UNIVERSIDADE D
COIMBRA

Ali Moghanni Dehkharghani

FINDING DISSIMILAR PATHS: FORMULATIONS AND VARIANTS

Tese no âmbito do Programa Interuniversitário de Doutoramento em Matemática, orientada pela Professora Doutora Marta Margarida Braz Pascoal e apresentada ao Departamento de Matemática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Maio de 2021

Finding dissimilar paths: formulations and variants

Ali Moghanni Dehkharghani

UNIVERSIDADE D COIMBRA

U.PORTO

UC|UP Joint PhD Program in Mathematics

Programa Interuniversitário de Doutoramento em Matemática

PhD Thesis | Tese de Doutoramento

May 2021

To my wife: Mina

Acknowledgements

First and foremost, I would like to thank my supervisor, Prof. Marta Pascoal, for all the help and support I have received from her during the past three years. This thesis would not have been completed without her guidance. I would like to thank her for her patience in reviewing many inferior drafts and guiding me with almost every single detail. I also want to thank her for giving me the opportunity to travel to many conferences during my studies. This was the start of a three-year journey where I got to experience her kindness as well as her rigor, through numerous friendly yet intense discussions. I hope that this thesis reflects the positive influence she had on me, and I certainly expect it to continue in the future. *Muito Obrigado*, Marta.

I am grateful to Prof. Luís Nunes Vicente and Prof. José Luis Santos for providing me research scholarship through the "MobiWise: From mobile sensing to mobility advising" (P2020 SAICTPAC/0011/2015) project. Luís welcomed me in the amazing city of Coimbra during my first year, introducing me to the world of optimization giving me an opportunity to work on the research project. Thank you for your supports: I am honored that you considered me as a member of this project.

My endless thanks go to my co-author, Prof. Maria Teresa Godinho. It was a real pleasure to work and interact with her.

I am grateful to CMUC and MobiWise project for financing me to travel to many interesting and useful conferences. I would also like to acknowledge all committee members of the UC|UP Ph.D. program throughout my Ph.D. years which provided me this opportunity to do my research in a pleasant environment. Moreover, I would also like to thank the University of Coimbra and in particular, the Department of Mathematics, for all the academic, administrative, and technical supports.

I also acknowledge many professors, library staff, friends, and all the people who followed, advised, and supported my studies and research efforts over the years.

Last but not least, words cannot describe how thankful I am to my family, and especially my parents for always supporting me and my choices. I thank my wife Mina for her emotional support and constant encouragement and for standing beside me to overcome every single situation, in particular, her incredible support for my Ph.D. studies and especially during the final stage of my thesis writing.

Cofinanciado por:

Abstract

In the present work, the problem of determining sets of K dissimilar paths between a pair of nodes in a given graph is studied. The concept of path dissimilarity, not being uniform, aims to translate some diversity in the solution sought, which generally relates what the paths of this solution have in common and how they differ. The characteristics of the dissimilarity functions make it difficult to use them as objective functions for combinatorial optimization problems, which was the motivation for looking for alternative ways of representing this concept.

In the first part of the thesis, integer linear optimization formulations are introduced that model the dissimilarity of a set of paths based on three assumptions: the minimization of the number of overlapping arcs between the pairs of paths in the set; the minimization of the number of overlapping arcs, and the minimization of the number of repetitions of the overlapping arcs. Additionally, the last two types of formulations are combined with capacity constraints, dependent on an upper-bound obtained by means of an auxiliary problem, which have the double role of limiting the number of uses of the solution arcs and promoting diversification in the event of ties. The suitability of each formulation for solving the original problem, as well as its limitations, is assessed.

In the second part of the thesis, it is assumed that each arc is associated with a given real value and that it is intended to simultaneously minimize two conflicting objective functions: the dissimilarity of K paths and a linear function of its arc values. The extension of some of the previous formulations to the mono-objective case is investigated and an ϵ-constraint type method for the calculation of the Pareto frontier of the bi-objective problem is presented, following two strategies: one in which the ϵ parameter is updated in a decreasing way and another in which the same parameter increases, and which allows to explore the relationship between the sequence of sub-problems that has to be solved.

Keywords: K alternative paths, Dissimilarity, Integer linear optimization formulations, Bi-objective optimization

Resumo

No presente trabalho estuda-se o problema da determinação de conjuntos de K caminhos dissimilares entre um par de nós de um grafo dado. O conceito de dissimilaridade de caminhos, não sendo uniforme, pretende traduzir alguma diversidade na solução procurada, que geralmente relaciona o que os caminhos dessa solução têm de comum e de diferente. As características de funções de dissimilaridade dificultam a sua utilização como funções objetivo de problemas de otimização combinatória, o que serviu de motivação para a procura de formas alternativas de representar este conceito.

Na primeira parte da tese introduzem-se formulações de otimização linear inteira que modelam a dissimilaridade de um conjunto de caminhos com base em três pressupostos: a minimização da sobreposição de arcos entre os pares de caminhos do conjunto; a minimização do número de arcos que se sobrepõem e a minimização do número de repetições dos arcos sobrepostos. Os dois últimos tipos de formulações são combinados com restrições de capacidade, dependentes de um majorante obtido através de um problema auxiliar, que têm a dupla função de limitar o número de utilizações dos arcos da solução e promover a diversificação em caso de empates. Para cada formulação é estudada a adequação à resolução do problema original, assim como as suas limitações.

Na segunda parte da tese supõe-se que cada arco está associado a um valor real dado e que se pretende minimizar simultaneamente duas funções objetivo conflituosas: a dissimilaridade de K caminhos e uma função linear dos valores dos seus arcos. Investiga-se a extensão de algumas das formulações anteriores para o caso mono-objetivo e apresenta-se um método do tipo ϵ-restrito para o cálculo da fronteira de Pareto do problema bi-objetivo em causa, seguindo duas estratégias: uma em que o parâmetro ϵ é atualizado de forma decrescente e outra em que o mesmo parâmetro aumenta, e que permite explorar a relação entre a sequência de sub-problemas que tem de ser resolvida.

Palavras-Chave: K caminhos alternativos, Dissimilaridade, Formulações de otimização linear inteira, Otimização bi-objetivo

Contents

List of Figures **xiii**

List of Tables **xv**

1 Introduction **1**

2 Literature review **7**

3 The K dissimilar paths problem **11**
3.1 The K dissimilar paths problem 11
3.2 Minimization of the number of arc overlaps for each pair of paths 12
3.2.1 Computational experiments 16
3.3 Minimization of the number of repeated arcs 20
3.3.1 Computational experiments 22
3.4 Minimization of the number of arc repetitions 24
3.4.1 Computational experiments 28
3.5 Bounding the number of arc presences 30
3.5.1 Computational experiments 32
3.6 Application to finding K dissimilar paths 35
3.6.1 Unconstrained formulations 37
3.6.2 Constrained formulations 41
3.7 Concluding remarks 45

4 The bi-objective K shortest - dissimilar paths problem **47**
4.1 Bi-objective optimization 47
4.1.1 The ϵ-constraint method 49
4.2 The bi-objective K dissimilar paths problem 53
4.2.1 Minimization of the number of repeated arcs 54
4.2.2 Minimization of the number of arc repetitions 58
4.3 Computational results 59
4.3.1 Test conditions 59
4.3.2 Test results 60
4.4 Concluding remarks 70

5 Conclusion **71**

References **75**

Appendix A **79**
A.1 Dissimilarities for the formulations 79
A.2 Dissimilarities and run times for the formulations including additional constraints 89
A.2.1 Individual results 89
A.2.2 Comparative results 93

List of Formulations

3.2 MAO : Minimization of the number of arc overlaps for each pair of paths 13
3.4 MRA : Minimization of the number of repeated arcs 21
3.5 MRO : Minimization of the number of repeated arc occurrences 25
3.6 MAR : Minimization of the number of arc repetitions 27
3.10 MRAA : Formulation MRA including the constraints (3.10) 32
3.10 MROA : Formulation MRO including the constraints (3.10) 32
3.10 MARA : Formulation MAR including the constraints (3.10) 32
4.5 BORA : The bi-objective problem that results from extending formulation MRA . . 54
4.8 BORAA : The bi-objective problem that results from extending formulation MRAA . 57
4.10 BOAR : The bi-objective problem that results from extending formulation MAR . . 58
4.12 BOARA : The bi-objective problem that results from extending formulation MARA . 59

List of Figures

1.1 Different sets of $K = 4$ paths . . . 3

3.1 Application of Algorithm 1 to a solution with loops . . . 14
3.2 Grid network . . . 16
3.3 Run times of `MAO` (seconds) . . . 19
3.4 Run times of `MRA` (seconds) . . . 23
3.5 Different sets of $K = 3$ paths . . . 24
3.6 Different sets of $K = 4$ paths . . . 26
3.7 Run times of `MRO` (seconds) . . . 29
3.8 Run times of `MAR` (seconds) . . . 30
3.9 Average `AvDi` values of the unconstrained formulations in random networks . 37
3.10 Average dissimilarity of the unconstrained formulations in random networks . 38
3.11 `AvDi` dispersion of the unconstrained formulations in random networks . . . 39
3.12 Average `MiDi` values of the unconstrained formulations in random networks . 39
3.13 Average `AvDi` values of the unconstrained formulations in grid networks . . . 40
3.14 Average dissimilarity of the unconstrained formulations in grid networks . . . 40
3.15 `AvDi` dispersion of the unconstrained formulations in grid networks . . . 41
3.16 Average `MiDi` of the unconstrained formulations in grid networks . . . 41
3.17 Average `AvDi` values of the constrained formulations in random networks . . . 43
3.18 `AvDi` dispersion of the constrained formulations in random networks . . . 43
3.19 Run times of the constrained formulations in random networks (seconds) . . . 44
3.20 Average `AvDi` values of the constrained formulations in grid networks . . . 44
3.22 Run times for the constrained formulations in grid networks (seconds) . . . 44
3.21 `AvDi` dispersion of the constrained formulations in grid networks . . . 45

4.1 The ϵ-constraint method . . . 51
4.2 Finding $K = 2$ shortest and dissimilar paths from node 1 to node 5 . . . 55
4.3 Finding $K = 3$ shortest and dissimilar paths from node 1 to node 5 . . . 56
4.4 Run times (in seconds) in random networks with the unconstrained version of minimizing the number of repeated arcs . . . 62
4.5 Mean run times (in seconds) in random networks when minimizing the number of arcs repetitions . . . 64
4.6 Comparison of the costs for `BORA` and `BOAR` . . . 64
4.7 Comparison of the dissimilarity for `BORA` and `BOAR` . . . 65

4.8 Comparison of the costs for BORAA and BOARA 67
4.9 Comparison of the dissimilarity for BORAA and BOARA 68
4.10 Comparison of the costs for the unconstrained and the constrained problems when minimizing the number of repeated arcs 68
4.11 Comparison of the dissimilarity for the unconstrained and the constrained problems when minimizing the number of repeated arcs 69
4.12 Comparison of the costs for the unconstrained and the constrained problems when minimizing the number of arc repetitions 69
4.13 Comparison of the dissimilarity for the unconstrained and the constrained problems when minimizing the number of arc repetitions 70

A.1 AvDi dispersion of MAO in random networks 83
A.2 MiDi dispersion of MAO in random networks 83
A.3 AvDi dispersion of MRA in random networks 84
A.4 MiDi dispersion of MRA in random networks 84
A.5 AvDi dispersion of MRO in random networks 84
A.6 MiDi dispersion of MRO in random networks 85
A.7 AvDi dispersion of MAR in random networks 85
A.8 MiDi dispersion of MAR in random networks 85
A.9 AvDi dispersion of MAO in grid networks 86
A.10 MiDi dispersion of MAO in grid networks 86
A.11 AvDi dispersion of MRA in grid networks 86
A.12 MiDi dispersion of MRA in grid networks 87
A.13 AvDi dispersion of MRO in grid networks 87
A.14 MiDi dispersion of MRO in grid networks 87
A.15 AvDi dispersion of MAR in grid networks 88
A.16 MiDi dispersion of MAR in grid networks 88
A.17 Average dissimilarity in random networks 93
A.18 Run times in random networks (seconds – log scale) 94
A.19 Minimum dissimilarity in random networks 95
A.20 Average dissimilarity in grid networks 96
A.21 Run times in grid networks (seconds – log scale) 96
A.22 Minimum dissimilarity in grid networks 97

List of Tables

3.1 Number of instances with K disjoint paths in random networks 17
3.2 Final number of random network instances 17
3.3 Number of instances solved to optimality by `MAO` (%) 18
3.4 Run times of `MAO` (seconds) 19
3.5 Average integer programming gaps of $\texttt{MAO}_L$ (%) 20
3.6 Number of instances solved to optimality by `MRA` (%) 22
3.7 Run times of `MRA` (seconds) 23
3.8 Average integer programming gaps of $\texttt{MRA}_L$ (%) 24
3.9 Run times of `MRO` (seconds) 29
3.10 Run times of `MAR` (seconds) 30
3.11 Average integer programming gaps of $\texttt{MRO}_L$ (%) 31
3.12 Average of R^* value 33
3.13 Run times variation for `MRAA`, `MROA` and `MARA` (%) 34
3.14 Average `AvDi` and `MiDi` of `IPM` in random networks 36
3.15 Run times of `IPM` (seconds) 36
3.16 Average `AvDi` and `MiDi` of `IPM` in grid networks 37
3.17 Average dissimilarity variation for `MRAA`, `MROA` and `MARA` (%) 42

4.1 Description of the column headings 60
4.2 Characteristics of the non-dominated points with the unconstrained version of minimizing the number of repeated arcs 61
4.3 Number of sub-problems, path dissimilarities and run times (in seconds) when minimizing the number of repeated arcs 61
4.4 Characteristics of the non-dominated points with the unconstrained version of minimizing the number of arc repetitions 63
4.5 Number of sub-problems, path dissimilarities and run times (in seconds) when minimizing the number of arc repetitions 63
4.6 Characteristics of the non-dominated points with the constrained version of minimizing the number of repeated arcs 65
4.7 Number of sub-problems, path dissimilarities and run times (in seconds) with constrained version of minimizing the number of repeated arcs 66
4.8 Characteristics of the non-dominated points with the constrained version of minimizing the number of arc repetitions 66

4.9 Number of sub-problems, path dissimilarities and run times (in seconds) with constrained version of minimizing the number of arc repetitions 67

A.1 Average `AvDi` and `MiDi` of `MAO` in random networks 79
A.2 Average `AvDi` and `MiDi` of `MRA` in random networks 80
A.3 Average `AvDi` and `MiDi` of `MRO` in random networks 80
A.4 Average `AvDi` and `MiDi` of `MAR` in random networks 81
A.5 Average `AvDi` and `MiDi` of `MAO` in grid networks 81
A.6 Average `AvDi` and `MiDi` of `MRA` in grid networks 82
A.7 Average `AvDi` and `MiDi` of `MRO` in grid networks 82
A.8 Average `AvDi` and `MiDi` of `MAR` in grid networks 83
A.9 Average `AvDi` and `MiDi` of `MRAA` in random networks 89
A.10 Average `AvDi` and `MiDi` of `MARA` in random networks 90
A.11 Run times of `MRAA` for random networks (seconds) 90
A.12 Run times of `MARA` for random networks (seconds) 90
A.13 Average `AvDi` and `MiDi` of `MRAA` in grid networks 91
A.14 Average `AvDi` and `MiDi` of `MARA` in grid networks 91
A.15 Run times of `MRAA` for grid networks (seconds) 91
A.16 Run times of `MARA` for grid networks (seconds) 92

List of Algorithms

1 Algorithm for removing loops from a given solution x 14

2 The ϵ-constraint method – Decreasing ϵ version 50

3 The ϵ-constraint method – Increasing ϵ version 52

Chapter 1

Introduction

In classic optimization problems a solution is sought which optimizes a known objective function. In multi-objective optimization this translates into a set of solutions, but, like in the previous case, the focus is on the objective functions to be optimized, rather than on the composition of the solution(s). The enumeration of solutions by order of the objective function values is generally more complex than the original problem, but it does allow to know alternatives to the optimal solution, once again, guided by the objective function. However, this model does not guarantee the "diversity" of the several solutions. On the contrary, the second best solution is often not "very different" from the first. The first difficulty in formalizing this issue lies in the definition of "diversity" that should be considered. The second is related with a dependency of the objective function on the relationship between the solutions in a given set.

Such issues are particularly important when, despite looking for solutions with a small objective function value, it is important that these are true alternatives to one another. In practical terms, this topic applies to the growing demand for security and reliability of networks and services, or when dealing with concerns such as the transport of hazardous materials or money distribution. Each of these applications has particular characteristics, but they all have the common purpose of searching for a set of paths which are as "diverse" as possible with respect to the nodes/arcs that compose them. The literature describes some specific approaches to this theme, which determine solutions for a sequence of simplified problems, related to the intended objective and mainly solved using heuristic methods. However, there is a lack of formalization of the main problem in an integrated, rather than a sequential way, which allows a better understanding of the structure of the problem and also taking into account the interdependence between the solutions, which are the main motivations of this work.

Let (N, A) denote a given directed network consisting of a set $N = \{1, \ldots, n\}$ of nodes and a set $A \subseteq N \times N$ of m arcs. Let s and t be two different nodes of N, called the source node and the target node, respectively. Finding a path in the network (N, A) between the nodes s and t is one of the most classical and widely used network optimization problems, and the basis for several applications in operations research. Studying the determination of alternative paths, on the other hand, is an interesting problem by itself that stems from different real-life

problems but has been considerably less studied than the former. For instance, in a modern and industrialized society, routing hazardous materials like poisonous gases or radioactive materials is an important issue, so the need for alternative safe routes is crucial for reducing the risk of disaster in case of accidents or if the best route becomes infeasible due to road construction [1, 6, 15, 25]. Repeating paths is also avoided in money collection, where having alternative paths/routes decreases the risk of robberies and can be used in case of danger of robberies [5, 13]. Additionally, in telecommunications, a backup path is often replaced by a primary one if a failure occurs along it or if it can be used simultaneously to spread information transmitted at a specific time [23, 24, 49].

Let $K \in \mathbb{N}$ be a given number of alternative paths to be found. The definition of alternative paths may vary depending on the application, the common denominator being that the paths in the solution should share the least possible network resources. Several works use dissimilarity measures between two paths as the metric for achieving this purpose, nevertheless, also this notion is not uniquely defined, nor would that be desirable provided that the metrics are often tailored to the application. For instance, [21] developed four indices for measuring the similarity between two paths, defined as follows:

$$\text{Index 1: } S_1(p,q) = \frac{1}{2}\left(\frac{L(p \cap q)}{L(p)} + \frac{L(p \cap q)}{L(q)}\right)$$

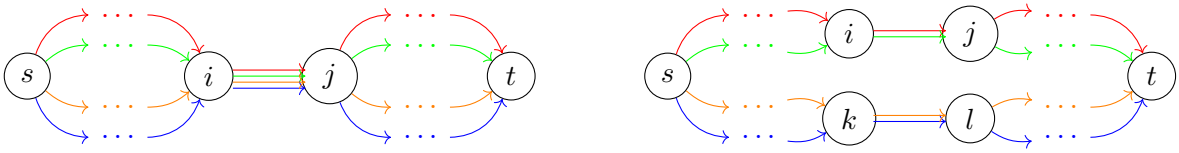
$$\text{Index 2: } S_2(p,q) = \sqrt{\frac{L^2(p \cap q)}{L(p)L(q)}}$$

$$\text{Index 3: } S_3(p,q) = \frac{L(p \cap q)}{\max\{L(p), L(q)\}}$$

$$\text{Index 4: } S_4(p,q) = \frac{L(p \cap q)}{L(p \cup q)}$$

where p and q are two paths and $L(p)$ denotes the length of path p, that is, its number of arcs and $L(p \cap q)$ denotes the number of arcs in common between the two paths p and q. The dissimilarity between p and q is then given by $D_i(p,q) = 1 - S_i(p,q)$, for $i = 1, 2, 3, 4$. The dissimilarities vary between 0 and 1, the first when the two paths coincide and the latter when they are arc disjoint. The authors also showed that there is a strong correlation between these indices. Other works have extended these concepts by including information about the underlying area affected by the paths or the distance between them, once again depending on the problem [15, 38].

Additionally, in concrete applications the problem has frequently been handled from a bi-objective point of view, having the goals of optimizing both the total paths length/cost as well as the dissimilarity of the set of paths. Now, while the shortest path problem or the ranking of K shortest paths problem are well-known and well-studied problems, the same is not true when the objective function represents paths dissimilarity. Many of these bi-objective problems have been addressed from an algorithmic approach whose primary goal is not to optimize the dissimilarity and, to our knowledge, there are no published studies considering the paths dissimilarity problem from an integer programming point of view.



(a) Solution with one repeated arc, shared by several paths

(b) Solution with many repeated arcs, shared by few paths

Fig. 1.1 Different sets of $K = 4$ paths

Filling this gap and deepening the understanding of such problems are the main motivations for the present work, also because this may be a relevant contribution for an efficient treatment of a bi-objective problem involving paths dissimilarity.

As mentioned earlier, it is not uncommon to find different understandings of the term "dissimilarity" in the literature. In this work, we focus on the conception of dissimilarity as defined by D_1. Thus, the presented models aim at producing sets of K paths with good scores in terms of D_1.

However, modeling D_1 as the objective function of an integer linear programming (ILP) model, presents some difficulties as this is a non linear metric involving an underlying combinatorial problem. On the other hand, it is intuitive that minimizing the number of arcs shared by the K paths or minimizing the number of paths that share a common arc in the K paths, favor the dissimilarity of the solution. Furthermore, these problems can be modeled quite easily, obviating the above mentioned difficulties. Thus these alternative ways of looking into the K dissimilar paths problem seem promising and are worth exploring.

In the present work we introduce and compare three families of ILP formulations, each one addressing one of the strategies mentioned before. Figure 1.1 illustrates their differences and emphasizes some of their limitations.

For simplicity, assume that all the paths represented in Figure 1.1 have the same length. There are 6 arc overlaps for every pair of paths in Figure 1.1a (that is, the sum of $L(p \cap q)$ for every pair of paths p, q in the solution is equal to 6), while in the solution depicted in Figure 1.1b there are only 2. Therefore, the second is a better solution than the first with regard to their dissimilarity. However, if one chooses to count the arcs shared by more than one path, there is only 1 in the solution in Figure 1.1a and there are 2 in the solution in Figure 1.1b. Thus, the first solution is the best with regard to this metric. This situation illustrates the major drawback of the strategy devised by this approach – the lack of control over the number of overlaps associated to arcs that are used by more than one path. By contrast, if counting the number of arc repetitions in the solution (given by the number of times an arc is present in the solution, besides its first use), there are 3 in the solution depicted in Figure 1.1a and 2 in the solution depicted in Figure 1.1b, favoring the solution that seems to be the most dissimilar one. Still, this third approach is not exempted of drawbacks, as it does not distinguish between alternative solutions with different dissimilarities.

To (partially) overcome the drawbacks of the two approaches a set of additional constraints is proposed, which imposes a bound on the number of paths that use each arc in the solution.

In this thesis, we also address the determination of K paths between two network nodes, with two goals:

- the minimization of the cost of the K paths,
- the maximization of the dissimilarity of the K paths.

In Chapter 3, we introduce and empirically compare four integer linear programming models for the K dissimilar paths. To model the K shortest – dissimilar paths problem, we use the two best integer linear formulations introduced in Chapter 3. In Chapter 4, we study how these models behave in the context of a bi-objective problem, through a set of empirical tests.

Contribution of the thesis

This thesis address the problem of finding K dissimilar paths connecting a given pair of nodes in a directed graph.

The first main contribution of the thesis is the proposal of four integer formulations for finding K dissimilar paths [45]. The formulations have different motivations, but their general goal is to minimize the number of arcs that appear in more than one path or the total number of those overlaps, while searching for sets of paths with good dissimilarity. The inclusion of an additional set of constraints to the previous formulations, with the goal of improving the solutions dissimilarity, was also proposed. The performance of the new formulations and of a traditional method in the literature, the iterative penalty method, was tested over random and grid networks, assessing the required run time as well as the average and the minimum dissimilarities of the solutions.

The second contribution of this work is the study of the bi-objective shortest–dissimilar K paths problem [46]. In this case, it is assumed that each arc is assigned with a given cost, and the main goal is then to address the problem of finding K dissimilar paths while simultaneously minimizing the total cost. Previous formulations are extended with the cost objective function, one of them also considers the minimization of the number of repeated arcs, whereas the other one considers the minimization of the number of arc repetitions. Properties of the resulting bi-objective problems are studied and the ϵ-constraint method is adapted to solve them by using two strategies: one based on decreasing the parameter ϵ and the other increasing it. The two variants of the method are tested for finding sets of 10 paths with the two formulations.

Structure of the thesis

This thesis is structured in five chapters. The first of which is this introduction. Chapter 2 gives a literature overview of problems related with finding alternative paths. Chapter 3 starts with the problem of finding K dissimilar paths between two nodes, and then ILP formulations are introduced to provide solutions based on the approaches described above. In Chapter 4, we attempt to make use of the ϵ-constraint method to consider bi-objective

shortest–dissimilar path problem. To do so, we first review basic properties of bi-objective problems and then we propose a modification of ϵ-constraint algorithm. In Chapter 5 we draw some concluding remarks based on the material covered during the thesis.

Chapter 2

Literature review

Several problems focus on finding a single path between two nodes in a network, which optimizes either a certain criterion or several criteria simultaneously, while others aim at finding a set with a given number of paths, again with respect to one, or several, criteria. The single path problems have practical interest by themselves, but finding a set of paths may still be relevant, for instance to ensure reliability and having a replacement path in case of failure in the primary one, or simply if several alternatives should avoid sharing resources with other paths. In this case ranking paths provides a pre-defined number of paths from the source node to the target node by increasing order of the objective function.

Several network optimization problems search for a path between two nodes which optimizes a certain criterion. In many cases it is of interest to extend this problem by searching not only for the best solution but also for the second best, the third best and so on, that is, to rank paths by increasing order of the objective function. In practice this is useful, for instance, when the paths need to satisfy additional constraints, which can be checked as new solutions are found. This problem, known as the K shortest paths problem, was first proposed in 1959, by [29], and is usually classified into two variants, one that aims at the determination of unconstrained paths and another one for which the nodes in each solution cannot appear more than once. Despite the first being easier to solve than the second, both can be solved in polynomial space and time, depending on the number K of paths to be found and on the size of the network. See for example [19, 32, 39, 41] for works on ranking unconstrained paths and [34, 40, 57] for works on ranking loopless paths.

The search for solutions when ranking paths is guided by the objective function, therefore, very often the solutions which are close in terms of the cost are also similar in terms of their composition. In the K disjoint paths problem, a cost objective function of K paths is minimized, while the overlaps between them are forbidden. The problem can be classified into arc disjoint or node disjoint, the second one being a particular case of the first (for instance, if every node is split into two nodes that are linked by an arc). The disjointness of the paths is often a requirement in telecommunications, in order to ensure the reliability of communications. In practice this is managed by the computation of a pair of paths connecting two given nodes, a primary path to be used as a first option and a backup path to replace the first one if there is a failure along its arcs or nodes. The determination of K disjoint simple

paths has been studied by [2, 52, 53]. Their approaches consist of formulating the problem as a minimum cost flow problem and propose the application of a labeling algorithm, changing the given network. The arc disjoint version of the problem has been studied in [22, 26, 56]. A review on disjoint path problems can be found in [31].

A handicap of the K disjoint paths problem is that the disjointness condition may be too demanding for some instances and no solution is returned in those cases. The dissimilar paths problem has been studied in the context of hazardous materials transportation, where the alternative paths should not share a large number of arcs and they should be relatively short in length.

Different methods have been proposed for approaching the dissimilar paths problem. The iterative penalty method (IPM) [33] is one of the most intuitive methods, based on the iterative application of shortest path algorithms. At each iteration, a cost penalty is associated to each selected arc to discourage them of appearing in the forthcoming iteration; hence, generating dissimilar paths. Another proposed method is the Gateway Shortest Path, [36]. In this case, the generated shortest paths should go through a given set of nodes called a "gateway". Additionally, the concept of "area under a path" is used to evaluate the similarity between two paths. The minimax method, by [35], selects paths starting from K assigned paths using some dissimilarity indices. [1] reviews the three mentioned methods for generating dissimilar paths, and proposes another dissimilar paths model that makes use of a p-dispersion location model, [20]. [21] presents four indices to measure the dissimilarity among two paths, one of which will be used later. In [7], the authors introduce a model for generating dissimilar paths that takes into account also the risk induced on the arcs in the neighborhood of a selected path.

In [8], the authors also considers the need to distribute the risk of the paths in an equitable way with respect to both the space and the time, avoiding as much as possible the presence of more than one hazardous vehicle at the same time on the same zone. Later on [15] study the problem from a multi-objective perspective. They introduce the concept of "buffer zone" in the measure of similarity. [38] choose approaches different from the previous and consider a spatial point of view in their dissimilarity index. More recently, [58] works on the bi-objective K dissimilar vehicle routing problem (kd-VRP). The work considers two dissimilarity indices: the "grid metric", which treats spatial dissimilarity, as well as the "edge metric", which defines dissimilarity via shared arcs between different routes.

The definition of dissimilarity given in Chapter 1 and the description above suggest a multi-objective perspective of this type of problems, for two main reasons: the objective function is intrinsically bi-objective, as it relates what the paths have in common as well as their lengths; but also because its extension to a case where at least another objective function is considered comes naturally and it is certainly useful in terms of applications.

Most real-life problems require taking into account several conflicting objectives. Multi-objective optimization consists in optimizing simultaneously several objective functions that are subject to several constraints. Usually these objectives are in conflict with each other and have different natures, for instance, they may be measured in different units. Therefore, normally, there is not a single solution that simultaneously optimizes every objective. Instead,

the goal of solving a multi-objective optimization problem is to find solutions that can only be improved in one objective by degrading at least one of the remaining objectives. This notion of optimality is called Pareto optimality. In general, the multi-objective optimization problems are hard to solve with exact methods, even if they are extensions simultaneously of easy single objective optimization problems [16].

One of the problems in that class which has interested researchers is the multi-objective shortest path problem, dedicated to the case where all objective functions are linear [10, 28, 55]. A survey, regarding the bi-objective case, has been published in [51]. Objective functions of different types have also been addressed, like bottleneck functions or the number of distinct arc colors [4, 30, 48], to name a few, which have been reviewed in the more general survey [11]. The solutions to the problem addressed in the present work are sets of K paths. Therefore, the previous studies do not apply straightforwardly to the linear integer approach considered in the following.

Additionally, some of the works mentioned above already incorporate several objectives and propose approximate methods for finding possible solutions that may be of interest from a practical point of view. The literature on bi-objective integer programming, on the other hand, is rich for general problems and also when considering particular problems [3, 18, 42, 47, 50]. Surveys on these works can be found in [17, 54].

Chapter 3

The K dissimilar paths problem

In this chapter, the problem of finding K dissimilar paths between two nodes is presented, and in the next three sections ILP formulations are introduced to provide solutions based on the approaches described in introduction. Computational results for different variants of each formulation are presented at the end of each section. A set of constraints which help the new formulations to obtain more dissimilar solutions is introduced in Section 3.5. Overall computational experiments for all approaches are presented in Section 3.6. The performance of the proposed formulations is analyzed and this is compared to an intuitive and classical approach in the literature of finding alternative paths, the iterative penalty method (IPM) [33]. The formulations are compared both in terms of the run time and of the dissimilarity of the output solutions, derived from D_1. Some conclusions are outlined in Section 3.7.

3.1 The K dissimilar paths problem

Let (N, A) be a directed graph with $|N| = n$ nodes and $|A| = m$ arcs, where s denotes a given source node and t denotes a given target node, $s, t \in N$. Let also P denote the set of paths in (N, A) from node s to node t and K be a given positive integer. The goal of the K dissimilar paths problem from s to t is to find a set of K paths in P, such that the paths in the set are fairly distributed throughout the network. Considering dissimilarity based in the index S_1 introduced in Chapter 1, the problem can be defined as

$$\begin{aligned} \max \quad & \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} D_1(p_i, p_j) / \binom{K}{2} \\ \text{subject to} \quad & p_1, p_2, \ldots, p_K \in P \end{aligned}$$

where

$$D_1(p, q) = 1 - S_1(p, q) = 1 - \frac{1}{2}\left(\frac{L(p \cap q)}{L(p)} + \frac{L(p \cap q)}{L(q)}\right),$$

for any paths $p, q \in P$, and this is equivalent to minimizing the similarity of the set of K paths,

$$\sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \left(\frac{L(p_i \cap p_j)}{L(p_i)} + \frac{L(p_i \cap p_j)}{L(p_j)}\right).$$

This objective function is fractional and difficult to handle directly. Therefore, we will consider simplifications of the problem.

The next sections introduce four formulations, simpler than this one, but which try to capture the main characteristic of this problem based on different assumptions.

3.2 Minimization of the number of arc overlaps for each pair of paths

The length of the paths overlap is a common term to the several similarity indices described in Chapter 1. For now we focus on that length and, by doing so, the objective function becomes linear and simpler to handle. Thus, the current goal is to find solutions for

$$\begin{aligned} \min \quad & \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} L(p_i \cap p_j) \\ \text{subject to} \quad & p_1, p_2, \ldots, p_K \in P \end{aligned} \tag{3.1}$$

Given two paths $p, q \in P$, it is said that there is an overlap whenever there is an arc $(i,j) \in A$ that belongs to both p and q, that is, if $(i,j) \in p$ and $(i,j) \in q$. Thus, the number of overlaps between those paths is the number of arcs that appear in both, $OL(p,q) = |\{(i,j) \in A : (i,j) \in p \wedge (i,j) \in q\}|$, which coincides with $L(p \cap q)$. The number of overlaps in a given set of K paths is the total number of overlaps for each pair of paths, that is,

$$OL(\{p_1, p_2, \ldots, p_K\}) = \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} OL(p_i, p_j),$$

which is the objective function in problem (3.1). To illustrate these concepts, we recall the example in Figure 1.1. The arc (i,j) belongs to all four paths in Figure 1.1a. Therefore, the number of overlaps for those paths is 6. In Figure 1.1b the arc (i,j) appears in the paths in red and green, while the arc (k,l) appears in the blue and orange paths. So, there are 2 arc overlaps in the 4 paths. This latter solution is better than the first with respect to problem (3.1).

The first formulation intends to model problem (3.1) as an integer linear program. Considering the decision variables x_{ij}^k equal to 1 if the arc (i,j) lies in the k-th path from s to

t, or 0 otherwise, for any $(i,j) \in A$ and $k = 1, \ldots, K$, the problem is formulated as follows:

$$\min \quad f_1(x,z,v) = \sum_{(i,j)\in A} v_{ij} \tag{3.2a}$$

$$\text{subject to} \quad \sum_{j\in N:(i,j)\in A} x_{ij}^k - \sum_{j\in N:(j,i)\in A} x_{ji}^k = \begin{cases} 1 & i = s \\ 0 & i \neq s,t \\ -1 & i = t \end{cases}, \quad k = 1, \ldots, K \tag{3.2b}$$

$$z_{ij}^{kl} \leq x_{ij}^k, \quad (i,j) \in A, \quad k = 1, \ldots, K-1, \quad l = k+1, \ldots, K \tag{3.2c}$$

$$z_{ij}^{kl} \leq x_{ij}^l, \quad (i,j) \in A, \quad k = 1, \ldots, K-1, \quad l = k+1, \ldots, K \tag{3.2d}$$

$$z_{ij}^{kl} \geq x_{ij}^k + x_{ij}^l - 1, \ (i,j) \in A, \ k = 1, \ldots, K-1, \ l = k+1, \ldots, K \tag{3.2e}$$

$$v_{ij} = \sum_{k=1}^{K-1} \sum_{l=k+1}^{K} z_{ij}^{kl}, \quad (i,j) \in A \tag{3.2f}$$

$$x_{ij}^k \in \{0,1\}, \ z_{ij}^{kl} \in \{0,1\}, \ (i,j) \in A, \ k = 1, \ldots, K, \ l = k+1, \ldots, K \tag{3.2g}$$

This formulation may considerably big for problems of modest size, given that the variables z are related with any pair of paths from s to t. Thus, it has $O(mK^2)$ variables and $O(mK^2 + nK)$ constraints.

