





WILEY

INTERNATIONAL
TRANSACTIONS
IN OPERATIONAL
RESEARCHIntl. Trans. in Op. Res. 0 (2023) 1–23
DOI: 10.1111/itor.13407

The multiobjective traveling salesman–repairman problem with profits: design and implementation of a variable neighborhood descent algorithm for a real scenario

R. Morante-González^a , A. D. López-Sánchez^{b,*} , J. Sánchez-Oro^a 
and A. G. Hernández-Díaz^b ^aUniversidad Rey Juan Carlos, Tulipán s/n., Móstoles 28933, Madrid, Spain^bUniversidad Pablo de Olavide, Ctra. de Utrera km 1., Sevilla 41013, Spain

E-mail: r.morante@alumnos.urjc.es [Morante-González]; adlopsan@upo.es [López-Sánchez];

jesus.sanchezoro@urjc.es [Sánchez-Oro]; agarher@upo.es [Hernández-Díaz]

Received 16 March 2023; received in revised form 6 November 2023; accepted 6 November 2023

Abstract

This paper introduces a problem that can be seen as a combination of the traveling salesman problem with profits and the traveling repairman problem with profits, coined as the multi-objective traveling salesman–repairman problem with profits (Mo-TSRPP). The objective of the Mo-TSRPP is to simultaneously optimize three objectives: the total cost, total latency, and total profit. Indirectly, the number of nodes visited is also considered although not as an objective itself since it is determined by the size of every efficient solution in the Pareto front. The Mo-TSRPP emerges as a real-world problem in which a freelancer, which repairs appliances, wants to plan the daily route. Moreover, the daily plan does not require to visit all customers. To solve the problem, first, a greedy randomized adaptive procedure is designed to generate a set of high-quality nondominated solutions and then, a variable neighborhood descent algorithm is applied for further improving the initial set. This procedure allows us to attain a good approximation of the Pareto front. To prove the performance of the proposal a comparison is done against three well-known evolutionary algorithms: NSGA-II, SPEA-2, and MOEA/D. Finally, a realistic problem is shown and solved to illustrate the potential of the algorithm.

Keywords: traveling salesman problem; traveling repairman problem; profit; multiobjective optimization problem; greedy randomized adaptive search procedure; variable neighborhood descent

*Corresponding author.

© 2023 The Authors.

International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies.

This is an open access article under the terms of the Creative Commons Attribution-NonCommercial License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited and is not used for commercial purposes.

1. Introduction

The traveling salesman problem (TSP) is one of the most widely studied combinatorial optimization problem in the literature. Actually, more than 12, 000 works (according to ScienceDirect) have been published from both theoretical and practical points of view, studying its complexity, its properties, or being solved using exact or approximate algorithms. The objective of the TSP is to find a Hamiltonian cycle, which is a closed path that visits every node in the graph exactly once, except the first and final nodes that coincide, with minimum total cost. In other words, it is needed to determine an order in which each node should be visited such that each node is visited only once, and the total cost incurred (distance traveled or travel time) is minimal. To study the TSP and its variations, see Gutin and Punnen (2006) and, from a computational perspective, we refer the reader to Applegate et al. (2007).

A more general problem than the TSP is the TSP with profits (TSPP), in which each node has associated a profit and a feasible solution does not require to visit all nodes. Namely, one must find a Hamiltonian cycle over a subset of nodes such that the collected profit is maximized while the travel cost is minimized. As Angelelli et al. (2014) holds, the goal is to find an appropriate trade-off between the total collected profit and the travel cost of the tour since the profit collection maximization implies that the salesman should visit a large number of nodes, while the cost minimization implies that the salesman should visit a short number of nodes. Nevertheless, most of the papers solve the TSPP by considering just one objective instead of those two. When this happens, several variants arise. The profitable tour problem, also known as the simple cycle problem and introduced by Dell' Amico et al. (1995), maximizes the difference between the total collected profits and the total travel cost of the tour; the orienteering problem, also known as the selective TSP or the maximum collection problem, introduced by Tsiligirides (1984), seeks to maximize the total collected profit while maintaining a maximum traveling cost (a given fix value; see Laporte and Martello, 1990); in the prize-collecting TSP (PCTSP), the salesman collects a profit of each visited node and pays a penalty for each non-visited node being the goal the minimization of the sum of the travel costs and penalties, but collecting a minimum preset amount of profits; and the quota TSP, where the goal is to minimize the total travel cost but the salesman needs to fulfill a quota. Note that the quota TSP is a particular case of the PCTSP where all penalties are zero (see Silva et al., 2020, for further details).

As far as we know, the first attempt to address the TSPP from a multiobjective perspective was done by Jozefowicz et al. (2008). They apply a hybrid metaheuristic to solve the problem using an ejection chain process as local search combined with a multiobjective evolutionary algorithm that generates a population of solutions. Specifically, the authors initialize the population by solving the insert and shake procedure proposed by Gendreau et al. (1998) and by extending the tour until no more nodes can be added without violating a length limit. Then, GENIUS by Gendreau et al. (1992) is applied in an attempt to obtain a shorter tour on the nodes in the solution. If GENIUS fails to produce a better tour, the procedure terminates. Then, all the nondominated solutions are saved in the population and the procedure is repeated for a number of iterations. Two parents are selected from the population and genetic operators are applied to obtain an offspring. Then, the ejection chain process is applied with the offprints as the starting solution. The worst solution in the population is replaced by the offprint or by a better solution found when applying the ejection chain process. One year later, Bérubé et al. (2009) solved the TSPP using an ϵ -constraint method

combined with two improvement heuristics to accelerate the convergence of the algorithm. The ϵ -constraint method turns a multiobjective optimization problem in single-objective subproblems by transforming all objectives except one into constraints. The first improvement heuristic applies cutting-plane algorithms in order to reduce the size of the relaxed solution space by adding cuts. The second improvement heuristic uses the similarities between consecutive solutions obtained with the ϵ -constraint method since a better initial solution increases the upper bound on the optimal solution value and prunes branches in advance allowing to reduce the number of explored nodes.

Another well-studied problem related to the TSP is the traveling repairman problem (TRP), also called the minimum latency problem. The latency of a node is defined as the accumulated cost (length or time) of the path from the depot to each node. Then, this problem seeks to find an open Hamiltonian circuit starting from the depot, which minimizes the sum of path costs to all the nodes. Note that this problem minimizes the total waiting time of all customers instead of the total traveling time of the salesman such as is minimized in the TSP. See, for instance, Lucena (1990), where an early exact algorithm can be found to solve the TRP; Silva et al. (2012) who proposed a simple metaheuristic, based on greedy randomized constructions coupled with random neighborhood ordering for solution improvement to solve the TRP; or Pei et al. (2020) who proposed a general variable neighborhood search (VNS) that uses as local search step five neighborhoods included in a VND algorithm to deal with the TRP.

A problem related to the TRP is the TRP with profits (TRPP) whose objective is to find an open Hamiltonian circuit such that the collected profit is maximized while the latency is minimized. Most of the papers deal with the TRPP as a single-objective optimization problem instead of as biobjective one. Specifically, they minimize the revenue that is defined as the difference between the profit of a node and the number of times the edge preceding that node is counted in the total latency. See, for example, Dewilde et al. (2013) whose authors proposed a mathematical model even though they solve the problem using a metaheuristic based on tabu search; Avci and Avci (2017) who implemented a greedy randomized adaptive search procedure (GRASP) to build initial solutions and an iterated local search with an adaptive perturbation mechanism to improve them. The only attempt to solve this problem from a biobjective point of view without aggregating the functions has been done by Bruni et al. (2020) whose authors proposed an iterated local search to that end. A different biobjective variant was addressed by Arellano-Arriaga et al. (2019) whose objectives are to minimize the total travel cost and the latency and the authors assume that the profit can be viewed as opposite to the total travel cost. They implement an evolutionary algorithm based on an intelligent local search based on the ideas followed by Molina et al. (2018).

The main contribution of this paper is to describe and solve an innovative problem coined as the multiobjective traveling salesman–repairman problem with profits (Mo-TSRPP). This problem can be seen as a generalization of the TSPP and the TSRP. The Mo-TSRPP determines the order in which a subset of the set of nodes should be visited in order to minimize the total travel cost (distance traveled or travel time needed), minimize the total latency, and maximize the total profit. Given that this problem is motivated by a real-world application, it is interesting to consider routes with all combinations of the number of nodes visited, therefore, we have opted to not include solution size as an objective but to consider solutions with every possible number of visited nodes in the set of nondominated solutions.

