Finding K shortest and dissimilar paths

Ali Moghanni* Marta Pascoal†‡ Maria Teresa Godinho§

September 2021

Abstract

The purpose of the K dissimilar paths problem is to find a set of K paths, between the same pair of nodes, which share few arcs. The problem has been addressed from an application point of view and integer programming (IP) formulations have also been introduced recently. In the present work it is assumed that each arc is assigned with a cost, and the goal is then to find K dissimilar paths while simultaneously minimizing the total cost. Some of the previous formulations: one minimizing the number of repeated arcs, another one minimizing the number of arc repetitions, as well as modifications that bound the number of paths in which the arcs appear, are extended with a cost function. Properties of the resulting bi-objective problems are studied and the ϵ-constraint method is adapted to solve them using a decreasing and an increasing strategy for updating ϵ. These methods are tested for finding sets of 10 paths in random and grid instances to assess the efficiency of the ϵ-constraint methods and the performance of the formulations to calculate shortest and dissimilar paths. Results show that minimizing the number of arc repetitions produces efficient solutions with higher dissimilarities faster than minimizing the number of repeated arcs. The cost range of the solutions is similar in both approaches. Additionally, bounding the number of paths in which each arc appears improves the quality of the solutions as to the dissimilarity while worsening its cost.

Keywords: K alternative paths; Cost; Dissimilarity; IP formulations; Bi-objective optimization

1 Introduction

The present paper addresses the determination of sets of K paths between two nodes in a network, with two goals: the minimization of the total cost of the K paths; and the maximization

*University of Coimbra, CMUC, Department of Mathematics, 3001-501 Coimbra, Portugal, (ali.moghanni@mat.uc.pt)

†Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo da Vinci, 32, Milan, 20133, Italy, University of Coimbra, CMUC, Department of Mathematics, 3001-501 Coimbra, Institute for Systems Engineering and Computers – Coimbra, rua Sílvio Lima, Pólo II, 3030-290 Coimbra, Portugal (marta.brazpascoal@polimi.it)

‡Corresponding author

§Department of Mathematics and Physical Sciences, Polytechnic Institute of Beja, Campus do Instituto Politécnico de Beja, rua Pedro Soares, 7800-295 Beja, Portugal, and CMAFcIO, Faculdade de Ciências da Universidade de Lisboa, Campo Grande, 1749-016 Lisboa (mtgodinho@ipbeja.pt)

of their dissimilarity. The first goal translates into one of the most well studied network optimization problems, the shortest path problem. This problem arises in many contexts, whenever a path is needed which minimizes the sum of the values associated with its arcs, and several polynomial algorithms are known to solve it (Ahuja et al., 1993). One related problem has also interested researchers: the bi-objective shortest path problem, dedicated to the case where the objective functions are linear. A survey about this problem was presented by Raith and Ehrgott (2009). Bi-objective path problems involving different types of objective functions have also been studied. These works have been reviewed in the more general survey by Clímaco and Pascoal (2012). The problem addressed in the present work looks for sets of K paths, so the previous studies do not apply to the linear integer approach considered in the following. The literature on bi-objective integer programming, on the other hand, is rich for general problems and also when considering particular problems (Ehrgott and Gandibleux, 2000; Ulungu and Teghem, 1994).

The situation is different for the second goal. The term dissimilarity is often found in the literature intending to measure the diversity between two entities, and, in practical terms, it is useful in a number of situations, whenever we look for alternative solutions. Nevertheless, we are not aware of the existence of a universal definition for it. On the contrary, it is usual to find different specifications of this concept, according to the context in which it arises, although, in general, when talking about paths, the dissimilarity is measured according to the amount of network resources that are shared. For instance, Constantino et al. (2015) study an arc routing problem with an additional constraint on the number of shared nodes, whereas Hughes et al. (2021) study a problem where path conflict is found whenever an arc or a node is used more than once. In the present work, we focus on arc dissimilarity.

Perhaps, the most straightforward application of the arc dissimilarity problem is found in navigation systems and map-based services. These systems provide the fastest path from a source location s to a target location t, as well as a set of alternative paths, giving the user more options to choose from. Recently, Li et al. (2021) presented a study where the quality of the alternative paths produced by several different methods within this framework is evaluated by the end users.

In more specific applications, the alternative paths are designed to fit their intended purpose. For instance, in hazardous materials routing, where spatially dissimilar paths are sought to reduce the impact of an accident over the traversed regions, Akgün et al. (2000) suggest that when computing the similarity between two paths, the area of the intersection of a buffer zone defined for each path should be considered. Differently, Dadkar et al. (2008) propose a system to derive a set of routes so as to strike a balance between the quality of the worst path (regarding travel time, accident probability and population exposure) and the largest amount of overlap between any two paths. Dell'Olmo et al. (2005)'s approach combines these two ideas. They start with a set of Pareto-optimal paths found by a multi-objective shortest path algorithm. Then, for each path previously found, a buffer zone approximating the impact area of a material being released after an accident is constructed and a dissimilarity index for every pair of paths is derived to find the most spatially different routes. Both Erkut et al. (2007) and Batta and Kwon (2013) present surveys on this type of problems. More recently, Jabbarzadeh et al. (2020) extend these studies to the impact of disruption on hazardous materials shipment, in the case of railway transport.

Another class of applications for this problem is related with the unpredictability of the routes, in order to prevent robberies, when collecting or transporting cash, or to ensure more efficient street patrols. In this case, besides spatial dissimilarity, time dissimilarity is also rele-

vant. Calvo and Cordone (2003), address a real-world case, submitted by a security company operating in Milan, Italy. It is required that space and time are covered and the response to alerts is secured, using as few guards as possible and so that each guard desirably performs a different route every night. The authors propose both an integer linear programming (ILP) model and a heuristic to approach this problem. Constantino et al. (2017) study a dissimilar arc routing problem that arises in a Portuguese company. The firm needs to collect safes from parking meters located along the streets, minimizing the total collecting time and avoiding similar tours. In this case, the similarity between two tours is assessed on the basis of the number of tasks that are visited by both in the same time periods of the day. The authors address this problem through both an ILP model and a heuristic where the total collecting time is minimized and constraints are used to prevent the selection of similar tours. Talarico et al. (2015) study a k-dissimilar vehicle routing problem, where the aim is to generate a set of k alternative solutions of a single vehicle routing problem instance, in such a way that each alternative solution differs from all the others by at least a given threshold. Again, an ILP model and a heuristic are the approaches used to deal with the problem. Tikani et al. (2021) extend such applications given them a time-dependent perspective, while still taking the risk of robberies and the effects of traffic congestion into account. They propose an evolutionary algorithm for finding solutions for the problem, which includes a caching mechanism and fuzzy logic approach to dynamically adjust the rates of operators during the searching process.

In some reliability and survivability problems in telecommunications, it is common to require pairs of paths linking two nodes, which minimize a cost function and simultaneously can work as alternatives in case a failure occurs along the first one (Hu, 2003; Gomes et al., 2020). Such paths are known as the primary and the backup paths. Ideally the two paths would be disjoint, however it is not always possible to meet this requirement, therefore, a more flexible approach imposes only that the overlap between them does not exceed a certain value. Another telecommunications problem that also seeks for the minimization of shared resources consists in minimizing the number of shared risk link groups (SRLG) in pairs of paths. In this case each arc is associated with a given set of risk groups and the goal is to find alternative paths that have as few groups in common as possible (Gomes et al., 2016; Pascoal and Clímaco, 2020). In either case, the simultaneous minimization of the path cost is also involved in the problem. Related problems were surveyed in the recent work (Clímaco and Craveirinha, 2019).

Finally, Chang et al. (2020) study the k-discriminative paths problem, defined as the problem of finding k paths between two given nodes in a network, such that the total path overlapping and the total path length is minimized, while imposing an upper-bound on maximal path length and on the maximal length of overlapping segments. This problem is discussed having in mind several applications including queries for emergency-purpose applications, queries for pre-schedule transportation plan, and queries with multiple sources and destinations for regional evacuation. In the work a heuristic strategy based on the ant colony optimization is presented and tested.

It is worth noting that all the above mentioned works deal with arc dissimilarity in the context of different and specific multi-objective problems, where often the presence of other type of constraints has a significant impact on the solutions to be obtained. In a recent study, Moghanni et al. (2020) addressed the single-objective arc dissimilarity problem – several different integer linear programming models where proposed and compared in terms of the solutions' dissimilarity and of the run time. Based on this study, good ILP approaches to the single objective problem were identified. In this paper, we analyse if that study's findings hold in the bi-objective context. The conclusions to be drawn from the present work will be useful for the

applications community for two reasons: the proposed models can be used in various contexts by performing adequate adjustments; and, furthermore, they allow a better understanding on the trade-off between the two objective functions in different types of networks, avoiding the distorting effects caused by the presence of other constraints, involved in the above mentioned studies.

This is done by revisiting four of the models presented: one which intends to minimize the number of arcs repeated in the paths; another one which intends to minimize the number of arc repetitions; as well as variants of these two resulting from imposing an upper bound on the number of presences of each arc in the solution. These models are extended by adding a linear objective function that depends on the arc costs, resulting in a bi-objective problem. Two ϵ-constraint algorithms are described for solving the bi-objective problems, which differ on the strategy used for updating the parameter ϵ. The performance of the formulations in the bi-objective context is assessed by means of a set of empirical tests.

The rest of this text is organized into five parts. Section 2 is dedicated to review general concepts of bi-objective optimization and of the ϵ-constraint method. In Section 3, notation and the K dissimilar paths problem are introduced. Recent formulations for the K dissimilar paths problem are also reviewed. In Section 4 the K shortest and dissimilar paths problem is defined. Moreover, the ϵ-constraint methods described in Section 2 are adapted, based on the formulations for the K shortest and dissimilar paths problem. Finally, the methods developed are tested and computational results of these experiments are presented in the next section. Concluding remarks are provided in Section 6.

