 6.7 displays the average runtimes of the three considered alternatives of NAMOA* with $q \in \{3, 4, 5\}$ objectives for random grid problems. The last columns show the relative percentage improvement of NAMOA$^*_{\text{dr}}$ over NAMOA$^*_{\text{lex}}$ and NAMOA$^*_{\text{lin}}$, respectively.

All evaluated algorithms perform op-pruning in the same way. However, NAMOA$^*_{\text{dr}}$ uses a different technique for cl-pruning and filtering. Figure 6.9 displays some results of the execution of NAMOA$^*_{\text{dr}}$: the percentage of labels pruned by G_{op} (op-pruning), truncated closed node labels $T(G_{cl})$ (cl-pruning), and filtered by T(C*) over the total number of discarded labels. Results are displayed as a function of solution depth d, (a) for $q = 3$ objectives, (b) for $q = 4$ objectives, and (c) for $q = 5$ objectives.

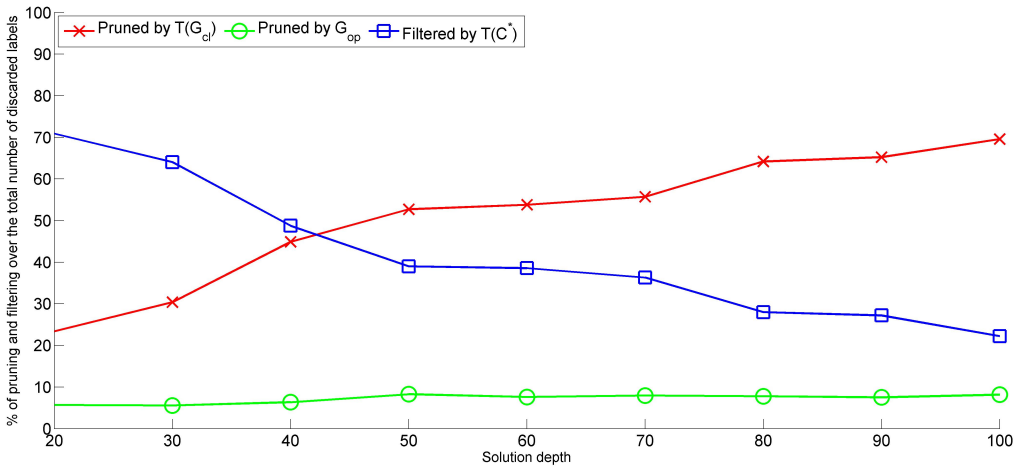
Figure 6.10 shows runtimes of NAMOA$^*_{\text{lex}}$, NAMOA$^*_{\text{lin}}$ and NAMOA$^*_{\text{dr}}$ in logarithmic scale against solution depth for $q = \{3, 4, 5\}$ objectives. The items in the legend

Table 6.6. Average size of relevant sets of labels for random grid problems solved by NAMOA$^*_{\text{dr}}$.

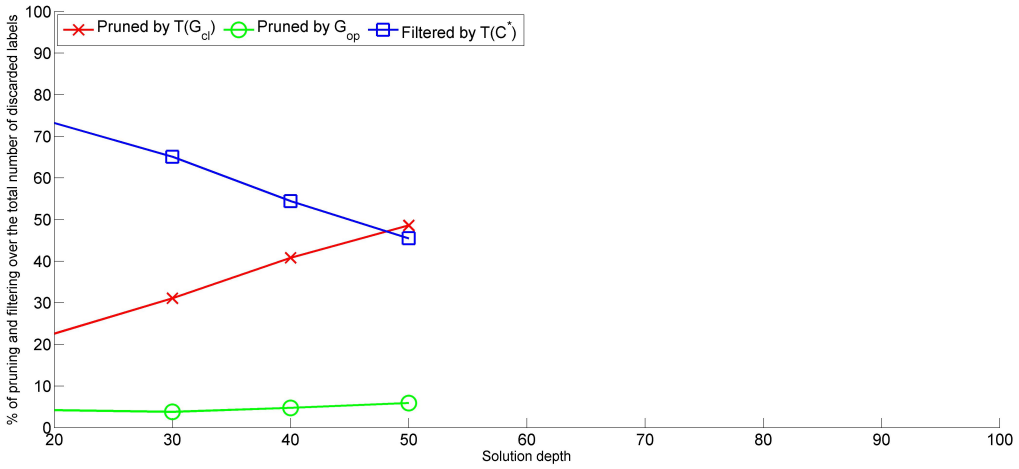
q	d	Max OPEN	$\sum G_{cl}$	$\sum T(G_{cl})$	%	C*	T(C*)	%
3	20	194	1,985	476	23.98	122	13	10.66
3	30	723	9,164	2,091	22.82	302	32	10.60
3	40	2,233	36,557	4,923	13.47	694	44	6.34
3	50	8,327	145,823	12,450	8.54	1,599	60	3.75
3	60	11,091	257,935	21,026	8.15	2,007	80	3.99
3	70	17,312	420,056	29,845	7.11	2,561	74	2.89
3	80	38,512	1,231,565	61,457	4.99	5,423	108	1.99
3	90	51,817	1,789,607	81,036	4.53	5,912	122	2.06
3	100	72,062	2,550,354	97,160	3.81	8,307	137	1.65
4	20	531	6,192	2,061	33.28	493	83	16.83
4	30	3,183	49,735	13,150	26.44	2,230	320	14.34
4	40	14,409	283,811	44,191	15.57	7,826	774	9.89
4	50	72,112	1,542,793	153,639	9.95	24,942	1,382	5.54
5	20	1,127	15,681	6,539	41.70	1,819	522	28.69
5	30	10,019	172,238	51,145	29.69	10,830	1,917	17.70
5	40	62,280	1,371,885	319,333	23.27	49,634	8,320	16.76

Table 6.7. Average runtimes in seconds for random grid problems.

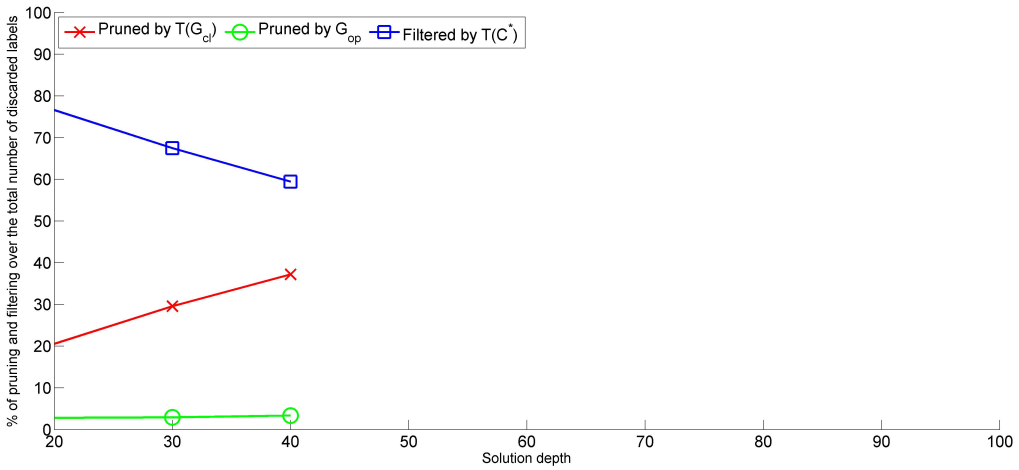
q	d	NAMOA$^*_{\text{lex}}$	σ_{lex}	NAMOA$^*_{\text{lin}}$	σ_{lin}	NAMOA$^*_{\text{dr}}$	σ_{dr}	$(\frac{dr}{lex})\%$	$(\frac{dr}{lin})\%$
3	20	0.06	0.02	0.04	0.01	0.0622	0.01	99.67	132.91
3	30	0.46	0.26	0.36	0.18	0.293	0.11	62.74	80.23
3	40	3.53	1.21	2.36	0.74	1.32	0.30	37.39	56.13
3	50	36.83	15.56	18.37	5.67	6.87	2.02	18.65	37.41
3	60	86.93	49.57	43.83	22.94	11.94	4.48	13.73	27.26
3	70	178.53	106.75	83.37	47.77	20.93	7.69	11.72	25.11
3	80	1,164.11	293.72	533.79	135.96	76.01	14.22	6.52	14.24
3	90	2,030.06	730.05	981.32	313.79	120.66	33.19	5.94	12.30
3	100	3,662.93	1,100.91	1,754.43	583.28	196.13	52.74	5.35	11.18
4	20	0.39	0.14	0.30	0.11	0.23	0.07	58.52	76.66
4	30	11.64	4.22	8.88	3.87	4.31	1.97	37.02	48.53
4	40	292.17	188.27	166.36	105.31	50.41	29.47	17.25	30.30
4	50	5,604.29	2,356.35	3,173.81	1,171.11	645.37	195.84	11.51	20.33
5	20	4.01	1.51	2.99	1.14	2.20	0.76	54.86	73.57
5	30	204.69	135.77	114.91	73.32	57.89	34.61	28.28	50.37
5	40	10,848.24	7,404.86	6,141.63	4,212.81	1,919.56	1,269.98	17.69	31.25



(a) $q = 3$



(b) $q = 4$



(c) $q = 5$

Figure 6.9. Percentage of pruned and filtered labels over the total number of discarded labels by NAMOA$^*_{\text{dr}}$ per solution depth for $q \in \{3, 4, 5\}$ objectives in grid problems.

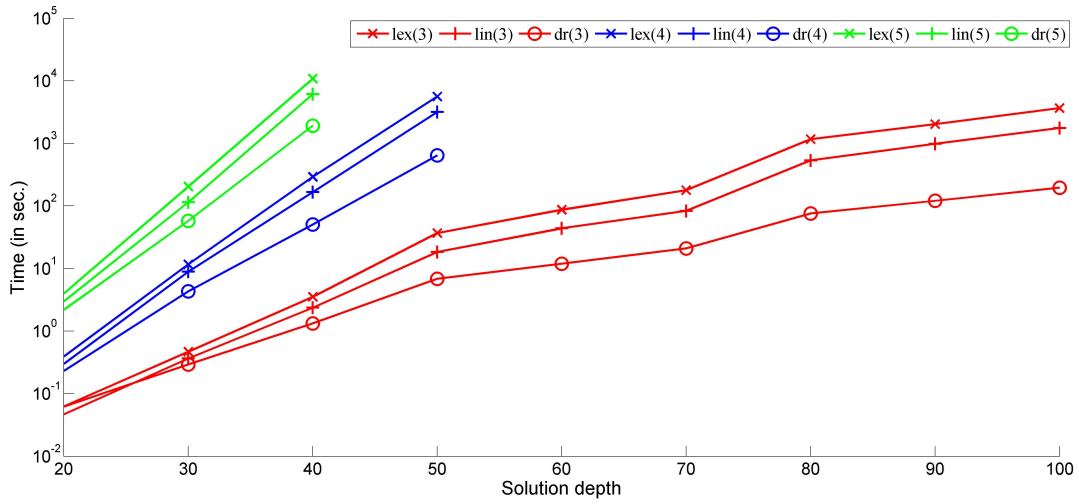


Figure 6.10. Average runtimes for $q \in \{3, 4, 5\}$ objectives per solution depth in grid problems.

indicate the version of the algorithm and the objectives, e.g. dr(4) indicates NAMOA$^*_{dr}$ with $q = 4$. Finally, Figure 6.11 displays the percentage of runtimes of NAMOA$^*_{dr}$ and NAMOA$^*_{lin}$ over NAMOA$^*_{lex}$ for $q = 3$ experiments displayed as a function of solution depth.

6.3.2 Summary

A linear aggregation selection function is consistently more efficient in runtime than a lexicographic function when both are applied to standard NAMOA*. However, the lexicographic order can be exploited by the t-discarding technique for filtering and cl-pruning (NAMOA$^*_{dr}$). Results over grid problems reveal a dramatic improvement in runtime performance of over an order of magnitude for three-objective problems (see Figure 6.10). The speedup of NAMOA$^*_{dr}$ over NAMOA$^*_{lex}$ and NAMOA$^*_{lin}$ even grows with problem difficulty (see Table 6.7 and Figure 6.11), reducing time requirements over 90% for the harder $q = 3$ problems. When more objectives are considered, $q = \{4, 5\}$, similar results can be observed, although a smaller number of experiments can be presented due to the increasing computational difficulty.

As expected, multiobjective label-setting search spends most of the time performing dominance checks between labels. Every new label has to be checked for op-pruning, cl-pruning, and filtering. Figure 6.9 shows that, for the harder grid problems, cl-pruning is the operation that tends to discard most labels for deeper solutions. The same tendency can be observed regardless the number of objectives.

While the ratio of labels discarded by filtering decreases with problem difficulty, it was always larger than the ratio of those discarded by op-pruning in our grid experiments. This is important for the efficiency of NAMOA$^*_{dr}$, since cl-pruning and filtering can both benefit from t-discarding. Table 6.6 reveals that the truncated sets of labels used by t-discarding are significantly smaller than the original ones, and their relative size even decreases with problem difficulty. For example, with solution depth $d = 100$

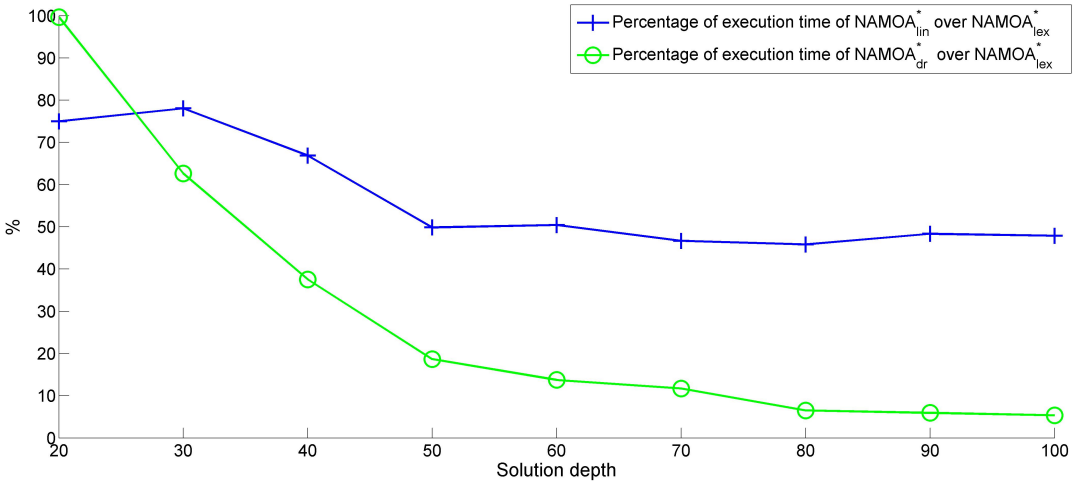


Figure 6.11. Percentage of average runtime of $\text{NAMOA}^*_{\text{dr}}$ and $\text{NAMOA}^*_{\text{lin}}$ over $\text{NAMOA}^*_{\text{lex}}$ for $q = 3$ grid problems.

and $q = 3$, a label is checked against a set of 8,307 labels for filtering in the worst case with the standard procedure, while with t-discarding the worst case involves only a set of 137 labels (or 1.65%).

6.4 $\text{LEXGO}^*_{\text{dr}}$ vs LEXGO^*

In this section we examine the performance of the dimensionality reduction technique applied to LEXGO^*, called $\text{LEXGO}^*_{\text{dr}}$. A detailed description of LEXGO^* and $\text{LEXGO}^*_{\text{dr}}$ are presented in Section 4.1 and Section 4.4, respectively. The experiments presented below are applied to the random grid problems with three objectives and goals defined in Section 6.1.

6.4.1 Analysis on class I experiments

Table 6.8 displays average runtimes for $\text{LEXGO}^*_{\text{lex}}$, $\text{LEXGO}^*_{\text{lin}}$, and $\text{LEXGO}^*_{\text{dr}}$ for depth $d = 100$. The last column shows the speed-up obtained by $\text{LEXGO}^*_{\text{dr}}$ over the best version of standard LEXGO^* for each value of k_1. $\text{LEXGO}^*_{\text{dr}}$ achieves important reductions in runtime when goals can be satisfied, i.e. when $k_1 = \{1, 0.75, 0.5\}$ and it barely affects runtimes when goals cannot be satisfied, i.e. $k_1 = \{0.25, 0\}$. Figure 6.12 shows graphically the relative runtime performance of $\text{LEXGO}^*_{\text{dr}}$ over the best previous runtimes of LEXGO^*, as a function of solution depth. Values corresponding to $k_1 = 0$ are not shown since they are practically zero.

6.4.2 Analysis on class II experiments

Table 6.9 shows the average runtimes of $\text{LEXGO}^*_{\text{dr}}$ for all possible values of k_2 with $k_1 = 0.75$ and $k_1 = 0.5$. Figures 6.13(a) and 6.13(b) display, as a function of solution depth, the relative runtime performance of $\text{LEXGO}^*_{\text{dr}}$ over the best previous runtimes of LEXGO^* in $k_1 = 0.75$ and $k_1 = 0.5$, respectively.

Table 6.8. Class I experiments on grids, average runtimes in seconds of LEXGO$^*_{\text{lex}}$, LEXGO$^*_{\text{lin}}$ and LEXGO$^*_{\text{dr}}$ for $d = 100$. Speed-up of LEXGO$^*_{\text{dr}}$ over the best standard version of LEXGO*.

k_1	LEXGO$^*_{\text{lex}}$	LEXGO$^*_{\text{lin}}$	LEXGO$^*_{\text{dr}}$	Speedup
1	3,763.78	2,114.18	254.40	8.31
0.75	3,381.33	1,847.88	300.86	6.14
0.5	819.28	1,145.15	282.49	2.90
0.25	33.47	37.45	35.37	0.94
0	0.07	0.09	0.10	0.77

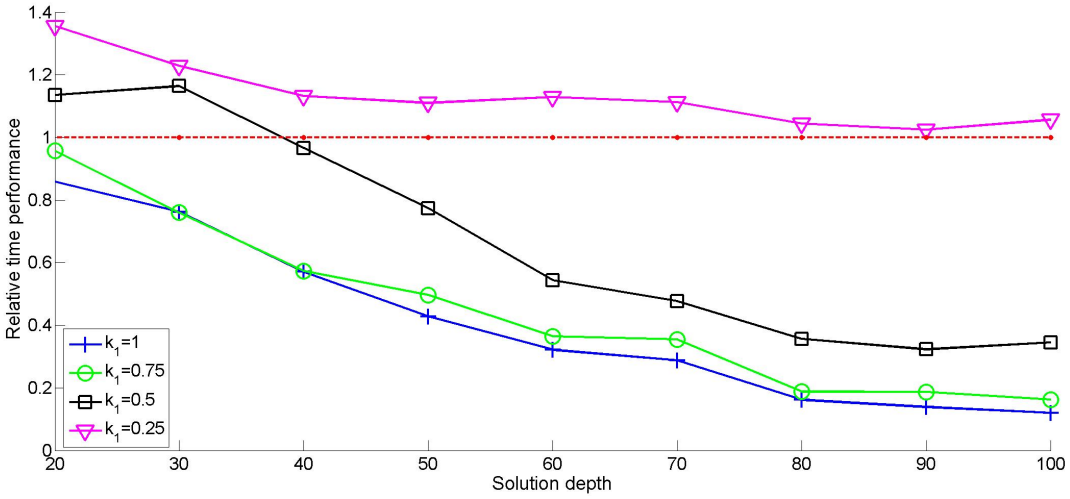


Figure 6.12. Class I experiments on grids, relative runtime performance of LEXGO$^*_{\text{dr}}$ over the best runtimes of standard LEXGO*.

Table 6.9. Class II experiments on grids, $\text{LEXGO}^*_{\text{dr}}$ runtimes in seconds.

	$\text{LEXGO}^*_{\text{dr}}$								
	0.75				0.5				k_1
d	0.75	0.5625	0.375	0.1875	0.5	0.375	0.25	0.125	k_2
20	0.07	0.06	0.04	0.01	0.03	0.02	0.01	<0.01	
30	0.38	0.35	0.23	0.09	0.22	0.15	0.08	0.05	
40	1.7	1.6	1.1	0.38	1.1	0.81	0.42	0.24	
50	10.3	10.6	6.5	1.7	6.5	4.2	1.9	0.84	
60	17.3	19.3	13.5	3.6	14.4	10.3	4.8	1.8	
70	30.8	32.2	22.8	6.8	24.0	17.7	8.5	2.9	
80	105.9	133.1	95.0	24.0	97.8	68.5	29.4	17.4	
90	187.9	219.8	156.5	41.2	158.7	117.8	53.3	40.0	
100	300.8	389.6	283.7	63.4	282.4	199.1	84.7	82.1	

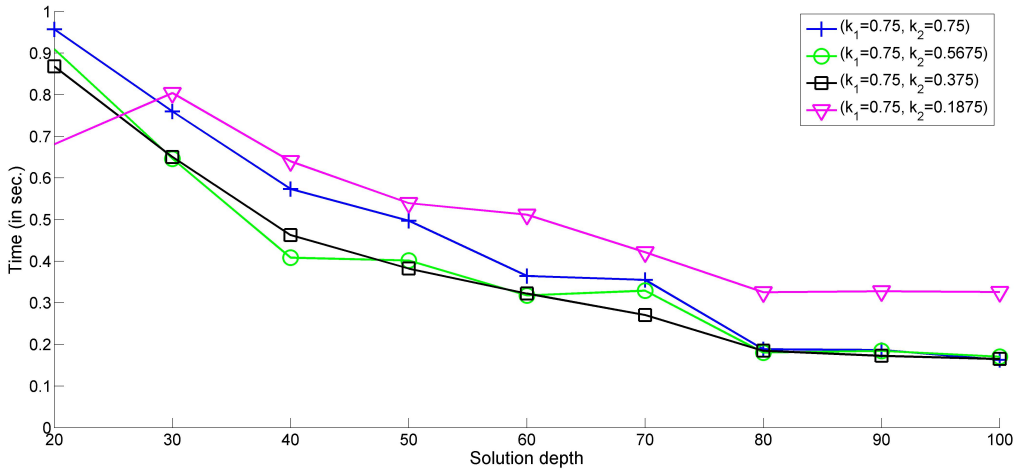
Just as in the case of class I experiments, $\text{LEXGO}^*_{\text{dr}}$ obtains a better efficiency when the number of goal-optimal solution costs is greater. Thus, the relative runtime for $k_1 = 0.75$ is around 20% and for $k_1 = 0.5$ around 30% to 70% of $\text{LEXGO}^*_{\text{lin}}$, the fastest version of the LEXGO* algorithm so far. The sole case when $\text{LEXGO}^*_{\text{dr}}$ does not show improvement in practice is when $k_1 = 0.5$ and $k_2 = 0.125$. We can observe in Table 6.3 that in this case the goals are rarely satisfied by any problem instance.

6.4.3 Summary

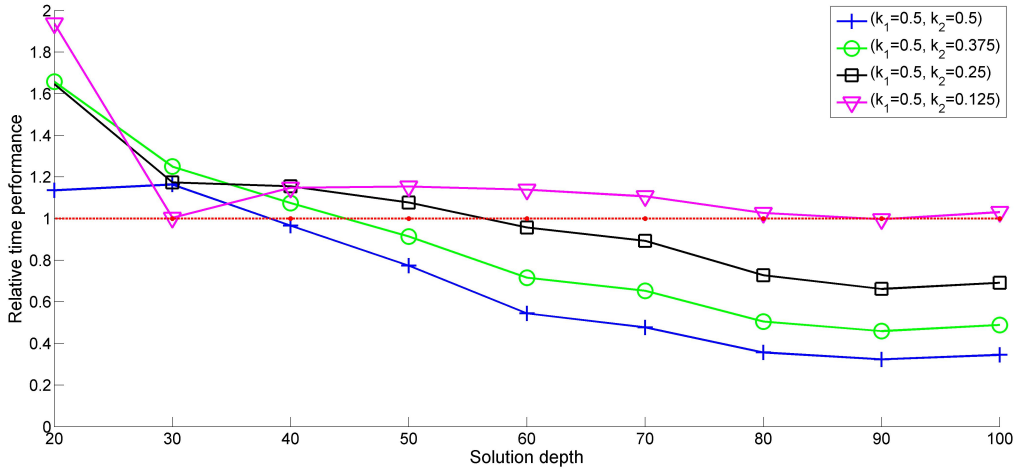
Just as in the case of NAMOA*, LEXGO* can also be improved when applying the t-discarding method. This method can be applied to LEXGO* when goals can be satisfied, i.e. when deviation vectors are $\vec{0}$.