To solve the problem a VND algorithm is implemented since it has been proved to be very effective when solving similar problems; see Mjirda et al. (2017) and Wang et al. (2017). From a

multiobjective point of view, VND has been successfully applied to solve other difficult combinatorial optimization problems; see Issaoui et al. (2015), López-Sánchez et al. (2018), and Hernández-Pérez et al. (2009), and given the versatility of this methodology we have found it appropriated to solve the Mo-TSRPP. To generate an initial approximation of the Pareto front, we have opted by implementing a greedy randomized adaptive (GRA) procedure that constructs solutions from scratch using a greedy function and including randomization during the construction (more details are given in Section 3). To be more accurate, GRA procedure is the construction phase of the GRASP algorithm but without applying the improvement phase; see Feo et al. (1994) and Feo and Resende (1995). In fact, to strengthen the search and to enrich the Pareto front, every time a node is included in the partial solution, it is sent to the nondominated set of solutions trying to explore all kinds of solution. Next, all solutions will be improved using three neighborhoods, included in a VND framework, keeping the solution size, to find a good approximation of the Pareto front.

Although the Mo-TSRPP arises to find the solution of a real situation in which a freelancer, which repairs appliances, would like to plan the service planning to the customers, there are other real-world applications that can be modeled through the Mo-TSRPP. Next, some examples are briefly commented. The first example, unknown to many people, is the way in which some businesses work in emerging countries. The shortage of many commodities makes that a traveling salesman tries to sell them maximizing the total profit, minimizing the total travel cost, maximizing the number of citizens who receive the commodity to be fair with the citizens, and minimizing the latency since some products are perishable. Additionally, the latency affects to the shift duration of the workers, which is a relevant aspect in a real-life scenario such as the one evaluated in this research. Another interesting real-world application that can be modeled using the Mo-TSRPP is related to the design of routes to nurses at rural health centers. In some health centers located in small areas has assigned a nurse who every day must visit a set of patients in an area. The nurse must visit the maximum number of patients in the minimum time possible. Furthermore, the patients are waiting to receive the service and they prefer to wait the minimum possible time. Of course, the nurse will obtain a hypothetical profit measure as the patient severity. In the ideal situation, all patients should be visited, but this usually does not happen and the nonvisited patients will be visited the next day. To include a third real situation, suppose there are several populations (representing as nodes in the problem) affected by a water cut and there is available just a tanker truck in an area able to deliver drinking water. Each population requires a quantity of water proportional to the number of inhabitants (representing the profit in the model). The objective of the delivery company is to minimize the total distance traveled and maximize the profit, however the objective of the municipalities is to obtain the supplies as soon as possible, which is optimized by minimizing the total latency, and of course, it would be important to serve as many municipalities as possible to avoid the dissatisfaction of the inhabitants.

Next, the main contributions of this work are highlighted.

- A GRA procedure based on the contribution of each node considering all the objectives is proposed to generate high-quality initial solutions to approximate the initial Pareto front.
- Three different neighborhoods are implemented with the particularity that they do not modify the size of every single solution to be improved. This allows us to simplify the difficulty of the problem, but without reducing its performance.

- The initial approximation of the Pareto front obtained with the GRA procedure is included in a VND framework, resulting in an algorithm that is able to improve the initial approximation of the Pareto front.
- The algorithm considers an improvement when a new solution strictly dominates the efficient solution of the same size belonging to the Pareto front, in this way, on the one hand, it prevents an unnecessary increment of the number of efficient solutions in the Pareto front, and on the other hand, it sidesteps the issue of comparing the incumbent solution with all the solutions belonging to the Pareto front.
- Although in this paper we include and solve the real-world instance, we also address a more complex dataset in which more challenging instances are considered in order to prove the performance of the proposed algorithm.

The rest of the paper is organized as follows. Section 2 describes the Mo-TSRPP and some concepts of multiobjective optimization. Section 3 presents the algorithm by explaining all the features that have been considered to simplify the algorithm without reducing the overall performance. Section 4 shows the fitting parameter to calibrate the algorithm, a comparison against three of the most widely implemented multiobjective evolutionary algorithms to prove the performance and furthermore, the resolution of the real-world application is included. Finally, Section 5 summarizes the paper and discusses future work.

2. Problem definition

The Mo-TSRPP can be formally stated as follows. Let $G = (V, A)$ be a complete graph, where $V = \{1, \dots, n\}$ is the node set and $A = \{(i, j) : i, j \in V, i \neq j\}$ is the arc set. Node 1 represents the depot where the salesman starts and ends the route and $N = \{2, \dots, n\}$ is the set of customers requiring a visit/service. Each arc $(i, j) \in A$ has an associated finite cost $c_{ij} \geq 0$ (this cost could be measured as a length or even as a time). Furthermore, each customer $i \in N$ pays a price $p_i > 0$ when receiving the service, or, in other words a profit is collected every time that a node is visited/served (note that the depot has associated a null profit, $p_1 = 0$). The Mo-TSRPP seeks to plan a route by visiting a subset of customers, $S \subset N$, in order to optimize the following objectives: the total travel cost, the total latency, and the total profit, even though indirectly the number of customers served is also considered.

The Mo-TSRPP is a multiobjective optimization problem since it involves to optimize more than one objective function simultaneously and furthermore, they are in conflict which means that it is not possible to improve one objective without deteriorating, at least, one of the remaining objectives. Next, all the pairwise of objective functions are combined in order to see a possible conflict among them; see Table 1. Note that at first glance, the total travel cost of the salesman (or total travel time or total travel distance) is not in conflict with the total latency that the customers wait to be served, but it could be a conflict depending on the considered instance, therefore in Table 1 is written “Sometimes.” However, those objectives are in conflict with the number of customers served and with the profit. Of course, the number of customers that received the service and the total profit obtained by the salesman are not in conflict.

Table 1
Conflicting objectives

Conflict?	Travel cost	Latency	Customers served	Profit
Travel cost	-	Sometimes	Yes, always	Yes, always
Latency	-	-	Yes, always	Yes, always
Customers served	-	-	-	No, never
Profit	-	-	-	-

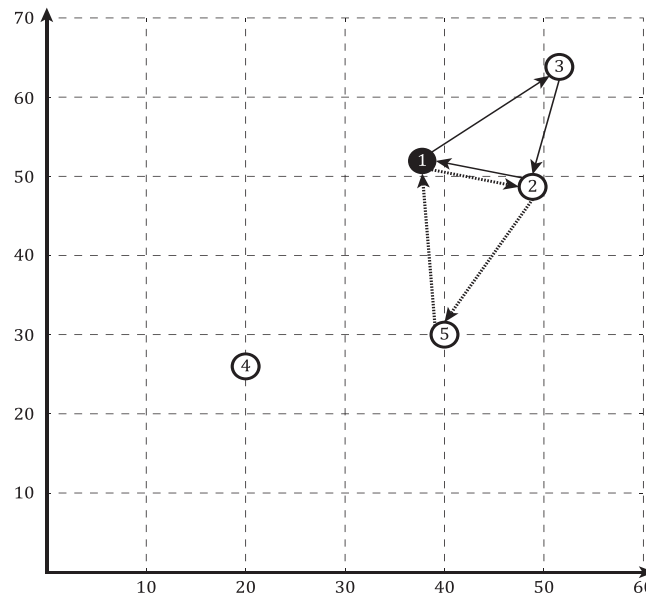


Fig. 1. Two feasible solutions with two nodes served.

In order to graphically illustrate the conflict among the objectives considered, a simple example is included in Figures 1 and 2 where a feasible solution with two and three customers served are represented, respectively. In those examples, the depot is located at (37,52) and four customers require service $N = \{(49, 49), (52, 64), (20, 26), (40, 30)\}$ by paying $p = (23, 35, 56, 40)$.

Both figures represent two solutions that will be compared, one is represented in solid line and the other one in dashed line.

Solutions in Figure 1 serve two customers. In the first solution, the route traveled by the traveler first visits customer 2 and then customer 3, and the values of the objective functions are 46.88 (total travel cost), 53.72 (total latency), 58 (total profit), and 2 (nodes served), meanwhile the second solution performed by the traveler first visits customer 2 and then customer 5 with 55.60 (total travel cost), 45.76 (total latency), 63 (total profit), and 2 (nodes served) as values of the objective functions. Comparing both solutions, it can be observed that the first solution obtains a better result for the total travel cost but the second solution is able to attain better results for the total latency and the profit when serving the same number of customers.