2 Bi-objective optimization

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

$$\text{minimize } \{f(x) = (f_1(x), f_2(x)) : \ x \in X\} \tag{1}$$

where $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$ are two objective functions and $X \subseteq \mathbb{R}^n$ is a 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.

A feasible solution of (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 $i \in \{1, 2\}$.

The solution is called efficient or Pareto optimal if there is no other feasible solution, $x \in X$, which dominates $\hat{x}$. If $\hat{x}$ is efficient, then its outcome vector $f(\hat{x})$ is called a non-dominated point. Some dominated solutions are dominated for only one of the objective functions. The feasible solution $\hat{x} \in X$ 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. The set of all efficient solutions is denoted by X_E and called the efficient set. The set of all non-dominated points $\hat{y} = f(\hat{x}) \in Y$, $\hat{x} \in X_E$, is denoted by Y_N and called the non-dominated set.

The ideal and the nadir points of (1) are lower and upper bounds on the set of non-dominated points, respectively. They give an indication of the range of the values that the non-dominated points can attain (Ehrgott, 2005). The point $y^I = (y_1^I, y_2^I)$, where $y_k^I = \min_{x \in X}\{f_k(x)\}$, for $k = 1, 2$, is called the ideal point of the BOP (1). Moreover, the point $y^N = (y_1^N, y_2^N)$ where $y_k^N = \max_{x \in X_E}\{f_k(x)\}$, $k = 1, 2$, is called the nadir point of the BOP (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. Moreover, the ideal and the nadir points can be obtained 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), thus avoiding the presence of weakly efficient solutions.

Traditional approaches to the bi-objective optimization problems are based on scalarization. This involves a single objective optimization problem related to the BOP (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 constraints 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 (Cohon, 1978), which solves the weighted-sum problem $\min_{x \in X} \lambda_1 f_1(x) + \lambda_2 f_2(x)$, 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 solutions within the convex hull formed by the extreme non-dominated points (Ehrgott, 2005).

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, all efficient solutions can be generated. The method was first introduced by Haimes et al. (1971), and extensive discussions about the topic can be found in Chankong and Haimes (1983) or Mavrotas (2009). 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 (1) is replaced by the ϵ-constraint problem

$$\text{minimize}\,\{f_1(x) :\ x \in X \wedge f_2(x) \leq \epsilon\} \tag{2}$$

where $\epsilon \in \mathbb{R}$. Furthermore, updating ϵ as $\hat{f}_2 - \Delta$, where $\hat{f}_2$ is the value of a feasible solution with regard to the second objective 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 the weakly efficiency can be ensured.

It can be shown that with appropriate choices of ϵ all non-dominated solutions can be found.

These ϵ values are equal to the actual objective function values of the efficient solution one would like to find.

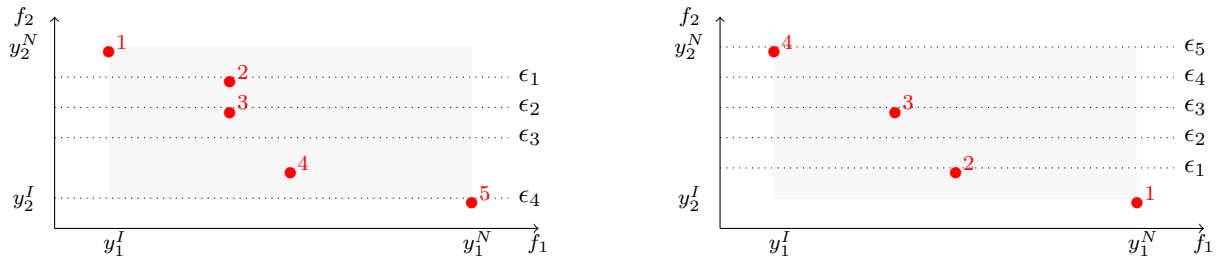
The choice of which function to optimize and which to include in the constraints, as well as the strategy adopted for updating the bound ϵ, can vary and may depend on the particular form of the problem. An outline of a generic ϵ-constraint method is given in Algorithm 1.

Algorithm 1: The ϵ-constraint method – Decreasing ϵ version

1 $(y_1^I, y_2^I) \leftarrow$ ideal point for (f_1, f_2) in X; $(y_1^N, y_2^N) \leftarrow$ nadir point for (f_1, f_2) in X
2 $Y_E \leftarrow \{(y_1^I, y_2^N)\}$
3 $\bar{x} \leftarrow (y_1^I, y_2^N)$; $\epsilon \leftarrow y_2^N - \Delta$
4 **while** $\epsilon \geq y_2^I$ **do**
5 $\quad x^* \leftarrow$ optimal solution of problem (2)
6 $\quad$ **if** $f_1(x^*) > f_1(\bar{x}_1)$ **then** $Y_E \leftarrow Y_E \cup \{(f_1(\bar{x}), f_2(\bar{x}))\}$
7 $\quad \bar{x} \leftarrow x^*$; $\epsilon \leftarrow f_2(x^*) - \Delta$
8 $Y_E \leftarrow Y_E \cup \{(f_1(\bar{x}), f_2(\bar{x}))\}$

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 1 is illustrated in Figure 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 solution represented by point 5, which is also a non-dominated point.



(a) Decreasing ϵ (Algorithm 1)

(b) Increasing ϵ (Algorithm 2)

Figure 1: The ϵ-constraint method

The ϵ-constraint algorithm has two main drawbacks (Gadegaard et al., 2018). One is the

weakly dominated solutions which can be found along the process, as shown above. In this case the method presumably may solve more ϵ-constraint 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, making it harder to solve. The first of these drawbacks can be overcome by:

- Comparing the solutions as they are computed, and filtering the dominated, as shown in Algorithm 1.
- Or transforming every sub-problem into a lexicographic problem with respect to (f_1, f_2), i.e., minimizing f_1 and considering the best value for f_2 whenever there is a tie in function f_1. Then the solution is certainly efficient, and the conditions in lines 8 and 9 of Algorithm 1 can be skipped.
- Or deleting the dominated solutions from the set of computed solutions after they have been found.

Additionally, the objective function can be perturbed in order to avoid the computation of these weakly efficient solutions (Mavrotas, 2009).

Like in Algorithm 1, 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, however, may as well be swept by increasing the parameter ϵ rather than by decreasing it. The solutions of problems (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 1. *Let x^* and x' be optimal solutions for the constrained problems*

$$\textit{minimize}\,\{f_1(x):\ x \in X \wedge f_2(x) \leq \epsilon^*\} \tag{3}$$

and

$$\textit{minimize}\,\big\{f_1(x):\ x \in X \wedge f_2(x) \leq \epsilon'\big\} \tag{4}$$

respectively, where $\epsilon^* \leq \epsilon'$. Then the following hold: i. x^* is a feasible solution of problem (4); ii. $f_1(x^*) \geq f_1(x')$.

The first point in Proposition 1 implies that x^* can be used as a feasible solution and the starting point for solving problem (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 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 2.

Like what happens with Algorithm 1, weakly efficient solutions can still be obtained with Algorithm 2. However these can be easily discarded after being compared with the latest computed solution.

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

$(y_1^I, y_2^I) \leftarrow$ ideal point for (f_1, f_2) in X; $(y_1^N, y_2^N) \leftarrow$ nadir point for (f_1, f_2) in X
$Y_E \leftarrow \{(y_1^N, y_2^I)\}$
$\bar{x} \leftarrow (y_1^N, y_2^I)$; $\epsilon \leftarrow y_2^I + \Delta$
while $\epsilon \leq y_2^N$ **do**
    $x^* \leftarrow$ optimal solution of problem (2)
    if $f_1(x^*) < f_1(\bar{x}_1)$ **then** $Y_E \leftarrow Y_E \cup \{(f_1(\bar{x}), f_2(\bar{x}))\}$
    $\bar{x} \leftarrow x^*$; $\epsilon \leftarrow \max\{\epsilon, f_2(x^*)\} + \Delta$
$Y_E \leftarrow Y_E \cup \{(f_1(\bar{x}), f_2(\bar{x}))\}$

The application of Algorithm 2 is illustrated in Figure 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.

3 The K dissimilar paths problem

Let (N, A) be a directed graph with $|N| = n$ nodes and $|A| = m$ arcs, and let s and t denote given source and terminal nodes, respectively, both in N. The goal of the K dissimilar paths problem in (N, A) is to find a set of K paths from node s to node t, such that the paths in the set are "diverse" enough. Needless to say, this notion permits a wide range of interpretations and, thus, many different dissimilarity measures have been proposed in the literature. With this regard we follow Erkut and Verter (1998), where index D is introduced for measuring the dissimilarity between two given paths, p_i and p_j, as follows:

$$D(p_i, p_j) = 1 - \frac{1}{2}\left(\frac{L(p_i \cap p_j)}{L(p_i)} + \frac{L(p_i \cap p_j)}{L(p_j)}\right) \tag{5}$$

where $L(p)$ denotes the number of arcs in the sequence p. Moghanni et al. (2020) used the same metric to evaluate four new linear integer formulations for the K dissimilar paths. In the same work, those models were also assessed as to the quality of their linear programming gaps and run times. Although two of the proposed models clearly stand out, both as to the quality of integrality gaps as with regard to the balance between run times and dissimilarities, we extended our study to two other models from the same work, in order to test their behaviour in the bi-objective context.

Next, we revisit the four models mentioned above. The formulations presented may be grouped into two pairs, based on the strategy used to model the problem:

- minimizing the number of repeated arcs (i.e., the number of arcs used more than once) in the paths;
- minimizing the number of arc repetitions (i.e., the number of copies of the repeated arcs) in the paths.