In a similar way as $\text{NAMOA}^*_{\text{dr}}$ performs against NAMOA* (see Section 6.3), the speed-up of $\text{LEXGO}^*_{\text{dr}}$ over LEXGO* grows with problem difficulty (see Figures 6.12, and 6.13), reducing time requirements over 88% for the harder problems. In addition, $\text{LEXGO}^*_{\text{dr}}$ speed-up grows with the number of goal-optimal solution costs. Notice in Table 6.8 that $\text{LEXGO}^*_{\text{dr}}$ for $d = 100$ runs 11% faster with $k_1 = 1$ than with $k_1 = 0.5$, despite of the fact that the number of label expansions is 40% greater for $k_1 = 1$ than for $k_1 = 0.5$ (see Tables 6.2(a) and 6.2(b)). This is because the number of opportunities to apply t-discarding grows with the relative percentage of goal-optimal solution costs over the Pareto set. Hence, t-discarding is triggered on a more regular basis on these cases where k_1 is greater.

$\text{LEXGO}^*_{\text{dr}}$ does not show any improvement when goals cannot be satisfied, since the t-discarding can be barely applied to these problems. However, LEXGO* achieves important reductions in time when goals can be satisfied, being three to eight times faster in class I experiments and up to six times faster in the second class of experiments.



(a) $k_1 = 0.75$



(b) $k_1 = 0.5$

Figure 6.13. Class II experiments on grids, relative runtime performance (in seconds) of LEXGO$^*_{\text{dr}}$ over the best previous runtimes of LEXGO* as a function of solution depth.

Table 6.10. Class I experiments on grids, runtimes of $\text{LEXGO}^*_{\text{dr}}$ and $\text{NAMOA}^*_{\text{dr}}$ for $d = 100$.

	$\text{LEXGO}^*_{\text{dr}}$				
$\text{NAMOA}^*_{\text{dr}}$ Runtime (s)	$(k_1 = 1)$	$(k_1 = 0.75)$	$(k_1 = 0.5)$	$(k_1 = 0.25)$	$(k_1 = 0)$
196.1	254.4	300.8	282.5	35.3	0.1

6.5 $\text{LEXGO}^*_{\text{dr}}$ vs $\text{NAMOA}^*_{\text{dr}}$

This section deals with the algorithms that employ the dimensionality reduction technique, $\text{NAMOA}^*_{\text{dr}}$ and $\text{LEXGO}^*_{\text{dr}}$. We have already seen the improvement of $\text{NAMOA}^*_{\text{dr}}$ over NAMOA^*, as well as $\text{LEXGO}^*_{\text{dr}}$ over LEXGO^*. In the following, a final time performance comparison between the two best alternatives is presented.

The experiments included in this section are applied to random grid problems with three objectives and goals described in Section 6.1.

6.5.1 Analysis on class I experiments

Table 6.10 shows the relative performance of $\text{LEXGO}^*_{\text{dr}}$ over $\text{NAMOA}^*_{\text{dr}}$ for the solution depth equal to 100. $\text{LEXGO}^*_{\text{dr}}$ performs better than $\text{NAMOA}^*_{\text{dr}}$ when goals cannot be satisfied, i.e. $k_1 = \{0.25, 0\}$ but it is significantly slower when goals can be satisfied. It is also worth to pay attention to the relative performance when $k_1 = 1$ in comparison with $k_1 = 0.5$. The former performs around 40% more label expansions (see Table 6.2) and it is still faster than the latter. Notice that the number of scanned labels of both algorithms that employ the dimensionality reduction technique are equal to their versions without t-discarding, although the number of dominance checks is greatly reduced in $\text{NAMOA}^*_{\text{dr}}$ and $\text{LEXGO}^*_{\text{dr}}$.

Figure 6.14 displays the average runtimes of $\text{NAMOA}^*_{\text{dr}}$ and $\text{LEXGO}^*_{\text{dr}}$ as a function of solution depth. The trend here is slightly in favor of $\text{NAMOA}^*_{\text{dr}}$ over $\text{LEXGO}^*_{\text{dr}}$ for $k_1 \geq 0.5$.

6.5.2 Analysis on class II experiments

Regarding the second class of experiments, Table 6.11 shows $\text{NAMOA}^*_{\text{dr}}$ and $\text{LEXGO}^*_{\text{dr}}$ runtimes for $k_1 = 0.75$ and $k_1 = 0.5$ with all k_2 possible values. Those cases where $\text{LEXGO}^*_{\text{dr}}$ outperforms $\text{NAMOA}^*_{\text{dr}}$ are shown in bold. Figures 6.15(a) and 6.15(b) display the runtimes as a function of solution depth for $k_1 = 0.75$ and $k_1 = 0.5$, respectively.

In these experiments, it can be observed that there are cases when goals are satisfiable and $\text{LEXGO}^*_{\text{dr}}$ is still faster than $\text{NAMOA}^*_{\text{dr}}$. On one hand, when $k_1 = 0.75, k_2 = 0.1875$ approximately 10% of the Pareto frontier is returned (see Table 6.3) and $\text{LEXGO}^*_{\text{dr}}$ is somewhat three times faster than $\text{NAMOA}^*_{\text{dr}}$. On the other hand, when $k_1 = 0.5$ and $k_2 = 0.25$ goals can also be satisfied and $\text{LEXGO}^*_{\text{dr}}$ outperforms $\text{NAMOA}^*_{\text{dr}}$.

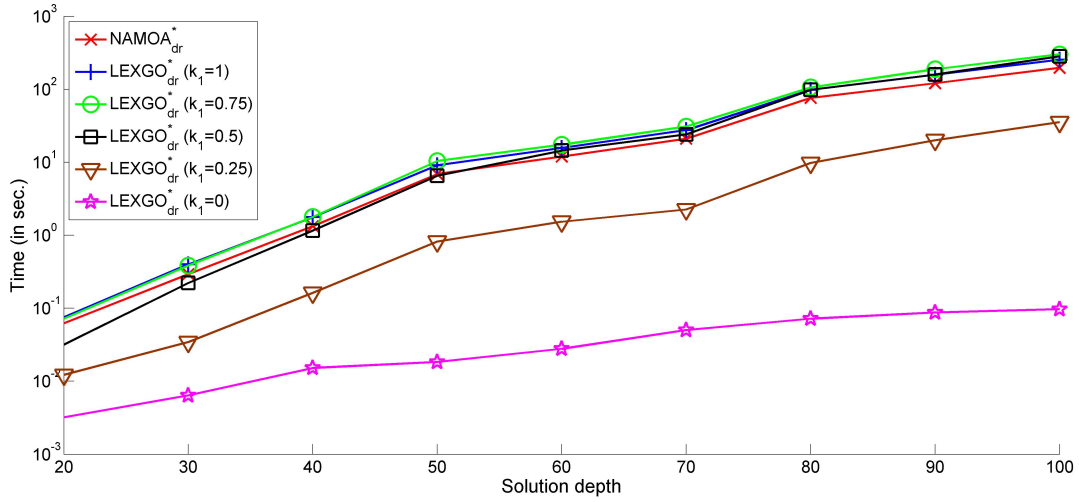
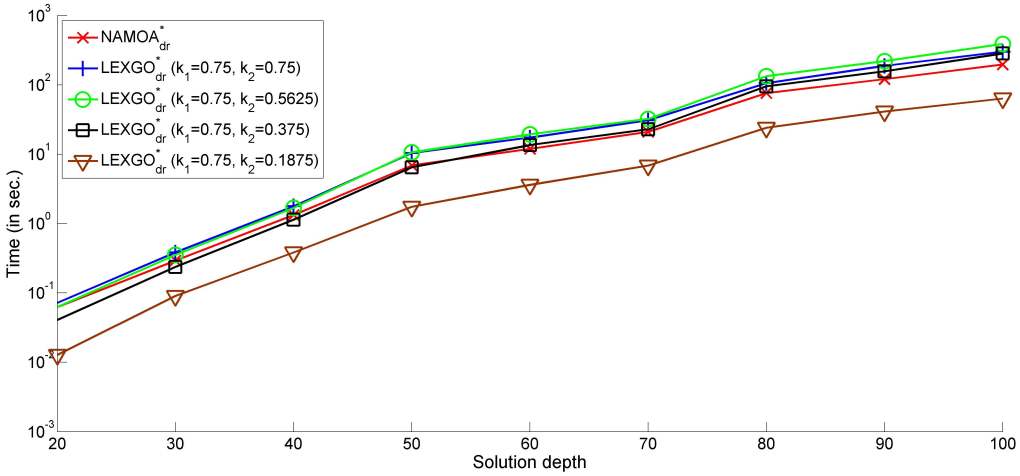


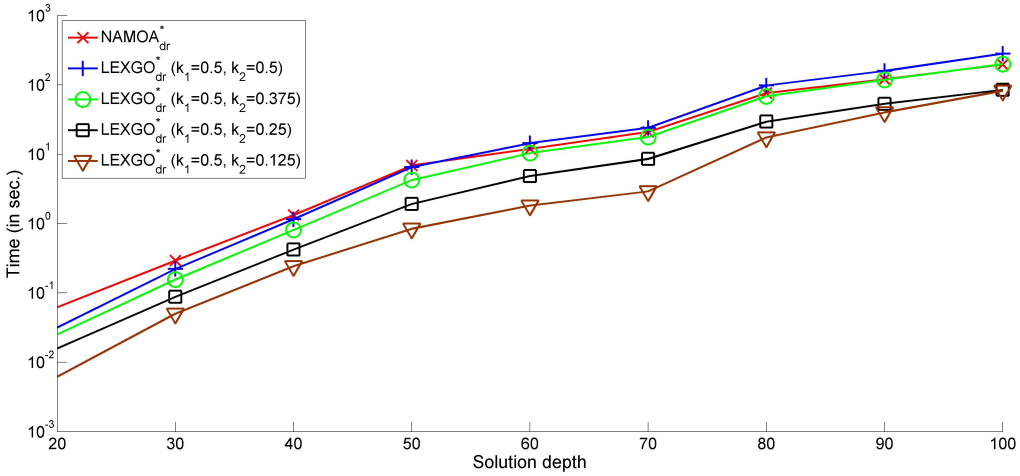
Figure 6.14. Class I experiments on grids, average runtimes (in seconds) of NAMOA*dr and LEXGO*dr per solution depth.

Table 6.11. Class II experiments on grids, LEXGO*dr and NAMOA*dr runtimes (in seconds). Cases were LEXGO*dr outperforms NAMOA*dr are highlighted in bold.

		LEXGO*dr								
	NAMOA*dr	0.75				0.5				k_1
d	Runtime (s)	0.75	0.5625	0.375	0.1875	0.5	0.375	0.25	0.125	k_2
20	0.06	0.07	**0.06**	**0.04**	**0.01**	**0.03**	**0.02**	**0.01**	**<0.01**	
30	0.29	0.38	0.35	**0.23**	**0.09**	**0.22**	**0.15**	**0.08**	**0.05**	
40	1.3	1.7	1.6	**1.1**	**0.38**	**1.1**	**0.81**	**0.42**	**0.24**	
50	6.8	10.3	10.6	**6.5**	**1.74**	**6.5**	**4.2**	**1.9**	**0.84**	
60	11.9	17.3	19.3	13.5	**3.6**	14.4	**10.4**	**4.8**	**1.8**	
70	20.9	30.8	32.2	22.8	**6.8**	24.0	**17.7**	**8.5**	**2.9**	
80	76.0	105.9	133.1	95.0	**24.0**	97.8	**68.5**	**29.4**	**17.4**	
90	120.6	187.9	219.8	156.5	**41.2**	158.7	**117.8**	**53.3**	**40.0**	
100	196.1	300.8	389.6	283.7	**63.4**	282.4	199.1	**84.7**	**82.1**	



(a) $k_1 = 0.75$



(b) $k_1 = 0.5$

Figure 6.15. Class II experiments on grids, average runtimes in seconds of $\text{LEXGO}^*_{\text{dr}}$ and $\text{NAMOA}^*_{\text{dr}}$ per solution depth.

6.5.3 Summary

The experimental comparison between NAMOA$^*_{\text{dr}}$ and LEXGO$^*_{\text{dr}}$ is one of the contributions of this thesis. On one hand, LEXGO$^*_{\text{dr}}$ is the application of the successful technique of t-discarding dominated paths to the algorithm LEXGO*, which has also been proposed in this thesis. In fact, these results are new to the literature and expose a significant different behavior than the comparison between LEXGO* and NAMOA* showed in Section 6.2. Despite LEXGO* has been proved to be more efficient than NAMOA* (see Section 5.2.1), the same cannot be asserted when the t-discarding technique is applied to both algorithms.

Both NAMOA$^*_{\text{dr}}$ and LEXGO$^*_{\text{dr}}$ achieve a speed-up of about an order of magnitude for our random grid problem experiments over NAMOA* and LEXGO*, respectively. However, LEXGO$^*_{\text{dr}}$ cannot apply the t-discarding method when a path does not satisfy goals and therefore, the applicability of this technique is partial, even though practically is applied to an average of 99.5% of the pruning operations. This small percentage of iterations that must perform the classical dominance checks clearly affect the runtime performance of LEXGO$^*_{\text{dr}}$ in comparison with NAMOA$^*_{\text{dr}}$. Thus, NAMOA$^*_{\text{dr}}$ becomes the option to be selected when goals can be satisfied while LEXGO$^*_{\text{dr}}$ is still preferred when goals cannot be satisfied and the path which minimizes the deviation from goals has to be returned.

6.6 Summary on random grid experiments

Previous results to the experiments showed in this thesis were reported in Pulido et al. (2014) and Pulido et al. (2015). The experiments reported in this chapter go beyond those previously reported and add LEXGO$^*_{\text{lin}}$ and LEXGO$^*_{\text{dr}}$ algorithms to the analyses. Furthermore, additional cases of LEXGO* have been evaluated, for instance, $k_1 = 0.75$ in class II experiments.

A summary of the runtime performance obtained from the evaluation of the algorithms NAMOA$^*_{\text{lex}}$, NAMOA$^*_{\text{lin}}$, LEXGO$^*_{\text{lex}}$, LEXGO$^*_{\text{lin}}$, NAMOA$^*_{\text{dr}}$ and LEXGO$^*_{\text{dr}}$ is presented in Table 6.12. A detailed analysis for class I and II experiments follows.

6.6.1 Summary on class I experiments

Table 6.13 shows all runtimes for class I experiments and all algorithms evaluated. NAMOA$^*_{\text{dr}}$ runtimes and those experiments of LEXGO* which perform faster than NAMOA$^*_{\text{dr}}$ are highlighted in bold. Thus, the empirical evaluation of all considered alternatives draws a clear picture on the performance of each algorithm, resulting generally in NAMOA$^*_{\text{dr}}$ as the best algorithm when goals can be satisfied and LEXGO$^*_{\text{dr}}$ when goals cannot be satisfied. Moreover, the t-discarding technique improves over an order of magnitude both algorithms NAMOA$^*_{\text{dr}}$ and LEXGO$^*_{\text{dr}}$ over NAMOA* and LEXGO*, and does not affect significantly the runtime of LEXGO* when goals cannot be satisfied.

Table 6.12. Runtime comparison - summary table for random grid experiments.

Comparison	Results
NAMOA$^*_{\text{lin}}$ vs NAMOA$^*_{\text{lex}}$	NAMOA$^*_{\text{lin}}$ outperforms NAMOA$^*_{\text{lex}}$ by a factor of two in all cases. Its comparative advantage slightly grows with problem difficulty.
LEXGO$^*_{\text{lin}}$ vs LEXGO$^*_{\text{lex}}$	LEXGO$^*_{\text{lin}}$ outperforms LEXGO$^*_{\text{lex}}$ by a factor of 1.8 when a high percentage of the non-dominated cost vectors satisfy the goals. When this percentage is smaller LEXGO$^*_{\text{lex}}$ outperforms LEXGO$^*_{\text{lin}}$. Both have similar performance when goals cannot be satisfied.
LEXGO* vs NAMOA*	The lexicographic order is comparatively more advantageous in runtime for LEXGO* than the linear. With both selection orders, LEXGO* runs faster than NAMOA* for $k_1 = 0.5$, around two orders of magnitude faster when $k_1 = 0.25$, and several orders of magnitude faster for $k_1 = 0$. A small time overhead can be observed for LEXGO* with $k_1 = 1$ and in some cases with $k_1 = 0.75$.
NAMOA$^*_{\text{dr}}$ vs NAMOA*	NAMOA$^*_{\text{dr}}$ is consistently faster than NAMOA$^*_{\text{lex}}$ and NAMOA$^*_{\text{lin}}$. Its advantage in runtime clearly grows with problem difficulty.
LEXGO$^*_{\text{dr}}$ vs LEXGO*	Their performance is similar when goals cannot be satisfied. However, LEXGO$^*_{\text{dr}}$ achieves important reductions in runtime when goals can be satisfied. LEXGO$^*_{\text{dr}}$ runtime advantage grows with the number of goal-optimal solution costs and with problem difficulty.
LEXGO$^*_{\text{dr}}$ vs NAMOA$^*_{\text{dr}}$	LEXGO$^*_{\text{dr}}$ is faster than NAMOA$^*_{\text{dr}}$ when goals cannot be satisfied. NAMOA$^*_{\text{dr}}$ performs faster when goals can be satisfied.

Table 6.13. Class I experiments on grids, runtimes (in seconds) of all algorithms studied in this thesis as a function of solution depth.

Depth	20	30	40	50	60	70	80	90	100
$\text{NAMOA}^*_{\text{lex}}$	0.06	0.47	3.53	36.83	86.94	178.53	1,164.12	2,030.07	3,662.93
$\text{NAMOA}^*_{\text{lin}}$	0.06	0.37	2.36	18.37	43.84	83.38	533.80	981.32	1,754.44
$\text{NAMOA}^*_{\text{dr}}$	**0.06**	**0.29**	**1.33**	**6.87**	**11.95**	**20.94**	**76.01**	**120.67**	**196.14**
$\text{LEXGO}^*_{\text{lex}}$									
$(k_1 = 1)$	0.07	0.52	3.70	37.35	88.89	179.67	1,202.71	2,115.23	3,763.78
$(k_1 = 0.75)$	0.07	0.46	3.23	31.20	78.87	157.80	1,074.36	1,781.12	3,381.33
$(k_1 = 0.5)$	**0.03**	**0.19**	**1.20**	8.44	26.54	50.37	274.42	490.95	819.28
$(k_1 = 0.25)$	**0.01**	**0.03**	**0.14**	**0.74**	**1.36**	**2.02**	**9.34**	**19.55**	**33.47**
$(k_1 = 0)$	**<0.01**	**<0.01**	**0.01**	**0.02**	**0.02**	**0.03**	**0.05**	**0.07**	**0.07**
$\text{LEXGO}^*_{\text{lin}}$									
$(k_1 = 1)$	0.09	0.52	3.10	21.18	48.77	95.36	606.91	1,141.97	2,114.18
$(k_1 = 0.75)$	0.07	0.51	3.10	20.88	47.68	87.01	563.05	1,006.66	1,847.88
$(k_1 = 0.5)$	**0.03**	**0.31**	2.36	14.55	39.54	60.51	387.28	620.69	1,145.15
$(k_1 = 0.25)$	**0.01**	**0.03**	**0.18**	**0.92**	**1.70**	**2.55**	**11.04**	**21.58**	**37.45**
$(k_1 = 0)$	**<0.01**	**0.01**	**0.01**	**0.02**	**0.02**	**0.04**	**0.07**	**0.09**	**0.09**
$\text{LEXGO}^*_{\text{dr}}$									
$(k_1 = 1)$	0.08	0.40	1.77	9.08	15.69	27.46	98.24	158.32	254.40
$(k_1 = 0.75)$	0.07	0.38	1.78	10.38	17.38	30.89	105.95	187.93	300.86
$(k_1 = 0.5)$	**0.03**	**0.22**	**1.15**	**6.53**	14.44	24.03	97.85	158.71	282.49
$(k_1 = 0.25)$	**0.01**	**0.03**	**0.16**	**0.82**	**1.53**	**2.25**	**9.76**	**20.03**	**35.37**
$(k_1 = 0)$	**<0.01**	**0.01**	**0.02**	**0.02**	**0.03**	**0.05**	**0.07**	**0.09**	**0.10**

6.6.2 Summary on class II experiments

With respect to class II experiments, Table 6.14 shows all runtimes for class II experiments and all algorithms evaluated. $\text{NAMOA}^*_{\text{dr}}$ runtimes and those experiments of LEXGO^* which perform faster than $\text{NAMOA}^*_{\text{dr}}$ are also highlighted in bold.

There are eight different cases for each solution depth with the combinations of $k_1 = \{0.75, 0.5\}$ and k_2 calculated as described in Equation 3.2. It can be observed that the number of cases where $\text{LEXGO}^*_{\text{dr}}$ outperforms $\text{NAMOA}^*_{\text{dr}}$ decreases with solution depth. For instance, the number of cases where $\text{LEXGO}^*_{\text{dr}}$ is faster than $\text{NAMOA}^*_{\text{dr}}$ decreases from 7 out of 8 when $d = 20$ to 3 out of 8 when $d = 100$. It is expected that $\text{NAMOA}^*_{\text{dr}}$ will outperform $\text{LEXGO}^*_{\text{dr}}$ in runtime performance in more difficult problems.

Table 6.14. Class II experiments on grids, runtimes (in seconds) of all algorithms studied in this thesis as a function of solution depth.

