Modeling and Solving the Non-Smooth Arc Routing Problem with Realistic Soft Constraints

Jesica de Armas[a], Albert Ferrer[b], Angel A. Juan[c,*], Eduardo Lalla-Ruiz[d]

[a]*Department of Economics and Business, Universitat Pompeu Fabra & Barcelona GSE, Barcelona, Spain*
[b]*Dept. of Applied Mathematics, Technical University of Catalonia, Barcelona, Spain*
[c]*IN3 – Dept. of Computer Science, Open University of Catalonia, Castelldefels, Spain*
[d]*Institute of Information Systems, University of Hamburg, Hamburg, Germany*

Abstract

This paper considers the non-smooth arc routing problem (NS-ARP) with soft constraints in order to capture in more perceptive way realistic constraints violations arising in transportation and logistics. To appropriately solve this problem, a biased-randomized procedure with iterated local search (BRILS) and a mathematical model for this ARP variant is proposed. An extensive computational study is conducted on rich and diverse problem instances. The results highlight the competitiveness of BRILS in terms of quality and time, where it provides high-quality solutions within reasonable computational times. In the context of real-world environments, the performance exhibited by BRILS motivates its incorporation in intelligent and integrative systems where frequent and fast solutions are required.

Keywords: arc routing problem, soft constraints, non-smooth optimization, biased-randomization, metaheuristics

*Supported by the Spanish Ministry of Economy and Competitiveness & FEDER (TRA2015-71883-REDT, TRA2013-48180-C3-P, SEV-2015-0563)

Email addresses: jesica.dearmas@upf.edu (Jesica de Armas), alberto.ferrer@upc.edu (Albert Ferrer), ajuanp@uoc.edu (Angel A. Juan), eduardo.lalla-ruiz@uni-hamburg.de (Eduardo Lalla-Ruiz)

1. Introduction

In real-life logistics and transportation, many decision-making processes can be modeled as combinatorial optimization problems (OPs) (Montoya-Torres et al., 2012). In a combinatorial OP, a large number of feasible solutions need to be explored in order to select one that minimizes cost, maximizes benefit, etc. Frequently, combinatorial OPs have a well-structured definition consisting of an objective function to be optimized and a set of hard constraints that need to be satisfied (Papadimitriou and Steiglitz, 1982). These OPs can usually be formulated using mixed integer linear programming (MILP) models. However, most of them are NP-hard in nature, which implies that only small- to medium-sized instances can be solved in reasonable computing times by means of classical MILP methods (Garey and Johnson, 1990). Thus, heuristics and metaheuristics are usually required to solve medium- to large-sized instances. An additional difficulty arises when the mathematical model of the problem does not meet certain characteristics. In effect, when properties such as convexity and smoothness are not fulfilled, the solution space might become highly irregular. In such situations, finding an optimal or near-optimal solution becomes a challenging task even for medium-sized instances.

In this work, we study a relevant combinatorial OP under a non-smooth environment. In particular, we focus on the arc routing problem (ARP) with realistic soft constraints, which impose penalty costs on the objective function. These penalty costs are usually defined as piecewise functions, thus transforming the objective function into a non-smooth one. The ARP is a combinatorial OP originally introduced by Golden and Wong (1981). In its basic version, the ARP is composed of a central depot, a fleet of identical vehicles, and a network of arcs connecting nodes. Some of these arcs have a demand that must be served (see Figure 1). Also, there is a cost associated with traversing each of the arcs. Under these circumstances, the usual goal is to find a set of routes (solution) which minimizes the total delivering cost while satisfying the following constraints: *(i)* every route starts and ends at the depot, *(ii)* each arc with a positive demand is served exactly by one vehicle, and *(iii)* the total demand to be served in any route does not exceed the vehicle loading capacity. The main novelty we introduce to this basic version is the inclusion of soft constraints, i.e., we consider a non-smooth penalty term to the objective function to penalize every time a given constraint is not fully satisfied. As discussed in Hashimoto et al. (2006), *"in*

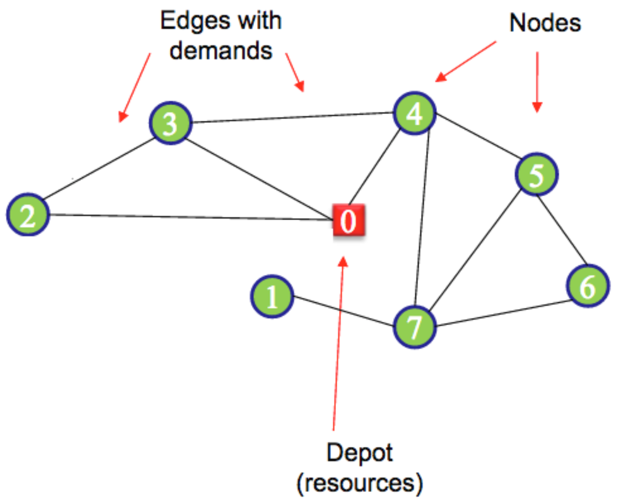

Figure 1: Illustration of the arc routing problem

real-world simulations, time windows and capacity constraints can be often violated to some extent". In this paper thus, a threshold is imposed on the maximum cost (distance-based or time-based) any given route can take. Then, whenever a route exceeds this threshold a penalty cost is added to the total route cost. As in many real-life applications, this penalty cost will depend on the size of the gap between the actual route cost and the threshold.

Accordingly, the main contributions of this paper are: *(i)* a discussion on the importance of considering non-smooth objective functions in realistic combinatorial OPs, mainly due to the existence of soft constraints; *(ii)* an original mathematical model that formally describes the non-smooth ARP (NS-ARP); *(iii)* a metaheuristic approach, based on the integration of biased-randomized techniques (Grasas et al., 2017) inside an iterated local search framework (Lourenço et al., 2010), to efficiently cope with the NS-ARP; and *(iv)* the use of CPLEX to solve a truncated version of the NS-ARP model, which allows us to obtain lower bounds for assessing our metaheuristic approach.

Our current work contributes to the field of Expert and Intelligent Systems in several ways. Firstly, it considers of a more realistic ARP incorporating penalty costs when certain constraints are violated, i.e., introducing soft constraints. This is a more intelligent way to address constraints violations, as we demonstrate comparing to other perspectives. Additionally, as mentioned before, we propose a solving methodology combining biased randomized techniques with an iterated local search metaheuristic framework. This system leads to important savings for the related logistic and transportation industry, and a mathematical model for this non-smooth version of the ARP help us to confirm its goodness. In the context of expert systems, several works have analyzed non-smooth optimization problems. Thus, for instance, Roy et al. (2010) deal with non-smooth power-flow problems, Lu et al. (2010) propose an adaptive hybrid differential evolution algorithm to cope with a non-smooth version of the dynamic economic dispatch problem, and Ferrer et al. (2016) propose a biased-randomized metaheuristic to solve the non-smooth flow-shop problem. As far as we know, however, there is a lack of publications considering realistic non-smooth cost functions in routing problems, which enhances the importance of this expert system.

The remaining of the paper is structured as follows: Section 2 reviews some basic concepts related to non-convex / non-smooth optimization problems, as well as how metaheuristics have been used in solving them. Section 3 presents a survey on recent works on the ARP. A mathematical description of the NS-ARP is provided in Section 4. In Section 5 a biased-randomized iterated local search algorithm is proposed as a solving method for the NS-ARP. Section 6 is devoted to explain the computational experiments performed. Finally, conclusions and future work are described in Section 7.

2. Metaheuristics in non-convex / non-smooth OPs

Optimization problems can be classified as either convex or non-convex. In general, convex OPs have two parts: a series of constraints that represent convex regions and an objective function that is also convex. Linear programming (LP) problems represent a well-known example of convex OPs, since linear functions are trivially convex. Other examples of convex OPs include quadratic programming, geometric programming, conic optimization, or least squares (Boyd and Vandenberghe, 2004). In a convex OP, every constraint restricts the space of solutions to a certain convex region. By taking the intersection of all these regions we obtain the set of feasible solutions, which is also convex. Due to the structure of the solution space, every single local optimum is a global optimum too. This is the key property that allows to efficiently solve convex OPs up to very large instances. Unfortunately, the algorithms applied for solving convex OPs cannot be easily extended to solve the non-convex case. The solution space of the non-convex OPs is far

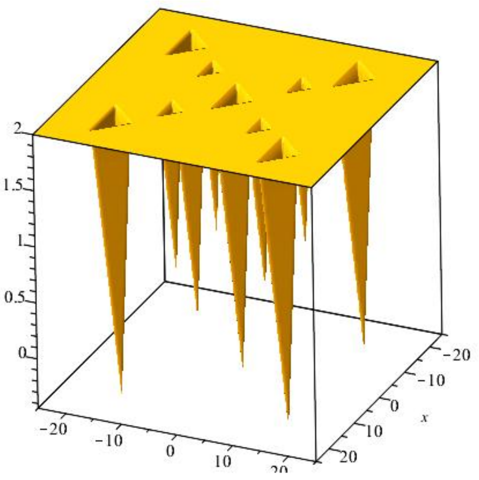

Figure 2: Example of a non-convex and non-smooth objective function.

more complex since the objective function and/or the region of feasible solutions are not convex. As a consequence, there exist many disjoint regions and multiple locally optimal points within each of them (Figure 2). Thus, a traditional local search is not enough since there is a risk of ending in a local optimum that may still be far from the global one. In addition, it can take an unreasonable amount of time to conclude that a non-convex OP is unfeasible, or that the objective function is unbounded, or even that the best solution found so far is, in fact, the global optimum.

Non-smooth OPs are similar to non-convex OPs in the sense that they are much more difficult to solve than traditional smooth and convex problems. A function is smooth if it is differentiable and it has continuous derivatives of all orders. Therefore, a non-smooth function is one that has missed some of these properties. In this case, the solution space might contain again multiple disjoint regions and many locally optimal points within each of them. The computational techniques that can be used to solve non-smooth OPs are often fairly complex and depend on the particular structure of the problem. Unlike convex optimization, non-convex or non-smooth OPs represent a serious challenge in terms of building solution methods that provide the global optimum. Developing such techniques is, in general, a difficult and time-consuming process. Also, the resulting application range is very limited. Due to the inclusion of soft constraints and the associated penalty costs, many real-life objective functions are either non-convex or non-smooth.

An example of a non-convex and non-smooth OP is the minimum sum-of-squares clustering problem described in Bagirov and Yearwood (2006). The authors emphasize that a large number of approaches, such as dynamic programming, branch and bound, or K-means algorithms have been used to deal with it. However, these methods are efficient only in certain special settings. For example, a dynamic programming approach only works when the number of objects is small. Similarly, branch and bound methods are effective only if the number of clusters is not too large. The efficiency of K-means algorithms also decreases as the number of clusters is increased. Moreover, alternative techniques such as bi-linear programming fail as well when they face non-convex and non-smooth objective functions with many local optima. The aforementioned authors remark that, in general, better results are generated when metaheuristics are used for the clustering problem. Thus, for instance, a tabu search approach that outperforms the K-means has been proposed in Al-Sultan (1995). Likewise, Selim and Alsultan (1991) propose a solving approach based on a simulated annealing metaheuristic. Other examples of non-smooth OPs addressed in the literature are the pin fin profile optimization (Badescu, 2017) and the optimal profiles of one dimensional slider bearings under technological constraints (Badescu, 2015). Despite coming from the application of Pontryagin principle, finally these problem reduces to constrained non-linear programming problems.

