Computers and Operations Research 115 (2020) 104854

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

A hybrid reactive GRASP heuristic for the risk-averse k-traveling repairman problem with profits

M.E. Bruni[*], P. Beraldi, S. Khodaparasti

Department of Mechanical, Energy and Management Engineering, University of Calabria, Cosenza, Rende, Italy

ARTICLE INFO

Article history:
Received 7 June 2018
Revised 26 September 2019
Accepted 9 November 2019
Available online 11 November 2019

Keywords:
Traveling repairman problem
Mean-risk model
GRASP
Adaptive neighbourhood search

ABSTRACT

This paper addresses the k-traveling repairman problem with profits and uncertain travel times, a vehicle routing problem aimed at visiting a subset of customers in order to collect a revenue, which is a decreasing function of the uncertain arrival times. The introduction of the arrival time in the objective function instead of the travel time, which is common in most vehicle routing problems, poses compelling computational challenges, emphasized by the incorporation of the stochasticity in travel times. For tackling the solution of the risk-averse k-traveling repairman problem with profits, in this paper is proposed a hybrid heuristic, where a reactive greedy randomized adaptive search procedure is used as a multi-start framework, equipped with an adaptive local search algorithm. The effectiveness of the solution approach is shown through an extensive experimental phase, performed on a set of instances, generated from three sets of benchmark instances containing up to 200 nodes.

1. Introduction

The k-traveling repairman problem with profits ($k-$TRPP) is a routing problem where a set of customers should be serviced by a fleet of k vehicles with the aim of collecting a profit, which is a monotonically decreasing function of the arrival time of the vehicle at the customer. For this reason, it might not be worth servicing all the customers and the visit of each location is optional. The problem belongs to the class of vehicle routing problems with profits, a flourishing research stream that has attracted the attention of the operations research community in the last ten years. Generally speaking, in this broad class of problems, a profit is associated with each customer and only a subset of customers to be served is chosen; the decision is made on the basis of an objective function that might include the collected profit and/or the travel cost. Depending on the type of the objective function chosen, different names can be found in the literature. When the objective function is the maximization of the total profit collected, and a maximum duration constraint or a capacity constraint (or both) are imposed on vehicle routes, the problems are usually referred to as problem with profits. If the objective function is the minimization of the total traveling cost and a constraint on the minimum amount of profit to be collected is imposed, we categorize the problem as prize-collecting. If the objective function is the maximization of the difference between the total collected profit and the traveling cost, then the routing problem is cast as a profitable problem. It is worthwhile remarking that some discrepancies in problem definition and taxonomy might be encountered in the literature. In fact, notwithstanding the objective function in the TRPP is the collected profit minus the total arrival time, the problem is commonly referred in the literature as the Traveling Repairman Problem with profits. The TRPP is a more faithful modelling paradigm than the corresponding problem without profits- the Traveling Repairman Problem (TRP)- in situations where the service involves the distribution of perishable items, and the customer's satisfaction, expressed as a function of the arrival time, should be considered. For instance, in the aftermath of a disaster, we may consider the utility of servicing each location, as a proxy of the severity of the disaster and of the urgency of providing help. The utility is a decreasing function of the arrival times and it assumes the maximum value when the arrival time is zero. Other application examples include real-time services, where there is usually a service level agreement, such as timeliness, established between a client and a service provider. It is well known that shorter mean delivery time, and less variance with respect to delivery time are general principles for increasing demand and profits (Boyaci and Ray, 2003; Ray and Jewkes, 2004). In fact, depending on how fast a request is served, the price paid by the client may be different.

Despite the similarities between the TRPP and the TRP, solving a typical TRPP is more challenging since, in general, its objective function does not show a monotonic behaviour with respect to the

[*] Corresponding author.
E-mail addresses: mariaelena.bruni@unical.it (M.E. Bruni), patrizia.beraldi@unical.it (P. Beraldi), sara.khodaparasti@unical.it (S. Khodaparasti).

https://doi.org/10.1016/j.cor.2019.104854
0305-0548/© 2019 Elsevier Ltd. All rights reserved.

number of visited customers. In this paper, we consider an extension of the $k-$TRPP that incorporates travel time uncertainty. In the extant literature, it is usually assumed that the repairman travels at a constant speed, which is not a realistic assumption in many cases. The travel time might vary depending on different factors such as fluctuations in weather condition, traffic congestion, and vehicle breakdowns. To the best of our knowledge, notwithstanding there is a vast literature on the incorporation of stochasticity for vehicle routing problems (Beraldi et al. (2015); Bruni et al. (2019a, 2014); Gendreau et al. (2015)), a few contributions exist on latency-based routing problems at the presence of uncertainty. We mention Van Ee and Sitters (2017), where the a priori traveling repairman problem is defined as a stochastic problem with recourse, and the second-stage cost is defined as the sum of expected latencies arising from the revealed set of uncertain vertices to be visited and Bruni et al. (2018b), where the k-TRPP with uncertain travel times under a risk-averse perspective has been introduced. This approach is interesting because it enables the decision-maker to consider the risk associated to the variability of the stochastic parameters, differently from the risk-neutral approach, in which only the expected values are considered. This perspective becomes particularly important in customer-centric vehicle routing problems (and hence both in TRPs and TRPPs), where the attention is directed towards customers related measures. In disaster relief operations and humanitarian logistics operations, for instance, controlling the arrival time variability is crucial and might significantly contribute to decrease the rate of mortality Bruni et al. (2018a).

The focus of this study is on the design of an efficient heuristic solution methods able to tackle the computational challenges of this involved problem. In particular, we propose a metaheuristic approach based on a reactive greedy randomized adaptive search procedure (RGRASP) and a two-level adaptive neighbourhood search, reflecting the two levels of decisions in the TRPP, namely the selection of customers to visit and the visiting sequence of the selected customers in each vehicle route. The proposed algorithm is self-adaptive because different neighbourhoods are applied depending on their performance, which is learnt throughout the search process.

The rest of the paper is organized as follows. We review the main scientific literature in Section 2. In Section 3, we introduce the problem and three procedures providing satisfying upper bounds. The hybrid framework is presented in Section 4. Section 5 is devoted to the presentation of the computational results collected on a set of instances adapted from three benchmark test beds. Finally, concluding remarks are discussed in Section 6.

2. Literature review

There are two classes of problems closely related to the problem studied in this paper: The TRP or minimum latency problem (MLP) and the cumulative vehicle routing problem (CumVRP).

The TRP or MLP is a customer-centric routing problem where the customer satisfaction is taken into account mostly through the arrival time at the customer's location. It consists in finding a tour on a given set of nodes which minimizes the sum of the elapsed times (or latencies) to reach the nodes, starting from a depot node. Various practical applications have been described in the literature, from food distribution to machine and maintenance scheduling and virtual network recovery (Bruni et al., 2019b). The TRP has been extensively studied by a large number of researchers who proposed several exact and non-exact approaches.

Lucena (1990) and Bianco et al. (1993) early proposed exact enumerative algorithms, in which lower bounds are derived using a Lagrangian relaxation. Fischetti et al. (1993) proposed an enumerative algorithm that makes use of lower bounds obtained from a linear integer programming formulation. Van Eijl (1995), Méndez-Díaz et al. (2008), and Ezzine et al. (2010) developed mixed integer programming formulations with various families of valid inequalities. Bigras et al. (2008) suggested a number of integer programming formulations as well as a branch-and-bound algorithm. Unlike the classical traveling salesman problem, it is not easy to provide constant-factor approximation algorithms for the MLP. The first one was suggested by Blum et al. (1994), whereas the current best approximation algorithm for the MLP is due to Chaudhuri et al. (2003) for general metric spaces and to Archer and Blasiak (0000) for the case where an edge-weighted tree is considered.

Up to this date, a few metaheuristics are available for the TRP. Salehipour et al. (2011) first proposed a simple composite algorithm based on a GRASP, improved with a variable neighbourhood search procedure. In Mladenović et al. (2013), presented a general variable neighborhood search metaheuristic enhanced with a move evaluation procedure facilitating the update of the incumbent solution. Silva et al. (2012) presented a composite multi-start metaheuristic approach consisting of a GRASP and a randomized variable neighbourhood descent algorithm for the construction and improvement phases, respectively. They also adopted an efficient move evaluation procedure to speed up the search over different neighbourhoods.

Recently, some interesting problem variants of the TRP/MLP have been proposed. A direct generalization of the TRP is the multiple traveling repairman problem (referred to as k-TRP) that considers identical vehicles (Fakcharoenphol et al. (2007)). Recently, Nucamendi-Guillén et al. (2016) presented an efficient new formulation, defined on a multi-level network, for the deterministic k-traveling repairman problem without profits enhanced by an iterative greedy metaheuristic. The CumVRP generalizes the TRP, by considering a demand associated to each node and by adding capacity constraints. The CumVRP can be further regarded as a special case of the weighted vehicle routing problem proposed by Zhang et al. (2010). Later on, the presence of multiple depots was included in the model by Zhang et al. (2011). Kara et al. (2008) considered a flow-based formulation of the CumCVRP, where the objective function to be minimized is not the sum of arrival times, but rather the sum of arrival times multiplied by the demand of the node. Ngueveu et al. presented in (Ngueveu et al., 2010) a memetic heuristic aimed at visiting a set of customers with a homogeneous capacitated vehicle fleet. The proposed metaheuristic seeks appropriate upper bounds while the lower bounds are obtained by exploiting the properties of the problem. In Ribeiro and Laporte (2012), presented an adaptive large neighbourhood metaheuristic, applying different repair and destroy procedures adopted from the literature. In a recent paper, Sze et al. (2017) presented a hybrid metaheuristic algorithm for the min-max CumCVRP, in which a two-stage adaptive variable neighbourhood search algorithm that incorporates large neighbourhood search as a diversification strategy is designed.

The TRPP is also a variant of the TRP, where a revenue is obtained the first time a location is visited and the visit of each location is optional. Dewilde et al. (2013) first presented the mathematical model for the deterministic TRPP with a single vehicle. In the integer linear programming formulation proposed, the number of visited customers, or equivalently the path length, is an input parameter. The model returns the index of visited customers as well as the order of visiting them along the path. Since the model should be separately solved for different path lengths, this solution framework can be considerably time-consuming for real applications, hence they proposed a tabu search metaheuristic searching over different neighbourhood structures. A stochastic variant of the TRPP has been proposed in Beraldi et al. (2019), where chance constraints are used to limit the probability of not achieving a target profit.

To the best of our knowledge, none of the above-mentioned papers deal with the more difficult case of the k-TRPP, even in the deterministic setting. Considering both a fleet of vehicles and the presence of profits, recently Bruni et al. (2018b, 2019c) studied an extension of the k-TRPP under risk. The novelty of the problem motivated, in the present paper, the investigation of suitable upper bounds and the design of an efficient metaheuristic approach. In the following Sections, we will present our approach.

3. Problem definition and upper bounds

Let us denote by $G = (V, E)$ a complete undirected graph. $V = \{1, \ldots, n\}$ denotes the set of customers, and 0 represents a depot where a homogeneous fleet of k vehicles is located. $E = \{(ij) \subseteq V \times V\}$ is the set of edges whose travel time is random. We assume to know the expected travel time and the variance associated with edge, denoted by μ_{ij} and σ_{ij}^2, respectively. We denote by π_i the revenue associated to each customer $i \in V$ and we assume that the profit collected by visiting a customer decreases with the random arrival time, expressed as a function of the edge travel times (in line with Dewilde et al., 2013). The problem is to find k disjoint tours starting at the depot and covering a subset of customers, which maximize the collected random profit.

We address the problem under a risk-averse perspective. In this case, the decision-maker is willing to trade-off some profit against less risky solutions. In order to represent different risk attitudes, from the risk-neutral to the more risk-adverse one, we adopt a mean-risk approach, striking the balance between two conflicting objectives, i.e. the expected profit maximization and the risk minimization. The resulting model (reported in the Appendix in Section A.1) presents indeed a bi-objective structure, following the classical mean-risk framework proposed by Markowitz (1952), in the portfolio optimization theory, where we account for risk by controlling the standard deviation (Krokhmal et al., 2011; Rockafellar, 2007). We should mention that there is relatively little literature in the routing devoted to this issue and no general consensus on the most appropriate risk measure on the context of network routing. The mean risk approach proposed in this paper is very general, since it can be applied whenever the first and the second moments of the random travel times are known, regardless the specific distribution function we consider. This assumption is reasonable since it is unlikely that the full distribution of travel times, if ever available, is known.

In what follows, we specify how the expected profit and the variance can be computed for a given path.

Let $(0, [1], \ldots, [q], \ldots, [\mathsf{l}])$, be a generic path. here $[q]$ denotes the index of the node in position q in the path, and $\mathsf{l} = 1, \ldots, n$ a possible path length. We define the expected arrival time function at node $[p]$ as $\bar{t}_{[p]} = \sum_{q=0}^{p-1} \mu_{[q]\ [q+1]}$. The associated variance is $\hat{t}_{[p]} = \sum_{q=0}^{p-1} \sigma^2_{[q]\ [q+1]}$. Hence, the expected profit collected on the path is $\sum_{q=0}^{\mathsf{l}} (\pi_{[q]} - \bar{t}_{[q]})$ and the corresponding variance is $\sum_{q=0}^{\mathsf{l}} \hat{t}_{[q]}$. Note that the latency of a tour can also be written in the (often more insightful) form $\sum_{q=0}^{\mathsf{l}} \bar{t}_{[q]} = \sum_{q=0}^{\mathsf{l}} \mu_{[q]\ [q+1]}(\mathsf{l} - q + 1)$.