In turn, the elements of each pair differ due to the presence of an extra constraint that imposes a bound on the maximum number of occurrences of each arc in the solution. The bound used in this "capacity" constraint is obtained solving a simple auxiliary problem that aims at finding a set of K paths with the minimum maximum number of arc occurrences (see (Moghanni et al., 2020) for more details).

In the following, we use the acronyms `MRA` (from Minimizing Repeated Arcs) to designate the models associated to the first strategy, and `MAR` (from Minimizing Arc Repetitions) to designate the models associated to the second strategy. To differentiate the unconstrained and the constrained versions of the two pairs, we add an `A` to the later.

3.1 Minimization of the number of repeated arcs

Let the binary variables x_{ij}^k be 1 if the arc (i,j) lies in the k-th path from node s to node t, or 0 otherwise, for any arc $(i,j) \in A$ and $k = 1, \ldots, K$. The `MRA` model for the K dissimilar path is as follows:

$$\text{minimize} \quad v_1(x,y) = \sum_{(i,j)\in A} y_{ij} \tag{6a}$$

$$\text{subject to} \quad \sum_{j\in N:(i,j)\in A} x_{ij}^k - \sum_{j\in N:(j,i)\in A} x_{ji}^k = \begin{cases} 1 & i = s \\ 0 & i \neq s,t \\ -1 & i = t \end{cases}, \quad k = 1, \ldots, K \tag{6b}$$

$$y_{ij} \leq \sum_{k=1}^{K} x_{ij}^k, \quad (i,j) \in A \tag{6c}$$

$$(K-1)y_{ij} \geq \sum_{k=1}^{K} x_{ij}^k - 1, \quad (i,j) \in A \tag{6d}$$

$$x_{ij}^k \in \{0,1\},\ y_{ij} \in \{0,1\}, \quad (i,j) \in A, \quad k = 1, \ldots, K \tag{6e}$$

The constraints (6b) are flow conservation constraints that define a set of K paths from node s to node t. The constraints (6c) and (6d) relate the x and the y variables, in a way that y_{ij} is 1 if and only if arc $(i,j) \in A$ is used in more than one path, or 0 otherwise. Therefore, the objective function counts the number of arcs that are used in more than one of the K paths.

Using the same variables, the corresponding constrained model is obtained by adding the conditions:

$$\sum_{k=1}^{K} x_{ij}^k \leq\ R, \quad (i,j) \in A \tag{7}$$

where R is the optimal value of the auxiliary problem mentioned above. Then, the `MRAA` model is:

$$\begin{aligned} &\text{minimize} && v_1(x,y) = \sum_{(i,j)\in A} y_{ij} \\ &\text{subject to} && (6b) - (6d) \\ &&& \sum_{k=1}^{K} x_{ij}^k \leq\ R, \quad (i,j) \in A \\ &&& x_{ij}^k \in \{0,1\},\ y_{ij} \in \{0,1\}, \quad (i,j) \in A, \quad k = 1, \ldots, K \end{aligned} \tag{8}$$

Observe that not all the feasible solutions of (6) satisfy the new constraints, thus, even though the two problems have the same objective function, their optimal solutions may be quite different.

3.2 Minimization of the number of arc repetitions

Considering the same set of variables x_{ij}^k, $(i,j) \in A$, $k = 1, \ldots, K$, the `MAR` formulation is the following:

$$\text{minimize} \quad v_2(x,w,u) = \sum_{(i,j)\in A} u_{ij} \tag{9a}$$

$$\text{subject to} \quad \sum_{j\in N:(i,j)\in A} x_{ij}^k - \sum_{j\in N:(j,i)\in A} x_{ji}^k = \begin{cases} 1 & i = s \\ 0 & i \neq s,t \\ -1 & i = t \end{cases}, \quad k = 1, \ldots, K \tag{9b}$$

$$\sum_{k=1}^{K} x_{ij}^k \leq K\, w_{ij}, \quad (i,j) \in A \tag{9c}$$

$$u_{ij} = \sum_{k=1}^{K} x_{ij}^k - w_{ij}, \quad (i,j) \in A \tag{9d}$$

$$x_{ij}^k \in \{0,1\},\ w_{ij} \in \{0,1\},\ u_{ij} \geq 0, \quad (i,j) \in A, \quad k = 1, \ldots, K \tag{9e}$$

Constraints (9c), together with (9d) and the non-negativity constraints of the variables u_{ij}, ensure that the binary variables w_{ij} are equal to 1 if and only if the arc $(i,j) \in A$ is used in at least one path, or 0 otherwise. Constraints (9d) define the auxiliary variables u_{ij}, that count the number of times that arc (i,j) is repeated, for $(i,j) \in A$. These variables are implicitly integers. Like before, (9b) are flow conservation constraints that define sets of K paths between the nodes s and t. The objective function v_2 counts the number of repetitions of all the arcs in the K paths.

Finally, the constrained version of the model above, `MARA`, is formulated as follows:

$$\begin{aligned} \text{minimize} \quad & v_2(x,w,u) = \sum_{(i,j)\in A} u_{ij} \\ \text{subject to} \quad & (9b) - (9d) \\ & \sum_{k=1}^{K} x_{ij}^k \leq\ R, \quad (i,j) \in A \\ & x_{ij}^k \in \{0,1\},\ w_{ij} \in \{0,1\},\ u_{ij} \geq 0, \quad (i,j) \in A, \quad k = 1, \ldots, K \end{aligned} \tag{10}$$

Both formulations (9) and (10) admit optimal solutions that contain loops. Nevertheless, it was shown that both have a loopless optimal solution and that such solution can be obtained by applying a polynomial in time algorithm to any given optimal solution Moghanni et al. (2020).

4 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 x^k is the characteristic vector of a path between nodes s and t, its 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 distance or cost. This is the solution of a BOP with the goals of minimizing the number of arcs shared by the paths and minimizing their total cost. 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 shortest and dissimilar K paths problem.

Following the two approaches for the K dissimilar paths problem reviewed in Section 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 (7). Next, we introduce the four models.

4.1 Minimization of the number of repeated arcs

The bi-objective problem that results from extending formulation `MRA`, (6), is formulated as:

$$\text{minimize} \quad v_3(x,y) = \sum_{(i,j)\in A} c_{ij} \sum_{k=1}^{K} x_{ij}^k \tag{11a}$$

$$\text{minimize} \quad v_1(x,y) = \sum_{(i,j)\in A} y_{ij} \tag{11b}$$

$$\text{subject to} \quad \sum_{j\in N:(i,j)\in A} x_{ij}^k - \sum_{j\in N:(j,i)\in A} x_{ji}^k = \begin{cases} 1 & i = s \\ 0 & i \neq s,t \\ -1 & i = t \end{cases}, \quad k = 1,\ldots,K \tag{11c}$$

$$y_{ij} \leq \sum_{k=1}^{K} x_{ij}^k, \quad (i,j) \in A \tag{11d}$$

$$(K-1)y_{ij} \geq \sum_{k=1}^{K} x_{ij}^k - 1, \quad (i,j) \in A \tag{11e}$$

$$x_{ij}^k \in \{0,1\},\ y_{ij} \in \{0,1\}, \quad (i,j) \in A, \quad k = 1,\ldots,K \tag{11f}$$

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

Proposition 2. *Any efficient solution of problem (11) is loopless.*

Proof. Let (x^*, y^*) be an efficient solution of (11). By contradiction, assume that the k'-th path defined by x^* contains one loop, defined by the set $L \subseteq A$. (The reasoning can be replicated if more than one loop exists or several paths contain several loops.)

Let $x^*_{A-L} \in \{0,1\}^{Km}$ be 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 $y^*_{A-L} \in \{0,1\}^m$ be a vector equal to y^* for the positions $(i,j) \in A-L$

and satisfying the constraints (11d) and (11e) for the remaining positions. Then,

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

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 (11) such that

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

Thus, x^* is dominated by x^*_{A-L}, which contradicts the assumption. □

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 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 (11). 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:

$$\begin{aligned} \text{minimize} \quad & v_3(x, y) = \sum_{(i,j)\in A} c_{ij} \sum_{k=1}^{K} x^k_{ij} \\ \text{subject to} \quad & (6b) - (6d) \\ & \sum_{(i,j)\in A} y_{ij} \leq \epsilon \\ & x^k_{ij} \in \{0,1\},\ y_{ij} \in \{0,1\}, \quad (i,j) \in A, \quad k = 1, \ldots, K \end{aligned} \tag{12}$$

and

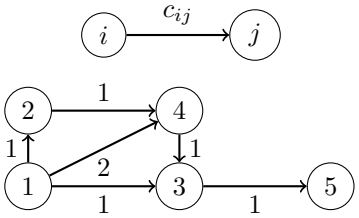
$$\begin{aligned} \text{minimize} \quad & v_1(x, y) = \sum_{(i,j)\in A} y_{ij} \\ \text{subject to} \quad & (6b) - (6d) \\ & \sum_{(i,j)\in A} c_{ij} \sum_{k=1}^{K} x^k_{ij} \leq \epsilon \\ & x^k_{ij} \in \{0,1\},\ y_{ij} \in \{0,1\}, \quad (i,j) \in A, \quad k = 1, \ldots, K \end{aligned} \tag{13}$$

The first of these problems is close to an extension of K shortest path problems. The first point in the result below follows from a reasoning similar to Proposition 2; the second is due to the fact that formulation (13) is closer to formulation (6).