The flow conservation constraints (3.2b) ensure the existence of K paths from node s to node t; the constraints (3.2c) – (3.2e) guarantee that variables $z_{ij}^{kl} \in \{0,1\}$ are equal to 1 if and only if the arc (i,j) is used both in the paths defined by the variables x_{ij}^k and x_{ij}^l, for $(i,j) \in A$, $k \in \{1, \ldots, K-1\}$, $l \in \{k+1, \ldots, K\}$. In fact:

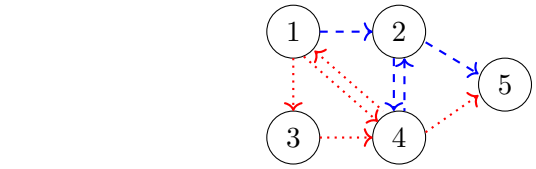
- If $x_{ij}^k = 0$ or $x_{ij}^l = 0$, then (3.2c) or (3.2d) imply that $z_{ij}^{kl} = 0$, for any $(i,j) \in A$, $k \in \{1, \ldots, K-1\}$, $l \in \{k+1, \ldots, K\}$.

- If $x_{ij}^k = x_{ij}^l = 1$, then both (3.2c) and (3.2d) imply that $z_{ij}^{kl} \leq 1$, whereas (3.2e) imply that $z_{ij}^{kl} \geq 1$. Thus, $z_{ij}^{kl} = 1$ for $(i,j) \in A$, $k \in \{1, \ldots, K-1\}$, $l \in \{k+1, \ldots, K\}$.

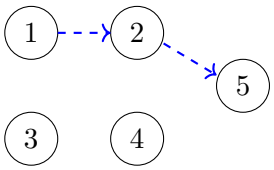
The constraints (3.2f) state that the auxiliary variables v_{ij} correspond to the number of paths that use arc $(i,j) \in A$ and constraints (3.2g) define the variables. Observe that because the x_{ij}^k are binary variables, then by (3.2c) – (3.2e), the variables z_{ij}^{kl} are binary as well and, consequently, by (3.2f), the variables v_{ij} are implicitly defined as non-negative integers, for any $(i,j) \in A$, $k \in \{1, \ldots, K-1\}$, $l \in \{k+1, \ldots, K\}$. The objective function (3.2a) corresponds to $OL(\{p_1, p_2, \ldots, p_K\})$, the total number of arcs that are shared by at least two paths from s to t.

Formulation (3.2) may admit optimal solutions that contain subtours.

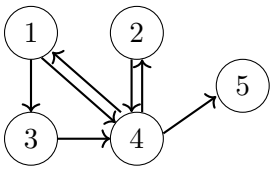
Example 3.2.1 *The network depicted in Figure 3.1a shows a possible solution for finding* $K = 2$ *paths from node* 1 *to node* 5*. In this case, the solution has objective value* 0*, given that the two paths* $p_1 = \{(1,2),(2,4),(4,2),(2,5)\}$ *and* $p_2 = \{(1,3),(3,4),(4,1),(1,4),(4,5)\}$ *represented in red and in blue respectively. They do not have any arcs in common. However, each of those paths contains one subtour, namely* $s_1 = \{(2,4),(4,2)\}$ *and* $s_2 = \{(1,3),(3,4),(4,1)\}$.



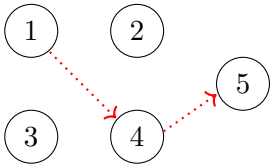
(a) Network (N', A')



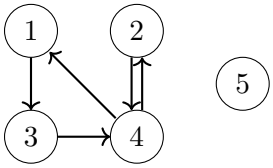
(b) Path x^1



(c) Updated network (N', A')



(d) Path x^2



(e) Updated network (N', A')

Fig. 3.1 Application of Algorithm 1 to a solution with loops

Nevertheless, if the problem is feasible, there always exists a loopless optimal solution to the problem, and this is easy to compute after a first optimal solution has been found.

Solutions with loops can be avoided by adding subtour elimination constraints to the formulation or by adding a term to the objective function that penalizes the utilization of arcs. Another alternative is to apply a post processing algorithm that allows to extract one loopless optimal solution from a given optimal solution. Algorithm 1 outlines the procedure to obtain such a loopless solution, when given a solution x.

Algorithm 1: Algorithm for removing loops from a given solution x

1 $N' \leftarrow \{i \in N : x_{ij}^k = 1 \text{ for some } j \in N \wedge k = 1, \ldots, K\} \cup \{t\}$
2 $A' \leftarrow \{(i,j) \in A : x_{ij}^k = 1 \text{ for some } k = 1, \ldots, K\}$
3 **for** $(i,j) \in A'$ **do** $u_{ij} \leftarrow \sum_{k=1}^{K} x_{ij}^k$
4 **for** $k = 1, \ldots, K$ **do**
5 **for** $(i,j) \in A$ **do** $\bar{x}_{ij} \leftarrow 0$
6 $\bar{x}^k \leftarrow$ shortest path from s to t in terms of the number of arcs in network (N', A')
7 **for** $(i,j) \in A'$ *such that* $\bar{x}_{ij}^k = 1$ **do**
8 $u_{ij} \leftarrow u_{ij} - 1$
9 **if** $u_{ij} = 0$ **then** Delete arc (i,j) from A'

In lines 1 and 2 of Algorithm 1 the network corresponding to the arcs in the given solution x is built. Line 3 assigns each arc $(i,j) \in A'$ with the number of times it appears in x, u_{ij}. This value works like the arc (i,j)'s capacity. Then, in each iteration of the loop in lines

4 to 9, one path is determined and the capacity of an arc is updated every time it is used. The loop runs K times exactly, so that K paths are found. Moreover, the paths determined on line 5 are always loopless. These paths can be found by means of breadth-first search, Cormen et al. [14], and therefore the algorithm runs in $O(Km)$ time. Also, the empirical tests reported later showed that its run time is very small when compared to solving any of the formulations here presented.

Proposition 3.2.1 *Let (x, z, v) be a feasible solution for formulation (3.2) and $\bar{x}$ be the corresponding vector output by Algorithm 1. Then, $\bar{x}$ defines K loopless paths from s to t.*

Proof. The remarks above show that the result holds. Shortly, every path defined by $\bar{x}^k$ is loopless, for any $k = 1, \ldots, K$, because it is the solution of a shortest path problem with all costs positive (unitary). Additionally, x is the characteristic vector of K paths from s to t, given that it satisfies the constraints (3.2b). Therefore, $\bar{x}^k$ defines K loopless paths from s to t. □

Proposition 3.2.2 *Let (x, z, v) be an optimal solution for formulation (3.2). Let $\bar{x}$ be the vector output by Algorithm 1 when applied to x, and $\bar{z} \in \{0,1\}^{mK^2}$ and $\bar{v} \in \mathbb{N}_0^m$ be vectors which satisfy the constraints (3.2c) – (3.2f). Then, $(\bar{x}, \bar{z}, \bar{v})$ is a loopless optimal solution for problem (3.2).*

Proof. Suppose $(\bar{x}, \bar{z}, \bar{v})$ is obtained from (x, z, v) according to Algorithm 1 and the directions above. Then, $x_{ij}^k, \bar{x}_{ij}^k \in \{0,1\}$ and

$$x_{ij}^k \geq \bar{x}_{ij}^k, \quad (i,j) \in A, \quad k = 1, \ldots, K. \tag{3.3}$$

Two aspects need to be considered:

1. According to Proposition 3.2.1, $\bar{x}$ corresponds to K paths, so the constraints (3.2b) hold. Moreover, $\bar{z}$ and $\bar{v}$ satisfy (3.2c) and (3.2f), therefore $(\bar{x}, \bar{z}, \bar{v})$ is a feasible solution of (3.2).

2. Because of condition (3.3), it also holds that

$$z_{ij}^{kl} \geq \bar{z}_{ij}^{kl}, \quad (i,j) \in A, \quad k = 1, \ldots, K-1, \quad l = k+1, \ldots, K,$$

$$v_{ij} \geq \bar{v}_{ij}, \quad (i,j) \in A,$$

   therefore $f_1(x, z, v) \geq f_1(\bar{x}, \bar{z}, \bar{v})$, which shows that the new solution is optimal.

It can then be concluded that $(\bar{x}, \bar{z}, \bar{v})$ is an optimal loopless solution of (3.2). □

In the next section we describe some computational experiments performed on formulation (3.2). Results will show that formulation (3.2) outputs solutions with very good dissimilarity scores, indicating that problem (3.1) might be a good approach to the K dissimilar paths problem, even though it neglects the role of the length of the paths in the dissimilarity of the K paths. However, the high run times required to solve even problems of modest dimension

compromise its usability in practical applications. These results are not at all unexpected, due to the combinatorial nature of the model, but they reinforce the need for recurring to alternative models, as mentioned in Chapter 1.

3.2.1 Computational experiments

The purpose of the tests presented in the following is to study the behavior of formulation (3.2) in terms of the solutions it outputs, its run times and its integer programming gaps.

The code designated by `MAO`, standing for the implementation for minimizing the number of arcs overlaps for each pair of paths, formulation (3.2), was implemented in C. This code uses IBM ILOG CPLEX version 12.7 as the ILP solver. The Algorithm 1, also coded in C, was applied to the result of this implementation, in order to remove the loops from the obtained solutions. The tests were carried out on a 64-bit PC with an Intel®Core™ i7-6700 Quad core at 3.40GHz with 64GB of RAM.

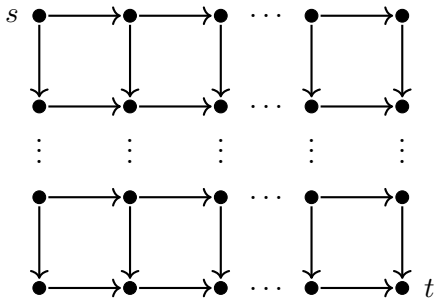


Fig. 3.2 Grid network

In the performed tests $K = 3, 4, \ldots, 10$ paths were computed in directed networks. Two types of instances were considered:

- Random networks, denoted by $R_{n,m}$, with $n = 100, 300, 500$ nodes, and $m = dn$ arcs, for average degrees $d = 5, 10$. A Hamiltonian cycle is created for all the nodes in the network, and afterwards the remaining arcs are generated randomly. This Hamiltonian cycle is directed, therefore the strong connectivity of the graph is not fully ensured. There are no parallel arcs between pairs of nodes. For each size of these networks, 30 instances were generated based on different seeds. The reported results are based only on the 22, out of the 30, that originated feasible problems.

- Grid networks, denoted by $G_{p,q}$, with $p = 3, 4, 6, 12$ rows, $q = 6, 12, 36$ columns and $n = pq = 36, 144$ nodes, arranged in a planar grid, numbered consecutively from left to right and top to bottom – as shown in Figure 3.2. Any pair of adjacent nodes is connected by an arc, thus $m = 2pq - p - q = 57, 60, 248, 264$.

In both cases the source and the target nodes are $s = 1$ and $t = n$, respectively, with no loss of generality. It is worth noting that the considered grid networks are acyclic, therefore Algorithm 1 was not applied for such instances. Moreover, the run times of Algorithm 1 were small (at most 4% of the total run times, that is, up to 50 milliseconds), therefore they are not included in the results reported in the following.

We begin by observing that in a non negligible number of the random instances, K disjoint paths can be found. This happens for 274 of the 1056 instances. Table 3.1 exhibits the distribution of those instances.

Table 3.1 Number of instances with K disjoint paths in random networks

	K							
$R_{n,m}$	3	4	5	6	7	8	9	10
$R_{100,500}$	12	2	1	0	0	0	0	0
$R_{100,1000}$	22	18	14	12	7	4	2	1
$R_{300,1500}$	14	7	1	0	0	0	0	0
$R_{300,3000}$	21	18	13	9	7	4	2	1
$R_{500,2500}$	14	4	3	1	1	0	0	0
$R_{500,5000}$	17	14	12	8	5	3	0	0

Unless otherwise stated, hereafter only the instances with no disjoint solutions are considered. There are two reasons for this:

- the main goal of this work is to find good methods to obtain dissimilar paths in networks where finding those paths is not easy (finding disjoint paths can be formulated as a minimum cost flow problem [52, 53]);
- such instances are not evenly distributed throughout the set of instances, thus their presence in different numbers in each group could skew the results.

The drawback of this decision is that each set $R_{n,m}$ now has a smaller and different number of instances. Table 3.2 indicates the final number for each type of instances. It should be remarked that there are no disjoint solutions in the grid instances.

Table 3.2 Final number of random network instances

	K							
$R_{n,m}$	3	4	5	6	7	8	9	10
$R_{100,500}$	10	20	21	22	22	22	22	22
$R_{100,1000}$	0	4	8	10	15	18	20	21
$R_{300,1500}$	8	15	21	22	22	22	22	22
$R_{300,3000}$	1	4	9	13	15	18	20	21
$R_{500,2500}$	8	18	19	21	21	22	22	22
$R_{500,5000}$	5	8	10	14	17	19	22	22

An upper bound of 300 seconds was set for the elapsed time when running the solver. The results presented in the following for random networks are average values for solving the instances with the same characteristics except for a seed, and with no disjoint solutions. The results shown for grid networks are based on a single instance, again with no disjoint solutions.

Approximately 84 % of the random instances were solved to optimality (corresponding to 659 out of 782 instances) and 44 % of the grid instances (corresponding to 14 out of 32 instances).

Table 3.3 Number of instances solved to optimality by `MAO` (%)

	K							
$R_{n,m}$	3	4	5	6	7	8	9	10
$R_{100,500}$	100	100	100	96	77	50	27	23
$R_{100,1000}$	–	100	100	100	100	100	100	95
$R_{300,1500}$	100	100	100	100	95	80	59	41
$R_{300,3000}$	100	100	100	100	100	100	100	86
$R_{500,2500}$	100	100	100	100	86	73	55	32
$R_{500,5000}$	100	100	100	100	100	100	86	82

	K							
$G_{p,q}$	3	4	5	6	7	8	9	10
$G_{3,12}$	100	100	100	100	0	0	0	0
$G_{4,36}$	100	100	0	0	0	0	0	0
$G_{6,6}$	100	100	100	100	0	0	0	0
$G_{12,12}$	100	100	100	100	0	0	0	0

Due to the combinatorial nature of the model, it would be expected that the results of its application depended heavily on the size of the network. However, according to Tables 3.3[1] and 3.4 many of the instances that were not solved within the time limit are actually associated to the smaller networks. In fact, the results behaved as expected only for the first values of $K \leq 4$, for both the random and the grid networks. As K increases, other features seem to have a more decisive influence. A finer analysis of the results allows to enumerate three possible explanations for that situation: the layout of the network; the sparsity of the network; and the relation between the number of paths K and the size of the network.

The differences in the layout of the network account for the fact that grid instances are, in general, harder to solve than random instances, even though they are much smaller (recall that $36 \leq n \leq 144$ and $57 \leq m \leq 264$ for the grid instances, whereas $100 \leq n \leq 500$ and $500 \leq m \leq 5000$ for the random instances). Furthermore, even among grid networks, there are differences related to the layout, as the problem seems to be more difficult to solve in rectangular grids rather than in square grids – see Table 3.4 and Figure 3.3.

Moreover, the effect of the sparseness of the network is particularly clear in the results of the random instances. According to Figure 3.3, the run times decrease as the density of the network[2] increases when $K \geq 5$, in all sets of instances but one, and that this factor overcomes the one of the size of the network. The only exception is the smallest of the instances. In spite of its density, second only to $R_{100,1000}$, $R_{100,500}$ instances have proven to be, in average, the hardest of the random instances to solve. Another interesting conclusion is that the impact of sparseness in grid networks is not so significant as it is in random

[1] There is only one grid instance of each kind, therefore, the listed values are either 0% or 100%.

[2] The density of a network is defined as $m/(n(n-1))$.

Table 3.4 Run times of MAO (seconds)

	K								
$R_{n,m}$	3	4	5	6	7	8	9	10	
$R_{100,500}$	0.124	0.490	1.135	32.149	76.488	175.405	222.937	249.626	
$R_{100,1000}$	–	0.854	3.245	2.910	5.067	10.116	18.692	48.099	
$R_{300,1500}$	0.391	1.611	2.637	10.127	37.499	111.854	174.106	212.429	
$R_{300,3000}$	0.745	1.934	5.063	11.508	23.537	35.912	63.022	132.342	
$R_{500,2500}$	0.793	2.179	4.579	12.781	66.327	121.559	196.307	238.856	
$R_{500,5000}$	1.888	1.491	7.253	17.309	34.013	57.627	118.488	196.436	
Average	0.788	1.426	3.985	14.464	40.488	85.412	132.258	179.631	57.306

	K								
$G_{p,q}$	3	4	5	6	7	8	9	10	
$G_{3,12}$	0.072	2.665	10.150	52.917	300	300	300	300	
$G_{4,36}$	0.218	0.684	300	300	300	300	300	300	
$G_{6,6}$	0.096	0.265	2.776	11.783	300	300	300	300	
$G_{12,12}$	0.240	0.464	5.212	19.772	300	300	300	300	
Average	0.156	1.019	79.534	96.118	300	300	300	300	172.103

networks. This finding becomes evident when comparing the run times for the $G_{4,36}$ and $G_{12,12}$ networks, which have the lowest of the densities of the grid networks.

As for the third of the reasons presented: the proportion between the value of K and the size of the network, finding K dissimilar paths in a given network becomes more difficult as the value of K grows and this difficulty is increased in very small networks. This is a plausible reason why the results obtained for the $R_{100,500}$ cases do not follow the pattern of the remaining random instances. Moreover, it is also expected that this factor has an impact on the results for the grid instances. However, results for more grid networks would be recommended for a sound conclusion.

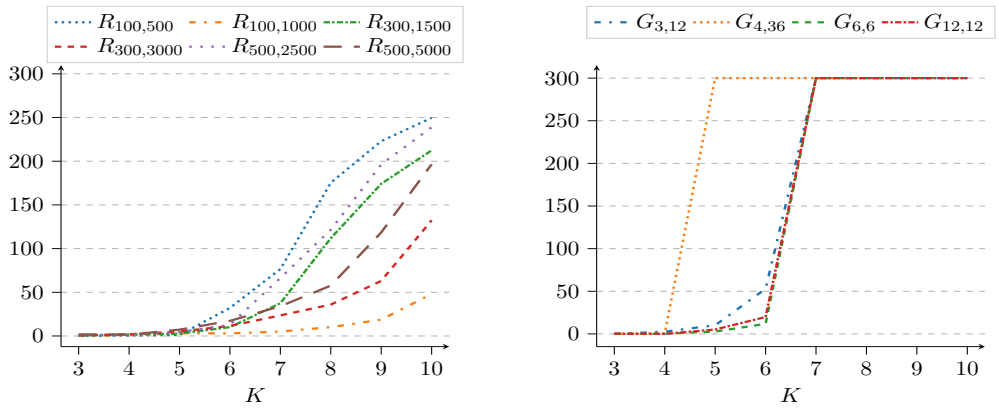


Fig. 3.3 Run times of MAO (seconds)

For the sake of completeness, the lower bounds obtained by solving the linear programming relaxation of the model are also presented. Table 3.5 presents the average integer programming

gaps, as well as the run times for solving the corresponding linear programming relaxations, for each group of instances. The integer programming gaps are computed as $100(f_1^* - f_{LR_1}^*)/|f_1^*|$ %, where f_1^* denotes the optimum value of (3.2) and $f_{LR_1}^*$ denotes the optimum value of its linear programming relaxation. Whenever the optimum value f_1^* is unknown, the best known integer is used to compute these gaps. The gaps associated with the grid instances were all 100%, therefore no such table is presented.

Table 3.5 Average integer programming gaps of $\texttt{MAO}_L$ (%)

	K							
$R_{n,m}$	3	4	5	6	7	8	9	10
$R_{100,500}$	43	73	75	77	78	78	79	79
$R_{100,1000}$	–	100	100	100	100	100	100	100
$R_{300,1500}$	75	89	92	93	93	93	93	93
$R_{300,3000}$	100	100	100	100	100	100	100	100
$R_{500,2500}$	91	96	96	97	97	97	97	97
$R_{500,5000}$	60	75	80	86	87	90	91	91

As it can be observed from Table 3.5, the linear programming relaxation of formulation (3.2) produces very weak lower bounds. This often happens in models that use the same type of linking constraints (3.2e) used in this formulation.

3.3 Minimization of the number of repeated arcs

While taking into account all the overlaps in the pairs of paths in the solutions, the formulation (3.2) is not easy to handle from a practical point of view, as shown by the experiments reported in Section 3.2.1. In the following an alternative, and simpler, approach to problem (3.1) is introduced.

A given arc $(i, j) \in A$ is said to be repeated in a set of K paths if it belongs to more than one of them. A way to ensure that the K paths are sufficiently different from each other is to consider a relaxed version of the K disjoint paths problem, where instead of forbidding the occurrence of repeated arcs, the number of arcs in those conditions is minimized. This concept was explored in the example illustrated in Figure 1.1, given in the introductory section. Because this approach simply privileges the number of repeated paths, it favors the solution depicted in Figure 1.1a, rather than the one in Figure 1.1b.

In order to model such a problem, let us consider, as before, the decision variables x_{ij}^k equal to 1 if the arc (i, j) lies in the k-th path from s to t, or 0 otherwise, for any $(i, j) \in A$ and $k = 1, \ldots, K$. The problem of minimizing the number of repeated arcs can then be

formulated as follows:

$$\min \quad f_2(x,y) = \sum_{(i,j)\in A} y_{ij} \tag{3.4a}$$

$$\text{subject to} \quad \sum_{j\in N:(i,j)\in A} x_{ij}^k - \sum_{j\in N:(j,i)\in A} x_{ji}^k = \begin{cases} 1 & i = s \\ 0 & i \neq s,t \\ -1 & i = t \end{cases}, \quad k = 1,\dots,K \tag{3.4b}$$

$$y_{ij} \leq \sum_{k=1}^{K} x_{ij}^k, \quad (i,j) \in A \tag{3.4c}$$

$$(K-1)y_{ij} \geq \sum_{k=1}^{K} x_{ij}^k - 1, \quad (i,j) \in A \tag{3.4d}$$

$$x_{ij}^k \in \{0,1\},\ y_{ij} \in \{0,1\}, \quad (i,j) \in A, \quad k = 1,\dots,K \tag{3.4e}$$

This formulation has $O(Km)$ binary variables and $O(Kn+m)$ constraints. The constraints (3.4b) are flow conservation constraints that model K paths from node s to node t. The constraints (3.4c) and (3.4d) relate the x and the y variables, in a way that y_{ij} is 1 if and only if the arc (i,j) is used in more than one path, that is, if this arc is repeated, and 0 otherwise. In fact, given the arc $(i,j) \in A$:

1. If $x_{ij}^k = 0$ for every $k = 1,\dots,K$, then by (3.4c) we have $y_{ij} = 0$, whereas (3.4d) has no implications on the value of y_{ij}.
2. If $x_{ij}^k = 1$ for exactly one $k \in \{1,\dots,K\}$, then neither (3.4c) nor (3.4d) have implications on the value of y_{ij}.
3. If $x_{ij}^k = 1$ for more than one $k \in \{1,\dots,K\}$, then (3.4c) has no implications on the value of y_{ij}, whereas by (3.4d) we have $y_{ij} = 1$.

When in situation 2., that is, if only one variable x_{ij}^k has value 1, the value of y_{ij} can be arbitrary. However, the objective function minimizes the sum of all these variables and this minimization is achieved if the arbitrary y_{ij}'s are equal to 0, as intended. Therefore, the objective function counts the number of repeated arcs, that is the number of arcs that are used more than once.

The same reasoning can be used to prove that the set of constraints (3.4c) can be dropped. In fact, because the goal of this formulation is to minimize $\sum_A y_{ij}$, the values of y_{ij} are 0 by default.

Just like for the formulation presented in the previous section, an optimal solution of formulation (3.4) may contain loops, as long as they do not include any arc that is common to several paths. However, given any such optimal solution, Algorithm 1 can be applied in order to remove its loops. The arguments for proving Proposition 3.3.1 are similar to those used for Proposition 3.2.2, therefore its proof is omitted.

Proposition 3.3.1 *Let (x,y) be an optimal solution for problem (3.4). Let $\bar{x} \in \{0,1\}^{Km}$ be the corresponding vector output by Algorithm 1 when applied to x, and $\bar{y} \in \{0,1\}^m$ be such*

that the constraints (3.4c) and (3.4d) are satisfied. Then, $(\bar{x}, \bar{y})$ *is a loopless optimal solution for problem (3.4).*

3.3.1 Computational experiments

In the following, formulation (3.4), which minimizes the number of repeated arcs, is analyzed empirically. The code that implements this formulation is designated by `MRA`. It was written in C and uses IBM ILOG CPLEX version 12.7 to solve the integer programs. The variant of the same formulation obtained by removing the constraints (3.4c) was also implemented. Note that both models are valid formulations of the problem, the latter being a weaker (in terms of its linear programming bound) but smaller variant of the first. Because the differences in the run times obtained with both variants were not significant, only the original one is presented below.

The experimental setup was as described in Section 3.2.1. Algorithm 1 was applied to the results of the code `MRA` to remove the loops from the obtained solutions in the random instances.

Table 3.6 Number of instances solved to optimality by `MRA` (%)

	K							
$G_{p,q}$	3	4	5	6	7	8	9	10
$G_{3,12}$	100	100	100	100	100	100	100	100
$G_{4,36}$	100	100	100	100	100	0	0	0
$G_{6,6}$	100	100	100	100	100	100	100	100
$G_{12,12}$	100	100	100	100	100	0	0	0

`MRA` was able to solve to optimality all the random network instances, within the 300 seconds time window. Additionally, Table 3.6 shows the same number but for grids. It was possible to solve 81% of these instances, corresponding to 26 out of the 32 instances. The interruptions after 300 seconds occurred in problems of finding 8 or more paths in the 144 node grids. Finally, it is worth noting that even though formulations (3.2) and (3.4) model different problems it is still of interest to compare them, since they both are relaxations of the K dissimilar paths problem, and that far more instances were solved by `MRA` than by `MAO`.

The run times of code `MRA` are summarized in Table 3.7 and depicted in Figure 3.4. The results were particularly sensitive to the number of paths. However, other features have negative repercussions as well.

First, the results depend greatly on the layout of the network. In fact, unsolved instances were only found solo in the case of grids. Furthermore, the magnitude of the run times associated to the solved grid instances is several times higher than the run times for solving the random instances (up to 221.5 seconds in the former case and to 2.2 seconds in the latter). Another evidence of the difficulty associated to solving the grid instances, is the fact that between 4 and 256 arcs have to be repeated in these instances, against between 1 and 24 for random networks. In this way, both formulations (3.2) and (3.4) are very sensitive to the layout of the network and both work worse in the case of grid networks. In the case of

Table 3.7 Run times of `MRA` (seconds)

	K								
$R_{n,m}$	3	4	5	6	7	8	9	10	
$R_{100,500}$	0.051	0.061	0.098	0.173	0.254	0.657	0.981	2.092	
$R_{100,1000}$	–	0.108	0.145	0.189	0.206	0.351	0.373	0.385	
$R_{300,1500}$	0.138	0.204	0.279	0.312	0.432	0.539	1.057	2.144	
$R_{300,3000}$	0.280	0.347	0.411	0.485	0.614	0.723	0.855	0.949	
$R_{500,2500}$	0.256	0.323	0.457	0.557	0.723	0.964	1.458	1.586	
$R_{500,5000}$	0.394	0.542	0.770	0.944	1.074	1.271	1.488	1.836	
Average	0.223	0.261	0.360	0.443	0.550	0.750	1.053	1.498	0.640

	K								
$G_{p,q}$	3	4	5	6	7	8	9	10	
$G_{3,12}$	0.054	0.154	0.455	0.859	0.696	0.902	1.078	1.235	
$G_{4,36}$	0.103	0.107	1.833	5.047	221.466	300	300	300	
$G_{6,6}$	0.019	0.093	0.327	1.200	11.392	3.075	2.176	1.441	
$G_{12,12}$	0.166	0.239	1.070	4.359	47.101	300	300	300	
Average	0.085	0.148	0.921	2.866	70.163	150.994	150.813	150.669	65.832

the grid networks, the run times of `MRA` vary both with the shape of the grid and with K. Further conclusions would require more exhaustive tests.

Second, as made evident by analyzing Table 3.7 and Figure 3.4, the pattern behavior of the `MRA` run times for random networks changes for $K \geq 7$. Prior to that value, the run times vary with the size of the networks; afterwards other factors become dominant. This type of behavior was identified in Section 3.2.1, when analyzing `MAO` results. Then, the density of the network and the proportion between the value of K and the size of the network were identified as determinant factors of the observed deviation. The corresponding `MRA` results, seem to be affected in a similar way – the density of the network smooths the growth of the run times, whereas the increase of K causes abrupt increases from a certain threshold for the $R_{100,500}$ and $R_{300,1500}$ instances.

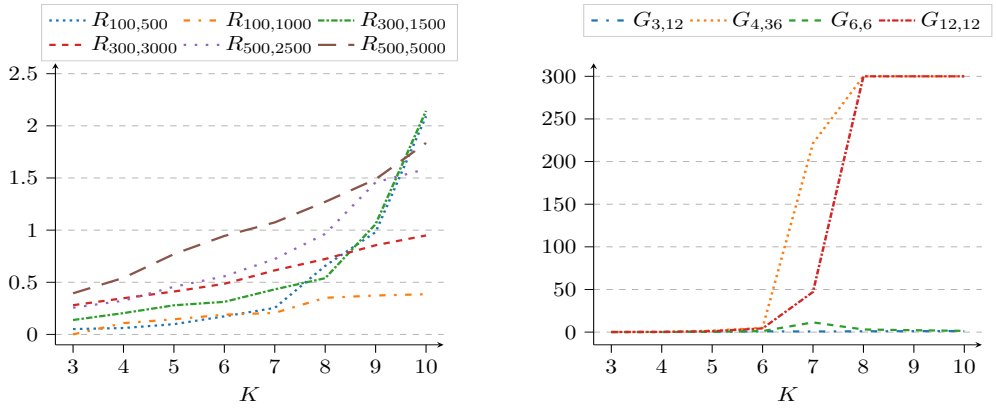


Fig. 3.4 Run times of `MRA` (seconds)

Table 3.8 Average integer programming gaps of $\texttt{MRA}_L$ (%)

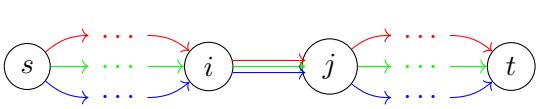
	K							
$R_{n,m}$	3	4	5	6	7	8	9	10
$R_{100,500}$	10	40	36	38	36	42	43	43
$R_{100,1000}$	–	12	23	25	40	46	48	49
$R_{300,1500}$	14	33	45	41	39	39	47	47
$R_{300,3000}$	2	11	25	35	38	44	49	47
$R_{500,2500}$	17	42	35	37	35	42	45	45
$R_{500,5000}$	7	14	17	30	38	41	48	42

	K							
$G_{p,q}$	3	4	5	6	7	8	9	10
$G_{3,12}$	50	61	50	46	41	37	35	32
$G_{4,36}$	50	67	72	59	51	46	41	37
$G_{6,6}$	50	67	67	70	70	66	60	56
$G_{12,12}$	50	67	67	70	70	71	71	72

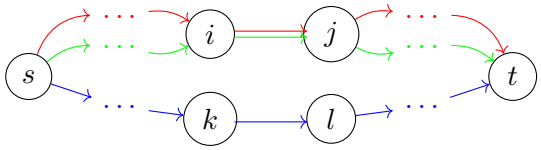
Finally, Table 3.8 presents the average integer programming gaps determined by the lower bound produced by the linear relaxation of the formulation (3.4). These gaps are computed as explained in Section 3.2.1. In the random instances the gap values are at most 49%. The gaps are even bigger for the grid networks, between 32% and 72%.

3.4 Minimization of the number of arc repetitions

The goal of formulation (3.4), presented in the previous section, is to minimize the number of arcs which are repeated in the solutions. The undesired effect of this single objective may be that few arcs appear in many different paths. This situation is illustrated in Figure 1.1. Another example is depicted by the two sets of $K = 3$ paths in Figure 3.5. Both the solutions in Figure 3.5 have a single repeated arc. However, the paths in the solution in Figure 3.5b are more dissimilar than the paths in Figure 3.5a, the reason being that in the first case the arc (i, j), which is repeated, appears only twice, while it appears in all the three paths in the latter case. In the following two approaches are presented which intend to model a more complete understanding of how dissimilar paths should look like.



(a) Solution with one repeated arc, shared by several paths



(b) Solution with two repeated arcs, shared by few paths

Fig. 3.5 Different sets of $K = 3$ paths

According to the example above, the way how paths spread in a network is affected by the number of repeated arcs as well as by the number of times that the repeated arcs appear in the paths. This issue will be addressed with two approaches in the following. First, by minimizing the number of times the repeated arcs appear in the set of the paths. Later, by minimizing the number of times they are repeated, which accounts also for the number of paths where they appear.

The number of occurrences of the arc $(i,j) \in A$ in a set of K paths P_K is defined as:

$$Occ(i,j;P_K) = \begin{cases} 0 & \text{if } |\{p \in P_k : (i,j) \in p\}| \leq 1 \\ |\{p \in P_k : (i,j) \in p\}| & \text{otherwise} \end{cases}$$

and the number of repeated arc occurrences in P_k is given by

$$RO(P_K) = \sum_{(i,j)\in A} Occ(i,j;P_K).$$

This value is $RO(P_3) = 3$ in Figure 3.5a, and $RO(P_3') = 2$ in Figure 3.5b, which makes the solution in the second plot better than the first with respect to the number of repeated arc occurrences. The purpose of the next formulation is to find a set of K paths which minimizes the number of repeated arc occurrences, $RO(P_K)$.

