DEPARTMENT OF ENGINEERING MANAGEMENT

The *k*-dissimilar vehicle routing problem

Luca Talarico, Kenneth Sörensen & Johan Springael

UNIVERSITY OF ANTWERP
Faculty of Applied Economics
City Campus
Prinsstraat 13, B.226
B-2000 Antwerp
Tel. +32 (0)3 265 40 32
Fax +32 (0)3 265 47 99
www.uantwerpen.be

FACULTY OF APPLIED ECONOMICS

DEPARTMENT OF ENGINEERING MANAGEMENT

The *k*-dissimilar vehicle routing problem

Luca Talarico, Kenneth Sörensen & Johan Springael

RESEARCH PAPER 2013-029
DECEMBER 2013

University of Antwerp, City Campus, Prinsstraat 13, B-2000 Antwerp, Belgium
Research Administration – room B.226
phone: (32) 3 265 40 32
fax: (32) 3 265 47 99
e-mail: joeri.nys@uantwerpen.be

**The research papers from the Faculty of Applied Economics
are also available at www.repec.org
(Research Papers in Economics - RePEc)**

D/2013/1169/029

The k-dissimilar vehicle routing problem

L. Talarico[*], K. Sörensen, J. Springael

University of Antwerp Operations Research Group ANT/OR

Prinsstraat 13, 2000 Antwerp, Belgium

November 2013

In this paper we define a new problem, the aim of which is to find a set of k dissimilar alternative solutions for a vehicle routing problem (VRP) on a single instance. This problem has several practical applications in the cash-in-transit sector and in the transportation of hazardous materials. A min-max mathematical formulation is developed that minimizes the objective function value of the worst solution. A distance measure is defined based on the edges shared between pairs of alternative solutions. An iterative heuristic algorithm to generate k dissimilar alternative solutions is also presented. The solution approach is tested using large and medium size benchmark instances for the capacitated vehicle routing problem.

Key words: Vehicle Routing Problem (VRP), Metaheuristic, Security, Similarity.

1 Introduction

In many European countries, cash-in-transit companies must by law determine several alternative routes for each of their vehicles when transporting cash. The aim of this measure is to allow the company to easily change its plans in case of unforeseen circumstances (e.g., accidents, road works) and to increase security by making the vehicle routes more unpredictable. In this paper, we define a new vehicle routing problem — the k-dissimilar vehicle routing problem or kd-VRP — to support this optimization problem.

A solution of this novel problem consists of k feasible solutions of a single capacitated vehicle routing problem (VRP). Each of these VRP solutions (which we will consistently call *alternative solutions*) must obey the traditional constraints of the VRP: all customers are visited exactly once, all vehicles begin and end at the depot, and the capacity of the vehicle is not exceeded. The quality of an alternative solution is measured as the total distance traveled by all vehicles.

[*]Corresponding author: University of Antwerp, Prinsstraat 13, 2000 Antwerp, Belgium, Tel:+3232654177, E-mail:luca.talarico@uantwerpen.be

Assuming that a similarity metric can be calculated between any pair of alternative solutions, a feasible solution to the *k*d-VRP is a set of *k* feasible alternative solutions for which the difference between each pair of alternative solutions is larger than a certain threshold. The objective of the *k*d-VRP is to minimize the cost of the worst alternative solution in the set. To the best of our knowledge, this problem has never been studied before in the literature.

The *k*d-VRP is closely related to the *m*-peripatetic vehicle routing problem (m-PVRP) studied in Ngueveu et al. (2010a,b). This problem consists in finding a set of edge-disjoint routes of minimal total cost over *m* periods so that each customer is visited exactly once per period and the edge between a pair of customers can be used at most once during the *m* periods. "Periods" in the *m*-PVRP are essentially the same concept as "alternative solutions" in the *k*d-VRP. The difference between the *k*d-VRP and the *m*-PVRP is twofold: in a feasible solution of the *m*-PVRP, no edge is used twice, whereas the double (or triple, ...) usage is not explicitly forbidden, but rather penalized in the objective function of the *k*d-VRP. In other words, multiple usage of an edge is a hard constraint in the *m*-PVRP and a soft constraint in the *k*d-VRP. Secondly, the *m*-PVRP minimizes the total cost over all periods, whereas the *k*d-VRP minimizes the worst-case cost over all alternative solutions. The motivation for the *k*d-VRP is that for some real-life applications (e.g., money collection), the constraint that imposes *k* edge-disjoint VRP solutions in which not a single edge is shared between the alternative solutions might be too stringent. The min-max objective function of the *k*d-VRP is a design choice, that can easily be changed to a total cost objective, in which case the *k*d-VRP generalizes the *m*-PVRP.

Figure 1 shows an example of a solution for the *k*d-VRP with $k = 3$, for which a set containing three dissimilar, but not disjoint, alternative solutions (see sub-figures 1(a), 1(b) and 1(c)) has been generated.