Proposition 3. *1. Any optimal solution of problem (12) is loopless.*

2. At least one optimal solution of problem (13) is loopless.

The loops in a given optimal solution can be discarded by applying a simple algorithm with time of $O(Km)$ presented in Moghanni et al. (2020). Nevertheless, experiments revealed that problem (13) is harder than problem (12). 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_2	$v_3(P_2)$	$v_1(P_2)$
$p_1 = \langle 1, 3, 5\rangle$	$\{p_1, p_1\}$	4	2
$p_2 = \langle 1, 4, 3, 5\rangle$	$\{p_1, p_2\}$	6	1
$p_3 = \langle 1, 2, 4, 3, 5\rangle$	$\{p_1, p_3\}$	6	1

(b) Efficient solutions for `BORA` and `BOAR`

Figure 2: Finding $K = 2$ shortest and dissimilar paths from node 1 to node 5

In order to illustrate the possible consequences of swapping the sub-problems in hand, we consider the problem of finding $K = 2$ paths from node 1 to node 5 in the network in Figure 2a. Figure 2b lists all those paths and the possible solutions, P_2, together with the corresponding objective values. There are three efficient solutions for `BORA`, listed 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 $\{p_1, p_2\}$, and therefore the dissimilarity of the solution is 0.625 in the first case, whereas it is 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.

To conclude this section, we consider the bi-objective version of formulation `MRAA`, obtained by adding the set of constraints (7) to formulation `MRA` to model the dissimilarity constraints.

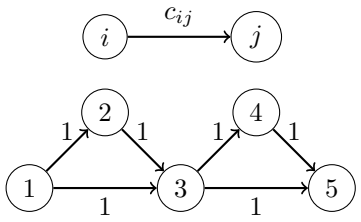
The sub-problems to solve in case of the bi-objective extension of formulation `MRAA` are as follows:

$$
\begin{aligned}
\text{minimize} \quad & v_3(x, y) = \sum_{(i,j)\in A} c_{ij} \sum_{k=1}^{K} x_{ij}^k \\
\text{subject to} \quad & (6b) - (6d) \\
& \sum_{k=1}^{K} x_{ij}^k \leq R, \quad (i,j) \in A \\
& \sum_{(i,j)\in A} y_{ij} \leq \epsilon \\
& x_{ij}^k \in \{0,1\},\ y_{ij} \in \{0,1\}, \quad (i,j) \in A, \quad k = 1, \ldots, K
\end{aligned}
\tag{14}
$$

and

$$
\begin{aligned}
\text{minimize} \quad & v_1(x,y) = \sum_{(i,j)\in A} y_{ij} \\
\text{subject to} \quad & (6b) - (6d) \\
& \sum_{k=1}^{K} x_{ij}^k \leq \; R, \quad (i,j) \in A \\
& \sum_{(i,j)\in A} c_{ij} \sum_{k=1}^{K} x_{ij}^k \leq \epsilon \\
& x_{ij}^k \in \{0,1\}, \; y_{ij} \in \{0,1\}, \quad (i,j) \in A, \quad k = 1,\ldots,K
\end{aligned}
\tag{15}
$$

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



(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		3
$\{p_1,p_1,p_3\}$	7		3
$\{p_1,p_1,p_4\}$	8		2
$\{p_1,p_2,p_3\}$	8		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`

Figure 3: Finding $K = 3$ shortest and dissimilar paths from node 1 to node 5

Figure 3 illustrates the effect of adding constraints (7) to the previous problem, when seeking for $K = 3$ paths. There is only one efficient solution for `BORA`, the set $\{p_1, p_1, p_1\}$ listed in Figure 3c. When imposing that the arcs cannot appear more than twice ($R = 2$) in the paths from 1 to 5 that solution becomes unfeasible. Therefore, there are two efficient solutions in this case, listed in Figure 3d.

The two questions discussed earlier regarding the sub-problems to solve by the ϵ-constraint method, (12) or (13), also arise when analysing (14) and (15). Since adding the constraints (7) 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 Minimization of the number of arc repetitions

Considering again the variables defined in Section 3, the bi-objective problem can be written as:

$$\text{minimize} \quad v_3(x,w,u) = \sum_{(i,j)\in A} c_{ij} \sum_{k=1}^{K} x_{ij}^k \tag{16a}$$

$$\text{minimize} \quad v_2(x,w,u) = \sum_{(i,j)\in A} u_{ij} \tag{16b}$$

$$\text{subject to} \quad \sum_{j\in N:(i,j)\in A} x_{ij}^k - \sum_{j\in N:(j,i)\in A} x_{ji}^k = \begin{cases} 1 & i = s \\ 0 & i \neq s,t \\ -1 & i = t \end{cases}, \quad k = 1,\ldots,K \tag{16c}$$

$$\sum_{k=1}^{K} x_{ij}^k \leq K\, w_{ij}, \quad (i,j) \in A \tag{16d}$$

$$u_{ij} = \sum_{k=1}^{K} x_{ij}^k - w_{ij}, \quad (i,j) \in A \tag{16e}$$

$$x_{ij}^k \in \{0,1\},\ w_{ij} \in \{0,1\},\ u_{ij} \geq 0, \quad (i,j) \in A, \quad k = 1,\ldots,K \tag{16f}$$

designated as `BOAR` in the following. The reasoning used in Proposition 2 holds to prove the result below.

Proposition 4. *Any efficient solution of problem (16) 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:

$$\begin{aligned} \text{minimize} \quad & v_3(x,w,u) = \sum_{(i,j)\in A} c_{ij} \sum_{k=1}^{K} x_{ij}^k \\ \text{subject to} \quad & (9b) - (9d) \\ & \sum_{(i,j)\in A} u_{ij} \leq \epsilon \\ & x_{ij}^k \in \{0,1\},\ w_{ij} \in \{0,1\},\ u_{ij} \geq 0, \quad (i,j) \in A, \quad k = 1,\ldots,K \end{aligned} \tag{17}$$

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

Again, the constrained version of (16) is obtained by adding the constraints (7) to the model, thus:

$$\begin{array}{ll} \text{minimize} & v_3(x,w,u) = \sum_{(i,j)\in A} c_{ij} \sum_{k=1}^{K} x_{ij}^k \\ \text{subject to} & (9b) - (9d) \\ & \displaystyle\sum_{k=1}^{K} x_{ij}^k \leq \; R, \quad (i,j) \in A \\ & \displaystyle\sum_{(i,j)\in A} u_{ij} \leq \epsilon \\ & x_{ij}^k \in \{0,1\}, \; w_{ij} \in \{0,1\}, \; u_{ij} \geq 0, \quad (i,j) \in A, \quad k = 1,\ldots,K \end{array} \tag{18}$$

hereinafter designated as `BOARA`. Like before, the premises of the discussion regarding (16) also hold when analysing (18). Therefore, we consider that function v_1 is minimized, while function v_2 is restricted, and that $\Delta = 1$ is used.

Figure 3 illustrates also the effect of adding constraints (7) to (16). There are five efficient solutions for `BOAR`, all the sets listed in Figure 3c. When imposing that the arcs cannot appear more than twice ($R = 2$) in the paths from 1 to 5, many of those solutions become unfeasible, remaining two efficient solutions, listed in Figure 3d.

To conclude this section, we consider the bi-objective version of formulation `MRAA`, obtained by adding the set of constraints (7) to formulation `MRAA` to model the dissimilarity constraints. Revisiting the problem depicted in Figure 3, the five sets of paths listed in Figure 3c are all efficient solutions for formulation `BOAR`. When preventing each arc from appearing in more than $R = 2$ paths from 1 to 5, the first three solutions in that table become unfeasible, and then there are two efficient solutions for formulation `BOARA`, shown in Figure 3d. It is interesting to observe that, unlike what happened when adding the new constraints to problem `BORA`, in this case the number of efficient solutions decreases.

5 Computational results

Empirical experiments were run in order to evaluate the presented methods and formulations from an empirical point of view. The purpose of the tests is bifold: i. to assess the extent to which the ϵ-constraint algorithm is efficient when increasing ϵ and when compared to the decreasing strategy; ii. to compare the performance of the introduced formulations for solving the bi-objective problem of finding sets of K shortest and dissimilar paths, based on the dissimilarity index (5).

The two ϵ-constraint algorithms were implemented for the four formulations presented in Section 4. The following codes were implemented:

- codes `DEC.BORA`, `DEC.BORAA`, `DEC.BOAR` and `DEC.BOARA`, for finding the non-dominated set using Algorithm 1 for formulations (11), (16), (14) and (18), respectively;
- codes `IEC.BORA`, `IEC.BORAA`, `IEC.BOAR` and `IEC.BOARA`, for finding the non-dominated set using Algorithm 2 for formulations (11), (16), (14) and (18), respectively.

The eight variants of the methods were coded in C language, calling CPLEX 20.1 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 1 or 2, taking $\Delta = 1$.

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

- Random graphs, $R_{n,m,\delta}$, with $n = 100, 500$ nodes, obtained generating randomly $m = dn$ arcs, with $d = 5, 10, 15$, and $\delta = \frac{100m}{n(n-1)}\%$ the density of the graph, equal to 5%, 10% and 15% if $n = 100$ and 1%, 2% and 5% if $n = 500$.
- Grid graphs, $G_{p,q}$, comprising 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.

Table 1: Description of the column headings

Heading	Description
$\bar{T}$	Average total run time, in seconds
$\bar{T}_R$	Average run time for finding R Moghanni et al. (2020), 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

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 of them 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 1. In this table `AvDi` stands for the average dissimilarity of a given set of K paths. The results are obtained over the 20 instances solved for each type of network.

In the following we discuss the results of the application of the bi-objective Algorithms 1 and 2 to the integer programming formulations (11), (14), (16) and (18) for the instances described above. We first consider the results for the formulations that aim at minimizing the number of repeated arcs and afterwards focus on the minimization of the number of arc repetitions.

5.1 Minimization of the number of repeated arcs

The average results obtained when finding the non-dominated set for the unconstrained formulation that looks for the repeated arcs minimization are presented on Tables 2 and 3.