Let x_{ij}^k be decision variables defined as before and let us consider the formulation:

$$\min \quad f_3(x,y,u) = \sum_{(i,j)\in A} u_{ij} \tag{3.5a}$$

$$\text{subject to} \quad \sum_{j\in N:(i,j)\in A} x_{ij}^k - \sum_{j\in N:(j,i)\in A} x_{ji}^k = \begin{cases} 1 & i = s \\ 0 & i \neq s,t \\ -1 & i = t \end{cases}, \quad k = 1,\ldots,K \tag{3.5b}$$

$$y_{ij} \leq \sum_{k=1}^{K} x_{ij}^k, \quad (i,j) \in A \tag{3.5c}$$

$$(K-1)y_{ij} \geq \sum_{k=1}^{K} x_{ij}^k - 1, \quad (i,j) \in A \tag{3.5d}$$

$$u_{ij} \leq K\, y_{ij}, \quad (i,j) \in A \tag{3.5e}$$

$$u_{ij} \leq \sum_{k=1}^{K} x_{ij}^k, \quad (i,j) \in A \tag{3.5f}$$

$$u_{ij} \geq y_{ij} + \sum_{k=1}^{K} x_{ij}^k - 1, \quad (i,j) \in A \tag{3.5g}$$

$$x_{ij}^k \in \{0,1\}, y_{ij} \in \{0,1\}, u_{ij} \in \mathbb{N}_0, \quad (i,j) \in A, \quad k = 1,\ldots,K \tag{3.5h}$$

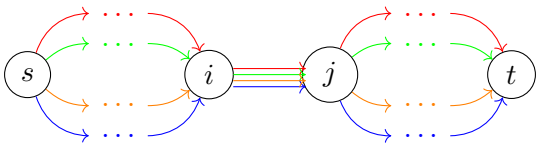
This formulation has $O(Km)$ variables and $O(Kn+m)$ constraints. For any $(i,j) \in A$, the variable y_{ij} is defined as in the previous formulation. Additionally, for a given $(i,j) \in A$:

- If $x_{ij}^k = 0$ for any $k = 1,\ldots,K$, then the constraints (3.5f) imply that $u_{ij} = 0$.

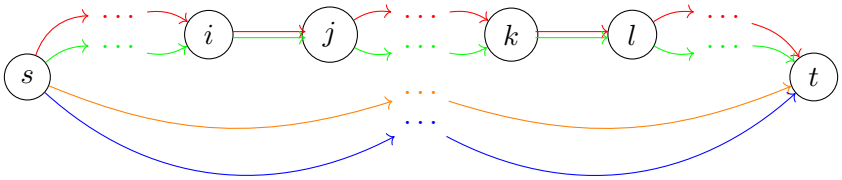
- If $x_{ij}^k = 1$ for exactly one $k \in \{1,\ldots,K\}$, then by constraints (3.5f), $u_{ij} \leq 1$, and by constraints (3.5g), $u_{ij} \geq 0$. Because the goal of the problem is to minimize $\sum_A u_{ij}$, then $u_{ij} = 0$.

- If $x_{ij}^k = 1$ for more than one $k \in \{1, \ldots, K\}$, because in the last section we saw that then $y_{ij} = 1$, then by constraints (3.5f), $u_{ij} \leq \sum_{k=1}^{K} x_{ij}^k$, and by constraints (3.5g), $u_{ij} \geq \sum_{k=1}^{K} x_{ij}^k$. Combining the two conditions implies that $u_{ij} = \sum_{k=1}^{K} x_{ij}^k$.

Thus, the variable u_{ij} counts the number of times the arc (i, j) appears in the solution, or is equal to 0 if (i, j) is not repeated in that solution, for any $(i, j) \in A$. According to these points it can also be concluded that the variables u_{ij} can be relaxed as $u_{ij} \geq 0$, without changing the solution. The constraints (3.5e) and (3.5f) can be dropped, because by default the minimization of the objective function implies that $u_{ij} = 0$. Moreover, also the constraints (3.5c) can be skipped because the variables y_{ij} are only useful when the arc (i, j) appears in more than one path and this constraint is not affected in that case.



(a) Solution with one repeated arc, shared by several paths



(b) Solution with two repeated arcs, shared by one path

Fig. 3.6 Different sets of $K = 4$ paths

In the case shown in Figure 3.6, counting the number of times that the repeated arcs appear in the solution is not enough to distinguish between the two depicted solutions. In fact, Figures 3.6a and 3.6b we have $RO(P_4) = RO(P_4') = 4$, where the repeated arcs are (i, j) in the first case, and (i, j) and (k, l) in the second. Nevertheless, the fact that the repetitions happen in different arcs should be valued, given that this is reflected in the dissimilarity of these two sets of paths. Therefore, a new concept will be introduced, the number of repetitions for any arc that appears in the solution more than once.

Let the number of repetitions of an arc $(i, j) \in A$ in the set P_K be the number of paths it belongs to, excluding its first utilization, that is,

$$Rep(i, j; P_K) = \begin{cases} 0 & \text{if } |\{p \in P_k : (i, j) \in p\}| \leq 1 \\ Occ(i, j; P_K) - 1 & \text{otherwise} \end{cases}$$

Then, the number of arc repetitions in P_k is given by

$$Rep(P_K) = \sum_{(i,j) \in A} Rep(i, j; P_K).$$

The number of arc repetitions defined above reflects two aspects: the number of arcs shared by more than one paths as well as the number of paths that share them. Recalling the example in Figure 1.1, for the solution P_4 in Figure 1.1a we have $Rep(P_4) = 3$, because the arc (i, j) is repeated 3 times, whereas for the solution P_4' in Figure 1.1b we have $Rep(P_4') = 2$, because both the arcs (i, j) and (k, l) are repeated once. The next formulation aims at minimizing $Rep(P_K)$.

Like before, for modeling the problem of finding K paths from s to t which minimize the number of arc repetitions, the variables x_{ij}^k represent K paths and have value 1 if the arc (i, j) is in the k-th path from s to t or are 0 otherwise, for any $(i, j) \in A$. The formulation is as follows:

$$\min \quad f_4(x, w, u) = \sum_{(i,j)\in A} u_{ij} \tag{3.6a}$$

$$\text{subject to} \quad \sum_{j\in N:(i,j)\in A} x_{ij}^k - \sum_{j\in N:(j,i)\in A} x_{ji}^k = \begin{cases} 1 & i = s \\ 0 & i \neq s, t \\ -1 & i = t \end{cases}, \quad k = 1, \ldots, K \tag{3.6b}$$

$$x_{ij}^k \leq w_{ij}, \quad (i, j) \in A, \quad k = 1, \ldots, K \tag{3.6c}$$

$$w_{ij} \leq \sum_{k=1}^{K} x_{ij}^k, \quad (i, j) \in A \tag{3.6d}$$

$$u_{ij} = \sum_{k=1}^{K} x_{ij}^k - w_{ij}, \quad (i, j) \in A \tag{3.6e}$$

$$x_{ij}^k \in \{0, 1\},\ w_{ij} \in \{0, 1\},\ u_{ij} \geq 0 \quad (i, j) \in A, \quad k = 1, \ldots, K \tag{3.6f}$$

This formulation has $O(Km)$ variables and $O(K(m + n))$ constraints. The constraints (3.6b) are flow conservation constraints that define a set of K paths from node s to node t. The constraints (3.6c) and (3.6d) are used to define the variables $w_{ij} \in \{0, 1\}$, each one equal to 1 if and only if the arc (i, j) is used in at least one path, or 0 otherwise, for any $(i, j) \in A$. Additionally, the constraints (3.6e) define the auxiliary variables u_{ij}, which corresponds to the number of times that arc $(i, j) \in A$ is repeated in different paths. For a given $(i, j) \in A$:

- If $x_{ij}^k = 0$ for any $k = 1, \ldots, K$, then the constraints (3.6d) imply that $w_{ij} = 0$. Therefore the constraints (3.6e) imply that $u_{ij} = 0$.
- If $x_{ij}^k = 1$ for exactly one $k \in \{1, \ldots, K\}$, then by constraints (3.6c), $w_{ij} \geq 1$ which together with (3.6f) imply $w_{ij} = 1$, and by constraints (3.6e), $u_{ij} = 0$.
- If $x_{ij}^k = 1$ for more than one $k \in \{1, \ldots, K\}$, then by constraints (3.6c), $w_{ij} \geq 1$ which together with (3.6f) imply that $w_{ij} = 1$, and by constraints (3.6e), $u_{ij} = \sum_{k=1}^{K} x_{ij}^k - 1$.

Because both x_{ij}^k and w_{ij} are binary variables, the variables u_{ij} are implicitly defined as integers for any $(i, j) \in A$, $k = 1, \ldots, K$. Additionally, (3.6e) together with the non-negative constraints of the variables u_{ij} imply (3.6d). Therefore, the constraints (3.6d) can be skipped from the formulation.

Finally, we observe that constraints (3.6c) can be aggregated as

$$\sum_{k=1}^{K} x_{ij}^{k} \leq K\, w_{ij}, \quad (i,j) \in A. \tag{3.7}$$

Like for formulations (3.2) and (3.4), both formulations (3.5) and (3.6) admit optimal solutions with loops. However, in such cases Proposition 3.4.1 holds and the Algorithm 1 can be applied in order to obtain loopless optimal solutions.

Proposition 3.4.1 *1. Let (x, y, u) be an optimal solution for problem (3.5). Let $\bar{x} \in \{0,1\}^{Km}$ be the vector output by Algorithm 1 when applied to x, $\bar{y} \in \{0,1\}^{m}$ and $\bar{u} \in \mathbb{N}_0^m$ be such that the constraints (3.5c) to (3.5h) are satisfied. Then, $(\bar{x}, \bar{y}, \bar{u})$ is a loopless optimal solution for problem (3.5).*

2. Let (x, w, u) be an optimal solution for problem (3.6). Let $\bar{x} \in \{0,1\}^{Km}$ be the vector output by Algorithm 1 when applied to x, and $\bar{w} \in \{0,1\}^{m}$ and $\bar{u}$ be such that the constraints (3.6c) to (3.6f) are satisfied. Then, $(\bar{x}, \bar{w}, \bar{u})$ is a loopless optimal solution for problem (3.6).

3.4.1 Computational experiments

The tests setup presented in the following for experimentally assessing formulations (3.5) and (3.6) is similar to what was described in Section 3.2.1.

While a number of relaxations of formulation (3.5) were tested, only the results for the one with the best behavior with regard to the run times are reported. These correspond to the variant that omits the set of constraints (3.5c) from the original model, (3.5). For simplicity, we keep the same designation and in the following refer to the new model as formulation (3.5). Likewise, several variants of (3.6) were tested. The model obtained from (3.6) by replacing constraints (3.6c) with its aggregated version, constraints (3.7), was significantly faster than the others. Thus, only the results for this new model are presented. Again, for simplicity, we keep the same designation and hereafter refer to this new model as formulation (3.6).

The following codes were written in C, while using IBM ILOG CPLEX version 12.7 for solving the integer programs:

- `MRO`: Implementation for minimizing the number of repeated arc occurrences, formulation (3.5).
- `MAR`: Implementation for minimizing the number of arc repetitions, formulation (3.6).

Unlike the code `MAR`, which was able to find the optimal solution for all the instances, the code `MRO` resumed after the 300 seconds limit for the 4×36 grids when seeking for more than 7 paths. This can be seen in Table 3.9, which reports their average run times, depicted in Figure 3.7.

The code `MAR` outperformed the previous in almost all cases in terms of run time. Figure 3.8 illustrates the results summarized in Table 3.10. The major differences are found in the

Table 3.9 Run times of MRO (seconds)

	K								
$R_{n,m}$	3	4	5	6	7	8	9	10	
$R_{100,500}$	0.051	0.067	0.108	0.183	0.274	0.572	0.836	1.322	
$R_{100,1000}$	–	0.123	0.157	0.220	0.238	0.307	0.386	0.497	
$R_{300,1500}$	0.161	0.214	0.253	0.331	0.439	0.706	0.998	1.601	
$R_{300,3000}$	0.303	0.370	0.444	0.508	0.677	0.724	0.846	1.000	
$R_{500,2500}$	0.280	0.382	0.485	0.594	0.757	1.035	1.326	1.687	
$R_{500,5000}$	0.486	0.603	0.746	0.906	1.083	1.285	1.500	1.781	
Average	0.256	0.293	0.365	0.457	0.578	0.771	0.892	1.314	0.627

	K								
$G_{p,q}$	3	4	5	6	7	8	9	10	
$G_{3,12}$	0.037	0.251	0.815	1.041	5.496	15.547	8.673	11.894	
$G_{4,36}$	0.094	0.293	2.204	43.209	140.598	300	300	300	
$G_{6,6}$	0.165	0.009	0.276	0.752	2.409	3.930	23.715	14.426	
$G_{12,12}$	0.075	0.274	0.880	3.660	15.674	9.391	87.845	146.961	
Average	0.092	0.206	1.043	12.165	41.044	82.217	150.058	118.320	45.018

results associated to the grid networks (where run times fall, in average, by 99.6%) and in the subset of the random networks previously identified as the harder to solve (which has reductions of 48% for the $R_{100,500}$ instances, 41% for the $R_{300,1500}$ instances and 25% for the $R_{500,2500}$ instances). Still, more than emphasizing the relative reductions towards MRO, it is important to point out that MAR solved all instances, in less than 2 seconds. Moreover, results suggest that MAR is far less susceptible to the effect of the variables that have been identified as inhibiting to other models (the layout of the network, its sparseness and the proportion between the value of K and the size of the network), indicating that this model can be used in a wider range of situations. As a final remark, there was some instability in the run time of the grid networks that should be clarified in a future work.

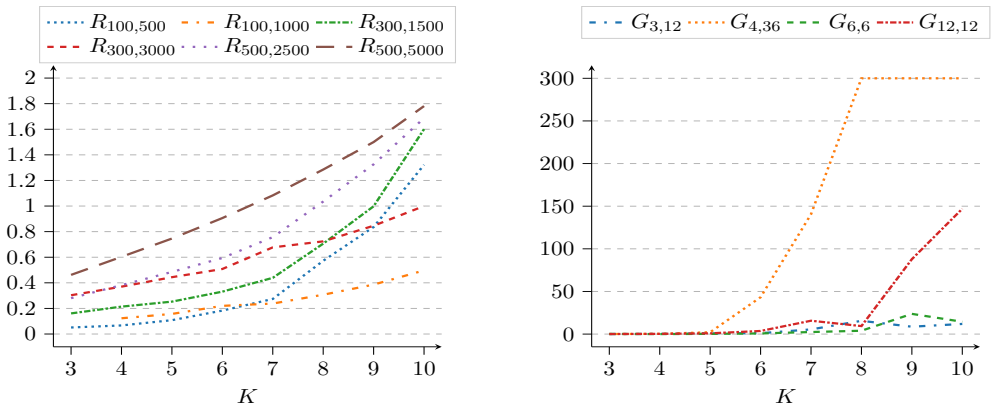


Fig. 3.7 Run times of MRO (seconds)

Table 3.10 Run times of MAR (seconds)

	K								
$R_{n,m}$	3	4	5	6	7	8	9	10	
$R_{100,500}$	0.051	0.078	0.116	0.172	0.181	0.301	0.342	0.519	
$R_{100,1000}$	–	0.097	0.140	0.143	0.193	0.265	0.236	0.316	
$R_{300,1500}$	0.139	0.193	0.222	0.259	0.313	0.426	0.486	0.717	
$R_{300,3000}$	0.226	0.323	0.391	0.458	0.609	0.706	0.773	0.905	
$R_{500,2500}$	0.249	0.318	0.374	0.442	0.593	0.733	1.003	1.172	
$R_{500,5000}$	0.384	0.571	0.825	1.024	1.211	1.369	1.576	1.730	
Average	0.209	0.263	0.344	0.416	0.516	0.633	0.736	0.893	0.501

	K								
$G_{p,q}$	3	4	5	6	7	8	9	10	
$G_{3,12}$	0.010	0.016	0.012	0.017	0.020	0.021	0.023	0.054	
$G_{4,36}$	0.047	0.083	0.126	0.157	0.215	0.250	0.620	0.382	
$G_{6,6}$	0.010	0.004	0.114	0.012	0.018	0.214	0.157	0.020	
$G_{12,12}$	0.034	0.049	0.142	0.242	0.124	0.383	0.323	1.292	
Average	0.025	0.038	0.098	0.107	0.094	0.217	0.280	0.437	0.162

The average integer programming gaps produced by the linear relaxation of (3.5), in Table 3.11, range between 1% and 18% for the random instances and between 7% and 33% for the grid instances. The same values were all 0 for the linear relaxation of formulation (3.6).

3.5 Bounding the number of arc presences

Formulation (3.4) aims at minimizing the number of arcs which appear in more than one path. The undesired consequence of the simplicity of this objective function may be that few arcs appear in many different paths, a situation which is illustrated in Figure 1.1. This

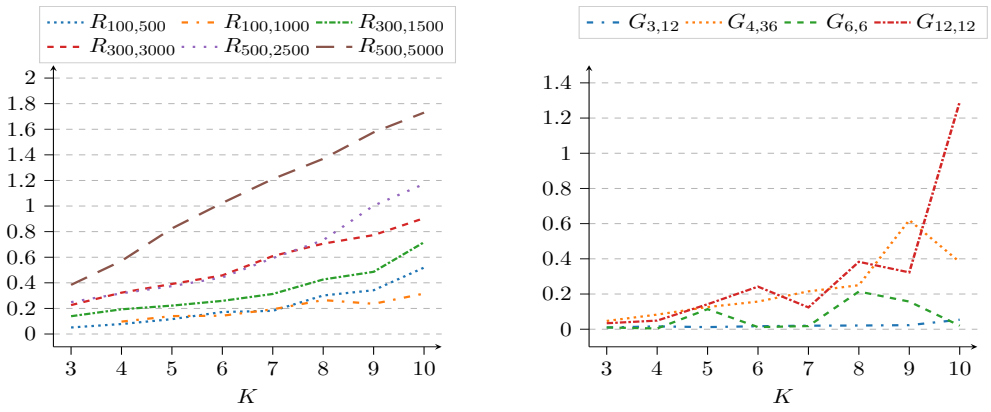


Fig. 3.8 Run times of MAR (seconds)

Table 3.11 Average integer programming gaps of $\texttt{MRO}_L$ (%)

	K							
$R_{n,m}$	3	4	5	6	7	8	9	10
$R_{100,500}$	5	18	12	11	8	9	9	10
$R_{100,1000}$	–	6	10	9	15	15	14	12
$R_{300,1500}$	7	15	18	12	10	8	11	11
$R_{300,3000}$	1	5	11	13	12	14	14	12
$R_{500,2500}$	8	19	11	11	9	11	12	11
$R_{500,5000}$	3	6	7	11	13	12	14	10

	K							
$G_{p,q}$	3	4	5	6	7	8	9	10
$G_{3,12}$	25	28	17	13	11	10	9	9
$G_{4,36}$	25	33	36	21	14	11	8	7
$G_{6,6}$	25	33	29	28	25	24	22	21
$G_{12,12}$	25	33	29	28	25	24	22	21

was the motivation to consider the number of times that each repeated arc appears in the objective function of formulations (3.5) and (3.6), presented in Section 3.4. In the following we propose an intermediate solution, consisting of overcoming this handicap by adding a constraint over the number of times that each arc is present in the solution. In addition, we found that this approach also had an interesting side effect in the paths dissimilarity of the solutions generated by formulations (3.5) and (3.6): when in the presence of multiple optimal solutions with respect to the number of arc repetitions, bounding the number of times that each arc appears gives an extra condition for untying those solutions, thus increasing their dissimilarity. Therefore, the new set of constraints will also be considered in the context of formulations (3.5) and (3.6). Naturally, on the downside, the new models may be more time consuming.

The new set of constraints are very similar to the set of constraints proposed by [13] to prevent repetitions of the arcs over the time horizon. However, whereas in [13] the bound is an external parameter, in our case the bound is fixed by solving a simple problem that optimizes the worst case in terms of the number of times that each arc is present in the solution.

The new formulation aims at finding a set of K paths with the minimum maximum number of arc presences. It is as follows

$$\min \quad \max_{(i,j)\in A} \left\{ \sum_{k=1}^{K} x_{ij}^k \right\} \tag{3.8a}$$

$$\text{subject to} \quad \sum_{j\in N:(i,j)\in A} x_{ij}^k - \sum_{j\in N:(j,i)\in A} x_{ji}^k = \begin{cases} 1 & i = s \\ 0 & i \neq s,t \\ -1 & i = t \end{cases}, \quad k = 1,\ldots,K \tag{3.8b}$$

$$x_{ij}^k \in \{0,1\}, \quad (i,j) \in A, \quad k = 1,\ldots,K \tag{3.8c}$$

where the decision variables are $x_{ij}^k \in \{0,1\}$ equal to 1 if and only if the arc (i,j) appears in path k, $(i,j) \in A$, $k = 1,\ldots,K$. This formulation can be linearized as

$$\min \quad r \tag{3.9a}$$

$$\text{subject to} \quad \sum_{j \in N:(i,j)\in A} x_{ij}^k - \sum_{j \in N:(j,i)\in A} x_{ji}^k = \begin{cases} 1 & i = s \\ 0 & i \neq s,t \\ -1 & i = t \end{cases}, \quad k = 1,\ldots,K \tag{3.9b}$$

$$r \geq \sum_{k=1}^{K} x_{ij}^k, \quad (i,j) \in A \tag{3.9c}$$

$$x_{ij}^k \in \{0,1\}, \quad (i,j) \in A, \quad k = 1,\ldots,K \tag{3.9d}$$

which is equivalent to its linear programming relaxation.

Now, let R^* be the optimal value for problem (3.9) computed in advance. Then a new constraint can be added to formulations (3.4), (3.5) and (3.6) in order to prevent the number of arc presences from exceeding that value,

$$\sum_{k=1}^{K} x_{ij}^k \leq R^*, \quad (i,j) \in A. \tag{3.10}$$

The problems modeled by the resulting formulations are constrained and, therefore, different versions of the original ones. To assess whether the new problems produce better solutions to the K dissimilar paths problem, a set of computational experiments was performed. The values of R^* for random and grid networks are reported on Table 3.12. Results are discussed in Sections 3.5.1 and 3.6.2.

3.5.1 Computational experiments

In this section we analyze the impact of adding the constraints (3.10) to formulations (3.4), (3.5) and (3.6) on the run times. Considering the experimental setup described in Section 3.2, the following codes were tested:

- `MRAA`: implementation of formulation (3.4) including the constraints (3.10);
- `MROA`: implementation of formulation (3.5) including the constraints (3.10);
- `MARA`: implementation of formulation (3.6) including the constraints (3.10).

Like before, the codes were written in C, calling the integer programming solver IBM ILOG CPLEX version 12.7. The impact of adding the constraints (3.10) to the formulations introduced before on the several parameters is measured as $100 \times (\texttt{MA} - \texttt{M})/\texttt{M}$ %, where `M` stands for each of the codes listed above.

Of the new codes, `MARA` was the only one able of finding the optimal solution for all the instances within the time limit of 300 seconds. Both `MRAA` and `MROA` resumed after that limit for the 4×36 grids and $K = 10$.

According to Table 3.13, in most cases the constrained problems require more time to solve when the networks are denser, while the run times do not change much for the sparser instances. In some of the latter cases there is even a speed up. The speed up happens mostly for `MRA` and `MRO`, and is particularly relevant in grids, which are instances where finding solutions is difficult. It should be added that the run times required for solving the problem (3.9) are included in the values on Table 3.13. The results regarding the run times for the grid networks were uneven.

The integer programming gap associated with `MARA` was equal to 0 in all the tested instances. In general these values increased for the remaining formulations after adding the new constraints, specially in the random instances and in the smaller grids and big K's. However, this variation is not very meaningful, as the unconstrained and the constrained problems are different.

Table 3.12 Average of R^* value

Instance	R^*
$R_{100,500}$	5.52
$R_{100,1000}$	2.75
$R_{100,1500}$	1.90
$R_{500,2500}$	4.68
$R_{500,5000}$	3.50
$R_{500,7500}$	2.00

Instance	R^*
$G_{12,12}$	5.00
$G_{4,36}$	5.00
$G_{15,15}$	5.00
$G_{5,45}$	5.00

Table 3.13 Run times variation for MRAA, MROA and MARA (%)

	MRAA								MROA								MARA							
	K								K								K							
$R_{n,m}$	3	4	5	6	7	8	9	10	3	4	5	6	7	8	9	10	3	4	5	6	7	8	9	10
$R_{100,500}$	32	56	38	25	19	-14	-17	-50	29	28	17	-14	2	-29	-30	-39	42	43	35	20	73	40	48	23
$R_{100,1000}$	–	64	50	38	71	42	59	56	–	38	22	8	37	15	10	9	–	83	75	87	77	49	97	86
$R_{300,1500}$	48	39	34	44	47	37	-3	-21	29	25	40	27	28	-2	-7	-23	58	60	67	80	92	67	66	42
$R_{300,3000}$	48	62	66	66	69	72	70	584	27	40	40	45	41	51	55	532	71	76	74	79	73	76	90	614
$R_{500,2500}$	50	49	45	41	56	52	11	33	31	22	25	24	35	43	10	9	60	63	79	78	81	104	56	55
$R_{500,5000}$	44	39	52	52	79	80	253	959	25	34	48	45	62	61	230	982	53	56	55	50	70	79	239	1031

	MRAA								MROA								MARA							
	K								K								K							
$G_{p,q}$	3	4	5	6	7	8	9	10	3	4	5	6	7	8	9	10	3	4	5	6	7	8	9	10
$G_{3,12}$	−13	−85	−70	−70	−61	−43	−22	−27	−43	−90	−81	−80	−95	−93	−90	−41	61	42	81	60	47	73	67	72
$G_{4,36}$	−24	9	7	294	−81	−99	−98	0	−36	−65	36	−90	−74	−97	−98	0	57	35	22	20	7	10	−43	−2
$G_{6,6}$	16	−87	−68	−92	−97	−84	14	−24	−90	79	−86	−94	−53	−91	−93	−93	78	273	−79	103	558	−86	−47	82
$G_{12,12}$	−56	−66	−57	−95	−97	−99	−98	−98	−44	−78	−77	−86	−95	−87	−95	−99	86	31	43	−46	348	−28	73	−14

3.6 Application to finding K dissimilar paths

In the previous sections, integer formulations were presented for the problems of finding K paths between a given pair of nodes, such that:

- the number of arc overlaps for each pair of paths is minimized;
- the number of repeated arcs is minimized;
- the number of occurrences of repeated arcs, or the number of arc repetitions, is minimized.

These problems were suggested with the purpose of capturing characteristics of sets of K dissimilar paths. Therefore in the current section the behavior of the presented approaches is discussed and compared from the perspective of the dissimilarity, based on the metric D_1 defined in Chapter 2. In addition, these approaches are also compared with the classical method known as the IPM and proposed in [33]. As mentioned in Chapter 2, the idea behind this method is to solve K shortest path problems and penalize the cost of the selected arcs every time one of those paths is computed, in order to prevent their overlap as much as possible. This approach has often been used for comparisons in the literature due to its flexibility to incorporate features of various problems as well as the simplicity of its implementation.

While a relative comparison of the dissimilarities produced by the different approaches is possible, an absolute assessment of the results requires the optimal value of D_1 to be known for each instance. As that is the case only for some of the grid instances, the analysis is based on a description of the results of each model followed by a relative overall comparison. For that purpose, both the average dissimilarity between the pairs of paths in the solution (`AvDi`) and their minimum dissimilarity (`MiDi`) are calculated for each instance. Then, the averages of `AvDi` and of `MiDi` for each set of instances are calculated.

The test bed used for this study and the testing conditions are the same described in Section 3.2. Furthermore, the `IPM` code implements the method with the same name, in C and calling the CPLEX solver for solving the shortest path problems. A set of preliminary tests was run, in order to decide how to parametrize the `IPM`. `IPM` works with unitary arc costs and was tested with the additive penalizations $\alpha = 0.25, 0.50, 0.75, 1.00$, applied to the cost of the arcs of the most recently found path. Since considering the penalization $\alpha = 1.00$ produced better results than the remaining penalties for `AvDi` and `MiDi` in all random instances, this was the value fixed in the rest of the experiments. Tables 3.14 and 3.15 present the dissimilarities and the run times of this implementation, which will be used for comparison with the introduced formulations. The dissimilarity results are far better for grids than for random networks. This is due to the fact that all paths in the used grids of a certain size have the same length. In effect, `IPM` naturally avoids the arcs previously used, unless the cost associated to the increase in the number of arcs exceeds the penalty. When applied to the random networks, the length of the paths varies and so does the dissimilarity of the solutions found by the method. Because this method essentially solves K shortest path

problems, it has polynomial complexity of $O(Km + Kn \log n)$ and it run fast for any of the considered instances: in less than 0.70 seconds in random networks and in less than 0.05 seconds in grid networks.

Table 3.14 Average `AvDi` and `MiDi` of `IPM` in random networks

`AvDi`	K								
$R_{n,m}$	3	4	5	6	7	8	9	10	
$R_{100,500}$	0.712	0.823	0.841	0.847	0.839	0.846	0.830	0.830	
$R_{100,1000}$	–	0.857	0.910	0.918	0.941	0.943	0.947	0.941	
$R_{300,1500}$	0.774	0.868	0.892	0.895	0.890	0.886	0.888	0.885	
$R_{300,3000}$	0.917	0.913	0.937	0.919	0.929	0.942	0.942	0.941	
$R_{500,2500}$	0.896	0.911	0.904	0.907	0.911	0.911	0.906	0.904	
$R_{500,5000}$	0.764	0.853	0.867	0.894	0.909	0.917	0.927	0.927	
Average	0.812	0.870	0.891	0.896	0.903	0.907	0.906	0.904	0.886
`MiDi`	K								
$R_{n,m}$	3	4	5	6	7	8	9	10	
$R_{100,500}$	0.429	0.485	0.484	0.428	0.373	0.301	0.212	0.158	
$R_{100,1000}$	–	0.145	0.312	0.316	0.430	0.419	0.423	0.369	
$R_{300,1500}$	0.613	0.642	0.621	0.621	0.491	0.347	0.338	0.251	
$R_{300,3000}$	0.750	0.531	0.550	0.541	0.503	0.521	0.500	0.464	
$R_{500,2500}$	0.736	0.640	0.634	0.576	0.569	0.526	0.498	0.448	
$R_{500,5000}$	0.525	0.588	0.539	0.530	0.564	0.564	0.580	0.580	
Average	0.610	0.505	0.523	0.502	0.488	0.446	0.425	0.378	0.484

Table 3.15 Run times of `IPM` (seconds)

	K								
$R_{n,m}$	3	4	5	6	7	8	9	10	
$R_{100,500}$	0.021	0.028	0.034	0.040	0.046	0.057	0.063	0.070	
$R_{100,1000}$	–	0.053	0.065	0.076	0.089	0.101	0.113	0.126	
$R_{300,1500}$	0.064	0.087	0.110	0.129	0.150	0.176	0.197	0.219	
$R_{300,3000}$	0.116	0.151	0.181	0.210	0.244	0.279	0.311	0.344	
$R_{500,2500}$	0.109	0.139	0.174	0.207	0.245	0.285	0.317	0.351	
$R_{500,5000}$	0.191	0.251	0.322	0.395	0.469	0.545	0.599	0.691	
Average	0.100	0.118	0.147	0.176	0.207	0.240	0.266	0.300	0.194
	K								
$G_{p,q}$	3	4	5	6	7	8	9	10	
$G_{3,12}$	0.000	0.002	0.003	0.004	0.005	0.003	0.005	0.005	
$G_{4,36}$	0.008	0.013	0.016	0.012	0.021	0.026	0.030	0.033	
$G_{6,6}$	0.003	0.003	0.001	0.002	0.004	0.000	0.007	0.008	
$G_{12,12}$	0.005	0.007	0.010	0.013	0.011	0.018	0.017	0.014	
Average	0.004	0.006	0.007	0.007	0.010	0.011	0.014	0.015	0.009

The rest of the section is organized as follows: first, the results for the four initial formulations and the `IPM` are compared; then the effect of adding the constraints (3.10) to the original models is discussed.

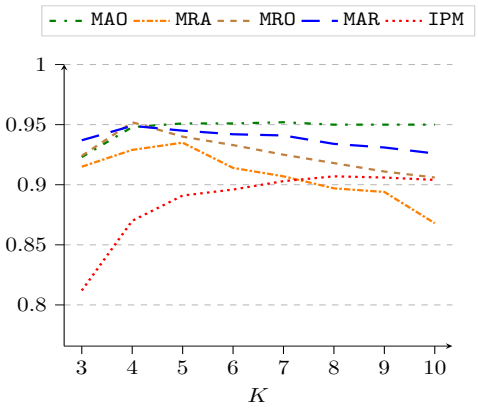
Table 3.16 Average `AvDi` and `MiDi` of `IPM` in grid networks

`AvDi`	K								
$G_{p,q}$	3	4	5	6	7	8	9	10	
$G_{3,12}$	0.948	0.833	0.800	0.769	0.758	0.741	0.732	0.728	
$G_{4,36}$	0.892	0.912	0.982	0.859	0.830	0.817	0.805	0.798	
$G_{6,6}$	0.933	0.933	0.900	0.893	0.866	0.857	0.844	0.840	
$G_{12,12}$	0.969	0.969	0.954	0.951	0.939	0.935	0.929	0.926	
Average	0.958	0.911	0.886	0.868	0.848	0.837	0.827	0.823	0.870

`MiDi`	K								
$G_{p,q}$	3	4	5	6	7	8	9	10	
$G_{3,12}$	0.846	0.153	0.153	0.153	0.153	0.000	0.000	0.000	
$G_{4,36}$	0.947	0.526	0.526	0.500	0.500	0.131	0.078	0.078	
$G_{6,6}$	0.900	0.900	0.600	0.600	0.600	0.600	0.500	0.500	
$G_{12,12}$	0.954	0.954	0.818	0.818	0.681	0.681	0.681	0.681	
Average	0.911	0.633	0.524	0.517	0.483	0.353	0.314	0.314	0.506

3.6.1 Unconstrained formulations

The average and minimum dissimilarities of `MAO`, `MRA`, `MRO`, and `MAR` are compared in the following. The detailed results for each formulation can be found in Appendix A.1.



	K							
	3	4	5	6	7	8	9	10
`MAO`	0.923	0.948	0.951	0.952	0.950	0.950	0.949	0.944
`MRA`	0.915	0.929	0.935	0.914	0.907	0.897	0.894	0.860
`MRO`	0.924	0.952	0.940	0.933	0.925	0.918	0.911	0.906
`MAR`	0.937	0.949	0.945	0.942	0.941	0.934	0.931	0.926
`IPM`	0.812	0.870	0.891	0.896	0.903	0.907	0.906	0.904

Fig. 3.9 Average `AvDi` values of the unconstrained formulations in random networks

We begin by analysing the results for the random networks. Figure 3.9 depicts the variation of the average `AvDi` for each model. In general, the highest values of the average `AvDi` are associated to the code `MAO`, which is followed closely by `MAR` (the difference between the values associated to the two models does not exceed 2%). Nevertheless, `MAR` surmounts `MAO` for $K = 3$ and $K = 4$. On the other hand, `IPM` has the worst performance with this regard, except for $K \geq 8$, where it outperforms `MRA`. In fact, the average dissimilarity obtained by `IPM` tends to improve when K grows, whereas it tends to worsen for all the formulations but `MAO`.

Figure 3.10 summarizes the average `AvDi` results for each random instance. The variation of the dissimilarities follows closely the pattern identified in Sections 3.2.1, 3.3.1 and 3.4.1 for the run times of the formulations: the instances recognised as harder to solve, namely $R_{100,500}$, are associated to the worst `AvDi` values.

Figure 3.11 allows a comparison of the dispersion of the results. The best scores are associated to `MAO`, followed by `MAR` and `MRO`, whereas the `IPM` is the code with more disperse values. In terms of the formulations, `MRA` was worse than the others. The box-plots in Figures A.1 – A.16, in Appendix A.1, allow a thorough analysis of the dispersion of the average `AvDi` and `MiDi` values for each formulation. In general, the dispersion of the dissimilarities increases with K. It can also be concluded that, as expected, the hardest instances have smaller dissimilarities and a bigger dispersion of values.