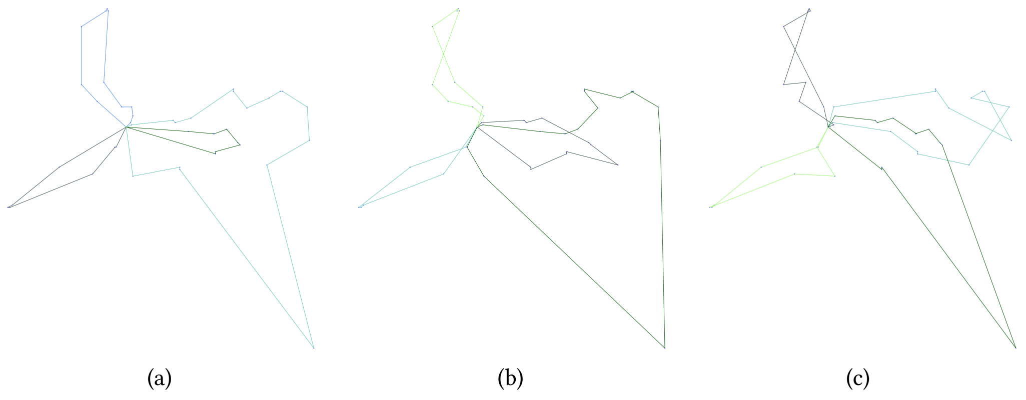


Figure 1: A solution for the *k*d-VRP with $k = 3$, instance F-n45-k4

The *k*d-VRP has many practical applications. A first application, which provided the motivation for this work, can be found in the context of money collection and distribution, also known as the *cash-in-transit* sector. In this context, a set of customers (e.g., banks, shops, casinos, jewelers) needs to be visited to pick up valuables and cash. As mentioned, companies in the cash-in-transit sector are often required by law (see SPFI (2003) for details) to determine several alternative solutions in advance. Additionally, the same alternative solution cannot be used more than two consecutive times. On the other hand, the travel distance of each of these alternative solutions should be minimized for obvious economic reasons. The aim is thus to decrease the predictability of the chosen alternative solution, reducing the risk to be assaulted, while maintaining economic viability.

A second application can be found in the transportation of dangerous materials (e.g., the routing of tankers which must serve a set of petrol stations). In this case a limitation of the number of shared edges between the routes might be required in order to better share and mitigate the risk of accidents (Gopalan et al., 1990).

A third application concerns the design of patrol routes for security agents who must follow partially different routes over time (Wolfler Calvo and Cordone, 2003).

The remainder of the paper is organized as follows. In Section 2, the literature on the peripatetic VRP and similar problems such as the dissimilar k paths problem and the disjoint paths problem is presented. In Section 3, some indices used to measure the (dis)similarity between paths and alternative solutions are also introduced. In Section 4, the k-dissimilar vehicle routing problem (kd-VRP) is described in detail and a mathematical formulation is developed. In Section 5, an iterative metaheuristic to find solutions for the kd-VRP is developed. In Section 6 the solution approach is tested using 51 benchmark instances from the VRP library. Finally Section 7 concludes and provides some suggestions for future research.

The contributions of this paper are the following: (1) A new NP-hard combinatorial optimization problem is proposed, the k-dissimilar vehicle routing problem or kd-VRP. The kd-VRP requires a similarity measure that can calculate the difference between alternative solutions. We define such an index, starting from the similarity index used for the dissimilar k-shortest path problem. (2) A mathematical formulation for the kd-VRP is proposed. (3) An iterative metaheuristic to solve medium and large instance of the kd-VRP is described, implemented and tested.

2 Literature review

As mentioned, the kd-VRP shares several properties with a set of routing problems called *peripatetic*. A solution of the m-peripatetic vehicle routing problem (m-PVRP) consists of a set of m different VRP solutions for which it holds that each pair of alternative solutions is edge-disjoint. In other words, each edge is used at most in one of the m VRP solution. The m-PVRP generalizes two well-known NP-hard problems: the vehicle routing problem (VRP) and the m-peripatetic salesman problem (m-PSP). The latter is a special case of the m-PVRP with one single vehicle of infinite capacity.

The m-PSP was introduced by Krarup (1975), and consists in finding m edge-disjoint Hamiltonian cycles on a graph in such a way that the total distance of all Hamiltonian cycles is minimal. Krarup (1975) proposed a two stage heuristic to find a feasible solution for the m-PSP: first solve a TSP on the initial graph (exactly or by means of a heuristic), remove from the graph all the edges used in the TSP solution and then solve a second TSP on the remaining graph. The heuristic is repeated until m Hamiltonian cycles have been obtained. More recent and efficient approaches to solve the m-PSP problem can be found in Wolfler Calvo and Cordone (2003); De Kort (1993); Duchenne et al. (2007).

Recently a problem similar to the m-PSP has been proposed within a competition (Kaggle, 2012) in which a peripatetic travelling salesman problem needs to be solved to help Santa Claus deliver his presents. Santa's dilemma is slightly different form the traditional m-PSP. In particular Santa likes to see new terrain every year and he does not want his route to be predictable. For this reason, two disjoint alternatives for each TSP are required to be obtained (i.e., if one of the solution contains an edge from A to B, the other path must not contain an edge from A to B or from B to A). The competition thus requires the solution of a m-PSP problem in order to allow Santa to follow a different path every year, and then, for each yearly Santa's TSP tour, a 2 edge-disjoint TSP needs to be solved.

The problem of finding dissimilar solutions has received some attention in the literature on shortest path problems. The k-shortest paths problem (k-SPP) is a generalization of the shortest path problem in which the shortest, the second shortest, until the k-th shortest path from an origin node to a destination node are sought, in increasing order of length. In the literature the generation of the k shortest paths has been widely studied (see, e.g., Yen (1971); Azevedo et al. (1993); Ahuja et al. (1990); Shier (1979); Hershberger et al. (2007)) and many algorithms have been proposed (see, e.g., Feillet et al. (2004); Gotthilf and Lewenstein (2009); Di Puglia Pugliese and Guerriero (2013)).