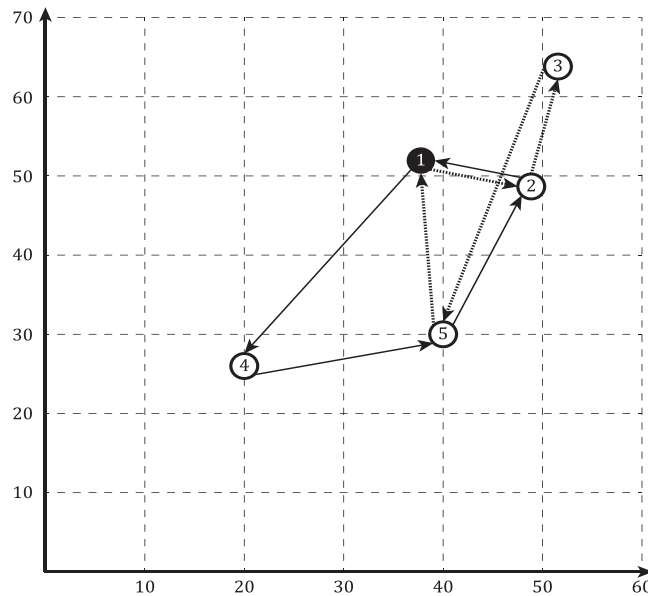


Fig. 2. Two feasible solutions with three nodes served.

Solutions in Figure 2 serve three customers. The first solution is the route that first serves customer 4, then customer 5, and finally, customer 2, and the values of the objective functions are 84.85 (total travel cost), 115.01 (total latency), 119 (total profit), and 3 (nodes served), meanwhile the second solution is the route that first of all visits customer 2, then customer 3, and finally customer 5 with 85.93 (total travel cost), 103.76 (total latency), 98 (total profit), and 3 (nodes served) as values of the objective functions. Comparing both solutions, it can be observed that the first solution obtains a better result for total travel cost but the second solution is able to attain better results for the total latency and the profit when serving the same number of customers.

Now, if we focus on both figures, solutions visiting two customers are better than solutions visiting three customers for the total travel cost and the total latency, however, when visiting three customers instead of two, the profit is better and so the satisfaction of the customers since more customers are served. As consequence, it is said that the four solutions are *efficient*, *non-dominated* or *Pareto-optimal* solutions. To formally define, this concept is needed to introduce the next concepts.

Formally and without loss of generality, a multiobjective minimization problem can be defined as follows:

$$\min_{S \in \Omega} F(S) = \{f_1(S), f_2(S), \dots, f_m(S)\},$$

where Ω is called the feasible set and the image of the feasible set is $F(S) = \{(f_1(S), f_2(S), \dots, f_m(S)) : S \in \Omega\}$ is named the objective space.

Therefore, given two feasible solutions $S_1 \in \Omega$ and $S_2 \in \Omega$ it is said that

- S_1 weakly dominates S_2 , denoted as $S_1 \preceq S_2$, if and only if $f_i(S_1) \leq f_i(S_2), \forall i = 1, \dots, m$;

- S_1 dominates S_2 , denoted as $S_1 < S_2$, if and only if $f_i(S_1) \leq f_i(S_2) \forall i = 1, \dots, m$, and $\exists i_0$ so that $f_{i_0}(S_1) < f_{i_0}(S_2)$;
- S_1 strictly dominates S_2 , denoted as $S_1 \leq \leq S_2$, if and only if $f_i(S_1) < f_i(S_2), \forall i = 1, \dots, m$.

Henceforth, a solution is *efficient*, *nondominated*, or *Pareto -optimal* if there is no other solution that *dominates* it. Hence, the *Pareto front*, also known as the *efficient frontier*, is the set of efficient solutions. In this paper our objective is to find a good approximation of the Pareto front (\hat{PF}) when solving the Mo-TSRPP.

An important decision when designing a multiobjective optimization heuristic is the method to generate solutions that needs a greedy function to that end (in our case, using a GRA procedure) and the improvement method where it is needed to determine when a good movement must be performed. As well as when solving single-objective optimization problems, an improvement is found if the objective function value of the solution S_1 is better (smaller or larger) than the objective function value of S_2 , when solving multiobjective problems an improvement or a good node insertion is not so easy.

The simplest way to consider an improvement is by means of the aggregation of all the objective functions to deal the problem as single objective, that is, $\sum_{i=1}^m \beta_i \cdot f_i(S)$, where $\sum_{i=1}^m \beta_i = 1$, and $\beta_i \in [0, 1] \forall i = 1, \dots, m$ representing the weight or importance of each objective function.

Another possibility to define an improvement was proposed by Martí et al. (2015) who defined pure improvement (random or ordered) and combined improvement (sequential or weighed).

- Pure improvement focuses on considering one single objective until no more improvements can be attained for such objective although permitting deterioration of the other objectives. The term of random or ordered is related to the way of selecting the objective to improve.
- Combined improvement considers, at each step of the procedure, a different objective function therefore, it does not permit deterioration of the other objectives since all of them can be applied. The term of sequential indicates that at each step a different objective function is considered and the weighted combine all objectives in a single objective.

Other authors use the most generalized way although time-consuming; see Pérez-Peló et al. (2019), López-Sánchez et al. (2018), or Sánchez-Oro et al. (2020) who consider an improvement if the Pareto front changes, that is, if a new efficient solution is obtained during the search even though that means that other solutions in the Pareto front are now dominated and so they must be removed from the Pareto front.

In this paper, we have considered that an improvement is reached if the new solution strictly dominates the efficient solution of the same size belonging to the Pareto front. We have opted by this strategy since it prevents an unnecessary increment of the number of efficient solutions in the Pareto front, and also, it sidesteps the issue of comparing the incumbent solution with all the solutions belonging to the Pareto front since this is the main reason why the previous proposals are slower than ours.

To close this section, we would like to highlight the motivation that prompts us to address the Mo-TSRPP. As was introduced in Section 1, the problem appears in a real-world situation in which a freelancer has interest to satisfy not only his/her self-interest by minimizing the total travel cost and maximizing the total profit but also the customers' needs by minimizing the total latency. The

freelancer would like also to obtain the maximum profit, of course, at the minimum cost (usually measured as length or time) however, both objectives are in conflict since it is not possible to improve one objective without deteriorating another one. Simultaneously, customers would like to be served waiting the minimum possible time.

3. Algorithm proposal

In this section, we present the methodology considered to solve the Mo-TSRPP. To that end a deterministic version of the VNS is implemented (Hansen and Mladenović, 2001; Hansen et al., 2017). This methodology is known as VND and it explores several neighborhoods iteratively to improve the incumbent solution. To apply this methodology it is needed to define a set of neighborhood structures. Every time a neighborhood is explored, it is analyzed if an improvement has been found. If so, the search starts again with the first neighborhood. Otherwise, the search continues with the next neighborhood. VND has been successfully applied to solve other combinatorial optimization problems; see, for example, Duarte et al. (2018) and Hertz and Mittaz (2001).

Two important decisions must be made to properly apply the VND algorithm: the selection of neighborhoods and the order of their application. We consider three different neighborhoods for the Mo-TSRPP.

The first neighborhood, named \mathcal{N}_1 , is confirmed by all the solutions that can be reached by applying the 2-exchange operator. This operator exchanges the position of any two nodes in a given solution, and it has a complexity of $O(1)$, since it requires only to evaluate the positions interchanged, and the previous and next nodes.

The operator considered to generate the second neighborhood, \mathcal{N}_2 , replaces a node in the solution by another one that is not still in it. The evaluation of this operator is similar to the previous one, resulting in a complexity of $O(1)$.

The third neighborhood, \mathcal{N}_3 , is based on the well-known 2-opt operator, which basically replaces two edges in the route by other two edges, resulting in a complexity of $O(n^2)$. In order to reduce the computational effort, only the most promising nodes are considered, which are those located at a distance smaller than a certain threshold γ which will be analyzed in the computational results.

In the context of VND, the order in which the neighborhoods explored is usually related to their complexity. In particular, the smallest and fastest to evaluate neighborhoods are first explored, while the largest and slowest neighborhoods are the last ones to be explored. With the aim of confirming or refuting this hypothesis, we have conducted an experiment to evaluate the effect of this order in the final quality of the set of nondominated solutions (see Section 4 for a detailed discussion).