The values of `MiDi` allow to study the worst case in terms of dissimilarity. Figure 3.12 depicts a summary of the average `MiDi` for random networks. In this case, the best results are found for the `MRO` and the `MAR` models. `IPM` is worse than all formulations with this regard, while all the latter show very similar performances (the inner differences do not exceed 5%). Also worthy of note, is the significant decrease of the `MiDi` values observed with the increase of K.

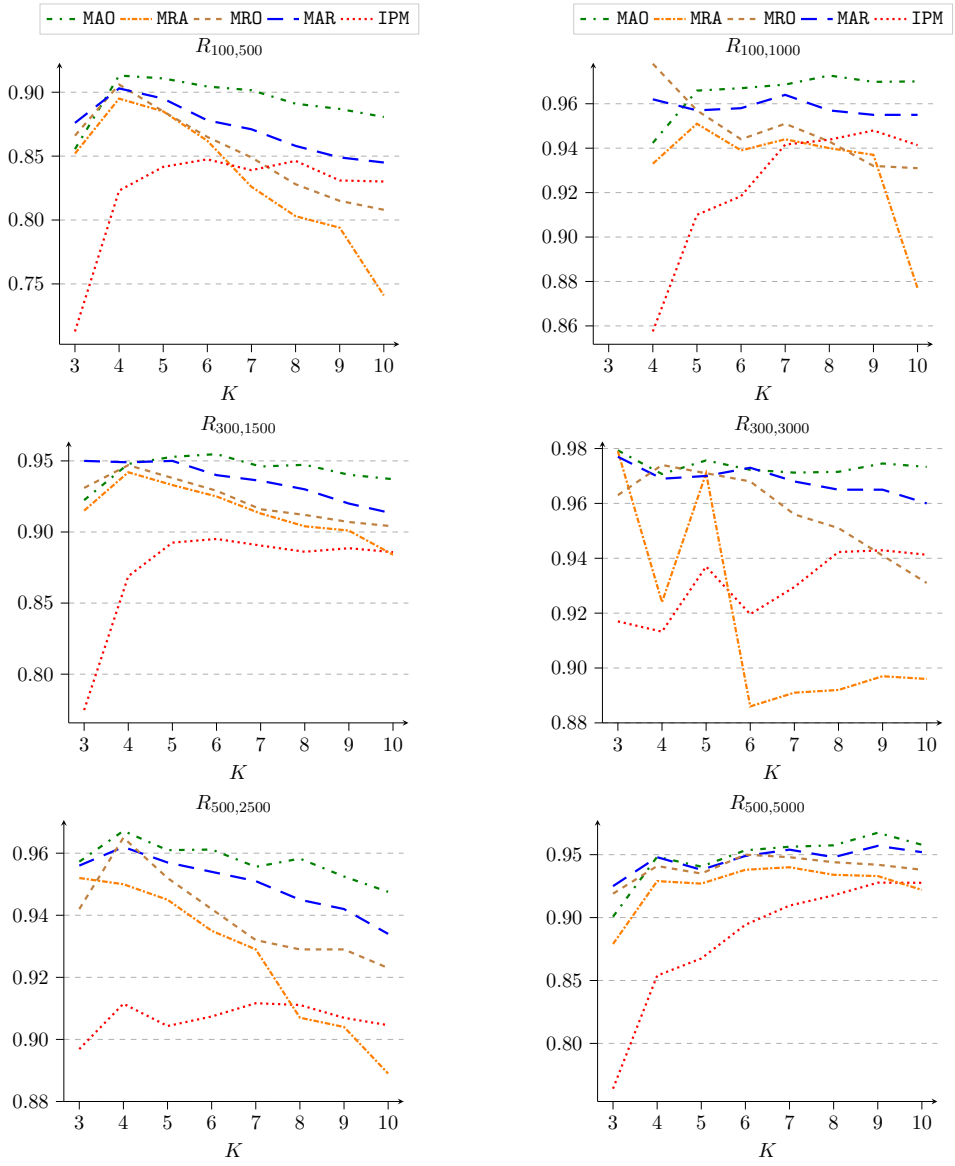
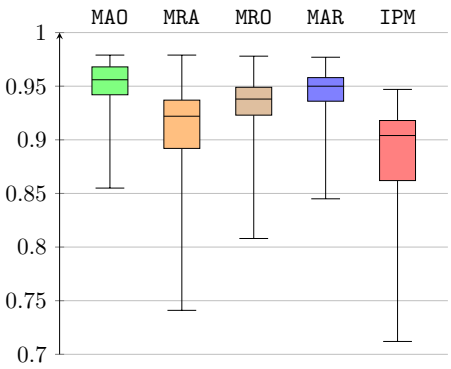
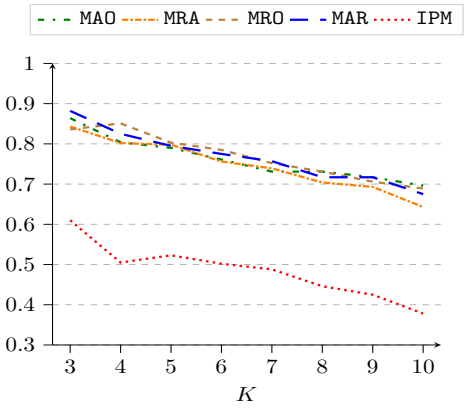


Fig. 3.10 Average dissimilarity of the unconstrained formulations in random networks



	MAO	MRA	MRO	MAR	IPM
Min. value	0.855	0.741	0.808	0.845	0.712
1st quartile	0.942	0.892	0.923	0.936	0.862
Median	0.956	0.922	0.938	0.950	0.904
3rd quartile	0.968	0.937	0.949	0.958	0.918
Max. value	0.979	0.979	0.978	0.977	0.947

Fig. 3.11 AvDi dispersion of the unconstrained formulations in random networks



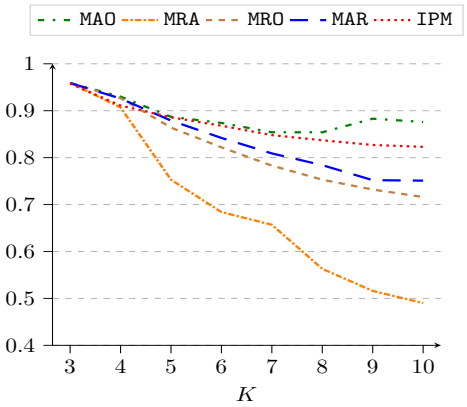
	K							
	3	4	5	6	7	8	9	10
MAO	0.864	0.804	0.790	0.761	0.731	0.717	0.696	0.687
MRA	0.843	0.802	0.790	0.761	0.731	0.717	0.696	0.687
MRO	0.836	0.851	0.803	0.785	0.752	0.731	0.706	0.689
MAR	0.882	0.825	0.795	0.775	0.757	0.717	0.717	0.675
IPM	0.610	0.505	0.523	0.502	0.488	0.446	0.425	0.378

Fig. 3.12 Average MiDi values of the unconstrained formulations in random networks

To summarize, the best and least disperse average AvDi's are associated to the MAO model. However, its high run times undermine its application. In contrast, MAR produced solutions with good average and dispersion dissimilarities in less than 2 seconds, for all instances. Furthermore, the MiDi analysis indicates that MAR is less likely to produce solution with very poor dissimilarities.

Next, the results for the grid networks are analyzed. As already mentioned, for some grid networks it is possible to know the optimal value of D_1. In fact, the length of the paths in a grid network $G_{p,q}$ is constant, namely $q + p - 2$. Consequently, maximizing D_1 is equivalent to minimizing only the numerator of the fractions in its expression, which is precisely the goal of formulation (3.2). Thus, the values of AvDi for all the instances solved to optimality by MAO are optimal dissimilarities. Figure 3.13 illustrates the main differences between the five codes. Like what happened for the random networks, MAO produces the best results. However, this model is now followed by IPM, and the differences towards MAR have become wider and go as high as 10% whereas they do not exceed 2% for IPM. MRA still provides the worst results. Despite the good dissimilarities obtained by MAO, it is worth noting that these are affected by the fact that most of these instances were not solved to optimality within the time limit.

Figure 3.14 allows a more comprehensive comparison of the AvDi values produced by each code. The results highlight the greater difficulty of solving the square instances, as pointed out in sections 3.2.1, 3.3.1 and 3.4.1, and also some inconsistency in the values associated to MRA. Figure 3.15 allows the analysis of the dispersion of these results. MAO is the formulation



	K							
	3	4	5	6	7	8	9	10
MAO	0.959	0.930	0.887	0.874	0.854	0.845	0.833	0.827
MRA	0.959	0.907	0.753	0.684	0.657	0.563	0.516	0.490
MRO	0.959	0.923	0.864	0.822	0.783	0.753	0.732	0.716
MAR	0.959	0.926	0.879	0.842	0.809	0.784	0.752	0.751
IPM	0.958	0.911	0.886	0.868	0.848	0.837	0.827	0.823

Fig. 3.13 Average AvDi values of the unconstrained formulations in grid networks

with the more uniform values, followed by IPM. On the other hand, the models MAR, MRO and MRA present the smallest dispersion (in this order). Overall, the differences between the models lay in the low quartiles, thus MAO and IPM offer less chances of obtaining solutions with low values AvDi for this type of networks. The figures and the tables in Appendix A.1, show that the trends discussed above regarding Figure 3.14 are also present when studying the dissimilarity.

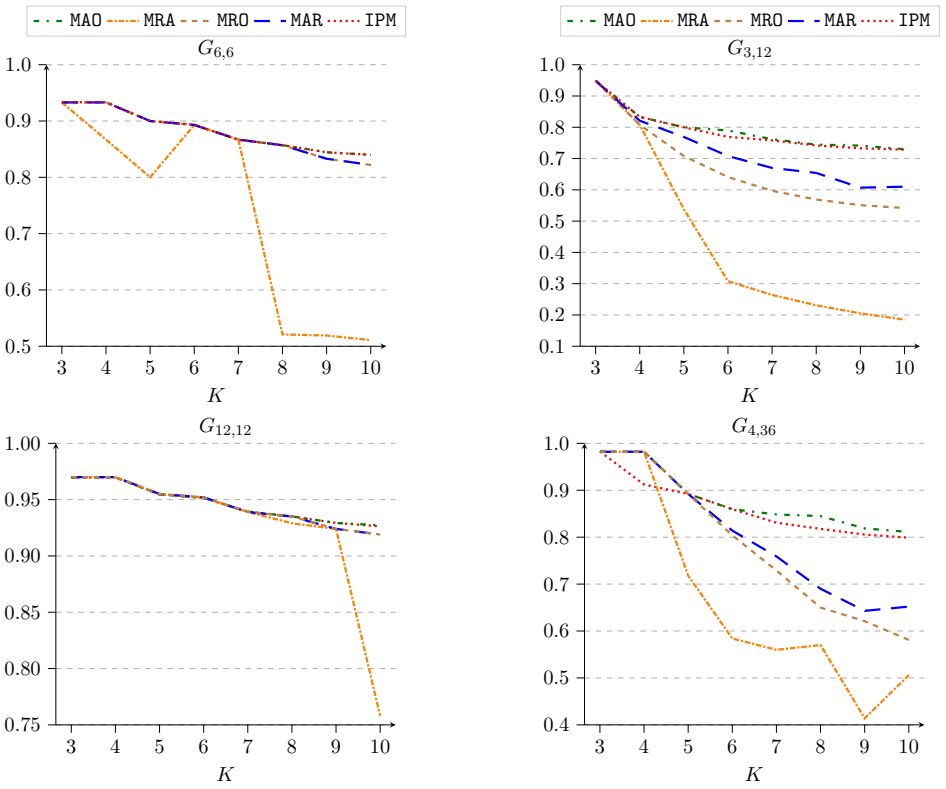
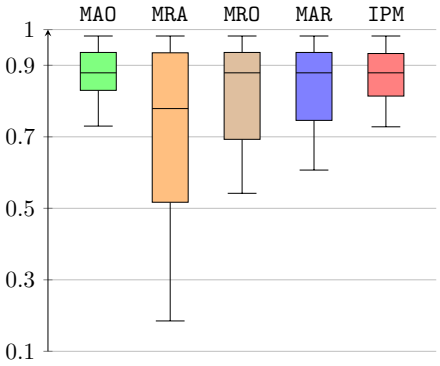


Fig. 3.14 Average dissimilarity of the unconstrained formulations in grid networks

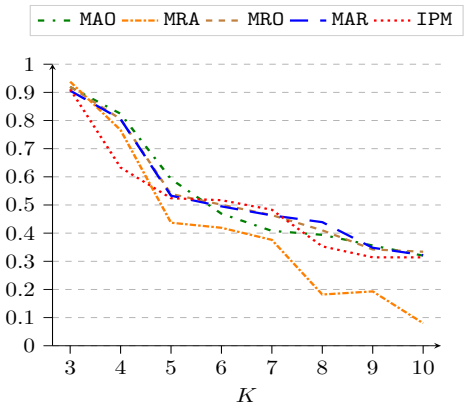
As shown in Figure 3.16, there is no clear dominance regarding the values of MiDi. Nevertheless, the worst results are associated to the IPM for $K = 3, 4$ and to MRA model for $K \geq 5$. It should be mentioned that, in all cases, the results follow heavily with the increase of K. In fact, the new formulations only look for good average dissimilarities between the



	MAO	MRA	MRO	MAR	IPM
Min. value	0.730	0.185	0.542	0.607	0.728
1^{st} quartile	0.830	0.517	0.693	0.746	0.814
Median	0.879	0.779	0.879	0.879	0.879
3^{rd} quartile	0.936	0.935	0.936	0.936	0.933
Max. value	0.982	0.982	0.982	0.982	0.982

Fig. 3.15 `AvDi` dispersion of the unconstrained formulations in grid networks

pairs of paths in the solution, which may hide solutions with pairs of paths with very different dissimilarities.



	K							
	3	4	5	6	7	8	9	10
MAO	0.919	0.824	0.592	0.469	0.408	0.394	0.356	0.320
MRA	0.938	0.766	0.437	0.419	0.376	0.182	0.193	0.080
MRO	0.920	0.805	0.539	0.501	0.464	0.410	0.342	0.334
MAR	0.906	0.805	0.533	0.495	0.463	0.439	0.348	0.322
IPM	0.911	0.633	0.524	0.517	0.483	0.353	0.314	0.314

Fig. 3.16 Average `MiDi` of the unconstrained formulations in grid networks

In conclusion, the dissimilarity results suggest the use of `MAO` model when dealing with grid networks. However, once again, its run times limit its application. On the other hand, `IPM` was also able of finding good solutions and has the strong advantage of running in little time. Therefore, `IPM` seems a sound approach for this specific type of networks. As to the remaining formulations, the `AvDi` values associated to `MAR` are smaller but close to the above mentioned methods (with an average difference of 5%, to both). Moreover, the `MiDi` results favor `MAR`. This information together with the good run times associated to this method, suggest that `MAR` is also worth considering in this context.

3.6.2 Constrained formulations

As shown in Table 3.17, the models obtained by adding the constraints (3.10) to formulations (3.4), (3.5) and (3.6) produce solutions with better dissimilarity results, with few exceptions. Furthermore, the increase in the dissimilarities is bigger when more paths need to be found and bigger in the grids than in the random networks. As expected, the most significant differences occurred to the pair `MRA`–`MRAA`, with dissimilarity improvements of up to 10% for the random networks and up to 244% for the grid networks. As to the pair `MRO`–`MROA`, Table 3.17 indicates

also an improvement of the average dissimilarity of the solutions. However, the differences were smaller both in the cases of the random and of the grid networks (up to 6% in the first case and to 24% in the latter). There were no changes in the dissimilarities obtained on the square grids. Finally, the smallest difference was found for `MAR` and `MARA`. Even so, the improvements on the rectangular grids were quite significant. Again, no differences were registered in the dissimilarity results for square grids. Detailed information about the three models can be found in Appendix A.2.

Table 3.17 Average dissimilarity variation for `MRAA`, `MROA` and `MARA` (%)

	`MRAA`								`MROA`								`MARA`							
	K								K								K							
$R_{n,m}$	3	4	5	6	7	8	9	10	3	4	5	6	7	8	9	10	3	4	5	6	7	8	9	10
$R_{100,500}$	1	1	2	4	6	7	7	12	0	1	2	4	3	6	6	5	−1	1	1	2	2	2	2	2
$R_{100,1000}$	–	2	2	3	2	3	3	10	–	0	1	2	2	3	4	4	–	1	1	0	0	1	1	1
$R_{300,1500}$	1	0	2	2	3	4	4	5	0	0	2	3	3	3	3	3	−2	0	0	1	1	1	2	2
$R_{300,3000}$	0	5	1	10	9	9	9	8	0	−1	1	1	2	3	3	4	−1	1	1	1	1	1	1	1
$R_{500,2500}$	0	1	1	3	2	4	5	6	1	0	0	2	2	3	2	2	1	0	0	1	0	1	1	1
$R_{500,5000}$	2	2	1	1	2	2	3	4	1	1	1	0	1	1	2	2	0	0	1	1	0	1	1	1

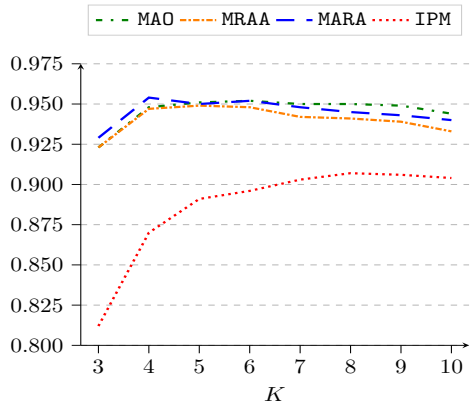
	`MRAA`								`MROA`								`MARA`							
	K								K								K							
$G_{p,q}$	3	4	5	6	7	8	9	10	3	4	5	6	7	8	9	10	3	4	5	6	7	8	9	10
$G_{3,12}$	0	3	37	138	158	202	218	244	0	3	4	16	18	23	22	24	0	2	1	11	6	10	13	15
$G_{4,36}$	0	0	24	38	31	16	73	28	0	0	0	0	0	16	14	24	0	0	0	0	−2	14	19	15
$G_{6,6}$	0	8	13	0	0	64	60	61	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$G_{12,12}$	0	0	0	0	0	1	0	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

In the following, we analyze the trade-off between the improvements in the dissimilarities and the variations in the run time, identified in Section 3.5.1, to assess to what extent the constrained models are worth considering. The new models are compared to `IPM` and `MAO`. The code `MROA` was ruled out of the study, because it follows closely `MARA` but always with some disadvantage.

Figure 3.17 depicts the variation of the average `AvDi` for `MRAA`, `MARA`, `MAO` and `IPM` in the random networks. In spite of the improvement on the average dissimilarities associated to the `MRAA` when compared to `MRA` (see Figures 3.9 and 3.17), the constrained version of `MAR` is still the best, in average. The differences, however, are now very tenuous.

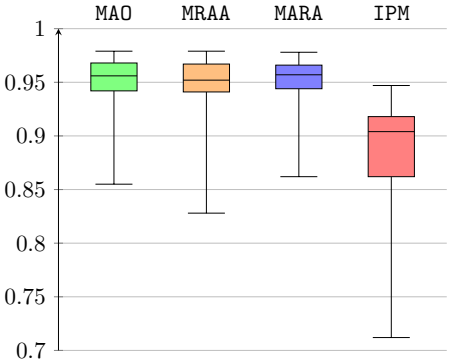
Comparing Figures 3.11 and 3.18 reveals a reduction in the dispersion of the results of both `MARA` and `MRAA`. Nevertheless, `MARA` outperforms `MRAA` and performs even better than `MAO` regarding the worst cases. Again, `IPM` produces the poorest results.

Figure 3.19 reports the average run times of the models. As shown in Section 3.5.1, adding the constraints (3.10) affected `MAR` more than `MRA`. In fact, `MARA` and `MRAA` are now very close – `MRAA` runs faster for the smaller networks and `MARA` for the larger. It is worth mentioning the increase of the run times from 1 to 5 seconds for the biggest instances. As expected, `IPM` is faster than the other codes.



	K							
	3	4	5	6	7	8	9	10
MAO	0.923	0.948	0.951	0.952	0.950	0.950	0.949	0.944
MRAA	0.923	0.947	0.949	0.948	0.942	0.941	0.939	0.933
MARA	0.929	0.954	0.950	0.952	0.948	0.945	0.943	0.940
IPM	0.812	0.870	0.891	0.896	0.903	0.907	0.906	0.904

Fig. 3.17 Average AvDi values of the constrained formulations in random networks

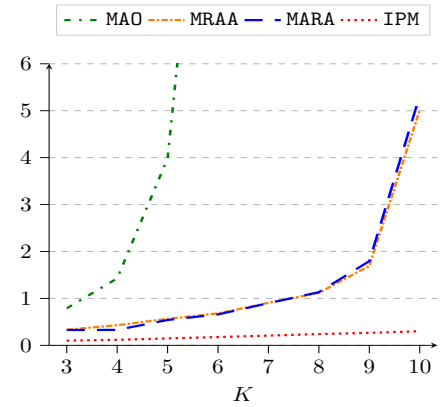


	MAO	MRAA	MARA	IPM
Min. value	0.855	0.828	0.862	0.712
1^{st} quartile	0.942	0.941	0.944	0.862
Median	0.956	0.952	0.957	0.904
3^{rd} quartile	0.968	0.967	0.966	0.918
Max. value	0.979	0.979	0.978	0.947

Fig. 3.18 AvDi dispersion of the constrained formulations in random networks

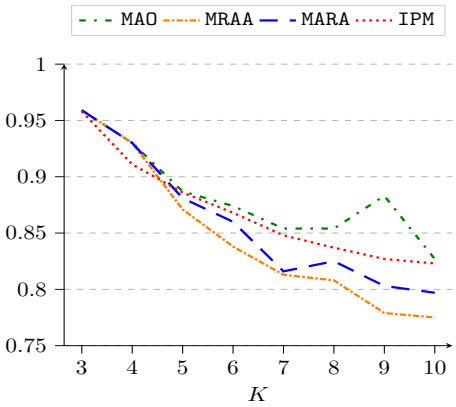
In short, both MRAA and MARA seem tailored for this type of network: the average difference in the dissimilarities towards MAO is of 0.50% for MRAA and of 0.08% for MARA; as to the run times, MRAA is in advantage for $K \leq 7$ and MARA for the remaining values of K. However, taking into account the earlier good dissimilarities and run times of MAR, it is not clear whether its constrained model is the best option.

Figure 3.20 presents the average AvDi for MRAA, MARA, IPM and MAO in grid networks. In spite of the significant improvement for the pair MRA–MRAA (see Figure 3.13), MARA still produced solutions with higher average dissimilarities. In addition, it solved all the instances to optimality, which did not happen for MRAA (it was not possible to find 10 paths in $G_{4,36}$ within 300 seconds). Nevertheless, MAO provides the solutions with the best average dissimilarities, now followed closely by IPM. On the other hand, there are reductions in the dispersion of the results for the pairs MRA–MRAA and MAR–MARA, when comparing Figures 3.15 and 3.21. Furthermore, as the most significant improvements occurred in the lower quartiles, it can be concluded that both MRAA and MARA are less likely to produce solutions with low dissimilarities than MRA and MAR for this type of network. Nevertheless, the best results are, again, associated to MAO and IPM.



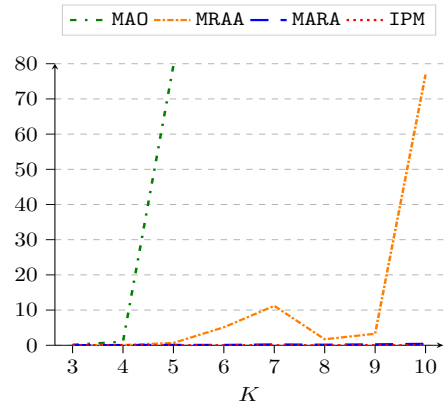
	K							
	3	4	5	6	7	8	9	10
MAO	0.079	1.426	3.985	14.464	40.488	85.412	132.258	179.631
MRAA	0.327	0.329	0.540	0.659	0.898	1.133	1.794	5.320
MARA	0.333	0.429	0.566	0.681	0.906	1.119	1.693	5.014
IPM	0.100	0.118	0.147	0.176	0.207	0.240	0.266	0.300

Fig. 3.19 Run times of the constrained formulations in random networks (seconds)



	K							
	3	4	5	6	7	8	9	10
MAO	0.959	0.930	0.887	0.874	0.854	0.845	0.833	0.827
MRAA	0.959	0.930	0.871	0.838	0.813	0.808	0.779	0.775
MARA	0.959	0.930	0.881	0.860	0.816	0.825	0.803	0.797
IPM	0.958	0.911	0.886	0.868	0.848	0.837	0.827	0.823

Fig. 3.20 Average AvDi values of the constrained formulations in grid networks

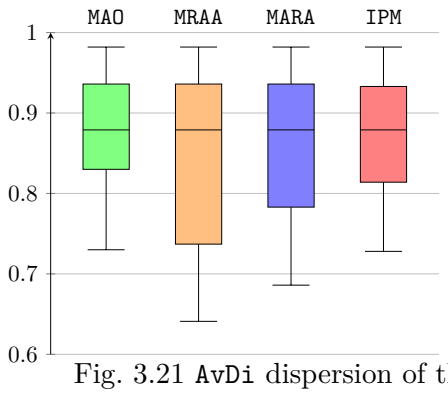


	K							
	3	4	5	6	7	8	9	10
MAO	0.158	1.019	79.534	96.118	300	300	300	300
MRAA	0.055	0.059	0.667	5.111	11.194	1.702	3.270	76.915
MARA	0.043	0.054	0.101	0.093	0.233	0.155	0.260	0.404
IPM	0.004	0.006	0.007	0.007	0.010	0.011	0.014	0.015

Fig. 3.22 Run times for the constrained formulations in grid networks (seconds)

According to Figure 3.22, MRAA is much slower than MARA for this type of instances, which is, in turn, much slower than IPM. Thus, even though MARA has good run times it is clearly overtaken by IPM. As mentioned earlier, in general, the run times associated to MAO are very high. Tables A.11 and A.12 in Appendix A.2 give detailed run times of the MRAA and MARA models.

The previous discussion confirms IPM as the most suitable method for solving instances in grid networks. IPM is the fastest of the codes being compared and produced the solutions



	MAO	MRAA	MARA	IPM
Min. value	0.730	0.641	0.686	0.728
1^{st} quartile	0.830	0.737	0.783	0.814
Median	0.879	0.879	0.879	0.879
3^{rd} quartile	0.936	0.936	0.936	0.933
Max. value	0.982	0.982	0.982	0.982

Fig. 3.21 AvDi dispersion of the constrained formulations in grid networks

with the highest overall average dissimilarity. On the other hand, the results associated to MARA are also to be considered. In average the difference towards MAO is 2%. This, together with the very reasonable run times, makes MARA also interesting for this type of network. In Section 3.6.1 we also pointed out MAR as an interesting option for this type of network. However, comparing to MARA, we realize that MAR offers worse dissimilarities in the same amount of time.

3.7 Concluding remarks

This chapter addressed the search for K dissimilar paths connecting a given pair of nodes in a directed graph. Four integer formulations were introduced. The formulations have different motivations, but their general goal is to minimize the number of arcs that appear in more than one path or the total number of those overlaps, while searching for sets of paths with good dissimilarity. The inclusion of an additional set of constraints to the previous formulations, with the goal of improving the solutions dissimilarity, was also proposed. The performance of the new formulations and of a traditional method in the literature, the iterative penalty method, was tested over random and grid networks, assessing the required run time as well as the average and the minimum dissimilarities of the solutions. The dissimilarity of a given solution is the average of the pairwise dissimilarity between their paths, which is measured based on the ratios between the number of overlaps of the pair of paths and the length of the involved paths. Three classes can be considered with this respect:

- The code IPM, which not surprisingly is the fastest, given that it simply solves K shortest path problems. Besides being the fastest code, IPM provided solutions with good dissimilarity for the grid networks. On the other hand, the solutions produced by this method for the random networks are very poor as to the dissimilarity of the K paths.

- The code MAO, clearly the slowest of the five codes and often unable of finding an optimal solution within the established 300 seconds. In spite of this drawback, in general, MAO produced the solutions with the best dissimilarity, both for the random and the grid networks.

- And the final group, composed of codes `MAR`, `MRAA` and `MARA`, which also provide solutions of good quality with regard to the dissimilarity. The three codes have similar behaviors for the random networks, however, for the grid networks, the latter unequivocally outperforms the others. Unlike `MRAA`, both `MAR` and `MARA` were able to find the optimum within the time bound for all instances. Furthermore, the run times of `MAR` and `MARA` are much smaller than the run times of the remaining codes in this group.

Chapter 4

The bi-objective K shortest - dissimilar paths problem

This chapter is organized into four parts. The next section is dedicated to the presentation of general concepts of bi-objective optimization. In Section 4.2 the K shortest and dissimilar paths problem is defined. The goal of the problem is to find K dissimilar paths with minimum total cost. The methods described in Section 4.2 are adapted to this problem, based on the two best formulations presented in Chapter 3. Finally, the methods developed in Section 4.3 are tested. Computational results of the experiments are presented in the next section. Concluding remarks are provided in Section 4.4.

4.1 Bi-objective optimization

In this section we cover some concepts of bi-objective optimization problems. Let us consider the bi-objective optimization problem (BOP) defined as

$$\begin{array}{ll} \min & f(x) = (f_1(x), f_2(x)) \\ \text{subject to} & x \in X \end{array} \tag{4.1}$$

where $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$ are two objective functions and $X \subseteq \mathbb{R}^n$ is the set of feasible solutions. The image of set X under the objective function f is denoted as $Y = f(X)$. In general the functions f_1, f_2 are conflicting, therefore there is no single solution which simultaneously optimizes both. Thus, instead of searching for an optimal solution, in bi-objective optimization we search for compromise solutions, that is, solutions which cannot be improved in one of the objective functions without worsening the other. In the following we adopt the definition below of Pareto optimality or efficiency.

Definition 4.1.1 ([16]) *A feasible solution of the BOP (4.1), $x^1 \in X$, is said to dominate another feasible solution for the same problem $x^2 \in X$ if*

1. *$f_i(x^1) \leq f_i(x^2)$, for $i = 1, 2$, and*
2. *$f_i(x^1) < f_i(x^2)$, for at least one index $i \in \{1, 2\}$.*

Definition 4.1.2 *A feasible solution of the BOP (4.1) $\hat{x} \in X$ is called efficient or Pareto optimal if there is no other feasible solution $x \in X$ that dominates $\hat{x}$. If the solution $\hat{x}$ is efficient, then its outcome vector $f(\hat{x})$ is called a non-dominated point.*

Some dominated solutions are dominated by others for only one of the objective functions.

Definition 4.1.3 *A feasible solution of the BOP (4.1) $\hat{x} \in X$ is weakly efficient if there is no $x \in X$ such that $f_1(x) < f_1(\hat{x})$ and $f_2(x) < f_2(\hat{x})$. The corresponding outcome vector $f(\hat{x})$ is then said to be weakly non-dominated.*

Definition 4.1.4 *A feasible solution of the BOP (4.1) $\hat{x} \in X$ is called strictly efficient (strictly Pareto optimal) if there is no $x \in X$, $x \neq \hat{x}$ such that $f_1(x) \leq f_1(\hat{x})$ and $f_2(x) \leq f_2(\hat{x})$. The strictly efficient sets are denoted X_{sE}.*

Whenever it is established that the two objective functions have different priorities, we talk about lexicographic optimization. This case is based on an order relation. Given $y^1, y^2 \in \mathbb{R}^2$ we say that y^1 is lexicographically smaller than y^2, and write $y^1 <_{lex} y^2$, if $y^1_q < y^2_q$ where $q = \min\{k : y^1_k \neq y^2_k\}$, and say that y^1 is lexicographically smaller than or equal to y^2, and write $y^1 \leq_{lex} y^2$, if $y^1 = y^2$ or $y^1 <_{lex} y^2$.

Definition 4.1.5 *A feasible solution of the BOP (4.1), $\hat{x} \in X$, is lexicographically optimal with respect to (f_1, f_2), or a lexicographic solution, if $f(\hat{x}) \leq_{lex} f(x)$, for all $x \in X$.*

An efficient solution is not necessarily a lexicographically optimal solution. However, the opposite holds, that is, any lexicographically optimal solution is also an efficient solution.

Lemma 4.1.1 ([16]) *Let $\hat{x} \in X$ be such that $f(\hat{x}) \leq_{lex} f(x)$ for any $x \in X$. Then $\hat{x}$ is an efficient solution to the BOP (4.1).*

The set of all efficient solutions will be denoted by X_E and called the efficient set. The set of all non-dominated points $\hat{y} = f(\hat{x}) \in Y$, where $\hat{x} \in X_E$, is denoted by Y_N and called the non-dominated set.

The ideal and the nadir points of the BOP (4.1) are defined as lower and upper bounds on its non-dominated points, respectively. These points give an indication of the range of the values that the non-dominated points can attain [43, 44, 59].

Definition 4.1.6

1. *The point $y^I = (y^I_1, y^I_2)$, where $y^I_k = \min_{x \in X}\{f_k(x)\}$, for $k = 1, 2$, is called the ideal point of the BOP (4.1).*
2. *The point $y^N = (y^N_1, y^N_2)$ where $y^N_k = \max_{x \in X_E}\{f_k(x)\}$, $k = 1, 2$, is called the nadir point of the BOP (4.1).*

In bi-objective optimization, the worst value of the second objective function is attained among the solutions that minimize the first objective function and vice versa, which makes it easy to compute the nadir point. In other words, both the ideal and the nadir point can be found by computing the lexicographic optimal solution with respect to (f_1, f_2) and the lexicographic optimal solution with respect to (f_2, f_1).

4.1.1 The ϵ-constraint method

Traditional approaches for solving bi-objective optimization problems are based on scalarization. This involves formulating a single objective optimization problem that is related to the BOP (4.1) by means of a real-valued scalarizing function which typically depends on the objective functions of the BOP, auxiliary scalar or vector variables, or scalar or vector parameters. Sometimes the feasible set of the BOP is additionally restricted by new constraint functions related to the objective functions of the BOP or the new variables introduced.

One of the simplest methods to solve bi-objective problems is the weighted-sum method [12, 37], which solves the weighted-sum problem

$$\min_{x \in X} \sum_{k=1}^{2} \lambda_k f_k(x).$$

where $\lambda_1, \lambda_2 \geq 0$ are parameters such that $\lambda_1 + \lambda_2 = 1$. This method solves a sequence of weighted-sum problems of this type, where the parameters λ_1, λ_2 vary in order to obtain different solutions. All the weighted-sum problems are of the same type as the original, given that the feasible region does not change. However, the method requires the normalization of the two objective functions if they represent different quantities. Furthermore, it is unable of finding non-dominated solutions that are not extreme (usually known as supported solutions) [16].

Together with the weighted-sum approach, the ϵ-constraint method is probably the best known technique to solve bi-objective optimization problems. In this case there is no aggregation of objectives. Instead, only one of the original objective functions is minimized, while the other is transformed to a constraint. The scalar ϵ represents the upper bound on the objective function involved in the new constraint, and by varying this scalar in an appropriate way, the complete set of non-dominated points can be generated. The method was first introduced by Haimes, Lasdon and Wismer [27], and extensive discussions about the topic can be found in Chankong and Haimes [9] or Mavrotas [42]. In the following some more details are given on this method. For easiness of presentation, without loss of generality, we consider that f_1 is the objective function to minimize and f_2 is the objective function included in the constraints.