Depth	20	30	40	50	60	70	80	90	100
$NAMOA^*_{lex}$	0.06	0.47	3.53	36.83	86.94	178.53	1,164.12	2,030.07	3,662.93
$NAMOA^*_{lin}$	0.06	0.37	2.36	18.37	43.84	83.38	533.80	981.32	1,754.44
$NAMOA^*_{dr}$	**0.06**	**0.29**	**1.33**	**6.87**	**11.95**	**20.94**	**76.01**	**120.67**	**196.14**
$LEXGO^*_{lex}$									
(0.75, 0.75)	0.07	0.46	3.23	31.20	78.87	157.80	1,074.36	1,781.12	3,381.33
(0.75, 0.5625)	**0.04**	0.39	2.47	22.14	58.66	120.67	809.09	1,401.87	2,444.38
(0.75, 0.375)	**0.03**	**0.22**	**1.25**	9.56	25.65	55.71	324.66	639.08	1,013.19
(0.75, 0.1875)	**0.01**	**0.08**	**0.34**	**1.76**	**3.98**	**8.95**	**39.93**	**76.38**	**108.27**
(0.5, 0.5)	**0.03**	**0.19**	**1.20**	8.44	26.54	50.37	274.42	490.95	819.28
(0.5, 0.375)	**0.02**	**0.12**	**0.75**	**4.64**	14.51	27.16	135.74	256.56	407.47
(0.5, 0.25)	**0.01**	**0.07**	**0.37**	**1.78**	**5.07**	**9.58**	**40.52**	**80.67**	**122.57**
(0.5, 0.125)	**<0.01**	**0.05**	**0.21**	**0.73**	**1.59**	**2.63**	**16.99**	**40.18**	**79.69**
$LEXGO^*_{lin}$									
(0.75, 0.75)	0.07	0.51	3.10	20.88	47.68	87.01	563.05	1,006.66	1,847.88
(0.75, 0.5625)	0.07	0.55	4.12	26.45	60.75	97.91	738.42	1,191.72	2,285.11
(0.75, 0.375)	**0.05**	0.37	2.46	16.99	41.98	84.46	514.54	906.98	1,719.23
(0.75, 0.1875)	**0.02**	**0.11**	**0.60**	**3.24**	**7.07**	**16.25**	**73.87**	125.71	194.55
(0.5, 0.5)	**0.03**	0.31	2.36	14.55	39.54	60.51	387.28	620.69	1,145.15
(0.5, 0.375)	**0.02**	**0.21**	1.48	9.26	27.15	47.45	264.65	473.42	809.28
(0.5, 0.25)	**0.02**	**0.10**	**0.62**	**3.33**	**9.89**	**17.70**	78.02	152.09	229.97
(0.5, 0.125)	**0.01**	**0.06**	**0.27**	**0.97**	**2.12**	**3.51**	**21.35**	**46.80**	**92.27**
$LEXGO^*_{dr}$									
(0.75, 0.75)	0.07	0.38	1.78	10.38	17.38	30.89	105.95	187.93	300.86
(0.75, 0.5625)	**0.06**	0.35	1.68	10.62	19.30	32.24	133.15	219.81	389.68
(0.75, 0.375)	**0.04**	**0.24**	**1.14**	**6.49**	13.53	22.85	95.03	156.58	283.77
(0.75, 0.1875)	**0.01**	**0.09**	**0.38**	**1.75**	**3.62**	**6.85**	**24.03**	**41.21**	**63.41**
(0.5, 0.5)	**0.03**	**0.22**	**1.15**	**6.53**	14.44	24.03	97.85	158.71	282.49
(0.5, 0.375)	**0.03**	**0.16**	**0.81**	**4.24**	**10.39**	**17.73**	**68.58**	**117.82**	199.12
(0.5, 0.25)	**0.02**	**0.09**	**0.42**	**1.92**	**4.85**	**8.55**	**29.48**	**53.39**	**84.79**
(0.5, 0.125)	**0.01**	**0.05**	**0.24**	**0.84**	**1.82**	**2.91**	**17.45**	**40.04**	**82.18**

Chapter 7

Empirical Analysis On Road Map Problems

The rise of Google, the rise of Facebook, the rise of Apple, I think are proof that there is a place for computer science as something that solves problems that people face every day.

Eric Schmidt (1955-)

This chapter is devoted to analyze the algorithmic performance on realistic road maps from the "9th DIMACS Implementation Challenge: Shortest Path". The challenge comprises a set of twelve road maps of increasing size. In particular, we use the New York city map. Additionally we have employed a second road map from the UA Census 2000 TIGER/Line Files, which was assembled by Dr. Dominik Schultes[1] and is available from the same site. In particular, we use the Vermont State map.

All DIMACS maps provide two different cost values: *physical distance* and *travel time*. An additional cost was introduced in Machuca & Mandow (2011) by calculating the *travel economic cost*. This was obtained combining tolls and fuel consumption according to road category. The resulting values are not linearly correlated to those of the other cost values. The experiments reported below consider the simultaneous minimization of these three attributes, physical distance (c_1), travel time (c_2), and travel economic cost (c_3) (further details about these experiments can be seen in Section 3.2.2).

The two maps selected for our experiments have different sizes. Table 7.1 shows the coordinates and number of nodes and arcs for each map. The NY city map represents a difficult problem to be tackled with three objectives and only some problems could be solved. The other map, VT_{cut}, corresponds to a trimmed version of the original map of Vermont, reduced to approximately 70% of its original size in order to allow the solution of the complete set of experiments by all the algorithms. Renderings of the NY and VT_{cut} maps are presented in Figures 7.1 and 7.2, respectively.

We selected the first twenty problems for the New York city road map proposed in Machuca et al. (2012). These problems were randomly generated using an uniform

[1] <mail@dominik-schultes.de>

Table 7.1. Maps employed in the road map experiments. (*) corresponds to a cut of the original map.

Description	Acronym	Nodes	Arcs	Longitude	Latitude
New York city	NY	264,346	730,100	[73.5-74.5]°W	[40.3-41.3]°N
Vermont State*	VT_{cut}	69,575	152,012	[72-73.5]°W	[43.5-45.0]°N

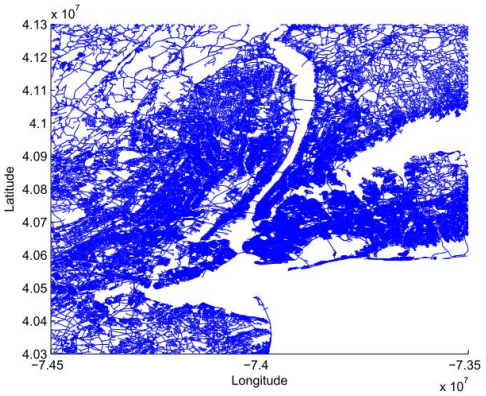


Figure 7.1. Rendering of NY city map

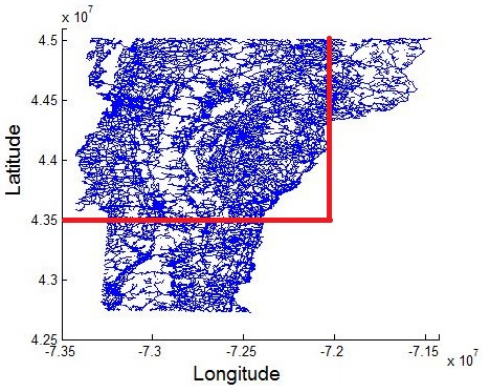


Figure 7.2. Cut of Vermont map (squared)

distribution to select start and destination nodes. In a similar manner, we generated twenty random problems for the VT_{cut} map. These test sets and the additional generated map files are available online[2].

Since previously reported runtimes of NAMOA* solving a biobjective version of these problems (minimizing c_2 and c_3, i.e. much easier problems) were up to several days (Machuca, 2011), we also established a runtime limit of 8 hours for the experiments over the NY city map.

The algorithms were implemented in Common Lisp using LispWorks Professional 6.01 (64-bit), and run on a Sun Fire X4140 server with 2 six-core AMD Opteron 2435 at 2.60 GHz processors and 64 GB of DDR2 RAM memory. This machine is slower than the one employed to solve the random grid experiments, however, it was chosen due to the greater memory space requirements of the problem analyzed here. All experiments were run on a single thread.

In the following, we conduct an experimental evaluation of the algorithms in an analogous fashion of that performed in Chapter 6 for random grid problems.

7.1 LEXGO* vs NAMOA*

This section analyzes NAMOA* and LEXGO*on the realistic road maps of Vermont and New York city. The study is applied to the lexicographic and linear selection

[2]http://alef.iaia.lcc.uma.es/projects/alef-public/wiki/Benchmarks

orders (see Definitions 2.13 and 2.14 for further description), and performed over the two classes of experiments already described for random grids in Section 6.1.

The full set of experiments on Vermont was solved by all the algorithms, however, out of the twenty problems which compose the problem set of New York city, NAMOA$^*_{\text{lex}}$ and NAMOA$^*_{\text{lin}}$ were capable of solving only four of them, NY#2, NY#4, NY#5 and NY#16. The comparison to their LEXGO* counterpart using the same selection order follows.

7.1.1 Analysis on class I experiments

Target values for LEXGO* were defined as in Equation 3.1 for random grids class I experiments. Table 7.2 displays the percentage of goal-optimal solution vectors returned by LEXGO* relative to the full set (C^*) returned by NAMOA*.

Table 7.3 shows the relative percentage number of scanned labels by LEXGO$^*_{\text{lin}}$ to NAMOA$^*_{\text{lin}}$in Vermont and NY city maps. As it was mentioned before, the number of explored labels by NAMOA$^*_{\text{lin}}$ is nearly the same as NAMOA$^*_{\text{lex}}$. This slight difference is attributed to the *lazy filtering* procedure applied to both alternatives. This is also the case of LEXGO$^*_{\text{lex}}$ and LEXGO$^*_{\text{lin}}$.

Finally, Tables 7.4 and 7.5 display the runtimes of NAMOA* and the relative time of LEXGO* to NAMOA* for the maps of Vermont and NY city with the lexicographic and linear selection orders, respectively. When the runtime of NAMOA* was smaller than 0.01 seconds, we display < 0.01 in the tables, and complete the relative percentages of LEXGO* with dashes to indicate that those are not significative. Likewise, if the relative percentage of runtime of LEXGO* to NAMOA* was smaller than 0.01%, we simply display < 0.01.

The results obtained for the road map problems are much more heterogeneous than the ones for random grids. Thus, we can observe a wide range of goal-optimal solution ratios, for instance, the range of ratios of goal-optimal solution vectors returned for $k_1 = 0.75$ varies from 1.64% to 92.82% (see Table 7.2) for Vermont map problems (VT#16 and VT#13, respectively). This could be expected, since the road map experiments represent a realistic scenario, on the contrary, the random grids were generated using an uniform distribution. However, both scenarios share the inability to satisfy the goals when $k_1 = 0.25$ or $k_1 = 0$.

The heterogeneity of the results remains regarding the labels scanned and the runtime of LEXGO*, however, it shall be noticed that the runtime comparison between NAMOA$^*_{\text{lex}}$ and NAMOA$^*_{\text{lin}}$ does not show a clear advantage to NAMOA$^*_{\text{lin}}$, as it happens in the grids experiments.

Regarding the lexicographic order, LEXGO* does perform in a very similar manner for the road maps as for the random grids. LEXGO* explores for the most difficult problem solved by both algorithms, VT#11 of Vermont, 98.27%, 72.55% and 69.46% of the labels explored by NAMOA* for $k_1 = 1$, $k_1 = 0.75$ and $k_1 = 0.5$, respectively.

The small time overhead observed in grid experiments for $k_1 = 1$ is also found in some of the road map experiments, as well as the important reductions in labels scanned and runtimes when $k_1 = 0.25$ or $k_1 = 0$, regardless the function employed as selection order.

We will further analyze in depth these data and the class II experiments reported

Table 7.2. Class I experiments in road maps, percentage of goal-optimal solution vectors relative to C^* for solvable problems within the time limit by LEXGO* and NAMOA*. An asterisk (*) indicates that the goals could not be satisfied.

			LEXGO*					
		NAMOA*	1	0.75	0.5	0.25	0	k_1
	n	$\|C^*\|$	%	%	%	%	%	
Vermont	1	1,252	100	3.04	0.64	*0.08	*0.08	
	2	223	100	38.57	34.53	*0.45	*0.45	
	3	34	100	67.65	29.41	*2.94	*2.94	
	4	4,759	100	57.11	17.42	*0.02	*0.02	
	5	334	100	36.23	1.80	*0.30	*0.30	
	6	7,576	100	84.52	19.97	*0.01	*0.01	
	7	3	100	33.33	*33.33	*33.33	*33.33	
	8	5	100	60.00	20.00	*20.00	*20.00	
	9	206	100	47.09	23.30	*0.49	*0.49	
	10	9,712	100	82.22	20.15	*0.01	*0.01	
	11	14,537	100	54.95	48.81	*0.01	*0.01	
	12	1,648	100	32.52	*0.06	*0.06	*0.06	
	13	10,256	100	92.82	43.80	*0.01	*0.01	
	14	444	100	55.41	21.17	*0.23	*0.23	
	15	1,310	100	49.16	4.81	*0.08	*0.08	
	16	1,216	100	1.64	*0.08	*0.08	*0.08	
	17	8,189	100	58.30	23.70	*0.01	*0.01	
	18	38	100	65.79	18.42	*2.63	*2.63	
	19	1	100	100.00	100.00	100.00	100.00	
	20	4,949	100	33.97	9.23	*0.02	*0.02	
NY city	2	303	100	57.42	4.62	*0.33	*0.33	
	4	4,429	100	88.82	74.32	*0.02	*0.02	
	5	7	100	71.42	*14.28	*14.28	*14.28	
	16	1,640	100	48.35	*0.06	*0.06	*0.06	

Table 7.3. Class I experiments in road maps, relative percentage number of scanned labels by $\text{LEXGO}^*_{\text{lin}}$ to $\text{NAMOA}^*_{\text{lin}}$.

			$\text{LEXGO}^*_{\text{lin}}$					
		$\text{NAMOA}^*_{\text{lin}}$	1	0.75	0.5	0.25	0	k_1
	n	$\sum G_{cl}$	%	%	%	%	%	
Vermont	1	211,268	99.73	46.80	11.63	4.05	0.07	
	2	115,435	99.99	50.41	43.08	17.81	0.44	
	3	11,332	96.86	83.15	49.28	15.91	2.06	
	4	1,134,467	99.99	80.17	32.35	2.95	0.03	
	5	45,650	96.65	72.95	16.58	3.59	1.65	
	6	5,497,553	99.17	93.42	44.25	2.88	0.01	
	7	187	73.26	55.61	55.08	57.75	45.99	
	8	480	88.33	61.46	33.96	22.29	22.08	
	9	65,140	99.76	67.24	39.98	15.10	0.71	
	10	5,332,256	98.31	86.29	33.43	9.20	0.02	
	11	10,125,074	98.27	72.55	69.46	19.64	0.02	
	12	127,611	97.45	29.81	5.49	3.33	0.54	
	13	8,664,536	99.98	97.45	70.83	20.19	0.01	
	14	47,215	99.30	76.21	34.57	4.91	0.19	
	15	571,195	99.95	93.83	68.61	3.35	0.21	
	16	88,699	96.76	47.69	33.91	20.55	0.44	
	17	1,223,581	99.60	73.43	26.25	4.96	0.16	
	18	11,021	76.33	52.55	36.53	13.51	2.79	
	19	92	64.13	64.13	64.13	64.13	64.13	
	20	1,904,080	99.99	71.65	39.55	11.26	0.03	
NY city	2	17,294	93.96	61.91	17.83	2.80	3.64	
	4	3,390,656	99.40	92.23	74.00	9.14	0.01	
	5	719	47.98	32.96	10.43	6.53	8.20	
	16	2,445,191	81.14	47.17	11.67	0.53	0.01	

Table 7.4. Class I experiments in road maps, relative percentage runtimes in seconds for $\text{LEXGO}^*_{\text{lex}}$ and $\text{NAMOA}^*_{\text{lex}}$.

			$\text{LEXGO}^*_{\text{lex}}$					
		$\text{NAMOA}^*_{\text{lex}}$	1	0.75	0.5	0.25	0	k_1
	n	Runtime (s)	%	%	%	%	%	
Vermont	1	39.95	106.68	16.20	3.12	1.21	0.04	
	2	7.05	118.81	41.81	32.29	18.81	0.67	
	3	0.32	137.80	117.18	58.43	18.90	<0.01	
	4	1,386.53	99.14	59.88	7.81	0.28	<0.01	
	5	2.49	111.26	66.23	22.52	2.48	2.52	
	6	17,731.38	95.35	76.09	12.22	0.18	<0.01	
	7	<0.01	-	-	-	-	-	
	8	<0.01	-	-	-	-	-	
	9	3.60	121.23	61.48	28.56	16.46	0.42	
	10	17,828.99	100.05	73.27	6.57	1.62	<0.01	
	11	30,318.76	99.85	57.99	47.17	2.77	<0.01	
	12	40.21	97.90	11.09	1.09	0.85	0.04	
	13	29,337.51	104.79	97.68	38.46	7.92	<0.01	
	14	3.83	102.84	68.71	26.81	3.26	<0.01	
	15	115.33	102.54	83.58	42.54	0.78	0.03	
	16	8.54	94.53	18.61	18.25	8.57	<0.01	
	17	2,245.32	96.76	44.88	5.07	0.39	<0.01	
	18	0.34	104.66	113.70	40.82	18.37	<0.01	
	19	<0.01	-	-	-	-	-	
	20	1,899.15	95.95	33.55	7.18	2.23	<0.01	
NY city	2	1.48	97.97	54.72	15.54	0.67	0.67	
	4	4,752.77	101.94	84.97	59.52	1.57	<0.01	
	5	<0.01	-	-	-	-	-	
	16	559.76	81.93	41.86	11.30	0.18	<0.01	

Table 7.5. Class I experiments in road maps, relative percentage runtimes in seconds for $\text{LEXGO}^*_{\text{lin}}$ and $\text{NAMOA}^*_{\text{lin}}$.

			$\text{LEXGO}^*_{\text{lin}}$					
		$\text{NAMOA}^*_{\text{lin}}$	1	0.75	0.5	0.25	0	k_1
	n	Runtime (s)	%	%	%	%	%	
Vermont	1	40.02	98.71	20.78	4.80	1.29	0.04	
	2	7.35	125.05	47.35	42.46	22.71	0.22	
	3	0.30	194.93	166.66	94.59	26.35	5.41	
	4	1,428.44	88.94	55.66	7.46	0.33	<0.01	
	5	2.62	136.28	97.02	19.65	2.40	1.18	
	6	17,913.28	90.27	77.75	12.81	0.19	<0.01	
	7	<0.01	-	-	-	-	-	
	8	<0.01	-	-	-	-	-	
	9	3.42	131.02	73.95	45.22	21.45	0.91	
	10	17,167.54	100.51	70.97	8.03	1.83	<0.01	
	11	28,712.69	96.00	52.18	42.35	3.83	<0.01	
	12	39.25	106.76	12.44	1.31	0.60	0.20	
	13	27,556.06	101.59	95.62	34.33	9.40	<0.01	
	14	3.88	127.32	91.99	35.35	4.02	<0.01	
	15	108.65	102.89	93.65	60.95	1.38	0.04	
	16	7.64	109.59	28.98	21.23	15.50	0.20	
	17	2,077.78	100.42	51.94	5.04	0.48	<0.01	
	18	0.52	93.79	96.89	45.44	12.23	3.11	
	19	<0.01	-	-	-	-	-	
	20	1,699.91	104.33	29.30	9.66	2.87	<0.01	
NY city	2	1.43	129.37	86.01	25.87	2.09	0.69	
	4	3,963.98	110.34	105.57	85.97	4.35	<0.01	
	5	<0.01	-	-	-	-	-	
	16	712.59	71.32	52.04	11.02	0.14	<0.01	

Table 7.6. Class II experiments in road maps, LEXGO* percentage of goal-optimal solution vectors relative to the size of C^*. An asterisk (*) indicates that the goals could not be satisfied.

			LEXGO*								
			0.75				0.5				k_1
		NAMOA*	0.75	0.5625	0.375	0.1875	0.5	0.375	0.25	0.125	k_2
	n	$\lvert C^*\rvert$	%	%	%	%	%	%	%	%	
Vermont	1	1,252	3.04	3.04	0.32	0.08	0.64	0.08	0.08	0.08	
	2	223	38.57	36.77	24.66	14.80	34.53	24.66	15.70	11.21	
	3	34	67.65	64.71	55.88	35.29	29.41	20.59	5.88	2.94	
	4	4,759	57.11	54.32	29.57	8.01	17.42	6.41	1.41	0.02	
	5	334	36.23	24.25	12.28	0.30	1.80	0.30	0.30	0.30	
	6	7,576	84.52	68.99	42.56	9.75	19.97	6.89	0.01	0.01	
	7	3	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33	
	8	5	60.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	
	9	206	47.09	36.89	30.58	16.50	23.30	22.33	16.02	0.49	
	10	9,712	82.22	49.24	14.93	1.01	20.15	4.17	0.01	0.01	
	11	14,537	54.95	53.64	42.46	21.73	48.81	39.85	23.66	7.36	
	12	1,648	32.52	31.98	31.98	18.63	0.06	0.06	0.06	0.06	
	13	10,256	92.82	89.12	53.64	20.69	43.80	22.24	13.31	0.09	
	14	444	55.41	49.77	33.56	3.60	21.17	7.88	0.68	0.23	
	15	1,310	49.16	34.43	28.24	14.73	4.81	4.12	2.82	0.08	
	16	1,216	1.64	0.08	0.08	0.08	0.08	0.08	0.08	0.08	
	17	8,189	58.30	55.50	32.34	3.04	23.70	8.19	0.10	0.01	
	18	38	65.79	65.79	47.37	42.11	18.42	10.53	10.53	2.63	
	19	1	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	
	20	4,949	33.97	25.16	15.15	0.02	9.23	2.61	0.02	0.02	
NY city	2	303	57.42	50.49	30.69	*0.33	4.62	*0.33	*0.33	*0.33	
	4	4,429	88.82	87.76	70.28	8.19	74.32	63.44	21.49	0.29	
	5	7	71.42	42.85	28.57	*14.28	*14.28	*14.28	*14.28	*14.28	
	16	1,640	48.53	17.98	3.04	*0.06	*0.06	*0.06	*0.06	*0.06	

below in the summary section.

7.1.2 Analysis on class II experiments

In the second class of experiments, goal preferences and target values were defined using $k_1 = \{0.75, 0.5\}$ with all possible values of k_2 defined in Equation 3.2. Tables 7.6 displays the relative number of goal-optimal solution vectors to the full Pareto set. The number of scanned labels for these maps is shown in Table 7.7. Finally, Tables 7.8 and 7.9 show runtimes of LEXGO* and NAMOA* with lexicographic and linear selection orders, respectively.

There are several cases where the solution returned is exactly the same. When problems NY#5 and NY#16 are solved with $k_1 = 0.5$ the only solution returned, the one which minimizes the deviation from goals, is the same for all values of k_2.

It can be observed that both, $\text{LEXGO}^*_{\text{lin}}$ and $\text{LEXGO}^*_{\text{lex}}$, outperform their NAMOA* counterparts when either $k_1 = 0.75$ or $k_1 = 0.5$, except for the problem VT#3.

Table 7.7. Class II experiments in road maps, relative number of scanned labels by LEXGO* and NAMOA* on lexicographic selection order.

			LEXGO$^*_{\text{lin}}$								
			0.75				0.5				k_1
		NAMOA$^*_{\text{lin}}$	0.75	0.5625	0.375	0.1875	0.5	0.375	0.25	0.125	k_2
	n	$\sum G_{cl}$	%	%	%	%	%	%	%	%	
Vermont	1	211,268	46.80	44.21	33.78	31.55	11.63	9.91	9.91	9.91	
	2	115,435	50.41	49.98	43.35	32.32	43.08	37.83	30.73	19.11	
	3	11,332	83.15	78.08	72.31	43.98	49.28	44.98	26.45	23.01	
	4	1,134,467	80.17	79.10	54.28	25.85	32.35	15.50	4.90	3.69	
	5	45,650	72.95	61.01	32.07	16.42	16.58	11.82	11.41	11.41	
	6	5,497,553	93.42	83.51	55.34	14.48	44.25	28.19	8.94	8.94	
	7	187	55.61	55.61	55.61	55.61	55.08	55.08	55.08	55.08	
	8	480	61.46	47.29	22.29	22.08	33.96	22.29	22.08	22.08	
	9	65,140	67.24	61.04	45.71	32.52	39.98	32.14	26.18	11.74	
	10	5,332,256	86.29	63.39	26.69	4.26	33.43	13.75	5.17	5.17	
	11	10,125,074	72.55	72.24	68.17	40.86	69.46	65.85	51.88	19.27	
	12	127,611	29.81	26.58	26.54	15.69	5.49	5.49	5.49	5.49	
	13	8,664,536	97.45	97.09	84.46	27.20	70.83	60.09	31.32	1.89	
	14	47,215	76.21	66.15	37.68	7.09	34.57	18.48	3.60	2.37	
	15	571,195	93.83	87.18	75.67	56.63	68.61	60.84	52.78	28.48	
	16	88,699	47.69	41.25	41.25	41.25	33.91	33.91	33.91	33.91	
	17	1,223,581	73.43	71.77	57.03	20.66	26.25	14.31	1.89	1.66	
	18	11,021	52.55	52.21	49.68	46.14	36.53	35.68	35.01	31.46	
	19	92	64.13	64.13	64.13	64.13	64.13	64.13	64.13	64.13	
	20	1,904,080	71.65	66.27	46.16	16.18	39.55	24.51	11.51	11.51	
NY city	2	17,294	61.91	53.45	32.15	6.74	17.83	12.25	12.25	12.25	
	4	3,390,656	92.23	91.91	82.21	25.98	74.00	67.66	44.58	7.05	
	5	719	32.96	25.45	17.80	9.59	10.43	10.43	10.43	10.43	
	16	2,445,191	47.17	25.79	9.41	2.76	11.67	11.67	11.67	11.67	

Table 7.8. Class II experiments in road maps, runtimes in seconds of $\text{NAMOA}^*_{\text{lex}}$ and $\text{LEXGO}^*_{\text{lex}}$ percentage of runtime compared to $\text{NAMOA}^*_{\text{lex}}$.