In telecommunication networks, optimal routing in non-convex / non-smooth scenarios has also been investigated. The objective here is to find the best path for data transmission in a short amount of time. The routing strategy can greatly affect the system performance, so there is a high demand for efficient algorithms. Thus, for instance, a method that combines genetic algorithms with Hopfield networks is proposed in Hamdan and El-Hawary (2002), while an approach based on tabu search is used in Oonsivilai et al. (2009). Other authors present a non-smooth formulation for the location problem in wireless sensor networks (Bagirov et al., 2007). In addition to this, both indoor and outdoor localization systems have been also studied by many authors. However, most approaches have assumed accurate range measurements, which is unrealistic in the case of radio frequency signals. For that reason, some researchers propose probabilistic approaches to deal with range measurements inaccuracy. This is the case in Ramadurai and Sichitiu (2003), where the authors propose and test a probabilistic heuristic algorithm to cope with the problem of estimating the position (spatial coordinates) of nodes in wireless sensor networks. Their algorithm is distributed and takes into account uncertainty in the problem inputs given by inaccuracies in range measurement.

Artificial bee colony has been applied to solve the economic load dispatch problem with non-smooth cost functions (Hemamalini and Simon, 2010). Particle swarm optimization (PSO) has also been used to solve the economic dispatch problem (Neyestani et al., 2010; Basu, 2015; Niknam et al., 2011). An application of PSO to a non-smooth manpower assignment problem is illustrated in Yang and Chou (2011). Similarly, ant colony optimization (ACO) and PSO algorithms have been used to solve non-smooth optimal power flow problems in Vaisakh and Srinivas (2011) and Vaisakh et al. (2013), respectively. Finally, PSO has also been used to solve non-smooth portfolio selection problems in Corazza et al. (2013), while an extended ACO for non-convex mixed integer non-linear programming is introduced in Schlüter et al. (2009). Likewise, a heuristic to solve non-convex OPs is described in Toscano and Lyonnet (2010).

In the transportation and logistics field, there are multiple applications focusing on rich and real-life variants of the well-known vehicle routing problem (VRP)(Caceres-Cruz et al., 2014). Some studies consider realistic VRPs with soft constraints (Hashimoto et al., 2006), including soft time-windows and soft traveling-time constraints. In most real-life applications, soft constraints are more likely to occur than hard ones, i.e., it might be possible to violate them to some degree, although constraint violation might imply a specific penalty cost. The same idea can be applied to traveling time, e.g., sometimes traveling times can be shortened if paying for a turnpike toll or a faster transportation mode. The relaxation of these constraints might naturally lead to a problem formulation, which includes a more realistic, non-convex or non-smooth, objective function. Some VRP extensions refer to the inclusion of soft time windows, delivery and pickup operations, split deliveries, etc., which can be formulated as non-convex or non-smooth OPs (Derigs and Metz, 1992; Derigs et al., 2010). In particular, a local-search method to solve the split delivery VRP is presented in Derigs et al. (2010). Likewise, a non-smooth formulation for the VRP is provided in Juan et al. (2011), where a biased-randomized solving algorithm is proposed. In a similar way, Ferrer et al. (2016) propose a metaheuristic algorithm for solving a non-smooth scheduling problem with penalty costs generated by machine failures.

3. Related work on the ARP

Since ARPs are gaining attention among the research community, several authors have published surveys on the topic (Dror, 2000; Hertz, 2005; Wøhlk, 2008; Corberan and Prins, 2010; Corberan and Laporte, 2014). Among the exact methods, branch-and-bound and cutting plane are probably the most common approaches in the ARP literature. Both are based on integer linear programming (ILP) formulations. In fact, the ARP was first formulated as an ILP by Golden and Wong (1981). A number of exact approaches have been developed since then (Belenguer et al., 2014), including graph based branch-and-bound methods (Hirabayashi et al., 1992a,b), cutting plane approaches (Belenguer and Benavent, 2003), as well as branch-and-cut and price-based algorithms (Letchford and Oukil, 2009). Some of these techniques offer lower bounds to the problem, even if an optimal solution cannot be obtained in a reasonable computing time. Other lower bounding procedures are commented in Ahr and Reinelt (2014). Typically, exact methods can only be used to solve small- and medium-size ARP instances (Bode and Irnich, 2015).

Problem-specific heuristics have been a common method for providing reasonably good solutions to ARPs in extremely short computing times (just a few seconds). Classical ARP heuristics consider vehicle allocation and route sequence simultaneously. They include the construct-strike procedure, the path-scanning method, and the augment-merge method (Golden et al., 1983; Dror, 2000). The performance of these classical procedures is generally 10% to 20% worse than the best-known solutions (Wøhlk, 2008). Pearn (1989) proposed improved versions of some heuristics by adding randomness, while Pearn (1991) introduced the augment-insert algorithm. Also, Wøhlk (2005) proposed the double-outer-scan heuristic, which combines the augment-merge and the path-scanning methods. Later, Gonzalez et al. (2012) introduced the SHARP heuristic, which was able to outperform most of the previously published ones. Prins (2014) reviews both heuristic and metaheuristic approaches for the ARP.

In recent years, most approaches for solving ARP variants are based on metaheuristics. For instance, several tabu search algorithms have been proposed to solve the ARP. One of the first ones, CARPET, was introduced in Hertz et al. (2000). In this algorithm, infeasible solutions are allowed but also penalized. A deterministic tabu search algorithm has been suggested by Brando and Eglese (2008). Their algorithm penalizes infeasible solutions in the objective function and alternates between different neighborhood structures. Also, a combination of tabu search and scatter search was proposed in Greistorfer (2003). Lacomme presented both a genetic algorithm (Lacomme et al., 2001) and a memetic algorithm (Lacomme et al., 2004). The crossover operations were performed on a giant tour, and fitness of a chromosome was based on the partitioning of the tour into vehicle tours. Currently these algorithms are among the very best performing approaches to the ARP. Other approaches for the ARP are related to simulated annealing algorithms. For instance, Li (1992) combined this metaheuristic with tabu search to a road gritting problem. Likewise, Eglese (1994) designed a simulated annealing approach for a multi-depot gritting problem with side constraints. Finally, Wøhlk (2005) suggested a simulated annealing algorithm for the classical ARP. In his approach, the order of the edges on a giant tour is changed during the algorithm, and the optimal partitioning of the tour is computed at each iteration. A younger generation of metaheuristics are those based on ant colony systems, proposed by Lacomme et al. (2004) and Doerner et al. (2003). In the former, the authors reported competitive results in terms of quality of the solution, although employing longer computing times. A guided local search algorithm has been presented for the ARP by Beullens et al. (2003). In this work, the distance of each arc is penalized according to some function, which is adjusted throughout the numerical experiments. A variable neighborhood descent algorithm was proposed by Hertz and Mittaz (2001). Recently, a hybrid algorithm for the ARP has been introduced in Chen et al. (2016). This algorithm is extremely efficient, both in terms of the quality of the solutions it provide as well as in terms of computing times. Finally, regarding non-smooth ARP variants, Gonzalez et al. (2014) presented the non-smooth version of the ARP and proposed a multi-start approach to solve it. They test its performance over a reduced set of ARP instances. In this work, we significantly enhance and extend their approach by introducing a formal model of the problem, a novel metaheuristic framework, and an extensive set of computational experiments, which also include the use of CPLEX to solve a truncated version of the problem.

Table 3 summarizes the main strengths and weaknesses of the works mentioned above. More details can be found in the reviews of Bode and Irnich (2015) and Prins (2014). The first part of the table corresponds to exact methods, which could only be used to solve small- and medium-size instances while requiring large computational times. The second part of the table corresponds to heuristics, which could provide quality solutions for large

Table 1: Analysis of related works' strengths and weaknesses

Reference	Strengths (●) and weaknesses (○)
	Exact methods
Hirabayashi et al. (1992a,b)	● First algorithm in the literature addressing the capacitated ARP in an exact way ○ Instances are not online available, and no further details are given by the authors on how these instances were generated
Belenguer and Benavent (2003)	● The method outperforms all the existing lower bounding procedures for the capacitated ARP until 2003 ○ The method allows symmetric solutions. This fact causes serious problems in the branch-and-cut algorithm ○ The method only calculates lower bounds
Letchford and Oukil (2009)	● First time that the fundamental idea of exploiting the sparsity was coined ○ Running times are excessive for the val and egl instances
Ahr and Reinelt (2014)	● The method is able to detect violated capacity constraints that were not found by the heuristic separation routines of Belenguer and Benavent (2003) ○ The method only calculates lower bounds
	Heuristics
Golden et al. (1983)	● The method is flexible, the heuristic can be adapted to more complex networks ○ Poor performance compared to later heuristics (10% to 20% worse than the best-known solutions)
Pearn (1989)	● The method outperforms the existing heuristics until 1989 ○ The method is only tested with dense networks considering small arc demands
Pearn (1991)	● The heuristic structure is easy to implement ● The method outperforms existing heuristics, i.e., Construct-Strike, Path Scanning, Augment Merge, Modified-Construct-Strike, and Random Path Scanning ○ The method is only tested with sparse networks considering some large arc demands
Wøhlk (2005)	● The method improves the Path-Scanning heuristic ○ The method is able to provide some optimal solutions
Gonzalez et al. (2012)	● The method requires few parameters ● The method outperforms most Path Scanning heuristic results ○ Results are only compared with the Path Scanning heuristic results
	Metaheuristics
Hertz et al. (2000)	● The method outperforms all known heuristics until 2000 and often produces a proven optimum ● Only one single parameter setting is needed
Hertz and Mittaz (2001)	● The method provides small improvements over Hertz et al. (2000) requiring less computational time
Lacomme et al. (2001)	● The method provides improvements over Hertz et al. (2000) and Hertz and Mittaz (2001) ● Only one single parameter setting is needed ○ The computational performance in terms of running time is not provided
Greistorfer (2003)	● Merits of explicitly adding population components to a tabu search ● The method is competitive in terms of quality and running time
Doerner et al. (2003)	● Results are superior to those reported Greistorfer (2003) and Hertz et al. (2000) ○ The method is tested on few instances
Beullens et al. (2003)	● First algorithm to find definitive solution values for the gdb instances ● The method provides fast solutions, in the range of a few seconds
Lacomme et al. (2004)	● Only one single parameter setting is needed ● The method outperforms results by Doerner et al. (2003) ● The algorithm is already designed for tackling several extensions like mixed networks, parallel arcs and turn penalties
Wøhlk (2005)	● Innovative metaheuristic combining Dynamic Programming with Simulated Annealing ○ The method does not outperforms previous ones, although is competitive for some set of instances
Brando and Eglese (2008)	● The method provides high quality results within a reasonable computing time ● Some new best solutions are reported for a set of test problems used in the literature
Gonzalez et al. (2014)	● First and unique work addressing the non-smooth ARP up to now ○ The algorithm is tested on a reduced number of instances
Chen et al. (2016)	● The method is extremely efficient in terms of computing times ● The method outperforms previous works tackling the ARP

instances within reasonable computation times. Finally, the last part of the table corresponds to metaheuristics, which provide high quality solutions in short times. As can be observed, only Gonzalez et al. (2014) addresses the non-smooth version of the ARP.

A different strategy for solving the ARP is based on transforming the problem into a VRP. Then, the ARP can be solved by state-of-the-art VRP algorithms. A general transformation of the problem, using 3 nodes on each arc, was described by Pearn et al. (1987). VRP-derived approaches include Eglese and Letchford (2000), Longo et al. (2006), and Baldacci and Maniezzo (2006). Results reported in those works are highly competitive compared with the previously existing ones, which may reflect the comparatively more developed nature of VRP techniques. Likewise, methods providing lower bounds for the ARP were put forward by Pearn et al. (1987), Pearn (1988), Li (1882), Benavent et al. (1992), Welz (1994), Amberg et al. (2000), and Wøhlk (2006). There also exists a problem half-way between the ARP and the VRP, called the stringed VRP (Oppen and Lkketangen, 2006). In this problem, customers are to be served as in the VRP, but some of these customers are located along the arcs. Stochastic versions of the ARP have been also considered. Thus, for instance, Gonzalez et al. (2016) present a simulation-optimization approach for solving a ARP variant in which customers demands are modeled as random variables. Their solving method is based on the concept of 'simheuristics' (Juan et al., 2015a). The authors also provide a survey on similar works regarding stochastic versions of the ARP.