As explained above, in the ϵ-constraint method, the BOP (4.1) is replaced by the ϵ-constraint problem

$$\begin{array}{ll} \text{minimize} & f_1(x) \\ \text{subject to} & x \in X \\ & f_2(x) \leq \epsilon \end{array} \tag{4.2}$$

where $\epsilon \in \mathbb{R}$. Furthermore, updating ϵ as $\hat{f}_2 - \Delta$, where $\hat{f}_2$ is the solution value in the second objective of a feasible solution and $\Delta > 0$ is a small number, guarantees an improvement of the second objective. The solution of this problem may be an efficient solution of the BOP, although in a general case only its weakly efficiency can be ensured.

Proposition 4.1.1 ([16]) *Let $\hat{x}$ be an optimal solution of (4.2). Then $\hat{x}$ is weakly efficient.*

Proof. Assume $\hat{x}$ is not weakly efficient. Then there is an $x \in X$ such that $f_1(x) < f_1(\hat{x})$ and $f_2(x) < f_2(\hat{x})$. Since $f_2(x) < f_2(\hat{x}) \leq \epsilon$, the solution x is feasible for (4.2). This is a contradiction to $\hat{x}$ being an optimal solution of (4.2). □

In order to strengthen Proposition 4.1.1 to obtain efficiency we require the optimal solution of (4.2) to be unique.

Proposition 4.1.2 ([16]) *Let $\hat{x}$ be a unique optimal solution of (4.2). Then $\hat{x} \in X_{sE}$ (and therefore $\hat{x} \in X_E$).*

Proof. Assume there is some $x \in X$ with $f_2(x) < f_2(\hat{x}) \leq \epsilon$. If in addition $f_1(x) \leq f_1(\hat{x})$ we must have $f_1(x) = f_1(\hat{x})$ because $\hat{x}$ is an optimal solution of (4.2). So x is an optimal solution of (4.2). Thus, uniqueness of the optimal solution implies $x = \hat{x}$ and $\hat{x} \in X_E$. □

In general, the efficiency of $\hat{x}$ is related to $\hat{x}$ being an optimal solution of the constrained problem with respect to each of the objective functions with the same ϵ. It can be shown that with appropriate choices of ϵ all non-dominated solutions can be found.

The choice of which function to optimize and which to include in the constraints, as well as the strategy adopted for updating the bound ϵ, can vary and may depend on the particular form of the problem. An outline of a generic ϵ-constrained method is given in Algorithm 2.

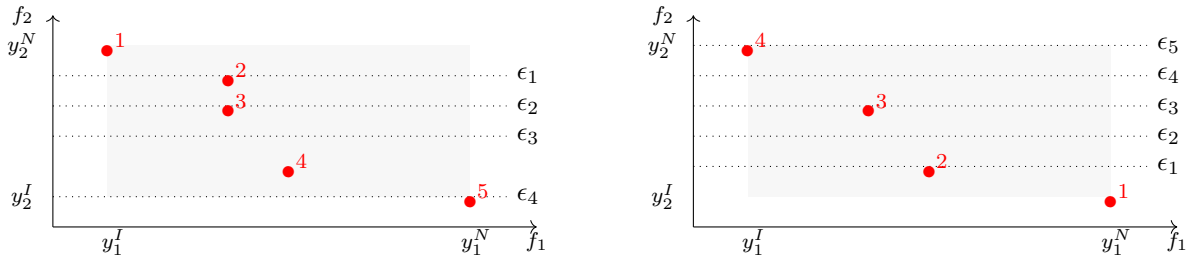
Algorithm 2: The ϵ-constraint method – Decreasing ϵ version

```
1  (y_1^I, y_2^I) ← ideal point for (f_1, f_2) in X
2  (y_1^N, y_2^N) ← nadir point for (f_1, f_2) in X
3  Y_E ← {(y_1^I, y_2^N)}
4  x̄ ← (y_1^I, y_2^N)
5  ε ← y_2^N − Δ
6  while ε ≥ y_2^I do
7      x* ← optimal solution of problem (4.2)
8      if f_1(x*) > f_1(x̄_1) then Y_E ← Y_E ∪ {(f_1(x̄), f_2(x̄))}
9      x̄ ← x*
10     ε ← f_2(x*) − Δ
11 Y_E ← Y_E ∪ {(f_1(x̄), f_2(x̄))}
```

The set Y_E stores the non-dominated points of the problem as they are computed. A new solution is found for each value of ϵ, and the parameter ϵ is updated according with its objective value. The variable $\bar{x}$ is an auxiliary variable that stores the latest solution found until it is concluded whether it is efficient or it is dominated. The line 8 in the pseudo-code corresponds to a dominance test for solution $\bar{x}$. As a result, in case $\bar{x}$ is an efficient solution, its image is included in the set Y_E.

Algorithm 2 is illustrated in Figure 4.1a. Point 1, image of x_1^*, is the first non-dominated point to be computed. Then ϵ_1 is set to $f_2(x_1^*) - \Delta$, where Δ is a suitably chosen and problem dependent value. Then the solution corresponding to point 2, image of x_2^*, is obtained. The procedure is then repeated for $\epsilon_2 = f_2(x_2^*) - \Delta$ and the new solution corresponds to point 3, image of x_3^*. Comparing points 2 and 3, it can be concluded that point 2 is weakly dominated, because it is dominated by point 3. Therefore, point 2 is not inserted in set Y_E. Instead, the next step consists of computing the solution corresponding to point 4, image of x_4^*, and when

this is compared to x_3^*, the latter solution is inserted in the set Y_E. These instructions are repeated until ϵ reaches the ideal value for f_2, thus computing the solutions represented by point 5, which is also a non-dominated point.



(a) Decreasing ϵ version (Algorithm 2)

(b) Increasing ϵ version (Algorithm 3)

Fig. 4.1 The ϵ-constraint method

The ϵ-constraint algorithm has two main drawbacks. One is that weakly dominated solutions can be found along the process, as shown above. In this case the method presumably may solve more ϵ-constrained programs than what is actually required, that is, more problems than the number of non-dominated points. The other is that the new constraint may ruin the structure of the underlying problem and, thus, make it harder to solve. The first of these drawbacks can be overcome by:

- Comparing the solutions as they are computed, and filtering those that are dominated, as already indicated in Algorithm 2 .
- Or transforming every sub-problem into a lexicographic problem with respect to (f_1, f_2), that is, minimizing f_1 and considering the best value for f_2 whenever there is a tie in function f_1. This way the solution obtained is guaranteed to be efficient, and the condition on line 8 of Algorithm 2 can be skipped.
- Or even deleting the dominated solutions from the set of all computed solutions after they have been determined.

Additionally, the objective function can be perturbed in order to avoid the computation of these weakly efficient solutions [42].

Like in Algorithm 2, in traditional implementations of the ϵ-constraint method for minimization problems, the values of ϵ decrease as solutions are computed. The region defined by the ideal and the nadir points, where non-dominated points lie. However, may as well be swept by increasing the parameter ϵ rather than by decreasing it. The solutions of sub-problems (4.2) are still weakly solutions regardless of the policy for updating ϵ. In practical terms the difference is that by increasing ϵ the feasible regions in the sequence of sub-problems become larger and each sub-problem is a relaxation of the previous, as stated next.

Proposition 4.1.3 *Let x^* and x' be optimal solutions for the constrained problems*

$$\begin{aligned} &\textit{minimize} && f_1(x) \\ &\textit{subject to} && x \in X \\ & && f_2(x) \leq \epsilon^* \end{aligned} \tag{4.3}$$

and

$$\begin{aligned} &\textit{minimize} && f_1(x) \\ &\textit{subject to} && x \in X \\ & && f_2(x) \leq \epsilon' \end{aligned} \tag{4.4}$$

respectively, where $\epsilon^* \leq \epsilon'$. Then the following hold:

1. *x^* is a feasible solution of problem (4.4);*
2. *$f_1(x^*) \geq f_1(x')$.*

The first point in Proposition 4.1.3 implies that x^* can be used as an initial feasible solution and the starting point for solving problem (4.4), which is expected to speed up the resolution of each sub-problem. On the other hand, a consequence of the second point in the same result is that the same solution may be obtained more than once, thus requiring more sub-problems to be solved. Additionally, ϵ needs to be updated differently in order to progress the search in the image space when a solution is repeated. In this case ϵ should be updated as $\max\{\epsilon, f_2(x^*)\} + \Delta$, if x^* denotes the optimal solution for the previous sub-problem.

The outline of the ϵ-constraint method when the parameter ϵ is increased is provided in Algorithm 3.

Algorithm 3: The ϵ-constraint method – Increasing ϵ version

```
(y_1^I, y_2^I) ← ideal point for (f_1, f_2) in X
(y_1^N, y_2^N) ← nadir point for (f_1, f_2) in X
Y_E ← {(y_1^N, y_2^I)}
x̄ ← (y_1^N, y_2^I)
ε ← y_2^I + Δ
while ε ≤ y_2^N do
    x* ← optimal solution of problem (4.2)
    if f_1(x*) < f_1(x̄_1) then Y_E ← Y_E ∪ {(f_1(x̄), f_2(x̄))}
    x̄ ← x*
    ε ← max{ε, f_2(x*)} + Δ
Y_E ← Y_E ∪ {(f_1(x̄), f_2(x̄))}
```

Like what happens with Algorithm 2, weakly efficient solutions can still be obtained with Algorithm 3. However these can be easily discarded after being compared with the latest computed solution.

The application of Algorithm 3 is illustrated in Figure 4.1b. In this case point 1 is the first to be obtained, and then ϵ is set to ϵ_1, thus generating point 2. At this point ϵ is updated to ϵ_2, which produces a problem with an optimal solution that corresponds again to point 2.

Nevertheless, that solution is discarded and ϵ is set to ϵ_3 which allows point 3 to be obtained. The procedure continues until ϵ reaches the second coordinate of the nadir point, that is, y_2^N.

4.2 The bi-objective K dissimilar paths problem

Let us now consider that each arc in the network is associated with a cost value $c_{ij} \in \mathbb{R}^+$, for any $(i,j) \in A$. Additionally, given K vectors $x^k \in \{0,1\}^m$, $k = 1, \ldots, K$, their total cost is defined as the sum of all their arc costs, that is

$$\sum_{(i,j)\in A} c_{ij} \sum_{k=1}^{K} x_{ij}^k.$$

If each vector x^k is the characteristic vector of a path from node s to node t, its total cost defines the cost of the K paths.

As mentioned before, besides finding dissimilar paths that can serve as alternative routes, it is of interest to find paths which are relatively short in terms of the travel distance or the cost. This can be the solution of a BOP with the goals of minimizing the total cost of their paths and minimizing the number of arcs that they share. We will refer to the two objective functions as a cost objective and an overlaps objective, respectively. The resulting bi-objective problem will be called the K shortest and dissimilar paths problem.

Following the two best approaches for the K dissimilar paths problem presented in Chapter 3, we formulate two versions of the K shortest and dissimilar paths problem. One of the versions focuses the minimization of the total cost as well as of the number of repeated arcs in the set of K paths; the other one focuses the minimization of the total cost as well as of the total number of arc repetitions. Each version is, in turn, associated to two models, the difference between them being the inclusion of constraint (3.10). Next, we introduce the four models.

4.2.1 Minimization of the number of repeated arcs

The bi-objective problem that results from extending formulation MRA, (3.4), is formulated as:

$$\min \quad v_3(x,y) = \sum_{(i,j)\in A} c_{ij} \sum_{k=1}^{K} x_{ij}^k \tag{4.5a}$$

$$\min \quad v_1(x,y) = \sum_{(i,j)\in A} y_{ij} \tag{4.5b}$$

$$\text{subject to} \quad \sum_{j\in N:(i,j)\in A} x_{ij}^k - \sum_{j\in N:(j,i)\in A} x_{ji}^k = \begin{cases} 1 & i = s \\ 0 & i \neq s,t \\ -1 & i = t \end{cases}, \quad k = 1,\ldots,K \tag{4.5c}$$

$$y_{ij} \leq \sum_{k=1}^{K} x_{ij}^k, \quad (i,j) \in A \tag{4.5d}$$

$$(K-1)y_{ij} \geq \sum_{k=1}^{K} x_{ij}^k - 1, \quad (i,j) \in A \tag{4.5e}$$

$$x_{ij}^k \in \{0,1\},\ y_{ij} \in \{0,1\}, \quad (i,j) \in A, \quad k = 1,\ldots,K \tag{4.5f}$$

where the decision variables have the same meaning as in Chapter 3. Hereinafter this model will be designated as BORA. We now analyze some properties of this formulation and of its solutions.

Proposition 4.2.1 *Any efficient solution of problem (4.5) is loopless.*

Proof. Let x^* be an efficient solution of (4.5). By contradiction, assume that the k'-th path defined by x^* contains one loop, which is defined by the set of arcs $L \subseteq A$. (The reasoning can be replicated if more than one loop exists or several paths contain several loops.)

Let $x^*_{A-L} \in \{0,1\}^{Km}$ denote a vector that results from removing the loop L from x^*, that is, the components of x^*_{A-L} are equal to 0 when $(i,j) \in L$ and $k = k'$, and equal to x^* for all other components. Moreover, let $y^*_{A-L} \in \{0,1\}^m$ denote a vector equal to y^* for the positions $(i,j) \in A - L$ and satisfying the constraints (4.5d) and (4.5e) for the remaining positions. Then,

$$v_3(x^*,y^*) = v_3(x^*_{A-L},y^*_{A-L}) + \sum_{(i,j)\in A-L} c_{ij} x_{ij}^{*k'} > v_3(x^*_{A-L},y^*_{A-L}),$$

because all the arc costs are positive. Similarly

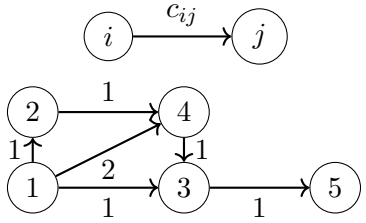
$$v_1(x^*,y^*) \geq v_1(x^*_{A-L},y^*_{A-L}),$$

because $A - L \subseteq A$ and therefore the set of repeated arcs in x^*_{A-L} is contained in the set of repeated arcs in x^*.

Additionally, a set of K paths is still obtained if the loop formed by L is deleted, therefore, x^*_{A-L} defines a feasible solution of problem (4.5) such that

$$v_3(x^*, y^*) > v_3(x^*_{A-L}, y^*_{A-L}) \quad \text{and} \quad v_1(x^*, y^*) \geq v_1(x^*_{A-L}, y^*_{A-L}).$$

We thus conclude that x^* is dominated by x^*_{A-L}, which contradicts the assumption. □



(a) Network

Paths from 1 to 5
$p_1 = \langle 1, 3, 5\rangle$
$p_2 = \langle 1, 4, 3, 5\rangle$
$p_3 = \langle 1, 2, 4, 3, 5\rangle$

P_2	$v_3(P_2)$	$v_1(P_2) = v_2(P_2)$
$\{p_1, p_1\}$	4	2
$\{p_1, p_2\}$	6	1
$\{p_1, p_3\}$	6	1

(b) Efficient solutions

Fig. 4.2 Finding $K = 2$ shortest and dissimilar paths from node 1 to node 5

In order to illustrate the problem in hand we consider the problem of finding $K = 2$ paths from node 1 to node 5 in the network in Figure 4.2a. Figure 4.2b lists all those paths and the possible solutions, P_2, together with the corresponding objective values. There are three efficient solutions for formulation `BORA` in the rightmost table: $\{p_1, p_1\}$, $\{p_1, p_2\}$ and $\{p_1, p_3\}$, the latest two of them have total cost 6 and in both cases the arc $(3, 5)$ is shared by the two paths. The ϵ-constraint method finds a set of efficient solutions, each one corresponding to one non-dominated point. As a consequence, only one of those solutions, either $\{p_1, p_2\}$ or $\{p_1, p_3\}$, is computed and stored.

Nevertheless, the solution $\{p_1, p_3\}$ is longer than the solution $\{p_1, p_2\}$, and therefore the dissimilarity of the solution is 0.625 in the first case, and solution 0.583 in the second. This means that listing the solutions according to the minimization of different objective functions may change the order by which the solutions are computed, originating different sets of non-dominated solutions, and thus translate into solutions with different dissimilarities.

Two questions need to be discussed before applying the ϵ-constraint method to this problem: the factor Δ which is used to update ϵ and which objective function to optimize versus which one to constrain.

Regarding the first point, the function v_1 is intrinsically integer, therefore $\Delta = 1$ is a natural choice if v_1 is constrained. The cost function v_3 is also integer when the arc costs are integers too, and then $\Delta = 1$ is a suitable choice if the function v_3 is constrained, but in a more general case a small Δ can be fixed, however it is difficult to find the right value that does not prevent any non-dominated point from being computed.

As to the second point, the two objective functions are bounded by

$$1 \leq v_3(x, y) \leq K(n-1) \max_{(i,j)\in A} \{c_{ij}\} \quad \text{and} \quad 0 \leq v_1(x, y) \leq m,$$

for any feasible solution (x, y) of (4.5). Thus, in general, the range of v_3 is larger than that of v_1, which suggests that restricting function v_1 may yield fewer sub-problems to solve than

restricting function v_3. Moreover, the sub-problems to solve in each case are:

$$\begin{aligned} \min \quad & v_3(x,y) = \sum_{(i,j)\in A} c_{ij} \sum_{k=1}^{K} x_{ij}^k \\ \text{subject to} \quad & (4.5c) - (4.5e) \\ & \sum_{(i,j)\in A} y_{ij} \leq \epsilon \\ & x_{ij}^k \in \{0,1\},\ y_{ij} \in \{0,1\}, \quad (i,j) \in A, \quad k = 1,\ldots,K \end{aligned} \tag{4.6}$$

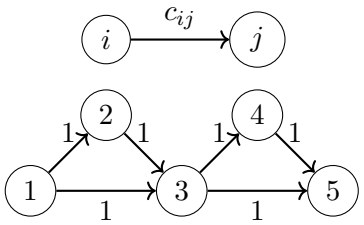
and

$$\begin{aligned} \min \quad & v_1(x,y) = \sum_{(i,j)\in A} y_{ij} \\ \text{subject to} \quad & (4.5c) - (4.5e) \\ & \sum_{(i,j)\in A} c_{ij} \sum_{k=1}^{K} x_{ij}^k \leq \epsilon \\ & x_{ij}^k \in \{0,1\},\ y_{ij} \in \{0,1\}, \quad (i,j) \in A, \quad k = 1,\ldots,K \end{aligned} \tag{4.7}$$

The first of these problems is close to an extension of K shortest path problems. Thus any optimal solution of it is loopless.

Contrarily, formulation (4.7) is closer to formulation (3.4) and may admit solutions with loops, which can however be discarded without compromising the objective value.

The loops in a given optimal solution can be discarded by applying algorithm 1 with time of $O(Km)$ presented in Chapter 3. Nevertheless, experiments revealed that problem (4.7) is harder than problem (4.6). For this reason, in the following we consider that function v_3 is minimized, while function v_1 is restricted, and $\Delta = 1$ will be used.



(a) Network

Paths from 1 to 5
$p_1 = \langle 1,3,5\rangle$
$p_2 = \langle 1,2,3,5\rangle$
$p_3 = \langle 1,3,4,5\rangle$
$p_4 = \langle 1,2,3,4,5\rangle$

(b) Paths from 1 to 5

P_3	$v_3(P_3)$	$v_1(P_3)$	$v_2(P_3)$
$\{p_1,p_1,p_1\}$	6	2	4
$\{p_1,p_1,p_2\}$	7	2	3
$\{p_1,p_1,p_3\}$	7	2	3
$\{p_1,p_1,p_4\}$	8	2	2
$\{p_1,p_2,p_3\}$	8	2	2

(c) Efficient solutions for `BORA` or `BOAR`

P_3	$v_3(P_3)$	$v_1(P_3)$	$v_2(P_3)$
$\{p_1,p_1,p_4\}$	8	2	2
$\{p_1,p_2,p_3\}$	8	2	2

(d) Efficient solutions for `BORAA` and `BOARA`

Fig. 4.3 Finding $K = 3$ shortest and dissimilar paths from node 1 to node 5

To conclude this section, we consider the bi-objective version of formulation `MRAA`, obtained by adding the set of constraints (3.10) to formulation `MRA` to model the dissimilarity constraints. Figure 4.3 illustrates the new problem, when seeking for $K = 3$ paths from node 1 to node 5.

There is only one efficient solution for `BORA`, the set $\{p_1, p_1, p_1\}$ listed in Figure 4.3c. When imposing that the arcs cannot appear more than twice ($R = 2$) in the paths from node 1 to node 5, that solution becomes unfeasible. Therefore, there are two efficient solutions in this case, listed in Figure 4.3d.

The sub-problems to solve in case of the bi-objective extension of formulation `MRAA` are as follows:

$$\begin{aligned}
\min \quad & v_3(x,y) = \sum_{(i,j)\in A} c_{ij} \sum_{k=1}^{K} x_{ij}^k \\
\text{subject to} \quad & (4.5c) - (4.5e) \\
& \sum_{k=1}^{K} x_{ij}^k \leq \ R, \quad (i,j) \in A \\
& \sum_{(i,j)\in A} y_{ij} \leq \epsilon \\
& x_{ij}^k \in \{0,1\},\ y_{ij} \in \{0,1\}, \quad (i,j) \in A, \quad k = 1,\ldots,K
\end{aligned} \tag{4.8}$$

and

$$\begin{aligned}
\min \quad & v_1(x,y) = \sum_{(i,j)\in A} y_{ij} \\
\text{subject to} \quad & (4.5c) - (4.5e) \\
& \sum_{k=1}^{K} x_{ij}^k \leq \ R, \quad (i,j) \in A \\
& \sum_{(i,j)\in A} c_{ij} \sum_{k=1}^{K} x_{ij}^k \leq \epsilon \\
& x_{ij}^k \in \{0,1\},\ y_{ij} \in \{0,1\}, \quad (i,j) \in A, \quad k = 1,\ldots,K
\end{aligned} \tag{4.9}$$

Hereinafter the corresponding formulation will be designated as `BORAA`.

The two questions discussed earlier regarding (4.6) and (4.7) also arise when analyzing (4.8) and (4.9). However, since adding the constraints (3.10) does not alter the premises of the previous discussion, the same line of reasoning applies. Thus, also in this case we consider that function v_3 is minimized, while function v_1 is restricted, and $\Delta = 1$ will be used.

4.2.2 Minimization of the number of arc repetitions

Considering again the decision variables defined as in Chapter 3, the bi-objective problem from extending formulation MAR, (3.6) can be written as:

$$\min \quad v_3(x,w,u) = \sum_{(i,j)\in A} c_{ij} \sum_{k=1}^{K} x_{ij}^k \tag{4.10a}$$

$$\min \quad v_2(x,w,u) = \sum_{(i,j)\in A} u_{ij} \tag{4.10b}$$

$$\text{subject to} \quad \sum_{j\in N:(i,j)\in A} x_{ij}^k - \sum_{j\in N:(j,i)\in A} x_{ji}^k = \begin{cases} 1 & i = s \\ 0 & i \neq s,t \\ -1 & i = t \end{cases}, \quad k = 1,\ldots,K \tag{4.10c}$$

$$\sum_{k=1}^{K} x_{ij}^k \leq K\, w_{ij}, \quad (i,j) \in A \tag{4.10d}$$

$$u_{ij} = \sum_{k=1}^{K} x_{ij}^k - w_{ij}, \quad (i,j) \in A \tag{4.10e}$$

$$x_{ij}^k \in \{0,1\},\ w_{ij} \in \{0,1\},\ u_{ij} \geq 0, \quad (i,j) \in A, \quad k = 1,\ldots,K \tag{4.10f}$$

which will be designated as BOAR in the following.

The same reasoning applied in Proposition 4.2.1 holds to prove that any efficient solution of problem (4.10) is loopless.

Moreover, the function v_3 is common to the previous problem and, like before, function v_2 assumes integer values, therefore the value of Δ can be set to 1 if v_2 is the function chosen to include in the constraints. Additionally,

$$0 \leq v_2(x,w,u) \leq Km,$$

for any feasible solution (x,w,u). Once again, in general, the range of v_3 is larger than that of v_2. Furthermore, the minimization of function v_3 is also easier than the minimization of v_2 and it produces loopless solutions. Therefore, v_3 will be the function to minimize and v_2 the function to restrict, and $\Delta = 1$ will be chosen. In this case the sub-problems to be solved in the ϵ-constraint method are:

$$\begin{aligned} \min \quad & v_3(x,w,u) = \sum_{(i,j)\in A} c_{ij} \sum_{k=1}^{K} x_{ij}^k \\ \text{subject to} \quad & (4.10c) - (4.10e) \\ & \sum_{(i,j)\in A} u_{ij} \leq \epsilon \\ & x_{ij}^k \in \{0,1\},\ w_{ij} \in \{0,1\},\ u_{ij} \geq 0, \quad (i,j) \in A, \quad k = 1,\ldots,K \end{aligned} \tag{4.11}$$

where the parameter $\epsilon > 0$ is updated as new solutions are found.

Again, the constrained version of formulation (4.10) is obtained by adding the set of constraints (3.10) to the model, thus:

$$\begin{aligned}
\min \quad & v_3(x,w,u) = \sum_{(i,j)\in A} c_{ij} \sum_{k=1}^{K} x_{ij}^k \\
\text{subject to} \quad & (4.10c) - (4.10e) \\
& \sum_{k=1}^{K} x_{ij}^k \leq \; R, \quad (i,j) \in A \\
& \sum_{(i,j)\in A} u_{ij} \leq \epsilon \\
& x_{ij}^k \in \{0,1\},\; w_{ij} \in \{0,1\},\; u_{ij} \geq 0, \quad (i,j) \in A, \quad k = 1,\ldots,K
\end{aligned} \tag{4.12}$$

hereinafter designated as `BOARA`. Like before, the premises of the discussion regarding (4.10) also hold when analyzing (4.12). Therefore, we consider that function v_3 is minimized, while function v_2 is restricted, and that $\Delta = 1$ is used.

To conclude this section, we consider the bi-objective version of formulation `MRAA`, obtained by adding the set of constraints (3.10) to formulation `MRAA` to model the dissimilarity constraints. Revisiting the problem depicted in Figure 4.3, the five sets listed in Figure 4.3c are all efficient solutions for formulation `BOAR`. When preventing each arc from appearing in more than $R = 2$ paths from node 1 to node 5, the first three solutions in that table become unfeasible, and then there are two efficient solutions for formulation `BOARA`, shown in Figure 4.3d. It is interesting to observe that, unlike what happened when adding the new constraints to the problem presented in the previous section, in this case the number of efficient solutions decreased.

4.3 Computational results

Empirical experiments were run in order to evaluate the presented methods and formulations. The purpose of the tests is bifold: assess the extent to which the ϵ-constrained algorithm is efficient when increasing ϵ, and compare the decreasing and the increasing algorithms for finding sets of K shortest and dissimilar paths.

4.3.1 Test conditions

The two ϵ-constraint algorithms described in Section 4.1 were implemented for the four formulations presented in Section 4.2. The following codes were implemented:

- codes `DEC.BORA`, `DEC.BORAA`, `DEC.BOAR` and `DEC.BOARA`, for finding the non-dominated set using the decreasing ϵ-constraint method, Algorithm 2, for formulations (4.5), (4.10), (4.8) and (4.12), respectively;
- codes `IEC.BORA`, `IEC.BORAA`, `IEC.BOAR` and `IEC.BOARA`, for finding the non-dominated set using the increasing ϵ-constraint method, Algorithm 3, for formulations (4.5), (4.10), (4.8) and (4.12), respectively.

The eight variants of the methods were coded in C language, calling CPLEX 12.7 to solve the intermediate mixed-integer programs. As mentioned earlier, for all codes, the cost function was selected as f_1 to be minimized, and the overlaps function f_2 was included in the set of constraints. The update of the parameter ϵ consisted of decreasing or increasing as described in Algorithms 2 and 3, taking $\Delta = 1$.

The codes ran for two sets of instances, namely random graphs and grid graphs, such that:

- Random graphs, $R_{n,m}$, with $n = 100, 500$ nodes were obtained generating randomly $m = dn$ arcs, with $d = 5, 10, 15$.
- The grid graphs, $G_{p,q}$, comprised the following sizes: $p \times q = 4 \times 36, 12 \times 12, 5 \times 45, 15 \times 15$.

In either case each arc $(i, j) \in A$ was associated with an integer cost value, c_{ij}, uniformly obtained in $\{1, 2, \ldots, 100\}$. The results presented in the following correspond to average values obtained after finding sets of $K = 10$ paths over 20 different instances generated for each dimension of these data sets.

All the tests ran on a 64-bit PC with an Intel ®Core™ i7-6700 Quad Core at 3.40GHz with 64GB of RAM. For all tests we used a time limit of 300 seconds for each of the sub-problems solved along the generation of the non-dominated set. To ease the reading, the test statistics are summarized in Table 4.1.

Table 4.1 Description of the column headings

Heading	Description
$\bar{T}$	Average total run time, in seconds
$\lvert\bar{Y_E}\rvert$	Average number of computed non-dominated solutions
$\bar{N}$	Average number of solved sub-problems
$\bar{f}^j_{\min}$	Average value for the minimum value of f_j in each set of paths, $j = 1, 2$
$\bar{f}^j_{\max}$	Average value for the maximum value of f_j in each set of paths, $j = 1, 2$
$\bar{D}_{\min}$	Average value for the minimum of `AvDi` in each set of paths
$\bar{D}_{\max}$	Average value for the maximum of `AvDi` in each set of paths

4.3.2 Test results

We first consider the results for the unconstrained formulations afterwards focus on constrained formulations.

Unconstrained formulations The average results obtained when finding the non-dominated set for the BOP (4.5) using the unconstrained formulation that looks for the repeated arcs minimization, codes `DEC.BORA` and `IEC.BORA`, are presented on Tables 4.2 and 4.3. The codes were not able to solve some of the grid network instances until the end within the established time limit (of 300 seconds). Therefore, the corresponding results in the tables are marked with a *.

According to Table 4.2, and as expected, the range of the cost objective function, f_1, is significantly larger than the range of the number of repeated arcs, f_2. The latter is rather

Table 4.2 Characteristics of the non-dominated points with the unconstrained version of minimizing the number of repeated arcs

Instance	$\lvert\bar{Y_E}\rvert$	$\bar{f}^1_{\min}$	$\bar{f}^1_{\max}$	$\bar{f}^2_{\min}$	$\bar{f}^2_{\max}$
$R_{100,500}$	3.00	906.80	1 530.60	2.21	4.42
$R_{100,1000}$	4.00	555.50	1 097.50	1.00	4.15
$R_{100,1500}$	3.70	387.00	827.50	0.80	3.55
$R_{500,2500}$	5.25	1 239.40	2 044.30	1.93	6.31
$R_{500,5000}$	4.75	761.00	1 218.20	1.10	5.10
$R_{500,7500}$	4.95	481.00	895.80	0.70	4.75
$G^*_{12,12}$	7.00	6 596.00	9 386.50	16.00	22.00
$G^*_{4,36}$	1.00	14 676.00	14 676.00	38.00	38.00
$G^*_{15,15}$	13.00	7 842.00	12 417.75	16.00	28.00
$G^*_{5,45}$	1.00	18 660.50	18 660.50	48.00	48.00

*: Sub-problems interrupted after 300 seconds.

Table 4.3 Number of sub-problems, path dissimilarities and run times (in seconds) when minimizing the number of repeated arcs

		DEC.BORA					IEC.BORA				
Instance	$\lvert\bar{Y_E}\rvert$	$\bar{N}$	$\bar{D}_{\min}$	$\bar{D}_{\max}$	$\bar{T}$	$\bar{T}/\bar{N}$	$\bar{N}$	$\bar{D}_{\min}$	$\bar{D}_{\max}$	$\bar{T}$	$\bar{T}/\bar{N}$
$R_{100,500}$	3.21	4.21	0.000	0.518	4.746	1.18	4.73	0.000	0.518	3.631	0.77
$R_{100,1000}$	4.00	5.00	0.015	0.855	12.544	2.68	5.45	0.015	0.855	10.828	2.08
$R_{100,1500}$	3.70	4.70	0.010	0.905	12.941	2.55	4.75	0.000	0.905	11.650	2.23
$R_{500,2500}$	5.25	6.25	0.013	0.833	183.447	27.40	6.63	0.013	0.833	166.086	22.20
$R_{500,5000}$	4.75	5.75	0.006	0.867	178.601	28.47	6.25	0.006	0.867	174.767	25.96
$R_{500,7500}$	4.95	5.95	0.000	0.936	338.883	47.48	6.30	0.010	0.936	319.571	41.58
$G^*_{12,12}$	7.00	8.00	0.010	0.779	2 700.823	337.60	8.50	0.010	0.779	2 101.022	248.03
$G^*_{4,36}$	1.00	2.00	0.000	0.000	600.694	300.34	3.00	0.000	0.000	300.824	100.27
$G^*_{15,15}$	13.00	14.00	0.004	0.865	4 501.163	321.51	14.60	0.003	0.864	3 901.710	267.54
$G^*_{5,45}$	1.00	2.00	0.000	0.000	601.116	300.55	3.00	0.000	0.000	301.466	100.48

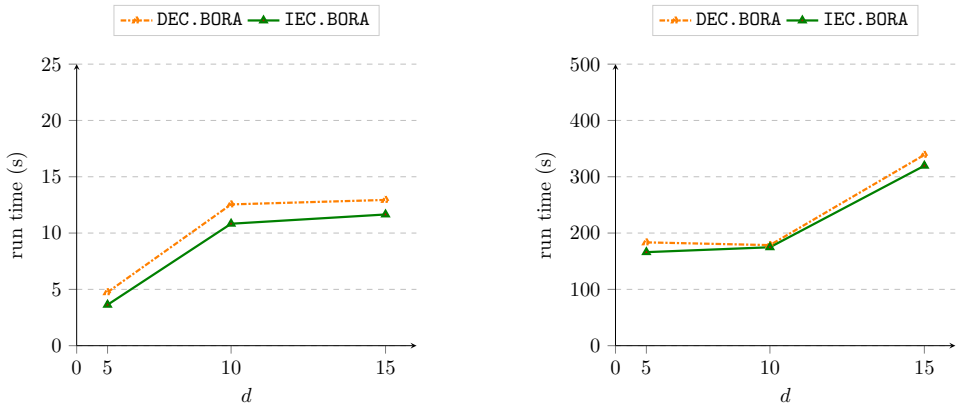
*: Sub-problems interrupted after 300 seconds.

small, which results in a small number of non-dominated points to be found, between 3 and 4 for random networks with 100 nodes and around 5 for random networks with 500 nodes. Also, the number of elements in Y_E seems to be very close to the range of the number of repeated arcs, given by function f_2. This indicates that there is approximately one non-dominated point for each of those values. Few conclusions can be drawn from the results on grids, as the code was not able to solve all the sub-problems within the time limit. Yet, in the case of rectangular grids only one solution was found, while, taking the range of function f_2 into account, for the square grids all the non-dominated points seem to have been found nonetheless.

Table 4.3 summarizes the results in terms of the number of solved sub-problems and solution dissimilarities for this set of codes. In average, one extra sub-problem had to be solved for finding the non-dominated set and this number increased to around 1.5 when talking about the increasing version of the ϵ-constrained algorithm. In the random instances the

average maximum dissimilarity grows with the average degree of the network, and specially with the number of nodes, varying from 0.518 to 0.936. The average minimum dissimilarity, on the other hand, is always nearly 0.