			$\text{LEXGO}^*_{\text{lex}}$								
			0.75				0.5				k_1
		$\text{NAMOA}^*_{\text{lex}}$	0.75	0.5625	0.375	0.1875	0.5	0.375	0.25	0.125	k_2
	n	Runtime (s)	%	%	%	%	%	%	%	%	
Vermont	1	39.95	16.20	16.01	18.04	32.95	3.12	3.71	4.06	5.15	
	2	7.05	41.81	43.37	33.19	30.76	32.29	27.43	25.88	17.03	
	3	0.32	117.18	104.57	97.5	52.13	58.43	52.44	29.06	28.66	
	4	1,386.53	59.88	62.11	23.77	4.94	7.81	1.45	0.25	0.32	
	5	2.49	66.23	56.85	41.87	25.64	22.52	9.98	12.5	21.23	
	6	17,731.38	76.09	56.61	23.61	1.25	12.22	4.62	0.74	1.59	
	7	<0.01	-	-	-	-	-	-	-	-	
	8	<0.01	-	-	-	-	-	-	-	-	
	9	3.60	61.48	55.87	41.99	25.53	28.56	27.70	20.79	14.29	
	10	17,828.99	73.27	29.91	3.95	0.19	6.57	1.15	0.41	0.75	
	11	30,318.76	57.99	56.67	42.22	16.53	47.17	38.78	24.12	4.01	
	12	40.21	11.09	9.12	9.62	4.15	1.09	1.09	1.05	1.09	
	13	29,337.51	97.68	91.96	53.87	6.39	38.46	22.80	6.83	0.08	
	14	3.83	68.71	57.30	38.61	7.71	26.81	12.19	1.64	1.64	
	15	115.33	83.58	75.59	53.82	36.58	42.54	35.66	31.07	16.96	
	16	8.54	18.61	18.43	23.00	22.27	18.25	17.71	15.51	16.42	
	17	2,245.32	44.88	41.85	21.15	2.45	5.07	1.11	0.06	0.05	
	18	0.34	113.7	63.85	59.18	100.00	40.82	45.48	40.82	41.11	
	19	<0.01	-	-	-	-	-	-	-	-	
	20	1,899.15	33.55	21.84	10.66	5.19	7.18	2.61	1.67	2.10	
NY city	2	1.48	57.14	50.00	37.85	4.50	16.42	15.57	17.85	17.78	
	4	4,752.77	84.97	85.43	55.18	5.61	59.52	39.85	13.19	0.86	
	5	<0.01	-	-	-	-	-	-	-	-	
	16	559.76	41.86	21.77	5.92	2.26	11.29	13.91	15.06	15.30	

Table 7.9. Class II experiments in road maps, runtimes in seconds of $\text{NAMOA}^*_{\text{lin}}$ and $\text{LEXGO}^*_{\text{lin}}$ percentage of runtime compared to $\text{NAMOA}^*_{\text{lin}}$.

			$\text{LEXGO}^*_{\text{lin}}$								
			0.75				0.5				k_1
		$\text{NAMOA}^*_{\text{lin}}$	0.75	0.5625	0.375	0.1875	0.5	0.375	0.25	0.125	k_2
	n	Runtime (s)	%	%	%	%	%	%	%	%	
Vermont	1	40.01	20.78	21.17	37.66	42.61	4.80	5.03	5.30	5.65	
	2	7.34	47.35	56.48	44.58	34.19	42.46	39.70	32.91	19.54	
	3	0.29	221.28	231.76	147.64	153.04	94.59	84.12	52.70	42.23	
	4	1,428.43	55.66	51.21	26.87	5.55	7.46	1.71	0.33	0.37	
	5	2.62	97.02	82.14	73.83	41.09	19.65	13.70	21.40	15.45	
	6	17,913.28	77.75	55.72	23.64	1.51	12.81	6.25	0.93	1.79	
	7	<0.01	-	-	-	-	-	-	-	-	
	8	<0.01	-	-	-	-	-	-	-	-	
	9	3.41	73.95	71.20	49.31	36.99	45.22	38.34	26.92	12.79	
	10	17,167.53	70.97	31.26	4.90	0.29	8.03	1.70	0.57	0.82	
	11	28,712.68	52.18	48.93	42.12	19.42	42.35	37.41	25.37	5.52	
	12	39.25	12.44	10.09	10.57	4.73	1.31	1.27	1.27	1.31	
	13	27,556.06	95.62	87.11	62.33	7.95	34.33	26.01	8.44	0.13	
	14	3.88	91.99	81.54	45.80	4.81	35.35	24.49	5.61	2.01	
	15	108.65	93.65	94.52	73.55	74.26	60.95	56.35	57.83	38.00	
	16	7.64	28.98	26.73	33.07	30.82	21.23	24.90	22.04	23.05	
	17	2,077.77	51.94	46.93	22.64	3.21	5.04	1.40	0.07	0.08	
	18	0.51	96.89	57.67	54.56	51.46	45.44	45.44	84.85	78.83	
	19	<0.01	-	-	-	-	-	-	-	-	
	20	1,699.91	29.30	24.70	18.36	8.33	9.66	6.06	2.40	2.67	
NY city	2	1.43	86.01	70.62	44.75	10.48	25.87	20.27	18.18	18.18	
	4	3,963.98	105.57	106.69	106.50	19.23	85.97	71.12	44.33	3.73	
	5	<0.01	-	-	-	-	-	-	-	-	
	16	712.59	52.04	30.86	7.84	2.12	11.02	12.47	13.04	13.47	

7.1.3 Summary

We have analyzed the relative space and runtime performance of LEXGO* over NAMOA* on road map problems. Two different functions to select the best alternative from the OPEN set have been also tested with both algorithms. Tables 7.10 and 7.11 summarize the outcome of the class I and class II experiments.

In class I experiments, the number of goal-optimal solution vectors found when $k_1 = 0.75$ is slightly smaller than for the grid experiments, whereas it is slightly greater when $k_1 = 0.5$. The scanned labels follow the same trend.

The experiments over random grids shown in Chapter 6 defined a clear advantage of the linear selection order over the lexicographic one when applied to NAMOA*. However, in our road map experiments, the practical advantage of $\text{NAMOA}^*_{\text{lin}}$ over $\text{NAMOA}^*_{\text{lex}}$ is greatly reduced to 4.3% (see Tables 7.11(a) and 7.11(b)).

The relative improvement of $\text{LEXGO}^*_{\text{lin}}$ over $\text{NAMOA}^*_{\text{lin}}$ is enhanced in comparison with the results of grids. In those, the majority of the experiments in class II with $k_1 = 0.75$ could not achieve a runtime improvement over $\text{NAMOA}^*_{\text{lin}}$. Nevertheless, $\text{LEXGO}^*_{\text{lin}}$ in road maps achieves a relative improvement over $\text{NAMOA}^*_{\text{lin}}$ very similar to the improvement achieved by $\text{LEXGO}^*_{\text{lex}}$ over $\text{NAMOA}^*_{\text{lex}}$.

7.2 $\text{NAMOA}^*_{\text{dr}}$ vs NAMOA*

This section analyzes the runtime performance of the three different versions of the NAMOA* algorithm, $\text{NAMOA}^*_{\text{lex}}$, $\text{NAMOA}^*_{\text{lin}}$ and $\text{NAMOA}^*_{\text{dr}}$. The experiments presented in this section analyze the impact of the dimensionality reduction technique on the sets of road map problems already presented.

The three versions of NAMOA* differ in the order of selection of OPEN labels and/or in the way dominance is checked in filtering and cl-pruning operations. The first and second variants are $\text{NAMOA}^*_{\text{lex}}$ and $\text{NAMOA}^*_{\text{lin}}$, and both use, to the best of our knowledge, the usual dominance pruning and filtering technique in previously reported experimental evaluations of multiobjective search algorithms. The third algorithm analyzed, $\text{NAMOA}^*_{\text{dr}}$, uses a lexicographic order of selection and the t-discarding technique, described in Section 4.2, for filtering and cl-pruning.

7.2.1 Analysis

Tables 7.12 and 7.13 show the size of relevant label sets for each problem instance solved by $\text{NAMOA}^*_{\text{dr}}$, as well as $\text{NAMOA}^*_{\text{lin}}$ and $\text{NAMOA}^*_{\text{dr}}$ runtimes for the NY city and Vermont maps, respectively. The first column displays the problem identifier (n). The description of these sets is the same presented previously in Table 6.6. Notice that for NY map only problems solved within the time limit are displayed in this table. Figure 7.3 shows the runtimes in logarithmic scale of $\text{NAMOA}^*_{\text{lin}}$ and $\text{NAMOA}^*_{\text{dr}}$ for VT_{cut} map sorted by the number of scanned labels by each problem.

Finally, Figures 7.4(a) and 7.4(b) show the percentage of labels filtered, pruned by open, and pruned by closed node labels over the total number of discarded labels by $\text{NAMOA}^*_{\text{dr}}$, for the maps of NY city and Vermont, respectively. The X-axis shows problem ids sorted by the number of scanned labels.

Table 7.10. Class I experiments in road maps, summary of the relative space and time performance of LEXGO* over NAMOA*.

(a) Relative average number of goal-optimal solution vectors for the Vermont problems

	LEXGO*					
NAMOA*	1	0.75	0.5	0.25	0	k_1
Avg. $\lvert C^* \rvert$	%	%	%	%	%	
3,334.6	100	64.34	27.89	0.03	0.03	

(b) Relative average number of scanned labels for the Vermont problems

	LEXGO$^*_{\text{lin}}$					
NAMOA$^*_{\text{lin}}$	1	0.75	0.5	0.25	0	k_1
Avg. $\sum G_{cl}$	%	%	%	%	%	
1,758,843.6	99.06	84.15	55.12	13.60	0.04	

(c) Relative average time performance of LEXGO$^*_{\text{lex}}$ to NAMOA$^*_{\text{lex}}$ for the Vermont problems

	LEXGO$^*_{\text{lex}}$					
NAMOA$^*_{\text{lex}}$	1	0.75	0.5	0.25	0	k_1
Avg. runtime (s)	%	%	%	%	%	
5,048.47	100.39	74.67	29.06	3.51	<0.01	

(d) Relative average time performance of LEXGO$^*_{\text{lin}}$ to NAMOA$^*_{\text{lin}}$ for the Vermont problems

	LEXGO$^*_{\text{lin}}$					
NAMOA$^*_{\text{lin}}$	1	0.75	0.5	0.25	0	k_1
Avg. runtime (s)	%	%	%	%	%	
4,838.46	97.49	72.28	26.61	4.25	<0.01	

Table 7.11. Class II experiments in road maps, summary of the relative space and runtime performance of LEXGO* over NAMOA* for the Vermont map experiments.

(a) Relative average number of goal-optimal solution vectors

	LEXGO*								
	0.75				0.5				k_1
NAMOA*	0.75	0.5625	0.375	0.1875	0.5	0.375	0.25	0.125	k_2
Avg. $\lvert C^* \rvert$	%	%	%	%	%	%	%	%	
3,334.6	64.34	55.25	33.59	11.04	27.89	15.47	7.50	1.68	

(b) Relative average number of scanned labels

	$\text{LEXGO}^*_{\text{lin}}$								
	0.75				0.5				k_1
$\text{NAMOA}^*_{\text{lin}}$	0.75	0.5625	0.375	0.1875	0.5	0.375	0.25	0.125	k_2
Avg. $\sum G_{cl}$	%	%	%	%	%	%	%	%	
1,758,843.6	84.15	78.37	61.34	25.29	55.12	43.97	26.89	9.74	

(c) Relative average runtime performance of $\text{LEXGO}^*_{\text{lex}}$ to $\text{NAMOA}^*_{\text{lex}}$

	$\text{LEXGO}^*_{\text{lex}}$								
	0.75				0.5				k_1
$\text{NAMOA}^*_{\text{lex}}$	0.75	0.5625	0.375	0.1875	0.5	0.375	0.25	0.125	k_2
Avg. runtime (s)	%	%	%	%	%	%	%	%	
5,048.47	74.67	61.26	34.25	7.36	29.06	19.43	9.51	1.71	

(d) Relative average runtime performance of $\text{LEXGO}^*_{\text{lin}}$ to $\text{NAMOA}^*_{\text{lin}}$

	$\text{LEXGO}^*_{\text{lin}}$								
	0.75				0.5				k_1
$\text{NAMOA}^*_{\text{lin}}$	0.75	0.5625	0.375	0.1875	0.5	0.375	0.25	0.125	k_2
Avg. runtime (s)	%	%	%	%	%	%	%	%	
4,838.46	72.28	57.51	36.81	8.77	26.61	20.20	10.33	2.26	

Table 7.12. Size of relevant sets of labels for VT$_{cut}$ road map experiments solved by NAMOA$^*_{\text{dr}}$

		Size of relevant sets of labels of NAMOA$^*_{\text{dr}}$						Runtime (sec)	
n	Max OPEN	$\sum G_{cl}$	$\sum T(G_{cl})$	%	C*	T(C*)	%	$t_{\text{NAMOA}^*\text{lin}}$	$t_{\text{NAMOA}^*\text{dr}}$
1	1,352	209,906	27,353	13.03	1,252	62	4.95	40.0	9.2
2	2,157	114,109	40,522	35.51	223	86	38.57	7.3	4.6
3	1,062	11,483	11,477	99.95	34	34	100.00	0.3	0.4
4	2,717	1,132,450	59,068	5.22	4,759	57	1.20	1,428.4	47.5
5	922	44,950	10,398	23.13	334	12	3.59	2.6	2.0
6	4,373	5,445,252	160,410	2.95	7,576	145	1.91	17,913.2	259.0
7	61	178	178	100.00	3	3	100.00	<0.01	<0.01
8	62	483	483	100.00	5	5	100.00	<0.01	<0.01
9	1,636	64,226	43,787	68.18	206	150	72.82	3.4	2.4
10	7,271	5,229,959	32,398	0.62	9,712	23	0.24	17,167.5	246.0
11	31,846	10,057,176	286,083	2.84	14,537	247	1.70	28,712.6	527.5
12	1,974	127,731	15,947	12.48	1,648	70	4.25	39.2	4.7
13	9,387	8,640,728	137,410	1.59	10,256	139	1.36	27,556.0	395.2
14	819	46,861	9,650	20.59	444	14	3.15	3.8	1.8
15	12,648	568,388	90,676	15.95	1,310	170	12.98	108.6	33.3
16	1,558	87,522	51,641	59.00	1,216	48	3.95	7.6	4.7
17	2,596	1,207,119	41,414	3.43	8,189	94	1.15	2,077.7	51.2
18	1,331	10,270	5,550	54.04	38	18	47.37	0.5	0.4
19	34	92	92	100.00	1	1	100.00	<0.01	<0.01
20	24,671	1,856,420	171,675	9.25	4,949	255	5.15	1,699.9	102.5

Table 7.13. Results of NY city road map experiments with size of relevant sets of labels of NAMOA$^*_{\text{dr}}$ and runtimes of NAMOA$^*_{\text{lin}}$ and NAMOA$^*_{\text{dr}}$.

		Size of relevant sets of labels of NAMOA$^*_{\text{dr}}$						Runtime (sec)	
n	Max OPEN	$\sum G_{cl}$	$\sum T(G_{cl})$	%	C*	T(C*)	%	$t_{\text{NAMOA}^*\text{lin}}$	$t_{\text{NAMOA}^*\text{dr}}$
1	379,060	274,567,814	2,158,829	0.79	93,464	45	0.05	-	27,632.3
2	185	17,294	1,771	10.24	303	12	3.96	1.4	0.7
3	-	-	-	-	-	-	-	-	-
4	8,636	3,390,656	28,088	0.83	4,429	24	0.54	3,963.9	149.3
5	44	719	613	85.26	7	1	14.29	<0.01	<0.01
6	152,988	80,721,099	628,829	0.78	40,606	163	0.40	-	11,641.6
7	160,079	182,473,300	1,118,218	0.61	58,410	308	5.27	-	21,768.3
8	-	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-	-
10	464,998	214,901,344	1,070,285	0.50	92,048	31	0.03	-	26,107.8
11	35,544	24,584,323	812,383	3.30	26,575	401	1.51	-	1,452.5
12	-	-	-	-	-	-	-	-	-
13	-	-	-	-	-	-	-	-	-
14	159,041	278,481,469	6,296,377	2.26	108,856	346	0.32	-	21,957.6
15	844,037	136,776,273	5,256,283	3.84	23,678	26	0.11	-	16,582.9
16	6,821	2,445,191	242,832	9.93	1,640	69	4.20	712.5	106.8
17	-	-	-	-	-	-	-	-	-
18	236,826	270,364,947	1,630,261	0.60	95,072	242	0.25	-	25,599.0
19	67,883	108,347,749	1,137,035	1.05	46,205	241	0.52	-	8,355.3
20	482,686	162,419,342	1,788,698	1.10	77,051	156	0.20	-	15,617.3

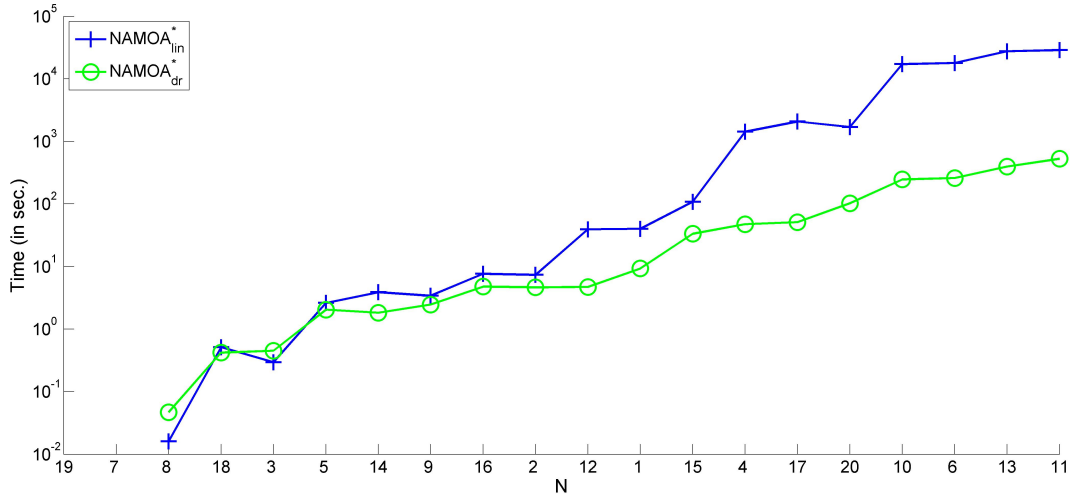


Figure 7.3. Runtimes of $\text{NAMOA}^*_{\text{lin}}$ and $\text{NAMOA}^*_{\text{dr}}$ for the VT_{cut} map problems sorted by the number of labels expanded.

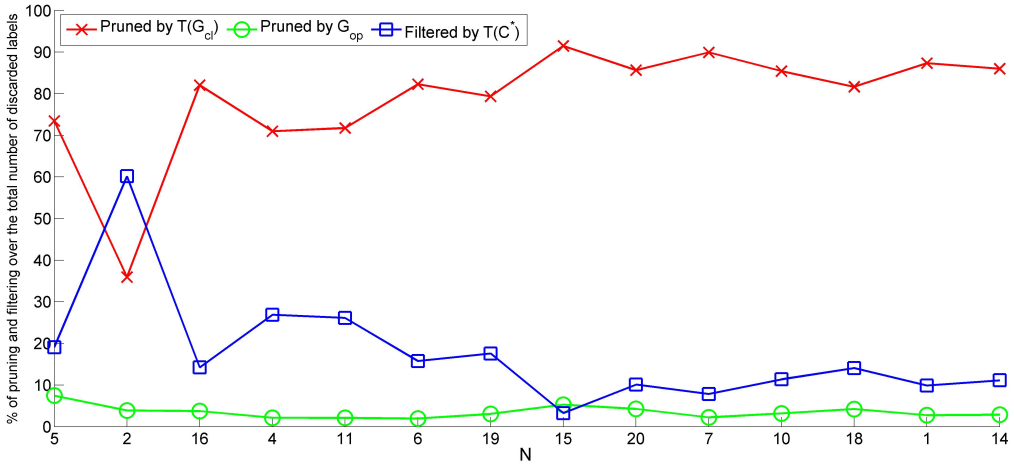
The experiments on the realistic road map problems are much harder than the random grid problems. The largest solvable one (NY#14) requiring up to 278 million label expansions (the deepest grid problems involved in average 2.5 million label expansions, and 3.3 million in the worst case). Again, $\text{NAMOA}^*_{\text{dr}}$ clearly outperformed $\text{NAMOA}^*_{\text{lin}}$, which could only solve problems involving less than 10.1 million label expansions (VT#11). For such a problem, $\text{NAMOA}^*_{\text{dr}}$ required only 1.83% of the time needed by $\text{NAMOA}^*_{\text{lin}}$.

On one hand, $\text{NAMOA}^*_{\text{lin}}$ was capable of solving within the time limit only 4 problems from the NY set (20%), while $\text{NAMOA}^*_{\text{dr}}$ solved 14 (70%) (see Table 7.13), and on the other hand, in the test set of VT_{cut} map, which was entirely solved by both algorithms, $\text{NAMOA}^*_{\text{dr}}$ requires 1.74% of the time needed by $\text{NAMOA}^*_{\text{lin}}$(see Table 7.14). Notice that for the NY map the algorithms were not capable of solving several problem instances in the given 8 hour time limit. These are indicated by symbol "-" in the table. Problems #11 and #6 were solved by $\text{NAMOA}^*_{\text{lin}}$ without time limit in 31 hours and 25 days, respectively.

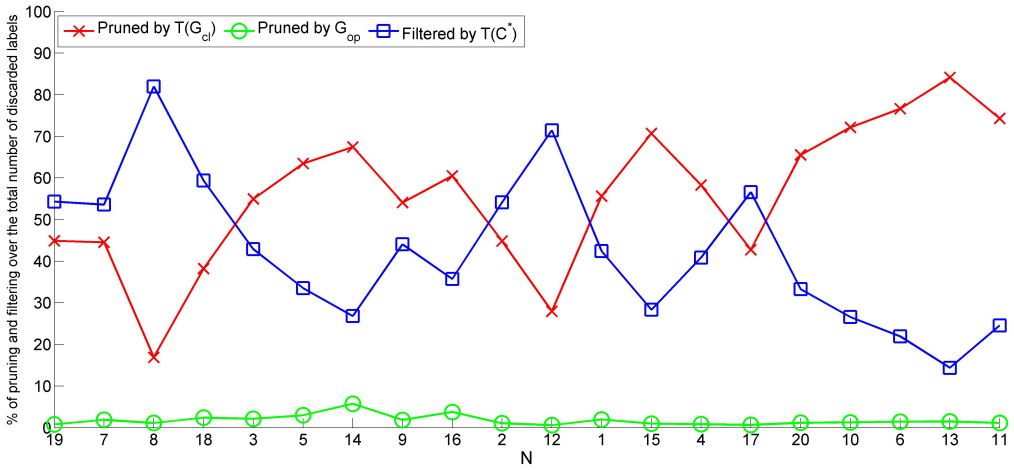
Except for the simpler problems, cl-pruning was responsible for around 70 to 80% of the discarded labels (see Figures 7.4(a) and 7.4(b)). In general, the ratio of filtered labels was larger than those discarded by op-pruning. Once again, this explains the efficiency achieved by t-discarding. Tables 7.12 and 7.13 show dramatic reductions in the sizes of the sets used for cl-pruning and filtering. For the hardest solved instance (NY#14), the size of $T(\text{C}^*)$ is just 0.32% the size of C^*. For the sets of closed labels, the ratio is 2.26%.