4. Modeling the non-smooth ARP

The goal of this section is twofold, on the one hand, it presents a formal and high-level description of the NS-ARP and, on the other hand, it introduces a more classical model that is used during the numerical experiments to obtain lower-bounds to some instances.

4.1. A novel mathematical model for the NS-ARP

We consider a connected and undirected graph, $G := (V, E)$, where V is a set of vertices and E is a set of edges connecting some of these vertices. Also, there is a finite and homogeneous fleet of $|T|$ vehicles indexed in a set T, each of them with capacity $Q \geq 0$. Each edge $e = \{i, j\} \in E$, considers a demand $d_{ij} \geq 0$ and a traversing cost $c_{ij} \geq 0$. Edges with positive demand form the subset $R \subseteq E$. The goal is to find a minimal-cost complete system of routes for the vehicles such that each edge with positive demand is serviced. In particular, the following constraints need to be satisfied:

(i) edges with a positive demand must be serviced by exactly one vehicle

(ii) all edges can be traversed multiple times

(iii) each vehicle starts and finishes its route at the depot

(iv) the total demand handled in any route is limited by Q

In graph theory, a direct path ρ of length r is a sequence of $r+1$ nodes, $\{i_0, i_1, \ldots, i_r\}$, together with a sequence of r arcs, $\{a^1, a^2, \ldots, a^r\}$, such that:

$$a^k = (i_{k-1}, i_k) \qquad k = 1, 2, \ldots, r$$

We will denote a direct path, ρ, as follows:

$$\rho : \; i_0 \to i_1 \to i_2 \to \cdots \to i_{r-1} \to i_r$$

where the node i_0 is called the start node and the node i_r is called the end node of the path. Both of them are called terminal nodes of the path. The other nodes in the path are called *internal nodes*. A finite cycle is a path such that the start and the end nodes are the same. A path with no repeated nodes is called a simple path, and a cycle with no repeated internal nodes is called a simple cycle or tour. In our context, the set V includes a *depot* node, represented by the node 0. Also, a *route* ρ of order r is a tour of length $r+2$ in which both the start node and the end node are the depot:

$$\rho : \; 0 \to i_1 \to i_2 \to \cdots \to i_{r-1} \to i_r \to 0$$

Based on the above notation, the cost of a route ρ for a given vehicle $k \in T$ is then given by:

$$c_\rho = \sum_{(i,j) \in \rho} c_{ij}$$

While the total demand handled by a vehicle $k \in T$, q_k, is given by:

$$q_k = \sum_{(i,j) \in \mathcal{A}} q_{ij} x_{ij}^k$$

In the NS-ARP, we contemplate the possibility of relaxing some of the constraints, i.e., we considered soft constraints instead of hard ones. As it often happens in real-life (Hashimoto et al., 2006), some constraints can be violated by incurring in well-defined penalty costs that must be added to the objective function. In our case, we will allow routes to exceed the maximum route cost –which could either a distance- or a time-based cost. In practice, if a given route exceeds a threshold cost, then some overhead must be added to the total route cost. This overhead is likely to be defined as a piecewise non-smooth function, i.e., the specific penalty to apply will depend (in a discontinuous way) on the size of the gap between the actual route cost and the threshold. Thus, we can generalize the cost function of any route, c^*_ρ as follows:

$$c^*_\rho := \begin{cases} c_\rho & \text{if } c_\rho \leq C, \\ \\ c_\rho + \lambda\left(c_\rho, C\right) & \text{otherwise.} \end{cases} \tag{1}$$

Where $\lambda\left(c_\rho,\ C\right)$ is a non-smooth function which will be applied whenever the actual route cost exceeds the maximum 'recommended' cost per route, C. Accordingly, the optimization problem can be expressed as follows, where CSR represents a complete set of routes:

$$\begin{array}{ll} \text{minimize} & \sum\limits_{\rho \in \mathcal{S}} c^*_\rho, \\ \text{subject to:} & \mathcal{S} \in CSR \\ & \sum_{(i,j)\in\rho} q_{ij} x^k_{ij} \leq Q \quad \forall\ \rho \in \mathcal{S}, \forall\ k \in T, \\ & x^k_{ij}(1 - x^k_{ij}) = 0 \quad \forall\ (i,j) \in \rho,\ \forall\ \rho \in \mathcal{S}, \forall\ k \in T. \end{array} \tag{2}$$

During the experimental section of this article, some specific numerical values have been assigned to the aforementioned penalty term. Thus, without loss of generality and with the purpose of testing our approach, the computational experiments carried out in this work will assume the following specific non-smooth expression for the penalty factor:

$$\lambda\left(c_\rho, C\right) = \min\left\{\theta\left(c_\rho, C\right), P\right\} \tag{3}$$

where $P > 0$ is an upper bound for the penalty cost and

$$\theta\left(c_\rho, C\right) = 5 + 5000\left(\frac{c_\rho - C}{c_\rho}\right)^4 \tag{4}$$

Notice that the coefficients of the function (4) are just a 'random' choice to illustrate our methodology with a numerical example of a non-smooth penalty function for the ARP problem. Nevertheless, other values and functions could be considered within the NS-ARP to address other scenarios and real-world cases.

4.2. A more classical MIP formulation

In order to assess the performance of approximate methods, a MIP formulation is also used in this paper. The model is a truncated version of the one proposed for the classical ARP in Baldacci and Maniezzo (2006). Those authors present a reformulation of the problem as a VRP. In our case, the penalty factor is linearized. This allow us to generate some bounds that are helpful in order to evaluate the quality of our algorithmic approach. In the proposed transformation, a graph $G' = (V', E')$ is considered, where the nodes $V' = \{0, 1, ..., 2|R|\}$ correspond to the endpoint nodes of the required edges plus the depot node 0. Also, the set of edges E' is defined such in a way that G' is completely connected.

With the aim of making this paper self-contained, the MIP model is described next. Let $V'' = V'/\{0\}$. Let $\Pi = \{S : S \subseteq V'', |S| \geq 2\}$. For a given $S \in \Pi$, we denote by $\overline{S}$ the complementary set of nodes V'/S. Let $r(S)$ be the minimum number of vehicles of capacity Q needed to satisfy the demand of customers in $S \in \Pi$. Also, $q'(S)$ is used to indicate the total demand of the node subset $S \subseteq V'$, i.e., $q'(S) = \sum_{i \in S} q'_i$ — note that q'_i is the transformed demand according to Baldacci and Maniezzo (2006). Similarly, the transformation of the costs c'_{ij} was followed in the same manner as in described in Baldacci and Maniezzo (2006) also including the truncated penalty function. Finally, let x_{ij} be an integer variable representing the number of times a vehicle travels between vertices i and j. Now, the resulting CVRP can be formulated as follows:

$$minimize \sum_{\{i,j\} \in E'} c'_{ij} x_{ij} \tag{5}$$

subject to:

$$\sum_{j \in V', i<j} x_{ij} + \sum_{j \in V', i>j} x_{ji} = 2, \forall i \in V'' \tag{6}$$

$$\sum_{i \in S} \sum_{j \in \overline{S}, i<j} x_{ij} + \sum_{i \in \overline{S}} \sum_{j \in S, i<j} x_{ij} \geq 2r(S), \forall S \in \Pi \tag{7}$$

$$\sum_{j \in V''} x_{0j} = 2|T| \tag{8}$$

$$x_{ij} \in \{0,1\}, \forall \{i,j\} \in E'/\{\{0,j\} : j \in V''\} \tag{9}$$

$$x_{0j} \in \{0,1,2\}, \forall \{0,j\} \in E', j \in V'' \tag{10}$$

Constraints (6) are the degree constraints for each customer. Constraints (7) are the capacity constraints (also called generalized subtour elimination constraints). For any subset S of customers, these constraints impose that $r(S)$ vehicles enter and leave S. It should be noted that the formulation remains valid if $r(S)$ is replaced by a lower bound on its value, such as $\lceil q'(s)/Q) \rceil$. Constraints (8) state that $|T|$ vehicles must leave and return to the depot while constraints (9) and (10) are the integrality constraints.

5. A BRILS approach for the NS-ARP

Our solving methodology for the NS-ARP is denoted as BRILS, since it combines a biased-randomized (BR) procedure with an iterated local search (ILS) metaheuristic. ILS is a well-tested framework that offers a balanced combination of efficiency and relative simplicity (Juan et al., 2014b; Dominguez et al., 2014; Juan et al., 2015b; Dominguez et al., 2016). Moreover, it can be easily extended to a simheuristic algorithm (Grasas et al., 2016), which can be specially useful when considering routing problems with stochastic demands or variable transportation costs (Gonzalez et al., 2016; Juan et al., 2014a). Algorithm 1 depicts our approach. It starts by generating an initial solution. Then, a searching procedure is repeated until a termination condition is reached. This procedure is composed of three main phases: a perturbation stage (which systematically changes the solution in order to start from different points of the search space), a local search stage (which aims at improving the solution by searching in its local neighborhood), and an acceptance stage (which helps the algorithm to avoid getting trapped in a local minimum). Each of these phases is described next in more detail:

- *Initial solution:* In order to generate an initial solution, we make use of the RandSHARP probabilistic algorithm (line 1). This procedure introduces a biased-randomized behavior into a savings-based heuristic, as described in Gonzalez et al. (2012).
- *Perturbation:* The procedure used to perturbate the base solution (line 6) is a destruction-reconstruction strategy. Thus, given a base solution, we select a random number of required edges and 'destroy' (delete) the routes serving them. Then, we reconstruct those routes by applying

Algorithm 1: Biased-Rand ILS algorithm

```
1  initSol ← getInitialSolution() % biased randomization
2  baseSol ← initSol
3  bestSol ← baseSol
4  t ← initialTemperature
5  while ((t > 0) and (time < limitTime)) do
6      newSol ← perturbate(baseSol) % biased randomization
7      newSol ← localSearch(newSol) % cache memory
8      delta ← cost(newSol) − cost(baseSol)
9      t ← t - alpha1 * random(0,1)
       % acceptance criterion
10     if (delta < 0) then
11         t ← t + delta * alpha2 * random(0,1)
12         baseSol ← newSol
13         if (cost(newSol) < cost(bestSol)) then
14             bestSol ← newSol
           end
       else
15         if (random(0,1) < $e^{(delta/t)}$) then
16             baseSol ← newSol
           end
       end
   end
17 return bestSol
```

the biased-randomized RandSHARP procedure on them. After that, a new solution emerges (Figure 3).

- *Local search:* Our local search stage (line 7) is based on the use of a growing and continuously updated memory cache: for each set of edges in a newly generated route, the best-found-so-far way of covering them is stored in a hash-map data structure; the information in this cache memory is quickly recovered / updated each time a cluster of edges is selected, so that newly generated routes are quickly improved as more information is stored in memory. Figure 4 shows an example of improvement achieved through the use of the cache memory. In

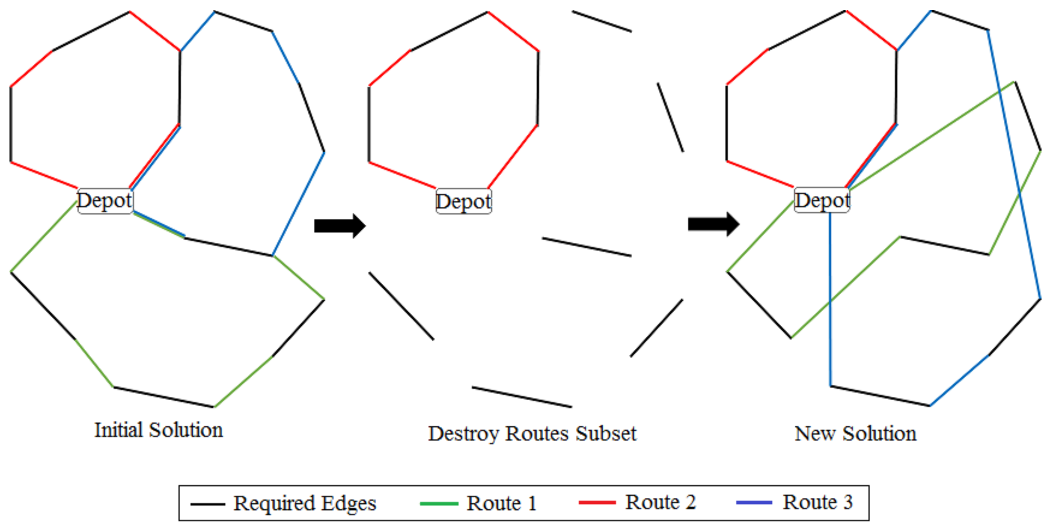

Figure 3: Perturbation

this example, it is assumed that the first solution has been obtained in previous iterations and that the way arcs a, b, and c are visited has been stored in the cache. If, during the current iteration, the second solution is obtained before the local search stage, then the cache is checked in order to improve the individual routes, obtaining the third solution.