With respect to the grid networks, as noted before only one solution was found on rectangular grids, which means that the only problem solved consisted in the single objective minimization of the cost function. In general, in this case, the solution is simply formed by $K = 10$ paths (see Figures 4.2 and 4.3), all equal to the shortest path and with full overlapping arcs. This is also shown by the average values $\bar{D}_{\min}$ reported in Table 4.3, which are always very close to 0.



(a) $n = 100$ (b) $n = 500$

Fig. 4.4 Run times (in seconds) in random networks with the unconstrained version of minimizing the number of repeated arcs

The average run times for finding non-dominated sets when counting the number of repeated arcs are presented in Table 4.3 and in Figure 4.4. The average time for solving each sub-problem was shorter when applying the increasing version of the ϵ-constrained algorithm, and this speedup is observed for the total run times as well. Therefore, although the two methods are fairly similar in terms of time, the increasing version of the ϵ-constrained algorithm was consistently faster than the decreasing version for all instances. The partial run times depend mainly on the number of nodes in the random instances and also tend to increase with their average degree. The biggest random instances were solved by code `IEC.BORA` in less than 320 seconds. The results obtained for grids follow the same trend, with the difference that the corresponding sub-problems are much harder to solve than on the random networks. In this case, `IEC.BORA` required about 3 900 seconds to find 13 efficient sets of paths in 15×15 grids.

A comparison of the average results of the codes `DEC.BOAR` and `IEC.BOAR` for random and grid networks is summarized in Tables 4.4 and 4.5, whereas Figure 4.5 shows the run times for solving instances. In general, the code `DEC.BOAR` was faster than the code `IEC.BOAR` for both the random and the grid instances. The average dissimilarity for square grid networks was higher than for rectangular ones with the same number of nodes.

For the biggest random networks, comprising 500 nodes, the method with decreasing ϵ, `DEC.BOAR`, computed 34 solutions in an average time of 344.693 seconds, with minimum dissimilarity 0.017 and maximum dissimilarity 0.977. As for the method `IEC.BOAR`, the same

Table 4.4 Characteristics of the non-dominated points with the unconstrained version of minimizing the number of arc repetitions

Instance	$\lvert\bar{Y_E}\rvert$	$\bar{f}^1_{\min}$	$\bar{f}^1_{\max}$	$\bar{f}^2_{\min}$	$\bar{f}^2_{\max}$
$R_{100,500}$	24.68	906.80	1 821.40	11.42	39.78
$R_{100,1000}$	28.55	555.50	1 137.20	4.70	36.90
$R_{100,1500}$	27.25	387.00	888.30	2.70	31.85
$R_{500,2500}$	38.12	1 239.40	2163.90	8.50	56.50
$R_{500,5000}$	34.15	761.00	1 246.10	5.75	45.60
$R_{500,7500}$	34.36	481.00	917.10	2.63	40.78
$G_{12,12}$	139.35	6 596.00	10 363.80	40.00	196.85
$G_{4,36}$	102.80	14 676.00	17 337.50	214.00	342.00
$G_{15,15}$	188.20	7 842.00	12 629.90	40.00	251.25
$G_{5,45}$	167.60	18 660.50	22 169.40	228.00	431.80

Table 4.5 Number of sub-problems, path dissimilarities and run times (in seconds) when minimizing the number of arc repetitions

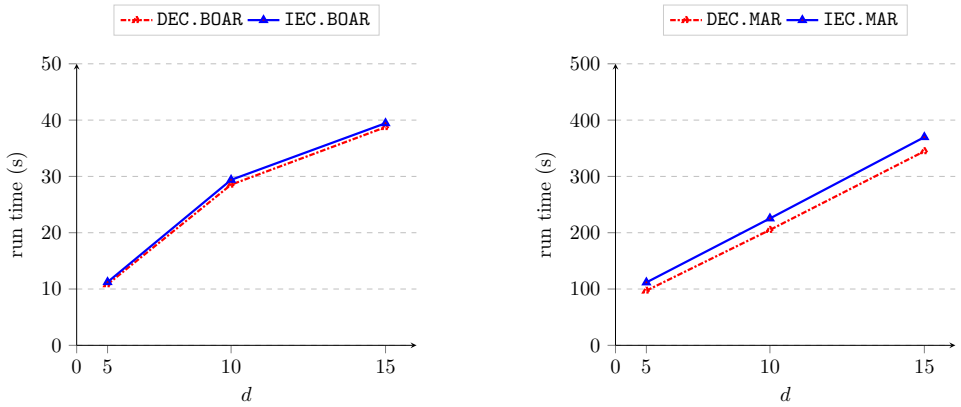
		DEC.BOAR					IEC.BOAR				
Instance	$\lvert\bar{Y_E}\rvert$	$\bar{N}$	$\bar{D}_{\min}$	$\bar{D}_{\max}$	$\bar{T}$	$\bar{T}/\bar{N}$	$\bar{N}$	$\bar{D}_{\min}$	$\bar{D}_{\max}$	$\bar{T}$	$\bar{T}/\bar{N}$
$R_{100,500}$	24.68	25.68	0.000	0.826	10.840	0.41	31.31	0.000	0.826	11.228	0.34
$R_{100,1000}$	28.55	29.55	0.015	0.931	28.531	0.92	35.20	0.012	0.931	29.388	0.80
$R_{100,1500}$	27.25	28.25	0.010	0.970	38.753	1.36	32.15	0.010	0.970	39.445	1.22
$R_{500,2500}$	38.12	39.12	0.013	0.910	97.103	2.48	50.93	0.013	0.910	111.698	2.20
$R_{500,5000}$	34.15	35.15	0.006	0.933	204.975	5.79	42.85	0.006	0.933	225.352	5.21
$R_{500,7500}$	34.36	35.36	0.017	0.977	344.693	9.84	41.15	0.017	0.977	369.727	8.86
$G_{12,12}$	139.35	140.35	0.016	0.919	264.034	1.88	158.90	0.010	0.919	268.740	1.69
$G_{4,36}$	102.80	103.80	0.000	0.675	167.345	1.61	130.20	0.000	0.675	183.490	1.40
$G_{15,15}$	188.20	189.20	0.008	0.936	602.591	3.18	213.25	0.008	0.936	615.861	2.88
$G_{5,45}$	167.60	168.60	0.001	0.757	537.591	3.17	205.85	0.000	0.757	583.330	2.83

solutions were obtained in an average time of 369.727 seconds. Finally, it should be recalled that the presented times refer to solving all sub-problems.

Figure 4.6 depicts the comparison of the costs of the solutions of each formulation in random and grid networks. In general the maximum cost for `BOAR` is bigger than for `BORA` in all networks. This difference is more observable in the sparser networks. The minimum costs for `BOAR` and `BORA` are the same.

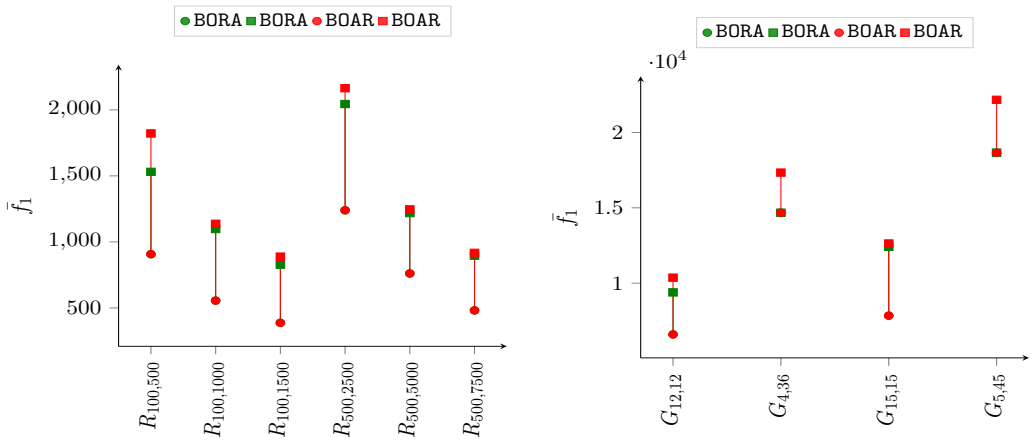
Figure 4.7 summarizes the comparison of the dissimilarities of the solutions obtained by `BORA` and `BOAR`. Clearly the minimum dissimilarity is 0 in the random and the grid networks. The code `BOAR` is better than `BORA` in terms of maximum dissimilarity for all kind of instances. Additionally, the maximum dissimilarity increases with d, in the random instances. Therefore, it is bigger in the denser networks. In grid networks, the maximum dissimilarity is higher in square grids than in rectangular ones.

Constrained formulations Tables 4.6 and 4.7 summarize the results obtained by the codes that implement the constrained version of `BORA`. The number of non-dominated points



(a) $n = 100$ (b) $n = 500$

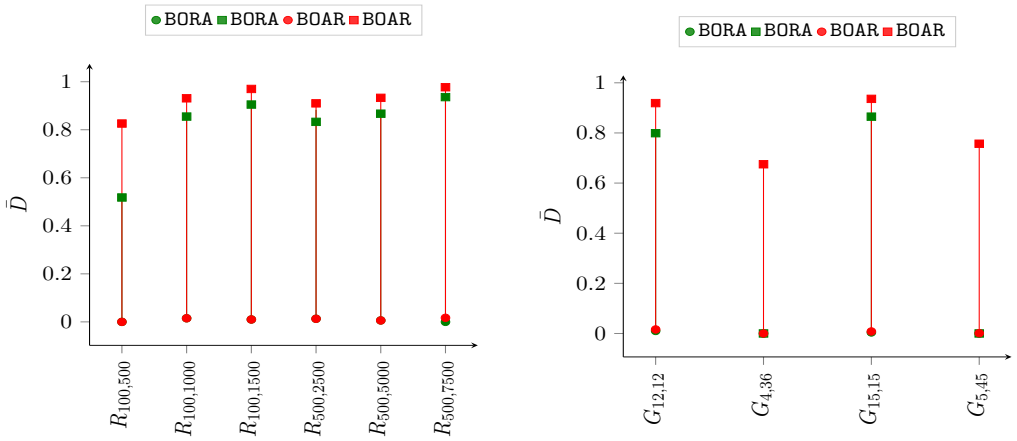
Fig. 4.5 Mean run times (in seconds) in random networks when minimizing the number of arcs repetitions



(a) Random networks (b) Grid networks

Fig. 4.6 Comparison of the costs for BORA and BOAR

change when constraint (3.10) is included in the formulation. Both the cost and the number of repeated arcs in the solutions change, as shown in the first table and in Figure 4.10. The problem also has more non-dominated points than the unconstrained, in average less than 16 in 500 node random networks and than 41 in 15×15 grids. It should be noted, though, that again not all sub-problems could be solved within the set time limit.



(a) Random networks

(b) Grid networks

Fig. 4.7 Comparison of the dissimilarity for `BORA` and `BOAR`

Table 4.6 Characteristics of the non-dominated points with the constrained version of minimizing the number of repeated arcs

Instance	$\lvert\bar{Y_E}\rvert$	$\bar{f}^1_{\min}$	$\bar{f}^1_{\max}$	$\bar{f}^2_{\min}$	$\bar{f}^2_{\max}$
$R_{100,500}$	7.84	1 264.37	1 825.05	3.74	11.05
$R_{100,1000}$	11.05	953.80	1 191.75	2.80	14.55
$R_{100,1500}$	11.45	784.25	934.10	2.40	16.25
$R_{500,2500}$	12.25	1 516.00	2 107.56	3.63	15.88
$R_{500,5000}$	15.15	1 057.70	1 329.35	3.30	22.05
$R_{500,7500}$	11.30	818.25	937.35	2.15	16.70
$G^*_{12,12}$	28.95	7 296.00	10 563.90	16.00	44.00
$G^*_{4,36}$	3.00	16 415.50	17 252.90	74.00	76.00
$G^*_{15,15}$	40.70	8 764.75	13 672.05	16.00	56.00
$G^*_{5,45}$	7.00	21 035.00	22 853.00	90.00	96.00

*: Sub-problems interrupted after 300 seconds.

Table 4.7 Number of sub-problems, path dissimilarities and run times (in seconds) with constrained version of minimizing the number of repeated arcs

		DEC.BORAA					IEC.BORAA				
Instance	$\lvert\bar{Y_E}\rvert$	$\bar{N}$	$\bar{D}_{\min}$	$\bar{D}_{\max}$	$\bar{T}$	$\bar{T}/\bar{N}$	$\bar{N}$	$\bar{D}_{\min}$	$\bar{D}_{\max}$	$\bar{T}$	$\bar{T}/\bar{N}$
$R_{100,500}$	7.84	8.84	0.514	0.814	15.634	1.76	10.31	0.514	0.814	15.367	1.49
$R_{100,1000}$	11.05	12.05	0.831	0.958	17.590	1.46	14.70	0.829	0.958	18.382	1.25
$R_{100,1500}$	11.45	12.45	0.903	0.982	12.350	0.99	16.85	0.903	0.982	14.783	0.87
$R_{500,2500}$	12.25	13.25	0.627	0.911	393.120	29.66	15.18	0.624	0.911	392.795	25.86
$R_{500,5000}$	15.15	16.15	0.729	0.940	164.442	10.18	21.45	0.730	0.940	177.550	8.27
$R_{500,7500}$	11.30	12.30	0.899	0.984	143.932	11.70	17.50	0.899	0.984	160.079	9.14
$G^*_{12,12}$	28.95	29.95	0.556	0.919	6 482.459	216.44	30.95	0.558	0.919	5 940.803	191.94
$G^*_{4,36}$	3.00	4.00	0.556	0.651	1 318.946	329.73	5.00	0.556	0.651	746.092	149.21
$G^*_{15,15}$	40.70	41.70	0.559	0.937	11 367.102	272.59	42.85	0.557	0.937	10 840.249	252.98
$G^*_{5,45}$	7.00	8.00	0.556	0.704	2 507.128	313.39	9.00	0.556	0.691	1 907.911	211.99

*: Sub-problems interrupted after 300 seconds.

Due to the higher number of solutions for the constrained problem, Table 4.7 also shows that the improvement in the time for solving each sub-problem does not always compensate the bigger number of sub-problems to be solved for the random instances. The improvement in the partial run times is bigger for grids, and therefore the IEC.BORAA was always faster than DEC.BORAA for these instances. Another important remark regards the improvement in the dissimilarity of the solutions. This improvement is constant in what the maximum average dissimilarity is concerned, but it is equally constant and bigger for the minimum average dissimilarity, which was always 0 with the unconstrained version.

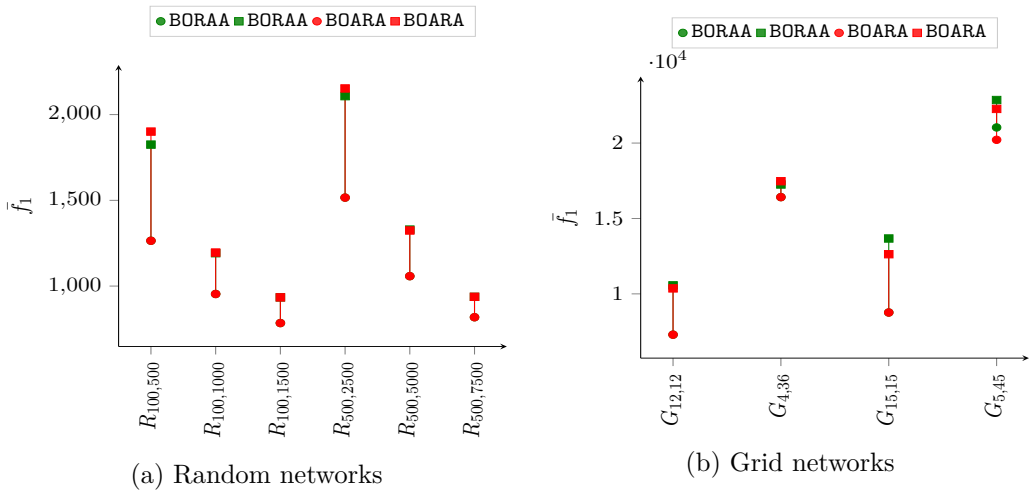
Table 4.8 Characteristics of the non-dominated points with the constrained version of minimizing the number of arc repetitions

Instance	$\lvert\bar{Y_E}\rvert$	$\bar{f}^1_{\min}$	$\bar{f}^1_{\max}$	$\bar{f}^2_{\min}$	$\bar{f}^2_{\max}$
$R_{100,500}$	22.10	1 264.36	1 900.94	12.00	37.57
$R_{100,1000}$	15.65	953.80	1 195.50	4.70	22.20
$R_{100,1500}$	11.00	784.25	934.10	2.70	17.35
$R_{500,2500}$	28.68	1 516.00	2 152.50	9.62	44.00
$R_{500,5000}$	20.15	1 057.70	1 324.15	5.95	33.90
$R_{500,7500}$	12.40	818.25	937.35	2.75	19.90
$G_{12,12}$	120.10	7 296.00	10 363.80	40.00	175.40
$G_{4,36}$	59.40	16 415.50	17 470.00	220.00	303.65
$G_{15,15}$	159.45	8 764.75	12 629.90	40.00	222.80
$G_{5,45}$	114.95	20 215.50	22 265.30	232.00	383.80

Table 4.9 Number of sub-problems, path dissimilarities and run times (in seconds) with constrained version of minimizing the number of arc repetitions

		DEC.BOARA					IEC.BOARA				
Instance	$\lvert\bar{Y}_E\rvert$	$\bar{N}$	$\bar{D}_{\min}$	$\bar{D}_{\max}$	$\bar{T}$	$\bar{T}/\bar{N}$	$\bar{N}$	$\bar{D}_{\min}$	$\bar{D}_{\max}$	$\bar{T}$	$\bar{T}/\bar{N}$
$R_{100,500}$	22.10	23.10	0.514	0.853	12.117	0.52	28.57	0.514	0.853	13.119	0.45
$R_{100,1000}$	15.65	16.65	0.831	0.959	20.142	1.21	20.50	0.830	0.959	22.603	1.10
$R_{100,1500}$	11.00	12.00	0.903	0.982	23.524	1.96	17.65	0.903	0.982	30.030	1.70
$R_{500,2500}$	28.68	29.68	0.627	0.923	91.706	3.08	37.25	0.627	0.923	105.053	2.82
$R_{500,5000}$	20.15	21.15	0.730	0.947	157.248	7.43	30.90	0.730	0.947	203.888	6.59
$R_{500,7500}$	12.40	13.40	0.900	0.984	180.441	13.46	20.15	0.900	0.984	231.502	11.48
$G_{12,12}$	120.10	121.10	0.559	0.920	221.821	1.83	137.65	0.558	0.920	228.539	1.66
$G_{4,36}$	59.40	60.40	0.556	0.746	94.450	1.56	85.95	0.556	0.746	116.652	1.35
$G_{15,15}$	159.45	160.45	0.560	0.937	517.730	3.22	184.85	0.560	0.937	537.129	2.90
$G_{5,45}$	114.95	115.95	0.556	0.796	374.270	3.22	154.20	0.556	0.796	440.711	2.85

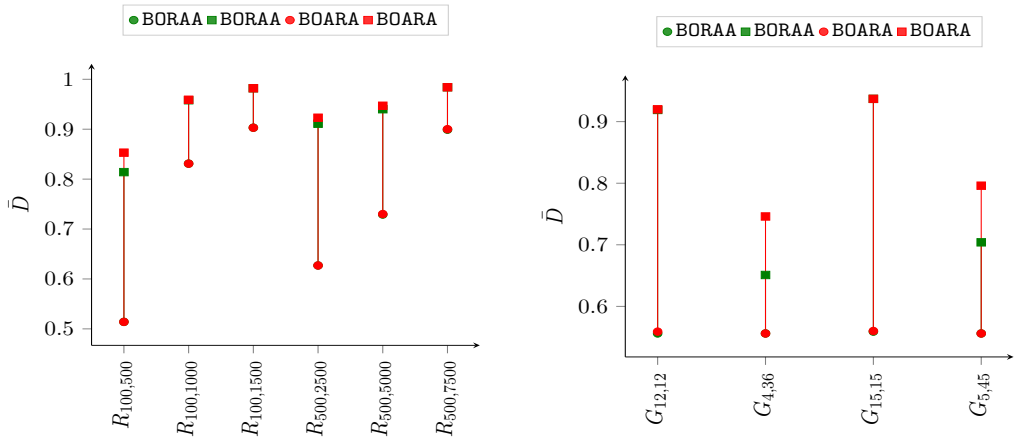
Figure 4.8 allows a comparison of the results obtained by BORAA and BOARA. Both the minimum and maximum cost are sensitive to the average degree of the network in random networks. They decrease when d increases. In total, the costs for the BOARA are higher than for the BORAA. In grid networks, the costs for rectangular grids are higher than the square ones for both models. We should also consider that BORAA has bigger maximum cost than BOARA in grids $G_{15,15}$ and $G_{5,45}$.



(a) Random networks

(b) Grid networks

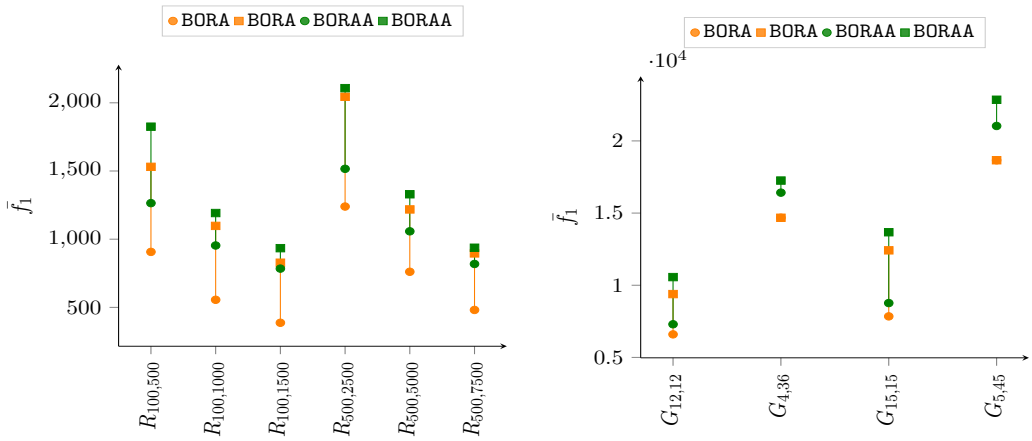
Fig. 4.8 Comparison of the costs for BORAA and BOARA



(a) Random networks (b) Grid networks

Fig. 4.9 Comparison of the dissimilarity for `BORAA` and `BOARA`

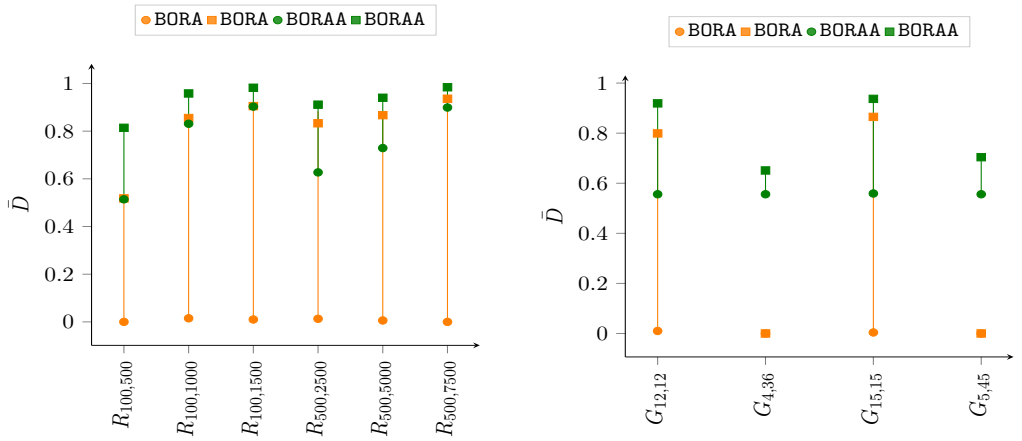
Figure 4.9 depicts the comparison of dissimilarities for constrained formulations. The best scores are associated to `BOARA` in random and grid networks. Like what was seen for unconstrained formulations, the dissimilarity increases in denser networks. We thus conclude that adding the capacity constraints improves the minimum dissimilarity in random networks and grid networks.



(a) Random networks (b) Grid networks

Fig. 4.10 Comparison of the costs for the unconstrained and the constrained problems when minimizing the number of repeated arcs

Comparison of the costs for the unconstrained and the constrained problems when minimizing the number of repeated arcs are depicted in Figure 4.10. The plots show that adding the constraints increases the minimum and maximum costs in all kind of networks. The costs decrease with the increasing of the average degree of the network and the costs are higher in rectangular grids than square ones.

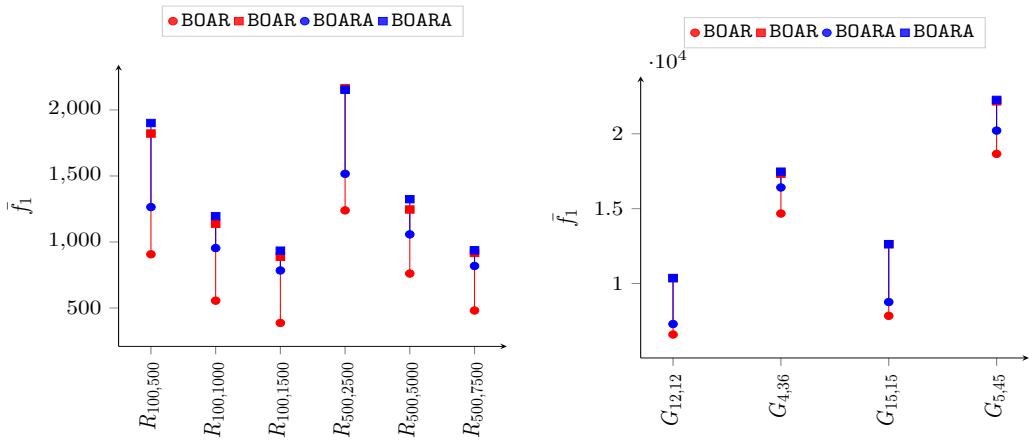


(a) Random networks

(b) Grid networks

Fig. 4.11 Comparison of the dissimilarity for the unconstrained and the constrained problems when minimizing the number of repeated arcs

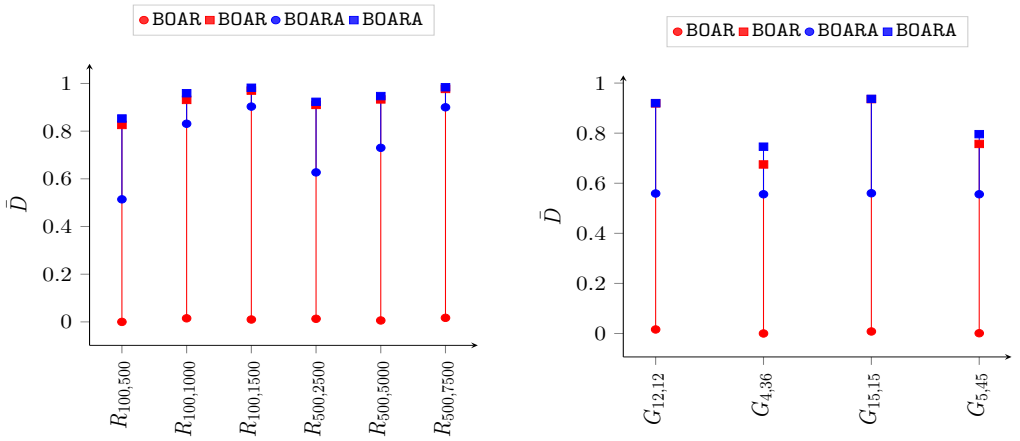
Figure 4.11 summarizes comparison of the dissimilarities for the unconstrained and the constrained problems when minimizing the number of repeated arcs for random and grid networks. The plots show that the additional constraints increase the minimum and maximum dissimilarity in all kind of networks. The difference between minimum and maximum dissimilarity in constrained formulations is small in denser networks. This difference in rectangular grids is smaller than square ones in grid networks.



(a) Random networks

(b) Grid networks

Fig. 4.12 Comparison of the costs for the unconstrained and the constrained problems when minimizing the number of arc repetitions



(a) Random networks

(b) Grid networks

Fig. 4.13 Comparison of the dissimilarity for the unconstrained and the constrained problems when minimizing the number of arc repetitions

Figures 4.12 and 4.13 illustrate the comparison of the cost and dissimilarity for the unconstrained and the constrained problems when minimizing the number of arc repetitions respectively. The conclusions are similar to the previous plots for minimizing the number of repeated arcs.

4.4 Concluding remarks

In this chapter, we addressed the problem of finding set of K paths that are dissimilar as much as possible, while minimizing the cost. A bi-objective approach to this problem (the BOP problem) was introduced. The approach consists of a modification of the ϵ-constraint method, while finding the nondominated solutions. The two versions of the proposed algorithm were compared on random and grid instances and extensive computational results were obtained and discussed.

The increasing ϵ-constraint method outperformed the original when applied to the formulations based in `BORA`, as the number of non-dominated points is not very large and the sub-problems are difficult to solve. Contrarily, the original version of the ϵ-constraint method was more efficient than the increasing version when applied to `BOAR`. In both cases, the constrained version of the problems, namely `BORAA` and `BOARA` greatly improved the dissimilarity of the solutions up to 0.937, specially the minimum dissimilarity, allowing to discard some solutions of little practical interest. In fact, the minimum dissimilarity for unconstrained problems was near 0, while for the constrained version of the problems, it was at least 0.514 in the random, and 0.556 in the grid, networks. However, the cost of the solutions slightly increased as well. The code `BORA` was faster than `BOAR` for the random instances, because less non-dominated points are computed in that case, while the opposite happened for the grid networks. As to the run times for the constrained version of the problems, either they were not affected by the new constraints or even decreased, and the two approaches behaved very similarly with this respect.

Chapter 5

Conclusion

The search for alternative paths linking two fixed nodes arises as a natural question related with an important combinatorial problem with many interesting applications. If no further conditions are imposed it is highly likely that the several alternatives have a lot in common, which may be undesired. Similarity indexes measure the degree to which objects resemble one another, while dissimilarity represents the inverse situation. Including the concept of dissimilarity as one of the objectives of a paths problem is not trivial, though. Despite the obvious interest of including dissimilarity as the goal of paths problems when looking for several solution, the complexity of those indexes makes difficult its integration in problems. Therefore, many works have gave dealt with it as a metric used as a way of filtering solutions a posteriori, rather than incorporating it as the main objective.

Instead we look at dissimilarity as the objective of the problem, when considering the determination of a set of K paths between a given pair of nodes, and consider alternative linear integer formulation that attempt to model it.

After revising some literature related with paths and routing problems involving dissimilarity requirements, in Chapter 3 the K dissimilar paths problem was defined and integer linear formulations were introduced with the goal of capturing its main characteristics, while assuming the simplification that all paths are of the same length. Three types of approaches were proposed:

- A formulation of combinatorial nature, `MAO`, which takes the overlaps between all pairs of K paths into account. This formulation provides exact dissimilarities under certain assumptions and it showed empirically to be accurate in other cases as well, which can also be useful for assessing the quality of the solutions obtained by the remaining formulations. The main handicap of this formulation is that it is too sensitive to the size of the instances, and, thus, its practical application becomes rather limited.

- A simpler formulation focused on the minimization of the number of arcs that appear in more than one path, `MRA`. Although able of solving a wider range of instances than `MAO`, the dissimilarities it produced were far from the best.

- Two other formulations focused the total number of times that the shared arcs appear in the paths, `MRO` and `MAR`, which somehow extends the latter. One of these formulations

showed a very good compromise with respect to the integer programming gaps, the instances it was able to solve under a fixed time limit, and the dissimilarity of the solutions.

Additionally, the formulation `MRA` was combined with a set of capacity constraints, in order to prevent the dissimilarity from worsening in some cases. These constraints were also added to the formulation `MAR`. While the impact on `MRA` was significant, it is not clear whether the improvement in the dissimilarity of `MAR` pays off, as it also slows down the method.

This preliminary study of the K dissimilar paths problem is important for setting the foundations for the K shortest and dissimilar paths problem addressed in Chapter 4, which is clearly more challenging to solve but also more interesting from an application point of view. In this case the formulations `MRA` and `MAR`, both unconstrained and with capacity constraints were extended to the bi-objective case. The ϵ-constraint method was adapted for finding the non-dominated points for the new problems. A new ϵ-constraint method was developed with the same purpose. It takes advantage of the way the parameter ϵ is updated to use information obtained earlier for solving the several sub-problems. The drawback of this approach is that it may require solving more sub-problems.

The increasing ϵ-constraint method outperformed the original when applied to the formulations based in `BORA`, as the number of non-dominated points is not very large and the sub-problems are difficult to solve. Contrarily, the original version was more efficient than the increasing ϵ-constraint method when applied to `BOAR`. In both cases, the constrained version of the problems greatly improved the dissimilarity of the solutions, specially the minimum, allowing to discard some solutions of little practical interest. However, the cost of the solutions slightly increased. As to the run times, either they were not affected by the new constraints or even decreased.

To our knowledge this is the first time that integer linear formulations are developed with the goal of addressing directly the optimization of dissimilarity. In this sense the present work is more a starting point than a finished study and several questions remain to be investigated as well as others are raised. For instance, it would be most interesting to investigate a simpler/smaller alternative formulation to `MAO`, accounting for the number of overlaps between each pair of paths, given that this is more in accordance with the dissimilarity measure that has been used for assessing the solutions. Knowing the optimum dissimilarity value for a set of instances would also enable a more accurate assessment of the ability of each proposed formulation to model that measure.

Another important line of research, would be to better understand the relation between the dissimilarity and the run times of the formulations for the K dissimilar paths problem and some influencing characteristics of the networks, such as their density, average degree and topology. This aspect would allow a better use of the new models, particularly in the context of real world applications of the problem, for, in those cases, a wider variety of networks can arise.

As mentioned above, the total number of arcs of the solutions was not taken into account in the proposed formulations, despite the fact that it is part of the definition of dissimilarity. Therefore, it is worth considering unitary arc costs in the K shortest and dissimilar paths

problem, and, thus, analyzing the relation between the dissimilarity of the solutions and their total number of arcs.

References

[1] Akgün, V., Erkut, E., and Batta, R. (2000). On finding dissimilar paths. *European Journal of Operational Research*, 121:232–246.

[2] Bhandari, R. (1998). *Survivable Networks: Algorithms for Diverse Routing*. Kluwer Academic Publishers, USA.

[3] Boland, N., Charkhgard, H., and Savelsbergh, M. (2015). A criterion space search algorithm for biobjective integer programming: The balanced box method. *INFORMS Journal on Computing*, 27(4):735–754.

[4] Bornstein, C., Maculan, N., Pascoal, M., and Pinto, L. (2012). Multiobjective combinatorial optimization problems with a cost and several bottleneck objective functions: an algorithm with reoptimization. *Computers & Operations Research*, 39(9):1969–1976.

[5] Calvo, R. and Cordone, R. (2003). A heuristic approach to the overnight security service problem. *Computers & Operations Research*, 30(9):1269–1287.

[6] Caramia, M., Giordani, S., and Iovanella, A. (2010). On the selection of k routes in multiobjective hazmat route planning. *IMA Journal of Management Mathematics*, 21(3):239–251.