The k shortest paths are likely to share a large number of edges, and tend to be very similar to each other. For some applications in which dissimilar alternatives are needed, a different approach is required. In Park et al. (2002) a path is considered a reasonable dissimilar alternative to another existing path, by evaluating multiple attributes (e.g., distance, travel time, variability) associated to the edges used in both solutions from an individual's perspective.

To find dissimilar paths, the disjoint-path problem (DPP) can also be used, a classical and important combinatorial optimization problem with several applications. Differently from the classical k shortest paths problem, in the DPP no common edges (edge-disjoint paths) or shared vertices (vertex-disjoint paths) are allowed between the alternative paths (see, e.g., Nguyen (2007)).

However (as also highlighted in Kuby et al. (1997)) for many real-life applications (e.g., hazardous material transportation, couriers, routing in congested network) the constraint that the paths have no edges in common may be too stringent and, due to the impossibility to reuse all the shortest edges employed in the previous solutions, the resulting disjoint paths may be impractically long. In fact, in the majority of real-life transportation applications, in which a minimum number of dissimilar solutions, not necessarily disjoint, is required, the cost of each alternative should be as small as possible. Therefore, a valid alternative to DPP is represented by the path dissimilarity problem (PDP) in which a set of dissimilar solutions with minimum cost are generated. The PDP (see, e.g., Akgün et al. (2000) and Dell'Olmo et al. (2005)) is a bi-objective routing problem in which a set of k paths, from an origin to a destination, must be generated with minimum length and maximum dissimilarity.

In the PDP a set of k alternative paths from an origin to a destination node is generated, using a specific index to measure the similarity between the alternative paths. Nowadays there is also a growing need for satellite navigation services to provide multiple dissimilar alternative paths which reflect a variety of user preferences and a dynamic/stochastic variety of travel times and costs (see Yeonjeong and Dong-Kyu (2011) for more details). For example, in the context of hazardous materials transportation, a spatially dissimilar paths which minimize the risk (distributing the risk over all regional zones to be crossed uniformly) need to be obtained. However, for routing hazardous materials, the spatial dissimilarity between alternative paths may depend of how localized the effects of a spill are. Several algorithm have been proposed for the k dissimilar shortest paths problem. Johnson et al. (1992) introduced the iterative penalty method (IPM) in which a shortest path algorithm is iteratively applied. After each application of the method, the weights associated to the edges in the constructed path are penalized (adding a penalty factor β) to discourage their selection in future constructions. Barra et al. (1993) proposed a edge penalty method in which the network is modified by increasing the cost of all the edges used within the shortest path. The main advantage of the IPM is that it only requires a shortest-path algorithm to generate paths. A drawback is that it relies heavily on the penalization parameter. For example, a small penalty may not achieve the goal of dissimilarity, while a large penalty may eliminate many viable paths from consideration.

To summarize, the PDP generalizes the DPP by replacing the constraint on the disjoint solutions with the constraint on similar solutions. Likewise, the m-PVRP can be extended and generalized by making the constraint that two solutions cannot have any edges in common less stringent. Some attempts in this direction are followed in the literature on VRP applications in which specific indices have been defined and used to measure similarities between solutions.

For example, in Løkketangen et al. (2012), a multiobjective decision support system (DSS) tool is developed in order to produce a set of k dissimilar VRP solution. The dissimilarity between the k VRP solutions is based on an attribute distance function. The distance function includes some measures typical of the decision process and comprise, for example, road accessibility, type and amount of load, road length, road quality, vehicle, and driver.

In Sörensen (2006) a multiobjective optimization approach is proposed in order to find a set of k VRP solution that are "close" (in the solution space) to a given baseline VRP solution and at the same time have a high quality in the sense that their total distance traveled is small. In particular a memetic algorithm with population management is implemented in order to offer to the decision maker a choice of Pareto-optimal solutions, allowing him to make a trade-off between changing his existing solution (i.e., baseline VRP solution) and allowing a longer travel distance.

3 Similarity indices

In order to measure the dissimilarities between solutions several methods and indices have been proposed. The index to measure similarity between alternative solutions that is used in this paper has been borrowed from the literature on shortest path problems in which the concept of dissimilarity between solutions has been widely studied. The dissimilarity measure used in this paper is discussed in Section 4 and can be found in Eq. (6).

In Lombard and Church (1993) the concept of "area under the path" is introduced. If the network is assumed to be representable on a plane, the "area under the path" is the area between the path and the x-axis. Therefore, the dissimilarity between two paths is measured by the absolute difference between the areas under the paths. In Martí et al. (2009) the dissimilarity $dis(P_i, P_j)$ between two paths P_i and P_j, is computed as the average of the distances between each vertex in P_i to the path P_j plus the average of the distances between each vertex in P_j to the path P_i. The dissimilarity measure is given by the formula:

$$Dis(P_i, P_j) = \frac{1}{2}\left[\frac{\sum_{v_i \in P_1} \delta(v_i, P_2)}{\mid P_1 \mid} + \frac{\sum_{u_j \in P_2} \delta(u_j, P_1)}{\mid P_2 \mid}\right] \tag{1}$$

where the value $\delta(v, P_1)$ represents the distance (e.g., euclidean distance) from a vertex to a path $P_1 = \{v_1, v_2, \ldots, v_n\}$ expressed as

$$\delta(v, P_1) = \min_{v_j \in P_1} \delta(v, v_j) \tag{2}$$

The similarity index in formula (1) considers spatial information hence the dissimilarity of the resulting paths will also be a dissimilarity from the spatial point of view. In Akgün et al. (2000) and Vanhove (2012) the dissimilarity is measured in terms of shared edges between paths, without considering spatial information concerning the physical location of the vertices. The expression to compute the dissimilarity between two paths P_1 and P_2, considering only the length (denoted by letter L) of the shared edges, is as follows:

$$Dis(P_1, P_2) = 1 - \frac{1}{2}\left[\frac{L(P_1 \cap P_2)}{L(P_1)} + \frac{L(P_1 \cap P_2)}{L(P_2)}\right] \tag{3}$$