Table 2 summarizes the results in terms of the number of solved sub-problems and run times. In average, when talking about the decreasing version of the ϵ-constraint algorithm, the number of solved sub-problems was $\bar{N} = |\bar{Y}_E| + 1$. This number increased by around 0.5 more sub-problems solved when the parameter ϵ is increased. When applied to grids this code did not solve all the sub-problems in 300 seconds time. In that case the algorithm proceeds using the best solution found within that time, even if it may be a sub-optimal solution. Moreover, 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

Table 2: Number of sub-problems and run times (in seconds) for the unconstrained version of minimizing the number of repeated arcs

		DEC.BORA			IEC.BORA		
Instance	$\bar{\lvert Y_E \rvert}$	$\bar{N}$	$\bar{T}$	$\bar{T}/\bar{N}$	$\bar{N}$	$\bar{T}$	$\bar{T}/\bar{N}$
$R_{100,500,5}$	3.21	4.21	15.702	3.72	4.73	12.563	2.65
$R_{100,1000,10}$	4.00	5.00	32.314	6.46	5.45	28.793	5.28
$R_{100,1500,15}$	3.70	4.70	26.685	5.67	4.75	23.654	4.97
$R_{500,2500,1}$	5.25	6.25	185.250	29.64	6.63	166.059	25.04
$R_{500,5000,2}$	4.75	5.75	199.338	34.66	6.25	190.491	30.47
$R_{500,7500,5}$	4.95	5.95	281.739	47.35	6.30	265.316	42.11
$G^*_{12,12}$	7.00	8.00	2 702.111	337.76	8.50	2 102.347	247.33
$G^*_{4,36}$	1.00	2.00	601.260	300.63	3.00	301.550	100.51
$G^*_{15,15}$	13.00	14.00	4 501.274	321.51	14.60	3 901.978	267.25
$G^*_{5,45}$	1.00	2.00	603.415	301.70	3.00	304.764	101.58

$*$ Sub-problems interrupted after 300 seconds.

solution is simply formed by $K = 10$ paths (see Figures 2 and 3), all equal to the shortest path and with fully overlapping arcs.

The average run times for solving the same problem are also presented in Table 2. The average time for solving each sub-problem was shorter when applying the increasing version of the ϵ-constraint algorithm, and this speedup is observed for the total run times as well. It can also be noted that the difference in the run times of the two versions increases with the size of the instance and that its magnitude is bigger in the case of the grid instances. The partial run times depend mainly on n in the random instances and also tend to increase with d. The biggest random instances were solved by code `IEC.BORA` in less than 266 seconds. The results obtained for grids follow the same trend, with the difference that the corresponding sub-problems are 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.

Table 3: Characteristics of the non-dominated points for the unconstrained version of minimizing the number of repeated arcs

Instance	$\lvert Y_E \rvert$	$\bar{f}^1_{\min}$	$\bar{f}^1_{\max}$	$\bar{f}^2_{\min}$	$\bar{f}^2_{\max}$	$\bar{D}_{\min}$	$\bar{D}_{\max}$
$R_{100,500,5}$	3.21	906.80	1 530.60	2.21	4.42	0.000	0.518
$R_{100,1000,10}$	4.00	555.50	1 097.50	1.00	4.15	0.015	0.855
$R_{100,1500,15}$	3.70	387.00	827.50	0.80	3.55	0.010	0.905
$R_{500,2500,1}$	5.25	1 239.40	2 044.30	1.93	6.31	0.013	0.833
$R_{500,5000,2}$	4.75	761.00	1 218.20	1.10	5.10	0.006	0.867
$R_{500,7500,5}$	4.95	481.00	895.80	0.70	4.75	0.010	0.936
$G^*_{12,12}$	7.00	6 596.00	9 386.50	16.00	22.00	0.010	0.779
$G^*_{4,36}$	1.00	14 676.00	14 676.00	38.00	38.00	0.000	0.000
$G^*_{15,15}$	13.00	7 842.00	12 417.75	16.00	28.00	0.003	0.864
$G^*_{5,45}$	1.00	18 660.50	18 660.50	48.00	48.00	0.000	0.000

* Sub-problems interrupted after 300 seconds.

According to Table 3, and as expected, the range of the cost, f_1, is larger than the range of the number of repeated arcs, f_2. The latter is rather small, which results in a small number of non-dominated points, 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 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. As pointed out by the example in Figure 2, different solutions may be associated to the same non-dominated point and, when this happens, the associated dissimilarities may differ as well. For this reason, in some cases, the average dissimilarities of the solutions obtained by `IEC.BOAR` and `DEC.BOAR` may be different. In the tested instances, this situation rarely occurred and in that case the difference was always smaller than 0.006. Therefore, the reported values are averages of those values.

In the random instances the average maximum dissimilarity grows with d, and specially with n, varying from 0.518 to 0.936. The average minimum dissimilarity, on the other hand, is always nearly 0. As mentioned, this is explained by the fact that most of the determined sets contain one solution formed by paths that coincide with the shortest.

Few conclusions can be drawn from the results on grids, due to the interruptions. As noted before, only one solution was found on rectangular grids. In this case, the solution is formed by several shortest paths, which is confirmed by the average values $\bar{D}_{\min}$ in Table 3, always close to 0.

Table 4: Number of sub-problems and run times (in seconds) for the constrained version of minimizing the number of repeated arcs

			DEC.BORAA			IEC.BORAA		
Instance	$\lvert\bar{Y_E}\rvert$	$\bar{T}_R$	$\bar{N}$	$\bar{T}$	$\bar{T}/\bar{N}$	$\bar{N}$	$\bar{T}$	$\bar{T}/\bar{N}$
$R_{100,500,5}$	7.84	0.793	8.84	45.406	5.13	10.31	43.799	4.24
$R_{100,1000,10}$	11.05	1.507	12.05	70.850	5.90	14.70	76.057	5.17
$R_{100,1500,15}$	11.45	1.955	12.45	42.264	3.39	16.85	51.157	3.03
$R_{500,2500,1}$	12.25	1.891	13.25	334.251	25.22	15.18	333.326	21.95
$R_{500,5000,2}$	15.15	27.864	16.15	221.579	13.72	21.45	254.715	11.87
$R_{500,7500,5}$	11.30	59.171	12.30	185.666	15.09	17.50	216.820	12.38
$G^*_{12,12}$	28.95	0.123	29.95	6 714.120	224.17	30.95	6 133.271	198.16
$G^*_{4,36}$	3.00	0.232	4.00	1 319.615	329.90	5.00	745.411	149.08
$G^*_{15,15}$	40.70	0.180	41.70	11 251.766	269.82	42.85	10 687.353	249.41
$G^*_{5,45}$	7.00	0.681	8.00	2 604.887	325.61	9.00	2 007.209	223.02

$*$ Sub-problems interrupted after 300 seconds.

Tables 4 and 5 summarize the results obtained by the codes that implement the constrained versions of `BORA`, `DEC.BORAA` and `IEC.BORAA`. Besides the values presented before, Table 4 includes the average time for solving the auxiliary problem for finding the bound R to use in the new constraints (7). These values are small compared to the times required for solving the bi-objective problems. Still they seem to increase fast with n, or even with d, and the bigger instances were solved in 59.171 seconds. The first remark about the constrained version of `BORA` is that the number of non-dominated points it obtained is (two or three times) bigger than what was reported for the unconstrained one in Table 2. In general, the constrained sub-problems were easier to solve for the sparser instances, but more difficult in most cases. This effect is

Table 5: Characteristics of the non-dominated points for the constrained version of minimizing the number of repeated arcs

Instance	$\lvert Y_E \rvert$	$\bar{f}^1_{\min}$	$\bar{f}^1_{\max}$	$\bar{f}^2_{\min}$	$\bar{f}^2_{\max}$	$\bar{D}_{\min}$	$\bar{D}_{\max}$
$R_{100,500,5}$	7.84	1 264.37	1 825.05	3.74	11.05	0.514	0.814
$R_{100,1000,10}$	11.05	953.80	1 191.75	2.80	14.55	0.829	0.958
$R_{100,1500,15}$	11.45	784.25	934.10	2.40	16.25	0.903	0.982
$R_{500,2500,1}$	12.25	1 516.00	2 107.56	3.63	15.88	0.624	0.911
$R_{500,5000,2}$	15.15	1 057.70	1 329.35	3.30	22.05	0.730	0.940
$R_{500,7500,5}$	11.30	818.25	937.35	2.15	16.70	0.899	0.984
$G^*_{12,12}$	28.95	7 296.00	10 563.90	16.00	44.00	0.558	0.919
$G^*_{4,36}$	3.00	16 415.50	17 252.90	74.00	76.00	0.556	0.651
$G^*_{15,15}$	40.70	8 764.75	13 672.05	16.00	56.00	0.557	0.937
$G^*_{5,45}$	7.00	21 035.00	22 853.00	90.00	96.00	0.556	0.694

$*$ Sub-problems interrupted after 300 seconds.

magnified by the general increase in the number of non-dominated points. Therefore, the total run times for solving the constrained problems were only shorter than when solving their constrained versions for the smaller and sparser instances, with average degree 5.

Additionally, the increasing version of the method was between about 11% and 20% faster than the decreasing version to solve the sub-problems in the 100 node random instances, and between 4% and 10% faster than the decreasing version in the 500 node instances. Moreover, when applying the code `IEC.BORAA`, the number of solved sub-problems increased from 16% to 35%, and from 15% to 42%, compared to `DEC.BORAA`, again in 100 and 500 node random instances, respectively. The result of this trade-off is that `IEC.BORAA` only outperformed `DEC.BORAA` in the sparser random instances. The improvement in the partial run times is bigger for grids, and therefore `IEC.BORAA` was always faster than `DEC.BORAA` for these instances, despite the increase in the number of sub-problems that were solved. In terms of the total time, `IEC.BORAA` was between 5% and 43% faster than `DEC.BORAA`.