Since the decision-maker is risk-averse, the problem does not merely entail the maximization of the expected profit, but also considers the travel time variability.

Adopting this framework, the mean-risk profit function (assuming for the ease of exposition a single path) reads as follows.

$$\lambda \left[\sum_{q=0}^{\mathsf{l}} \pi_{[q]} - \bar{t}_{[q]} \right] - (1 - \lambda) \sqrt{\sum_{q=0}^{\mathsf{l}} \hat{t}_{[q]}}. \tag{1}$$

The weighting parameter $\lambda \in [0,\ 1)$ reflects different risk-averse preferences of the decision maker.

To assess the importance of incorporating travel time fluctuations in the problem, we present in Fig. 1 a fictitious small example, including five potential nodes to be visited by a fleet of two homogeneous vehicles. The ordered pair of over each edge represents the expected travel time and the standard deviation over the edge, respectively. The value under each node represents the revenue collected while visiting the node.

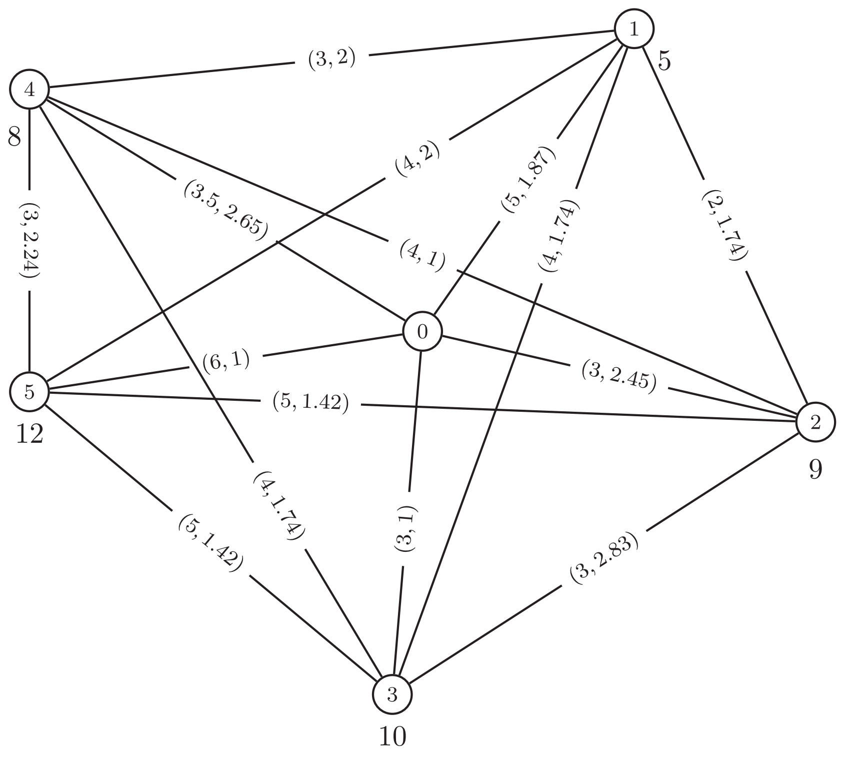

Fig. 1. Five nodes network.

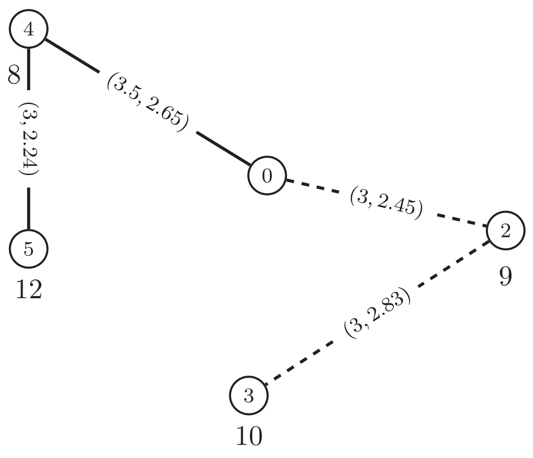

Fig. 2. Optimal paths in the risk-neutral model.

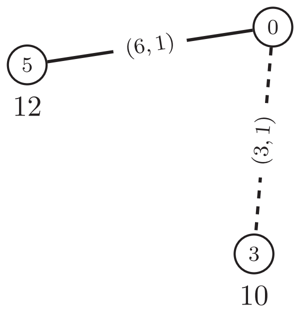

Fig. 3. Optimal paths in the risk-averse model.

Table 1
Simulation results.

	Mean	*Median*	*Max*	*Min*	*SD*
Risk-averse Model	12.99	13.11	17.33	2.34	1.42
Risk-neutral Model	20.01	21.58	35.70	−104.13	8.07

Considering only the expected travel time in the objective function (1), without including the variance term ($\lambda = 1$), we come up with the risk-neutral model for which the optimal paths are shown in Fig. 2.

The total expected profit gained by visiting nodes 2, 3, 4 and 5 is 20, calculated as the sum of the profits collected over the path $0-2-3$ $(19 - 2 \times (3) + 1 \times (3) = 10)$, and over the path $0-4-5$ $(20 - 2 \times (3.5) + 1 \times (3) = 10)$. In a similar way, the total standard deviation is 8.06, calculated as $\sqrt{(2^2 \times (6) + 1^2 \times (8)) + (2^2 \times (7.02) + 1^2 \times (5.01))}$

Fig. 3 shows the optimal solution of the risk-averse model ($\lambda = 0.1$ in this case).

The total expected profit of visiting nodes 3 and 5 is $12 - (6) + 10 - (3) = 13$ and the total standard deviation is $\sqrt{1+1} = 1.41$, which is much lower that the standard deviation of the risk-neutral solution.

To evaluate the reliability of the optimal solutions of both the risk-neutral and the risk-averse models, we have run a Monte Carlo simulation, including 50,000 different scenarios generated using the lognormal distribution, to represent different possible realizations of the travel times values over each edge (ij). The total expected profit under each scenario is expressed as the total difference between the collected revenue at a node and the realized arrival time to reach that node. Table 1 reports the arithmetic mean, the median, the max and min values, as well as the standard deviation (SD) of the simulation results. As expected, in terms of the total expected profit, the risk-neutral case provides better results than the risk-averse model, (supported by the higher *Mean*, *Median*, and the *Max* values reported for the risk-neutral model). The long range $Max - Min = 139.8$ and the high standard deviation ($SD = 8.07$) show the unstable behaviour of the risk-neutral model, compared with the risk-averse one with range of 14.99 and *SD* of 1.42. Figs. 4 and 5 report the frequency histograms of the simulated profits for the risk-neutral and the risk-averse solutions, respectively.

The long tail in Fig. 4 shows that the solution of the risk-neutral model is highly unstable with high variations in the total profit. This is an expected behaviour, since the risk-neutral model, ignoring completely the variance, can lead to high profit spreads. The negative profit values (on the left of the vertical line) in Fig. 4 underline the probability of experiencing losses. This occurrence is avoided by adopting a more risk-averse approach (see Fig. 5).

Hence, for a risk-averse decision maker, traditional risk-neutral models, that are mainly concerned with the optimization of the expected total profit, are not good strategies, since they are insensitive to profit variations. The risk averse decision maker would rather prefer a lower but certain profit to a higher but uncertain one, reflecting the fact that the disutility of a loss is much greater than the utility of an equivalent gain.

3.1. Upper bounds

The TRPP is NP−hard even in the deterministic case. The injection of uncertainty increases the complexity of the problem, preventing the exact solution within a reasonable time limit for large instances. In what follows, we present three upper bounds based on those proposed for the CumVRP by Ngueveu et al. (2010). The idea is to sort the customers in descending profit order and the edges in ascending expected travel time and variance order. Let $\bar{\pi}_i$ be the ith highest revenue assigned to a node and $\bar{\mu}_e$ and $\bar{\sigma}_e^2$ be the eth lowest travel time and the eth lowest variance among all the edges, respectively.

To derive a first upper bound, we first assume that l customers are directly served from the depot. We notice that the number of visited nodes $l = k, \ldots, n$, (since all the k vehicles are used and each vehicle serves at least one customer) is not known a priori and hence, different upper bound values ub_l^1 are separately evaluated. Considering the closest edges incident to the depot (in this case the index used will be e'), the upper bound is calculated as follows.

$ub_l^1 \;=\; \lambda[\sum_{i=1}^{l} \bar{\pi}_i - \sum_{e'=1}^{l} \bar{\mu}_{e'}] - (1-\lambda)\sqrt{\sum_{e'=1}^{l} \bar{\sigma}_{e'}^2}$ The final upper bound is $UB^1 \;=\; \max_{l=k}^{n} ub_l^1$.

In the second upper bound, the position of each edge in the path is considered for the evaluation of the arrival time. In fact, the travel time of the first edge appears in the evaluation of the arrival times of all the nodes serviced in path. For instance with $k = 1$ and $l = 5$ (five nodes serviced) the arrival time of the edge in the first position ($e = 1$) is counted $l + k - e$ times. When we consider k vehicles, the average number of times each edge is counted in a balanced solution, servicing l customers, is $\lceil \frac{k+l-e-(l \bmod k)}{k} \rceil$, where the term $\frac{k+l-e}{k}$ accounts for the number of times the edge in position e is counted, and the term $\frac{(l \bmod k)}{k}$ is for balancing the vehicle paths. The second upper bound can be computed as

$$ub_l^2 = \lambda\Big[\sum_{i=1}^{l} \bar{\pi}_i - \sum_{e=1}^{l} \Big\lceil \frac{k+l-e-(l \bmod k)}{k} \Big\rceil \bar{\mu}_e\Big]$$
$$-(1-\lambda)\sqrt{\sum_{e=1}^{l} \Big(\Big\lceil \frac{k+l-e-(l \bmod k)}{k} \Big\rceil\Big)^2 \bar{\sigma}_e^2}$$

and $UB^2 \;=\; \max_{l=k}^{n} ub_l^2$.

The third upper bound is computed by differentiating the set of edges incident to the depot (e') from those which are not incident

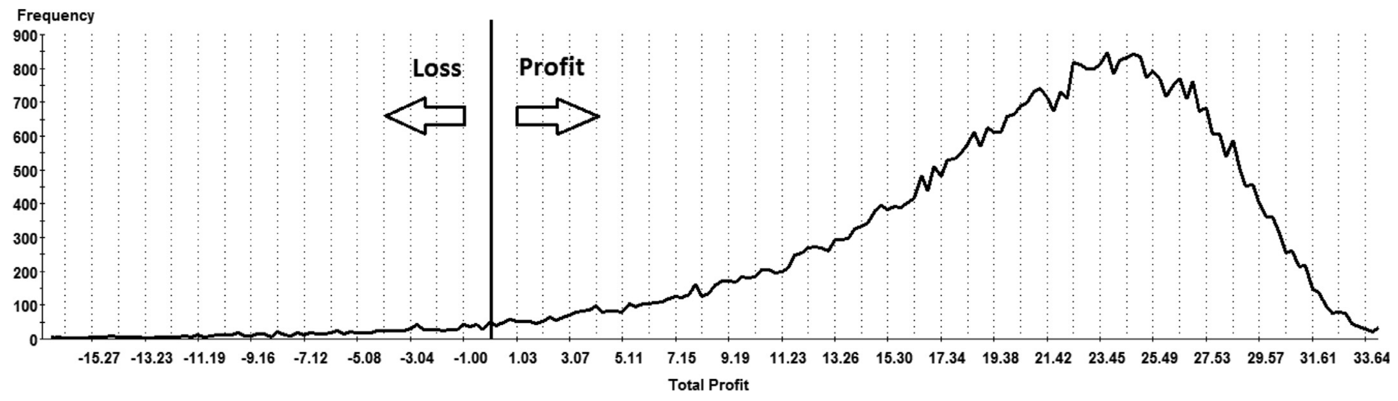

Fig. 4. Frequency histograms: Risk-neutral model.