In Dell'Olmo et al. (2005) a concept of "buffer zone" is included in the formula (3) to embed spatial information in the measure of similarity. The "buffer zone" is a zone determined by moving a circle along the path, whose center is the vehicle on the path itself and whose radius is proportional to the impact area due to a possible accident. In Thyagarajan et al. (2005) an extension of the dissimilarity measure in (3) is proposed considering the time context. In fact, in practical military missions, the time difference between routes must be considered.

In order to find k (dis)similar alternative solutions a distance measure to calculate the difference (or similarity) between two given solutions is required. Besides measuring the dissimilarity between two alternative solutions based on the number (or the length) of the common edges, the *edit distance*, which is based on the *Levenshtein distance* might be used (Levenshtein, 1966).

The edit distance can be used for permutation problems i.e., problems of which the solutions are most naturally represented as a permutation of a set of items (i.e., problem attributes), representing the order in which the items appear in each solution (e.g., a VRP solution can be represented as a set of permutations, one for each trip. Each trip is determined by the order in which the customers appear in it.). This distance measure is based on the idea that the distance between two solutions is equal to the "cost" required to transform the first solution into the second one.

A comparison between these distance measures is beyond the scope of this paper. For a more elaborate discussion of some issues related to distance measures, including some other distance measures for permutation problems, we refer to Sörensen (2007) and Løkketangen et al. (2012).

4 Problem description

In this section a mathematical formulation of the kd-VRP is developed, based on an MIP formulation for the VRP. The VRP is seen as a subproblem the solutions of which are the input of a master problem in which the k dissimilar alternative solutions are selected.

The VRP is defined on a complete graph $G = (V, E)$ with vertex set $V = \{0, \ldots, n\}$ and edge set E. Vertices $\{1, \ldots, n\}$ correspond to the customers, while vertex 0 corresponds to the depot. A non-negative cost c_{ij} is associated with each edge $(i, j) \in E$, representing the travel cost between vertices i and j. The cost structure is assumed to be symmetric, i.e., $c_{ij} = c_{ji}\ \forall i, j \in V$. To each customer $i \in \{1, \ldots, n\}$ is associated a known demand $d_i \geqslant 0$, which represents the quantity of goods to be delivered. A set of N identical vehicles, each with capacity C, is available at the depot (it is assumed that $d_i \leqslant C\ \forall i \in \{1, \ldots, n\}$). A general MIP formulation of the VRP (see, e.g., Van Leeuwen and Volgenant (1983) for more details) is presented in (4a)–(4f). This formulation uses a three-index decision variable x_{ij}^{h}, which assumes value 1 if edge (i, j) is traversed by vehicle h, and 0 otherwise.

$$\min \sum_{h \in N} \sum_{(i,j) \in E} c_{ij}\, x_{ij}^h \tag{4a}$$

s.t.

$$\sum_{j \in V \setminus \{0\}} x_{0j}^h = \sum_{j \in V \setminus \{0\}} x_{j0}^h = 1 \qquad \forall h \in N \tag{4b}$$

$$\sum_{h \in N} \sum_{j \in V} x_{ij}^h = \sum_{h \in N} \sum_{j \in V} x_{ji}^h = 1 \qquad \forall i \in V \setminus \{0\} \tag{4c}$$

$$\sum_{i \in V} \sum_{j \in V \setminus \{0\}} d_j\, x_{ij}^h \leq C \qquad \forall h \in N \tag{4d}$$

$$\sum_{h \in N} \sum_{i \in Q} \sum_{j \notin Q} x_{ij}^h \geq 1 \qquad \forall Q \subset V; Q \neq \emptyset \tag{4e}$$

$$x_{ij}^h \in \{0, 1\} \qquad \forall (i,j) \in E; \forall h \in N \tag{4f}$$

The objective function (4a) minimizes the total distance travelled by all vehicles combined. Constraints (4b) force each vehicle to start and finish its route at the depot, visiting at least one vertex along its tour. Constraints (4c) state that every vertex must be visited exactly once, implying that only one vehicle may arrive at a given vertex and depart from it. Constraints (4d) impose a restriction on the maximum load of each vehicle. Finally constraints (4e) ensure that no sub-tours occurs in the solution. Constraints (4f) limit the domain of the decision variable.