Figure 7 compares the costs and dissimilarities and puts in evidence some characteristics of the sets of non-dominated points for the unconstrained and the constrained problems when counting the number of repeated arcs. For the unconstrained problem, the costs decrease and the maximum dissimilarities worsen with the average degree of the random instances, regardless of its number of nodes. The trend is similar for the constrained version of the problem, but the minimum cost of the solutions increases from 22% to 103%, whereas the maximum cost increases only from 3% to 20%, both in the random instances. The same happens to the maximum dissimilarity, but specially with the minimum dissimilarity, which is in accordance with the situation illustrated in Figure 3. In a nutshell, more solutions are found in the constrained version of the problem, with higher costs, but slightly better maximum dissimilarities and also fairly better minimum dissimilarities.

The comparison is less clear for the grids due to the limitations of the code. Nevertheless, the general conclusions are similar to the above: more non-dominated points are found for the constrained problem than for the unconstrained one, with higher costs and better dissimilarities, in particular the minimum dissimilarities. In average, less than 16 in 500 node random networks and than 41 in 15×15 grids.

5.2 Minimization of the number of arc repetitions

A comparison of the results of DEC.BOAR and IEC.BOAR is summarized in Tables 6 and 7. The average number of non-dominated points computed by the methods was between 24 and 39 for the random instances and between 102 and 189 for grids. All sub-problems were solved within 300 seconds. The partial run times of the increasing version of the ϵ-constraint method were shorter than those of the decreasing version. Nevertheless, the difference between them does not compensate the bigger number of sub-problems solved by the former, and therefore DEC.BOAR was faster than IEC.BOAR for all instances. For the networks $R_{500,7500,5}$, the code DEC.BOAR computed 34 solutions in average time of 517.432 seconds. The method IEC.BOAR obtained the same solutions in an average time of 568.557 seconds. For the most difficult grid instances, $G_{15,15}$, 188 efficient solutions were found in 718.547 seconds.

Table 6: Number of sub-problems and run times (in seconds) for the unconstrained version of minimizing the number of arc repetitions

		DEC.BOAR			IEC.BOAR		
Instance	$\lvert\bar{Y_E}\rvert$	$\bar{N}$	$\bar{T}$	$\bar{T}/\bar{N}$	$\bar{N}$	$\bar{T}$	$\bar{T}/\bar{N}$
$R_{100,500,5}$	24.68	25.68	51.483	2.00	31.31	57.516	1.83
$R_{100,1000,10}$	28.55	29.55	108.051	3.65	35.20	117.345	3.33
$R_{100,1500,15}$	27.25	28.25	104.745	3.70	32.15	109.738	3.41
$R_{500,2500,1}$	38.12	39.12	339.510	8.67	50.93	401.132	7.87
$R_{500,5000,2}$	34.15	35.15	389.859	11.09	42.85	438.223	10.22
$R_{500,7500,5}$	34.36	35.36	517.432	14.63	41.15	568.557	13.81
$G_{12,12}$	139.35	140.35	701.966	5.00	158.90	743.731	4.68
$G_{4,36}$	102.80	103.80	639.974	6.16	130.20	755.640	5.80
$G_{15,15}$	188.20	189.20	718.547	3.79	213.25	729.979	3.42
$G_{5,45}$	167.60	168.60	1 605.432	9.52	205.85	1 799.108	8.73

Table 7 shows that the range of f_1 is wider than that of the number of arc repetitions, f_2, and also that not all the values in the range of f_2 correspond to one non-dominated point. Moreover, like for the previous problem, the average minimum dissimilarity in each efficient set of K paths is either 0 or near 0. The cost of the solutions increases with n and with d in the random instances, while it is bigger in the rectangular grids as well. Additionally, the average maximum dissimilarity is better for the bigger random instances and for the square grids. The best average results in random networks were obtained in the $R_{500,7500,5}$ instances, with 0.017 minimum dissimilarity and 0.977 maximum dissimilarity.

Finally, Tables 8 and 9 show the results obtained by the codes DEC.BOARA and IEC.BOARA, the constrained versions of the codes for minimizing the number of arc repetitions. In this case the number of non-dominated points is smaller than for the corresponding unconstrained problem. The number of sub-problems to solve when increasing ϵ can be 53% bigger than without the constraints, which slows down the method when compared with a decreasing implementation.

Taking Table 9 into account, the sub-problems are more difficult when including the constraints (7) than without them. The minimum cost of the solutions increased, due to paths that may not satisfy the new constraints. However, the most noticeable change was, again, the improvement of the minimum dissimilarity to at least 0.556, for rectangular grids. The maximum dissimilarity also increased, but less, which indicates that the constraints affect mostly

Table 7: Characteristics of the non-dominated points for the unconstrained version of minimizing the number of arc repetitions

Instance	$\lvert Y_E\rvert$	$\bar{f}^1_{\min}$	$\bar{f}^1_{\max}$	$\bar{f}^2_{\min}$	$\bar{f}^2_{\max}$	$\bar{D}_{\min}$	$\bar{D}_{\max}$
$R_{100,500,5}$	24.68	906.80	1 821.40	11.42	39.78	0.000	0.826
$R_{100,1000,10}$	28.55	555.50	1 137.20	4.70	36.90	0.012	0.931
$R_{100,1500,15}$	27.25	387.00	888.30	2.70	31.85	0.010	0.970
$R_{500,2500,1}$	38.12	1 239.40	2163.90	8.50	56.50	0.013	0.910
$R_{500,5000,2}$	34.15	761.00	1 246.10	5.75	45.60	0.006	0.933
$R_{500,7500,5}$	34.36	481.00	917.10	2.63	40.78	0.017	0.977
$G_{12,12}$	139.35	6 596.00	10 363.80	40.00	196.85	0.010	0.919
$G_{4,36}$	102.80	14 676.00	17 337.50	214.00	342.00	0.000	0.675
$G_{15,15}$	188.20	7 842.00	12 629.90	40.00	251.25	0.008	0.936
$G_{5,45}$	167.60	18 660.50	22 169.40	228.00	431.80	0.000	0.757

Table 8: Number of sub-problems and run times (in seconds) for the constrained version of minimizing the number of arc repetitions

		DEC.BOARA			IEC.BOARA		
Instance	$\lvert\bar{Y_E}\rvert$	$\bar{N}$	$\bar{T}$	$\bar{T}/\bar{N}$	$\bar{N}$	$\bar{T}$	$\bar{T}/\bar{N}$
$R_{100,500,5}$	22.10	23.10	53.156	2.30	28.57	60.411	2.11
$R_{100,1000,10}$	15.65	16.65	67.177	4.03	20.50	79.107	3.85
$R_{100,1500,15}$	11.00	12.00	60.632	5.05	17.65	83.174	4.71
$R_{500,2500,1}$	28.68	29.68	300.262	10.11	37.25	348.602	9.35
$R_{500,5000,2}$	20.15	21.15	317.971	15.03	30.90	419.021	13.56
$R_{500,7500,5}$	12.40	13.40	273.993	20.44	20.15	352.712	17.50
$G_{12,12}$	120.10	121.10	559.768	4.62	137.65	595.977	4.32
$G_{4,36}$	59.40	60.40	356.390	5.90	85.95	475.187	5.52
$G_{15,15}$	159.45	160.45	607.412	3.78	184.85	630.729	3.41
$G_{5,45}$	114.95	115.95	1 082.416	9.33	154.20	1 295.232	8.39

the solutions with small cost, containing many arc repetitions. This is also shown in Figure 8, in Appendix. Here, it is interesting to note that the cost and dissimilarity of the constrained solutions are almost a subset of the images obtained for the unconstrained problems.

5.3 Overall Comparison

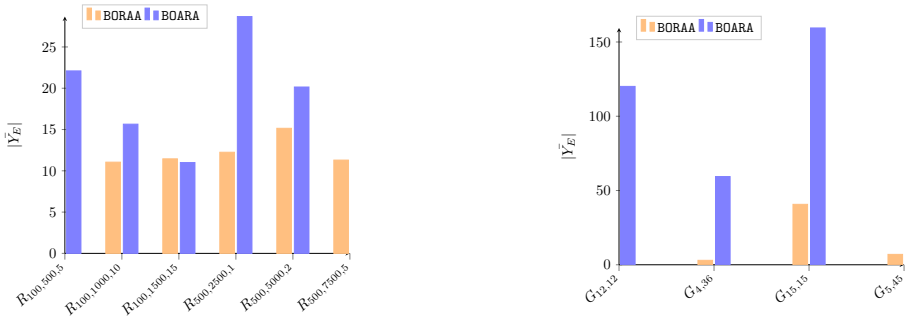
We now compare the constrained versions of the minimization of the number of repeated arcs and of the minimization of the number of arc repetitions. As noted in the previous sections, adding constraints on the number of each arc presences affects differently formulations `BORA` and `BOAR`:

- For the first of these approaches, most efficient sets of paths found for `BORA` became unfeasible and were excluded after adding the constraints. New sets of paths, with higher costs and greater dissimilarities, were obtained with `BORAA`. The sub-problems associated with this formulation were easier to solve than in the unconstrained formulation. Thus, in spite of more sub-problems being solved, the total run times decreased for all the instances, except the densest with 500 nodes.