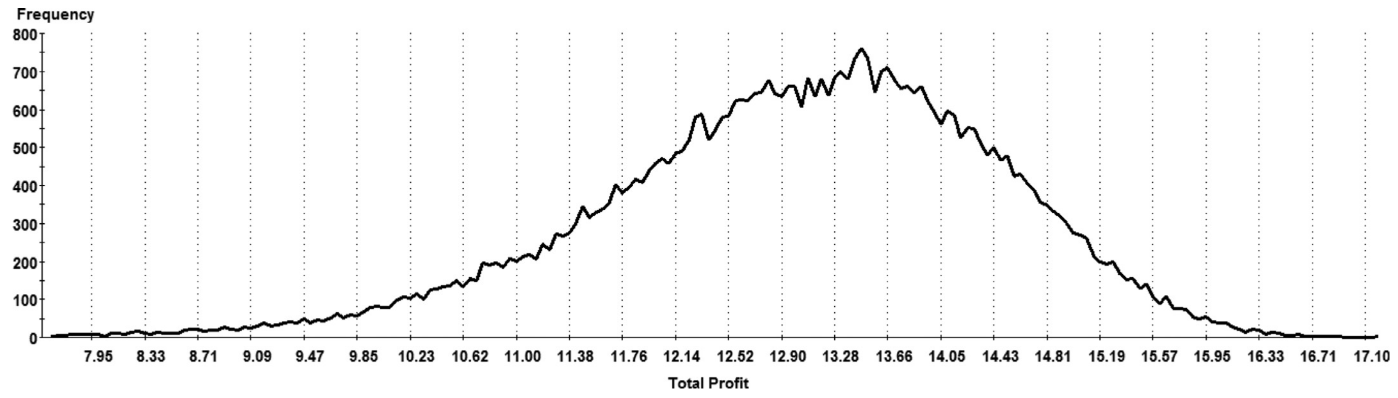

Fig. 5. Frequency histograms: Risk-averse model.

to the depot (e'').

$$ub_l^3 = \lambda \sum_{i=1}^{l} \bar{\pi}_i - \lambda\Big(\sum_{e'=1}^{k} \Big\lceil \frac{k+l-e'-(l \bmod k)}{k} \Big\rceil \bar{\mu}_{e'} + \sum_{e''=1}^{l-k} \Big\lceil \frac{l-e''-(l \bmod k)}{k} \Big\rceil \bar{\mu}_{e''}\Big)$$
$$-(1-\lambda)\sqrt{\sum_{e'=1}^{k} \Big(\Big\lceil \frac{k+l-e'-(l \bmod k)}{k} \Big\rceil\Big)^2 \bar{\sigma}_{e'}^2 + \sum_{e''=1}^{l-k} \Big(\Big\lceil \frac{l-e''-(l \bmod k)}{k} \Big\rceil\Big)^2 \bar{\sigma}_{e''}^2}$$

and $UB^3 = \max_{l=k}^{n} ub_l^3$.

These upper bounds will be used to evaluate the quality of the solutions generated by the heuristic method that we shall present in the next Section.

4. The hybrid reactive greedy randomized adaptive search heuristic

In this Section, we present a hybrid heuristic, based on an iterative multi-start algorithm called greedy randomized adaptive search heuristic (GRASP), first introduced by Feo and Bard (1989). In our scheme, the GRASP is made reactive (Guerriero et al., 2013; Prais and Ribeiro, 2000), allowing to self-tune some parameters during the iterations of the algorithm.

The Reactive GRASP (RGRASP) basically consists of a loop repeated until a termination criterion is met, embedding a construction phase. A local search phase is then applied to improve the solution provided by the first phase. The general structure of the RGRASP, is reported in Algorithm 1 . Let $\Gamma = \{\alpha_1, \ldots, \alpha_M\}$ a discrete set of values for the value of the parameter α, which is used inside the construction phase ($GRASP(\alpha)$) to control the randomness of the GRASP. During the iterations of the algorithm, the probability of picking a given $\alpha_m,\ m = 1, \ldots, M$ value is self-tuned (lines 8–15), making the scheme reactive. In particular, every G_{upd} iterations, the probabilities $p(\alpha_m),\ m = 1, \ldots, M$ associated to the values of $\alpha \in \Gamma$ are updated, to increase the probability associated to the most successful α values.

In the construction phase, described in the Algorithm 2 , a feasible solution is built, composed by a set $R = \{\rho_1, \ldots \rho_k\}$ of routes.

Algorithm 1: General structure of the RGRASP.

```
1  Input: $G_{upd}$, $\Gamma = \{\alpha_1, \cdots, \alpha_M\}$
2  Initialization: $Z_{\max} = -\infty$; $S_{\max} = null$; $s(\alpha_m) = null$, $p(\alpha_m) = \frac{1}{M}$, $\bar{z}_m = 0$, $z_m = 0$, $\gamma_m = 0\ \forall m = 1, \ldots, M$, $g = 1$.
3  while the termination criterion is not met do
4      Pick randomly $\alpha_{m*} \in \{\alpha_1, \ldots, \alpha_M\}$ according to probabilities $p(\alpha_m)$, $m = 1 \ldots, M$
5      $s(\alpha_{m*}) \leftarrow$ GRASP($\alpha_{m*}$);
6      Apply a local search obtaining a new solution $s*$ with value $z(s*)$
7      $z_{m*} \leftarrow (z_{m*} + z(s*))$, $\gamma_{m*} \leftarrow (\gamma_{m*} + 1)$
8      if ($g$ mod $G_{upd} == 0$) then
9          for ($m = 1$, $m \le M$, $m++$) do
10             if $\gamma_m > 0$ then
11                 $\bar{z}_m = \frac{z_m}{\gamma_m}$
12                 $p(\alpha_m) = \frac{1/\bar{z}_m}{\sum_{m=1}^{M} \frac{1}{\bar{z}_m}}$
13             end
14         end
15     end
16     if $z(s*) > Z_{\max}$ then
17         $S_{\max} := s*$
18         $Z_{max} := z(s*)$
19     end
20     $g+ = 1$
21 end
22 Return: $Z_{max}$, $S_{max}$
```

At the beginning, all the nodes are candidate to be inserted, and are therefore collected into the set $\mathfrak{U}$ of unvisited nodes. The nodes are then sorted in ascending order with respect to the following

Algorithm 2: GRASP(α): the construction phase.

```
1  Input: A value of $\alpha \in \Gamma$
2  Initialization: $s(\alpha) = null$ ; $\mathfrak{U} = V$; $CL = \mathfrak{U}$; $RCL = \emptyset$ ;
   $\Delta_j = -\infty, \forall j \in V,\ \rho_1 = \dots = \rho_k = \{0\}$
3  Sort $\mathfrak{U}$ in ascending order with respect to $\frac{\lambda \mu_{0i} + (1-\lambda)\sigma^2_{0i}}{\lambda p_i}$
4  for $\rho \in R$ do
5      $c \leftarrow argmin_{i \in \mathfrak{U}}(\frac{\lambda \mu_{0i} + (1-\lambda)\sigma^2_{0i}}{\lambda p_i})$
6      Insert $c$ in route $\rho$ in the best position
7      $\mathfrak{U} \leftarrow \mathfrak{U} \setminus \{c\}$
8  end
9  while $\mathfrak{U} \neq \{\emptyset\}$ do
10     for $j \in \mathfrak{U}$ do
11         Set $\Delta_j = 0$
12         for $\rho \in R$ do
13             for $i \in \{1, 2, \dots, length(\rho)\}$ do
14                 Evaluate $\delta(j, i, \rho)$
15                 $\Delta_j \leftarrow (\Delta_j + \delta(j, i, \rho))$
16             end
17         end
18     end
19     Sort $CL$ in decreasing order based on $\Delta_j$
20     $\Delta_{min} = \min_{j \in CL} \Delta_j$, $\Delta_{max} = \max_{j \in CL} \Delta_j$
21     $\hat{\Delta} = \Delta_{max} - \alpha(\Delta_{max} - \Delta_{min})$
22     for $(j = 1,\ j \leq CL.size,\ j++)$ do
23         if $(CL[j] \geq \hat{\Delta})$ then
24             $RCL \leftarrow RCL \cup CL[j]$
25         end
26     end
27     Choose a random element $\Delta_{j^*}$ in $RCL$
28     Insert node $j^*$ in position $i^*$ in route $\rho^*$
29     $\mathfrak{U} \leftarrow \mathfrak{U} \setminus \{j^*\}$
30 end
31 $s(\alpha) \leftarrow R$
32 Return $s(\alpha)$
```

criterion $\frac{\lambda \mu_{0i} + (1-\lambda)\sigma^2_{0i}}{\lambda p_i}$ which accounts for both the distance from the depot and the collected profit. One node is then inserted into each vehicle route, following the ordering criterion. Then, for each unvisited node $j \in \mathfrak{U}$, the possibility of being inserted in any position i over any route ρ is evaluated on the basis of its contribution into the objective function, denoted with $\delta(j, i, \rho)$. After the evaluation process, the aggregated contribution $\Delta_j = \sum_{\rho, i} \delta(j, i, \rho)$ as well as the best position i^* and the best route ρ^* are stored and all the eligible nodes are put into the list of candidate nodes, denoted by CL. The elements in CL are sorted following a decreasing order since, obviously, the nodes with higher aggregated contribution are more promising than the others. A *restricted candidate list (RCL)* is then built (lines 22-26 of Algorithm 2) by considering all the elements whose values are above some threshold $\hat{\Delta} = \Delta_{max} - \alpha(\Delta_{max} - \Delta_{min})$, where $\alpha \in [0,\ 1]$ and $\Delta_{min} = \min_{j \in CL} \Delta_j$ and $\Delta_{max} = \max_{j \in CL} \Delta_j$ are, respectively, the minimum and the maximum value attained by the candidates. The parameter α controls the size of RCL: a value of $\alpha = 0$ corresponds to the deterministic greedy algorithm, in which always the node associated with the best value is chosen. By increasing the value of α, more candidates are added to the RCL, improving the chances of escaping from local optimality. Once a random element is chosen from the RCL and the corresponding node is inserted in the best position in the best route (lines 27-29 of Algorithm 2), the candidate nodes are re-evaluated (this is the adaptive component of the heuristic, and reflects the changes implied by the selection of the previous element) and a new RCL is built again. Algorithm 2 ends when all the nodes have been inserted and returns a solution $s(\alpha)$ representing the set of routes built in this phase.

4.1. The adaptive local search

The local search phase to be effective must be carefully designed in order to deeply investigate the portion of the solution space under investigation, exploiting the specific problem structure. The problem presents, indeed, a two-level configuration, where the first level corresponds to a node selection problem, whereas the second one to a cumulative routing problem over the selected nodes. This suggests to adopt a heuristic method that systematically applies two different exploration mechanisms, one for modifying the set of served nodes– since it is not mandatory to serve all of them– and the other one to assign and order the nodes in different routes.

In the first level, since even a minor modification on the nodes to be selected might deeply affect the final result, we have implemented a systematic destroy mechanism. More in detail, the number of nodes to be served is iteratively reduced, discarding the node with the worst contribution into the objective function, evaluated as the difference between the objective function with and without the node. Whenever a node is deleted, the route is repaired by connecting directly the two ending nodes. The process terminates whenever the number of nodes in the current solution s (denoted with $|s|$) falls below a given number, set to $\lceil 0.75n \rceil$.

An adaptive local search is then applied in the second level, where different moves are used to systematically change the neighbourhood of the search. A scheme of the adaptive neighbourhood search heuristic is shown in Algorithm 3 .

Algorithm 3: Scheme of the local search.

```
1  Input: The current solution $s^*$, $T$
2  Initialization: $p_\nu = \frac{1}{\mathfrak{N}},\ \forall \nu = 1, \dots \mathfrak{N}$
3  repeat
4      for $t = 1$ to $T$ do
5          if $(t > warm\text{-}up\ \ period)$ then
6              Update the selection probabilities $p_\nu,\ \forall \nu = 1, \dots, \mathfrak{N}$
7          end
8          Select a Neighbourhood $\hat{\nu}$ according to the
           probabilities $p_\nu,\ \forall \nu = 1, \dots, \mathfrak{N}$
9          $s \leftarrow$ Local Search$(s^*, \hat{\nu})$
10         if $(s > s^*)$ then
11             $s^* \leftarrow s$
12         end
13         if No improvements after a certain number of Iterations
           then
14             Return $s^*$
15         end
16     end
17     Apply the destroy mechanism and let $i$ be the discarded
       nodeDelete node $i$ from $s*$ and repair the solution $s^*$
18 until $|s| \geq \lceil 0.75n \rceil$;
19 Return $s^*$
```

The algorithm performs T iterations where each iteration starts with the selection of a neighbourhood (chosen using the self-adaptive mechanism). Within the $warm - up\ \ period$, a roulette wheel mechanism is applied to determine which move to choose in each iteration. At the beginning, the selection probability is the same for all the neighbourhoods and set to $\frac{1}{\mathfrak{N}}$, where $\mathfrak{N}$ is the number of neighbourhoods, in such a way that all the neighbourhoods have an equal chance of being selected. The updated probability values p_ν are used for the neighbourhood selection procedure after the *warm − up period* and are dynamically updated using the following formula. $p_\nu = \frac{\bar{s}_\nu}{\sum_{\nu=1}^{\mathfrak{N}} \bar{s}_\nu}$ with $\bar{s}_\nu = \frac{s_\nu}{s_\nu + f_\nu} + \epsilon$. Here s_ν and f_ν, respectively, count the number of times the operator ν was successful and failed in improving the incumbent and ϵ is a small value added to provide all the neighbourhoods (even the unsuccessful ones with $s_\nu = 0$ and $f_\nu > 0$) with a chance. We have used five different operators. Since the objective function of the problem is non-separable and non-additive due to the presence of the standard deviation, we propose efficient methods to evaluate each move.