The output of the VRP subproblem in (4a)–(4f) is the input of the master problem in (5a)–(5c). The master problem requires two more parameters to be set: the value of the similarity threshold T_S and the number of alternative solutions ($k > 1$).

$$\min \max_{\forall i \in \{1,\ldots,k\}} f(y_i) \tag{5a}$$

s.t.

$$\delta(y_i, y_j) \leq T_S \qquad \forall i, j \in \{1, \ldots, k\}; i \neq j \tag{5b}$$

$$y_i \in \Omega \qquad \forall i \in \{1, \ldots, k\} \tag{5c}$$

A feasible solution for the kd-VRP consists of a subset $S \subseteq \Omega$ with Ω the set of all the feasible alternative solutions for which $\mid S \mid = k$. The objective function (5a) minimizes the cost of the worst alternative solution. Constraints (5b) impose that each pair of alternative solutions in the optimal solution are dissimilar by at least the threshold T_S. Constraints (5c) restrict the domain of the decision variables.

As mentioned, the kd-VRP is a variant of the k disjoint VRP problem. If the similarity threshold T_S assumes a value equal to 0, the alternative solutions are forbidden to have any shared edges.

In principle, if the objective function in equation (5a) is replaced with the the minimization of the total cost of the alternative solutions ($\sum_{i\in\{1,\ldots,k\}} f(y_i)$) then the kd-VRP can be considered as the generalization of the k disjoint VRP problem.

Both the mathematical model in (5a)–(5c) and the solution approach (developed in Section 5) are independent of the specific (dis)similarity index used. Hence, all of the similarity indices discussed in Section 2 could be used in both the formulation and the heuristic optimization algorithm.

In this paper, we have opted for a similarity index derived from the one in formula (3) (see Section 2). Given two alternative solutions y_i and y_j of the kd-VRP, this index compares each route of alternative solution y_i with each route of alternative solution y_j. Let $r^l_{y_i}$ and $r^m_{y_j}$ be the l-th route of solution y_i and the m-th route of solution y_j respectively, then the similarity between these routes must not exceed the similarity threshold T_s.

$$\delta(y_i, y_j) = \max_{l,m\in N} \frac{1}{2} \left[\frac{c_s(r^l_{y_i}, r^m_{y_j})}{c(r^l_{y_i})} + \frac{c_s(r^l_{y_i}, r^m_{y_j})}{c(r^m_{y_j})} \right] \tag{6}$$

The value $c_s(r^l_{y_i}, r^m_{y_j})$ represent the cost of the edges shared between the two routes $r^l_{y_i}$ and $r^m_{y_j}$, and $c(r^l_{y_i})$ (or $c(r^m_{y_j})$) is the cost of route $r^l_{y_i}$ (or $r^m_{y_j}$). A different option (which is not considered in the remainder of this paper) could consider the management of the number of shared edges between routes $r^l_{y_i}$ and $r^m_{y_j}$) instead of the cost of the edges shared between the two routes ($c_s(r^l_{y_i}, r^m_{y_j})$) and the number of edges included in each route instead of the cost of each route ($c(r^l_{y_i})$ or $c(r^m_{y_j})$).

Alternatively, it is possible to consider the number of shared edges between a couple of routes (for the value $w_s(r^l_{y_i}, r^m_{y_j})$) and the number of edges which compose the routes (for the values $w(r^l_{y_i})$ or $w(r^m_{y_j})$) instead of the weights of the shared edges inside the routes ($w_s(r^l_{y_i}, r^m_{y_j})$) or the cost of the routes themselves ($w(r^l_{y_i})$ or $w(r^m_{y_j})$), in formula (6).

5 Metaheuristic description

In this section an iterative metaheuristic to solve the kd-VRP problem is presented. The metaheuristic developed in this paper is similar to the iterative penalty method used in Johnson et al. (1992) and Barra et al. (1993) to find k dissimilar shortest paths. We call this method the Iterative Penalty Method for the kd-VRP (IPM_kd).

Our version of the iterative penalty method differs from the one proposed in Johnson et al. (1992) and Barra et al. (1993) because at each iteration alternative solutions are selected under the conditions of a maximum degree of similarity. To this end the metaheuristic examines solution cost and solution overlaps simultaneously while searching for k alternative solutions.

The IPM_*k*d algorithm is described as follows. After initialization, the IPM_*k*d sequentially generates k alternative solutions, storing each of them in a candidate set $S = \{y_1, y_2, \dots, y_k\}$. After each alternative solution has been found and added to set S, the cost matrix of the underlying VRP is updated, penalizing all edges used in previous alternative solutions. The procedure continues until the desired number of alternative solutions (k) is reached. The penalty structure is multiplicative, i.e., the new cost of each edge is based on the current cost (which may have been penalized before) multiplied with a factor β. The scheme of the IPM_*k*d metaheuristic is summarized in Algorithm 1.