7.2.2 Summary

The proposed t-discarding procedure proves to be very effective, reducing the time requirements over an order of magnitude over the most efficient search with the standard dominance checks. The new technique effectively extends the size of the three-objective



(a) NY city map



(b) Vermont state map

Figure 7.4. Percentage of pruned and filtered labels over the total number of discarded labels by NAMOA$^*_{\text{dr}}$ per solution depth in road map experiments.

Table 7.14. Summary of VT_{cut} map results.

Map	$(\frac{\sum G_{cl}}{\sum T(G_{cl})})\%$	$(\frac{\sum C^*}{\sum T(C^*)})\%$	C^*	$t_{\text{NAMOA}^*\text{lin}}$	$t_{\text{NAMOA}^*\text{dr}}$	%
VT_{cut}	3.43	2.45	3,334.6	4,838.4	84.6	1.74

problems that can be practically solved and opens a range of possibilities for multi-objective search algorithms to become more time-efficient. Thus, we then analyze the runtime performance of the application of this technique to the algorithm LEXGO*.

7.3 LEXGO$^*_{\mathbf{dr}}$ vs LEXGO*

This section analyzes the three versions of LEXGO* already presented, LEXGO$^*_{\text{lex}}$, LEXGO$^*_{\text{lin}}$, and LEXGO$^*_{\text{dr}}$. A detailed description of LEXGO* and LEXGO$^*_{\text{dr}}$ are presented in Sections 4.1 and 4.4, respectively.

The experiments below analyze the impact of the dimensionality reduction technique over the sets of random road map problems previously used, employing the goals already described in previous sections. The analysis is conducted over the Vermont problems and the NY city problems able to be solved by LEXGO$^*_{\text{lex}}$ or LEXGO$^*_{\text{lin}}$. Notice that for a problem to be considered solved, all class I and II experiments must be solved within the time limit. The standard versions of LEXGO* solved four and LEXGO$^*_{\text{dr}}$ six out of the twenty problems considered in the NY city map.

7.3.1 Analysis on class I experiments

Table 7.15 displays all the runtimes in seconds corresponding to Vermont and NY city problems solved by LEXGO$^*_{\text{dr}}$. Table 7.16 summarizes the average execution times of Vermont problems for LEXGO$^*_{\text{lex}}$, LEXGO$^*_{\text{lin}}$, and LEXGO$^*_{\text{dr}}$. Finally, Table 7.17 shows the runtimes of LEXGO$^*_{\text{lex}}$, LEXGO$^*_{\text{lin}}$, and LEXGO$^*_{\text{dr}}$ for problems NY#4 and NY#16 (runtimes of problems NY#2 and NY#5 are not shown since they are too small to be significative).

In this first class of problems, LEXGO$^*_{\text{dr}}$ has a substantial advantage over both standard versions of LEXGO*. LEXGO$^*_{\text{dr}}$ solves the full set of Vermont problems in 1.87% and 2.01% of the time needed by LEXGO$^*_{\text{lex}}$ and LEXGO$^*_{\text{lin}}$ when $k_1 = 1$, respectively. The time needed by LEXGO$^*_{\text{dr}}$ is 2.21% and 2.39%, and 3.91% and 4.45% with respect to LEXGO$^*_{\text{lex}}$ and LEXGO$^*_{\text{lin}}$ when $k_1 = 0.75$ and $k_1 = 0.5$. Regarding the experiments where the goals cannot be satisfied, the improvement of LEXGO$^*_{\text{dr}}$ is 11.45% and 23.55% over LEXGO$^*_{\text{lex}}$ and LEXGO$^*_{\text{lin}}$ when $k_1 = 0.25$, respectively, whereas all the runtimes when $k_1 = 0$ are equal to 0.03 seconds. It is also worth noting that the runtime of LEXGO$^*_{\text{dr}}$ when $k_1 = 0.25$ is greater than the runtimes when goals can be satisfied, i.e. when $k_1 = \{1, 0.75, 0.5\}$. This is due to the fact that the t-discarding technique is only applied to LEXGO* when goals can be satisfied, hence, few problems when $k_1 = 0.25$ can benefit from the dimensionality reduction technique.

Let's turn now our attention to the NY city problems. Regarding problem NY#4, LEXGO$^*_{\text{dr}}$ is more than twenty times faster than LEXGO$^*_{\text{lex}}$ and LEXGO$^*_{\text{lin}}$ when $k_1 = \{1, 0.75\}$, sixteen times faster when $k_1 = 0.5$, and it has a similar performance when goals cannot be satisfied, i.e. when $k_1 = \{0.25, 0\}$.

7.3.2 Analysis on class II experiments

Regarding the class II experiments, Table 7.18 displays all the runtimes in seconds corresponding to Vermont and NY city problems solved by LEXGO$^*_{\text{dr}}$. The average

Table 7.15. Class I experiments in road maps, runtimes in seconds of LEXGO$^*_{\text{dr}}$ for the experiments over Vermont and NY city maps.

		LEXGO$^*_{\text{dr}}$					
	n	1	0.75	0.5	0.25	0	k_1
Vermont	1	9.68	4.71	1.35	0.45	<0.01	
	2	4.99	2.49	2.44	1.29	0.06	
	3	0.68	0.42	0.20	0.18	0.03	
	4	55.58	43.72	18.42	4.10	0.03	
	5	2.37	1.82	0.59	0.06	0.04	
	6	291.90	278.64	175.31	32.85	0.03	
	7	<0.01	<0.01	<0.01	<0.01	<0.01	
	8	0.03	<0.01	<0.01	<0.01	<0.01	
	9	3.16	2.15	1.37	0.60	<0.01	
	10	265.2	258.7	117.81	274.95	0.10	
	11	584.45	427.56	401.99	357.66	0.09	
	12	5.50	1.96	0.60	0.18	0.03	
	13	464.60	491.77	332.82	2413.64	0.06	
	14	2.10	1.87	0.95	0.10	<0.01	
	15	33.36	30.60	23.35	1.32	0.09	
	16	4.52	1.90	1.70	0.88	<0.01	
	17	61.37	43.24	14.90	9.26	0.07	
	18	0.37	0.24	0.32	0.07	<0.01	
	19	<0.01	<0.01	<0.01	<0.01	<0.01	
	20	108.15	80.15	53.52	43.00	0.03	
NY city	2	0.87	0.65	0.28	0.01	<0.01	
	4	201.64	198.18	176.04	63.94	0.03	
	5	<0.01	<0.01	<0.01	<0.01	<0.01	
	11	1,753.57	1,668.50	1,366.86	3,664.96	0.07	
	16	118.63	116.22	62.97	0.96	0.01	
	19	10,873.42	10,960.56	10,005.21	1,506.29	0.45	

Table 7.16. Class I experiments in road maps, summary of Vermont problems runtimes in seconds of LEXGO$^*_{\text{lex}}$, LEXGO$^*_{\text{lin}}$ and LEXGO$^*_{\text{dr}}$.

	1	0.75	0.5	0.25	0	k_1
LEXGO$^*_{\text{lex}}$	5,068.00	3,769.51	1,467.00	177.33	0.03	
LEXGO$^*_{\text{lin}}$	4,716.90	3,497.26	1,287.37	205.39	0.03	
LEXGO$^*_{\text{dr}}$	94.90	83.59	57.38	157.02	0.03	

Table 7.17. Class I experiments in road maps, runtimes of $\text{LEXGO}^*_{\text{lex}}$, $\text{LEXGO}^*_{\text{lin}}$, and $\text{LEXGO}^*_{\text{dr}}$ for two NY city problems.

Problem	4			16		
Algorithm	$\text{LEXGO}^*_{\text{lex}}$	$\text{LEXGO}^*_{\text{lin}}$	$\text{LEXGO}^*_{\text{dr}}$	$\text{LEXGO}^*_{\text{lex}}$	$\text{LEXGO}^*_{\text{lin}}$	$\text{LEXGO}^*_{\text{dr}}$
$(k_1 = 1)$	4,845.19	4,379.98	201.64	458.64	508.23	118.63
$(k_1 = 0.75)$	4,038.80	4,184.89	198.18	234.34	370.87	116.22
$(k_1 = 0.5)$	2,829.04	3,407.90	176.04	63.26	78.54	62.97
$(k_1 = 0.25)$	74.89	172.52	63.94	1.03	1.01	0.96
$(k_1 = 0)$	0.01	0.03	0.03	0.01	0.01	0.01

runtimes of Vermont problems for $\text{LEXGO}^*_{\text{lex}}$, $\text{LEXGO}^*_{\text{lin}}$, and $\text{LEXGO}^*_{\text{dr}}$ are summarized in Table 7.19.

In these experiments, $\text{LEXGO}^*_{\text{dr}}$ also outperforms $\text{LEXGO}^*_{\text{lex}}$ and $\text{LEXGO}^*_{\text{lin}}$. The relative advantage of $\text{LEXGO}^*_{\text{dr}}$ over $\text{LEXGO}^*_{\text{lex}}$ and $\text{LEXGO}^*_{\text{lin}}$ grows when the number of goal-optimal solution vectors is greater, as well as with the size of the problem. Thus, the time needed by $\text{LEXGO}^*_{\text{dr}}$ to solve the full set of Vermont problems is 41.87% of the time needed by $\text{LEXGO}^*_{\text{lex}}$ when $k_1 = 0.5$ and $k_2 = 0.125$. This comparative advantage grows to its maximum when $k_1 = k_2 = 0.75$, where $\text{LEXGO}^*_{\text{dr}}$ runtime is 2.21% of $\text{LEXGO}^*_{\text{lex}}$ runtime, i.e. $\text{LEXGO}^*_{\text{dr}}$ is more than 45 times faster than $\text{LEXGO}^*_{\text{lex}}$.

Table 7.20 shows a comparative of $\text{LEXGO}^*_{\text{lex}}$, $\text{LEXGO}^*_{\text{lin}}$, and $\text{LEXGO}^*_{\text{dr}}$ runtimes for problems NY#4 and NY#16. Problem NY#16 does not show an apparent improvement when using $\text{LEXGO}^*_{\text{dr}}$ in many cases, since those cases cannot satisfy the provided goals.

7.3.3 Summary

In a similar manner as $\text{LEXGO}^*_{\text{dr}}$ outperforms LEXGO^* in random grids, the results do not change for road map problems. A greater speed-up can be observed in these experiments. When a large number of goal-optimal solution vectors from the Pareto frontier satisfy the goals, the t-discarding method is applied to a greater extent to $\text{LEXGO}^*_{\text{dr}}$ and speeds up the performance to up to 50 times faster than any previous version of LEXGO^*. Since this advantage grows with difficulty, the obtained results point out that $\text{LEXGO}^*_{\text{dr}}$ must be chosen over LEXGO^* in problems with a certain level of difficulty and satisfiable goals, although when goals cannot be satisfied $\text{LEXGO}^*_{\text{dr}}$ does only contribute with a slight advantage.

7.4 $\text{LEXGO}^*_{\text{dr}}$ vs $\text{NAMOA}^*_{\text{dr}}$

This section considers $\text{NAMOA}^*_{\text{dr}}$ and $\text{LEXGO}^*_{\text{dr}}$, versions of NAMOA^* and LEXGO^* that employ the t-discarding technique described in Section 4.2. Since t-discarding can only be applied to LEXGO^* when goals can be satisfied, the question of its comparative performance to $\text{NAMOA}^*_{\text{dr}}$ also arises for the road map problems.

$\text{NAMOA}^*_{\text{dr}}$ has been empirically proved to be more efficient than NAMOA^*. In the same manner, $\text{LEXGO}^*_{\text{dr}}$ outperforms LEXGO^*. $\text{NAMOA}^*_{\text{dr}}$ obtains a greater ratio

Table 7.18. Class II experiments in road maps, runtimes in seconds of LEXGO$^*_{\text{dr}}$ for the experiments over Vermont and NY city maps.

		LEXGO$^*_{\text{dr}}$								
		0.75				0.5				k_1
	n	0.75	0.5625	0.375	0.1875	0.5	0.375	0.25	0.125	k_2
Vermont	1	4.71	4.71	6.32	13.90	1.36	1.42	1.78	2.20	
	2	2.50	3.01	2.61	2.28	2.45	2.06	1.90	1.17	
	3	0.42	0.64	0.38	0.20	0.20	0.20	0.13	0.11	
	4	43.73	46.94	31.79	22.11	18.42	8.66	2.98	5.18	
	5	1.83	1.47	0.91	0.70	0.59	0.30	0.34	0.53	
	6	278.65	266.84	184.47	58.47	175.31	211.48	93.07	281.11	
	7	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	
	8	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	
	9	2.15	1.92	1.56	1.00	1.37	0.91	0.78	0.61	
	10	258.79	200.35	134.57	26.68	117.81	81.26	57.49	146.35	
	11	427.57	418.82	394.06	264.59	402.00	395.54	313.09	209.03	
	12	1.97	1.92	1.78	0.90	0.67	0.45	0.64	0.45	
	13	491.78	450.97	438.63	144.61	332.83	304.70	189.07	17.80	
	14	1.87	1.53	1.00	0.31	0.95	0.47	0.08	0.08	
	15	30.61	28.50	24.96	26.46	23.35	22.56	22.51	15.85	
	16	1.95	2.12	2.65	2.14	1.70	1.73	1.54	1.79	
	17	43.24	43.51	35.51	25.83	14.93	10.09	1.42	1.64	
	18	0.25	0.23	0.22	0.39	0.33	0.17	0.17	0.17	
	19	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	
	20	80.15	92.57	84.66	97.33	53.52	36.50	31.65	38.94	
NY city	2	0.65	0.60	0.42	0.15	0.28	0.23	0.25	0.25	
	4	198.18	220.77	225.99	128.49	176.04	177.32	170.47	35.67	
	5	<0.01	-	-	-	-	-	-	-	
	11	1,668.50	1,617.41	2,653.90	2,852.52	1,366.86	1,860.42	2,727.67	4,532.50	
	16	116.22	86.22	31.18	12.69	62.97	77.39	86.87	84.35	
	19	10,960.56	15,407.65	20,771.40	5,270.44	10,005.21	12,456.02	5,141.12	4,519.28	

Table 7.19. Class II experiments in road maps, runtimes in seconds of LEXGO$^*_{\text{dr}}$ for the experiments over Vermont and NY city maps.

	0.75				0.5				k_1
	0.75	0.5625	0.375	0.1875	0.5	0.375	0.25	0.125	k_2
LEXGO$^*_{\text{lex}}$	3,769.51	3,092.73	1,729.26	371.33	1,467.00	980.69	479.92	86.32	
LEXGO$^*_{\text{lin}}$	3,497.26	2,782.82	1,781.04	424.25	1,287.37	977.48	499.61	109.20	
LEXGO$^*_{\text{dr}}$	83.59	78.30	67.31	34.40	57.39	53.93	35.93	36.15	

Table 7.20. Class II experiments in road maps, runtimes in seconds of $\text{LEXGO}^*_{\text{lex}}$, $\text{LEXGO}^*_{\text{lin}}$, and $\text{LEXGO}^*_{\text{dr}}$ for two NY city problems.

Problem	4			16		
Algorithm	$\text{LEXGO}^*_{\text{lex}}$	$\text{LEXGO}^*_{\text{lin}}$	$\text{LEXGO}^*_{\text{dr}}$	$\text{LEXGO}^*_{\text{lex}}$	$\text{LEXGO}^*_{\text{lin}}$	$\text{LEXGO}^*_{\text{dr}}$
(0.75, 0.75)	4,038.8	4,184.9	198.1	234.3	370.8	116.2
(0.75, 0.5625)	4,060.7	4,229.3	220.7	121.9	219.9	86.2
(0.75, 0.375)	2,622.8	4,222.0	226.0	33.1	55.9	31.1
(0.75, 0.1875)	267.1	762.3	128.5	12.6	15.1	12.7
(0.5. 0.5)	2,829.0	3,407.9	176.0	63.2	78.5	62.9
(0.5. 0.375)	1,894.2	2,819.4	177.3	77.8	88.9	77.3
(0.5. 0.25)	627.1	1,757.5	170.5	84.3	92.9	86.8
(0.5. 0.125)	41.0	148.0	35.6	85.6	96.0	84.3

of improvement over the standard versions of NAMOA* than $\text{LEXGO}^*_{\text{dr}}$ achieves over their counterparts of LEXGO*. Thus, a final comparative analysis for random map problems between $\text{NAMOA}^*_{\text{dr}}$ and $\text{LEXGO}^*_{\text{dr}}$ is presented in this section.

The experiments are applied to the road map problems previously used. The goals employed to define the set of goal-optimal solution vectors are defined analogously as in Section 6.1. The analysis is conducted over the Vermont problems and the NY city problems able to be solved by $\text{LEXGO}^*_{\text{dr}}$ and $\text{NAMOA}^*_{\text{dr}}$, which were six and fourteen out of twenty, respectively.

7.4.1 Analysis on class I experiments

In Table 7.21 the results of $\text{NAMOA}^*_{\text{dr}}$ and $\text{LEXGO}^*_{\text{dr}}$ are presented. Cases where $\text{LEXGO}^*_{\text{dr}}$ runtimes are faster than runtimes of $\text{NAMOA}^*_{\text{dr}}$ are highlighted in bold. The average runtimes of the full set of Vermont map experiments are shown in Table 7.22.

An expected overhead in $\text{LEXGO}^*_{\text{dr}}$ over $\text{NAMOA}^*_{\text{dr}}$ is found when $k_1 = 1$. In this case, the overhead corresponding to the extra deviation pruning and filtering efforts represents 12.08% of the runtime of $\text{NAMOA}^*_{\text{dr}}$. $\text{LEXGO}^*_{\text{dr}}$ shows a very similar runtime in average when $k_1 = 0.75$, only 1.26% faster. When $k_1 = 0.5$, $\text{LEXGO}^*_{\text{dr}}$ clearly outperforms $\text{NAMOA}^*_{\text{dr}}$, being 32.22% more efficient in runtime. As always, when the goals are located in the ideal point ($k_1 = 0$), $\text{LEXGO}^*_{\text{dr}}$, with an average runtime of three hundredths of a second, outperforms $\text{NAMOA}^*_{\text{dr}}$.

A particular case of $\text{LEXGO}^*_{\text{dr}}$ can be observed in $k_1 = 0.25$ case. Problem VT#13 is solved by $\text{LEXGO}^*_{\text{dr}}$ more than six times slower than by $\text{NAMOA}^*_{\text{dr}}$, and problem NY#11 two and a half times slower. In order to understand this behavior we measured the percentage of times that a pruning operation is conducted in a time-efficient manner with respect to the total of pruning operations. We call this quotient $\eth$. Thus, $\eth_{13VT} = 0.4698$ and $\eth_{11NY} = 0.6938$ while other problems of similar difficulty, VT#11 for example, present a much higher value ($\eth_{11} = 0.9531$). In other words, problem VT#13 cannot use the Pareto pruning (dr) as often as the rest of the problems, in this case only 46.98% of the times, and hence, its runtime is much higher than the average.

Table 7.21. Class I experiments in road maps, runtimes in seconds of NAMOA$^*_{dr}$ and LEXGO$^*_{dr}$ for the experiments over Vermont and NY city maps.

			LEXGO$^*_{dr}$					
	n	NAMOA$^*_{dr}$	1	0.75	0.5	0.25	0	k_1
Vermont	1	**9.29**	9.68	**4.71**	**1.35**	**0.45**	**<0.01**	
	2	**4.64**	4.99	**2.49**	**2.44**	**1.29**	**0.06**	
	3	**0.45**	0.68	**0.42**	**0.20**	**0.18**	**0.03**	
	4	**47.53**	55.58	**43.72**	**18.42**	**4.10**	**0.03**	
	5	**2.04**	2.37	**1.82**	**0.59**	**0.06**	**0.04**	
	6	**259.05**	291.90	278.64	**175.31**	**32.85**	**0.03**	
	7	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	
	8	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	
	9	**2.46**	3.16	**2.15**	**1.37**	**0.60**	**<0.01**	
	10	**246.09**	265.20	258.70	**117.81**	274.95	**0.10**	
	11	**527.56**	584.45	**427.56**	**401.99**	**357.66**	**0.09**	
	12	**4.71**	5.50	**1.96**	**0.60**	**0.18**	**0.03**	
	13	**395.24**	464.60	491.77	**332.82**	2,413.64	**0.06**	
	14	**1.82**	2.10	1.87	**0.95**	**0.10**	**<0.01**	
	15	**33.35**	33.36	**30.60**	**23.35**	**1.32**	**0.09**	
	16	**4.77**	**4.52**	**1.90**	**1.70**	**0.88**	**<0.01**	
	17	**51.26**	61.37	**43.24**	**14.90**	**9.26**	**0.07**	
	18	**0.42**	**0.37**	**0.24**	**0.32**	**0.07**	**<0.01**	
	19	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	
	20	**102.57**	108.15	**80.15**	**53.52**	**43.00**	**0.03**	
NY city	2	**0.71**	0.87	**0.65**	**0.28**	**0.01**	**<0.01**	
	4	**149.30**	201.64	198.18	176.04	**63.94**	**0.03**	
	5	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	
	11	**1,452.58**	1,753.57	1,668.50	**1,366.86**	3,664.96	**0.07**	
	16	**106.89**	118.63	116.22	**62.97**	**0.96**	**0.01**	
	19	**8,355.32**	10,873.42	10,960.56	10,005.21	**1,506.29**	**0.45**	

Table 7.22. Class I experiments in road maps, average runtimes in seconds of NAMOA$^*_{dr}$ and LEXGO$^*_{dr}$ for the set of Vermont map experiments.

	LEXGO$^*_{dr}$					
NAMOA$^*_{dr}$	1	0.75	0.5	0.25	0	k_1
84.66	94.89	83.59	57.38	157.02	0.03	

Table 7.23. Class II experiments in road maps, runtimes in seconds of NAMOA$^*_{\text{dr}}$ and LEXGO$^*_{\text{dr}}$ for the experiments over Vermont and NY city maps.