Intra-route exchange operation

This operator exchanges the position of a pair of nodes within a vehicle route. Every pair is evaluated and the best move is then applied. In order to evaluate the contribution of the move, let i and j be two positions along the route ρ and let $[i]$, $[j]$ indicate the node that occupies position i, j, respectively. Since within the local search phase no new node is added, and, hence, the profits collected are the same, we can simplify the difference in the objective function Δz as

$$\Delta z = -\lambda \Delta z_\mu - (1-\lambda)\left(\sqrt{z^{old}_{\sigma^2} + \Delta z_{\sigma^2}} - \sqrt{z^{old}_{\sigma^2}}\right) \tag{2}$$

where $z^{old}_{\sigma^2}$ indicates the value of the variance before the move, and Δz_μ and $\Delta z_{\sigma 2}$ are, respectively, the differences in terms of expected values and variances which are defined as follows.

$$\Delta z_\mu = \begin{cases} a+b+c+d, & j \neq length(\rho) \\ a+b+c, & \text{otherwise} \end{cases}$$

and

$$\Delta z_{\sigma^2} = \begin{cases} \acute{a}+\acute{b}+\acute{c}+\acute{d}, & j \neq length(\rho) \\ \acute{a}+\acute{b}+\acute{c}, & \text{otherwise} \end{cases}$$

where

$$\begin{aligned}
a &= (length(\rho) - i + 1)\left(\mu_{[i-1][j]} - \mu_{[i-1][i]}\right),\\
b &= (length(\rho) - i)\left(\mu_{[j][i+1]} - \mu_{[i][i+1]}\right),\\
c &= (length(\rho) - j + 1)\left(\mu_{[j-1][i]} - \mu_{[j-1][j]}\right),\\
d &= (length(\rho) - j)\left(\mu_{[i][j+1]} - \mu_{[j][j+1]}\right),\\
\acute{a} &= (length(\rho) - i + 1)^2\left(\sigma^2_{[i-1][j]} - \sigma^2_{(i-1)i}\right),\\
\acute{b} &= (length(\rho) - i)^2\left(\sigma^2_{[j][i+1]} - \sigma^2_{[i[[i+1]}\right),\\
\acute{c} &= (length(\rho) - j + 1)^2\left(\sigma^2_{[j-1][i]} - \sigma^2_{[j-1][j]}\right),\\
\acute{d} &= (length(\rho) - j)^2\left(\sigma^2_{[i][j+1]} - \sigma^2_{[j][j+1]}\right).
\end{aligned}$$

2-opt neighbourhood This moves deletes two non-adjacent edges along the path and adds two other edges. Let consider two non-adjacent nodes $[i]$ and $[j]$; a 2−opt move is performed by deleting edges $([i], [i+1])$ and $([j], [j+1])$ and adding two new edges $([i], [j+1])$ and $([i-1], [j])$. While the difference in the objective function is evaluated using (2), the terms Δz_μ and Δz_σ are expressed as

$$\Delta z_\mu = \begin{cases} a_1+b_1+c_1, & j \neq length(\rho) \\ a_1+b_1, & \text{otherwise} \end{cases}$$

and

$$\Delta z_\sigma = \begin{cases} \acute{a}_1+\acute{b}_1+\acute{c}_1, & j \neq length(\rho) \\ \acute{a}_1+\acute{b}_1, & \text{otherwise} \end{cases}$$

where

$$\begin{aligned}
a_1 &= (length(\rho) - i + 1)\left(\mu_{[i-1][j]} - \mu_{[i-1][i]}\right),\\
c_1 &= (length(\rho) - j)\left(\mu_{[i][j+1]} - \mu_{[j][j+1]}\right),\\
b_1 &= \textstyle\sum_{k=0}^{j-i-1} (length(\rho) - (i+k))\left(\mu_{[j-k][j-k+1]} - \mu_{[i+k][i+k+1]}\right),\\
\acute{a}_1 &= (length(\rho) - i + 1)^2\left(\sigma^2_{[i-1][j]} - \sigma^2_{[i-1][i]}\right),\\
\acute{c}_1 &= (length(\rho) - j)^2\left(\sigma^2_{[i][j+1]} - \sigma^2_{[j][j+1]}\right),\\
\acute{b}_1 &= \textstyle\sum_{k=0}^{j-i-1} (length(\rho) - [i+k])^2\left(\sigma^2_{[j-k][j-k+1]} - \sigma^2_{[i+k][i+k+1]}\right),
\end{aligned}$$

Or− **opt neighborhood**

This move deletes any triple of non-adjacent edges within the route and reconnects them by adding three new edges such that the order of the other edges over the path is preserved. Let $([i], [i+1])$, $([j], [j+1])$, and $([k], [k+1])$ be a triple of edges such that $j - i > 1$, $k - j > 1$. An *or*−opt move is performed by deleting the aforementioned edges and by inserting three new edges $([i], [j+1])$, $([j], [k+1])$, and $([k], [i+1])$. The difference terms of Δz_μ and Δz_σ can be calculated as follows.

$$\Delta z_\mu = \begin{cases} a_1+b_1+c_1-d_1+e_1+f_1, & j \neq length(\rho) \\ a_1+b_1+c_1+e_1+f_1, & \text{otherwise} \end{cases}$$

and

$$\Delta z_\sigma = \begin{cases} \acute{a}_1+\acute{b}_1+\acute{c}_1-\acute{d}_1+\acute{e}_1+\acute{f}_1, & j \neq length(\rho) \\ \acute{a}_1+\acute{b}_1+\acute{c}_1+\acute{e}_1+\acute{f}_1, & \text{otherwise} \end{cases}$$

where

$$\begin{aligned}
a_1 &= (length(\rho) - i)\left(\mu_{[i]][j+1]} - \mu_{[i][i+1]}\right),\\
b_1 &= (length(\rho) - (i+k-j))\mu_{[k][i+1]},\\
c_1 &= (length(\rho) - j)\mu_{[j][j+1]},\\
d_1 &= (length(\rho) - k)\left(\mu_{[j][k+1]} - \mu_{[k][k+1]}\right),\\
e_1 &= \textstyle\sum_{s=1}^{k-j-1} (j-i)\mu_{[j+s][j+s+1]},\\
f_1 &= \textstyle\sum_{s=1}^{j-i-1} (k-j)\mu_{[i+s][i+s+1]},\\
\acute{a}_1 &= (length(\rho) - i)^2\left(\sigma^2_{[i]][j+1]} - \sigma^2_{[i][i+1]}\right),\\
\acute{b}_1 &= (length(\rho) - (i+k-j))^2\sigma^2_{[k][i+1]},\\
\acute{c}_1 &= (length(\rho) - j)^2\sigma^2_{[j][j+1]},\\
\acute{d}_1 &= (length(\rho) - k)^2\left(\sigma^2_{[j][k+1]} - \sigma^2_{[k][k+1]}\right),\\
\acute{e}_1 &= \textstyle\sum_{s=1}^{k-j-1} (i-j)(i+j+2(s - length(\rho))) \qquad \sigma^2_{[j+s][j+s+1]},\\
\acute{f}_1 &= \textstyle\sum_{s=1}^{j-i-1} (k-j)(j-k+2(length(\rho) - (i+s))\ \sigma^2_{[i+s][i+s+1]},
\end{aligned}$$

Inter-route exchange neighborhood

This move exchanges a pair of nodes belonging to two different paths. Consider two routes of $\rho_1, \rho_2 \in R$, respectively, and two nodes $[i]$ in ρ_1 and $[j]$ in ρ_2. The terms of Δz_μ and Δz_σ are computed as follows.

$$\Delta z_\mu = \begin{cases} a_1+b_1+a_2+b_2, & i \neq length(\rho_1), j \neq length(\rho_2) \\ a_1+a_2+b_2, & i = length(\rho_1), j \neq length(\rho_2) \\ a_1+b_1+a_2, & i \neq length(\rho_1), j = length(\rho_2) \\ a_1+a_2, & \text{otherwise} \end{cases}$$

and

$$\Delta z_\sigma = \begin{cases} \acute{a}_1+\acute{b}_1+\acute{a}_2+\acute{b}_2, & i \neq length(\rho_1), j \neq length(\rho_2) \\ \acute{a}_1+\acute{a}_2+\acute{b}_2, & i = length(\rho_1), j \neq length(\rho_2) \\ \acute{a}_1+\acute{b}_1+\acute{a}_2, & i \neq length(\rho_1), j = length(\rho_2) \\ \acute{a}_1+\acute{a}_2, & \text{otherwise} \end{cases}$$

where

$$\begin{aligned}
a_1 &= (length(\rho_1) - i + 1)\left(\mu_{[i-1][j]} - \mu_{[i-1][i]}\right),\\
b_1 &= (length(\rho_1) - i)\left(\mu_{[j][i+1]} - \mu_{[i][i+1]}\right),\\
a_2 &= (length(\rho_2) - j + 1)\left(\mu_{[j-1][i]} - \mu_{[j-1][j]}\right),\\
b_2 &= (length(\rho_2) - j\left(\mu_{[i][j+1]} - \mu_{[j][j+1]}\right),\\
\acute{a}_1 &= (length(\rho_1) - i + 1)^2\left(\sigma^2_{[i-1][j]} - \sigma^2_{[i-1][i]}\right),\\
\acute{b}_1 &= (length(\rho_1) - i)^2\left(\sigma^2_{[j][i+1]} - \sigma^2_{[i][i+1]}\right),\\
\acute{a}_2 &= (length(\rho_2) - j + 1)^2\left(\sigma^2_{[j-1][i]} - \sigma^2_{[j-1][j]}\right),\\
\acute{b}_2 &= (length(\rho_2) - j)^2\left(\sigma^2_{[i][j+1]} - \sigma^2_{[j][j+1]}\right),
\end{aligned}$$

Delete-Insert neighborhood

This procedure deletes a node from a path and adds it to an other one.

$$\Delta z_\mu = \begin{cases} -a_1 - b_1 + c_1 + a_2 + b_2 + c_2, & i \neq length(\rho_1) \\ -a_1 - b_1 + a_2 + b_2 + c_2, & \text{otherwise} \end{cases}$$

and

$$\Delta z_\sigma = \begin{cases} \acute{a}_1 + \acute{b}_1 + \acute{c}_1 + \acute{a}_2 + \acute{b}_2 + \acute{c}_2, & i \neq length(\rho_1) \\ \acute{a}_1 + \acute{b}_1 + \acute{a}_2 + \acute{b}_2 + \acute{c}_2 & \text{otherwise} \end{cases}$$

where

$$\begin{aligned}
a_1 &= \textstyle\sum_{k=0}^{i-2} \mu_{[k][k+1]}, \\
b_1 &= (length(\rho_1) - i + 1)\mu_{[i-1][i]}, \\
c_1 &= (length(\rho_1) - i)\big(\mu_{[i-1][i+1]} - \mu_{[i][i+1]}\big) \\
\acute{a}_1 &= \textstyle\sum_{k=0}^{i-2} (1 - 2(length(\rho_1) - k))\sigma^2_{[k][k+1]}, \\
\acute{b}_1 &= (length(\rho_1) - i + 1)^2\sigma^2_{[i-1][i]}, \\
\acute{c}_1 &= (length(\rho_1) - i)^2\big(\sigma^2_{[i-1][i+1]} - \sigma^2_{[i][i+1]}\big), \\
a_2 &= \textstyle\sum_{k=0}^{j-2} \mu_{[k][k+1]}, \\
b_2 &= (length(\rho_2) - j + 1)\big(\mu_{[i][j]} - \mu_{[j-1][j]}\big), \\
c_2 &= (length(\rho_2) - j + 2)\mu_{[j-1][i]}, \\
\acute{a}_2 &= \textstyle\sum_{k=0}^{i-2} (1 + 2(length(\rho_2) - k))\sigma^2_{[k][k+1]}, \\
\acute{b}_2 &= (length(\rho_2) - j + 1)^2\big(\sigma^2_{[i][j]} - \sigma^2_{[j-1][j]}\big), \\
\acute{c}_2 &= (length(\rho_2) - j + 2)^2\sigma^2_{[j-1][i]}.
\end{aligned}$$

Different move strategies can be implemented within the local search (Osman (1993)), namely the best admissible (BA), the first best admissible (FBA) and the least best admissible (LBA). In the BA strategy, the best admissible improving solution within the neighborhood, if any, is chosen to update the incumbent solution, otherwise the incumbent is retained. In the FBA strategy, the first improving solution within the neighborhood is selected, if present, otherwise the incumbent is retained. The LBA strategy allows the move to the least improving neighbour solution, if any, otherwise the incumbent is not updated. The results of the FBA strategy are in general superior, in terms of computational time, to those provided by the BA and LBA approaches, but the LBA provides more chance to escape from local optimality, by moving to those solutions with the smallest improvements. for this reason, to obtain good quality solutions, we have applied the LBA strategy in the computational experiments.

5. Computational results

We have tested the heuristic approach on three set of instances, including the P-instances (Augerat et al., 1995), the E-instances (Christofides and Eilon, 1969), and the largest CMT-instances (Christofides et al., 1979; Ngueveu et al., 2010). The number of customers (vehicles) in the data sets varies from $15-75$ $(2-15)$, $21-75$ $(3-14)$, and $100-199$ $(7-17)$, respectively. The expected travel time μ_{ij} over the edge (ij) has been set equal to the Euclidean distance between node i and j and the travel time variance σ^2_{ij} has been computed as $\lceil(\mu_{ij} b)^2\rceil$, where b is a random number uniformly distributed in the interval [0.1,0.32).

In order to generate proper profit values which are proportional to the distance, we have slightly modified the formula proposed in Dewilde et al. (2013) in order to account for the stochastic case by considering

$$p_i \sim U\left(\min_{i=1\ldots n}\left(\mu_{0i} - \beta\sqrt{\sigma^2_{0i}}\right), \frac{n}{2}\max_{i=1\ldots n}\left(\mu_{0i} + \beta\sqrt{\sigma^2_{0i}}\right)\right]$$

where β is a random number within the interval (0, 1]. Obviously, the profit assigned to the depot is zero.

The proposed metaheuristic was coded in C++ and the mathematical model was solved using the open source SCIP library, release 3.2.0. The parameters G_{upd} and T have been set to 10 and 1000, respectively. The $warm-up\ period$ has been set to 10 iterations and the local search in the second level is stopped after 20 iterations without improvements. All the experiments have been performed on an Intel® Core™ i7 2.90 GHz, with 8.0 GB of RAM memory, running under Windows operating system.