- *Acceptance criterion:* Every time a new solution is generated after the perturbation and local search stages, the algorithm must decide whether or not the base solution and the best solution have to be updated. Even in those cases in which the new solution does not outperforms the base solution, it might be convenient to update the latter to avoid getting trapped in a local minimum. To regulate this process, a simulated annealing procedure is employed (Kirkpatrick et al., 1983). Firstly, the temperature is initialized (line 4), and then it is reduced at each iteration using the *alpha* parameters (line 9 and line 11). Thus, whenever a given probabilistic criterion is satisfied (line 15), the current base solution is updated (line 16).

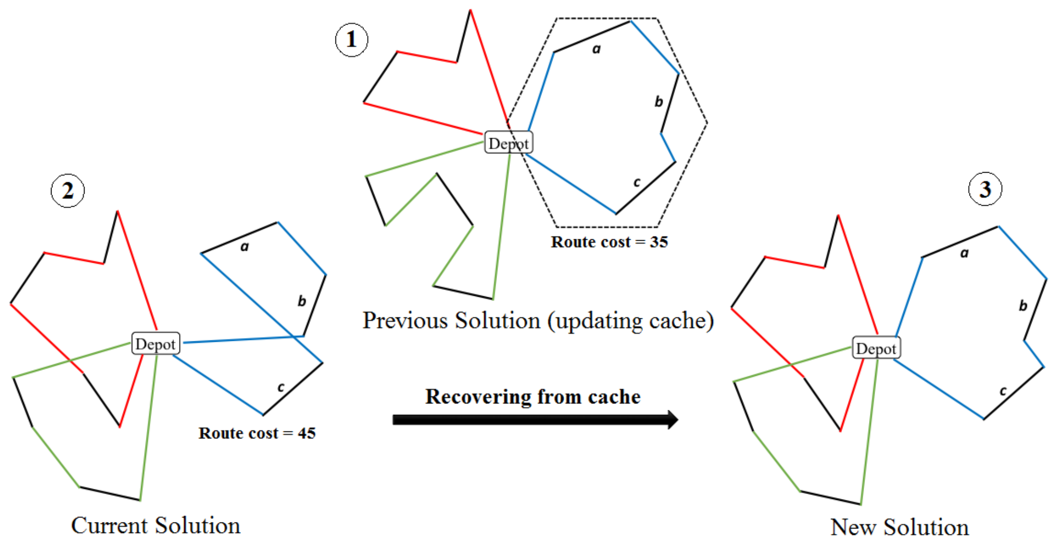

Figure 4: Example of improvement through Local Search

6. Computational experiments

The aim of this section is twofold: on the one hand, it validates the efficiency of the proposed algorithm for solving the proposed NS-ARP; on the other hand, it illustrates the benefits of considering more realistic models including non-smooth constraints. The numerical experiments are divided in two parts:

- Firstly, we compare the results obtained by our BRILS approach for the NS-ARP with soft constraints with the ones provided by the multi-start approach introduced in Gonzalez et al. (2014), as well as with the best-known solutions (BKS) for the classical ARP with hard constraints. In the latter, a maximum route cost is not considered. Additionally, we also consider the scenario with hard constraints –i.e., avoiding exceeding the route cost threshold– with the purpose of illustrating how total costs increase in comparison with the soft-constraints scenario.

- Secondly, we truncate the penalty function in order to make it linear and, thus, make it possible to use CPLEX to solve the truncated version of the NS-ARP. This way, we can obtain bounds for each instance, which contribute to measure the quality of our approach.

The BRILS algorithm has been implemented using Java SE 7 over the Netbeans IDE. All tests were run using a standard personal computer, equiped with an Intel Core2 Quad CPU Q9300 @ 2:50 GHz, 16 GB of RAM, and a Windows 7 Pro operating system. A set of classical benchmarks –traditionally used for the standard ARP– has been adapted by adding soft constraints as well as a penalty term –as described in Section 4– in the objective function. In particular, four different benchmarks have been used. These benchmarks are available at `http://www.uv.es/belengue/carp.html`. They correspond to: *(a)* the *gdb* dataset from Golden et al. (1983); *(b)* the *egl* dataset from Leon Y. O. Li (1996); *(c)* the *val* dataset proposed in Benavent et al. (1992); and *(d)* the *kshs* dataset proposed by Kiuchi et al. (1995). Table 3 summarizes the main characteristics of these sets, which are described next:

- The *egl* dataset consists of 24 instances which are derived from real world data. This data refers to winter gritting in the county of Lancashire, United Kingdom. These instances contain both required and not required arcs, so there are also arcs without associated demand. The instances are grouped in two different network configurations with very low arc density. The density of arcs is computed as the percentage of arcs of the complete graph which conform the network of the given instance.

- The *gdb* dataset consists of 23 instances of small and medium-size –between 10 and 50 arcs– with a mixture of dense and sparse graph networks. The instances are characterized by their number of nodes, number of arcs, number of required arcs, and the density of arcs on the network. These instances are artificially generated and contain only arcs with associated demand.

- The *kshs* dataset instances consist of 6 instances of small size and a medium- to high-density of arcs. They are artificially generated and all the instances have the same number of arcs. All of the arcs are required as they have an associated demand.

- Finally, the *val* dataset consists of 34 instances, modeled on 10 sparse graph networks, with varying vehicle capacities for each network. Thus, the same instance *valX* is to be solved with different capacity constraints, obtaining the variants *valXY* which conform the final instances. These instances include only required arcs and are networks with low arc density (around 10%).

Set	Instances	Nodes	Arcs	Density
egl	24	109	144	2.65%
gdb	23	12	29	53.77%
kshs	6	8	15	55.95%
val	34	36	63	10.59%

Table 3: Base datasets for the ARP

In order to adapt these instances to the NS-ARP, we have introduced a threshold parameter C_{max} which determines the maximum route cost allowed (or 'recommended' in the case of soft constraints). In our tests, this parameter has been defined as the average route cost for each ARP instance (rounded to a multiple of 10), considering the results obtained with our metaheuristic for the standard ARP. When this value is lower than the maximum cost of visiting a single required edge using the shortest path, the parameter takes this cost value. Instead of considering this parameter as a hard constraint, we have considered it as a soft one that could eventually be violated by assuming a penalty cost. Following the penalty function (1), in our tests we have used the specific non-linear function (3). The parameters in this function depend on the cost structure of each instance. In this case, the maximum route penalty P_{max} due to exceeding C_{max} can go up to a 10% of the cost of a route when hard constraints are considered.

Regarding the parameters setting of our algorithm, after some quick tests the initial temperature for the acceptance criterion has been set to 1.5, and the decreasing factors $alpha1$ and $alpha2$ have been set to 0.01 and 0.002, respectively. Finally, the *beta* parameter associated with the biased-randomization process takes random values in the interval [0.1, 0.3].

As mentioned before, in our computational experiments we have solved the ARP with maximum cost per route both assuming soft and hard constraints. When considering soft constraints, the objective function becomes non-smooth due to the introduction of penalty costs. Moreover, in order to test the efficiency of our BRILS approach, we compare its results with the results obtained using the multi-start (MS) approach introduced in Gonzalez et al. (2014). Tables 4 - 11 show the results obtained for each of the described settings. These tables are structured in several parts. The first one displays information about every problem dataset, i.e., the name of the instance, the maximum route cost allowed (C_{max}), the maximum penalty allowed (P_{max}), and the BKS for the original ARP problem instance as given in Chen et al.

(2016). Since these BKS do not consider a maximum cost per route, they will act as a lower-bound for the NS-ARP considered here. The second part of each table shows the results obtained using the MS algorithm, while the third part shows the results obtained using our BRILS algorithm. For each approach, our best and average solutions obtained after 10 executions of each instance (with a maximum running time of 180 seconds per execution) are depicted. The gap of these results with respect to the BKS is also provided.

6.1. Evaluation of the Solving Approaches for the NS-ARP

In order to assess the performance of our algorithm, Tables 4, 6, 8, and 10 report the solutions obtained for the NS-ARP when considering soft-constraints (*i.e.*, non-smooth objective functions with penalty costs). Similarly, Tables 5, 7, 9, and 11 show the solutions obtained when considering hard-constraints. Table 12 summarizes the gaps obtained for each approach. Notice that there is an appreciable difference between considering hard and soft constraints: in the latter case, solutions for the ARP with a maximum cost per route are relatively close to the BKS for the standard ARP. In the former case, solutions are much worse, since the existence of a hard limitation on the maximum distance requires the use of additional routes.

Besides, notice that our BRILS algorithm improves the results provided by the MS method proposed in Gonzalez et al. (2014), both for the scenario with soft constraints as well as for the one with hard constraints. Figure 5 shows a multiple boxplot offering a visual comparison of the algorithms performance. Notice that: *(i)* our BRILS approach clearly outperforms the MS approach; and *(ii)* as expected, using soft constraints allows to obtain better results than using hard constraints (i.e., the solution gets closer to the BKS for the standard ARP, which does not impose a maximum cost per route). Also, as depicted in Figure 6, the Fisher test for differences between pairs of means shows significant differences between the BRILS and MS approaches, as well as between the scenarios using hard and soft constraints.

Table 4: Results for the NS-ARP with soft constraints - Egl (BKS refers to the standard ARP)

Instance	C_{max}	P_{max}	**BKS**	**MS**				**BRILS**			
				Best	**Gap (%)**	**Avg**	**Gap (%)**	**Best**	**Gap (%)**	**Avg**	**Gap (%)**
			(1)	(2)	(1)-(2)	(3)	(1)-(3)	(4)	(1)-(4)	(5)	(1)-(5)
Egle e1-A	900	70	3548	3553.02	0.14	3553.02	0.14	3553.02	0.14	3553.02	0.14
Egle-e1-B	900	70	4498	4516.00	0.40	4516.00	0.40	4498.00	0.00	4507.80	0.22
Egle-e1-C	900	70	5595	5632.00	0.66	5632.00	0.66	5613.00	0.32	5613.00	0.32
Egle-e2-A	900	70	5018	5027.05	0.18	5052.34	0.68	5023.05	0.10	5023.05	0.10
Egle-e2-B	900	70	6317	6364.00	0.74	6371.20	0.86	6344.00	0.43	6353.20	0.57
Egle-e2-C	900	70	8335	8481.00	1.75	8518.60	2.20	8354.00	0.23	8392.70	0.69
Egle-e3-A	900	70	5898	5949.23	0.87	5967.58	1.18	5903.05	0.09	5908.52	0.18
Egle-e3-B	900	70	7775	7860.00	1.09	7883.90	1.40	7799.00	0.31	7807.50	0.42
Egle-e3-C	900	70	10292	10463.00	1.66	10499.90	2.02	10343.00	0.50	10362.20	0.68
Egle-e4-A	900	70	6444	6508.00	0.99	6530.41	1.34	6481.01	0.57	6491.41	0.74
Egle-e4-B	900	70	8961	9178.00	2.42	9211.53	2.80	9040.00	0.88	9065.30	1.16
Egle-e4-C	900	70	11529	11843.00	2.72	11888.80	3.12	11733.00	1.77	11787.10	2.24
Egle-s1-A	1000	70	5018	5028.00	0.20	5043.44	0.51	5028.00	0.20	5028.00	0.20
Egle-s1-B	1000	70	6388	6439.00	0.80	6446.82	0.92	6414.00	0.41	6434.41	0.73
Egle-s1-C	1000	70	8518	8531.00	0.15	8534.40	0.19	8523.00	0.06	8523.00	0.06
Egles2-A	1100	70	9874	10054.00	1.82	10100.21	2.29	9987.00	1.14	10012.60	1.40
Egle-s2-B	1000	70	13078	13443.16	2.79	13479.19	3.07	13285.05	1.58	13319.68	1.85
Egle-s2-C	1000	70	16425	16734.01	1.88	16788.91	2.22	16658.01	1.42	16697.51	1.66
Egle-s3-A	1100	70	10201	10484.00	2.77	10513.60	3.06	10376.00	1.72	10418.50	2.13
Egle-s3-B	1000	70	13682	13979.05	2.17	14035.51	2.58	13855.05	1.26	13909.87	1.67
Egle-s3-C	1000	70	17188	17560.01	2.16	17624.31	2.54	17371.01	1.06	17416.91	1.33
Egle-s4-A	1100	70	12216	12629.00	3.38	12680.60	3.80	12464.00	2.03	12535.70	2.62
Egle-s4-B	1100	70	16201	16700.00	3.08	16738.00	3.31	16513.00	1.93	16568.90	2.27
Egle-s4-C	1100	70	20476	21060.00	2.85	21104.30	3.07	20902.00	2.08	20959.90	2.36
Average					1.57		1.85		0.84		1.07