Table 9: Characteristics of the non-dominated points for the constrained version of minimizing the number of arc repetitions

Instance	$\lvert Y_E \rvert$	$\bar{f}^1_{\min}$	$\bar{f}^1_{\max}$	$\bar{f}^2_{\min}$	$\bar{f}^2_{\max}$	$\bar{D}_{\min}$	$\bar{D}_{\max}$
$R_{100,500,5}$	22.10	1 264.36	1 900.94	12.00	37.57	0.514	0.853
$R_{100,1000,10}$	15.65	953.80	1 195.50	4.70	22.20	0.830	0.959
$R_{100,1500,15}$	11.00	784.25	934.10	2.70	17.35	0.903	0.982
$R_{500,2500,1}$	28.68	1 516.00	2 152.50	9.62	44.00	0.627	0.923
$R_{500,5000,2}$	20.15	1 057.70	1 324.15	5.95	33.90	0.730	0.947
$R_{500,7500,5}$	12.40	818.25	937.35	2.75	19.90	0.900	0.984
$G_{12,12}$	120.10	7 296.00	10 363.80	40.00	175.40	0.558	0.920
$G_{4,36}$	59.40	16 415.50	17 470.00	220.00	303.65	0.556	0.746
$G_{15,15}$	159.45	8 764.75	12 629.90	40.00	222.80	0.560	0.937
$G_{5,45}$	114.95	20 215.50	22 265.30	232.00	383.80	0.556	0.796

- For the second approach, contrariwise, most efficient solutions found by `BOAR` are not discarded when adding the extra constraints, while a few others with a greater cost and slightly better maximum dissimilarity are found. The sub-problems associated with `BOARA` were more difficult to solve than when not considering the constraints, but fewer sub-problems are solved then, which results in a decrease of the total run time for most instances, except the sparser ones.



(a) Random networks

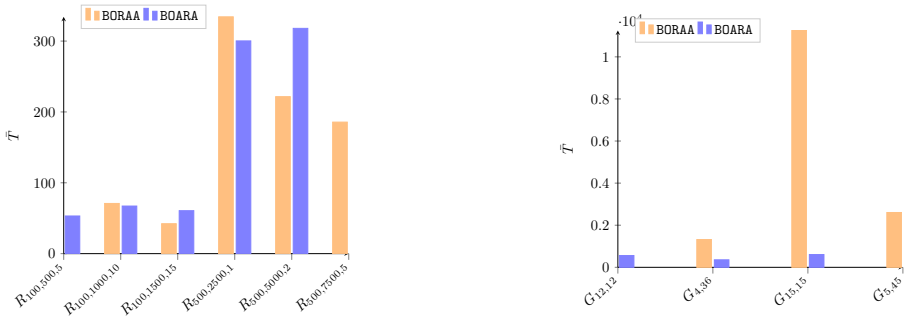
(b) Grid networks

Figure 4: Average number of computed non-dominated points for the constrained problems

In order to compare the two constrained formulations, the number of efficient sets of $K = 10$ paths and the total run time required by the codes `BOARA` and `BOARA` are summarized in Figures 4 and 5. The reported times are the best, considering those of the decreasing and the increasing versions of the ϵ-constraint method for each formulation.

According to Figure 4, $|\bar{Y}_E|$ increases slowly with the density of the network for `BORAA`, except for $R_{500,7500,5}$, while the opposite happens for `BOARA`. Moreover, this number is also considerably bigger when minimizing the number of arc repetitions than when minimizing the number of repeated arcs. In the grid instances the two models behave similarly to what was observed for the random instances. Additionally, there are more solutions in square grids than in rectangular grids.

In terms of the run time, the sub-problems of `BORAA` are more difficult to solve than those of



(a) Random networks

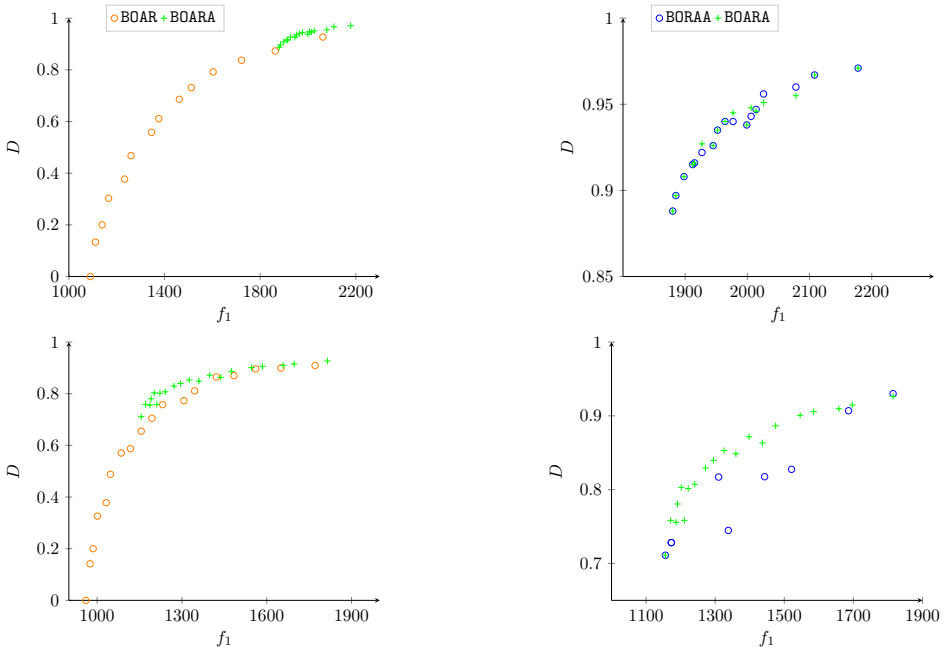
(b) Grid networks

Figure 5: Average total run times (in seconds) for the constrained problems

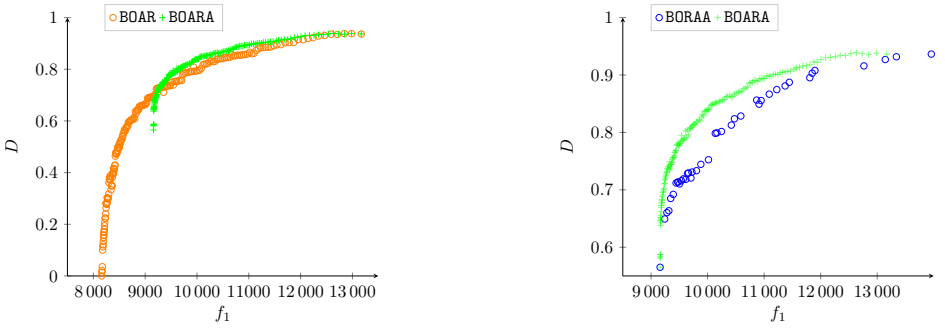
BOARA, in particular in the grid instances. This is balanced by the fact that fewer sub-problems are solved in the case of the first model. Therefore, the difference in the total run times of the codes is not significant in the random instances, besides for $R_{500,2500,1}$, as shown in Figure 5. The approach BOARA was clearly more efficient than BORAA in the grid instances.

In Figure 9, in Appendix, each rectangle shows the range of the cost and of the dissimilarity of the efficient solutions produced by the models in comparison, BOARA and BORAA. For the random instances, we observe an almost full overlap between the rectangles associated with each model, whereas, for the grid instances, BOARA gives slightly more costly and more dissimilar solutions than BORAA. Based on this observation, it seems reasonable to conclude that the two models behave similarly under the additional constraints. This finding together with the information gathered from Figures 7 and 8, in Appendix, would mean that the extra constraints permit to mitigate the shortcomings of BORA in relation to BOAR. To clarify this issue, the points associated to the cost and dissimilarity of the efficient solutions found by both methods were also represented, so that their distributions within the rectangles could be assessed (see Figure 6). In fact, although the points are uniformly distributed in the interval defined by the best for the cost and the best for the dissimilarity for both methods, the values of the points associated to BORA are almost always worse and its distribution is slightly inconsistent, regardless of the type of network under consideration. Therefore, although Figure 9 places the points in the same range, the differences observed in the distributions associated to those points allows to differentiate the quality of the solutions found by each one of the two models.

Finally, we also compared the distribution of the cost and dissimilarity points associated to the efficient solutions produced by BOAR and BOARA, since the previous results indicate these models as the most promising ones. Figure 6 shows that BOARA produces solutions in a narrower range, higher in cost and dissimilarity, whereas BOAR seems more versatile, offering solutions also in the lower range of both cost and dissimilarity. Thus, the first model seems more fitted to applications where the focus is on finding sets of highly dissimilar paths, even if this means substantially higher costs, while the latter seems more fitted to applications where the cost of the solution is a major concern. In this way, both models are of interest in the context of the bi-objective shortest-dissimilar K paths problem.



(a) Solutions for two instances in $R_{100,500,5}$



(b) Solutions for an instance in $G_{12,12}$

Figure 6: Solutions for particular instances

6 Concluding remarks

This paper addressed the problem of finding sets of K paths that are as dissimilar as possible, while minimizing their total cost. A bi-objective approach for finding efficient solutions to this problem was introduced. The approach consists of a modification of the ϵ-constraint method in two versions: decreasing and increasing the parameter ϵ. The two versions were empirically tested on random and grid instances and the computational results were 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 subproblems 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 versions of the models, `BORAA` and `BOARA`, improved the average maximum dissimilarity of the solutions up to 0.937. However, although the increase is quite significant in the first case it

had very little impact in the latter. In turn, the average minimum dissimilarity of the solution was greatly improved when using both `BORAA` and `BOARA`. 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 networks, and 0.556 in the grid networks, preventing some solutions of little practical interest for most applications from being found.

Finally, a detailed analysis of the results indicates that `BOARA` produces better and more consistent results when compared to `BORAA`. The code `BORA` was faster than `BOAR` for the random instances, because less non-dominated points were computed in that case, while the opposite happened for the grid networks. As to the run times for the constrained versions of the problems, either they were not affected by the new constraints or even decreased, and the two approaches behaved similarly with this respect.