Algorithm 1: IPM_*k*d metaheuristic structure

```
Initialize both $k$d-VRP and Heuristic parameters $k$, $T_s$, $I$, $P$, $\alpha$, $\beta$ and $\omega$;
$l \leftarrow 0$;
while ($l < I$) do
    $S \leftarrow \{\emptyset\}$;
    while ($| S | < k$) do
        $i \leftarrow | S |$;
        $p \leftarrow 0$;
        Let $y_i^*$ be the best $i$-th alternative solution found so far and $f(y_i^*)$ its cost;
        Let $y_i$ be the current $i$-th alternative solution and $f(y_i)$ its cost;
        $y_i^* \leftarrow \{\emptyset\}, y_i \leftarrow \{\emptyset\}, f(y_i^*) \leftarrow \infty, f(y_i) \leftarrow \infty$;
        while ($p < P$) do
            if ($p == 0$) then
                $y_i \leftarrow \mathit{Lin–Kernighan}(y_i) \cup \mathit{Splitting}(y_i)$;
            else
                $y_i \leftarrow \mathit{Perturbation}(y_i^*)$;
            $y_i \leftarrow \mathit{VND}(y_i)$;
            if ($f(y_i) < f(y_i^*)$) then
                $y_i^* \leftarrow y_i$;
                $f(y_i^*) \leftarrow f(y_i)$ ;
            $p++$;
        if ($i == 0$) then
            add $y_i^*$ to $S$;
            $y_i^* \leftarrow \mathit{PenalizationFunction}(y_i^*)$;
            $y_i^* \leftarrow \mathit{VND}(y_i^*)$;
        else
            while ($\delta(y_i^*, y_h) > T_s \; \forall h \in 1, \dots, i-1$) do
                $y_i^* \leftarrow \mathit{PenalizationFunction}(y_i^*)$;
            add $y_i^*$ to $S$;
            if ($| S | == k$) then
                $l++$;
Return the best set $S$ found so far;
```

The internal parameters used by the IPM_*k*d algorithm (referred to as *heuristic parameters*) as well the *k*d-VRP key controls (referred to as *kd-VRP parameters*) are summarized in Table 1.

As shown in Algorithm 1 four basic heuristic components (described in sub-paragraphs 5.1-5.4

Table 1: Heuristic and kd-VRP parameters

Parameter	Description
kd-VRP parameters	
k	Number of alternative solutions to be generated
T_s	Maximum similarity threshold
Heuristic parameters	
I	Number of restarts of the algorithm
P	Number of times the *Perturbation heuristic* is applied
α	Number of closer neighbour vertices to be considered in the Repair heuristic
β	Penalty factor used in the *Penalization function*
ω	Maximum percentage number of routes to be destroyed

respectively) are applied in the solution approach: (1) The *Lin–Kernighan heuristic* followed by the *Prins splitting procedure*; (2) a *Penalization function*; (3) a *VND (Variable Neighbourhood Descent) heuristic*; (4) a *Perturbation.*

5.1 *Lin-Kernighan heuristic* plus *Prins splitting procedure*

To find an initial alternative solution y_i the algorithm employs the *Lin-Kernighan heuristic*, followed by the *Prins splitting procedure*, both using the current cost matrix.

The current solution y_i is then improved using the *VND heuristic* (see Section 5.3) and a diversification mechanism (see Section 5.4) to escape from local optima.

The Lin-Kernighan heuristic described in Lin and Kernighan (1973) is a deterministic approach generally considered to be one of the most effective methods to generate optimal or near-optimal solutions for the symmetric travelling salesperson problem (TSP). The *Lin-Kernighan heuristic* employed in this paper uses the modified version as implemented in Helsgaun (1998, 2000, 2006).

To transform this TSP solution into a VRP solution the *Prins splitting procedure* described in Prins (2004) is used. This procedure creates an auxiliary graph containing $n + 1$ nodes (0 to n), and adds an arc between nodes $i - 1$ and j (with $i \leq j$) if the route visiting the i-th node to the j-th node in the order they appear in the giant tour is feasible. The best possible way to split the giant tour in feasible routes, is determined by finding the shortest path from node 0 to node n in the auxiliary graph. If the shortest path contains the arc from $i - 1$ to j, the giant tour is split between the $i - 1$-th node and the i-th node and between the j-th and the $j + 1$-th node. For a more detailed explanation of this procedure we refer to Prins (2004).

5.2 *Penalization function*

The *Penalization function* updates the cost matrix of the VRP, and is used: (1) after a new feasible alternative solution y_i is added to set S and (2) every time an infeasible alternative solution has been generated. Its purpose is twofold:

- Once a new solution y_i is added to set S, the *Penalization function* increases the cost of all edges used in y_i by a factor $1 + \beta$ ($\beta > 0$). In doing so, the search process is forced to move to a different part of the search space. In order to speed up the search for a new feasible alternative solution y_{i+1} which contains dissimilar routes from the ones of y_i, the *Penalization function* also penalizes the edges that can be obtained by combining the vertices which are in each route of y_i by half of the penalty. In other words, suppose that solution y_i contains a route r in which the sequence of vertices is visited in the following order $[0, 1, 3, 5, 0]$. Then the *Penalization function* increases the cost of the edges $(0; 1)$, $(1; 3)$, $(3; 5)$, $(5; 0)$, $(1; 0)$, $(3; 1)$, $(5; 3)$ and $(0; 5)$ by a factor $(1 + \beta)$. The edges that may be generated by reshuffling the vertices in route r (e.g., $(0; 3)$, $(3; 0)$, $(1; 5)$, $(5; 1)$, $(3; 0)$ and $(0; 3)$) are penalized by a value equal to $(1 + \beta/2)$ (e.g., if $c_{1,5} = 6$ and $\beta = 0.50$ the new cost of the edge $(1; 5)$ will be $\bar{c}_{1;5} = 7.5$).
- If solution y_i is not feasible because its similarity to solution y_j, already in S, exceeds the threshold T_S, the *Penalization function* penalizes the use of the edges which are in common between y_i and y_j. This is done by increasing the cost of the shared edges by a factor $1 + \beta$. This operation forces the algorithm to discard the shared edges, guiding the VND heuristic (see paragraph 5.3) towards a feasible solutions.

If a relatively large penalty is chosen, then edges that appear in the previous alternative solutions are discouraged more heavily. A smaller penalty, on the other hand, allows for more frequent recurrence of edges in the k alternative solutions.