Table 5: Results for the ARP with hard constraints - Egl (BKS refers to the standard ARP)

Instance	C_{max}	P_{max}	**BKS**	**MS**				**BRILS**			
				Best	**Gap (%)**	**Avg**	**Gap (%)**	**Best**	**Gap (%)**	**Avg**	**Gap (%)**
			(1)	(2)	(1)-(2)	(3)	(1)-(3)	(4)	(1)-(4)	(5)	(1)-(5)
Egle e1-A	900	70	3548	3568	0.56	3568.00	0.56	3568	0.56	3568	0.56
Egle-e1-B	900	70	4498	4516	0.40	4516.00	0.40	4498	0.00	4509.90	0.26
Egle-e1-C	900	70	5595	5632	0.66	5632.00	0.66	5613	0.32	5613.40	0.33
Egle-e2-A	900	70	5018	5135	2.33	5150.30	2.64	5132	2.27	5134.40	2.32
Egle-e2-B	900	70	6317	6370	0.84	6377.30	0.95	6347	0.47	6354.80	0.60
Egle-e2-C	900	70	8335	8506	2.05	8532.00	2.36	8354	0.23	8395.80	0.73
Egle-e3-A	900	70	5898	5983	1.44	6023.50	2.13	5938	0.68	5947.20	0.83
Egle-e3-B	900	70	7775	7867	1.18	7901.00	1.62	7799	0.31	7812.70	0.48
Egle-e3-C	900	70	10292	10476	1.79	10512.90	2.15	10338	0.45	10376.30	0.82
Egle-e4-A	900	70	6444	6516	1.12	6548.50	1.62	6495	0.79	6502.00	0.90
Egle-e4-B	900	70	8961	9182	2.47	9225.50	2.95	9004	0.48	9080.00	1.33
Egle-e4-C	900	70	11529	11864	2.91	11916.10	3.36	11791	2.27	11805.90	2.40
Egle-s1-A	1000	70	5018	5164	2.91	5169.40	3.02	5164	2.91	5164.00	2.91
Egle-s1-B	1000	70	6388	6455	1.05	6473.00	1.33	6435	0.74	6438.00	0.78
Egle-s1-C	1000	70	8518	8560	0.49	8566.00	0.56	8530	0.14	8533.80	0.19
Egles2-A	1100	70	9874	10078	2.07	10117.90	2.47	9997	1.25	10022.90	1.51
Egle-s2-B	1000	70	13078	13598	3.98	13651.60	4.39	13434	2.72	13493.00	3.17
Egle-s2-C	1000	70	16425	16869	2.70	16891.00	2.84	16753	2.00	16781.70	2.17
Egle-s3-A	1100	70	10201	10475	2.69	10539.30	3.32	10417	2.12	10467.20	2.61
Egle-s3-B	1000	70	13682	14174	3.60	14255.30	4.19	14095	3.02	14134.70	3.31
Egle-s3-C	1000	70	17188	17698	2.97	17790.50	3.51	17473	1.66	17617.80	2.50
Egle-s4-A	1100	70	12216	12651	3.56	12685.20	3.84	12488	2.23	12588.00	3.05
Egle-s4-B	1100	70	16201	16633	2.37	16738.70	3.32	16548	2.14	16599.80	2.46
Egle-s4-C	1100	70	20476	21054	2.82	21105.80	3.08	20947	2.30	20973.00	2.43
Average					2.05		2.39		1.34		1.61

Table 6: Results for the NS-ARP with soft constraints - gdb (BKS refers to the standard ARP)

Instance	C_{max}	P_{max}	**BKS**	**MS**				**BRILS**			
				Best	**Gap (%)**	**Avg**	**Gap (%)**	**Best**	**Gap (%)**	**Avg**	**Gap (%)**
			(1)	(2)	(1)-(2)	(3)	(1)-(3)	(4)	(1)-(4)	(5)	(1)-(5)
gdb1	70	5	316	321	1.58	321.00	1.58	321	1.58	321.00	1.58
gdb2	60	5	339	349	2.95	352.00	3.83	344	1.47	346.20	2.12
gdb3	60	5	275	284	3.27	284.00	3.27	280	1.82	280.00	1.82
gdb4	70	5	287	292	1.74	292.00	1.74	292	1.74	292.00	1.74
gdb5	70	5	377	387	2.65	387.00	2.65	382	1.33	382.00	1.33
gdb6	70	5	298	303	1.68	303.00	1.68	303	1.68	303.00	1.68
gdb7	60	5	325	330	1.54	330.00	1.54	330	1.54	330.00	1.54
gdb8	40	5	348	361	3.74	361.60	3.91	353	1.44	356.60	2.47
gdb9	40	5	303	320	5.61	321.70	6.17	312	2.97	314.80	3.89
gdb10	60	5	275	285	3.64	286.20	4.07	280	1.82	280.20	1.89
gdb11	80	5	395	404	2.28	405.30	2.61	400	1.27	400.00	1.27
gdb12	100	5	458	473	3.28	477.40	4.24	458	0.00	458.50	0.11
gdb13	130	5	536	544	1.49	544.00	1.49	541	0.93	543.70	1.44
gdb14	20	5	100	109	9.00	109.00	9.00	105	5.00	105.00	5.00
gdb15	20	5	58	58	0.00	58.00	0.00	58	0.00	58.00	0.00
gdb16	30	5	127	129	1.57	129.00	1.57	127	0.00	127.00	0.00
gdb17	20	5	91	91	0.00	91.00	0.00	91	000	91.00	0.00
gdb18	30	5	164	172	4.88	173.60	5.85	169	3.05	169.00	3.05
gdb19	20	5	55	60	9.09	60.00	9.09	60	9.09	60.00	9.09
gdb20	30	5	121	123	1.65	123.00	1.65	123	1.65	123.00	1.65
gdb21	30	5	156	158	1.28	158.00	1.28	156	0.00	156.00	0.00
gdb22	30	5	200	201	0.50	201.80	0.90	200	0.00	200.00	0.00
gdb23	20	5	233	245	5.15	246.80	5.92	240	3.00	242.50	4.08
Average					2.98		3.22		1.80		1.99

Table 7: Results for the ARP with hard constraints - gdb (BKS refers to the standard ARP)

Instance	C_{max}	P_{max}	**BKS**	**MS**				**BRILS**			
				Best	**Gap (%)**	**Avg**	**Gap (%)**	**Best**	**Gap (%)**	**Avg**	**Gap (%)**
			(1)	(2)	(1)-(2)	(3)	(1)-(3)	(4)	(1)-(4)	(5)	(1)-(5)
gdb1	70	5	316	323	2.22	323.00	2.22	323	2.22	323.00	2.22
gdb2	60	5	339	374	10.32	377.20	11.27	345	1.77	345.00	1.77
gdb3	60	5	275	296	7.64	296.00	7.64	289	5.09	289.00	5.09
gdb4	70	5	287	317	10.45	317.00	10.45	317	10.45	317.00	10.45
gdb5	70	5	377	395	4.77	398.20	5.62	395	4.77	395.00	4.77
gdb6	70	5	298	313	5.03	313.00	5.03	313	5.03	313.00	5.03
gdb7	60	5	325	389	19.69	389.00	19.69	389	19.69	389.00	19.69
gdb8	40	5	348	364	4.60	364.00	4.60	358	2.87	358.20	2.93
gdb9	40	5	303	321	5.94	322.00	6.27	313	3.30	315.50	4.13
gdb10	60	5	275	291	5.82	296.40	7.78	289	5.09	290.60	5.67
gdb11	80	5	395	409	3.54	411.80	4.25	403	2.03	403.00	2.03
gdb12	100	5	458	492	7.42	495.60	8.21	458	0.00	458.00	0.00
gdb13	130	5	536	544	1.49	544.00	1.49	544	1.49	544.00	1.49
gdb14	20	5	100	107	7.00	107.00	7.00	105	5.00	105.00	5.00
gdb15	20	5	58	58	0.00	58.00	0.00	58	0.00	58.00	0.00
gdb16	30	5	127	129	1.57	129.00	1.57	127	0.00	127.00	0.00
gdb17	20	5	91	91	0.00	91.00	0.00	91	0.00	91.00	0.00
gdb18	30	5	164	182	10.98	182.00	10.98	170	3.66	170.00	3.66
gdb19	20	5	55	63	14.55	63.00	14.55	63	14.55	63.00	14.55
gdb20	30	5	121	123	1.65	123.00	1.65	123	1.65	123.00	1.65
gdb21	30	5	156	158	1.28	158.00	1.28	156	0.00	156.00	0.00
gdb22	30	5	200	202	1.00	202.00	1.00	200	0.00	200.00	0.00
gdb23	20	5	233	243	4.29	243.00	4.29	239	2.58	239.40	2.75
Average					5.71		5.95		3.97		4.04

Table 8: Results for the NS-ARP with soft constraints - kshs (BKS refers to the standard ARP)

Instance	C_{max}	P_{max}	**BKS**	**MS**				**BRILS**			
				Best	**Gap (%)**	**Avg**	**Gap (%)**	**Best**	**Gap (%)**	**Avg**	**Gap (%)**
			(1)	(2)	(1)-(2)	(3)	(1)-(3)	(4)	(1)-(4)	(5)	(1)-(5)
kshs1	4000	300	14661	14666.00	0.03	14666.00	0.03	14666.00	0.03	14666.01	0.03
kshs2	3000	300	9863	9863.00	0.00	9863.00	0.00	9863.00	0.00	9863.00	0.00
kshs3	3000	300	9320	9666.00	3.71	9666.00	3.71	9320.00	0.00	9320.00	0.00
kshs4	3000	300	11498	11508.59	0.09	11508.59	0.09	11508.59	0.09	11508.59	0.09
kshs5	3000	300	10957	11010.68	0.49	11010.68	0.49	11010.68	0.49	11023.84	0.61
kshs6	4000	300	10197	10202.00	0.05	10202.00	0.05	10202.00	0.05	10202.00	0.05
Average					0.73		0.73		0.11		0.13

Table 9: Results for the ARP with hard constraints - kshs (BKS refers to the standard ARP)