Algorithm 1 shows the pseudo-code for the VND algorithm proposed in this paper, which receives as input parameters an initial approximation of the Pareto front, \hat{PF} , and the neighborhoods, \mathcal{N}_k for $k = 1, 2, 3$.

The algorithm applies the VND methodology to every efficient solution S belonging to the initial approximation of the Pareto front, \hat{PF} (steps 1–11). Every solution S is then improved with the neighborhood k (step 3). If the resulting solution S' is *better* than S , then the algorithm restarts the search from the first neighborhood (steps 5 and 6). Since the concept of improvement is always controversial in a multiobjective context, we would like to highlight that, in this point, an

Algorithm 1. Mo-VND ($\hat{P}F, \mathcal{N}_k$ for $k = 1, \dots, k_{max}$)

```

1: for  $S \in \hat{P}F$  do
2:   while  $k \leq k_{max}$  do
3:      $S' \leftarrow \mathcal{N}_k(S)$ 
4:     if  $S'$  is better than  $S$  then
5:        $S \leftarrow S'$ 
6:        $k = 1$ 
7:     else
8:        $k = k + 1$ 
9:     end if
10:  end while
11: end for return  $\hat{P}F$ 

```

improvement is found when a solution is better in one or more objectives than the original one without deteriorating the remaining objectives. Otherwise, the algorithm moves to the next neighborhood (step 8). Note that every generated solution is evaluated to enter in the set of efficient solutions but, for the sake of simplicity, we have not included that steps in the algorithm.

Having defined the proposed VND, it is necessary to describe how to generate the initial approximation of the Pareto front, $\hat{P}F$, since this will be one of the inputs of VND. The original hypothesis of VNS methodology (Hansen and Mladenović, 2001) suggests that considering random initial solutions does not affect to the performance of the algorithm, but it has been recently shown that VNS performs better (in the context of multiobjective optimization, we consider that a solution is better than another one if the new solution dominates the original one) when starting from promising regions of the search space, such as for the monitor placement problem (Casado et al., 2022a) and for the multiobjective community detection problem (Pérez-Peló et al., 2021). In this work, we propose a constructive method based on the constructive part of GRASP, which includes a diversification phase, namely GRA construction, useful in a multiobjective context such as this research (Feo et al., 1994; Feo and Resende, 1995).

Our proposal starts from an empty solution, in which only the depot is included, $S = \{1\}$, thereby the list of candidates nodes is initialized as $CL = V \setminus S$. The greedy method selects the node $v \in CL$ with the largest value of c_{1v}/p_v for all $v \in CL$ to be added into the solution. Then, once that node v has been added to the solution, $S = \{1, v\}$ and removed from CL , the method iterates until all nodes have been assigned to the solution under construction. In each iteration of the construction phase, all nodes $u \in CL$ are evaluated considering the greedy function $g(S, v) = \min_{u \in CL} c_{vu}/p_u$. This greedy function evaluates the objective function value if the node u were added to the solution under construction, attempting to reach a compromise between more than one objective function: the cost (distance or time) and the profit. In this way, we are not only considering good values for one objective but for more than one simultaneously, allowing a compromise among them.

Note that implementing a GRA construction is similar with the only particularity that is needed to keep the best and worst values of the greedy function, g_{min} and g_{max} , respectively. These values are considered since it is needed to calculate a threshold, $\mu \leftarrow g_{min} + \alpha(g_{max} - g_{min})$, to create a restricted candidate list, RCL . The RCL contains the most promising candidate nodes to be added to the solution under construction and not only the most promising as it will be done in greedy

Algorithm 2. GRA construction ($G = (V, E), \alpha$)

```

1:  $\hat{P}F \leftarrow \emptyset$ 
2:  $S \leftarrow 1$ 
3:  $CL \leftarrow N \setminus \{1\}$ 
4: while  $|S| \leq n$  do
5:    $RCL \leftarrow \{v \in CL : g(S, v) \leq th\}$ 
6:    $v' \leftarrow \text{SelectRandom}(RCL)$ 
7:    $S \leftarrow S \cup \{v'\}$ 
8:    $CL \leftarrow CL \setminus \{v'\}$ 
9:    $\hat{P}F \leftarrow S$ 
10: end while return  $\hat{P}F$ 

```

constructions or all the nodes as will be done in random constructions. Finally, the method randomly selects a node u from the RCL , and adds it to the solution S , then, updating the CL by removing the selected node. The method ends by returning the constructed solution when no more nodes can be added at the solution under construction. The constructive procedure is implemented following the details given in Casado et al. (2022b) to reduce the computational effort. The most relevant part is considering an implicit RCL instead of explicitly creating it, which drastically reduces the computational complexity of the method.

It is important to emphasize that in each iteration of the construction phase, we are generating one efficient solution since we are adding one new node to the solution. Therefore, every time that a node is added to the solution under construction, a new efficient solution will be obtained and, therefore, included in the initial approximation of the Pareto front, $\hat{P}F$ having a total of $|\hat{P}F| = n - 1$ efficient solutions since this is the total number of customers.

Algorithm 2 shows the pseudo-code for the GRA construction used in this paper, which receives as input parameters the graph $G = (V, E)$, and the parameter α that controls the size of the restricted candidate list. As it was previously mentioned, $\alpha = 0$ and $\alpha = 1$ makes the construction completely greedy and random, respectively. It is worth mentioning that as a multiobjective optimization problem is being addressed, the output of the constructive procedure is a set of efficient solutions that will be the initial approximation of $\hat{P}F$ (step 10).

The algorithm starts with an empty initial approximation of the Pareto front where no efficient solutions are included, $\hat{P}F$ (step 1). Then, the depot is forced to be in the solution, S , since the traveler will start the route at that node (step 2). The method will select new nodes until all nodes are included (steps 4–10). Node v' is added to the solution S after being randomly selected from the RCL (step 6) previously computed (step 5). Then, the selected node is included in the solution S (step 7), removed from the candidate list (step 8) and also the solution is included in $\hat{P}F$.

4. Computational results

This section shows and discusses the computational results obtained when solving the Mo-TSRPP using the VND algorithm. To generate instances we have considered the TSPLIB library with 77 instances properly transformed as proposed by Bérubé et al. (2009). For those TSP instances, the euclidean distance between every two points is computed (note that the value is rounded upward)

and furthermore, the profits p_i for all $i \in V$ are generated in three different ways to eliminate the bias that may be produced by the distribution of profits in the results.

- Set 1. $p_i = 1$, for all $i \in V$. In these instances all profits are same and fixed to 1. Note that in this set the maximization of the profit and the maximization of the total number of customers are equivalent.
- Set 2. $p_i = 1 + (7141i + 73) \bmod 100$, for all $i \in V$. Set 2 generates instances with pseudo-random profits between 1 and 100.
- Set 3. $p_i = 1 + \lfloor 99 \frac{c_{ij}}{\theta} \rfloor$, for all $i \in V$ where $\theta = \max_{j \in V} c_{ij}$. Set 3 produces more complex instances in which larger profits are associated with nodes that are further from the depot.

The instances have a size that varies from 51 nodes to 18,512 nodes and a total of $77 \cdot 3 = 231$ instances were solved on an AMD Ryzen 9 5950x 16 cores (3.4 GHz) with 128 GB RAM, and the algorithms were implemented using Java 11. The complete source code of the proposed algorithm will be available, as well as the complete results <https://grafo.etsii.urjc.es/Mo-TSRPP>.

Results presented in this section are divided into three subsections. The first subsection is devoted to fit the parameters of the algorithm and find the best configuration. In the second subsection, a comparison against other well-known multiobjective evolutionary algorithms is performed in order to prove the performance of our proposal. Finally, a subsection where the case of study that motivates this paper is shown and discussed.

Before moving on to the next subsection is needed to briefly describe the multiobjective performance indicators considered to measure the quality of the different algorithms. Thus, we have selected some of the most commonly used multiobjective indicators according to Durillo and Nebro (2011) or Li and Yao (2019). Those indicators will allow us to measure the cardinality of the Pareto front obtained by each algorithm, the proximity of the obtained efficient solutions to the Reference Pareto front, and the diversity of the efficient solutions.

First of all, it is needed to define R , the *Reference Pareto front*, that is an estimation of the optimal Pareto front computed by merging all solutions found by all the different procedures. The reference set is calculated because in this problem, the Mo-TSRPP, the optimal Pareto front is unknown and we have followed this reasoning widely known to estimate the Pareto front.