5.1. Results for the P-instances

The first set of experiments is devoted to the presentation of the results obtained by running the GRASP (Algorithm 2), for each value of $\alpha \in \Gamma = \{0, 0.1, \cdots, 0.9\}$. The GRASP is compared with the earlier approach recently proposed in the conference paper Bruni et al. (2018b) corresponding to the state-of-the art approach (SoA, for short). The SCIP solver, with a time limit of one hour, was used with the aim of finding (near) optimal solutions of the model (3)-(12) reported in the Appendix. Not for all the instances SCIP was able to optimally solve the problem, hence, we have reported its percentage optimality gap *Opt%* or, when any feasible solution is found within the time limit, a dash. The computational time in seconds of the two heuristics and of SCIP have been reported in the dedicated columns of the result tables with headings *CPU*. For the GRASP, the average, the maximum and the minimum value over the ten values of $\alpha \in \Gamma$ have been reported. To assess the performance of the heuristic algorithms, we have reported the percentage gaps *Gap%* evaluated considering the difference between the objective function value found by SCIP ($SCIP^{OF}_{3600}$) and by the heuristics RGRASP and GRASP ($(R)GRASP^{OF}$) (i.e. $Gap\% = \frac{(SCIP^{OF}_{3600} - (R)GRASP^{OF})}{SCIP^{OF}_{3600}} \times 100$. When the heuristic outperforms SCIP, the objective function value found by the heuristic is greater than the one found by SCIP (we have a maximization problem) and the gap is negative. For the GRASP, the average, the maximum and the minimum gap over the ten values of $\alpha \in \Gamma$ have been reported.

Tables 2–4 summarize the heuristic results obtained with different values of the trade-off parameter λ taken from the set $\{0.1, 0.5, 0.9\}$. Columns 1, 2, and 3 refer to the name of the instances, the number of nodes and the number of vehicles, respectively. A first general observation is that the results are influenced by the parameter λ: by decreasing its value more weight is put on the non-linear part of the objective function, reflecting a risk-averse behavior of the decision-maker. This claim is supported by the increase in the computational time spent by SCIP and in the number of instances solved to optimality within the time limit.

Analysing the performance of the GRASP reported in columns 6-11, we observe that the best solution obtained by the GRASP (which corresponds to a given value of α) in some cases corresponds to the optimal solution and, in general, is better than the one obtained by the SoA algorithm. As far as the solution time is concerned, we should consider that the algorithm should be run for all the ten values of α and therefore, the total time spent is the average time (for instance, 14.85 seconds for $\lambda = 0.1$) multiplied by ten. The worst solution can deteriorate the quality of the solution considerably, especially for $\lambda = 0.1$ and the deviation can be remarkable. For example, for the instance Pn21k2, the maximum gap is 9.6% and the difference between the maximum and the minimum gap found by the GRASP over ten runs with different values of α is 9.18%. On average the difference is around 2.73% for $\lambda = 0.1$, and around 1.5% for the other values of λ.

The SoA approach proves to be successful in terms of time efficiency, with an average *CPU* time of less than 30 seconds, but the solution quality is not satisfactory especially for $\lambda = 0.1$ (the average gap in this case is 6.73%). For increasing λ values the SoA approach is performing better and its gap exhibits a downward trend,

Table 2
GRASP versus SoA: P-Instances, $\lambda = 0.1$.

			SoA		GRASP						SCIP	
			CPU		*CPU*						*CPU*	*Opt%*
Instance	*n*	*k*			Avg	Max	Min	Avg	Max	Min		
Pn16k8	15	5	0.23	10.28	0.32	0.37	0.24	1.49	1.49	1.49	3.59	0
Pn19k2	18	2	1.07	6.36	1.00	1.18	0.82	1.82	7.71	0.06	360.91	0
Pn20k2	19	2	1.18	4.56	1.10	1.19	1.01	0.72	1.66	0.05	471	0
Pn21k2	20	2	1.52	9.01	1.28	1.46	1.15	5.75	9.6	0.42	822.1	0
Pn22k2	21	2	2.13	4.27	1.59	1.92	1.37	1.92	6.57	0.00	41.9	0
Pn22k8	21	8	0.77	4.97	0.59	0.62	0.57	1.79	1.79	1.79	14.3	0
Pn23k8	22	8	0.8	2.78	0.91	0.97	0.85	0.93	1.16	0.38	8.13	0
Pn40k5	39	5	10.18	2.65	7.29	7.72	6.54	1.03	1.59	0.24	3600	7.87
Pn44k5	44	5	16.08	3.79	9.13	10.08	7.9	1.09	2.79	0.15	3600	5.35
Pn50k7	49	7	16.36	9.57	11.68	13.23	10.37	2.49	3.62	1.29	3600	10.19
Pn50k8	49	8	17.8	9.2	10.37	12.94	7.84	2.33	5.51	0.99	3600	9.49
Pn50k10	49	10	17.72	6.13	11.31	12.72	9.75	1.58	2.53	0.71	697.51	0
Pn51k10	50	10	16.27	2.35	12.06	15.35	10.46	1.03	1.79	0.41	3600	2.15
Pn55k7	54	7	32.49	11.81	15.77	18.07	13.67	3.64	5.34	2.56	3600	9.36
Pn55k8	54	8	25.64	9.81	16.01	17.8	12.78	2.07	3.32	0.60	3600	10.03
Pn55k10	54	10	24.99	12.9	13.36	16.55	9.85	3.02	5.61	1.38	156.49	0
Pn55k15	54	15	17.62	7.73	13.28	13.96	12.59	0.99	1.52	0.42	3600	7.82
Pn60k10	59	10	32.12	6.56	17.67	20.56	14.07	2.45	4.24	0.93	595.33	0
Pn65k10	64	10	43.74	4.3	20.81	23.12	17.82	1.30	2.6	-0.04	3600	8.02
Pn70k10	69	10	57.53	5.66	29.11	32.88	25.8	1.63	2.57	0.17	3600	7.58
Pn76k4	75	4	147.96	–	73.01	77.51	65.75	–	–	–	3600	–
Pn76k5	75	5	138.99	-	59.01	64.92	51.71	–	–	–	3600	–
Avg			28.33	6.73	14.85	16.60	12.86	1.96	3.65	0.70	2107.78	3.89

Table 3
GRASP versus SoA: P-Instances, $\lambda = 0.5$.

			SoA		GRASP						SCIP	
			CPU		*CPU*						*CPU*	*Opt%*
Instance	*n*	*k*			Avg	Max	Min	Avg	Max	Min		
Pn16k8	15	5	0.33	3.78	0.28	0.34	0.22	1.01	1.01	1.01	1.99	0
Pn19k2	18	2	0.96	2.48	0.94	1.32	0.8	0.541	0.86	0.18	16.13	0
Pn20k2	19	2	1.35	4.35	1.20	1.31	1.08	2.12	4.17	1.47	35.97	0
Pn21k2	20	2	1.62	6.71	1.32	1.44	1.16	0.604	1.63	0.00	32.49	0
Pn22k2	21	2	1.9	4.44	1.77	2	1.46	1.293	3.26	0.27	45.68	0
Pn22k8	21	8	0.54	2.32	0.65	0.67	0.63	0.09	0.09	0.09	8.7	0
Pn23k8	22	8	0.75	1.34	0.77	0.87	0.69	0.195	0.21	0.06	6.53	0
Pn40k5	39	5	11.1	2.7	6.86	8.12	5.9	1.069	1.84	0.63	921.45	0
Pn44k5	44	5	17.27	2.71	9.49	10.6	8.45	1.154	1.76	0.52	3600	0.56
Pn50k7	49	7	18.93	8.25	12.33	14.15	8.93	1.94	3.43	0.74	3600	0.75
Pn50k8	49	8	20.47	5.07	11.06	13	8.86	1.649	2.96	0.36	639	0
Pn50k10	49	10	17.49	4.76	11.73	12.42	11.02	1.343	2.11	0.33	3600	0.14
Pn51k10	50	10	15.75	1.35	11.85	16.55	8.7	0.476	0.63	0.37	104.93	0
Pn55k7	54	7	28.08	6.69	16.01	17.9	12.85	2.565	5.17	1.17	3600	0.66
Pn55k8	54	8	26.6	4.55	15.64	17.89	13.08	2.351	3.38	1.08	3600	0.53
Pn55k10	54	10	21.77	9.2	13.67	15.85	11.82	1.809	2.9	0.46	215.05	0
Pn55k15	54	15	17.28	4.37	13.49	14.56	11.88	1.027	1.42	0.76	85	0
Pn60k10	59	10	27.74	4.74	17.75	20.05	16.59	1.642	2.75	0.74	3600	0.42
Pn65k10	64	10	45.14	4.75	21.56	26.11	16.26	1.655	3.05	0.46	3600	0.45
Pn70k10	69	10	63.01	4.3	30.73	33.8	26.15	1.84	2.62	1.12	3600	0.46
Pn76k4	75	4	162.97	-56.45	73.80	81.58	69.93	-56.35	-56.02	-56.73	3600	58.49
Pn76k5	75	5	136.95	-	59.06	66.6	43.5	-	-	-	3600	-
Avg			29.00	1.54	15.09	17.14	12.73	-1.43	-0.51	-2.14	1732.41	2.97

going from 6.73% to -1.48% for $\lambda = 0.9$, when the variance term is almost not considered. This might suggest to limit the use of the SoA heuristic to risk-neutral problems.

The GRASP heuristic provides solutions whose quality is influenced by the choice of the parameter $\alpha \in \Gamma$ and is only one component of the RGRASP. To assess the effectiveness of the reactive mechanism and of the local search, we have carried out a second set of experiments comparing the solutions obtained by the RGRASP and SCIP within the same time limit *TL* (reported in seconds).

Tables 5–7 show the performance of the RGRASP heuristic. In order to have a better evaluation, we have also compared the results with the ones obtained by SCIP within the larger time limit of 3600 seconds. and already reported in Tables 2–4. In particular, the columns 5,6 and 7 provide the objective function values provided by SCIP within both the allotted time of one hour and the new time limit *TL* ($SCIP_{3600}^{OF}$ and $SCIP_{TL}^{OF}$, respectively), and by the RGRASP. In the Tables, the figures highlighted in bold correspond to the optimal objective function values. The last two columns report the corresponding gaps evaluated, respectively as $\frac{(SCIP_{3600}^{OF} - RGRASP^{OF})}{SCIP_{3600}^{OF}} \times 100$, and $\frac{(SCIP_{TL}^{OF} - RGRASP^{OFI})}{SCIP_{TL}^{OF}} \times 100$.

Table 5 shows the results for $\lambda = 0.1$, which is the most involved and risk-averse case. SCIP in one hour is able to find the optimal solution for 10 out of 22 instances. For these instances, the RGRASP is able to find the optimal solution in two cases (Pn19k2

Table 4
GRASP versus SoA: P-Instances, $\lambda = 0.9$.

			SoA		GRASP						SCIP	
			CPU		*CPU*						*CPU*	*Opt%*
Instance	*n*	*k*			Avg	Max	Min	Avg	Max	Min		
Pn16k8	15	5	0.28	3.47	0.35	0.42	0.25	0.85	0.85	0.85	1.96	0
Pn19k2	18	2	1.23	2.47	1.08	1.24	0.85	0.2	0.58	0.11	16.77	0
Pn20k2	19	2	1.46	2.32	1.21	1.36	1.05	0.43	1.07	0.00	19.62	0
Pn21k2	20	2	1.64	2.23	1.30	1.44	1.18	0.71	2.9	0.00	25.91	0
Pn22k2	21	2	2.12	3.37	1.73	1.93	1.55	0.72	3.56	0.20	25.89	0
Pn22k8	21	8	0.65	2.39	0.75	0.79	0.73	0.51	0.51	0.51	8.92	0
Pn23k8	22	8	0.84	1.47	0.68	0.69	0.66	0.15	0.15	0.15	6	0
Pn40k5	39	5	8.83	2.75	6.89	7.57	6.24	1.31	2.48	0.56	160.83	0
Pn44k5	44	5	15.18	2.48	9.29	10.94	7.47	1.06	1.58	0.42	379.37	0
Pn50k7	49	7	19.95	5.61	11.86	13.73	8.83	1.57	2.98	0.59	310	0
Pn50k8	49	8	19.65	5.12	11.26	12.35	9.97	1.42	2.11	0.64	205.71	0
Pn50k10	49	10	17.51	3.61	11.58	12.25	10.79	0.87	1.56	0.52	63.52	0
Pn51k10	50	10	17.58	1.65	12.07	18.56	9.85	0.54	0.77	0.34	122.08	0
Pn55k7	54	7	24.07	7.46	15.09	18.44	10.69	2.53	3.98	0.85	425.51	0
Pn55k8	54	8	27.89	5.44	17.16	18.28	16.09	2.7	3.86	1.28	301.08	0
Pn55k10	54	10	20.7	7.5	13.45	15.8	10.99	1.62	4.1	0.71	98	0
Pn55k15	54	15	17.11	4.61	13.09	13.91	11.31	1.05	1.86	0.65	100	0
Pn60k10	59	10	27.65	4.23	17.16	20	14.56	1.28	2.4	0.65	170.62	0
Pn65k10	64	10	45.42	4.4	21.62	25.69	18.53	1.69	2.66	0.46	858	0
Pn70k10	69	10	57.49	4.65	29.84	34.02	25.11	1.78	2.69	1.01	728.51	0
Pn76k4	75	4	173.45	-108.27	74.20	77.46	65.24	-108.241	-107.83	-108.57	3600	110.28
Pn76k5	75	5	126.66	-	61.75	68.57	51.78	-	-	-	3600	-
Avg			28.52	-1.48	15.16	17.07	12.90	-4.06	-3.1	-4.67	510.38	5.25

Table 5
Results of the RGRASP: P-Instances, $\lambda = 0.1$.