			LEXGO$^*_{\text{dr}}$								
			0.75				0.5				k_1
	n	NAMOA$^*_{\text{dr}}$	0.75	0.5625	0.375	0.1875	0.5	0.375	0.25	0.125	k_2
Vermont	1	**9.29**	**4.71**	**4.71**	**6.31**	13.90	**1.35**	**1.42**	**1.77**	**2.20**	
	2	**4.64**	**2.49**	**3.01**	**2.60**	**2.27**	**2.44**	**2.05**	**1.90**	**1.17**	
	3	**0.45**	**0.42**	0.64	**0.37**	**0.20**	**0.20**	**0.20**	**0.12**	**0.10**	
	4	**47.53**	**43.72**	**46.94**	**31.79**	**22.10**	**18.42**	**8.65**	**2.97**	**5.17**	
	5	**2.04**	**1.82**	**1.46**	**0.90**	**0.70**	**0.59**	**0.29**	**0.34**	**0.53**	
	6	**259.05**	278.64	266.84	**184.47**	**58.46**	**175.31**	**211.47**	**93.07**	281.11	
	7	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	
	8	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	
	9	**2.46**	**2.15**	**1.91**	**1.56**	**0.99**	**1.37**	**0.90**	**0.78**	**0.60**	
	10	**246.09**	258.70	**200.35**	**134.56**	**26.67**	**117.81**	**81.26**	**57.48**	**146.34**	
	11	**527.56**	**427.56**	**418.81**	**394.05**	**264.59**	**401.99**	**395.54**	**313.09**	**209.02**	
	12	**4.71**	**1.96**	**1.91**	**1.77**	**0.90**	**0.60**	**0.45**	**0.64**	**0.45**	
	13	**395.24**	491.77	450.96	438.62	**144.61**	**332.82**	**304.70**	**189.07**	**17.80**	
	14	**1.82**	1.87	**1.52**	**0.99**	**0.31**	**0.95**	**0.46**	**0.07**	**0.07**	
	15	**33.35**	**30.60**	**28.50**	**24.96**	**26.45**	**23.35**	**22.55**	**22.51**	**15.84**	
	16	**4.77**	**1.90**	**2.12**	**2.65**	**2.13**	**1.70**	**1.73**	**1.54**	**1.79**	
	17	**51.26**	**43.24**	**43.50**	**35.50**	**25.83**	**14.90**	**10.09**	**1.41**	**1.63**	
	18	**0.42**	**0.24**	**0.23**	**0.21**	**0.39**	**0.32**	**0.17**	**0.17**	**0.17**	
	19	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	
	20	**102.57**	**80.15**	**92.57**	**84.66**	**97.32**	**53.52**	**36.50**	**31.65**	**38.93**	
NY city	2	**0.71**	**0.65**	**0.60**	**0.42**	**0.15**	**0.28**	**0.23**	**0.24**	**0.25**	
	4	**149.30**	198.18	220.77	225.99	**128.49**	176.04	177.32	170.47	**35.67**	
	5	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	**<0.01**	
	11	**1,452.58**	1,668.50	1,617.41	2,653.90	2,852.52	**1,366.86**	1,860.42	2,727.67	4,532.50	
	16	**106.89**	116.22	**86.22**	**31.18**	**12.69**	**62.97**	**77.39**	**86.87**	**84.35**	
	19	**8,355.32**	10,960.56	15,407.65	20,771.40	**5,270.44**	10,005.21	12,456.02	**5,141.12**	**4,519.28**	

7.4.2 Analysis on class II experiments

Regarding the second class of problems, unlike LEXGO$^*_{\text{lex}}$ and LEXGO$^*_{\text{lin}}$ with their NAMOA* counterparts, when applying the t-discarding technique to LEXGO* and NAMOA*, LEXGO$^*_{\text{dr}}$ does not perform always faster than NAMOA$^*_{\text{dr}}$ when k_2 values become stricter, in fact, the runtimes eventually increase notably although the number of label expansions decrease. This is due to the t-discarding method only applies to LEXGO$^*_{\text{dr}}$ when goals can be satisfied.

Table 7.23 displays the runtimes in seconds of NAMOA$^*_{\text{dr}}$ and LEXGO$^*_{\text{dr}}$ for the Vermont and New York city maps experiments. Table 7.24 displays a summary of the average runtime of the full set of Vermont map experiments. On one hand, LEXGO$^*_{\text{dr}}$ outperforms in average NAMOA$^*_{\text{dr}}$ in the Vermont map experiments, on the other hand, for the two most difficult problems solved by both algorithms, NY#11 and NY#19, LEXGO$^*_{\text{dr}}$ only runs faster than NAMOA$^*_{\text{dr}}$ in three of the sixteen cases available.

7.4.3 Summary

The results obtained for road map problems are significantly different from the experiments conducted on grid problems. LEXGO$^*_{\text{dr}}$ does not always perform faster than NAMOA$^*_{\text{dr}}$ when goals cannot be satisfied, in fact, its runtime in problems with unsatisfiable goals can be significantly higher. In addition, LEXGO$^*_{\text{dr}}$ may perform slower

Table 7.24. Class II experiments in road maps, average runtimes in seconds of $\text{NAMOA}^*_{\text{dr}}$ and $\text{LEXGO}^*_{\text{dr}}$ for the set of Vermont map experiments.

	$\text{LEXGO}^*_{\text{dr}}$								
	0.75				0.5				k_1
$\text{NAMOA}^*_{\text{dr}}$	0.75	0.5625	0.375	0.1875	0.5	0.375	0.25	0.125	k_2
84.66	83.59	78.30	67.31	34.40	57.38	53.93	35.93	36.15	

when the number of expansions is reduced, due to the fact that the t-discarding method may not be applied to prune and filter new scanned labels.

The average of the full set of Vermont problems gives $\text{LEXGO}^*_{\text{dr}}$ an advantage over $\text{NAMOA}^*_{\text{dr}}$ for all the analyzed class II experiments and most of the class I experiments. $\text{NAMOA}^*_{\text{dr}}$ outperforms $\text{LEXGO}^*_{\text{dr}}$ only when $k_1 = 1$ and $k_1 = 0.25$. The former case is obvious, since all goal-optimal solution vectors are also Pareto-optimal, therefore, $\text{LEXGO}^*_{\text{dr}}$ only introduces extra calculations to the process of returning the full Pareto set. The latter, it corresponds to an extreme case of bad performance of $\text{LEXGO}^*_{\text{dr}}$. When $k_1 = 0.25$ and the difficulty of the problem grows, the impact of the regular pruning and filtering on the runtime highly affects $\text{LEXGO}^*_{\text{dr}}$ performance. Thus, we can also observe than $\text{NAMOA}^*_{\text{dr}}$ was capable of solving 14 out of the 20 problems of New York city, while $\text{LEXGO}^*_{\text{dr}}$ solved only 6. This can only be attributed to the regular pruning and filtering operations, highly inefficient in time compared to t-discard alternatives.

7.5 Summary on road map problems

The experiments previously presented are a new contribution to the multicriteria route planning. To the best of our knowledge, these are the largest problems solved with 3 objectives.

The main contributions of this thesis are LEXGO^*, a new devised algorithm to cope with lexicographic goals on MSP, the t-discarding method, which applied to MSP achieves a significant runtime performance improvement, and the application of the t-discarding method to NAMOA^* and LEXGO^* within a broader set of experiments that were previously reported in the literature.

Previous results to the experiments shown in this thesis were reported in Pulido et al. (2015). The experiments reported in this chapter go beyond those previously reported and add $\text{LEXGO}^*_{\text{lin}}$ and $\text{LEXGO}^*_{\text{dr}}$ algorithms to the analyses. Furthermore, additional cases of LEXGO^* has been evaluated, for instance, $k_1 = 0.75$ in class II experiments.

The empirical evaluation conducted in this chapter over realistic road map problems confirms the previous results over random grids to some extent. LEXGO^* has been formally and empirically proved to be more efficient for the majority of the analyzed cases than NAMOA^*, regardless the selection order and the road map.

The t-discarding method applied to NAMOA^* and LEXGO^* is also proved formally and now empirically to improve significantly the performance of these algorithms.

Finally, the comparison between the best alternatives, $\text{NAMOA}^*_{\text{dr}}$ and $\text{LEXGO}^*_{\text{dr}}$, allow us to differentiate those cases where $\text{LEXGO}^*_{\text{dr}}$ may be chosen over $\text{NAMOA}^*_{\text{dr}}$.

Table 7.25. Runtime comparison - summary table for road map experiments.

Comparison	Results
NAMOA$^*_{\text{lin}}$ vs NAMOA$^*_{\text{lex}}$	In contrast to the experiment results of random grids, where NAMOA$^*_{\text{lin}}$ outperforms NAMOA$^*_{\text{lex}}$ by a factor of two, NAMOA$^*_{\text{lin}}$ has only a slight performance advantage on road maps.
LEXGO$^*_{\text{lin}}$ vs LEXGO$^*_{\text{lex}}$	In a slightly different way to the experiment results for random grids, LEXGO$^*_{\text{lin}}$ also obtain better runtimes than LEXGO$^*_{\text{lex}}$, although the difference between them is significantly smaller on road maps.
LEXGO* vs NAMOA*	The selection order does not make a difference in the comparative performance between LEXGO* and NAMOA* (in contrast to the experiment results for random grids). The speed-up of LEXGO* over NAMOA*, however, remains outstanding for $k_1 \leq 0.5$, and its time overhead for $k_1 = \{0.75, 1\}$ is practically zero.
NAMOA$^*_{\text{dr}}$ vs NAMOA*	NAMOA$^*_{\text{dr}}$ is almost two orders of magnitude faster than NAMOA$^*_{\text{lex}}$ and NAMOA$^*_{\text{lin}}$ for the most difficult problems. Moreover, NAMOA$^*_{\text{dr}}$ extends the size of the three objective problems that can be solved in practice.
LEXGO$^*_{\text{dr}}$ vs LEXGO*	In a similar way to the results for random grids, LEXGO$^*_{\text{dr}}$ achieves speed-ups of up to 50 times faster than any version of LEXGO* with standard discarding procedures and goals that can be satisfied. When goals can not be satisfied, LEXGO$^*_{\text{dr}}$ does have a slight advantage of performance. Its relative improvement also grows with problem difficulty.
LEXGO$^*_{\text{dr}}$ vs NAMOA$^*_{\text{dr}}$	The comparative performance of LEXGO$^*_{\text{dr}}$ over NAMOA$^*_{\text{dr}}$ is significantly different in road map problems than in random grids. NAMOA$^*_{\text{dr}}$ outperforms LEXGO$^*_{\text{dr}}$ when $k_1 = 1$ and $k_1 = 0.25$. The latter corresponds to an extreme case of bad performance for LEXGO*, since the standard pruning and filtering procedures highly impact the runtime performance. LEXGO$^*_{\text{dr}}$ outperforms NAMOA$^*_{\text{dr}}$ in the rest of the cases, i.e. when $k_1 = \{0.75, 0.5, 0\}$ in class I experiments and all class II experiments.

Part III

Conclusions

The last part of this thesis gathers the conclusions of this research work.

Chapter 8 sums up the conclusions collected from this research work into nine different points, deducted from the formal and empirical analyses conducted in the second part. Furthermore, some lines for future developments and improvements are also suggested.

Chapter 8
Conclusions and Future Work

Time is really the only capital that any human being has and the thing that he can least afford to waste or lose.

Thomas Edison (1847-1931)

Multicriteria Decision Making (MCDM) is a branch of Operations Research that considers multiple criteria in decision-making environments. Goal Programming is a branch of MCDM that models preferences using goals, i.e. establishing target values over a set of attributes. The Multicriteria Search Problem (MSP) is the natural extension for the multiobjective case of the Shortest Path Problems, which are one of the most extensively studied problems in Artificial Intelligence and Operations Research. New developments in these fields are of practical importance in current research.

The main goals of this thesis are to analyze the use of goal-based preferences in Multicriteria Shortest Path Problems, and provide new techniques and algorithms framed in this field. In particular, we are concerned with exact techniques where the preferences of the decision maker (DM) are modeled into levels of pre-emptive importance. Within each level, one or more targets are defined as the expectations of the DM and weights characterize the importance of the criteria.

In the literature, there is a general agreement that labeling multicriteria search algorithms are frequently the best choice to solve MSP (Skriver & Andersen, 2000; Raith, 2009; Raith & Ehrgott, 2009). Moreover, two strategies can be used in these algorithms, node-selection and label-selection, and the latter has been repeatedly proved to be more efficient in label-setting algorithms (Paixão & Santos, 2013; Pérez de la Cruz et al., 2013). Furthermore, the use of lower bound functions has been shown as a great advantage in the efficiency of these algorithms (Mandow & Pérez de la Cruz, 2010; Machuca, 2012). Thus, under the framework of lexicographic goal-based preferences we have analyzed two different strategies. The first one is to calculate the whole set of efficient solutions to the problem and extract the satisfactory solutions a posteriori. NAMOA* is included in these algorithms. The second one is to employ the goals given by the DM to bound the area of interesting solutions. The main goal of our new devised algorithm, LEXGO*, which falls in this second category, is to achieve improvements over the efficiency of NAMOA* in the calculation of this set of efficient

solutions according to the goals.

Time has been shown as the limiting factor in the performance of multicriteria search algorithms. The majority of the time effort is devoted to check dominance against set of labels and thereby discard new generated alternatives. Our new proposed dimensionality reduction technique, called t-discarding, speeds up the processes of pruning and filtering, which are responsible of discarding new paths by comparing to partial and full paths, respectively.

The new dimensionality reduction technique can be used to improve the time requirements of labeling multicriteria search algorithms. In particular, we introduced $\text{NAMOA}^*_{\text{dr}}$ and $\text{LEXGO}^*_{\text{dr}}$ as the versions of the algorithms previously analyzed that employ the dimensionality reduction technique. Finally, we perform extensive benchmarking to all the algorithms proposed over two scenarios, random grids and road maps problems.

In summary, a new exact label-setting multicriteria search algorithm is proposed to deal with MSP with lexicographic goal-based preferences and improve the performance of the full search. In addition, a new technique to speed up the processes of discarding new alternatives is also proposed, as well as formally and practically applied. Finally, empirical analyses test all the algorithmic alternatives.

8.1 Conclusions

The main conclusions of this doctoral dissertation can be enumerated as follows:

1. **LEXGO*, a new multicriteria search algorithm has been proposed.** Multiobjective search algorithms benefit from the principle of optimality, i.e. an optimal path is made up of optimal subpaths. Regrettably, this property does not hold for lexicographic goal-based preferences. We have introduced a new exact label-setting algorithm that returns the subset of Pareto-optimal paths that satisfy a set of lexicographic goals grouped in pre-emptive priority levels, or the subset that minimizes deviation from goals if these cannot be fully satisfied. Along with LEXGO*, it has been proposed a specific pruning condition that allows to reduce the number of paths explored in goal-based search. We have also provided formal proofs on the correctness of the new pruning procedure and LEXGO*.

2. **LEXGO* is more efficient than a full Pareto search.** LEXGO* is theoretically proved to expand a subset of the labels expanded by the full Pareto search. Since the number of expanded labels is not the only relevant factor in the analysis of multicriteria search algorithms, we compared LEXGO* and NAMOA* over random grids and road maps problems. In both cases, a small time overhead can be observed for LEXGO* when the full Pareto frontier satisfies the goals, and the relative performance of LEXGO* over NAMOA* improves progressively whenever the set of efficient solutions is reduced. Elsewhere in the test sets, LEXGO* achieves important reductions in time requirements of about one order of magnitude when goals are satisfied for some Pareto solutions, and up to four orders of magnitude whenever goals can not be fully satisfied.

3. **The label selection policy has an important impact on time performance.** We have conducted experiments with two different label selection policies, lexicographic and linear aggregation, over NAMOA* and LEXGO*. It can be observed that NAMOA$^*_{\text{lin}}$ is approximately 50% faster than NAMOA$^*_{\text{lex}}$/ in random grids. The comparison between LEXGO$^*_{\text{lex}}$ and LEXGO$^*_{\text{lin}}$ is however, not that straightforward. LEXGO$^*_{\text{lin}}$ is only faster than LEXGO$^*_{\text{lex}}$ when a high percentage of the Pareto frontier is returned, and slower whenever a reduced number of solutions satisfy all goals. When goals cannot be satisfied, LEXGO$^*_{\text{lex}}$ and LEXGO$^*_{\text{lin}}$ show a very similar performance.

4. **A new dimensionality reduction technique, t-discarding, is proposed.** Time rather than space is the limiting factor to increase the number of Multiobjective Search Problems that can be practically solved. Thus, we have devised a simple but very effective technique to decrease the number of labels belonging to the sets in charge of discarding new alternatives. This technique is applied under reasonable circumstances: when a consistent lower bound function is employed along with the algorithm and a lexicographic order is used to select alternatives from OPEN.

5. **t-discarding is extensively used to discard labels.** We employ sets of closed labels, $G_{cl}(n)$, and full solution paths, COSTS, to perform pruning (cl-pruning) and filtering processes, respectively. Nonetheless, sets of partial open labels, $G_{op}(n)$, can not benefit from this technique to speed up dominance checks (op-pruning). Therefore, we launched experiments to measure the percentage of op-pruned, cl-pruned and filtered labels over the total number of discarded labels. Neither in the experiments over random grids nor in the experiments over road maps op-pruning was employed in more than 10% of the cases. Then, we can ensure t-discarding is highly applied to discard new alternatives.

6. **t-discarding checks dominance against reduced sets of labels.** Relative size of the sets of truncated labels over the original sets is theoretically unknown. However, in practice, an experimental evaluation was conducted and this size was around two orders of magnitude smaller for the difficult grid problems and about two to three orders for difficult road maps problems. Moreover, the relative advantage of t-discarding is incremented gradually with problem size, hence, this new contribution can effectively extends the set of Multicriteria Search Problems that can be practically solved.

7. **NAMOA$^*_{\text{dr}}$ reduces NAMOA* time requirements.** The t-discarding technique can be applied to exact multicriteria search algorithms like NAMOA*. In random grid problems, the speed-up for the most difficult problems was 8.94 by NAMOA$^*_{\text{dr}}$ over NAMOA$^*_{\text{lin}}$, the fastest studied version of NAMOA*. In road maps problems, the speed-up for the most difficult problem solved by NAMOA$^*_{\text{lin}}$ was 26.55, i.e. NAMOA$^*_{\text{dr}}$ solved the problem in 3.77% of the time needed by NAMOA$^*_{\text{lin}}$. NAMOA$^*_{\text{dr}}$ also extended the number of solved problems from 4 out of 20 by NAMOA$^*_{\text{lin}}$ to 14 out of 20 in a very difficult set of realistic problems with three criteria.

8. **LEXGO$^*_{\mathbf{dr}}$ reduces standard LEXGO* time requirements in some cases.** The t-discarding technique requires the lexicographical selection of evaluation vectors. This imposed lexicographic order in LEXGO$^*_{\text{dr}}$ makes those cases where the goals cannot be satisfied incompatible with the correctness of t-discarding. However, LEXGO$^*_{\text{dr}}$ can benefit from the use of t-discarding when goals can be satisfied, which indeed are the most difficult problems, and achieve speed-ups of up to 8.31 and 21.72 for the most difficult random grid and road map problems, respectively.

9. **A comparative between NAMOA$^*_{\mathbf{dr}}$ and LEXGO$^*_{\mathbf{dr}}$.** Finally, we have conducted an experimental evaluation of the most successful alternatives studied. In random grids, the inclusion of the t-discarding technique makes NAMOA$^*_{\text{dr}}$ be the alternative to choose whenever goals can be satisfied, specially, when a higher percentage of the non-dominated solutions satisfy the goals. LEXGO$^*_{\text{dr}}$ is the best alternative whenever goals can not be fully satisfied or a smaller portion of the Pareto frontier is returned.

   In road maps problems, the results are compatible but significantly different. It must be pointed out that $k_1 = 0.25$ represents an extreme case of bad performance for LEXGO$^*_{\text{dr}}$. Thus, in class I experiments, NAMOA$^*_{\text{dr}}$ outperforms LEXGO$^*_{\text{dr}}$ when $k_1 = 1$ and $k_1 = 0.25$, and LEXGO$^*_{\text{dr}}$ has a better performance when $k_1 = \{0.75, 0.5, 0\}$. In class II experiments, the majority of LEXGO$^*_{\text{dr}}$ experiments perform faster than NAMOA$^*_{\text{dr}}$ experiments. When goals could be satisfied there existed a significant number of problems that were solved by NAMOA$^*_{\text{dr}}$ and could not be solved within the time limit by LEXGO$^*_{\text{dr}}$.

8.2 Future Work

This doctoral dissertation has contributed new algorithms and techniques. These contributions have also raised new future lines of research where current developments can be effectively improved. The following lines may justify further investigation:

- The experimental evaluation conducted in this thesis has remarked the importance of the label selection policy for time performance. The determination of an optimal policy is a desirable issue for a deeper study.

- The study and development of more efficient data structures to store the sets of open and closed labels, as well as the queue of alternatives, is a current research. All the algorithms presented in this thesis could benefit from more efficient implementations of these data structures.

- Two important formal analyses could be carried out regarding the optimality of LEXGO* when used with consistent lower bounds. The first one, research the possibility that LEXGO* is optimal in its class of exact goal-based algorithms according to the number of labels expanded. The second one, investigate theoretically the possibility that LEXGO* expands an equal or smaller number of labels when using more informed lower bound functions.

- LEXGO* can be extended to use other GP models or other formulas to measure the deviation from goals. New pruning and/or filtering rules will probably need to be developed for each particular case, in order to enhance their efficiency.

- Apply t-discarding to other multicriteria search algorithms with lower bounds. We have applied the t-discarding technique to NAMOA*, but it can be applied to a significant number of algorithms to reduce their time requirements.

- Both NAMOA* and LEXGO* return the set of all non-dominated (or goal-optimal for LEXGO*) solutions. In difficult problems seek for the whole set of non-dominated solutions leads to high runtime requirements. However, there are other multicriteria techniques that seek only for a single efficient solution, as Compromise Search. Some of the conclusions reported here, like the performance of selection orders or the importance of dominance checks in runtime, can be useful to extend other multicriteria decision models.

- Multicriteria search in road maps is becoming more popular in the last few years. Since the query time for single-objective shortest path problems is in the order of microseconds, the natural evolution is the resolution of problems that involve more than one criterion. The advanced optimization techniques applied to the single-objective problem, like multilevel graphs (Schulz et al., 2002) or contraction hierarchies (Geisberger et al., 2008) represent possibilities for multicriteria search which deserve further research. For instance, preprocessing techniques like contraction hierarchies would require a multicriteria bidirectional search. In fact, this is another field of research of the author (Pulido et al., 2011, 2012), hence, its application is of his great interest.

- Finally, a recurrent line of research is to identify new potential domains to apply multicriteria search, as well as the combination with other disciplines to approach MSP from a different perspective.

Part IV

Appendix

Appendix A
Resumen

Si supiese qué es lo que estoy haciendo, no lo llamaría investigación, ¿verdad?

Albert Einstein (1879-1955)

El problema del camino más corto (SPP) es uno de los más antiguos y extensamente estudiados en los campos de Inteligencia Artificial (AI) e Investigación Operativa (OR), el cual consiste en encontrar el camino entre dos nodos de un grafo tal que la suma de los pesos de los arcos que lo componen sea mínima. Sin embargo, los problemas en la vida real suelen implicar múltiples, y normalmente contradictorios, criterios. Cuando múltiples objetivos deben ser optimizados simultáneamente el concepto de una sola solución óptima pierde su validez, en su lugar, un conjunto de soluciones eficientes o Pareto-óptimas definen el equilibrio óptimo entre los objetivos bajo consideración.

El problema de búsqueda multicriterio es la extensión natural del problema del camino más corto cuando se consideran múltiples criterios. El problema de búsqueda multicriterio es computacionalmente más complejo que el que involucra un solo criterio. El número de expansiones de etiquetas puede crecer exponencialmente con la profundidad de la solución, incluso para el caso de dos objetivos (Hansen, 1980). Con el supuesto de costes enteros acotados y un número fijo de objetivos el problema se convierte en tratable para grafos de tamaño polinomial (por ejemplo, véase (Mandow & Pérez de la Cruz, 2009; Müller-Hannemann & Weihe, 2006)).

Un gran variedad de aplicaciones prácticas en diferentes campos pueden ser abordadas como problemas de búsqueda multicriterio, como la planificación de la trayectoria de robots (Wu et al., 2011), el transporte de materiales peligrosos (Caramia et al., 2010), la planificación de rutas en diferentes contextos (Jozefowiez et al., 2008), el transporte público (Raith, 2009) o la calidad del servicio en redes (Craveirinha et al., 2009).

La programación por metas es una de las técnicas de decisión multicriterio más exitosas utilizadas en la optimización de metas multiobjetivo. En esta tesis aplicamos una de sus variantes al problema de búsqueda multicriterio. Así, nuestro objetivo es resolver el problema de búsqueda multicriterio con preferencias lexicográficas basadas en metas. Para ello, proponemos un nuevo algoritmo llamado LEXGO*, un algoritmo exacto de etiquetado que devuelve el subconjunto de caminos óptimos de Pareto que

satisfacen un conjunto de metas lexicográficas, o el subconjunto de mínima desviación con respecto a las metas si estas no se pueden satisfacer completamente. Adicionalmente, se demuestran la admisibilidad de LEXGO* y la propiedad de expandir sólo un subconjunto de las etiquetas expandidas por un algoritmo de búsqueda completa multicriterio.