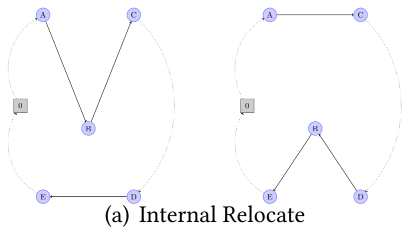
5.3 *VND (Variable Neighbourhood Descent) heuristic*

The *VND heuristic* has a dual purpose. First, it is used every time a new alternative solution has been generated in order to improve it. Secondly it is used after the *Penalization function* has been applied to the current infeasible solution y_i, to make it feasible, discarding the shared edges which make the current solution infeasible. In other words the *VND heuristic* is used both to improve the current solution and to guide the algorithm towards a feasible alternative solution.

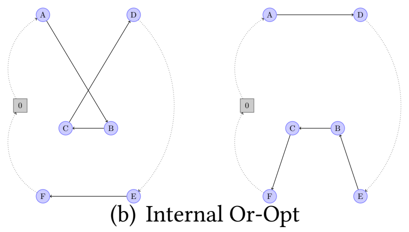
The *VND heuristic* in Algorithm 1 is a sequential Variable Neighbourhood Descent block in which seven different local search operators are used:

- *Intra Route Local Search Operators* which attempt to improve a single route: *Internal Or-Opt* and *Internal Relocate*, *Internal 2-Opt* shown in Figure 2).

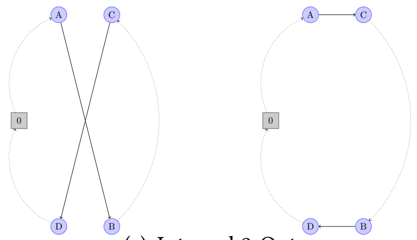
- *Inter Route Local Search Operators* which change more than one route simultaneously. Our VND heuristic implements *External Exchange*, *External Relocate*, *External 2-Opt* and *External Cross-Exchange* shown in Figure 3).



(a) Internal Relocate

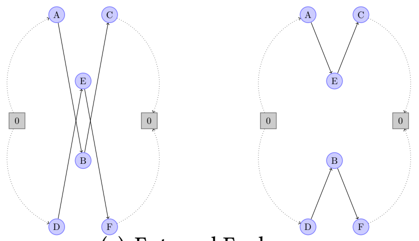


(b) Internal Or-Opt

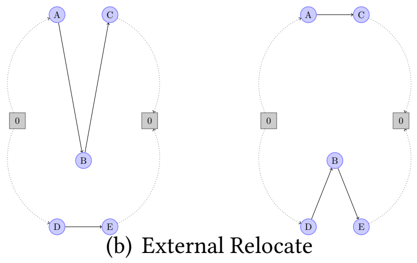


(c) Internal 2-Opt

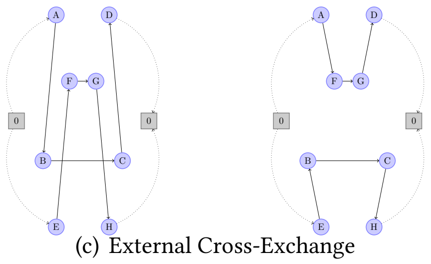
Figure 2: Intra Route Local Search Operators



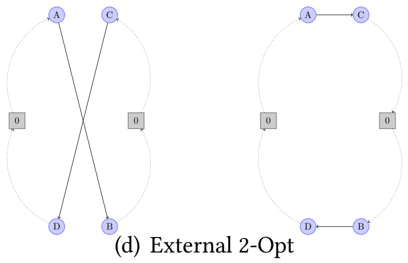
(a) External Exchange



(b) External Relocate



(c) External Cross-Exchange



(d) External 2-Opt

Figure 3: Inter Route Local Search Operators

The *VND heuristic* stops when the current solution cannot be further improved by any of the local search operators, and thus a local optimum has been reached. Each local search operator uses a first-improvement descent strategy, accepting a move that improves the current solution as soon as it is found and restarting the *VND heuristic* from the new current solution.

Finding the right order of neighbourhoods used in a deterministic *VND heuristic* may be of considerable importance for the quality of the solution. We tested different combinations of the order in which the local search operators are used. On average, the most promising order of neighbourhoods is the one shown in Algorithm 2.

Algorithm 2: VND structure

```
Let $y$ be the current solution and $f(y)$ its cost;
Let $y^*$ be the best solution found so far and $f(y^*)$ its cost;
$\lambda \leftarrow 1$ flag $\leftarrow$ false ;
while !flag do
    if ($\lambda = 1$) then
        $y \leftarrow N_{IntOr-Opt}(y)$ ;
    if ($\lambda = 2$) then
        $y \leftarrow N_{IntRelocate}(y)$ ;
    if ($\lambda = 3$) then
        $y \leftarrow N_{Int2-Opt}(y)$ ;
    if ($\lambda = 4$) then
        $y \leftarrow N_{ExtExchange}(y)$ ;
    if ($\lambda = 5$) then
        $y \leftarrow N_{ExtRelocate}(y)$ ;
    if ($\lambda = 6$) then
        $y \leftarrow N_{ExtCross-Exchange}(y)$ ;
    if ($\lambda = 7$) then
        $y \leftarrow N_{Ext2-Opt}(y)$ ;
    if ($f(y) < f(y^*)$) then
        $y^* \leftarrow y$;
        $\lambda \leftarrow 1$
    else
        if ($\lambda < 7$) then
            $\lambda$ + + ;
        else
            flag $\leftarrow$ true ;
return $y^*$ ;
```