Instance	C_{max}	P_{max}	**BKS**	**MS**				**BRILS**			
				Best	**Gap (%)**	**Avg**	**Gap (%)**	**Best**	**Gap (%)**	**Avg**	**Gap (%)**
			(1)	(2)	(1)-(2)	(3)	(1)-(3)	(4)	(1)-(4)	(5)	(1)-(5)
kshs1	4000	300	14661	14783	0.83	14783	0.83	14729	0.46	14729	0.46
kshs2	3000	300	9863	9863	0.00	9863	0.00	9863	0.00	9863	0.00
kshs3	3000	300	9320	9666	3.71	9666	3.71	9320	0.00	9320	0.00
kshs4	3000	300	11498	12844	11.71	12844	11.71	12768	11.05	12768	11.05
kshs5	3000	300	10957	15571	42.11	15571	42.11	15571	42.11	15571	42.11
kshs6	4000	300	10197	10305	1.06	10305	1.06	10305	1.06	10305	1.06
Average					9.90		9.90		9.11		9.11

Table 10: Results for the NS-ARP with soft constraints - val (BKS refers to the standard ARP)

Instance	C_{max}	P_{max}	**BKS**	**MS**				**BRILS**			
				Best	**Gap (%)**	**Avg**	**Gap (%)**	**Best**	**Gap (%)**	**Avg**	**Gap (%)**
			(1)	(2)	(1)-(2)	(3)	(1)-(3)	(4)	(1)-(4)	(5)	(1)-(5)
val1A	90	8	173	173.00	0.00	175.60	1.50	173.00	0.00	173.00	0.00
val1B	50	8	173	188.00	8.67	188.60	9.02	181.00	4.62	181.50	4.91
val1C	40	8	245	251.00	2.45	252.90	3.22	248.00	1.22	248.00	1.22
val2A	150	8	227	229.00	0.88	232.20	2.29	227.00	0.00	227.00	0.00
val2B	90	8	259	265.70	2.59	266.22	2.79	265.70	2.59	265.82	2.63
val2C	80	8	457	462.00	1.09	462.00	1.09	457.00	0.00	457.00	0.00
val3A	40	8	81	87.03	7.44	87.03	7.44	86.00	6.18	86.03	6.21
val3B	30	8	87	93.34	7.29	94.28	8.37	92.96	6.85	93.00	6.89
val3C	30	8	138	139.00	0.72	139.00	0.72	138.00	0.00	138.00	0.00
val4A	140	8	400	407.84	1.96	407.98	2.00	402.00	0.50	405.31	1.33
val4B	110	8	412	435.00	5.58	436.25	5.89	422.00	2.43	425.09	3.18
val4C	100	8	428	455.00	6.31	458.02	7.01	444.81	3.93	448.62	4.82
val4D	100	8	528	549.00	3.98	549.40	4.05	530.00	0.38	536.10	1.53
val5A	90	8	423	458.00	8.27	461.40	9.08	451.00	6.62	452.46	6.96
val5B	120	8	446	466.30	4.55	468.59	5.07	451.00	1.12	455.07	2.03
val5C	100	8	474	500.71	5.63	504.29	6.39	485.00	2.32	489.24	3.22
val5D	80	8	575	618.25	7.52	625.81	8.84	593.00	3.13	602.62	4.80
val6A	70	8	223	233.00	4.48	237.26	6.39	229.00	2.69	229.20	2.78
val6B	60	8	233	240.96	3.42	240.96	3.42	238.96	2.56	238.96	2.56
val6C	50	8	317	330.00	4.10	330.30	4.20	323.00	1.89	324.56	2.39
val7A	90	8	279	280.00	0.36	283.60	1.65	279.00	0.00	279.57	0.20
val7B	70	8	283	294.00	3.89	297.05	4.96	283.00	0.00	289.10	2.15
val7C	40	8	334	358.00	7.19	360.36	7.89	348.00	4.19	350.30	4.88
val8A	130	8	386	400.12	3.66	405.32	5.00	392.72	1.74	394.08	2.09
val8B	110	8	395	421.81	6.79	425.83	7.80	401.00	1.52	405.40	2.63
val8C	70	8	521	577.00	10.75	581.03	11.52	551.44	5.84	557.75	7.05
val9A	110	8	323	327.00	1.24	330.98	2.47	326.00	0.93	330.15	2.21
val9B	90	8	326	338.00	3.68	340.52	4.45	328.00	0.61	330.50	1.38
val9C	70	8	332	353.00	6.33	355.84	7.18	344.00	3.61	347.01	4.52
val9D	50	8	389	417.05	7.21	421.63	8.39	405.00	4.11	409.35	5.23
val10A	150	8	428	437.00	2.10	438.80	2.52	430.00	0.47	430.10	0.49
val10B	120	8	436	450.00	3.21	453.30	3.97	438.00	0.46	439.30	0.76
val10C	100	8	446	469.00	5.16	470.75	5.55	449.00	0.67	454.10	1.82
val10D	60	8	525	576.01	9.72	579.49	10.38	545.08	3.82	552.74	5.28
Average					4.65		5.37		2.27		2.89

Table 11: Results for the ARP with hard constraints - val (BKS refers to the standard ARP)

Instance	C_{max}	P_{max}	**BKS**	**MS**				**BRILS**			
				Best	**Gap (%)**	**Avg**	**Gap (%)**	**Best**	**Gap (%)**	**Avg**	**Gap (%)**
			(1)	(2)	(1)-(2)	(3)	(1)-(3)	(4)	(1)-(4)	(5)	(1)-(5)
val1A	90	8	173	173	0.00	173.20	0.12	173.00	0.00	173.00	0.00
val1B	50	8	173	185	6.94	185.40	7.17	181.00	4.62	181.00	4.62
val1C	40	8	245	250	2.04	250.80	2.37	248.00	1.22	248.00	1.22
val2A	150	8	227	231	1.76	233.60	2.91	227.00	0.00	227.00	0.00
val2B	90	8	259	270	4.25	295.60	14.13	270	4.25	295.60	14.13
val2C	80	8	457	462	1.09	462.00	1.09	457	0.00	457.00	0.00
val3A	40	8	81	85	4.94	85.00	4.94	83	2.47	83.00	2.47
val3B	30	8	87	100	14.94	100.00	14.94	100	14.94	100.00	14.94
val3C	30	8	138	139	0.72	139.00	0.72	138	0.00	138.00	0.00
val4A	140	8	400	410	2.50	410.40	2.60	402	0.50	407.80	1.95
val4B	110	8	412	431	4.61	438.80	6.50	418	1.46	430.60	4.51
val4C	100	8	428	452	5.61	459.60	7.38	436	1.87	445.40	4.07
val4D	100	8	528	550	4.17	551.30	4.41	530	0.38	537.50	1.80
val5A	90	8	423	545	28.84	549.60	29.93	506	19.62	511.80	20.99
val5B	120	8	446	467	4.71	469.60	5.29	451	1.12	453.90	1.77
val5C	100	8	474	509	7.38	519.60	9.62	480	1.27	494.20	4.26
val5D	80	8	575	629	9.39	631.90	9.90	591	2.78	602.40	4.77
val6A	70	8	223	235	5.38	235.00	5.38	229	2.69	229.00	2.69
val6B	60	8	233	251	7.73	251.00	7.73	245	5.15	245.00	5.15
val6C	50	8	317	326	2.84	329.60	3.97	321	1.26	323.60	2.08
val7A	90	8	279	284	1.79	286.50	2.69	279	0.00	279.00	0.00
val7B	70	8	283	295	4.24	295.90	4.56	284	0.35	286.00	1.06
val7C	40	8	334	369	10.48	371.30	11.17	352	5.39	354.00	5.99
val8A	130	8	386	407	5.44	407.50	5.57	395	2.33	396.70	2.77
val8B	110	8	395	413	4.56	422.50	6.96	401	1.52	403.80	2.23
val8C	70	8	521	609	16.89	614.60	17.97	549	5.37	566.30	8.69
val9A	110	8	323	328	1.55	331.90	2.76	326	0.93	327.20	1.30
val9B	90	8	326	335	2.76	338.10	3.71	329	0.92	331.80	1.78
val9C	70	8	332	356	7.23	358.00	7.83	344	3.61	347.30	4.61
val9D	50	8	389	414	6.43	422.90	8.71	403	3.60	409.60	5.30
val10A	150	8	428	437	2.10	439.70	2.73	430	0.47	430.80	0.65
val10B	120	8	436	447	2.52	451.60	3.58	438	0.46	439.50	0.80
val10C	100	8	446	468	4.93	469.60	5.29	448	0.45	454.80	1.97
val10D	60	8	525	581	10.67	583.50	11.14	544	3.62	549.30	4.63
Average					5.92		6.93		2.78		3.92

Table 12: Summary of obtained gaps w.r.t. BKS (standard ARP)

Instance	**Soft constraint (non-smooth)**				**Hard constraint**			
	MS		**BRILS**		**MS**		**BRILS**	
	Best	Avg	Best	Avg	Best	Avg	Best	Avg
egl	1.57	1.85	0.84	1.07	2.05	2.39	1.34	1.61
gdb	2.98	3.22	1.80	1.99	5.71	5.95	3.97	4.04
kshs	0.73	0.73	0.11	0.13	9.90	9.90	9.11	9.11
val	4.65	5.37	2.27	2.89	5.92	6.93	2.78	3.92
Average	3.07	3.61	1.25	1.52	5.90	6.29	4.30	4.67

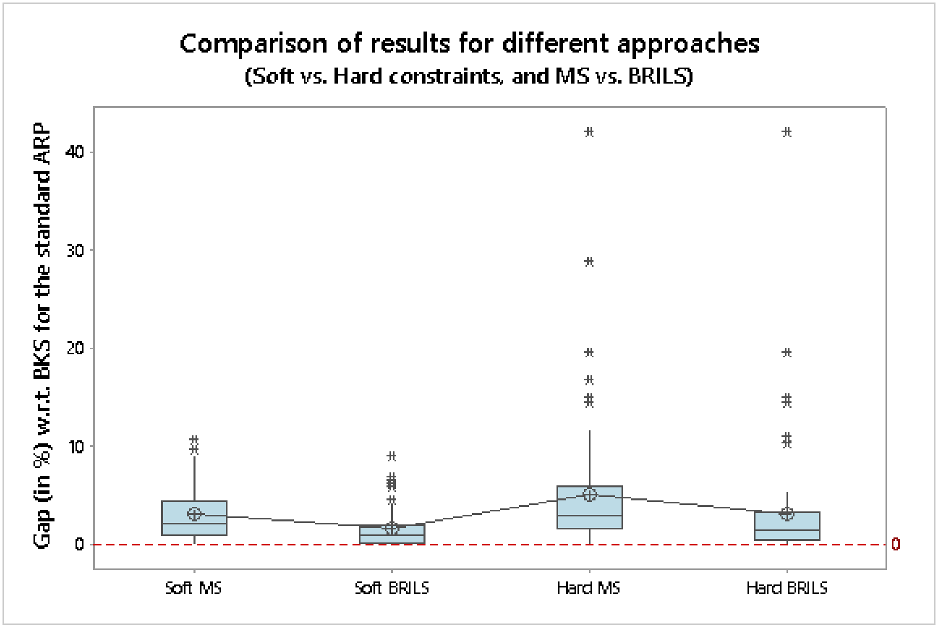

Figure 5: Visual comparison of the different approaches

6.2. Evaluation of the Solving Approaches for the Truncated NS-ARP

After testing the performance of our BRILS algorithm for the NS-ARP and analyzing the benefits of considering soft constraints, this section compares the BRILS results with the ones generated by CPLEX for the truncated version of the problem described in Section 4. As stated before, the model needs to be truncated in order to be solved using CPLEX. Thus, the penalty term has been linearized. By doing so, the total cost provided by CPLEX becomes an upper bound for the optimal cost associated with the non-smooth model. The total running time for CPLEX has been fixed to one hour.