Then, the following multiobjective performance indicators are considered.

- The *number of efficient solutions*, $|\hat{PF}|$, found by the algorithm to measure the cardinality.
- The *coverage metric*, denoted by CV, evaluates the proportion of solutions of the Reference Pareto front R that weakly dominates the solutions from the Pareto front obtained by the algorithm A . Note that the evaluation of this metric is performed as $CV(R, A)$, where R is the reference front and A is the front under evaluation. Therefore, the smaller the value, the better. For deeply details, see Zitzler (1999).
- The *spread*, represented as Δ , measures the distance from a given solution to its nearest neighbor. $\Delta = 0$ indicates a perfect spread of the points in the Reference Pareto front, that is, an ideal distribution. See Li and Yao (2019).
- The *hypervolume*, hereinafter HV , evaluates the volume in the objective space which is covered by the Pareto front obtained with the algorithm A . Then, the larger the HV value, the better the Pareto front obtained by the algorithm. We refer the reader to Zitzler and Thiele (1998).

- The ϵ -indicator, ϵ , measures the smallest distance ϵ to translate the approximation of Pareto front obtained by algorithm A to dominate every solution in the Reference Pareto front. See Zitzler et al. (2003) for further details.
- The *generated distance*, GD, measures how far are the efficient solutions in the Pareto front obtained by the algorithm A from those efficient solution in the Reference Pareto front. If this indicator reaches the value 0, all the solutions are in the Reference Pareto front. We refer the reader to Menchaca-Mendez and Coello Coello (2015).
- The *inverted generational distance*, IGD is an inversion of the generational distance metric with the aim of measuring the distance from the Reference Pareto front to the efficient solutions obtained by the algorithm A . The smallest the value of IGD, the better. Note that the only different between GD and IGD is that while comparing, the IGD does not miss any part of Reference Pareto front. See Bezerra et al. (2017) for further details.
- The *CPU time*, denoted as Time, required (in seconds) to obtain the set of efficient solutions by each algorithm.

4.1. Algorithm fitting parameter

A previously mentioned, in this subsection the parameters of our algorithm will be fitting. To that end, instead of using the total set of 231 instances, and in order to avoid overfitting, a representative subset with 25% of the instances with different characteristics has been randomly selected. Note that values are averaged among the set of considered instances.

The first experiment is devoted to check if every neighborhood structure applied separately is able to improve the initial set of efficient solutions being part of the Pareto front. To make that experiment, we have started with an initial approximation of the Pareto front that has been built using a totally greedy algorithm. The local search will start from every efficient solution being part of the initial approximation of the Pareto and it will progressively improve it. The algorithm follows a first improvement strategy, that is, it performs the first move that produces an improvement in the incumbent solution. Here, we bring out again the fact that we have considered an improvement if and only if the new solution found strictly dominates the incumbent solution. In this way, instead of comparing the new solution found with all the efficient solutions of the Pareto front, the comparison is just done between pairs of solutions and that is less time-consuming. Of course, when the local search finishes and all pairs of solutions with the same size have been compared to check if an improvement is found (in the sense of strict domination), a final step will be performed. This step consists on checking if all the solutions the Pareto front are nondominated solution.

According to the results of Table 2 all neighborhoods by separate are able to improve the initial approximation of the obtained Pareto front starting from the same totally greedy constructive method denoted as GRE in the table. In the experiment, the VND is executed during a single iteration until reaching the maximum neighborhood. Of course, the larger the complexity of the neighborhoods, the more time-consuming the VND algorithm. As we can see the 2-opt operator (\mathcal{N}_3 shown in the last row) is the most time-consuming and furthermore, the one that is able to get better results in almost all the metrics, except in the spread and the number of efficient solutions in the Pareto front. However, the other two neighborhoods, \mathcal{N}_1 and \mathcal{N}_2 , are also able to improve the

Table 2

Contribution of the neighborhoods (the best value for each metric is highlighted in bold type)

	$ \hat{PF} $	CV	Δ	HV	ϵ	GD	IGD	Time
GRE	573.7333	0.9785	0.2957	0.2516	0.0881	0.0048	0.0456	0.0362
\mathcal{N}_1	571.3778	0.6562	0.2703	0.2654	0.0751	0.0030	0.0335	80.1547
\mathcal{N}_2	573.1556	0.7162	0.2861	0.2512	0.0892	0.0049	0.0453	55.3366
\mathcal{N}_3	546.2000	0.1293	0.2928	0.2904	0.0593	0.0009	0.0121	942.0975

Table 3

Order of the neighborhoods in the VND framework (the best value for each metric is highlighted in bold type)

	$ \hat{PF} $	CV	Δ	HV	ϵ	GD	IGD	Time
$\mathcal{N}_1 + \mathcal{N}_2 + \mathcal{N}_3$	523.3111	0.6079	0.3395	0.2827	0.0723	0.0020	0.0270	643.0006
$\mathcal{N}_1 + \mathcal{N}_3 + \mathcal{N}_2$	527.2444	0.5688	0.3216	0.2831	0.0718	0.0022	0.0263	734.4080
$\mathcal{N}_2 + \mathcal{N}_1 + \mathcal{N}_3$	527.4444	0.5916	0.3283	0.2842	0.0734	0.0021	0.0248	719.8058
$\mathcal{N}_2 + \mathcal{N}_3 + \mathcal{N}_1$	529.7111	0.6008	0.3097	0.2831	0.0672	0.0018	0.0248	704.8509
$\mathcal{N}_3 + \mathcal{N}_1 + \mathcal{N}_2$	527.7333	0.5722	0.3003	0.2861	0.0639	0.0018	0.0238	499.8112
$\mathcal{N}_3 + \mathcal{N}_2 + \mathcal{N}_1$	527.0222	0.5722	0.3113	0.2853	0.0648	0.0019	0.0246	474.2296

initial approximation of the Pareto front obtained by the GRE since all performance metrics are better. In particular, the coverage of GRE is almost 1, which indicates that most of the initial solutions have been improved by any of the neighborhoods and they are now dominated, highlighting the performance of the neighborhood exploration. It is worth mentioning that even though in most indicators \mathcal{N}_1 and \mathcal{N}_2 are not as good as \mathcal{N}_3 , the computing time needed to get those results is too long in comparison with the time consumed by \mathcal{N}_1 and \mathcal{N}_2 .

Another important decision is related to settle the order in which the neighborhoods will be applied. To carry out this experiment, we have started again with the approximation of the Pareto front obtained with the totally greedy algorithm, GRE. According to the results shown in Table 3, we can affirm, first, that the results are very similar in all indicators no matter which order is chosen, but second, the order $\mathcal{N}_3 + \mathcal{N}_1 + \mathcal{N}_2$ is getting the best results. Specifically, first, the 2-opt operator is applied, which is denoted by \mathcal{N}_3 , followed by the removal and insertion of one node, that is, \mathcal{N}_1 , ending with the 2-exchange operator, that is, the \mathcal{N}_2 neighborhood. Thus, the order in which the neighborhoods are explored is inverse related to their complexity, on contrary to the hypothesis that many authors maintain. In particular, the largest and slowest one is first explored, while the smallest and fastest to evaluate are the last one to be explored, as happened in López-Sánchez et al. (2018).