[7] Carotenuto, P., Giordani, S., and Ricciardelli, S. (2007a). Finding minimum and equitable risk routes for hazmat shipments. *Computers & Operations Research*, 34(5):1304–1327.

[8] Carotenuto, P., Giordani, S., Ricciardelli, S., and Rismondo, S. (2007b). A tabu search approach for scheduling hazmat shipments. *Computers & Operations Research*, 34(5):1328–1350.

[9] Chankong, V. and Haimes, Y. (1983). *Multiobjective decision making*, volume 8 of *North-Holland Series in System Science and Engineering*. North-Holland Publishing Co., New York.

[10] Clímaco, J. and Martins, E. (1982). A bicriterion shortest path algorithm. *European Journal of Operational Research*, 11(4):399 – 404.

[11] Clímaco, J. and Pascoal, M. (2012). Multicriteria path and tree problems: discussion on exact algorithms and applications. *International Transactions in Operational Research*, 19(1-2):63–98.

[12] Cohon, J. (1978). *Multiobjective programming and planning*. Academic Press, New York.

[13] Constantino, M., Mourão, M. C., and Pinto, L. (2017). Dissimilar arc routing problems. *Networks*, 70(3):233–245.

[14] Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2009). *Introduction to algorithms*. The MIT press.

[15] Dell'Olmo, P., Gentili, M., and Scozzari, A. (2005). On finding dissimilar Pareto-optimal paths. *European Journal of Operational Research*, 162:70–82.

[16] Ehrgott, M. (2005). *Multicriteria optimization*. Springer-Verlag, Berlin, second edition.

[17] Ehrgott, M. and Gandibleux, X. (2000). A survey and annotated bibliography of multiobjective combinatorial optimization. *OR-Spektrum*, 22(4):425–460.

[18] Ehrgott, M. and Ruzika, S. (2008). Improved ε-constraint method for multiobjective programming. *Journal of Optimization Theory and Applications*, 138(3):375.

[19] Eppstein, D. (1998). Finding the k shortest paths. *SIAM Journal on Computing*, 28(2):652–673.

[20] Erkut, E. (1990). The discrete p-dispersion problem. *European Journal of Operational Research*, 46:48–60.

[21] Erkut, E. and Verter, V. (1998). Modeling of transport risk for hazardous materials. *Operations Research*, 46:625–642.

[22] Frank, A. (1985). Edge-disjoint paths in planar graphs. *Journal of Combinatorial Theory, Series B*, 39(2):164–178.

[23] Gomes, T. and Craveirinha, J. (2010). An algorithm for enumerating srlg diverse path pairs. *Journal of Telecommunications and Information Technology*, pages 5–12.

[24] Gomes, T., Jorge, L., Melo, P., and Girão-Silva, R. (2016). Maximally node and srlg-disjoint path pair of min-sum cost in gmpls networks: a lexicographic approach. *Photonic Network Communications*, 31(1):11–22.

[25] Gopalan, R., Kolluri, K., Batta, R., and Karwan, M. (1990). Modeling equity of risk in the transportation of hazardous materials. *Operations research*, 38(6):961–973.

[26] Guruswami, V., Khanna, S., Rajaraman, R., Shepherd, B., and Yannakakis, M. (2003). Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. *Journal of Computer and System Sciences*, 67(3):473–496.

[27] Haimes, Y., Lasdon, L., and Wismer, D. (1971). On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Trans. Systems Man Cybernet.*, SMC-1:296–297.

[28] Hansen, P. (1980). Bicriterion path problems. In Fandel, G. and Gal, T., editors, *Multiple Criteria Decision Making Theory and Application*, pages 109–127, Berlin, Heidelberg. Springer Berlin Heidelberg.

[29] Hoffman, W. and Pavley, R. (1959). A method for the solution of the n-th best path problem. *Journal of the ACM*, 6(4):506–514.

[30] Iori, M., Martello, S., and Pretolani, D. (2010). An aggregate label setting policy for the multi-objective shortest path problem. *European Journal of Operational Research*, 207(3):1489–1496.

[31] Iqbal, F. and Kuipers, F. (2015). *Wiley Encyclopedia of Electrical and Electronics Engineering*, chapter Disjoint paths in networks, pages 1–14. Wiley, New York.

[32] Jiménez, V. and Marzal, A. (1999). Computing the k shortest paths: A new algorithm and an experimental comparison. In *International Workshop on Algorithm Engineering*, pages 15–29. Springer.

[33] Johnson, P., Joy, D., and Clarke, D. (1992). Highway 3.01, an enhancement routing model: program, description, methodology and revised user's manual. *Arbeitsbericht, Oak Ridge National Laboratories, Washington, DC*.

[34] Katoh, N., Ibaraki, T., and Mine, H. (1982). An efficient algorithm for k shortest simple paths. *Networks*, 12(4):411–427.

[35] Kuby, M., Zhongyi, X., and Xiaodong, X. (1997). A minimax method for finding the k best "differentiated" paths. *Geographical Analysis*, 29:298–313.

[36] Lombard, K. and Church, R. (1993). The gateway shortest path problem: Generating alternative routes for a corridor location problem. *Geographical Systems*, 1(1):25–45.

[37] Marler, R. and Arora, J. (2010). The weighted sum method for multi-objective optimization: new insights. *Structural and multidisciplinary optimization*, 41(6):853–862.

[38] Martí, R., González-Velarde, J., and Duarte, A. (2009). Heuristics for the bi-objective path dissimilarity problem. *Computers & Operations Research*, 36(11):2905–2912.

[39] Martins, E. (1984). An algorithm for ranking paths that may contain cycles. *European Journal of Operational Research*, 18(1):123–130.

[40] Martins, E. and Pascoal, M. (2003). A new implementation of Yen's ranking loopless paths algorithm. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 1(2):121–133.

[41] Martins, E., Pascoal, M., and Santos, J. (1999). Deviation algorithms for ranking shortest paths. *International Journal of Foundations of Computer Science*, 10(3):247–261.

[42] Mavrotas, G. (2009). Effective implementation of the ε-constraint method in multi-objective mathematical programming problems. *Applied mathematics and computation*, 213(2):455–465.

[43] Miettinen, K. (2012). *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media.

[44] Miettinen, K., Ruiz, F., and Wierzbicki, A. (2008). *Introduction to Multiobjective Optimization: Interactive Approaches*, pages 27–57. Springer Berlin Heidelberg, Berlin, Heidelberg.

[45] Moghanni, A., Pascoal, M., and Godinho, M. T. (2020a). Finding k dissimilar paths using integer linear formulations. *Submitted for publication*.

[46] Moghanni, A., Pascoal, M., and Godinho, M. T. (2020b). Finding k shortest and dissimilar paths. *Submitted for publication*.

[47] Parragh, S. N. and Tricoire, F. (2019). Branch-and-bound for bi-objective integer programming. *INFORMS Journal on Computing*, 31(4):805–822.

[48] Pascoal, M., Captivo, M. E., Clímaco, J., and Laranjeira, A. (2013). Bicriteria path problem minimizing the cost and minimizing the number of labels. *4OR - Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 11(3):275–294.

[49] Pascoal, M. and Clímaco, J. (2020). On a relaxed maximally disjoint path pair problem: a bicriteria approach. *International Transactions in Operational Research*, 27(4):2045–2063.

[50] Perini, T., Boland, N., Pecin, D., and Savelsbergh, M. (2020). A criterion space method for biobjective mixed integer programming: The boxed line method. *INFORMS Journal on Computing*, 32(1):16–39.

[51] Raith, A. and Ehrgott, M. (2009). A comparison of solution strategies for biobjective shortest path problems. *Computers & Operations Research*, 36(4):1299 – 1331.

[52] Suurballe, J. (1974). Disjoint paths in a network. *Networks*, 4(2):125–145.

[53] Suurballe, J. and Tarjan, R. (1984). A quick method for finding shortest pairs of disjoint paths. *Networks*, 14(2):325–336.

[54] Ulungu, E. L. and Teghem, J. (1994). Multi-objective combinatorial optimization problems: A survey. *Journal of Multi-Criteria Decision Analysis*, 3(2):83–104.

[55] Vincke, P. (1974). Problèmes multicritères. *Cahiers Centre Études Rech. Opér.*, 16:425–439. Colloque sur la Programmation Mathématique (Brussels, 1974).

[56] Vygen, J. (1995). NP-completeness of some edge-disjoint paths problems. *Discrete Applied Mathematics*, 61(1):83–90.

[57] Yen, J. (1971). Finding the k shortest loopless paths in a network. *Management Science*, 17(11):712–716.

[58] Zajac, S. (2018). On a two-phase solution approach for the bi-objective k-dissimilar vehicle routing problem. *Journal Heuristics*, 24(3):515–550.

[59] Zeleny, M. (1973). Compromise programming. In Cochrane, J. and Zeleny, M., editors, *Multiple Criteria Decision Making*, pages 262–301. University of South Carolina Press, Columbia.

Appendix A

A.1 Dissimilarities for the formulations

Table A.1 Average `AvDi` and `MiDi` of `MAO` in random networks

`AvDi`	K								
$R_{n,m}$	3	4	5	6	7	8	9	10	
$R_{100,500}$	0.855	0.913	0.911	0.904	0.902	0.891	0.887	0.886	
$R_{100,1000}$	–	0.942	0.965	0.967	0.968	0.973	0.970	0.970	
$R_{300,1500}$	0.922	0.947	0.952	0.954	0.946	0.947	0.940	0.937	
$R_{300,3000}$	0.979	0.970	0.976	0.972	0.971	0.972	0.975	0.973	
$R_{500,2500}$	0.957	0.967	0.961	0.961	0.956	0.958	0.952	0.947	
$R_{500,5000}$	0.900	0.948	0.940	0.953	0.956	0.957	0.967	0.958	
Average	0.923	0.948	0.951	0.951	0.952	0.950	0.950	0.950	0.944
`MiDi`	K								
$R_{n,m}$	3	4	5	6	7	8	9	10	
$R_{100,500}$	0.855	0.763	0.720	0.691	0.650	0.615	0.555	0.547	
$R_{100,1000}$	–	0.654	0.750	0.740	0.700	0.698	0.682	0.684	
$R_{300,1500}$	0.822	0.813	0.801	0.782	0.763	0.745	0.704	0.705	
$R_{300,3000}$	0.938	0.850	0.829	0.971	0.723	0.691	0.721	0.707	
$R_{500,2500}$	0.887	0.882	0.822	0.814	0.774	0.792	0.730	0.720	
$R_{500,5000}$	0.819	0.862	0.819	0.794	0.778	0.760	0.780	0.756	
Average	0.864	0.804	0.790	0.761	0.731	0.731	0.717	0.696	0.687

Table A.2 Average AvDi and MiDi of MRA in random networks

AvDi	K								
$R_{n,m}$	3	4	5	6	7	8	9	10	
$R_{100,500}$	0.852	0.895	0.885	0.862	0.826	0.803	0.794	0.741	
$R_{100,1000}$	–	0.933	0.951	0.939	0.944	0.940	0.937	0.877	
$R_{300,1500}$	0.915	0.942	0.933	0.925	0.913	0.904	0.901	0.884	
$R_{300,3000}$	0.979	0.924	0.971	0.886	0.891	0.892	0.897	0.896	
$R_{500,2500}$	0.952	0.950	0.945	0.935	0.929	0.907	0.904	0.889	
$R_{500,5000}$	0.879	0.929	0.927	0.938	0.940	0.934	0.933	0.922	
Average	0.915	0.929	0.935	0.914	0.907	0.897	0.894	0.868	0.909
MiDi	K								
$R_{n,m}$	3	4	5	6	7	8	9	10	
$R_{100,500}$	0.772	0.735	0.720	0.666	0.603	0.532	0.513	0.416	
$R_{100,1000}$	–	0.726	0.746	0.753	0.743	0.727	0.723	0.652	
$R_{300,1500}$	0.817	0.824	0.815	0.766	0.750	0.729	0.695	0.642	
$R_{300,3000}$	0.938	0.833	0.850	0.744	0.741	0.704	0.722	0.710	
$R_{500,2500}$	0.888	0.866	0.848	0.803	0.804	0.742	0.738	0.704	
$R_{500,5000}$	0.798	0.831	0.811	0.806	0.795	0.791	0.766	0.734	
Average	0.843	0.802	0.798	0.756	0.739	0.704	0.693	0.643	0.751

Table A.3 Average AvDi and MiDi of MRO in random networks

AvDi	K								
$R_{n,m}$	3	4	5	6	7	8	9	10	
$R_{100,500}$	0.866	0.906	0.885	0.865	0.849	0.828	0.815	0.808	
$R_{100,1000}$	–	0.978	0.957	0.944	0.951	0.943	0.932	0.931	
$R_{300,1500}$	0.931	0.947	0.938	0.929	0.916	0.912	0.907	0.904	
$R_{300,3000}$	0.963	0.974	0.971	0.968	0.956	0.951	0.941	0.931	
$R_{500,2500}$	0.942	0.965	0.952	0.942	0.932	0.929	0.929	0.923	
$R_{500,5000}$	0.919	0.941	0.935	0.950	0.948	0.944	0.942	0.938	
Average	0.924	0.952	0.940	0.933	0.925	0.918	0.911	0.906	0.926
MiDi	K								
$R_{n,m}$	3	4	5	6	7	8	9	10	
$R_{100,500}$	0.763	0.758	0.719	0.674	0.632	0.577	0.533	0.525	
$R_{100,1000}$	–	0.869	0.757	0.749	0.755	0.711	0.703	0.710	
$R_{300,1500}$	0.835	0.836	0.792	0.789	0.746	0.740	0.707	0.687	
$R_{300,3000}$	0.889	0.899	0.858	0.831	0.759	0.763	0.736	0.707	
$R_{500,2500}$	0.851	0.893	0.862	0.837	0.814	0.787	0.780	0.710	
$R_{500,5000}$	0.841	0.849	0.832	0.833	0.805	0.809	0.777	0.793	
Average	0.836	0.851	0.803	0.785	0.752	0.731	0.706	0.689	0.768

Table A.4 Average `AvDi` and `MiDi` of `MAR` in random networks

`AvDi`	K								
$R_{n,m}$	3	4	5	6	7	8	9	10	
$R_{100,500}$	0.876	0.903	0.895	0.878	0.871	0.858	0.849	0.845	
$R_{100,1000}$	–	0.962	0.957	0.958	0.964	0.957	0.955	0.955	
$R_{300,1500}$	0.950	0.949	0.950	0.940	0.936	0.930	0.920	0.913	
$R_{300,3000}$	0.977	0.969	0.970	0.973	0.968	0.965	0.965	0.960	
$R_{500,2500}$	0.956	0.962	0.957	0.954	0.951	0.945	0.942	0.934	
$R_{500,5000}$	0.925	0.948	0.938	0.949	0.954	0.948	0.957	0.952	
Average	0.937	0.949	0.945	0.942	0.941	0.934	0.931	0.926	0.938
`MiDi`	K								
$R_{n,m}$	3	4	5	6	7	8	9	10	
$R_{100,500}$	0.820	0.751	0.720	0.657	0.635	0.567	0.537	0.503	
$R_{100,1000}$	–	0.773	0.739	0.725	0.727	0.717	0.732	0.685	
$R_{300,1500}$	0.895	0.827	0.818	0.780	0.762	0.722	0.713	0.678	
$R_{300,3000}$	0.930	0.868	0.833	0.842	0.808	0.748	0.730	0.712	
$R_{500,2500}$	0.891	0.869	0.841	0.836	0.806	0.774	0.779	0.716	
$R_{500,5000}$	0.875	0.866	0.820	0.813	0.806	0.777	0.809	0.754	
Average	0.882	0.825	0.795	0.775	0.757	0.717	0.717	0.675	0.766

Table A.5 Average `AvDi` and `MiDi` of `MAO` in grid networks

`AvDi`	K								
$G_{p,q}$	3	4	5	6	7	8	9	10	
$G_{3,12}$	0.949	0.833	0.800	0.787	0.762	0.745	0.741	0.730	
$G_{4,36}$	0.982	0.982	0.892	0.859	0.848	0.845	0.819	0.811	
$G_{6,6}$	0.933	0.933	0.900	0.893	0.867	0.857	0.844	0.840	
$G_{12,12}$	0.970	0.970	0.955	0.952	0.939	0.935	0.929	0.927	
Average	0.959	0.930	0.887	0.874	0.854	0.854	0.833	0.876	0.883
`MiDi`	K								
$G_{p,q}$	3	4	5	6	7	8	9	10	
$G_{3,12}$	0.846	0.538	0.231	0.154	0.154	0.154	0.000	0.000	
$G_{4,36}$	0.974	0.947	0.526	0.158	0.105	0.105	0.105	0.053	
$G_{6,6}$	0.900	0.900	0.700	0.700	0.600	0.500	0.500	0.500	
$G_{12,12}$	0.955	0.909	0.909	0.864	0.773	0.818	0.818	0.727	
Average	0.919	0.824	0.592	0.469	0.408	0.394	0.356	0.320	0.535

Table A.6 Average `AvDi` and `MiDi` of `MRA` in grid networks

`AvDi`	K								
$G_{p,q}$	3	4	5	6	7	8	9	10	
$G_{3,12}$	0.949	0.808	0.538	0.308	0.264	0.231	0.205	0.185	
$G_{4,36}$	0.982	0.982	0.718	0.584	0.560	0.570	0.414	0.506	
$G_{6,6}$	0.933	0.867	0.800	0.893	0.867	0.521	0.519	0.511	
$G_{12,12}$	0.970	0.970	0.955	0.952	0.939	0.929	0.924	0.758	
Average	0.959	0.907	0.753	0.684	0.657	0.563	0.516	0.490	0.691
`MiDi`	K								
$G_{p,q}$	3	4	5	6	7	8	9	10	
$G_{3,12}$	0.923	0.462	0.308	0.000	0.000	0.000	0.000	0.000	
$G_{4,36}$	0.974	0.947	0.132	0.158	0.132	0.000	0.000	0.000	
$G_{6,6}$	0.900	0.700	0.400	0.600	0.000	0.000	0.000	0.000	
$G_{12,12}$	0.955	0.955	0.909	0.818	0.773	0.727	0.773	0.318	
Average	0.938	0.766	0.437	0.419	0.376	0.182	0.193	0.080	0.424

Table A.7 Average `AvDi` and `MiDi` of `MRO` in grid networks

`AvDi`	K								
$G_{p,q}$	3	4	5	6	7	8	9	10	
$G_{3,12}$	0.949	0.808	0.708	0.641	0.597	0.569	0.551	0.542	
$G_{4,36}$	0.982	0.982	0.892	0.804	0.729	0.650	0.621	0.581	
$G_{6,6}$	0.933	0.933	0.900	0.893	0.867	0.857	0.833	0.822	
$G_{12,12}$	0.970	0.970	0.955	0.952	0.939	0.935	0.924	0.919	
Average	0.959	0.927	0.864	0.822	0.783	0.753	0.732	0.716	0.819
`MiDi`	K								
$G_{p,q}$	3	4	5	6	7	8	9	10	
$G_{3,12}$	0.923	0.538	0.462	0.308	0.308	0.308	0.308	0.231	
$G_{4,36}$	0.947	0.974	0.132	0.132	0.132	0.158	0.132	0.132	
$G_{6,6}$	0.900	0.800	0.700	0.700	0.600	0.400	0.200	0.200	
$G_{12,12}$	0.909	0.909	0.864	0.864	0.818	0.773	0.727	0.773	
Average	0.920	0.805	0.539	0.501	0.464	0.410	0.342	0.334	0.539

Table A.8 Average `AvDi` and `MiDi` of `MAR` in grid networks

`AvDi`	K								
$G_{p,q}$	3	4	5	6	7	8	9	10	
$G_{3,12}$	0.949	0.821	0.769	0.708	0.670	0.654	0.607	0.610	
$G_{4,36}$	0.982	0.982	0.892	0.814	0.759	0.690	0.643	0.652	
$G_{6,6}$	0.933	0.933	0.900	0.893	0.867	0.857	0.833	0.822	
$G_{12,12}$	0.970	0.970	0.955	0.952	0.939	0.935	0.924	0.919	
Average	0.959	0.926	0.879	0.842	0.809	0.784	0.752	0.751	0.839
`MiDi`	K								
$G_{p,q}$	3	4	5	6	7	8	9	10	
$G_{3,12}$	0.923	0.538	0.231	0.385	0.231	0.308	0.308	0.231	
$G_{4,36}$	0.947	0.974	0.237	0.132	0.158	0.132	0.158	0.132	
$G_{6,6}$	0.800	0.800	0.600	0.600	0.600	0.500	0.200	0.200	
$G_{12,12}$	0.955	0.909	0.864	0.864	0.864	0.818	0.727	0.727	
Average	0.906	0.805	0.533	0.495	0.463	0.439	0.348	0.322	0.539