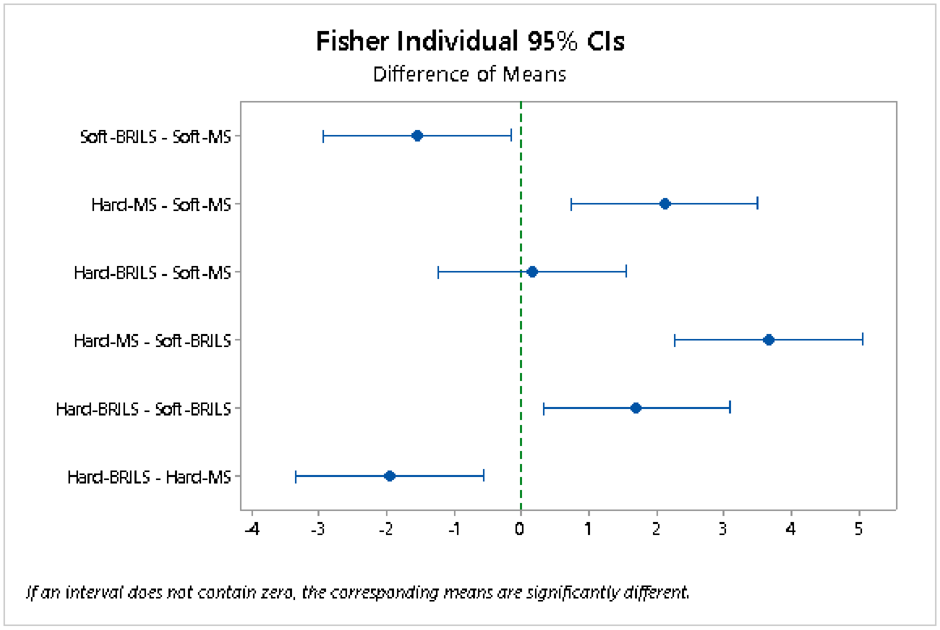

Figure 6: Fisher test for differences of means

Tables 13 and 14 show the results for the small-size instances, *kshs* and *gdb*. Their structure is the following: the first column corresponds to the instance name, the next three columns correspond to the CPLEX results for the truncated model (i.e., cost, computing time, and maximum gap with respect to the optimal solution), and the final two columns show the solution provided by our BRILS algorithm for the NS-ARP and the corresponding gap with the one provided by CPLEX for the truncated model. Notice that our BRILS approach reaches the value provided by CPLEX for some instances in the *kshs* set (Table 13), while it is also able to obtain better results for other instances by considering the non-smooth model instead of a truncated one. For the next set of instances, *gdb*, CPLEX is not always able to report an optimal solution after one hour of computing. In fact, it runs out of memory for some of these instances (*gdb*8, *gdb*9, and *gdb*11). All in all, in those cases in which CPLEX is able to find an optimal solution, BRILS is also able to reach the same value. Moreover, in those cases in which CPLEX could not even reach an optimal solution for the truncated model, BRILS is able to improve the solution provided by CPLEX. Finally, it is worthy to mention that the remaining sets of instances – *val* and *egle* – could not be solved

Table 13: Comparison with exact results for *kshs* instances

Instance	**Truncated Model**			**BRILS (non-smooth)**	
	Opt (1)	Time (s.)	Gap (%)	Best (2)	Gap (%) (1)-(2)
kshs1	14674.88	6.00	0.00	14666.00	-0.06
kshs2	9863.00	24.27	0.00	9863.00	0.00
kshs3	9320.00	5.70	0.00	9320.00	0.00
kshs4	11546.25	10.45	0.00	11508.59	-0.33
kshs5	11170.02	6.69	0.00	11010.68	-1.43
kshs6	10204.43	1.50	0.00	10202.00	-0.02

Table 14: Comparison with exact results for *gdb* instances

Instance	**Truncated Model**			**BRILS (non-smooth)**	
	Opt (1)	Time (s.)	Gap (%)	Best (2)	Gap (%) (1)-(2)
gbd1	326	3600.00	8.10	321	-1.53
gbd2	349	3600.00	12.14	344	-1.43
gbd3	284	3600.00	5.69	280	-1.41
gbd4	292	0.00	495.38	292	0.00
gbd5	382	3600.00	9.95	382	0.00
gbd6	303	1052.21	0.36	303	0.00
gbd7	330	3600.00	12.44	330	0.00
gbd8	–	–	–	353	–
gbd9	–	–	–	312	–
gbd10	280	3600.00	5.71	280	0.00
gbd11	–	–	–	400	–
gbd12	497	3600.00	24.78	458	-7.85
gbd13	551	3600.00	4.72	541	-1.81
gbd14	105	3.92	0.00	105	0.00
gbd15	58	2.73	0.00	58	0.00
gbd16	127	223.15	0.00	127	0.00
gbd17	91	3600.00	4.40	91	0.00
gbd18	169	227.13	0.00	169	0.00
gbd19	60	1.00	0.00	60	0.00
gbd20	126	3600.00	5.55	123	-2.38
gbd21	156	3600.00	2.56	156	0.00
gbd22	210	3600.00	9.05	200	-4.76
gbd23	443	3600.00	48.53	240	-45.82

using CPLEX, since the computer run out of memory during the execution.

7. Conclusions

This paper analyzes a non-smooth version of the well-known ARP (NS-ARP). In this version, a realistic soft constraint regarding a maximum cost per route is considered, and a penalty cost is added to the objective function in order to account for violations of this constraint. The introduction of these penalty costs makes the objective function to become a non-smooth one, hindering the use of classical optimization methods. A novel mathematical model is provided for the NS-ARP, and a hybrid metaheuristic is proposed to solve it. Our BRILS algorithm integrates the biased-randomization of a fast heuristic inside an iterated local search framework.

Through an extensive set of numerical experiments we have shown that the introduction of a new hard constraint can severely affect the quality of the solutions, and this effect is diminished when considering realistic soft constraints, as we proposed. Our BRILS algorithm constitutes a system able to address this issue and outperforms previously published methods for solving the NS-ARP. Namely, regarding previous methods, the gap is being reduced from 3.61 to 1.52 on average and regarding the hard constraint consideration, the gap is being reduced from 4.67 to 1.52 on average. This means that our proposal exhibits relevant advantages to solve the problem at hand, in terms of both the way the problem is raised and the way it is solved. Finally, we make use of general purpose solver (i.e., CPLEX) to solve a truncated model of the problem in order to obtain tight upper bounds for the NS-ARP. This allows us to illustrate that even the linearized problem cannot be easily solved using exact methods. This shortcoming is overcome by means of the BRILS algorithm which is always able to provide a feasible solution for this non-smooth version – even for large-size instances –, where the exact truncated model cannot provide any solution. This feature of our approach is an important distinction, bearing in mind that the NS-ARP may arise in real-world scenarios where non-smooth functions have to considered.

Counting with approaches as those provided in this paper, allows, on the one hand, to compare the performance of a given decision support system by using the truncated model, and on the other hand, to address dynamic changes in integrative systems where a solution to the ARP is frequently required in the context of re-planning and disturbance management. The above requires fast and high-quality solutions making thus the BRILS approach a very suitable option as it provides solutions in the range of seconds. Finally, thanks to the realistic constraints considered in the NS-ARP, the use of our proposed approaches can be incorporated into expert systems where by means of friendly user interfaces – e.g., Heilig et al. (2017) – the user can either provide and assess different parameter alternatives or machine-learning systems – e.g., de León et al. (2017) – where among different solution approaches, that one providing a better performance for the problem at hand is recommended.

Despite its relevance in real-life applications, soft constraints and non-smooth objective functions have been rarely considered in the extensive literature on arc and vehicle routing problems. For that reason, as future work we propose *(i)* to extend the analysis presented here to other combinatorial optimization problems in the logistics and transportation field; *(ii)* to integrate this problem with related ones in order to provide decision makers integrative and realistic solutions;*(iii)* to incorporate the solution approaches and the problem consideration in specialized systems as the above-mentioned ones in order to directly assist managers during the whole decision making process; and *(iv)* to consider the stochasticity within the NS-ARP environment in order to study and provide robust solutions adapted to the scenario at hand.

References

Ahr, D. and Reinelt, G. (2014). The capacitated arc routing problem: Combinatorial lower bounds. In Corberan, A. and Laporte, G., editors, *Arc routing: problems, methods, and applications*, pages 159–181.

Al-Sultan, K. S. (1995). A tabu search approach to the clustering problem. *Pattern Recognition*, 28(9):1443 – 1451.

Amberg, A., Domschke, W., and Vo, S. (2000). Multiple center capacitated arc routing problems: A tabu search algorithm using capacitated trees. *European Journal of Operational Research*, 124(2):360 – 376.

Badescu, V. (2015). Optimal profiles for one dimensional slider bearings under technological constraints. *Tribology International*, 90(Supplement C):198 – 216.

Badescu, V. (2017). Smooth and non-smooth optimal pin fin profiles beyond the schmidt optimality assumption and length-of-arc approximation. *Applied Mathematical Modelling*, 47(Supplement C):358 – 380.

Bagirov, A., Lai, D. T. H., and Palaniswami, M. (2007). A nonsmooth optimization approach to sensor network localization. In *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on*, pages 727–732.

Bagirov, A. M. and Yearwood, J. (2006). A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems. *European Journal of Operational Research*, 170(2):578 – 596.

Baldacci, R. and Maniezzo, V. (2006). Exact methods based on node-routing formulations for undirected arc-routing problems. *Networks*, 47(1):52–60.

Basu, M. (2015). Modified particle swarm optimization for non-smooth non-convex combined heat and power economic dispatch. *Electric Power Components and Systems*, 43(19):2146–2155.

Belenguer, J. M. and Benavent, E. (2003). A cutting plane algorithm for the capacitated arc routing problem. *Computers & Operations Research*, 30(5):705 – 728.

Belenguer, J. M., Benavent, E., and Irnich, S. (2014). The capacitated arc routing problem: Exact algorithms. In Corberan, A. and Laporte, G., editors, *Arc routing: problems, methods, and applications*, pages 183–221.

Benavent, E., Campos, V., Corberan, A., and Mota, E. (1992). The capacitated chinese postman problem: Lower bounds. *Networks*, 22(7):669 – 690.

Beullens, P., Muyldermans, L., Cattrysse, D., and Oudheusden, D. V. (2003). A guided local search heuristic for the capacitated arc routing problem. *European Journal of Operational Research*, 147(3):629 – 643.

Bode, C. and Irnich, S. (2015). In-depth analysis of pricing problem relaxations for the capacitated arc-routing problem. *Transportation Science*, 49(2):369–383.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization.* Cambridge University Press, Cambridge.

Brando, J. and Eglese, R. (2008). A deterministic tabu search algorithm for the capacitated arc routing problem. *Computers & Operations Research*, 35(4):1112 – 1126.

Caceres-Cruz, J., Arias, P., Guimarans, D., Riera, D., and Juan, A. A. (2014). Rich vehicle routing problem: Survey. *ACM Comput. Surv.*, 47(2):32:1–32:28.

Chen, Y., Hao, J.-K., and Glover, F. (2016). A hybrid metaheuristic approach for the capacitated arc routing problem. *European Journal of Operational Research*, 253(1):25 – 39.

Corazza, M., Fasano, G., and Gusso, R. (2013). Particle swarm optimization with non-smooth penalty reformulation, for a complex portfolio selection problem. *Applied Mathematics and Computation*, 224:611 – 624.

Corberan, A. and Laporte, G. (2014). *Arc routing: problems, methods, and applications.* MOS-SIAM series on optimization., Philadelphia, PA.

Corberan, A. and Prins, C. (2010). Recent results on arc routing problems: An annotated bibliography. *Networks*, 56(1):50–69.

de León, A. D., Lalla-Ruiz, E., Melián-Batist, B., and Moreno-Vega, J. M. (2017).

A machine learning-based system for berth scheduling at bulk terminals. *Expert Systems with Applications*.

Derigs, U., Li, B., and Vogel, U. (2010). Local search-based metaheuristics for the split delivery vehicle routing problem. *The Journal of the Operational Research Society*, 61(9):1356–1364.