Once come to this point, we would like to check if the initial approximation of the Pareto front will have an impact on the final approximation of the Pareto front. To that end, we have constructed six different initial Pareto fronts using different α parameters of the GRA constructive algorithm. To be more accurate in the definition of the algorithm, this methodology can be viewed as a hybrid combination of GRASP with VND and it has been applied to solve similar problems by López-Sánchez et al. (2018). Table 4 includes $\alpha = 0$, a totally greedy constructive method since the size of the candidate list is restricted just to the most promising node to include in the solution, previously denoted by GRE, and a totally random constructive method with $\alpha = 1$ since the size

Table 4

GRA construction combined with VND algorithm (the best value for each metric is highlighted in bold type)

α	$ \hat{P}F $	CV	Δ	HV	ϵ	GD	IGD	Time
0.00	527.7333	0.0622	0.2856	0.2949	0.0341	0.0004	0.0041	499.8112
0.25	401.1556	0.8942	0.5094	0.1805	0.2729	0.2056	0.1339	1706.6703
0.50	406.0444	0.9060	0.5181	0.1743	0.2873	0.2459	0.1447	1752.7561
0.75	411.6222	0.9229	0.5192	0.1712	0.3020	0.2764	0.1508	1759.8881
1.00	411.2667	0.9131	0.5121	0.1699	0.3030	0.2980	0.1531	1769.9987
RND	401.9778	0.8806	0.5291	0.1812	0.2778	0.2039	0.1356	1738.4523

Table 5

Percentage limitation of the neighborhoods (the best value for each metric is highlighted in bold type)

γ	$ \hat{P}F $	CV	Δ	HV	ϵ	GD	IGD	Time
0.25	531.0667	0.5406	0.2922	0.2778	0.0627	0.0016	0.0205	191.8928
0.50	520.6667	0.5459	0.3189	0.2735	0.0696	0.0020	0.0244	365.8388
0.75	537.9333	0.3827	0.2985	0.2785	0.0642	0.0017	0.0197	452.0900
1.00	527.7333	0.4792	0.3020	0.2804	0.0624	0.0016	0.0194	499.8112

of the restricted candidate list is not restricted at all, but furthermore, α values 0.25, 0.50, 0.75, and *RND*, which means that the restricted candidate list is limited the selection to the 25%, 50%, 75%, and a random value of the most promising nodes according to the greedy function. To perform this experiment, we have fixed as order to explore the different neighborhoods $\mathcal{N}_3 + \mathcal{N}_1 + \mathcal{N}_2$. Then, thanks to the greedy function defined in Section 3, the totally greedy construction (inserting always the best node in the candidate list) find a high-quality approximation of the Pareto front, finishing first and being unbeatable.

To close this section, we have devoted a last experimentation to limit the percentage of nodes that will be candidate to move. Usually, when exploring a neighborhood, all nodes are usually checked to perform the best movement. However, sometimes it is not worth it because some nodes are far away from others it is a very costly task to explore all possible moves. As a measure to reduce the computing time of the overall algorithm, we have limited the number of nodes in the search, of course, the movements will be performed only with a proportion γ of the closest ones, 0.25, 0.50, 0.75, 1.00. Results are depicted in Table 5. As expected, when $\gamma = 1$, that is, when all nodes in the neighborhood are considered in the search, better results are reached regarding almost all the metrics, meaning that the approximation of the Pareto front is better. However, we realize that when we restrict to the 25% of the nodes in the search, the quality does not change in a significant way and the computing time is considerable reduced (−61, 61%). Therefore, we may risk losing some quality in the solution in exchange for being faster. This decision could be made when the size of the problem is large, however, for a small problem, it would not make sense to make this decision. In conclusion, it is not always advisable to narrow down the search. Note that this parameter has not been included in the explanation of the algorithm, since as mentioned, it is just a strategy to reduce the computing time if needed.

4.2. Comparison against other algorithms

Once all the parameters have been set, this section presents and discusses the results of the computational testing conducted when our algorithmic proposal is compared against three of the most widely used multiobjective evolutionary algorithms that will be briefly described: elitist nondominated sorting genetic algorithm (NSGA-II), multiobjective evolutionary algorithm based on decomposition (MOEA/D), and strength Pareto evolutionary algorithm (SPEA2). These genetic algorithms have been solved using an open software MOEA Framework (<http://moeaframework.org>).

- NSGA-II, proposed by Deb et al. (2002), generates an initial population of solutions that will be sorted based on nondomination by means of the allocation of a rank value. Once the sorting is complete, a crowding distance value is calculated to each solution and is measured as the density of solutions around each solution, that is, how close a solution is to its neighbors being the less dense solutions, the preferred to maintain diversity. Next, the selection of every individual solution is carried out using a binary tournament selection with crowded-comparison operator. The codification is done using simulated binary crossover and polynomial mutation. The selected population generates offsprings using crossover and mutation operators. The offsprings are sorted again based on nondomination and only the best N individuals are selected, being N the population size. The selection is based on rank and on crowding distance on the last front.
- MOEA/D, proposed by Zhang and Li (2007), decomposes the multiobjective optimization problem into a number of scalar single-objective optimization subproblems using a set of spread weight vectors. All these single-objective optimization problems are simultaneously solved to approximate the set of efficient solutions. Diversity among these subproblems will naturally lead to diversity in the population. This can be performed using a uniform distribution of efficient solutions. The optimal solutions of two neighboring subproblems should be very similar and the offspring should hopefully be a good solution of such subproblem as well. The process is repeated until a stopping criteria is reached.
- SPEA2, proposed by Zitzler et al. (2001), is another genetic algorithm based on the concept of Pareto domination for fitness allocation and selection operations. SPEA2 uses external archiving elite retention mechanism. The algorithm starts by generating an initial population of solutions and an empty archive set. Next, all efficient solutions are included into the archive and any dominated solutions or any solution that is duplicated are removed from the archive without exceeding the limit size of the updated archive. Next, fitness values are assigned to the archive and to the population members and again individual solutions are selected using a binary tournament selection. The selected population generates offsprings using conventional crossover and mutation operators.

Next, the computational results are shown in a similar way that all results in the previous section. As the literature instances have a wide range of sizes we have considered all instances (231 instances) where the smallest one has 51 nodes and the larger one 18,512 nodes as previously introduced.

As can be observed in Table 6, on average our algorithm proposal consumes 875 seconds approximately. To obtain a fair comparison, we have included a similar limit of time as the stopping criteria for the multiobjective evolutionary algorithms. In all indicators, VND is able to outperform

Table 6

Comparison of VND with the most widely used multi-objective evolutionary algorithms (the best value for each metric is highlighted in bold type)

Algorithm	$ \hat{P}F $	CV	Δ	HV	ϵ	GD	IGD	Time
VND	736.4848	0.0594	0.5038	0.4912	0.3607	0.0005	0.0806	875.4405
MOEA/D	664.8268	0.5705	0.7572	0.3859	0.3847	0.0009	0.1713	754.0176
NSGA-II	675.7446	0.5979	0.7599	0.3829	0.3874	0.0010	0.1723	757.6662
SPEA2	453.5065	0.6884	0.7607	0.2795	0.4539	0.0085	0.2361	752.8614

the multiobjective evolutionary algorithms. It is important to highlight that the number of efficient solutions obtained are significantly larger when using the VND, but also the quality shown by the remaining indicators is able to confirm the high performance of our proposal. Even though if we increase the stopping criteria for the multiobjective evolutionary algorithms, the performance is very similar. Obviously, our algorithm has been specifically designed to solve the problem under consideration and the evolutionary algorithms are more generic algorithms valid to solve many different variants of multiobjective problems.

4.3. Case of study: real-world problem

Once the performance of our proposal has been shown in the previous section, we move on to solve the real-world application using the proposed algorithm where 103 customers require a service.

As previously explained, the problem appears in a real-world situation in which a freelancer, that repairs appliances, has interest to plan the sequence in which all the customers or a subset of it need to be served. To that end, the freelancer provides a list with information about the customers such as the location, to measure distances and travel times, an estimation of the service time according to the incidence, that is, the time to repair the damage, and finally, and the profit obtained once the repair is finished. To show the performance of the proposal we have randomly selected one of the workdays provided by the freelancer with more requests. The freelancer needs an algorithm to plan the daily service in few seconds and therefore, our proposal is a good tool to that end.

The freelancer would like to obtain the maximum profit at the minimum distance, and on the other hand, the customers would like to be served waiting the minimum possible, so the objectives are in conflict.

The algorithm is able to obtain the Pareto front in 0.204 seconds. Moreover, the algorithm reports 103 efficient solutions with every possible solution size (visiting from just one customer to a solution with all customers). Next, 4 of 103 solutions are depicted. It has been selected according to four possible and realistic scenarios according to the four possible total latencies (that can be seen as the working day duration).

- Figure 3 shows the planning when 43 customers are served since its total latency is close to 24 hours, that is, the time that the last customer will wait to be served. On the other hand, the freelancer will obtain a total profit of 2331 monetary units traveling 24.603 kilometers.
- Figure 4 shows the planning when 25 customers are served. The freelancer will require to work

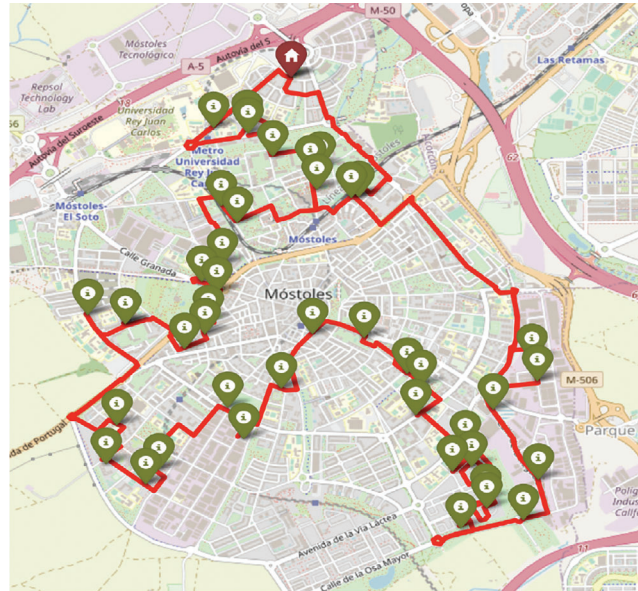


Fig. 3. Planning of a 24-hour shift.