					Gap%			
Instance	*n*	*k*	*TL*	$SCIP^{OF}_{3600}$	$SCIP^{OF}_{TL}$	$RGRASP^{OF}$	$SCIP_{3600}$	$SCIP_{TL}$
Pn16k8	15	5	5	**53.83**	**53.83**	53.61	0.42	0.42
Pn19k2	18	2		**233.77**	-	**233.77**	0.00	-
Pn20k2	19	2		**227.32**	63.68	227.20	0.05	-256.78
Pn21k2	20	2		**177.35**	46.06	176.59	0.43	-283.39
Pn22k2	21	2		**351.41**	250.41	**351.41**	0.00	-40.34
Pn22k8	21	8		**164.77**	**164.77**	164.52	0.15	0.15
Pn23k8	22	8		**236.70**	**236.70**	236.63	0.03	0.03
Pn40k5	39	5	50	604.94	-	607.88	-0.49	-
Pn44k5	44	5		919.59	-	920.34	-0.08	-
Pn50k7	49	7		363.10	-	363.82	-0.20	-
Pn50k8	49	8		372.06	-	373.06	-0.27	-
Pn50k10	49	10		**388.42**	-	387.13	0.33	-
Pn51k10	50	10		1166.56	-	1161.78	0.41	-
Pn55k7	54	7	100	410.87	-	400.52	2.52	-
Pn55k8	54	8		420.53	-	422.03	-0.36	-
Pn55k10	54	10		**247.51**	207.10	242.33	2.09	-17.01
Pn55k15	54	15		260.19	-	258.68	0.58	-
Pn60k10	59	10		**372.72**	-	372.39	0.09	-
Pn65k10	64	10		559.70	-	565.23	-0.99	-
Pn70k10	69	10		630.35	-	636.73	-1.01	-
Pn76k4	75	4	500	-	-	5980.24	-	-
Pn76k5	75	5		-	-	6073.02	-	-
Avg							0.19	-85.27

and Pn22k2), within 5 seconds. On average, the gap is around 0.19%. When the same time limit is considered for both SCIP and the RGRASP, we can observe that in most of the cases SCIP is not even able to find a feasible solution. When feasible solutions are available, their quality is considerably low and outperformed by the RGRASP solutions (the gap on average is -85.27%). The only exceptions are the instances Pn16k8, Pn22k8 and Pn23k8, where SCIP is able to find the optimal solution within the *TL* (without proving the optimality in the last two instances). Tables 6 and 7 report the results for $\lambda = 0.5$ and 0.9, respectively. When $\lambda = 0.5$ and SCIP runs for one hour, 12 out of 22 instances are solved to optimality. Over these instances the RGRASP gap is around 0.27%. The RGRASP outperforms SCIP considerably, when the same time limit is imposed (the gap is -108.69%). For $\lambda = 0.9$ the number of instances solved to optimality by SCIP in one hour is 20. Over these instances the average RGRASP gap is around 0.42%, whereas over all the instances the gap is -4.77%. On the other hand, even though this case is quite easy, SCIP fails to find feasible solutions in 7 instances within the time limit *TL*. When it succeeds, it is outperformed by the RGRASP in all but three instances, namely Pn16k8, Pn55k10 and Pn55k15 (in these cases the optimal solution has been found by SCIP). The average gap is -136.8%.

5.2. Results for the E-instances

The computational results of the SoA and the GRASP heuristics are reported inTables 8–10 for the E-instances. Analyzing the effect of the parameter λ, the decrease in the value of λ, makes

Table 6
Results of the RGRASP: P-Instances, $\lambda = 0.5$.

					Gap%			
Instance	n	k	TL	$SCIP^{OF}_{3600}$	$SCIP^{OF}_{TL}$	$RGRASP^{OF}$	$SCIP_{3600}$	$SCIP_{TL}$
Pn16k8	15	5	5	**338.76**	**338.76**	**338.76**	0.00	0.00
Pn19k2	18	2		**1331.57**	830.20	**1331.57**	0.00	-60.39
Pn20k2	19	2		**1355.55**	368.71	1335.56	1.48	-262.22
Pn21k2	20	2		**1037.12**	297.14	**1037.12**	0.00	-249.03
Pn22k2	21	2		**1928.18**	988.18	1922.85	0.28	-94.58
Pn22k8	21	8		**915.90**	**915.90**	915.43	0.05	0.05
Pn23k8	22	8		**1286.05**	1284.60	1285.21	0.06	-0.05
Pn40k5	39	5	50	**3272.68**	670.56	3256.65	0.49	-385.66
Pn44k5	44	5		4841.90	-	4803.26	0.80	-
Pn50k7	49	7		2013.01	-	2003.41	0.48	-
Pn50k8	49	8		**2051.71**	-	2044.86	0.33	-
Pn50k10	49	10		2103.15	-	2102.78	0.02	-
Pn51k10	50	10		**5997.42**	2976.42	5983.92	0.23	-101.04
Pn55k7	54	7	100	2261.54	-	2244.15	0.77	-
Pn55k8	54	8		2310.28	-	2272.13	1.65	-
Pn55k10	54	10		**1386.10**	968.10	1383.15	0.21	−42.87
Pn55k15	54	15		**1439.63**	1439.63	1436.44	0.22	0.22
Pn60k10	59	10		2053.86	-	2044.00	0.48	-
Pn65k10	64	10		3018.38	-	2993.78	0.81	-
Pn70k10	69	10		3386.29	-	3378.80	0.22	-
Pn76k4	75	4	500	19407.31	-	30419.05	-56.74	-
Pn76k5	75	5		-	-	30869.45	-	-
Avg							-2.29	-108.69

Table 7
Results of the RGRASP: P-Instances, $\lambda = 0.9$.

					Gap%			
Instance	n	k	TL	$SCIP^{OF}_{3600}$	$SCIP^{OF}_{TL}$	$RGRASP^{OF}$	$SCIP_{3600}$	$SCIP_{TL}$
Pn16k8	15	5	5	**625.35**	**625.35**	620.87	0.72	0.72
Pn19k2	18	2		**2433.51**	1033.00	**2433.51**	0.00	-135.58
Pn20k2	19	2		**2443.91**	673.74	**2443.91**	0.00	-262.74
Pn21k2	20	2		**1899.42**	548.22	**1899.42**	0.00	-246.47
Pn22k2	21	2		**3502.83**	352.38	3495.77	0.20	-892.05
Pn22k8	21	8		**1667.98**	1667.98	**1667.98**	0.00	0.00
Pn23k8	22	8		**2335.61**	2330.20	2334.60	0.04	-0.19
Pn40k5	39	5	50	**5935.33**	1228.51	5907.33	0.47	-380.85
Pn44k5	44	5		**8764.33**	-	8729.28	0.40	-
Pn50k7	49	7		**3663.40**	3300.40	3646.85	0.45	-10.50
Pn50k8	49	8		**3731.94**	-	3698.36	0.90	-
Pn50k10	49	10		**3822.93**	2922.93	3804.12	0.49	-30.15
Pn51k10	50	10		**10828.28**	8280.28	10805.37	0.21	-30.50
Pn55k7	54	7	100	**4112.30**	-	4087.83	0.59	-
Pn55k8	54	8		**4198.85**	3198.85	4115.91	1.98	-28.67
Pn55k10	54	10		**2529.23**	**2529.23**	2509.69	0.77	0.77
Pn55k15	54	15		**2621.52**	**2621.52**	2619.72	0.07	0.07
Pn60k10	59	10		**3737.97**	2737.90	3718.80	0.51	-35.83
Pn65k10	64	10		**5471.67**	-	5466.86	0.09	-
Pn70k10	69	10		**6137.35**	-	6098.26	0.64	-
Pn76k4	75	4	500	26337.00	-	54971.51	-108.72	-
Pn76k5	75	5		-	-	55638.61	-	-
Avg							-4.77	-136.80

Table 8
GRASP versus SoA: E-Instances, $\lambda = 0.1$.

			SoA		GRASP						SCIP	
			CPU		*CPU*						*CPU*	*Opt%*
Instance	n	k			Avg	Max	Min	Avg	Max	Min		
En22k4	21	4	0.99	5.96	1.02	1.32	0.78	0.64	2.64	0.29	26.95	0
En23k3	22	3	1.84	7.45	1.25	1.63	1.04	3.14	6.62	0.66	18.56	0
En30k3	29	3	5.61	3.60	3.38	3.90	3.09	-1.07	0.02	-2.40	3600	17.45
En33k4	32	4	6.30	1.19	3.97	4.5	3.20	-0.32	1.18	-1.41	3600	14.04
En51k5	50	5	22.93	2.79	13.61	15.61	10.77	0.85	1.46	0.18	3600	6.61
En76k7	75	7	108.8	-	48.82	53.17	43.85	-	-	-	3600	-
En76k8	75	8	95.38	-1.73	41.45	49.39	32.05	-1.54	-1.08	-1.88	3600	5.42
En76k10	75	10	85.95	0.33	43.30	45.46	37.27	0.34	0.64	0.10	3600	2.56
En76k14	75	14	62.26	0.27	32.93	36.96	29.94	0.40	0.60	0.22	3600	1.64
Avg			43.34	2.48	21.08	23.55	18.00	0.31	1.51	-0.53	2805.06	5.97

Table 9
GRASP versus SoA: E-Instances, $\lambda = 0.5$.

			SoA		GRASP						SCIP	
			CPU		*CPU*						*CPU*	*Opt%*
Instance	*n*	*k*			Avg	Max	Min	Avg	Max	Min		
En22k4	21	4	1.13	3.18	1.13	1.22	1.06	0.861	2.87	0.35	16.44	0
En23k3	22	3	1.79	2.18	1.33	1.68	1.11	1.63	3.98	0.84	15.73	0
En30k3	29	3	5.44	2.24	3.50	4.06	2.85	0.89	2.36	0.35	1885.35	0
En33k4	32	4	7.12	1.22	4.14	4.54	3.59	1.68	2.63	1.08	3600	0.88
En51k5	50	5	29.03	2.97	13.87	17.23	11.53	1.12	1.45	0.61	3600	0.56
En76k7	75	7	112.89	0.07	46.20	55.22	39.05	0.31	0.51	0.09	3600	1.11
En76k8	75	8	96.15	1.12	42.24	48.57	34.69	0.76	1.4	0.49	3600	0.47
En76k10	75	10	86.96	0.58	43.82	46.08	39.38	0.53	0.67	0.43	3600	0.26
En76k14	75	14	64.17	0.25	32.55	34.57	30.49	0.29	0.38	0.16	3600	0.18
Avg			44.96	1.53	20.98	23.69	18.19	0.90	1.81	0.49	2613.06	0.38

Table 10
GRASP versus SoA: E-Instances, $\lambda = 0.9$.

			SoA		GRASP						SCIP	
			CPU		*CPU*						*CPU*	*Opt%*
Instance	*n*	*k*			Avg	Max	Min	Avg	Max	Min		
En22k4	21	4	1.29	2.19	1.02	1.32	0.78	0.925	1.54	0.19	18.05	0
En23k3	22	3	2.12	4.95	1.25	1.63	1.04	2.12	3.09	0.42	21.43	0
En30k3	29	3	5.65	2.48	3.38	3.9	3.09	0.98	2.35	0.17	128.36	0
En33k4	32	4	6.72	1.34	3.97	4.5	3.2	1.31	2.38	0.54	123.18	0
En51k5	50	5	25.22	2.38	13.61	15.61	10.77	1.18	2.32	0.36	520.88	0
En76k7	75	7	107.73	0.77	48.82	53.17	43.85	0.71	0.99	0.50	3600	0.38
En76k8	75	8	90.49	1.17	41.45	49.39	32.05	0.69	0.96	0.49	3600	0.31
En76k10	75	10	92.19	0.72	43.30	45.46	37.27	0.47	0.6	0.33	1539	0
En76k14	75	14	70.77	0.28	32.93	36.96	29.94	0.25	0.35	0.16	1436	0
Avg			44.69	1.81	21.08	23.55	18.00	0.96	1.62	0.35	1220.77	0.08

Table 11
Results of the RGRASP: E-Instances, $\lambda = 0.1$.

					Gap%			
Instance	*n*	*k*	*TL*	$SCIP^{OF}_{3600}$	$SCIP^{OF}_{TL}$	$RGRASP^{OF}$	$SCIP_{3600}$	$SCIP_{TL}$
En22k4	21	4	5	**198.61**	179.05	198.02	0.29	-10.6
En23k3	22	3		**308.06**	278.23	305.99	0.67	-9.98
En30k3	29	3	50	466.47	-	475.70	-1.98	-
En33k4	32	4		1183.52	-	1211.86	-2.39	-
En51k5	50	5	100	1097.11	-	1106.60	-0.87	-
En76k7	75	7	300	-	-	2793.90	-	-
En76k8	75	8		2763.57	-	2821.86	-2.11	-
En76k10	75	10		2863.89	-	2867.29	-0.12	-
En76k14	75	14		2913.47	-	2911.26	0.08	-
Avg							-0.8	-10.29

the problem more difficult, as we have already observed for the P-Instances. This claim is supported by the increase in the optimality gap and the *CPU* time of the SCIP solutions.