Derigs, U. and Metz, A. (1992). A matching-based approach for solving a delivery/pick-up vehicle routing problem with time constraints. *Operations-Research-Spektrum*, 14(2):91–106.

Doerner, K., Hartl, R., Maniezzo, V., and Reimann, M. (2003). An ant system metaheuristic for the capacitated arc routing problem. In *Preprints of 5th Metaheuristics International Conference*, Kyoto.

Dominguez, O., Juan, A. A., Barrios, B., Faulin, J., and Agustin, A. (2016). Using biased randomization for solving the two-dimensional loading vehicle routing problem with heterogeneous fleet. *Annals of Operations Research*, 236(2):383–404.

Dominguez, O., Juan, A. A., and Faulin, J. (2014). A biased-randomized algorithm for the two-dimensional vehicle routing problem with and without item rotations. *International Transactions in Operational Research*, 21(3):375–398.

Dror, M., editor (2000). *Arc Routing : Theory, Solutions, and Applications.* Boston, MA: Kluwer Academic.

Eglese, R. (1994). Routeing winter gritting vehicles. *Discrete Applied Mathematics*, 48(3):231 – 244.

Eglese, R. W. and Letchford, A. N. (2000). *Polyhedral Theory for Arc Routing Problems*, pages 199–230. Springer US, Boston, MA.

Ferrer, A., Guimarans, D., Ramalhinho, H., and Juan, A. A. (2016). A brils metaheuristic for non-smooth flow-shop problems with failure-risk costs. *Expert Systems with Applications*, 44:177–186.

Garey, M. R. and Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA.

Golden, B., Dearmon, J., and Baker, E. (1983). Computational experiments with algorithms for a class of routing problems. *Computers & Operations Research*, 10(1):47 – 59.

Golden, B. L. and Wong, R. T. (1981). Capacitated arc routing problems. *Networks*, 11(3):305–315.

Gonzalez, S., Ferrer, A., Juan, A. A., and Riera, D. (2014). Solving non-smooth arc routing problems throughout biased-randomized heuristics. In *Computer-based Modelling and Optimization in Transportation*, pages 451–462. Springer.

Gonzalez, S., Juan, A., Riera, D., Elizondo, M., and Ramos, J. (2016). A simheuristic algorithm for solving the arc routing problem with stochastic demands. *Journal of Simulation*.

Gonzalez, S., Juan, A. A., Riera, D., Castella, Q., Munoz, R., and Perez, A. (2012). Development and assessment of the sharp and randsharp algorithms for the arc routing problem. *AI Communications*, 25(2):173–189.

Grasas, A., Juan, A. A., Faulin, J., de Armas, J., and Ramalhinho, H. (2017). Biased randomization of heuristics using skewed probability distributions: a survey and some applications. *Computers & Industrial Engineering*.

Grasas, A., Juan, A. A., and Ramalhinho, H. (2016). Simils: A simulation-based extension of the iterated local search metaheuristic for stochastic combinatorial optimization. *Journal of Simulation*, 10(1):69–77.

Greistorfer, P. (2003). A tabu scatter search metaheuristic for the arc routing problem. *Comput. Ind. Eng.*, 44(2):249–266.

Hamdan, M. and El-Hawary, M. E. (2002). Hopfield-genetic approach for solving the routing problem in computer networks. In *Electrical and Computer Engineering, 2002. IEEE CCECE 2002. Canadian Conference on*, volume 2, pages 823–827 vol.2.

Hashimoto, H., Ibaraki, T., Imahori, S., and Yagiura, M. (2006). The vehicle routing problem with flexible time windows and traveling times. *Discrete Applied Mathematics*, 154(16):2271 – 2290.

Heilig, L., Lalla-Ruiz, E., and Voß, S. (2017). port-io: an integrative mobile cloud platform for real-time inter-terminal truck routing optimization. *Flexible Services and Manufacturing Journal*, 29(3-4):504–534.

Hemamalini, S. and Simon, S. P. (2010). Artificial bee colony algorithm for economic load dispatch problem with non-smooth cost functions. *Electric Power Components and Systems*, 38(7):786–803.

Hertz, A. (2005). *Recent Trends in Arc Routing*, pages 215–236. Springer US, Boston, MA.

Hertz, A., Laporte, G., and Mittaz, M. (2000). A tabu search heuristic for the capacitated arc routing problem. *Operations Research*, 48(1):129–135.

Hertz, A. and Mittaz, M. (2001). A variable neighborhood descent algorithm for the undirected capacitated arc routing problem. *Transportation Science*, 35(4):425–434.

Hirabayashi, R., Nishida, N., and Saruwatari, Y. (1992a). Node duplication lower bounds for the capacitated arc routing problems. *Journal of the Operations Research Society of Japan*, 35(2):119133.

Hirabayashi, R., Nishida, N., and Saruwatari, Y. (1992b). Tour construction algorithm for the capacitated arc routing problem. *Asia-Pacific Journal of Operational Research*, 9(2):155175.

Juan, A. A., Faulin, J., Ferrer, A., Lourenço, H. R., and Barrios, B. (2011). Mirha: multi-start biased randomization of heuristics with adaptive local search for solving non-smooth routing problems. *TOP*, 21(1):109–132.

Juan, A. A., Faulin, J., Grasman, S., Rabe, M., and Figueira, G. (2015a). A review of simheuristics: extending metaheuristics to deal with stochastic optimization problems. *Operations Research Perspectives*, 2:62–72.

Juan, A. A., Grasman, S. E., Caceres-Cruz, J., and Bektaş, T. (2014a). A simheuristic algorithm for the single-period stochastic inventory-routing problem with stock-outs. *Simulation Modelling Practice and Theory*, 46:40–52.

Juan, A. A., Loureno, H. R., Mateo, M., Luo, R., and Castella, Q. (2014b). Using iterated local search for solving the flow-shop problem: Parallelization, parametrization, and randomization issues. *International Transactions in Operational Research*, 21(1):103–126.

Juan, A. A., Pascual, I., Guimarans, D., and Barrios, B. (2015b). Combining biased randomization with iterated local search for solving the multidepot vehicle routing problem. *International Transactions in Operational Research*, 22(4):647–667.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Kiuchi, M., Shinano, Y., Hirabayashi, R., and Saruwatari, Y. (1995). An exact algorithm for the capacitated arc routing problem using parallel branch and bound method. In *Abstracts of the 1995 Spring National Conference of the Operations Research Society of Japan*, pages 28–29.

Lacomme, P., Prins, C., and Ramdane-Cherif, W. (2001). Competitive genetic algorithms for the capacitated arc routing problem and its extensions. In *Proceedings of the 4th European Conference on Genetic Programming*, volume 2037, pages 473–483.

Lacomme, P., Prins, C., and Ramdane-Cherif, W. (2004). Competitive memetic algorithms for arc routing problems. *Annals of Operations Research*, 131(1):159–185.

Leon Y. O. Li, R. W. E. (1996). An interactive algorithm for vehicle routeing for winter - gritting. *The Journal of the Operational Research Society*, 47(2):217–228.

Letchford, A. N. and Oukil, A. (2009). Exploiting sparsity in pricing routines for the capacitated arc routing problem. *Computers & Operations Research*, 36(7):2320 – 2327.

Li, L. (1882). *Contributions to routing problems.* PhD thesis, University of Augsburg, Germany.

Li, L. (1992). *Vehicle routeing for winter gritting.* PhD thesis, Department of Management Science, Lancaster University.

Longo, H., de Aragão, M. P., and Uchoa, E. (2006). Solving capacitated arc routing problems using a transformation to the cvrp. *Computers & Operations Research*, 33(6):1823 – 1837.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2010). *Handbook of Metaheuristics*, chapter Iterated Local Search: Framework and Applications, pages 363–397. Springer US, Boston, MA.

Lu, Y., Zhou, J., Qin, H., Li, Y., and Zhang, Y. (2010). An adaptive hybrid differential evolution algorithm for dynamic economic dispatch with valve-point effects. *Expert Systems with Applications*, 37(7):4842–4849.

Montoya-Torres, J. R., Juan, A. A., Huaccho Huatuco, L., Faulin, J., and Rodriguez-Verjan, G. L. (2012). *Hybrid Algorithms for Service, Computing and Manufacturing Systems: Routing and Scheduling Solutions*. IGI Global, Hershey, PA.

Neyestani, M., Farsangi, M. M., and Nezamabadi-pour, H. (2010). A modified particle swarm optimization for economic dispatch with non-smooth cost functions. *Engineering Applications of Artificial Intelligence*, 23(7):1121 – 1126.

Niknam, T., Mojarrad, H. D., Meymand, H. Z., and Firouzi, B. B. (2011). A new honey bee mating optimization algorithm for non-smooth economic dispatch. *Energy*, 36(2):896 – 908.

Oonsivilai, A., Srisuruk, W., Marungsri, B., and Kulworawanichpong, T. (2009). Tabu search approach to solve routing issues in communication networks. *International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, 3(5):1211 – 1214.

Oppen, J. and Lkketangen, A. (2006). Arc routing in a node routing environment. *Computers & Operations Research*, 33(4):1033 – 1055. Part Special Issue: Optimization Days 2003.

Papadimitriou, C. H. and Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Pearn, W. L. (1988). New lower bounds for the capacitated arc routing problem. *Networks*, 18(3):181–191.

Pearn, W. L. (1989). Approximate solutions for the capacitated arc routing problem. *Computers & Operations Research*, 16(6):589 – 600.

Pearn, W. L. (1991). Augment-insert algorithms for the capacitated arc routing problem. *Computers & Operations Research*, 18(2):189 – 198.

Pearn, W.-L., Assad, A., and Golden, B. L. (1987). Transforming arc routing into node routing problems. *Computers & Operations Research*, 14(4):285 – 288.

Prins, C. (2014). The capacitated arc routing problem: Heuristics. In Corberan, A. and Laporte, G., editors, *Arc routing: problems, methods, and applications*, pages 131–157.

Ramadurai, V. and Sichitiu, M. L. (2003). Localization in wireless sensor networks: A probabilistic approach. In *International conference on wireless networks*, pages 275–281.

Roy, P., Ghoshal, S., and Thakur, S. (2010). Biogeography based optimization for multi-constraint optimal power flow with emission and non-smooth cost function. *Expert Systems with Applications*, 37(12):8221–8228.

Schlüter, M., Egea, J. A., and Banga, J. R. (2009). Extended ant colony optimization for non-convex mixed integer nonlinear programming. *Computers & Operations Research*, 36(7):2217 – 2229.

Selim, S. Z. and Alsultan, K. (1991). A simulated annealing algorithm for the clustering problem. *Pattern Recognition*, 24(10):1003 – 1008.

Toscano, R. and Lyonnet, P. (2010). A new heuristic approach for non-convex optimization problems. *Information Sciences*, 180(10):1955 – 1966. Special Issue on Intelligent Distributed Information Systems.

Vaisakh, K. and Srinivas, L. (2011). Genetic evolving ant direction HDE for OPF with non-smooth cost functions and statistical analysis. *Expert Systems with Applications*, 38(3):2046 – 2062.

Vaisakh, K., Srinivas, L., and Meah, K. (2013). Genetic evolving ant direction particle swarm optimization algorithm for optimal power flow with non-smooth cost functions and statistical analysis. *Applied Soft Computing*, 13(12):4579 – 4593.

Welz, S. (1994). *Optimal solutions for the capacitated arc routing problem using integer programming.* PhD thesis, Department of QT and OM, University of Cincinnati.

Wøhlk, S. (2005). *Contributions to arc routing.* PhD thesis, University Southern Denmark.

Wøhlk, S. (2006). New lower bound for the capacitated arc routing problem. *Computers & Operations Research*, 33(12):3458 – 3472.

Wøhlk, S. (2008). *A Decade of Capacitated Arc Routing*, pages 29–48. Springer US, Boston, MA.

Yang, I.-T. and Chou, J.-S. (2011). Multiobjective optimization for manpower assignment in consulting engineering firms. *Applied Soft Computing*, 11(1):1183 – 1190.