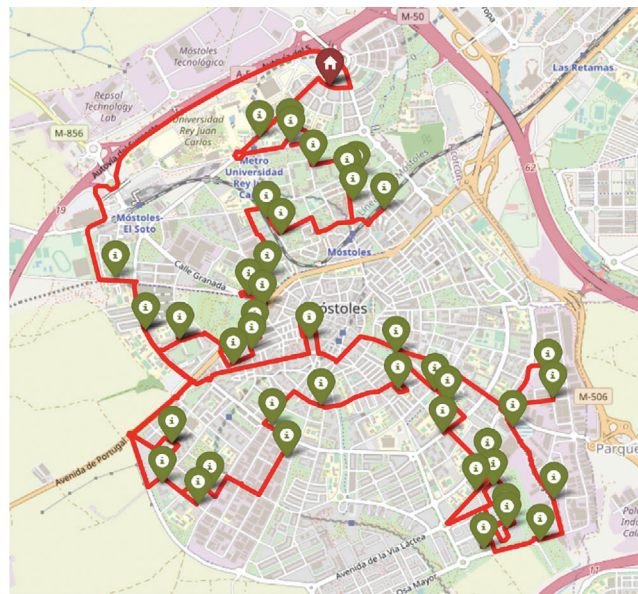


Fig. 4. Planning of a 12-hour shift.

for 12 hours. The distance traveler would be 20.496 kilometers, obtaining a total profit of 1337 monetary units.

- For a full-time day, that is, eight-hour shift, Figure 5 shows the planning visiting 20 customers. In that case, the distance would be 19.192 kilometers and the total profit 1029 monetary units.
- For a partial-time day, that is, four-hour shift, Figure 6 shows the planning in such situation

© 2023 The Authors.

International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies.

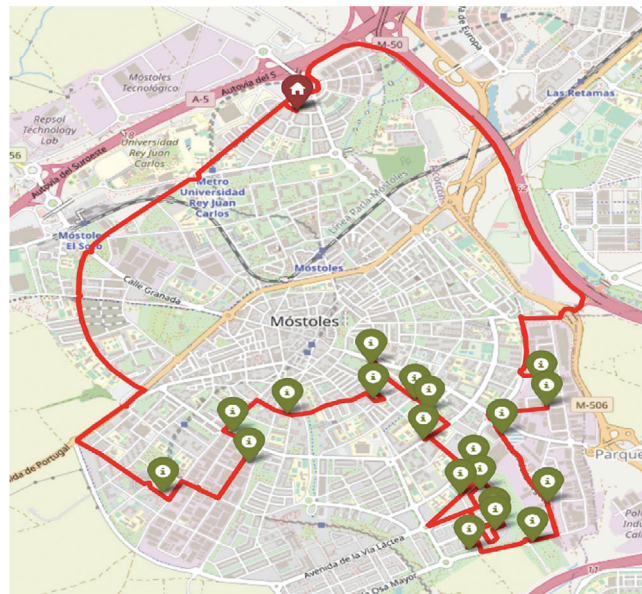


Fig. 5. Planning of an eight-hour shift.

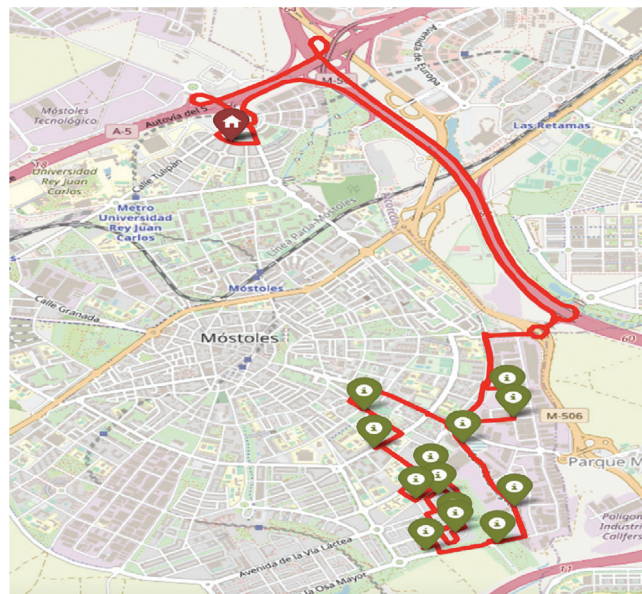


Fig. 6. Planning of a four-hour shift.

visiting 13 customers, traveling 18.638 kilometers and getting a more reduced profit, 720 monetary units.

To close this section, we have included Table 7 to show how our VND algorithm outperforms the greedy algorithm when solving the real-world problem. In view of the results obtained and as

Table 7
VND algorithm versus greedy algorithm

Algorithm	Travel cost (kilometers)	Latency (hours)	Customers served	Profit
VND algorithm	18.638	4	13	720
	19.192	8	20	1029
	20.496	12	25	1337
	18.638	24	43	2331
Greedy algorithm	18.638	4	13	720
	19.512	8	20	1029
	21.306	12	25	1327
	24.091	24	37	1988

expected, the larger the number of customers served, the more difficult it is to solve the considered problem and therefore, the better our VND algorithm works. In all cases, VND algorithm strictly dominates the greedy algorithm except when the latency is four hours that both algorithms get exactly the same solution.

5. Conclusions

This paper introduces an interesting problem, the Mo-TSRPP, in which a subset of nodes must be served optimizing three different objective functions: the total cost (total distance or total working time), the total latency, and the profit. To address the Mo-TSRPP a VND algorithm is proposed. Details to design the VND are carefully considered in order to simplify its implementation. First of all, three neighborhoods have been considered to improve the solutions, with the particularity that they do not change the size of the solutions, controlling in this way the number of customers served. This issue also allows us to define and simplify the concept of an *improvement* to determine when a solution is *better* than other solution. To that end, the algorithm considers an *improvement* if a solution *strictly dominates* another solution of the same size, avoiding a meaningless increment of the Pareto front and furthermore, simplifying the way to check when a new solution becomes part of the Pareto front.

Our computational results indicate that the VND algorithm is able to find a good set of efficient solutions within short computational time and is able to solve a real case of study in few seconds providing the possibility to the freelancer of deciding the most appropriate solution of the Pareto front according to their preferences (the maximum working time, the latency, the profit he wants to earn, or even the number of customers to visit).

As future work, we could merge the proposed algorithm with an interactive method in order to provide a more realistic solution to the repairman. Furthermore, it would be interesting to solve a reasonable extension of the Mo-TSRPP that is more realistic, the k -Mo-TSRPP. In this extension, instead of having just one salesman, k salesmen are available to give the service.

Acknowledgments

The authors wish to express their gratitude to OGA for providing the realistic dataset. J. Sánchez-Oro acknowledges support from the Spanish Ministry of “Ciencia, Innovación y Universidades”

under grant ref. PID2021-125709OA-C22, the “Comunidad de Madrid” and “Fondos Estructurales” of the European Union with grant reference S2018/TCS-4566. A.G. Hernández-Díaz and A.D. López-Sánchez acknowledge support from the Spanish Ministry of “Economía, Industria y Competitividad” through projects PID2019-104263RB-C41, PDC2021-121021-C21, and from Junta de Andalucía, FEDER-UPO Research & Development Call, reference number UPO-1263769.