Puesto que los requisitos temporales en lugar de los espaciales son el factor limitante para el rendimiento de los algoritmos de búsqueda multicriterio, proponemos una nueva técnica, llamada *t-discarding*, para disminuir el número y dimensionalidad de las comprobaciones de dominancia durante la búsqueda. La aplicación del *t-discarding* a los algoritmos previamente estudiados, NAMOA* y LEXGO*, da lugar a dos nuevos algoritmos eficientes en tiempo, $\text{NAMOA}^*_{\text{dr}}$ y $\text{LEXGO}^*_{\text{dr}}$, respectivamente.

Todas las alternativas algorítmicas han sido testadas en dos escenarios, mallas aleatorias y mapas de carreteras. La evaluación experimental muestra la efectividad de LEXGO* en ambos bancos de prueba, así como reducciones espectaculares en los requisitos temporales de ambos algoritmos con respecto a sus contrapartidas que utilizan las técnicas de comprobación de dominancia estándar.

A.1 Objetivos

Los principales objetivos de esta tesis doctoral pueden ser esbozados de la siguiente manera:

1. **Abordar el problema de búsqueda multicriterio con metas.** El primer objetivo de esta tesis es presentar la descripción del problema de búsqueda multicriterio y concretamente, el problema de búsqueda multicriterio con preferencias basadas en metas. Describiremos dos enfoques para abordar este problema.

2. **Desarrollar un nuevo algoritmo para preferencias basadas en metas lexicográficas.** El principio de optimalidad se cumple para los problemas multiobjetivo del camino más corto, pero desafortunadamente no se cumple para las preferencias basadas en metas lexicográficas. Para abordar el problema de búsqueda multicriterio con un algoritmo específicamente diseñado para metas, concentraremos nuestros esfuerzos en el desarrollo de un nuevo algoritmo basado en una política de selección de etiquetas.

3. **Probar formalmente la corrección y eficiencia de este nuevo algoritmo.** Otro objetivo de esta tesis es complementar el nuevo algoritmo desarrollado con análisis formales de su corrección, así como su eficiencia con respecto a un algoritmo óptimo que realiza una búsqueda completa multiobjetivo.

4. **Estudio de posibles mejoras a algoritmos multiobjetivo del camino más corto.** Nuestros esfuerzos también se centrarán en general en la mejora de algoritmos multiobjetivo del camino más corto. De manera más específica, nuestro objetivo es proponer una nueva técnica para mejorar el rendimiento en tiempo de ejecución de algoritmos de etiquetado multiobjetivo del camino más corto con límites superiores consistentes.

5. **Demostrar formalmente la corrección de la técnica eficiente en tiempo.** Desarrollaremos formalmente la aplicación de la técnica eficiente en tiempo al nuevo algoritmo propuesto con preferencias basadas en metas, así como la aplicación a NAMOA*. En particular, demostraremos teóricamente la corrección de ambos algoritmos empleando la técnica de reducción de la dimensionalidad.

6. **Realizar una evaluación empírica de todas las alternativas algorítmicas propuestas.** Finalmente, el último objetivo de esta tesis es proporcionar una evaluación extensiva de todos los algoritmos propuestos. Para ello, emplearemos bancos de pruebas generados aleatoriamente y escenarios realistas.

A.2 Contribuciones

Las contribuciones de esta tesis se enumeran de forma resumida a continuación,

1. **Descripción del problema de búsqueda multicriterio basado en metas.** En primer lugar, describimos la técnica de programación por metas dentro de la disciplina de la teoría de la decisión, así como la búsqueda en grafos multicriterio dentro de los problemas de optimización multiobjetivo. Dado este marco de trabajo, describiremos y definiremos formalmente el problema de búsqueda en grafos multicriterio basados en metas y los diferentes enfoques algorítmicos para abordarlo.

2. **Un nuevo algoritmo de búsqueda multicriterio.** Presentamos LEXGO*(A^* con metas lexicográficas), un nuevo algoritmo exacto de etiquetado multicriterio con preferencias basadas en metas. LEXGO* devuelve el subconjunto de caminos óptimos de Pareto que satisfacen un conjunto de metas lexicográficas, o el subconjunto de caminos con menor desviación con respecto a estas metas si no pueden ser completamente satisfechas.

3. **Caracterización formal de la admisibilidad y eficiencia de LEXGO*.** Demostramos teóricamente la admisibilidad de LEXGO*, es decir, LEXGO* es un algoritmo exacto y devuelve el conjunto completo de soluciones a un problema, así como su eficiencia, es decir, el número de etiquetas expandidas por el algoritmo decrece con límites superiores más informados con respecto a una búsqueda multicriterio completa.

4. **Proponemos la técnica *t-discarding*.** Presentamos una nueva técnica que reduce los requisitos temporales de los algoritmos de búsqueda multiobjetivo. Esta técnica es aplicable a algoritmos de etiquetado multicriterio basados en A^*. Su clave reside al comprobar la dominancia de un conjunto de etiquetas sobre una sola etiqueta. En lugar de realizar las comprobaciones de dominancia teniendo en cuenta n criterios, lo haremos reduciendo todos los vectores implicados a $n-1$ criterios. Además, cuando uno de los criterios no se considera, el tamaño de los conjuntos de vectores no dominados también se reduce drásticamente. Así podemos usar conjuntos reducidos de vectores truncados en lugar de los originales para descartar alternativas. Todo esto es posible gracias a las propiedades formales de las funciones consistentes de cotas inferiores.

5. **Proporcionar la caracterización formal de la técnica *t-discarding*.** Aplicamos t-discarding a NAMOA* y LEXGO*, y presentamos dos nuevos algoritmos, NAMOA$^*_{\text{dr}}$ y LEXGO$^*_{\text{dr}}$. Probaremos la corrección de NAMOA$^*_{\text{dr}}$ y LEXGO$^*_{\text{dr}}$, evaluaremos su eficiencia y analizaremos su rendimiento.

6. **Evaluación empírica de las nuevas contribuciones.** Dos escenarios principales van a ser usados para comprobar la efectividad de nuestros nuevos algoritmos: mallas aleatorias y problemas realistas en mapas de carreteras. En ellos, presentamos reducciones importantes de los requisitos temporales para problemas con tres objetivos. Con nuestro conocimiento a día de la presentación de esta tesis, los resultados mostrados en mapas de carreteras representan los problemas de búsqueda con tres objetivos más difíciles resueltos hasta la fecha.

A.3 Resumen de los capítulos de la Tesis

A continuación, se presenta un resumen de cada capítulo de esta tesis. Los Capítulos 1 al 3 se integran dentro de la parte dedicada a la motivación y fundamentos de esta tesis, los Capítulos 4 al 7 dentro de las contribuciones y finalmente, el Capítulo 8 muestra las conclusiones y lineas futuras de esta investigación. Puesto que ya hemos hablado en las secciones anteriores de los objetivos y contribuciones expuestos en el primer capítulo de esta tesis, comenzamos este resumen en el segundo capítulo y hablaremos sobre las conclusiones y líneas futuras de trabajo en las dos siguiente secciones.

A.3.1 Búsqueda Multicriterio en Grafos: Problemas y Algoritmos

El Capítulo 2 de esta tesis presenta el problema de Búsqueda Multicriterio con preferencias lexicográficas basadas en metas. Para ello, se introduce el popular método de Programación por Metas así como sus variantes en la Sección 2.3. Concretamente, la Programación por Metas es una de las técnicas más exitosas de Decisión Multicriterio. Rosenthal (1983) categorizó en tres las formulaciones disponibles de la Programación por Metas. En esta tesis, ponemos énfasis en la variante que agrupa los criterios en niveles de prioridad definidos por el decisor. Dentro de cada nivel, las metas proporcionadas también tienen pesos, que establecen la importancia de cada criterio dentro del nivel.

El problema de Búsqueda Multicriterio ha sido estudiado extensivamente por las comunidades de Inteligencia Artificial e Investigación Operativa. En la Sección 2.5 de este capítulo se proporcionan algunas definiciones relevantes y la formulación del problema. También se enumeran algunos de sus campos de aplicación, como son la vigilancia robotizada (Delle Fave et al., 2009), la planificación de la trayectoria de robots (Wu et al., 2011), el transporte de materiales peligrosos (Caramia et al., 2010), la planificación de satélites (Gabrel & Vanderpooten, 2002), la planificación de rutas en diferentes contextos (Jozefowiez et al., 2008), el transporte público (Raith, 2009) o la calidad del servicio en redes (Craveirinha et al., 2009). Por último, se repasan las propiedades formales de las funciones multicriterio de límites inferiores.

En la Sección 2.5 se enumeran las tres categorías de algoritmos de búsqueda multicriterio (Clímaco & Pascoal, 2012). Nos centramos en dos, en primer lugar, los

algoritmos *a priori* son aquellos en que las preferencias del decisor son proporcionadas con anterioridad, y en segundo lugar, los algoritmos *a posteriori* son aquellos que no poseen ningún tipo de información del decisor con anterioridad a la ejecución del algoritmo, por lo que recuperan todas las soluciones eficientes para que sea el decisor con posterioridad quien tome la decisión.

Dentro de este sección también se describirá el algoritmo NAMOA*(Mandow & Pérez de la Cruz, 2010), que se engloba en la categoría de algoritmos *a posteriori*, y que se empleará en esta tesis como referencia para evaluar las posteriores contribuciones. Además describiremos la función de cotas inferiores propuesta por Tung & Chew (1992), que emplearemos en todos los algoritmos estudiados en esta tesis.

A.3.2 Bancos de pruebas para búsqueda multicriterio

El Capítulo 3 está dividido en tres secciones diferenciadas. En primer lugar, en la Sección 3.1 se revisan, sin entrar en un gran grado de detalle, los bancos de pruebas usados en la literatura para evaluar el rendimiento de la búsqueda multicriterio. Complementariamente, se enuncian los dos factores claves en la evaluación del rendimiento de algoritmos, la reproducibilidad y la equidad de las pruebas, así como los diversos factores que pueden influir en los resultados de la evaluación, como el tamaño y forma del grafo de búsqueda, número de arcos, profundidad de la solución o el número de criterios, por nombrar algunos.

En la Sección 3.2 se describen en detalle los bancos de pruebas empleados en esta tesis doctoral. Primeramente, se detallan los parámetros de generación de los bancos de pruebas basados en mallas aleatorias, como el número de nodos y arcos, las profundidades de solución consideradas o el número de costes eficientes resultantes. Posteriormente, se detallan los bancos de pruebas sobre mapas de carreteras. En concreto, se han empleado veinte problemas aleatorios sobre los mapas de Nueva York y Vermont, con el objetivo de minimizar alternativamente la distancia, tiempo y coste del camino planificado. El mapa de Nueva York corresponde al 9th Dimacs Implementation Challenge: Shortest Path[1], un banco de pruebas basado en datos reales de los mapas de carreteras de Estados Unidos, donde los arcos representan carreteras y los nodos las intersecciones entre carreteras. El mapa de Vermont fue obtenido del UA Census 2000 TIGER/Line y está disponible en el mismo sitio web.

Para concluir, en la Sección 3.3 se define la importancia de los bancos de pruebas utilizados en esta tesis con respecto a los empleados en la literatura. Tanto por lo que se refiere a los bancos de pruebas sobre mallas, como a los bancos de pruebas sobre mapas de carreteras, nuestra evaluación experimental es de una dificultad lo suficientemente elevada como para considerarse adecuada en este contexto.

A.3.3 Contribuciones

El Capítulo 4 de esta tesis doctoral hace un revisión de las contribuciones aportadas por este trabajo. LEXGO*, nuestra proposición algorítmica para resolver problemas de búsqueda multicriterio basados en metas lexicográficas, es presentado en la Sección 4.1. Esta sección incluye las condiciones especiales de poda y filtrado de LEXGO*,

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

así como un ejemplo de uso del algoritmo. Aquí también se establece la definición de solución eficiente con respecto a un conjunto de metas lexicográficas agrupadas en niveles de prioridad, así como una nueva regla de poda para descartar etiquetas por su desviación con respecto a estas metas.

Por otro lado, otra de las grandes contribuciones de esta tesis se describe en este capítulo. La Sección 4.2 muestra la técnica t-discarding para acelerar las comprobaciones de dominancia de un conjunto de etiquetas. Esta técnica ya fue empleada para mejorar los requisitos espaciales de la búsqueda frontera multicriterio, y en este trabajo la consideramos para mejorar considerablemente los requisitos temporales de los algoritmos de búsqueda multicriterio, como veremos en los capítulos de evaluación experimental.

Las dos últimas secciones de este capítulo repasan la aplicación de la técnica de t-discarding a los algoritmos NAMOA* y LEXGO*, dando lugar a los algoritmos, NAMOA$^*_{\text{dr}}$ y LEXGO$^*_{\text{dr}}$, respectivamente.

A.3.4 Análisis formal de los algoritmos de búsqueda multicriterio

El principal propósito del Capítulo 5 de esta tesis es proporcionar un análisis formal de los algoritmos ideados como resultado de esta tesis, además de encontrar la mejor alternativa algorítmica cuando las preferencias de un decisor son modeladas basándose en unas metas de satisfacción y estas son ordenadas en niveles de prioridad con pesos dentro de cada nivel.

Dos posibilidades surgen ante tal problema, la primera y más obvia, es el cálculo de todo el conjunto de soluciones eficientes o no dominadas a través de un algoritmo de búsqueda multicriterio, como por ejemplo NAMOA*, y posteriormente extraer el subconjunto de soluciones que satisfacen esas metas. La segunda alternativa consiste en concentrar el esfuerzo de la búsqueda solo en aquellas soluciones que son óptimas con respectos a las metas, como se realiza por ejemplo en LEXGO*, es decir, descartar desde las primeras fases de la búsqueda aquellos caminos en el grafo que no nos llevarán a soluciones que sean eficientes con respecto a las metas.

La otra gran contribución de esta tesis, la técnica de t-discarding, puede ser aplicada a los procesos de poda y filtrado de los algoritmos nombrados anteriormente sin afectar su admisibilidad. Las propiedades de estos nuevos algoritmos, que utilizan la técnica de reducción de la dimensionalidad, son analizadas en este capítulo.

Las propiedades formales analizadas en los algoritmos de búsqueda multicriterio presentados en esta tesis son:

Admisibilidad Se prueba formalmente la admisibilidad de todos los algoritmos estudiados. Las pruebas formales de admisibilidad de NAMOA* fueron presentadas en Mandow & Pérez de la Cruz (2005); Mandow & Pérez de la Cruz (2010). Las propiedades que prueban la admisibilidad de LEXGO* son descritas en la Sección 5.2. Finalmente, la técnica de t-discarding empleada por NAMOA$^*_{\text{dr}}$ y LEXGO$^*_{\text{dr}}$ se analiza en la Sección 5.3.

Eficiencia El método estándar para medir la eficiencia de los algoritmos de búsqueda multicriterio es el número de etiquetas exploradas. Un estudio reciente muestra

que NAMOA* es óptimo con respecto a esta medida cuando se emplea una función consistente de límites inferiores (Mandow & Pérez de la Cruz, 2010). La misma propiedad puede ser aplicada a LEXGO*, puesto que siempre expande un subconjunto de las etiquetas expandidas por NAMOA*(ver Sección 5.2.1 para mayor detalle de esta demostración). En cuanto a los algoritmos que emplean la técnica de t-discarding, el conjunto de etiquetas expandidas por las versiones básicas y las que usan la técnica de la reducción de la dimensionalidad se demuestran equivalentes, puesto que las nuevas reglas de poda y filtrado descartan exactamente las mismas etiquetas que los procesos originales de poda y filtrado (para más detalles consultar el Teorema 5.9).

NAMOA* ha demostrado expandir un número igual o menor de etiquetas cuando se emplea un función consistente de límites inferiores (Mandow & Pérez de la Cruz, 2010). De manera equivalente por las razones explicadas anteriormente, LEXGO* y las variantes algorítmicas introducidas en esta tesis también comparten esta propiedad.

Adicionalmente, introducimos otras importantes medidas de rendimiento para los algoritmos de búsqueda multicriterio: el número de comparaciones de dominancia y la cardinalidad de los conjuntos de etiquetas no dominadas que se utilizan para descartar nuevas alternativas. Estos nuevos parámetros son la clave del impresionante rendimiento de los algoritmos que emplean t-discarding, y serán formalmente analizados en la Sección 5.3.2.

A.3.5 Evaluación empírica en mallas aleatorias

El Capítulo 6 está dividido en seis subsecciones y tiene como principal objetivo la evaluación de las alternativas algorítmicas propuestas sobre un escenario de problemas aleatorios en mallas. La Sección 6.1 describe la configuración de los experimentos. Por un lado, el estudio experimental analiza los siguientes aspectos para la comparación de NAMOA* y LEXGO* como función de la profundidad de la solución:

- El número de etiquetas expandidas.
- Los requisitos temporales.
- El porcentaje relativo de soluciones eficientes con respecto a las metas. El conjunto de soluciones devueltas es comparado con el tamaño completo del conjunto de soluciones no dominadas del problema.

Por otro lado, para analizar el rendimiento de la técnica de t-discarding, se analizan los siguientes aspectos como función de la profundidad de la solución:

- El tamaño relativo de los conjuntos de etiquetas truncadas con respecto a los conjuntos completos de NAMOA*.
- El porcentaje de etiquetas podadas y filtradas sobre el total de etiquetas descartadas.

- Los requisitos temporales de los algoritmos que emplean la técnica de t-discarding ($\text{NAMOA}^*_{\text{dr}}$ y $\text{LEXGO}^*_{\text{dr}}$) con respecto a sus algoritmos de referencia (NAMOA^* y LEXGO^*).

Los experimentos se dividen en dos clases, que consideran la minimización simultánea de tres criterios agrupados en dos niveles. Las metas se definen con respecto a los puntos óptimos y nadir del espacio de costes vectoriales. Así, la primera clase propone cinco tipos de experimentos donde las metas son gradualmente más flexibles, desde situarlas en el punto óptimo, hasta estar localizadas en el punto nadir, y en la que los dos niveles de prioridad comparten el mismo nivel de dificultad de las metas. En la segunda clase de experimentos, los valores de las metas del primer nivel se dejan fijos y se parametrizan gradualmente las metas correspondientes al segundo nivel.

Las Secciones 2 a 5 de este capítulo analizan las diferentes comparaciones entre algoritmos. Los resultados se pueden ver esquematizados en la Sección A.4. Por último, se realiza un resumen de todos los resultados obtenidos en esta evaluación experimental en la Sección 6.6.

A.3.6 Evaluación empírica en mapas de carreteras

El Capítulo 7 de esta tesis analiza el rendimiento en mapas de carreteras de los algoritmos estudiados. El banco de pruebas utilizado es el 9th Dimacs Implementation Challenge. Este concurso estaba compuesto de doce mapas de carreteras de diferentes tamaños.

Los mapas originales de DIMACS proporcionan dos criterios de coste diferentes: *distancia* y *tiempo de trayecto*. Un coste adicional fue propuesto en Machuca & Mandow (2011) calculando el *coste económico del trayecto*. Los experimentos contenidos en este capítulo consideran la minimización simultánea de estos tres atributos (consultar la Sección 3.2.2 para ver con mayor detalle la descripción de este banco de pruebas).

La evaluación experimental sobre el mapa de Nueva York se realizó con los veinte primeros problemas de los propuestos en Machuca et al. (2012). El conjunto experimental sobre el mapa de Vermont se definió de manera análoga. Una explicación más detallada de la generación de estos problemas puede ser encontrada en la Sección 3.2.2. El tiempo límite de ejecución para cada problema se estableció en 8 horas.

Con respecto a las dos clases de problemas analizados, estos están definidos de manera análoga a los problemas del Capítulo 6 sobre mallas aleatorias.

A.4 Conclusiones

Las principales conclusiones de esta tesis doctoral pueden ser enumeradas de la siguiente forma:

1. **Proponemos un nuevo algoritmo de búsqueda multicriterio, LEXGO^*.** El nuevo algoritmo exacto que presentamos devuelve el conjunto de caminos óptimos de Pareto que satisface un conjunto de metas lexicográficas agrupadas en niveles de prioridad, o el subconjunto con la desviación mínima con respecto

a las metas, si estas no pueden ser completamente satisfechas. LEXGO* también se acompaña de demostraciones formales de su correctitud.

2. **LEXGO* es más eficiente que una búsqueda completa de todas las soluciones óptimas de Pareto.** LEXGO* demuestra formalmente expandir un subconjunto de las etiquetas expandidas por una búsqueda multicriterio sin preferencias. Puesto que el número de etiquetas expandidas no corresponde a un indicador de confianza completamente fiel del rendimiento de los algoritmos de búsqueda multicriterio, hemos comparado LEXGO* y NAMOA* en bancos de pruebas con mallas aleatorias y mapas de carreteras. En ambos casos, se observa una pequeña sobrecarga de tiempo en LEXGO* cuando el conjunto completo de soluciones de Pareto satisface todas las metas. Por otra parte, el rendimiento relativo de LEXGO* sobre NAMOA* mejora progresivamente con la reducción del conjunto de soluciones que satisface las metas, concretamente, los requisitos temporales de LEXGO* son de alrededor de un orden de magnitud menor cuando las metas se satisfacen para algunas de las soluciones de Pareto, y de hasta cuatro órdenes de magnitud menor cuando las metas no pueden ser completamente satisfechas.

3. **La política de selección de etiquetas afecta el rendimiento temporal.** Se han realizado experimentos con dos políticas de selección de etiquetas, el orden lexicográfico y la agregación lineal, sobre los algoritmos LEXGO* y NAMOA*. Se observa que $\text{NAMOA}^*_{\text{lin}}$ es aproximadamente un 50% más rápido que $\text{NAMOA}^*_{\text{lex}}$ en mallas. Sin embargo, $\text{LEXGO}^*_{\text{lin}}$ solo es más eficiente que $\text{LEXGO}^*_{\text{lex}}$ cuando un alto porcentaje de las soluciones Pareto óptimas satisfacen todas las metas. Con respecto a los experimentos en los que las metas no pueden ser completamente satisfechas, $\text{LEXGO}^*_{\text{lex}}$ y $\text{LEXGO}^*_{\text{lin}}$ son prácticamente equivalentes puesto que no se realiza filtrado.

4. **Proponemos una nueva técnica de reducción de dimensionalidad (t-discarding).** El rendimiento temporal, más que el rendimiento de memoria, se ha mostrado como el factor limitante en el número de problemas que pueden ser resueltos con búsqueda multicriterio. Así, presentamos una técnica simple, pero muy efectiva, para reducir el número de etiquetas pertenecientes a los conjuntos responsables de descartar nuevas alternativas. Esta técnica se aplica bajo condiciones razonables: el uso de una función consistente como límite inferior para el algoritmo y el orden lexicográfico para seleccionar alternativas de la cola de ABIERTOS. Cabe la pena reseñar que el cálculo de una función consistente de límites inferiores puede ser es una tarea simple gracias a Tung & Chew (1992), que propusieron usar el punto ideal como límite inferior para la búsqueda multicriterio.

5. **T-discarding se utiliza ampliamente para descartar etiquetas.** Se emplean conjuntos de etiquetas cerradas, $G_{cl}(n)$, y etiquetas pertenecientes a caminos solución, COSTS, para realizar la poda (*poda en cerrados*) y el proceso de filtrado, respectivamente. Sin embargo, los conjuntos de etiquetas abiertas, $G_{op}(n)$, no pueden beneficiarse de nuestra técnica de mejora de las comprobaciones de dominancia (*poda en abiertos*). Por lo tanto, se han lanzado experimentos para

medir el porcentaje de etiquetas que son descartadas en los procesos de *poda en abiertos*, *poda en cerrados* y filtrado. En ninguno de los experimentos realizados, ni sobre mallas ni sobre mapas de carreteras, el proceso de *poda en abiertos* fue responsable del descarte de más del 10% de las etiquetas. De esta manera, podemos asegurar que la técnica de t-discarding es altamente utilizada por los algoritmos de búsqueda multicriterio para descartar nuevas alternativas y que, por lo tanto, tendrá un alto grado de impacto en el rendimiento temporal del algoritmo.