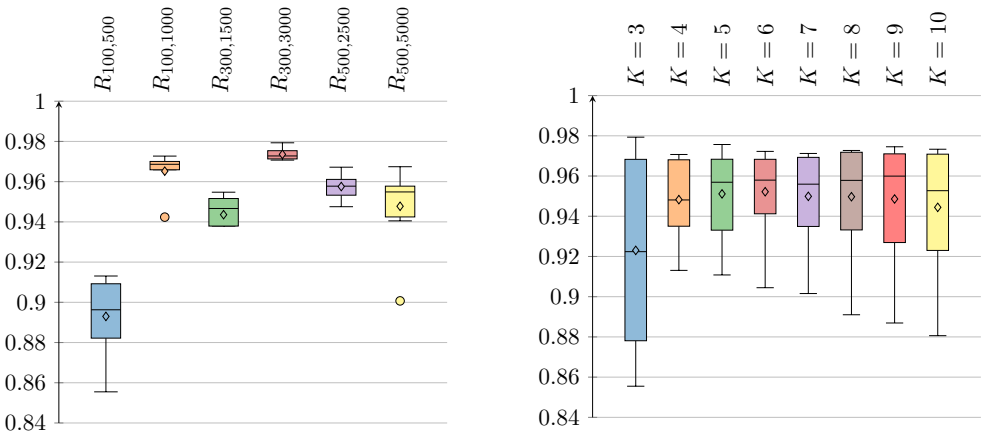


Fig. A.1 `AvDi` dispersion of `MAO` in random networks

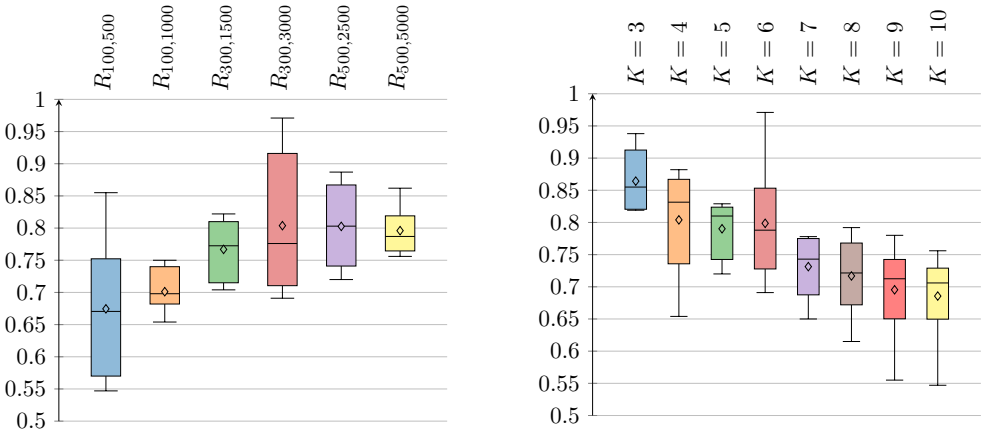


Fig. A.2 `MiDi` dispersion of `MAO` in random networks

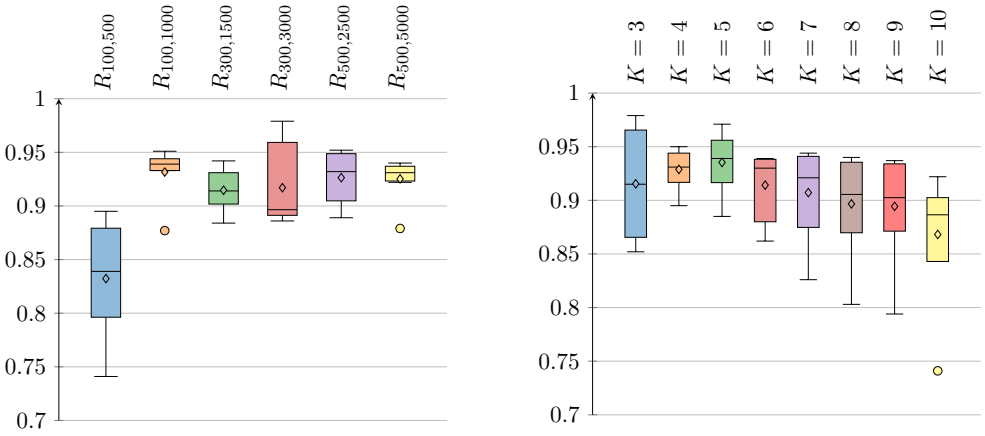


Fig. A.3 `AvDi` dispersion of `MRA` in random networks

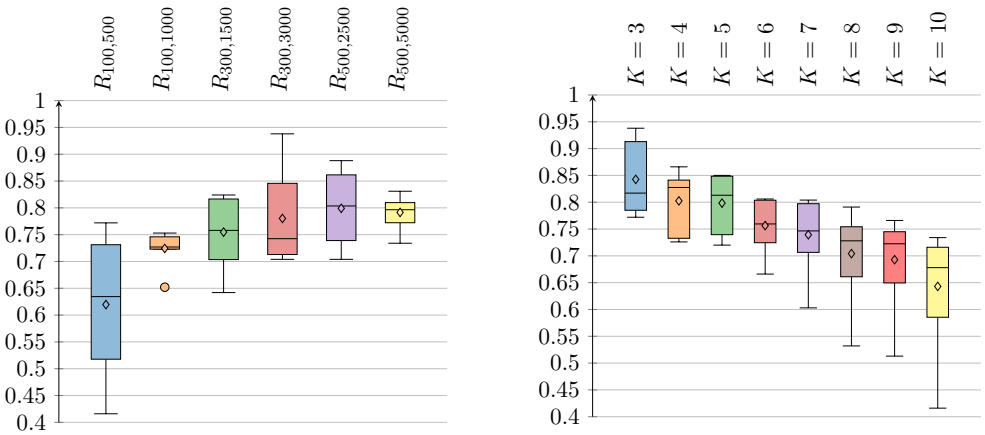


Fig. A.4 `MiDi` dispersion of `MRA` in random networks

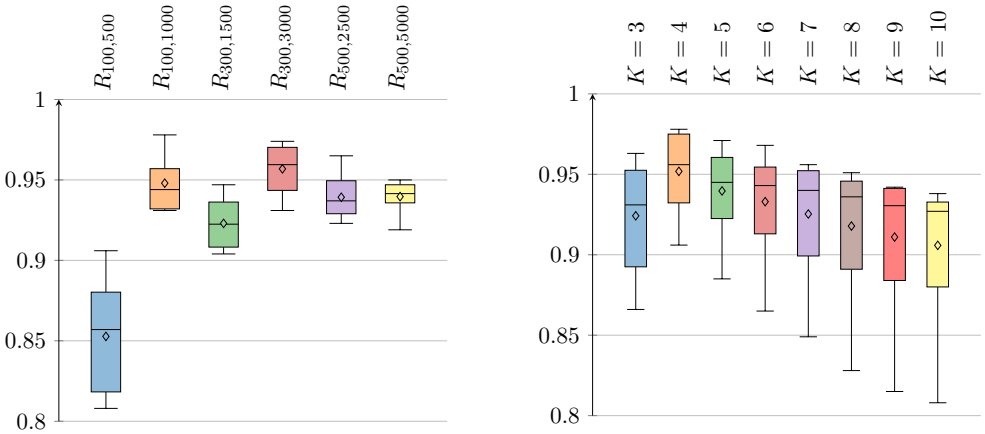


Fig. A.5 `AvDi` dispersion of `MRO` in random networks

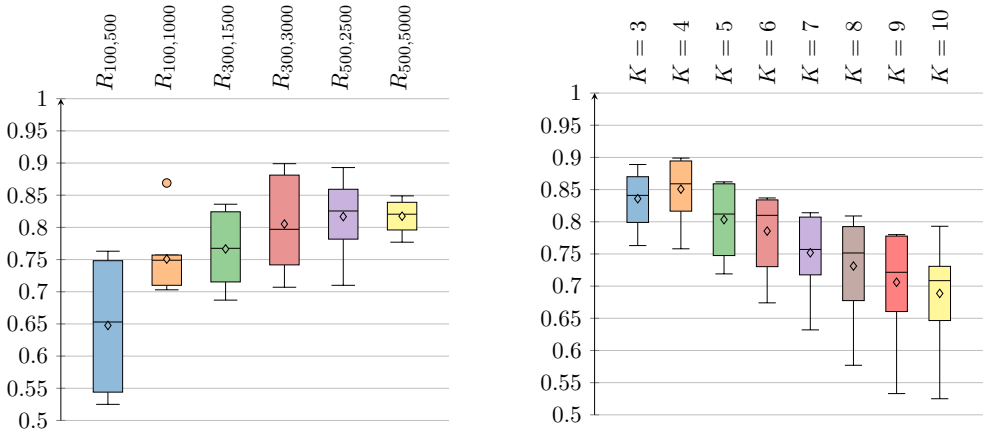


Fig. A.6 `MiDi` dispersion of `MRO` in random networks

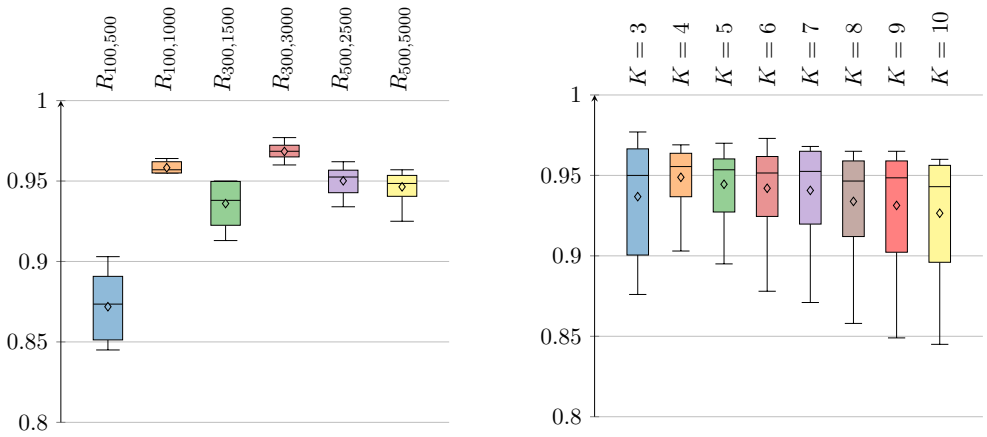


Fig. A.7 `AvDi` dispersion of `MAR` in random networks

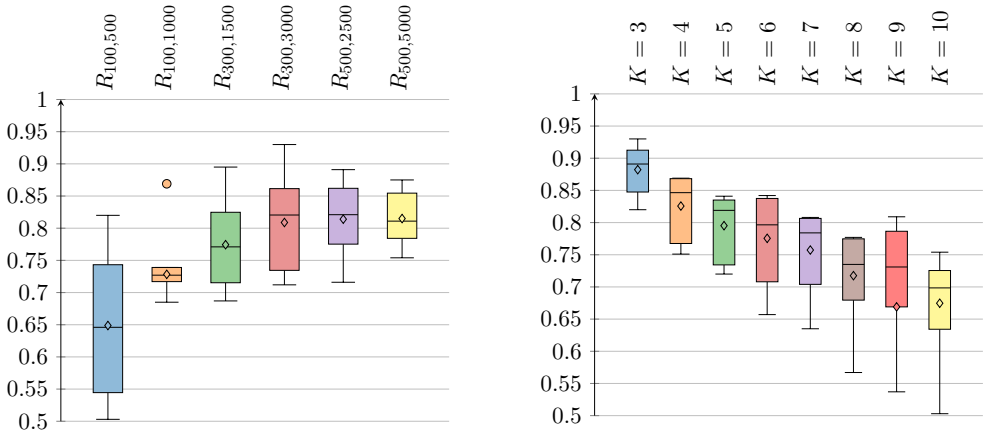


Fig. A.8 `MiDi` dispersion of `MAR` in random networks

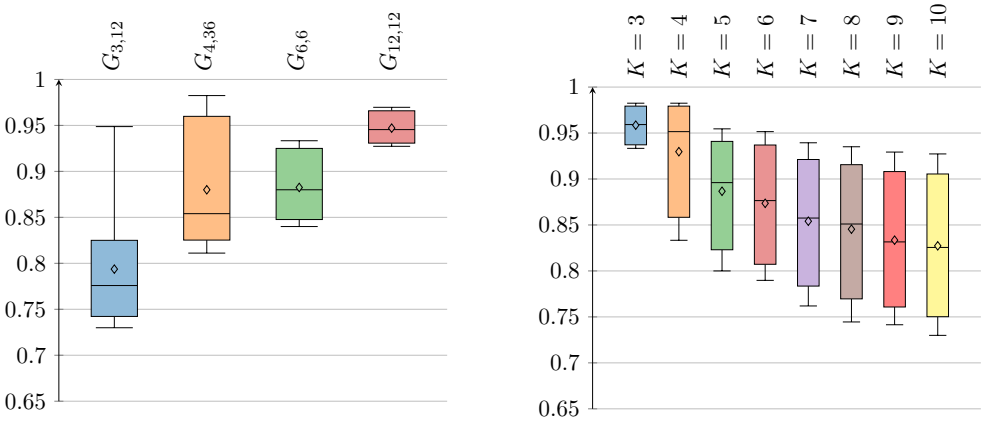


Fig. A.9 `AvDi` dispersion of `MAO` in grid networks

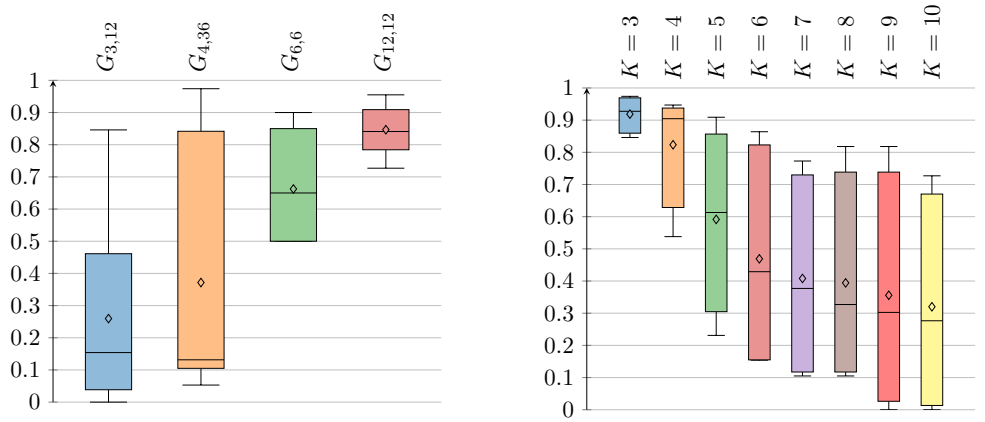


Fig. A.10 `MiDi` dispersion of `MAO` in grid networks

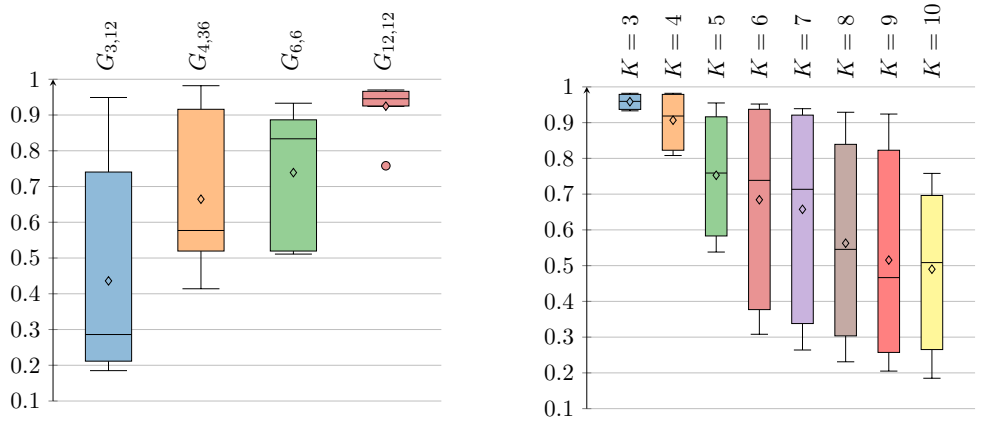


Fig. A.11 `AvDi` dispersion of `MRA` in grid networks

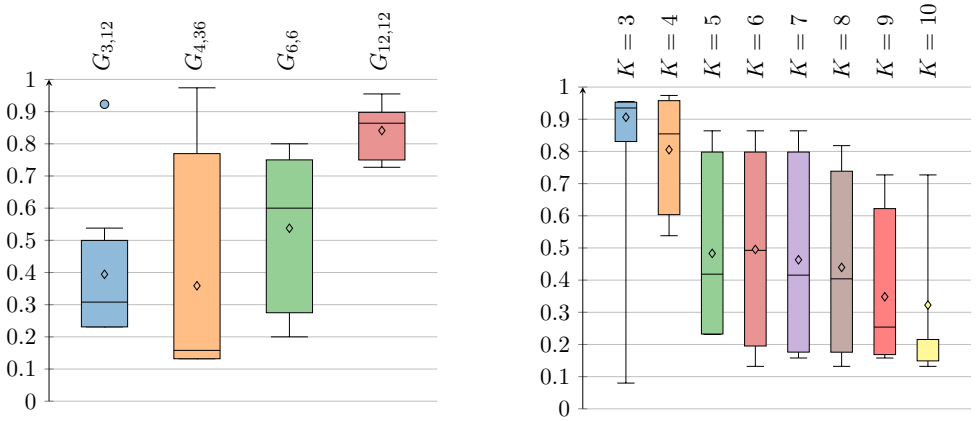


Fig. A.12 `MiDi` dispersion of `MRA` in grid networks

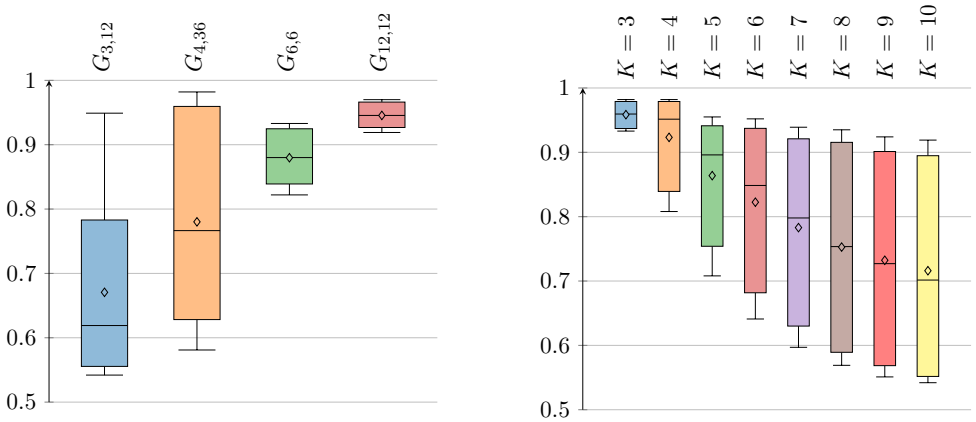


Fig. A.13 `AvDi` dispersion of `MRO` in grid networks

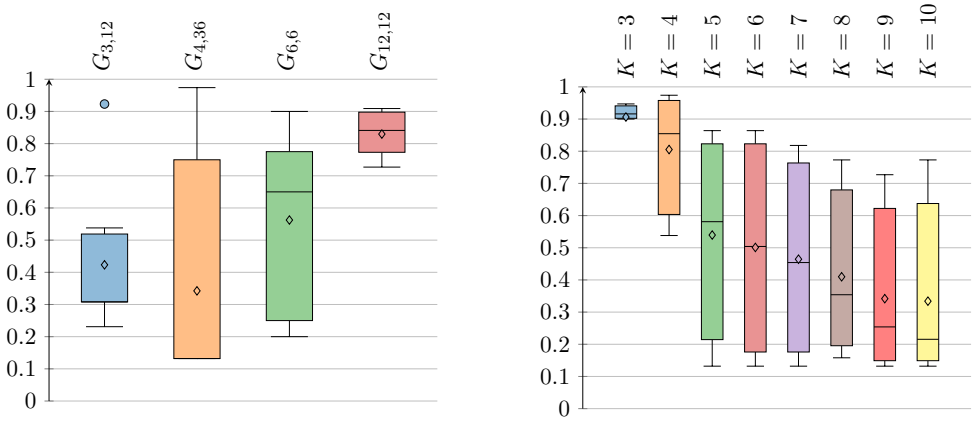


Fig. A.14 `MiDi` dispersion of `MRO` in grid networks

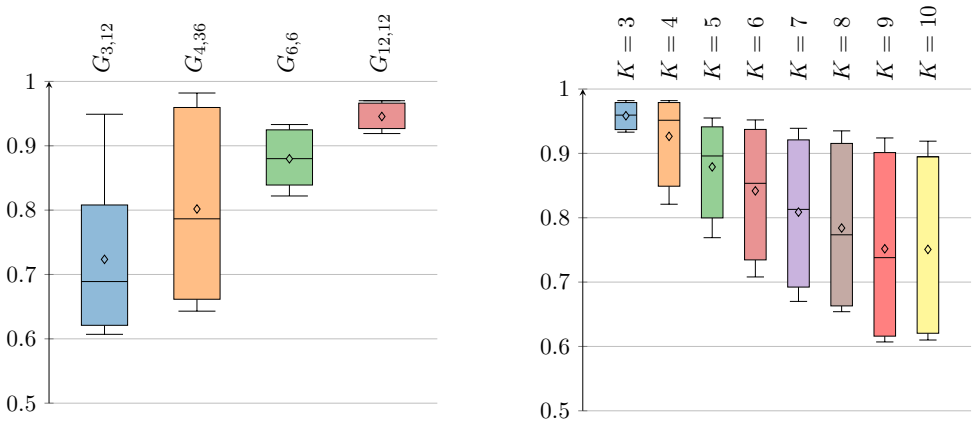


Fig. A.15 `AvDi` dispersion of `MAR` in grid networks

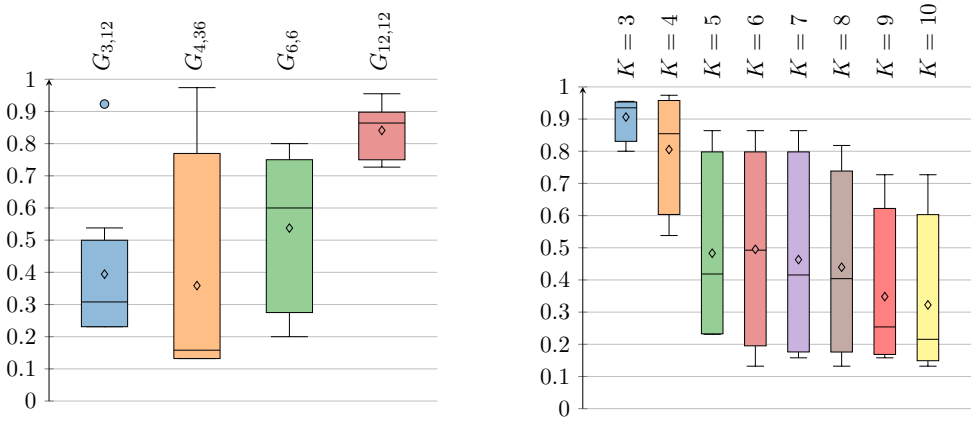


Fig. A.16 `MiDi` dispersion of `MAR` in grid networks

A.2 Dissimilarities and run times for the formulations including additional constraints

A.2.1 Individual results

Table A.9 Average `AvDi` and `MiDi` of `MRAA` in random networks

`AvDi`	K								
$R_{n,m}$	3	4	5	6	7	8	9	10	
$R_{100,500}$	0.864	0.910	0.907	0.893	0.873	0.859	0.851	0.828	
$R_{100,1000}$	–	0.951	0.968	0.967	0.968	0.968	0.968	0.966	
$R_{300,1500}$	0.923	0.943	0.952	0.947	0.941	0.943	0.934	0.932	
$R_{300,3000}$	0.979	0.973	0.977	0.972	0.968	0.973	0.974	0.971	
$R_{500,2500}$	0.952	0.961	0.955	0.959	0.949	0.948	0.948	0.942	
$R_{500,5000}$	0.896	0.944	0.939	0.952	0.954	0.954	0.960	0.958	
Average	0.923	0.947	0.949	0.948	0.942	0.941	0.939	0.933	0.941
`MiDi`	K								
$R_{n,m}$	3	4	5	6	7	8	9	10	
$R_{100,500}$	0.774	0.742	0.723	0.669	0.625	0.582	0.515	0.498	
$R_{100,1000}$	–	0.704	0.754	0.743	0.720	0.680	0.675	0.670	
$R_{300,1500}$	0.811	0.790	0.799	0.766	0.747	0.730	0.702	0.701	
$R_{300,3000}$	0.938	0.855	0.837	0.738	0.719	0.709	0.728	0.707	
$R_{500,2500}$	0.888	0.860	0.819	0.790	0.769	0.743	0.739	0.680	
$R_{500,5000}$	0.798	0.841	0.810	0.802	0.803	0.766	0.771	0.761	
Average	0.842	0.799	0.790	0.752	0.731	0.702	0.688	0.670	0.745

Table A.10 Average AvDi and MiDi of MARA in random networks

AvDi	K								
$R_{n,m}$	3	4	5	6	7	8	9	10	
$R_{100,500}$	0.867	0.911	0.904	0.900	0.885	0.873	0.864	0.862	
$R_{100,1000}$	–	0.970	0.962	0.962	0.969	0.969	0.969	0.965	
$R_{300,1500}$	0.929	0.951	0.949	0.951	0.944	0.941	0.935	0.933	
$R_{300,3000}$	0.966	0.977	0.976	0.979	0.976	0.976	0.972	0.973	
$R_{500,2500}$	0.960	0.967	0.961	0.963	0.955	0.955	0.950	0.947	
$R_{500,5000}$	0.922	0.948	0.948	0.960	0.957	0.958	0.966	0.961	
Average	0.929	0.954	0.950	0.952	0.948	0.945	0.943	0.940	0.945

MiDi	K								
$R_{n,m}$	3	4	5	6	7	8	9	10	
$R_{100,500}$	0.794	0.752	0.718	0.671	0.636	0.582	0.540	0.497	
$R_{100,1000}$	–	0.819	0.731	0.719	0.740	0.706	0.710	0.671	
$R_{300,1500}$	0.832	0.819	0.793	0.770	0.763	0.721	0.714	0.697	
$R_{300,3000}$	0.899	0.883	0.833	0.822	0.785	0.752	0.707	0.705	
$R_{500,2500}$	0.905	0.867	0.845	0.831	0.793	0.759	0.756	0.723	
$R_{500,5000}$	0.867	0.848	0.803	0.836	0.802	0.764	0.796	0.755	
Average	0.859	0.831	0.787	0.775	0.753	0.714	0.714	0.675	0.760

Table A.11 Run times of MRAA for random networks (seconds)

	K								
$R_{n,m}$	3	4	5	6	7	8	9	10	
$R_{100,500}$	0.068	0.095	0.135	0.216	0.301	0.564	0.815	1.048	
$R_{100,1000}$	–	0.177	0.218	0.261	0.353	0.499	0.594	0.601	
$R_{300,1500}$	0.204	0.285	0.372	0.451	0.637	0.741	1.028	1.702	
$R_{300,3000}$	0.414	0.561	0.682	0.805	1.039	1.241	1.455	6.490	
$R_{500,2500}$	0.382	0.480	0.661	0.787	1.131	1.468	1.625	2.109	
$R_{500,5000}$	0.567	0.755	1.171	1.432	1.924	2.283	5.247	19.440	
Average	0.327	0.329	0.540	0.659	0.898	1.133	1.794	5.320	1.394

Table A.12 Run times of MARA for random networks (seconds)

	K								
$R_{n,m}$	3	4	5	6	7	8	9	10	
$R_{100,500}$	0.073	0.112	0.156	0.206	0.313	0.422	0.508	0.639	
$R_{100,1000}$	–	0.178	0.245	0.267	0.341	0.395	0.466	0.588	
$R_{300,1500}$	0.220	0.309	0.370	0.466	0.599	0.710	0.806	1.021	
$R_{300,3000}$	0.398	0.517	0.671	0.787	1.075	1.496	1.567	1.814	
$R_{500,2500}$	0.398	0.517	0.671	0.787	1.075	1.496	1.567	1.814	
$R_{500,5000}$	0.588	0.891	1.276	1.540	2.055	2.447	5.342	19.563	
Average	0.333	0.429	0.566	0.681	0.906	1.119	1.693	5.014	1.364

Table A.13 Average `AvDi` and `MiDi` of `MRAA` in grid networks

`AvDi`	K								
$G_{p,q}$	3	4	5	6	7	8	9	10	
$G_{3,12}$	0.949	0.833	0.738	0.703	0.714	0.681	0.656	0.641	
$G_{4,36}$	0.982	0.982	0.892	0.804	0.732	0.759	0.702	0.718	
$G_{6,6}$	0.933	0.933	0.900	0.893	0.867	0.857	0.833	0.822	
$G_{12,12}$	0.970	0.970	0.955	0.952	0.939	0.935	0.924	0.919	
Average	0.959	0.930	0.871	0.838	0.813	0.808	0.779	0.775	0.847
`MiDi`	K								
$G_{p,q}$	3	4	5	6	7	8	9	10	
$G_{3,12}$	0.923	0.538	0.231	0.231	0.154	0.154	0.000	0.000	
$G_{4,36}$	0.974	0.947	0.237	0.053	0.105	0.105	0.079	0.105	
$G_{6,6}$	0.900	0.800	0.800	0.700	0.500	0.500	0.300	0.200	
$G_{12,12}$	0.955	0.909	0.818	0.818	0.727	0.727	0.727	0.682	
Average	0.938	0.805	0.521	0.450	0.372	0.372	0.277	0.247	0.498

Table A.14 Average `AvDi` and `MiDi` of `MARA` in grid networks

`AvDi`	K								
$G_{p,q}$	3	4	5	6	7	8	9	10	
$G_{3,12}$	0.949	0.833	0.777	0.785	0.711	0.720	0.686	0.699	
$G_{4,36}$	0.982	0.982	0.982	0.812	0.747	0.787	0.767	0.748	
$G_{6,6}$	0.933	0.933	0.900	0.893	0.867	0.857	0.833	0.822	
$G_{12,12}$	0.970	0.970	0.955	0.952	0.939	0.935	0.924	0.919	
Average	0.959	0.930	0.881	0.860	0.816	0.825	0.803	0.797	0.859
`MiDi`	K								
$G_{p,q}$	3	4	5	6	7	8	9	10	
$G_{3,12}$	0.846	0.308	0.308	0.154	0.231	0.154	0.154	0.154	
$G_{4,36}$	0.974	0.947	0.211	0.184	0.079	0.184	0.105	0.132	
$G_{6,6}$	0.800	0.800	0.800	0.700	0.700	0.500	0.200	0.200	
$G_{12,12}$	0.909	0.955	0.909	0.818	0.727	0.818	0.727	0.773	
Average	0.876	0.752	0.557	0.464	0.434	0.414	0.297	0.315	0.514

Table A.15 Run times of `MRAA` for grid networks (seconds)

	K								
$G_{p,q}$	3	4	5	6	7	8	9	10	
$G_{3,12}$	0.047	0.024	0.137	0.253	0.272	0.514	0.842	0.905	
$G_{4,36}$	0.078	0.117	0.197	19.864	42.651	3.999	4.611	300	
$G_{6,6}$	0.022	0.012	0.103	0.097	0.370	0.493	2.487	1.093	
$G_{12,12}$	0.073	0.082	0.456	0.227	1.484	1.802	5.142	5.582	
Average	0.055	0.059	0.667	5.111	11.194	1.702	3.270	76.915	12.371

Table A.16 Run times of `MARA` for grid networks (seconds)

	K								
$G_{p,q}$	3	4	5	6	7	8	9	10	
$G_{3,12}$	0.016	0.022	0.022	0.027	0.029	0.036	0.039	0.092	
$G_{4,36}$	0.074	0.112	0.153	0.188	0.231	0.276	0.357	0.377	
$G_{6,6}$	0.017	0.016	0.024	0.025	0.116	0.030	0.083	0.036	
$G_{12,12}$	0.063	0.065	0.204	0.130	0.558	0.277	0.561	1.111	
Average	0.043	0.054	0.101	0.093	0.233	0.155	0.260	0.404	1.364

A.2.2 Comparative results

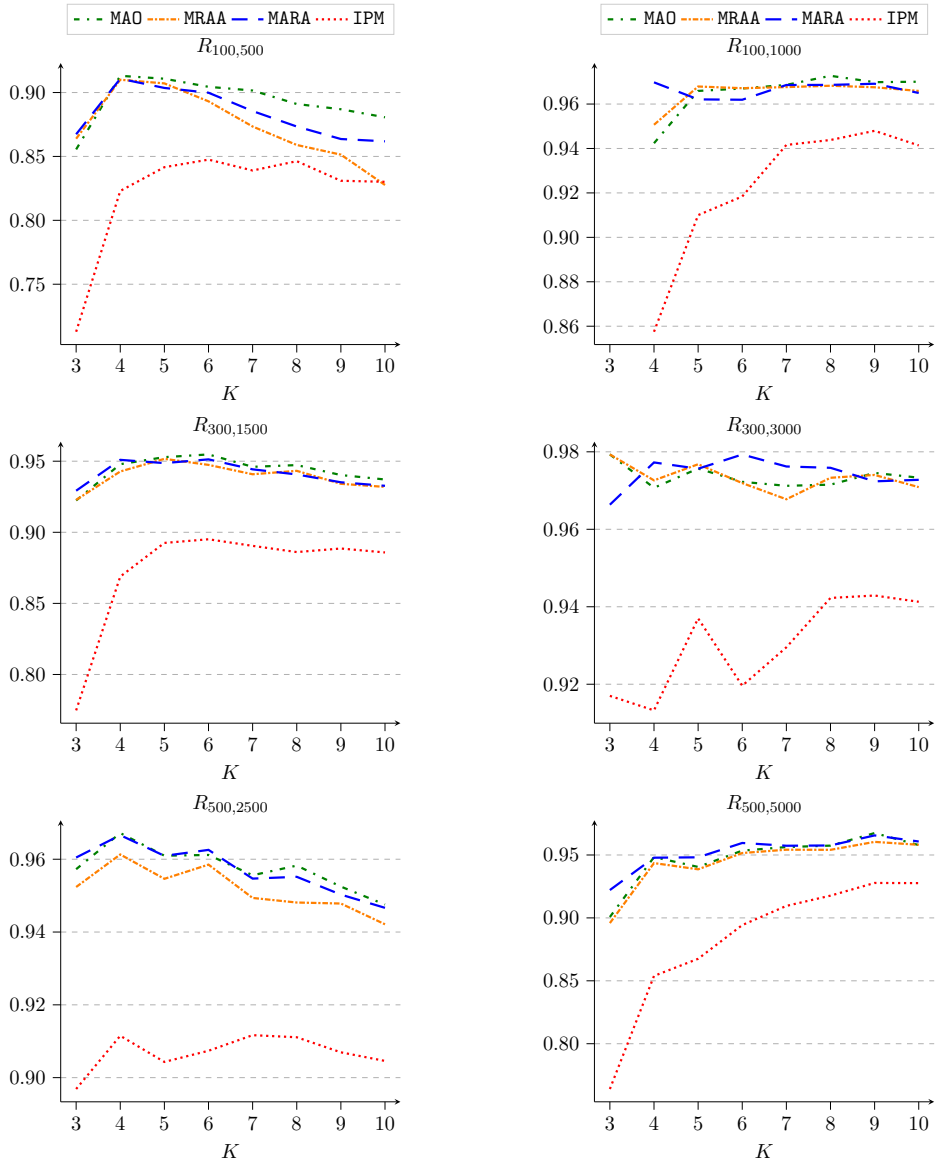


Fig. A.17 Average dissimilarity in random networks

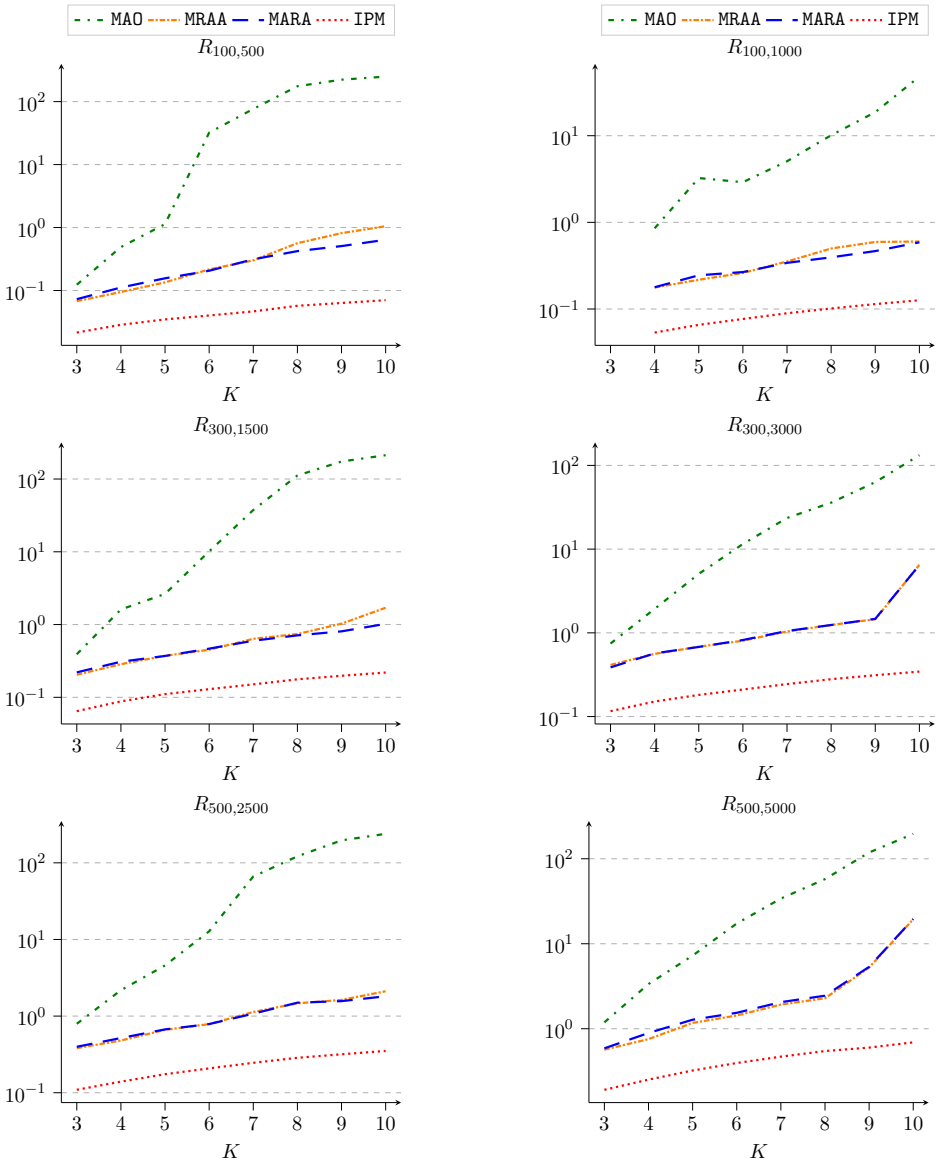


Fig. A.18 Run times in random networks (seconds – log scale)

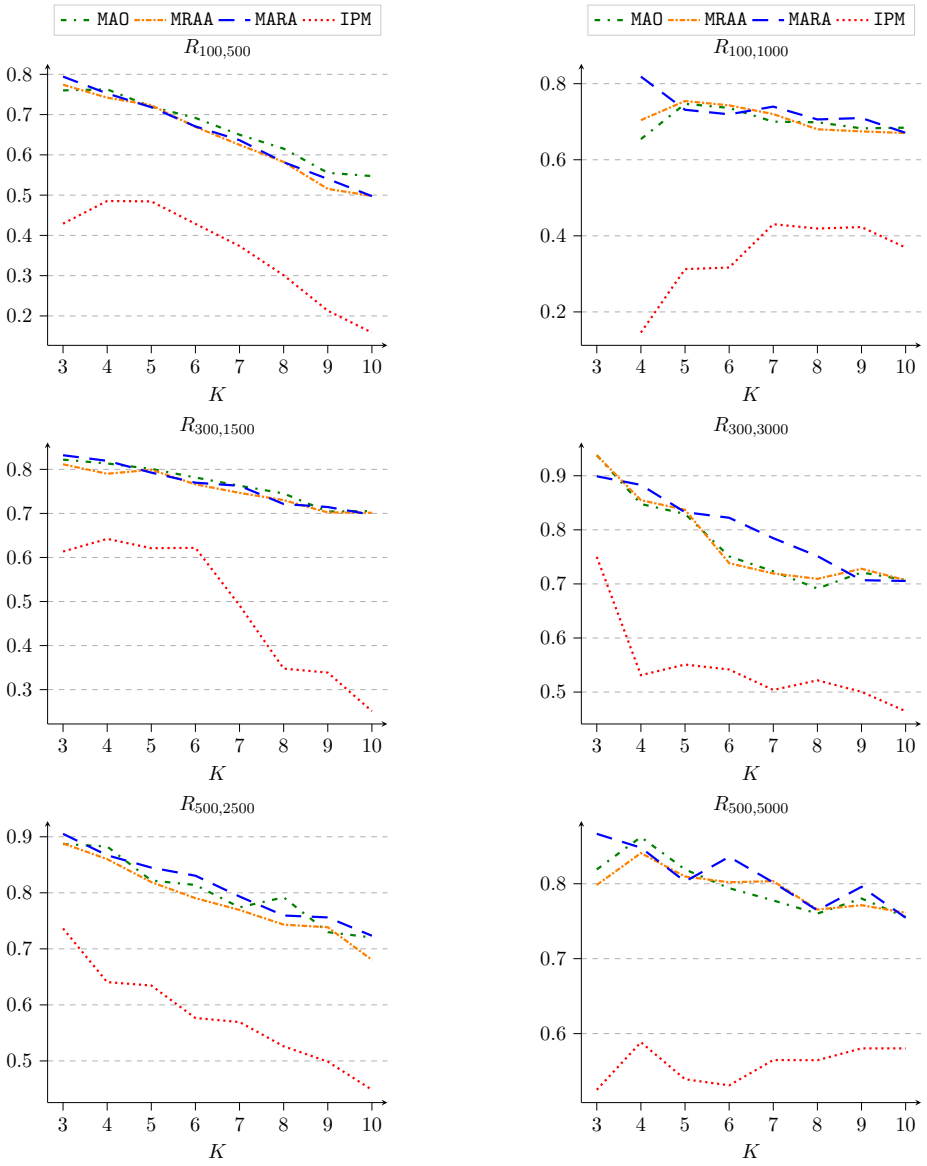


Fig. A.19 Minimum dissimilarity in random networks

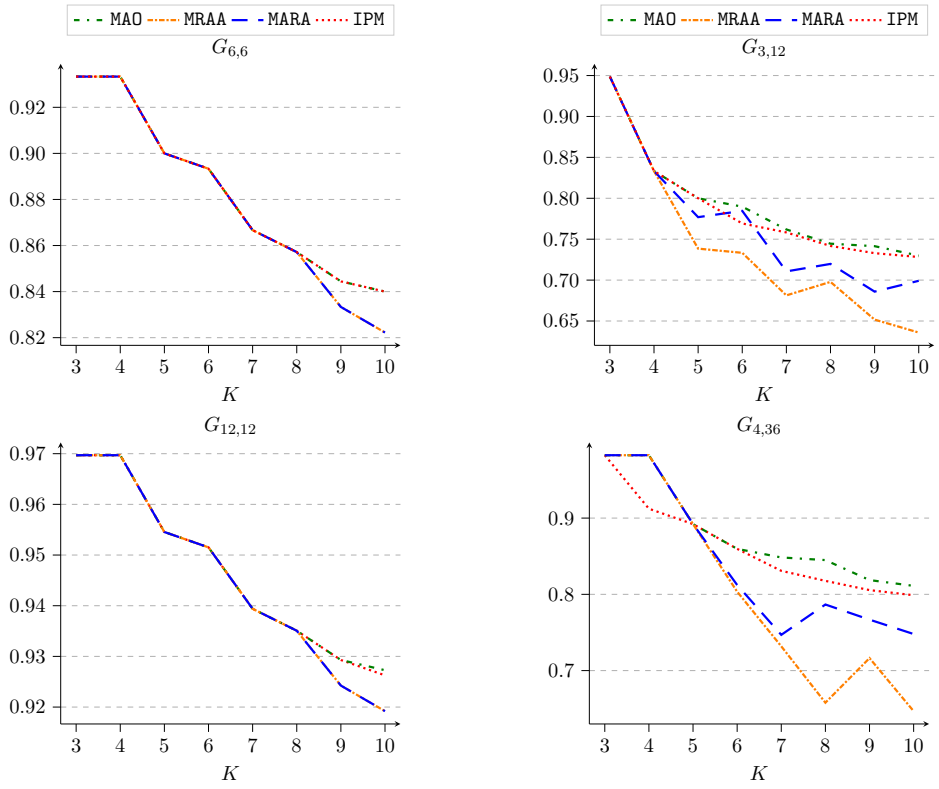


Fig. A.20 Average dissimilarity in grid networks

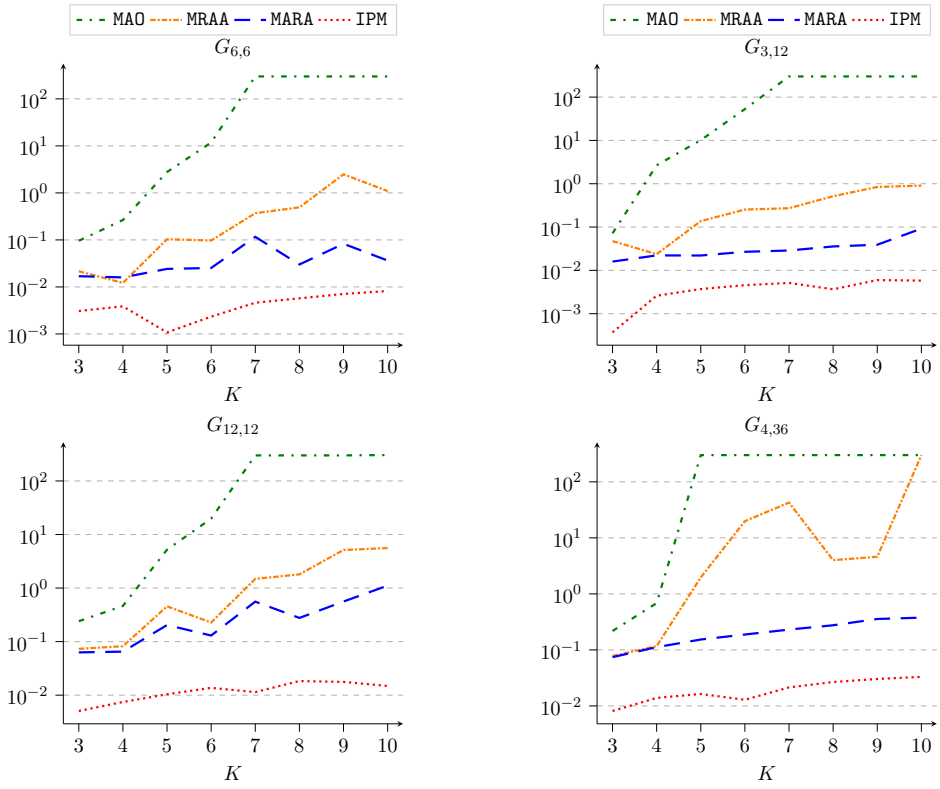


Fig. A.21 Run times in grid networks (seconds – log scale)

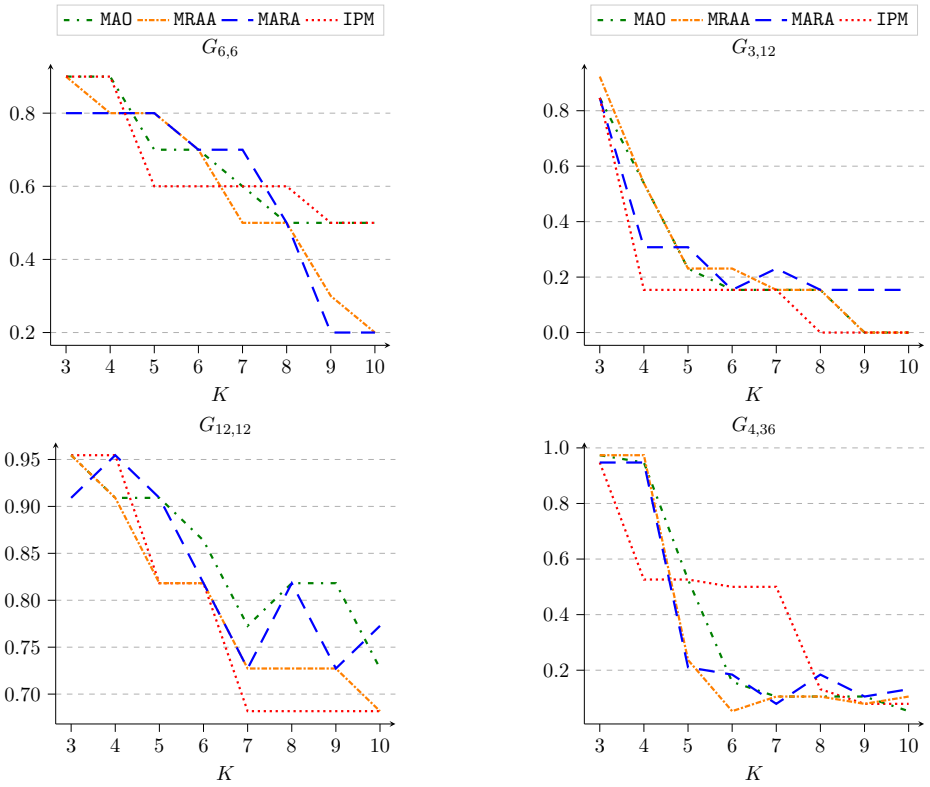


Fig. A.22 Minimum dissimilarity in grid networks