References

- Angelelli, E., Bazgan, C., Speranza, M.G., Tuza, Z., 2014. Complexity and approximation for traveling salesman problems with profits. *Theoretical Computer Science* 531, 54–65.
- Applegate, D.L., Bixby, R.E., Chvatal, V., Cook, W.J., 2007. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ.
- Arellano-Arriaga, N., Molina, J., Schaeffer, S.E., Álvarez Socarrás, I. A.M. and Martínez-Salazar, 2019. A bi-objective study of the minimum latency problem. *Journal of Heuristics* 25, 3, 431–454.
- Avci, M., Avci, M.G., 2017. A GRASP with iterated local search for the traveling repairman problem with profits. *Computers & Industrial Engineering* 113, 323–332.
- Bérubé, J.F., Gendreau, M., Potvin, J.Y., 2009. An exact epsilon-constraint method for bi-objective combinatorial optimization problems: application to the traveling salesman problem with profits. *European Journal of Operational Research* 194, 1, 39–50.
- Bezerra, L.C., López-Ibáñez, M., Stützle, T., 2017. An empirical assessment of the properties of inverted generational distance on multi-and many-objective optimization. International Conference on Evolutionary Multi-Criterion Optimization. Springer, Berlin, pp. 31–45.
- Bruni, M.E., Khodaparasti, S., Nucamendi-Guillén, S., 2020. The bi-objective minimum latency problem with profit collection and uncertain travel times. ICORES, pp. 109–118. <https://doi.org/10.5220/0009181801090118>.
- Casado, A., Mladenović, N., Sánchez-Oro, J., Duarte, A., 2022a. Variable neighborhood search approach with intensified shake for monitor placement. *Networks* 81, 319–333.
- Casado, A., Pérez-Peló, S., Sánchez-Oro, J., Duarte, A., 2022b. A GRASP algorithm with tabu search improvement for solving the maximum intersection of k-subsets problem. *Journal of Heuristics* 28, 1, 121–146.
- Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2, 182–197.
- Dell' Amico, M., Maffioli, F., Värbrand, P., 1995. On prize-collecting tours and the asymmetric travelling salesman problem. *International Transactions in Operational Research* 2, 297–308.
- Dewilde, T., Catrysse, D., Coene, S., Spieksma, F.C.R., Vansteenwegen, P., 2013. Heuristics for the traveling repairman problem with profits. *Computers & Operations Research* 40, 7, 1700–1707.
- Duarte, A., Mladenovic, N., Sánchez-Oro, J., Todosijević, R., 2018. Variable neighborhood descent. In *Handbook of Heuristics*, pp. 1–27. https://doi.org/10.1007/978-3-319-07153-4_9-1.
- Durillo, J.J., Nebro, A.J., 2011. Jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software* 42, 10, 760–771.
- Feo, T.A., Resende, M.G.C., 1995. Greedy randomized adaptive search procedures. *Journal of Global Optimization* 6, 2, 109–133.
- Feo, T.A., Resende, M.G.C., Smith, S.H., 1994. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research* 42, 5, 860–878.
- Gendreau, M., Hertz, A., Laporte, G., 1992. New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research* 40, 6, 1086–1094.
- Gendreau, M., Laporte, G., Semet, F., 1998. A tabu search heuristic for the undirected selective travelling salesman problem. *European Journal of Operational Research* 106, 2, 539–545.
- Gutin, G., Punnen, A.P., 2006. *The Traveling Salesman Problem and Its Variations*, Vol. 12. Springer Science & Business Media, Berlin.

- Hansen, P., Mladenović, N., 2001. Variable neighborhood search: principles and applications. *European Journal of Operational Research* 130, 3, 449–467.
- Hansen, P., Mladenović, N., Todosijević, R., Hanafi, S., 2017. Variable neighborhood search: basics and variants. *EURO Journal on Computational Optimization* 5, 3, 423–454.
- Hernández-Pérez, H., Rodríguez-Martín, I., Salazar-González, J.J., 2009. A hybrid GRASP/VND heuristic for the one-commodity pickup-and-delivery traveling salesman problem. *Computers & Operations Research* 36, 5, 1639–1645.
- Hertz, A., Mittaz, M., 2001. A variable neighborhood descent algorithm for the undirected capacitated arc routing problem. *Transportation Science* 35, 4, 425–434.
- Issaoui, B., Zidi, I., Marcon, E., Ghedira, K., 2015. New multi-objective approach for the home care service problem based on scheduling algorithms and variable neighborhood descent. *Electronic Notes in Discrete Mathematics* 47, 181–188.
- Jozefowicz, N., Glover, F., Laguna, M., 2008. Multi-objective meta-heuristics for the traveling salesman problem with profits. *Journal of Mathematical Modelling and Algorithms* 7, 177–195.
- Laporte, G., Martello, S., 1990. The selective travelling salesman problem. *Discrete Applied Mathematics* 26, 2, 193–207.
- Li, M., Yao, X., 2019. Quality evaluation of solution sets in multiobjective optimisation: a survey. *ACM Computing Surveys* 52, 2.
- López-Sánchez, A., Hernández-Díaz, A., Gortázar, F., Hinojosa, M., 2018. A multiobjective GRASP–VND algorithm to solve the waste collection problem. *International Transactions in Operational Research* 25, 2, 545–567.
- Lucena, A., 1990. Time-dependent traveling salesman problem? The deliveryman case. *Networks* 20, 6, 753–763.
- Martí, R., Campos, V., Resende, M.G., Duarte, A., 2015. Multiobjective grasp with path relinking. *European Journal of Operational Research* 240, 1, 54–71.
- Menchaca-Mendez, A., Coello Coello, C.A., 2015. GD-MOEA: a new multi-objective evolutionary algorithm based on the generational distance indicator. *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, Berlin, pp. 156–170.
- Mjirda, A., Todosijević, R., Hanafi, S., Hansen, P., Mladenović, N., 2017. Sequential variable neighborhood descent variants: an empirical study on the traveling salesman problem. *International Transactions in Operational Research* 24, 3, 615–633.
- Molina, J., Lopez, A., Hernández-Díaz, A., Martínez-Salazar, I., 2018. A multi-start algorithm with intelligent neighborhood selection for solving multi-objective humanitarian vehicle routing problems. *Journal of Heuristics* 24, 111–133.
- Pei, J., Mladenović, N., Urošević, D., Brimberg, J., Liu, X., 2020. Solving the traveling repairman problem with profits: a novel variable neighborhood search approach. *Information Sciences* 507, 108–123.
- Pérez-Peló, S., Sanchez-Oro, J., Gonzalez-Pardo, A., Duarte, A., 2021. A fast variable neighborhood search approach for multi-objective community detection. *Applied Soft Computing* 112, 107838.
- Pérez-Peló, S., Sánchez-Oro, J., López-Sánchez, A.D., Duarte, A., 2019. A multi-objective parallel iterated greedy for solving the p-center and p-dispersion problem. *Electronics* 8, 12.
- Sánchez-Oro, J., López-Sánchez, A.D., Colmenar, J.M., 2020. A general variable neighborhood search for solving the multi-objective open vehicle routing problem. *Journal of Heuristics* 19, 3, 423–452.
- Silva, B.C., Fernandes, I.F., Goldberg, M.C., Goldberg, E.F., 2020. Quota travelling salesman problem with passengers, incomplete ride and collection time optimization by ant-based algorithms. *Computers & Operations Research* 120, 104950.
- Silva, M.M., Subramanian, A., Vidal, T., Ochi, L.S., 2012. A simple and effective metaheuristic for the minimum latency problem. *European Journal of Operational Research* 221, 3, 513–520.
- Tsiligrídes, T., 1984. Heuristic methods applied to orienteering. *Journal of the Operational Research Society* 35, 797–809.
- Wang, Y., Chen, Y., Lin, Y., 2017. Memetic algorithm based on sequential variable neighborhood descent for the minmax multiple traveling salesman problem. *Computers & Industrial Engineering* 106, 105–122.
- Zhang, Q., Li, H., 2007. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation* 11, 6, 712–731.
- Zitzler, E., 1999. Evolutionary algorithms for multiobjective optimization: methods and applications. Available at <https://sop.tik.ee.ethz.ch/publicationListFiles/zitz1999a.pdf>.
- Zitzler, E., Laumanns, M., Thiele, L., 2001. Spea2: improving the strength Pareto evolutionary algorithm. *TIK Report* 103. <https://doi.org/10.3929/ethz-a-004284029>.

- Zitzler, E., Thiele, L., 1998. Multiobjective optimization using evolutionary algorithms—a comparative case study. Parallel Problem Solving from Nature—PPSN V: 5th International Conference Amsterdam, The Netherlands, September 27–30, 1998, Proceedings 5, Springer, Berlin, pp. 292–301.
- Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G., 2003. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation* 7, 2, 117–132.