On average the gap of the GRASP heuristic is around 1% for the values of $\lambda \geq 0.5$, but slightly lower for $\lambda = 0.1$. In the latter case, the quality of the solutions obtained for different values of α might considerably vary, as evident from the gap variation (Max-Min) which reaches a peak of around 6% for the test problem En23k3. The solution time is around 20 s for one α value (on average over the ten values of α) and is quite insensitive to the λ value.

The computational time required by the SoA approach is around 45 s for all the λ values. The best solution quality is achieved for $\lambda = 0.5$ with a gap of 1.53%. For $\lambda = 0.9$, the performance deteriorates a bit, but the gap is still less than 2%.

Tables 11–13 report the results of the RGRASP. As a general observation, we note that it is able to obtain good quality solutions. Especially for $\lambda = 0.1$ (Table 11), it is able to find in the allotted time limit *TL* solutions that are better than those obtained by SCIP in one hour, with an average gap of around -0.8%. For the other values of λ the gap is on average 0.19% and always less than 0.52% (instance En33k4 with $\lambda = 0.9$). The SCIP solver, within the time limit *TL* is not able to find any feasible solution in most of the cases. If a solution is eventually found, it is clearly of poor quality. This claim is supported by the average gaps, reported in the last columns of the Tables 11–13 which are, respectively -10.29% for $\lambda = 0.1$,-12.1% for $\lambda = 0.5$ and -484.56% for $\lambda = 0.9$.

5.3. Large size instances

Further computational tests have been carried out on larger size instances, with up to 199 customers and 17 vehicles. SCIP was run in this case with a time limit of four hours. In all the instances, SCIP (not initialized) was not able to find a feasible solution within the allotted time limit and often went out of memory. We also tried to initialize SCIP with the RGRASP, but after four hours no improvement in the objective function was observed before going out of memory or reaching the time limit. We report, in Table 14, the results for the RGRASP heuristic, considering as a termination criterion a time limit of one hour, to serve as a benchmark for future comparisons. The gaps have been obtained considering the upper

Table 12
Results of the RGRASP: E-Instances, $\lambda = 0.5$.

					Gap%			
Instance	n	k	TL	$SCIP^{OF}_{3600}$	$SCIP^{OF}_{TL}$	$RGRASP^{OF}$	$SCIP_{3600}$	$SCIP_{TL}$
En22k4	21	4	5	**1140.58**	1133.36	**1140.58**	0	-0.64
En23k3	22	3		**1794.21**	1454.21	**1794.21**	0	-23.38
En30k3	29	3	50	**2709.34**	-	2704.61	0.17	-
En33k4	32	4		6717.93	-	6673.08	0.67	-
En51k5	50	5	100	5826.22	-	5826.22	0	-
En76k7	75	7	300	14319.51	-	14359.79	−0.28	-
En76k8	75	8		14498.19	-	14377.15	0.83	-
En76k10	75	10		14653.77	-	14608.5	0.31	-
En76k14	75	14		14795.6	-	14790.22	0.04	-
Avg							0.19	-12.1

Table 13
Results of the RGRASP: E-Instances, $\lambda = 0.9$.

					Gap%			
Instance	n	k	TL	$SCIP^{OF}_{3600}$	$SCIP^{OF}_{TL}$	$RGRASP^{OF}$	$SCIP_{3600}$	$SCIP_{TL}$
En22k4	21	4	5	**2084.91**	1833.54	**2084.91**	0	−13.71
En23k3	22	3		**3281.24**	2981.24	**3281.24**	0	−10.06
En30k3	29	3	50	**4946.28**	2846.28	4939.323	0.14	−73.54
En33k4	32	4		**12243.81**	10143.81	12179.8	0.52	−20.07
En51k5	50	5	100	**10548.39**	438.39	10545.04	0.03	−2305.4
En76k7	75	7	300	25948.89	–	25857.41	0.35	–
En76k8	75	8		26146.86	–	26059.99	0.33	–
En76k10	75	10		**26439.54**	–	26379.92	0.23	–
En76k14	75	14		**26682.19**	–	26658.78	0.09	-
			Avg.				0.19	-484.56

Table 14
Results for the CMT-Instances.

Instance	n	k	λ	$RGRASP^{OF}$	UB	*Gap%*
CMT3	100	8	0.1	10789.45	10885.48	0.88
CMT3	100	8	0.5	54501.22	54740.82	0.44
CMT3	100	8	0.9	98205.39	98596.16	0.40
CMT4	150	12	0.1	14641.59	14706.19	0.44
CMT4	150	12	0.5	73466.47	73801.42	0.45
CMT4	150	12	0.9	132531.1	132897.1	0.28
CMT5	199	17	0.1	33041.47	33134.81	0.28
CMT5	199	17	0.5	165742.6	166232.7	0.29
CMT5	199	17	0.9	298517.7	299330.5	0.27
CMT11	120	7	0.1	29691.7	29846.77	0.52
CMT11	120	7	0.5	149583.6	149662.7	0.05
CMT11	120	7	0.9	269281.3	269448.5	0.06
CMT12	100	10	0.1	47193.75	47322.82	0.27
CMT12	100	10	0.5	236894.3	236919.6	0.01
CMT12	100	10	0.9	426001.4	426516.7	0.12
Avg						0.32

bound procedure proposed in Section 3.1, hence the actual optimality gap is possibly less than the one reported. This evaluation is quite fair, since, on the P-instances, the upper bounds, reported in Table 15 in Section A.2 of the Appendix, are quite tight, and the best one has an average gap, evaluated with respect to the optimal solution, around 5% (see Table 15). The upper bound values for the E-instances are slightly looser than for the P-instances with an average gap of 8.46% (see Table 16 of the Appendix, Section A.2).

The results in Table 14 show that the RGRASP metaheuristic is relatively stable in terms of solution quality, with an average gap, evaluated considering the upper bound, of 0.32%. Better gaps are obtained in this case for higher λ values.

6. Conclusions

This paper deals with the solution of the multi-vehicle traveling repairman problem with profits under uncertain travel times, which is an important problem in many customer-centric real-world applications. We have developed a RGRASP metaheuristic, tailored to cope with the peculiarities of the problem at hand equipped with a local search procedure based on different effective moves. To enhance the performance, a move evaluation procedure over different neighborhood structures has been also introduced. We have adapted existing instances to generate new instances for the multi-vehicle traveling repairman problem with profits under uncertain travel times, and we have run several numerical experiments on them to assess the quality of the heuristic methodology. We have shown that the solution method performs well by comparing it to solutions obtained with the commercial solver. Moreover, we have tested our heuristic on the deterministic counterparts of the instances, providing also new benchmark solutions for the related k-TRPP.

Future work can be directed along the following four directions. First, we can assume that different edges may have different correlated travel times. This situation can be relevant, especially in disaster management applications. Second, advanced heuristics that use more destroy operators in the spirit of adaptive large neighborhood search heuristics may be devised. Third, exact algorithms may be proposed to solve small instances, exploiting the specific structure of the problem. Finally, we may consider variants of the problem to capture real-life characteristics, such as the presence of a heterogeneous and capacitated vehicle fleet.

Appendix A

A1. Mathematical model

For the sake of completeness, we report here the mathematical formulation of the problem Bruni et al. (2018b). Following the multi-level network proposed in Nucamendi-Guillén et al. (2016), we define a set of levels $L = \{1, \cdots, r, \cdots, N\}$, where $N = n - k + 1$ is un upper bound on the number of nodes visited in each route. In any level, a copy of nodes is present, amended also with depot in levels from 2 to N. In this network, each tour is represented by a route that ends in a first level node and starts in a copy of the depot in some level. The level number represent the position of the node in the route, in such a way that nodes at level 1 are the last, nodes at level two are the second to last, at level three are the third to last and so on. Two paths cannot visit the same node, nor in the same level neither in different levels. Using the multilevel network, the following decision variables are defined. Corresponding to each pair of demand node $i \in \bar{V}$ ($\bar{V} = V \setminus \{0\}$) and visiting level $r \in L$, a binary variable x_i^r is defined taking the value 1 iff node i is visited at level r, and 0 otherwise. In a similar way, the binary variable y_{ij}^r is assigned to each edge $(ij) \in E$ such that $i \in V$, $j \in \bar{V}$ and takes the value 1 if edge $(ij) \in E$ is used to link node i in level $r+1$ to node j in level r, otherwise its value is set to 0.

Defining $\bar{L} = L \setminus \{N\}$, the mathematical model can be formulated as follows.

$$\max: \; z = \lambda \left(\sum_{j \in \bar{V}} \sum_{r \in L} (\pi_j - r\,\mu_{0j})\, y_{0j}^r + \sum_{i \in \bar{V}} \sum_{\substack{j \in \bar{V} \\ j \neq i}} \sum_{r \in \bar{L}} (\pi_j - r\mu_{ij}\, y_{ij}^r) \right)$$

$$-(1-\lambda) \sqrt{\sum_{j \in \bar{V}} \sum_{r \in L} r^2 \sigma_{0j}^2 y_{0j}^r + \sum_{i \in \bar{V}} \sum_{\substack{j \in \bar{V} \\ j \neq i}} \sum_{r \in \bar{L}} r^2 \sigma_{ij}^2 y_{ij}^r} \qquad (3)$$

$$\sum_{r \in L} x_i^r \leq 1 \quad i \in \bar{V} \qquad (4)$$

$$\sum_{i \in \bar{V}} x_i^1 = k \qquad (5)$$

$$\sum_{r \in L} \sum_{j \in \bar{V}} y_{0j}^r = k \qquad (6)$$

$$\sum_{\substack{j \in \bar{V} \\ j \neq i}} y_{ij}^r = x_i^{r+1} \quad i \in \bar{V}, \; r \in \bar{L} \qquad (7)$$

$$y_{0j}^r + \sum_{\substack{i \in \bar{V} \\ i \neq j}} y_{ij}^r = x_j^r \quad j \in \bar{V}, \; r \in \bar{L} \qquad (8)$$

$$y_{0j}^N = x_j^N \quad j \in \bar{V} \qquad (9)$$

$$x_i^r \in \{0, 1\} \quad i \in \bar{V}, \; r \in L \qquad (10)$$

$$y_{ij}^r \in \{0, 1\} \quad i \in V, \; j \in \bar{V}, i \neq j, \; r \in \bar{L} \qquad (11)$$

$$y_{0j}^r \in \{0, 1\}, \; j \in \bar{V}, \; r \in L. \qquad (12)$$

The objective function (3) maximizes the total profit collected by visiting a subset of nodes which is expressed as the difference between the revenue and the arrival time of the visited nodes. The parameter $\lambda \in (0, 1]$ plays the role of the trade-off weight in mean-risk models Lecluyse et al. (2009).

Constraints (4) require that each demand node is visited at most in one level. Constraints (5) ensure that each vehicle is assigned to exactly one demand node at the end of its tour (level 1). Constraints (6) impose the dispatch of exactly k vehicles from the depot. The constraints in (7)–(9) are the connectivity constraints and show the relation between the binary variables x_i^r and y_{ij}^r. Constraints (7) require that any node i visited at the upper level $r+1$ should be connected to exactly one upcoming visited node (let say j) by traversing edge (ij) at the lower level r. The constraints (8) impose that any node j visited at level r should be linked to exactly one recently visited node (let say i) by traversing edge (ij) or linked directly to the depot by traversing edge $(0j)$ at the same level. Constraints (9) require that each node visited at the highest level N should be the first visited node over the route which is connected to the depot by traversing edge $(0j)$ at the same level. The constraints in (10)–(12) express the nature of variables.

A2. Upper bound evaluation

In order to evaluate the quality of the upper bounds proposed in Section 3.1, we have reported the gaps Δ^i $i = 1, \ldots, 3$ between the upper bounds UB^i, $i = 1, \ldots, 3$, and the optimal solution of SCIP for which a time limit of one hour has been imposed. If,

Table 15
Upper bound performance for the P-instances.