6. **T-discarding realiza comprobaciones de dominancia sobre conjuntos reducidos de etiquetas.** El tamaño relativo de los conjuntos de etiquetas truncadas sobre el tamaño de los conjuntos originales es teóricamente desconocido. Sin embargo, hemos realizados una evaluación experimental y este tamaño en la práctica es alrededor de dos órdenes de magnitud menor para los problemas más difíciles sobre mallas y de dos a tres órdenes de magnitud para los problemas de mayor dificultad sobre mapas de carreteras. Por otra parte, la ventaja relativa de los algoritmos que emplean la técnica de t-discarding se incrementa gradualmente con el tamaño del problema, por lo que esta contribución puede efectivamente extender el conjunto de problemas de búsqueda multicriterio que pueden ser resueltos en la práctica.

7. **NAMOA$^*_{\mathbf{dr}}$ reduce los requisitos temporales de NAMOA*.** La técnica de t-discarding puede ser aplicada a algoritmos exactos de búsqueda multicriterio como NAMOA*. En problemas aleatorios sobre mallas el coeficiente de mejora para los problemas de mayor dificultad fue de 8.94 para NAMOA$^*_{\text{dr}}$, la versión eficiente en tiempo de NAMOA* que emplea t-discarding, sobre NAMOA$^*_{\text{lin}}$, la versión estudiada más eficiente de NAMOA*. En mapas de carreteras el factor de mejora sobre NAMOA$^*_{\text{lin}}$ fue de 26.55, es decir, NAMOA$^*_{\text{dr}}$ resolvió el conjunto experimental en el 3.77% del tiempo necesario por NAMOA$^*_{\text{lin}}$. Para el conjunto experimental de Nueva York, un conjunto de problemas realistas de elevada dificultad con tres criterios simultáneos, NAMOA$^*_{\text{dr}}$ también amplió el número de problemas resueltos por NAMOA$^*_{\text{lin}}$, del 20% al 70% de los problemas propuestos.

8. **LEXGO$^*_{\mathbf{dr}}$ reduce los requisitos temporales de LEXGO* en algunos casos.** Al contrario que NAMOA$^*_{\text{dr}}$, LEXGO$^*_{\text{dr}}$, la versión eficiente de LEXGO* con t-discarding, no puede beneficiarse completamente de esta técnica. Aquellos casos donde las metas no pueden ser completamente satisfechas, el orden lexicográfico impone un orden de expansión de etiquetas que no es compatible con la corrección de la mencionada técnica. Sin embargo, LEXGO$^*_{\text{dr}}$ puede ser empleado mientras todas las metas puedan ser satisfechas, lo cual, de hecho, sucede en los problemas de mayor dificultad, y lograr coeficientes de mejora en los tiempos de ejecución sobre LEXGO* de 8.31 y 21.72 para los problemas más difíciles sobre mallas y mapas de carreteras, respectivamente.

9. **Una comparativa final entre NAMOA$^*_{\mathbf{dr}}$ y LEXGO$^*_{\mathbf{dr}}$.** Por último, se ha llevado a cabo una evaluación experimental de las alternativas estudiadas que han demostrado ser más eficientes. En problemas aleatorios sobre mallas, la inclusión de la técnica de t-discarding ha conseguido que NAMOA$^*_{\text{dr}}$ sea la mejor

alternativa cuando un alto porcentaje de las soluciones eficientes del problema satisfacen las metas del decisor. Por otro lado, $\text{LEXGO}^*_{\text{dr}}$ es la mejor alternativa cuando las metas no pueden ser completamente satisfechas o una pequeña porción del conjunto de soluciones eficientes es devuelto. Los resultados en problemas de mapas de carreteras son compatibles, aunque se debe puntualizar que cuando las metas pueden ser satisfechas existió un número significativo de problemas que fueron resueltos por $\text{NAMOA}^*_{\text{dr}}$ que no pudieron ser resueltos por $\text{LEXGO}^*_{\text{dr}}$ dentro del límite temporal.

A.5 Trabajo Futuro

Esta tesis doctoral ha realizado contribuciones, en varios campos, con nuevos algoritmos y técnicas de optimización. Estas contribuciones también plantean al mismo tiempo nuevas líneas de investigación, donde los enfoques actuales pueden ser efectivamente mejorados. Las siguientes líneas merecen ser investigadas con posterioridad a la lectura de esta tesis:

- La evaluación experimental llevada a cabo en esta tesis ha remarcado la importancia de la política de selección de etiquetas en los requisitos temporales de algoritmos de búsqueda multicriterio. La determinación de una política óptima sería una contribución deseable y merece un estudio más profundo.

- El estudio y desarrollo de estructuras de datos más eficientes para almacenar los conjuntos de etiquetas abiertas y cerradas, así como la cola de alternativas, son investigaciones en curso. Todos los algoritmos presentados en esta tesis podrían beneficiarse de implementaciones más eficientes de estas estructuras de datos.

- LEXGO^* puede ser extendido para usar otro modelos de programación por metas u otras fórmulas para medir la desviación con respecto a las metas. Sin embargo, necesitaríamos probablemente desarrollar nuevas reglas de poda y/o filtrado para cada caso particular, con el objetivo de mejorar su eficiencia.

- Comprobar la eficacia de la técnica de t-discarding en otros algoritmos de búsqueda multicriterio con límites inferiores. Hemos aplicado t-discarding a NAMOA^*, pero podría ser aplicado a otros algoritmos para reducir sus requisitos temporales.

- Tanto NAMOA^* como LEXGO^* devuelven el conjunto de todas las soluciones no dominadas (y que satisfacen las metas en el caso de LEXGO^*). Ya hemos podido comprobar en nuestros experimentos que aquellos problemas de mayor dificultad que buscan la frontera de Pareto completa llevan aparejados altos requisitos temporales. Sin embargo, hay otras técnicas de decisión multicriterio que buscan solo una solución eficiente, como es el caso de la búsqueda compromiso. Algunas de las conclusiones alcanzadas en esta tesis pueden ser útiles y extensibles a otros modelos de búsqueda multicriterio.

- Aplicar t-discarding a problemas con más de tres criterios simultáneos. Teóricamente, la mejora relativa debería ser incluso mayor que en los experimentos

mostrados en esta tesis. Sin embargo, estaremos probablemente acotados a resolver problemas sobre escenarios artificiales o problemas realistas sobre grafos de menor tamaño, debido principalmente a la extraordinaria dificultad que entrañaría resolver problemas significativamente grandes con más de tres criterios.

- La búsqueda multicriterio sobre mapas de carreteras es un tema muy popular durante los últimos años. Puesto que el tiempo medio de consulta para los problemas de camino más corto de un solo objetivo se ha reducido al orden de microsegundos, la evolución natural es la resolución de problemas que involucren la optimización de más de un criterio simultáneamente. Las técnicas avanzadas de optimización aplicadas al problema con un solo objetivo, como los grafos multinivel (Schulz et al., 2002) o la contracción de jerarquías (Geisberger et al., 2008) representan posibilidades para la búsqueda multicriterio que merecen una investigación más profunda. Por ejemplo, técnicas de preprocesamiento como la contracción de jerarquías requieren realizar una búsqueda bidireccional multicriterio. De hecho, este es otro campo de investigación de interés del autor de esta tesis (Pulido et al., 2011, 2012), por lo que su aplicación sería uno de los siguientes desarrollos a realizar.

- Por último, una linea de investigación recurrente es la identificación de nuevos dominios potenciales de aplicación para la Búsqueda Multicriterio, así como la combinación con otras disciplinas con las que enfocar el problema desde diversas perspectivas.

Bibliography

Ahuja, R. K., Mehlhorn, K., Orlin, J., & Tarjan, R. E. (1990). Faster algorithms for the shortest path problem. *Journal of the ACM*, 37(2), 213–223.

Aouni, B., Colapinto, C., & La Torre, D. (2014). Financial portfolio management through the goal programming model: Current state-of-the-art. *European Journal of Operational Research*, 234(2), 536–545.

Aouni, B. & Kettani, O. (2001). Goal programming model: A glorious history and a promising future. *European Journal of Operational Research*, 133, 225–231.

Azevedo, J. & Martins, E. (1991). An algorithm for the multiobjective shortest path problem on acyclic networks. *Investigação Operacional*, 11(1), 52–69.

Bankian-Tabrizi, B., Shahanaghi, K., & Saeed Jabalameli, M. (2012). Fuzzy multi-choice goal programming. *Applied Mathematical Modelling*, 36(4), 1415–1420.

Baum, M., Dibbelt, J., Hübschle-Schneider, L., Pajor, T., & Wagner, D. (2014). Speed-consumption tradeoff for electric vehicle route planning. In *14th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*.

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.

Branke, J., Deb, K., Miettinen, K., & Slowinski, R., Eds. (2008). *Multiobjective Optimization, Interactive and Evolutionary Approaches [outcome of Dagstuhl seminars].*, volume 5252 of *Lecture Notes in Computer Science*. Springer.

Brumbaugh-Smith, J. & Shier, D. (1989). An empirical investigation of some bicriterion shortest path algorithms. *European Journal of Operational Research*, 43, 216–224.

Caballero, R., Gómez, T., & Ruiz, F. (2009). Goal programming: realistic targets for the near future. *Journal of Multi-Criteria Decision Analysis*, 16(3-4), 79–110.

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.

Cazenave, T. (2006). Optimizations of data structures, heuristics and algorithms for path-finding on maps. *2006 IEEE Symposium on Computational Intelligence and Games*, (pp. 27–33).

Chankong, V. & Haimes, Y. Y. (1983). *Multiobjective Decision Making Theory and Methodology*. Elsevier Science, New York.

Charnes, A. & Cooper, W. (1961). Management models and industrial applications of linear programming. *Naval Research Logistics Quarterly*, 9(1), 63–64.

Charnes, A. & Cooper, W. (1977). Goal programming and multiple objective optimization. *European Journal of Operational Research*, 1, 39–54.

Cherkassky, B. V., Goldberg, A., & Silverstein, C. (1999). Buckets, heaps, lists, and monotone priority queues. *SIAM Journal on Computing*, (pp. 83–92).

Cherkassky, B. V., Goldberg, A. V., & Radzik, T. (1996). Shortest path algorithms: Theory and experimental evaluation. *Mathematical programming*, 73(2), 129–174.

Climaco, J. a. 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.

Climaco, J. a. C. N. & Martins, E. (1982). A bicriterion shortest path algorithm. *European Journal of Operational Research*, (pp. 399–404).

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.

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.

Craveirinha, J., Girão-Silva, R., Clímaco, J., & Martins, L. (2009). A hierarchical multiobjective routing model for mpls networks with two service classes. In *System Modeling and Optimization* (pp. 196–219). Springer.

Current, J. R., Revelle, C. S., & Cohon, J. L. (1990). An interactive approach to identify the best compromise solution for two objective shortest path problems. *Computers & Operations Research*, 17(2), 187–198.

da Silva, A. F. & Marins, F. A. S. (2014). A fuzzy goal programming model for solving aggregate production-planning problems under uncertainty: A case study in a brazilian sugar mill. *Energy Economics*, 45, 196–204.

Deb, K. (1999). Solving goal programming problems using multi-objective genetic algorithms. *Evolutionary Computation, 1999. CEC 99*, 1.

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. M., Canu, S., Iocchi, L., Nardi, D., & Ziparo, V. a. (2009). Multi-objective multi-robot surveillance. *2009 4th International Conference on Autonomous Robots and Agents*, (pp. 68–73).

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 *Proceedings of the 8th International Symposium on Experimental Algorithms (SEA'09)*, volume 2 (pp. 125–136).: Springer Verlag.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1, 269–271.

Disser, Y., Müller-Hannemann, M., & Schnee, M. (2008). Multi-criteria shortest paths in time-dependent train networks. *Experimental Algorithms*, (pp. 347–361).

Ehrgott, M. (2005). *Multicriteria Optimization*. Springer, 2nd edition.

Ehrgott, M. & Gandibleux, X. (2000). A survey and annotated bibliography of multiobjective combinatorial optimization. *OR-Spektrum*, (pp. 425–460).

Erkut, E., Tjandra, S., & Verter, V. (2007). Hazardous materials transportation. *Handbooks in operations research and management science*, 14, 539–621.

Fujimura, K. (1996). Path planning with multiple objectives. *Robotics & Automation Magazine, IEEE*, 3(March), 33–38.

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(3), 533–542.

Galand, L., Ismaili, A., Perny, P., & Spanjaard, O. (2013). Bidirectional preference-based search for multiobjective state space graph problems. In *6th Annual Symposium on Combinatorial Search (SoCS)* (pp. 80–88).

Galand, L. & Perny, P. (2006). Search for compromise solutions in multiobjective state space graphs. *17th European Conference on Artificial Intelligence, ECAI'2006*.

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.

Geisberger, R., Sanders, P., Schultes, D., & Delling, D. (2008). Contraction hierarchies: Faster and simpler hierarchical routing in road networks. *Experimental Algorithms*, 2.

Goodrich, M. & Pszona, P. (2014). Two-phase bicriterion search for finding fast and efficient electric vehicle routes. In *22nd ACM SIGSPATIAL International Conference on Advancesin Geographic Information Systems*.

Granat, J. & Guerriero, F. (2003). The interactive analysis of the multicriteria shortest path problem by the reference point method. *European Journal of Operational Research*, 151(1), 103–118.

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.

Hansen, P. (1980). Bicriterion path problems. In *Lecture Notes in Economics and Mathematical Systems*, volume 177 (pp. 109–127).: Springer.

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.

Ignizio, J. (1976). An approach to the capital budgeting problem with multiple objectives. *The Engineering Economist*, 21(4), 259–272.

Ignizio, J. (1978). A review of goal programming: A tool for multiobjective analysis. *Journal of the Operational Research Society*, 29(11), 1109–1119.

Ignizio, J. & Thomas, L. (1984). An enhanced conversion scheme for lexicographic, multiobjective integer programs. *European journal of Operational Research*, 18, 57–61.

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.

Jones, D. & Tamiz, M. (2010). *Practical goal programming*, volume 141 of *International Series in Operations Research & Management Science*. Springer.

Jozefowiez, N., Semet, F., & Talbi, E.-G. (2008). Multi-objective vehicle routing problems. *European Journal of Operational Research*, 189(2), 293–309.

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. (1997). Finding optimal solutions to rubik's cube using pattern databases.

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.

Korf, R. E. & Taylor, L. (1996). Finding optimal solutions to the twenty-four puzzle.

Larbani, M. & Aouni, B. (2007). On the pareto optimality in goal programming. *ASAC*, 2(1961), 127–131.

Machuca, E. (2011). An analysis of multiobjective search algorithms and heuristics. *Proceedings of the Twenty-Second international joint Conference on Artificial Intelligence*.

Machuca, E. (2012). *An Analysis of Some Algorithms and Heuristics for Multiobjective Graph Search*. PhD thesis, Universidad de Málaga.

Machuca, E. & Mandow, L. (2011). Multiobjective route planning with precalculated heuristics. *Proc. of the 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(7), 6435–6445.

Machuca, E., Mandow, L., & Galand, L. (2013). An evaluation of best compromise search in graphs. *Advances in Artificial Intelligence*, 8109(2013), 1–11.

Machuca, E., Mandow, L., Pérez de la Cruz, J., & Ruiz-Sepulveda, A. (2010). An empirical comparison of some multiobjective graph search algorithms. *KI 2010*, (pp. 238–245).

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.

Mali, G., Michail, P., Paraskevopoulos, A., & Zaroliagis, C. (2013). A new dynamic graph structure for large-scale transportation networks. In *Algorithms and Complexity*, volume 7878 of *Lecture Notes in Computer Science* (pp. 312–323). Springer Berlin Heidelberg.

Mali, G., Michail, P., & Zaroliagis, C. (2012). Faster multiobjective heuristic search in road maps. In *Proc. of Int. Conf. on Advances in Information and Communication Technologies*, volume 3 (pp. 67–72).

Mandow, L. & Pérez de la Cruz, J. (2007). A multiobjective frontier search algorithm. In *Proceedings of the 20th international joint conference on Artifical intelligence* (pp. 2340–2345).: Morgan Kaufmann Publishers Inc.

Mandow, L. & Pérez de la Cruz, J. (2008a). 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. (2008b). Path recovery in frontier search for multiobjective shortest path problems. *Journal of Intelligent Manufacturing*, 21(1), 89–99.

Mandow, L. & Pérez de la Cruz, J. (2009). A memory-efficient search strategy for multiobjective shortest path problems. *KI 2009: Advances in Artificial Intelligence*, 5803, 25–32.

Mandow, L. & Pérez de la Cruz, J. (2010). Multiobjective a * search with consistent heuristics. *Journal of the ACM*, 57(5), 1–25.

Mandow, L. & Pérez De La Cruz, J. L. (2001). A heuristic search algorithm with lexicographic goals. *Engineering Applications of Artificial Intelligence*, 14, 751–762.

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.

Martins, E. (1984a). On a multicriteria shortest path problem. *European Journal of Operational Research*, 16, 236–245.

Martins, E. (1984b). On a special class of bicriterion path problems. *European Journal of Operational Research*, 17(1), 85–94.

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. Q. V. (1984c). An algorithm for ranking paths that may contain cycles. *European Journal of Operational Research*, 18(1), 123 – 130.

Mehlhorn, K. & Sanders, P. (2008). *Algorithms and Data Structures.* Springer.

Miettinen, K. (1998). *Nonlinear multiobjective optimization*, volume 12 of *International Series in Operations Research & Management Science.* Kluwer Academic Publishers, Dordrecht.

Modesti, P. & Sciomachen, A. (1998). A utility measure for finding multiobjective shortest paths in urban multimodal transportation networks. *European Journal of Operational Research*, 111, 495–508.

Mote, J., Murthy, I., & Olson, D. (1991). A parametric approach to solving bicriterion shortest path problems. *European Journal of Operational Research*, 53, 81–92.

Müller-Hannemann, M. & Schnee, M. (2004). Finding all attractive train connections by multi-criteria pareto search. *Railway Optimization 2004. LNCS*, 4359, 246–263.

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.

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.

Orumie, U. & Ebong, D. (2014). A glorious literature on linear goal programming algorithms. *American Journal of Operations Research*, (pp. 59–71).

Paixão, J. & Santos, J. (2013). Labelling methods for the general case of the multi-objective shortest path problem-a computational study. In *Computational Intelligence and Decision Making*, volume 61 of *Intelligent Systems, Control and Automation: Science and Engineering* (pp. 489–502). Springer Netherlands.

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.

Pal, B., Biswas, P., & Mukhopadhyay, A. (2012). Using genetic algorithm to goal programming model of solving economic-environmental electric power generation problem with interval-valued target goals. In P. Balasubramaniam & R. Uthayakumar (Eds.), *Mathematical Modelling and Scientific Computation*, volume 283 of *Communications in Computer and Information Science* (pp. 156–169). Springer Berlin Heidelberg.

Pareto, V. (1897). *Course d'économie politique*. F. Pichou, Lausanne and Paris.

Pearl, J. (1984). *Heuristics: Intelligent search strategies for computer problem solving*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Pérez de la Cruz, J., Mandow, L., & Machuca, E. (2013). A case of pathology in multiobjective heuristic search. *Journal of Artificial Intelligence Research*, 48, 717–732.

Perny, P. & Spanjaard, O. (2005). A preference-based approach to spanning trees and shortest paths problems. *European Journal of Operational Research*, 23.

Perny, P. & Weng, P. (2010). On finding compromise solutions in multiobjective markov decision processes. *ECAI*, (pp. 969–970).

Pulido, F. J., Mandow, L., & Pérez de la Cruz, J. (2011). An analysis of bidirectional heuristic search in game maps. *aepia.aic.uniovi.es*, 1.

Pulido, F. J., Mandow, L., & Pérez de la Cruz, J. (2014). Multiobjective shortest path problems with lexicographic goal-based preferences. *European Journal of Operational Research*, 239(1), 89–101.

Pulido, F. J., Mandow, L., & Pérez de la Cruz, J. (2015). Dimensionality reduction in multiobjective shortest path search. *Computers & Operations Research*, 64, 60–70.

Pulido, F. J., Mandow, L., & Pérez de la Cruz, J. L. (2012). A two-phase bidirectional heuristic search algorithm. In K. Kersting & M. Toussaint (Eds.), *Frontiers in Artificial Intelligence and Applications, Volume 241: STAIRS 2012* (pp. 240 – 251).: IOS Press.

Pyrga, E., Schulz, F., Wagner, D., & Zaroliagis, C. (2008). Efficient models for timetable information in public transportation systems. *ACM Journal of Experimental Algorithmics*, 12(2), 1.

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.

Romero, C. (1986). A survey of generalized goal programming (1970 - 1982). *European Journal of Operational Research*, 25(2), 183–191.

Romero, C. (1991). Handbook of critical issues in goal programming. *Mathematical Social Sciences*, 22(2), 185.

Romero, C. (1993). *Teoría de la decisión multicriterio: conceptos, técnicas y aplicaciones*. Alianza universidad textos. Alianza Editorial.

Rosenthal, R. (1983). Goal programming- a critique. *NZOR*, 11(1), 1–8.

Sanders, P. & Mandow, L. (2013). Parallel label-setting multi-objective shortest path search. *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*, (pp. 215–224).

Sauvanet, G. & Néron, E. (2010). Search for the best compromise solution on multiobjective shortest path problem. *Electronic Notes in Discrete Mathematics*, 36, 615–622.

Schniederjans, M. (1995). *Goal programming: methodology and applications*. Springer US.

Schultes, D. (2008). *Route Planning in Road Networks*. PhD thesis, KIT.

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.

Sen, S. & Pal, B. B. (2013). Interval goal programming approach to multiobjective fuzzy goal programming problem with interval weights. *Procedia Technology*, 10, 587–595.

Shahnazari-Shahrezaei, P., Tavakkoli-Moghaddam, R., & Kazemipoor, H. (2013). Solving a multi-objective multi-skilled manpower scheduling model by a fuzzy goal programming approach. *Applied Mathematical Modelling*, 37(7), 5424–5443.

Simon, H. A. (1956). Rational choice and the structure of the environment. *Psychological Review*, 63(2), 129–138.

Skriver, a. & Andersen, K. (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.

Tamiz, M., Jones, D., & Romero, C. (1998). Goal programming for decision making: An overview of the current state-of-the-art. *European Journal of Operational Research*, 111(3), 569–581.

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.

Tung, C. T. & Chew, K. L. (1992). Theory and methodology a multicriteria pareto-optimal path algorithm. *European Journal of Operational Research*, 62, 203–209.

Tzeng, G. & Huang, J. (2011). *Multiple Attribute Decision Making: Methods and Applications*. A Chapman & Hall book. Taylor & Francis.

Wierzbicki, A. (1986). On the completeness and constructiveness of parametric characterizations to vector optimization problems. *Operations-Research-Spektrum*, 8, 73–87.

Wu, P., Campbell, D., & Merz, T. (2011). Multi-objective four-dimensional vehicle motion planning in large dynamic environments. *IEEE Transactions on Systems, Man and Cybernetics*, 41(3), 621–634.

Wu, Q. & Hartley, J. (2004). Using k-shortest paths algorithms to accommodate user preferences in the optimization of public transport travel. *Applications of Advanced Technologies in Transportation Engineering (2004)*, (pp. 181–186).

Yu, P.-L. (1985). *Multiple criteria decision making*, volume 30 of *Mathematical Concepts and Methods in Science and Engineering*. Springer.

Zanakis, S. H. & Gupta, S. K. (1985). A categorized bibliographic survey of goal programming. *Omega*, 13(3), 211–222.

Zeleny, M. (1981). The pros and cons of goal programming. *Computers & Operations Research*, (pp. 357–359).

Zeleny, M. (1982). *Multiple criteria decision making*. McGraw-Hill, New York.

Zeleny, M. (1984). *(MCDM) Past Decade and Future Trends, A Source Book of Multiple Criteria Decision Making*. London: JAI Press.

Zhan, F. (1997). Three fastest shortest path algorithms on real road networks: Data structures and procedures. *Journal of Geographic Information and Decision Analysis*, 1(1), 69–82.

Zhan, F. & 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.