Acknowledgments

This work was partially financially supported by the Portuguese Foundation for Science and Technology (FCT) under project grants UID/MAT/04561/2019, UID/MAT/00324/2020 and UID/MULTI/00308/2020. The work was also partially supported by project P2020 SAICT-PAC/0011/2015, co-financed by COMPETE 2020, Portugal 2020 – Operational Program for Competitiveness and Internationalization (POCI), European Union's European Regional Development Fund, and FCT, and by FEDER Funds and National Funds under project CENTRO-01-0145-FEDER-029312.

References

Ahuja, R., Magnanti, T., Orlin, J., 1993. *Network Flows : Theory, Algorithms and Applications*. Prentice Hall, Englewood Cliffs, NJ.

Akgün, V., Erkut, E., Batta, R., 2000. On finding dissimilar paths. *European Journal of Operational Research* 121, 232–246.

Batta, R., Kwon, C., 2013. *Handbook of OR/MS Models in Hazardous Materials Transportation* (first edn.). Springer, New York.

Calvo, R., Cordone, R., 2003. A heuristic approach to the overnight security service problem. *Computers & Operations Research* 30, 9, 1269–1287.

Chang, C.W., Chen, C.D., Chuang, K.T., 2020. Queries of K-discriminative paths on road networks. *Knowledge and Information Systems* 62, 1751–1780.

Chankong, V., Haimes, Y., 1983. *Multiobjective decision making*, *North-Holland Series in System Science and Engineering*. Vol. 8. North-Holland Publishing Co., New York.

Clímaco, J., Craveirinha, J., 2019. MCDA/M in Telecommunication Networks: Challenges and Trends, Chapman and Hall/CRC, New York, NY. pp. 11–55.

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

Cohon, J., 1978. *Multiobjective programming and planning.* Academic Press, New York.

Constantino, M., Gouveia, L., M.C.Mourão, Nunes, A., 2015. The mixed capacitated arc routing problem with non-overlapping routes. *European Journal of Operational Research* 244, 3, 445–456.

Constantino, M., Mourão, M.C., Pinto, L., 2017. Dissimilar arc routing problems. *Networks* 70, 3, 233–245.

Dadkar, Y., Jones, D., Nozick, L., 2008. Identifying geographically diverse routes for the transportation of hazardous materials. *Transportation Research Part E: Logistics and Transportation Review* 44, 3, 333–349.

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

Ehrgott, M., 2005. *Multicriteria optimization* (second edn.). Springer-Verlag, Berlin.

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

Erkut, E., Tjandra, S., Verter, V., 2007. Hazardous materials transportation. *Handbooks in operations research and management science* 14, 539–621.

Erkut, E., Verter, V., 1998. Modeling of transport risk for hazardous materials. *Operations Research* 46, 625–642.

Gadegaard, S., Klose, A., Nielsen, L., 2018. A bi-objective approach to discrete cost-bottleneck location problems. *Annals of Operations Research* 267, 1-2, 179–201.

Gomes, T., Jorge, L., Girão-Silva, R., Yallouz, J., Babarczi, P., Rak, J., 2020. Fundamental Schemes to Determine Disjoint Paths for Multiple Failure Scenarios, Springer International Publishing, Cham. pp. 429–453.

Gomes, T., Jorge, L., Melo, P., 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.

Haimes, Y., Lasdon, L., 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.

Hu, J.Q., 2003. Diverse routing in optical mesh networks. *IEEE Transactions on Communications* 51, 3, 489–494.

Hughes, M., Lundaya, B., Weir, J., Hopkinson, K., 2021. The multiple shortest path problem with path deconfliction. *Discrete Optimization* 292, 3, 818–829.

Jabbarzadeh, A., Azad, N., Verma, M., 2020. An optimization approach to planning rail hazmat shipments in the presence of random disruptions. *Omega* 96, 102078.

Li, L., Cheema, M.A., Lu, H., Ali, M.E., Toosi, A.N., 2021. Comparing alternative route planning techniques: A comparative user study on melbourne, dhaka and copenhagen. *IEEE Transactions on Knowledge and Data Engineering*

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

Moghanni, A., Pascoal, M., Godinho, M.T., 2020. Finding k dissimilar paths using integer linear formulations. *submitted for publication*

Pascoal, M., Clímaco, J., 2020. On a relaxed maximally disjoint path pair problem: A bicriteria approach. *International Transactions in Operational Research* 27, 2045–2063.

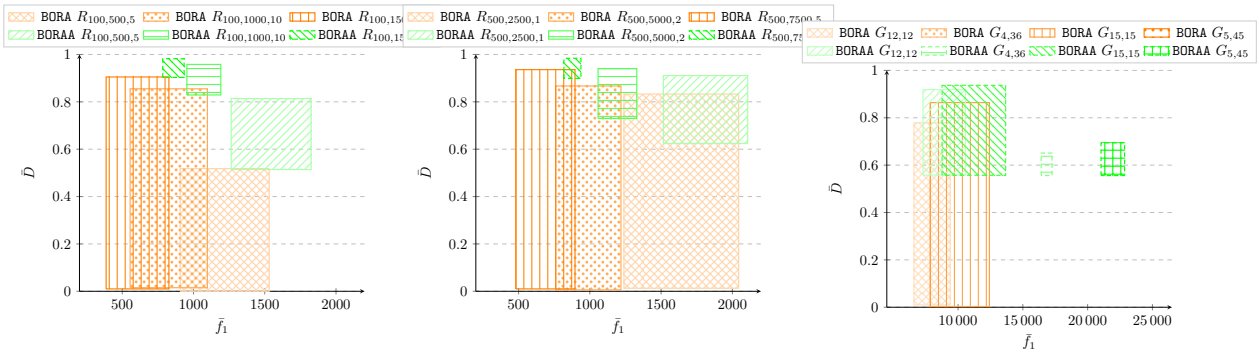
Raith, A., Ehrgott, M., 2009. A comparison of solution strategies for biobjective shortest path problems. *Computers & Operations Research* 36, 4, 1299 – 1331.

Talarico, L., Sörensen, K., Springael, J., 2015. The k-dissimilar vehicle routing problem. *European Journal of Operational Research* 244, 1, 129–140.

Tikani, H., Setak, M., Demir, E., 2021. Multi-objective periodic cash transportation problem with path dissimilarity and arrival time variation. *Expert Systems with Applications* 164, 114015.

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

A Appendix: Cost and dissimilarity for the unconstrained and the constrained problems

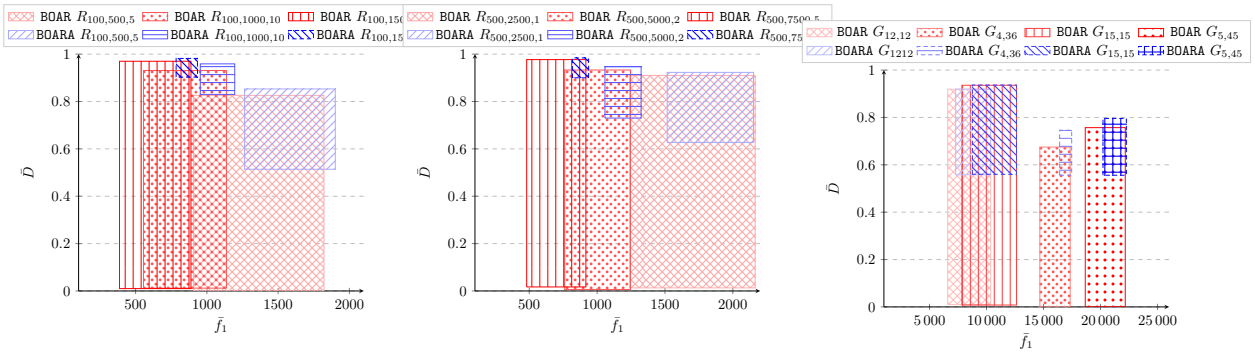


(a) Random networks with 100 nodes

(b) Random networks with 500 nodes

(c) Grid networks

Figure 7: Cost and dissimilarity for the unconstrained and the constrained problems when minimizing the number of repeated arcs

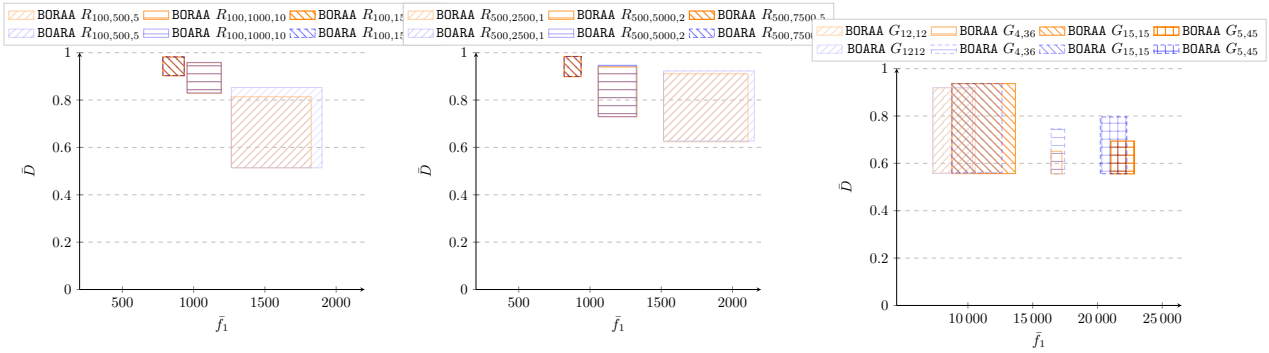


(a) Random networks with 100 nodes

(b) Random networks with 500 nodes

(c) Grid networks

Figure 8: Cost and dissimilarity for the unconstrained and the constrained problems when minimizing the number of arc repetitions



(a) Random networks with 100 nodes

(b) Random networks with 500 nodes

(c) Grid networks

Figure 9: Cost and dissimilarity for the constrained problems