Instance	n	k	λ	Δ^1	Δ^2	Δ^3	Best Δ
Pn16k8	15	5	0.1	12.78	40.02	7.6	7.6
			0.5	4.21	23.35	3.13	3.13
			0.9	3.39	21.59	3.02	3.02
Pn19k2	18	2	0.1	19.31	15.96	8.87	8.87
			0.5	10.73	8.23	4.71	4.71
			0.9	9.72	7.31	4.13	4.13
Pn20k2	19	2	0.1	24.88	16.93	8.54	8.54
			0.5	11.01	5.58	2.21	2.21
			0.9	11.56	6.3	3.28	3.28
Pn21k2	20	2	0.1	26.84	12.2	5.92	5.92
			0.5	17.81	7.28	3.82	3.82
			0.9	16.81	6.77	3.61	3.61
Pn22k2	21	2	0.1	14.34	9.03	5.22	5.22
			0.5	8.98	4.7	2.37	2.37
			0.9	8.51	4.41	2.26	2.26
Pn22k8	21	8	0.1	2.87	20.16	5.49	2.87
			0.5	2.04	13.33	1.77	1.77
			0.9	1.9	12.59	1.34	1.34
Pn23k8	22	8	0.1	3.17	15.47	2.11	2.11
			0.5	1.34	10.16	2.57	1.34
			0.9	1.2	9.61	2.61	1.2
Pn40k5	39	5	0.1	–	–	–	–
			0.5	7.88	5.15	3.72	3.72
			0.9	7.47	4.8	3.53	3.53
Pn44k5	44	5	0.1	–	–	–	–
			0.5	–	–	–	–
			0.9	7.05	4.21	3.21	3.21
Pn50k7	49	7	0.1	–	–	–	–
			0.5	–	–	–	–
			0.9	8.6	9.54	6.59	6.59
Pn50k8	49	8	0.1	–	–	–	–
			0.5	7.06	9.51	5.89	5.89
			0.9	6.6	8.84	5.47	5.47
Pn50k10	49	10	0.1	6.93	14.65	7.62	6.93
			0.5	–	–	–	–
			0.9	4.07	8.64	4.52	4.07
Pn51k10	50	10	0.1	–	–	–	–
			0.5	2	2.83	1.48	1.48
			0.9	1.89	2.72	1.36	1.36
Pn55k7	54	7	0.1	–	–	–	–
			0.5	–	–	–	–
			0.9	9.2	8.8	5.65	5.65
Pn55k8	54	8	0.1	–	–	–	–
			0.5	–	–	–	–
			0.9	6.95	9.05	5.62	5.62
Pn55k10	54	10	0.1	12.51	23.2	13.23	12.51
			0.5	8.07	15.19	8.42	8.07
			0.9	7.46	14.41	7.75	7.46
Pn55k15	54	15	0.1	–	–	–	–
			0.5	4.05	18.32	9.25	4.05
			0.9	3.68	17.42	8.76	3.68
Pn60k10	59	10	0.1	12.12	19.53	11.1	11.1
			0.5	–	–	–	–
			0.9	7.18	12.75	7.57	7.18
Pn65k10	64	10	0.1	–	–	–	–
			0.5	–	–	–	–
			0.9	5.29	8.91	4.78	4.78
Pn70k10	69	10	0.1	–	–	–	–
			0.5	–	–	–	–
			0.9	6.03	8.59	6.08	6.03
Pn76k4	75	4	0.1	–	–	–	–
			0.5	–	–	–	–
			0.9	–	–	–	–
Pn76k5	75	5	0.1	–	–	–	–
			0.5	–	–	–	–
			0.9	–	–	–	–
Average				8.46	11.86	5.15	4.71

Table 16
Upper bound performance for the E-instances

Instance	n	k	λ	Δ^1	Δ^2	Δ^3	Best Δ
En22k4	21	4	0.1	16.76	19.94	15.14	15.14
			0.5	10.88	12.52	10.5	10.5
			0.9	10.2	11.69	9.93	9.93
En23k3	22	3	0.1	21.23	26.57	17.17	17.17
			0.5	14.48	15.96	12.09	12.09
			0.9	13.87	14.93	11.58	11.58
En30k3	29	3	0.1	–	–	–	–
			0.5	9.48	17.24	14.57	9.48
			0.9	8.81	16.13	13.62	8.81
En33k4	32	4	0.1	–	–	–	–
			0.5	–	–	–	–
			0.9	2.09	15.88	6.12	2.09
En51k5	50	5	0.1	–	–	–	–
			0.5	–	–	–	–
			0.9	7.21	3.13	2.54	2.54
En76k7	75	7	0.1	–	–	–	–
			0.5	–	–	–	–
			0.9	–	–	–	–
En76k8	75	8	0.1	–	–	–	–
			0.5	–	–	–	–
			0.9	–	–	–	–
En76k10	75	10	0.1	–	–	–	–
			0.5	–	–	–	–
			0.9	1.6	2.06	1.49	1.49
En76k14	75	14	0.1	–	–	–	–
			0.5	–	–	–	–
			0.9	0.67	2.26	1.34	0.67
Avg				9.77	13.19	9.67	8.46

within the allotted time limit, the optimal solution is not found a dash is reported. In Table 15, we report the results obtained on the P-instances. The first upper bound outperforms the second one in 27 out of 42 instances, and the third one in 15 out of 42 instances. On average the third upper bound provides slightly better gaps (5.15% %). The second upper bound provides the same gap of the third one in 24 cases. Over all the instances, the average gap of the best upper bound is around 5%. As far as the computational time is concerned, it is neglectable in all the three cases.

In Table 16 we assess the quality of the upper bounds proposed in Section 3.1 for the E-instances. The first upper bound outperforms the third one in 12 out of 27 instances. However, the third upper bound provides, on average, slightly better gaps (around 1% of improvement). The second upper bounds seems to be dominated by the other two, as for the P-Instances. The quality of the upper bound improves as far as the number of nodes increases. For example, considering as a threshold a number of nodes equal to 29, we observe that when the number of nodes is below the threshold, the average gap of the best upper bound is around 12.7%, and that it decreases to 4.18% when the number of nodes exceeds the threshold. Over all the instances the average gap of the best upper bound is around 8.46%.

References

Bianco, L., Mingozzi, A., Ricciardelli, S., 1993. The traveling salesman problem with cumulative costs. Networks 23 (2), 81–91.
Christofides, N., Eilon, S., 1969. An algorithm for the vehicle dispatching problems. Oper. Res. Q. 20 (3), 309–318.
Guerriero, F., Bruni, M.E., Greco, F., 2013. A hybrid greedy randomized adaptive search heuristic to solve the dial-a-ride problem. Asia-Pacific J. Oper. Res. 30 (01), 125–146.
Archer, A., Blasiak, A., 2010. Improved Approximation Algorithms for the Minimum Latency Problem via Prize-collecting Strolls. In: InProceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms 2010 Jan 17 (pp. 429–447). Society for Industrial and Applied Mathematics.
Augerat, P., Belenguer, J.M., Benavent, E., Corberán, A., Naddef, D., Rinaldi, G., 1995. Computational Results with a Branch-and-cut Code for the Capacitated Vehicle Routing Problem. 1998 IASI Research Report n. 495.
Beraldi, P., Bruni, M.E., Laganà, D., Musmanno, R., 2015. The mixed capacitated general routing problem under uncertainty. Eur. J. Oper. Res. 240 (2), 382–392.
Beraldi, P., Bruni, M.E., Laganà, D., Musmanno, R., 2019. The risk-averse traveling repairman problem with profits. Soft Comput. 23 (9), 2979–2993.
Bigras, L.P., Gamache, M., Savard, G., 2008. The time-dependent traveling salesman problem and single machine scheduling problems with sequence dependent setup times. Discrete Optim. 5 (4), 685–699.
Blum, A., Chalasani, P., Coppersmith, D., Pulleyblank, B., Raghavan, P., Sudan, M., 1994. The Minimum Latency Problem. In: InProceedings of the twenty-sixth annual ACM symposium on Theory of computing, p. ACM. May 23, (pp. 163–171)
Boyaci, T., Ray, S., 2003. Product differentiation and capacity cost interaction in time and price sensitive markets. Manuf. Serv. Oper. Manage. 5 (1), 18–36.
Bruni, M.E., Beraldi, P., Khodaparasti, S., 2018. A fast heuristic for routing in post-disaster humanitarian relief logistics. Transp. Res. Procedia 30, 304–313.
Bruni, M.E., Beraldi, P., Khodaparasti, S., 2018. A heuristic approach for the k-traveling repairman problem with profits under uncertainty. Electron. Notes Discrete Math. 69, 221–228.
Bruni, M.E., Brusco, L., Ielpa, G., Beraldi, P., 2019. The Risk-averse Profitable Tour Problem. In: ICORES 2019 - Proceedings of the 8th International Conference on Operations Research and Enterprise Systems pp. 459–466.
Bruni, M.E., Forte, M., Scarlato, A., Beraldi, P., 2019. The Traveling Repairman Problem App for Mobile Phones: A Case on Perishable Product Delivery. New Trends in Emerging Complex Real Life Problems. AIRO Springer Series. In press
Bruni, M.E., Guerriero, F., Beraldi, P., 2014. Designing robust routes for demand-responsive transport systems. Transp. Res. Part E: Logist.Transp. Rev. 70, 1–16.
Bruni, M.E., Nucamendi-Guillen, S., Khodaparasti, S., Beraldi, P., 2019. The Cumulative Capacitated Vehicle Routing Problem with Profits under Uncertainty. New Trends in Emerging Complex Real Life Problems. AIRO Springer Series. In press
Chaudhuri, K., Godfrey, B., Rao, S., Talwar, K., 2003. Paths, Trees, and Minimum Latency Tours. in Foundations of Computer Science. In: Proceedings. 44th Annual IEEE Symposium on 2003 Oct 11 (pp. 36–45). IEEE.
Christofides, N., Mingozzi, A., Toth, P., 1979. The Vehicle Routing Problem. In: Christofides, N., Mingozzi, A., Toth, P., Sandi, C. (Eds.), Combinatorial Optimization. John Wiley and Sons, London.
Dewilde, T., Cattrysse, D., Coene, S., Spieksma, F.C., Vansteenwegen, P., 2013. Heuristics for the traveling repairman problem with profits. Comput. Oper. Res. 40 (7), 1700–1707.
Ezzine, I.O., Semet, F., Chabchoub, H., 2010. New Formulations for the Traveling Repairman Problem. In: Proceedings of the 8th International conference of modeling and simulation.
Fakcharoenphol, J., Harrelson, C., Rao, S., 2007. The k-traveling repairmen problem. ACM Trans. Alg. (TALG) 3 (4), 40.
Feo, T., Bard, J., 1989. Flight scheduling and maintenance base planning. Manage. Sci. 35 (12), 1415–1432.
Fischetti, M., Laporte, G., Martello, S., 1993. The delivery man problem and cumulative matroids. Oper. Res. 41 (6), 1055–1064.
Gendreau, M., Ghiani, G., Guerriero, E., 2015. Time-dependent routing problems: a review. Comput. Oper. Res. 64, 189–197.
Kara, I., Kara, B.Y., Yetis, M.K., 2008. Cumulative vehicle routing problems. Veh. Routing Probl.. InTech
Krokhmal, P., Zabarankin, M., Uryasev, S., 2011. Modeling and optimization of risk. Surv. Oper. Res. Manage.Sci. 16 (2), 49–56.
Lecluyse, C., Van Woensel, T., Peremans, H., 2009. Vehicle routing with stochastic time-dependent travel times. 4OR: Q. J. Oper. Res. 7 (4), 363–377.
Lucena, A., 1990. Time-dependent traveling salesman problem-the deliveryman case. Networks 20 (6), 753–763.
Markowitz, H.M., 1952. Portfolio selection. J. Finance 7, 77–91.
Méndez-Díaz, I., Zabala, P., Lucena, A., 2008. A new formulation for the traveling deliveryman problem. Discrete Appl. Math. 156 (17), 3223–3237.
Mladenović, N., Urošević, D., Hanafi, S., 2013. Variable neighborhood search for the travelling deliveryman problem. 4OR 11 (1), 57–73.
Ngueveu, S.U., Prins, C., Calvo, R.W., 2010. An effective memetic algorithm for the cumulative capacitated vehicle routing problem. Comput. Oper. Res. 37 (11), 1877–1885.
Nucamendi-Guillén, S., Martínez-Salazar, I., Angel-Bello, F., Moreno-Vega, J.M., 2016. A mixed integer formulation and an efficient metaheuristic procedure for the k-travelling repairmen problem. J. Oper. Res. Soc. 67 (8), 1121–1134.
Osman, I.H., 1993. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. Ann. Oper. Res. 41, 421–451.
Prais, M., Ribeiro, C., 2000. Reactive GRASP: an application to a matrix decomposition problem in TDMA traffic assignment. INFORMS J. Comput. 12 (3), 164–176. Ray and Jewkes, 2004
Ray, S., Jewkes, E.M., 2004. Customer lead time management when both demand and price are lead time sensitive. Eur. J. Oper. Res. 153, 769–781.
Ribeiro, G.M., Laporte, G., 2012. An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. Comput. Oper. Res. 39 (3), 728–735.
Rockafellar, R., 2007. Coherent approaches to risk in optimization under uncertainty. INFORMS, Tutor. Oper. Res. 38–61.
Salehipour, A., Sörensen, K., Goos, P., Bräysy, O., 2011. Efficient GRASP+ VND and GRASP + VNS metaheuristics for the traveling repairman problem. 4OR: Q. J. Oper. Res. 9 (2), 189–209.
Silva, M.M., Subramanian, A., Vidal, T., Ochi, L.S., 2012. A simple and effective metaheuristic for the minimum latency problem. Eur. J. Oper. Res. 221 (3), 513–520.
Sze, J.F., Salhi, S., Wassan, N., 2017. The cumulative capacitated vehicle routing problem with min-sum and min-max objectives: an effective hybridisation of adap-

tive variable neighbourhood search and large neighbourhood search. Transp. Res. Part B: Methodol. 101, 162–184.
Van Ee, M., Sitters, R., 2017. The A Priori Traveling Repairman Problem. Algorithmica.
Van Eijl, C.A., 1995. A Polyhedral Approach to the Delivery Man Problem. Department of Math. and Computing Science University of Technology.
Zhang, J., Tang, J., Fung, R.Y., 2011. A scatter search for multi-depot vehicle routing problem with weight-related cost. Asia-Pacific J. Oper. Res. 28 (03), 323–348.
Zhang, J., Tang, J.F., Pan, Z.D., Yuan, K., 2010. Scatter search algorithm for solving weighted vehicle routing problem. J. Syst. Eng. 25, 91–97.