5.4 *Perturbation*

The *perturbation heuristic* is used in the IPM_*k*d algorithm as a diversification mechanism to escape from local optima, while looking for the current alternative solution. During the perturbation heuristic a *destroy-and-repair operator*, similar to the one described in Talarico et al. (2013), is used. First, the best alternative solution found so far (y_i^*) is partially destroyed and then it is repaired obtaining a new current solution (y_i). The destroy-and-repair operator takes ω as a parameter, which is the number of routes to be destroyed, as a percentage of the total number of routes, from y_i^*. The destroy-and-repair operator works as follows:

- *Destroy phase*: a random route from the alternative solution y_i^* is selected. All vertices are removed from this route and inserted in a list of unvisited vertices ($\mathscr{L}$). This step is repeated $\omega \cdot N$ times where N represents the number of routes in y_i^*.
- *Repair phase*: the new current alternative solution y_i is generated starting from the non destroyed routes of y_i^* and adding new routes which contain the nodes in $\mathscr{L}$. These new routes are generated applying a greedy randomized nearest neighbourhood heuristic,

using the parameter α, which represents the first α nearest vertices from which selecting the next vertex.

After the application of the destroy-and-repair operator, the new solution y_i, which has been generated, is saved as the current solution and is improved using the *VND heuristic*.

6 Experiments

The IPM_*k*d metaheuristic described in Section 5 has been coded in Java language and it has been extensively tested using a large set of benchmark instances taken from the VRP library. As mentioned, given a similarity measure between alternative solutions, an instance of the VRP can be transformed into an instance of the *k*d-VRP by adding only two parameters: the number of alternative solutions k and the similarity threshold T_s.

In our computational experiments we used 51 medium and large instances available in the VRP literature from different sources (Augerat et al., 1998; Christofides et al., 1979; Fisher, 1994; Taillard, 1993; Golden et al., 1998)[1] ranging from 45 to 484 nodes. The characteristics of the instances are reported in Table 2, the results of the experiment are summarized in Section 6.1.

All computational experiments were performed using a machine with an Intel core i7-2760QM 2.40GHz processor with 4GB RAM.

6.1 Computational results

The computational experiments have been carried out in three different phases. In the first phase the *heuristic parameters* of the IPM_*k*d were tuned (see Table 1) by running a full factorial statistical experiment on a subset of the benchmark instances. A brief description of the heuristic parameters, as well as the tested values, the number of tested values, and the optimal parameter configuration is given in Table 3.

Analyzing the average results obtained over all possible parameter levels the optimal configuration of the IPM_*k*d algorithm was determined (see also Figures 4 and 5). These results show that if the number of times that the IPM_*k*d algorithm is restarted (I) or the number of times that the *Perturbation heuristic* is applied (P) are increased, the quality of the solutions (lower values of the objective functions) improves, but at the expense of increasing the running time.

In the second step of our computational experiments, using the best configuration of the heuristic parameters, all test instances are solved.

In order to analyze the relationship between these values and both the quality of the solution and the computational time, the IPM_*k*d algorithm is tested using different values of the *k*d-VRP parameters: (1) k (number of alternative alternative solutions to be generated); (2) T_s (similarity threshold) and (3) β (penalty factor).

[1]The instances are available at `http://neo.lcc.uma.es/vrp.`

Table 2: VRP instances

Author	Name	# Nodes	Best known VRP Solution
	A-n53-k7	53	1010.0
	A-n65-k9	65	1177.0
	A-n80-k10	80	1764.0
Augerat et al	A-n69-k9	69	1168.0
	B-n56-k7	56	707.0
	B-n68-k9	68	1272.0
	B-n78-k10	78	1221.0
	B-n63-k10	63	1496.0
	vrpnc1	51	524.6
	vrpnc2	76	835.26
	vrpnc3	101	826.14
Christofides et al.	vrpnc4	151	1028.42
	vrpnc5	200	1291.29
	vrpnc11	121	1042.11
	vrpnc12	101	819.56
	F-n45-k4	45	724.0
Fisher	F-n72-k4	72	237.0
	F-n135-k7	135	1162.0
	tai75a	76	1618.36
	tai75b	76	1344.62
	tai75c	76	1291.01
	tai75d	76	1365.42
	tai100a	101	2041.34
	tai100b	101	1940.61
Taillard	tai100c	101	1406.2
	tai100d	101	1581.25
	tai150a	151	3055.23
	tai150b	151	2656.47
	tai150c	151	2341.84
	tai150d	151	2645.39
	tai385	386	24431.44
	kelly1	241	5627.54
	kelly2	321	8447.92
	kelly3	401	11036.23
	kelly4	481	13624.52
	kelly5	201	6460.98
	kelly6	281	8412.88
	kelly7	361	10195.56
	kelly8	441	11663.55
	kelly9	256	583.39
Golden et al.	kelly10	324	742.03
	kelly11	400	918.45
	kelly12	484	1107.19
	kelly13	253	859.11
	kelly14	321	1081.31
	kelly15	397	1345.23
	kelly16	481	1622.69
	kelly17	241	707.79
	kelly18	301	998.73
	kelly19	361	1366.86
	kelly20	421	1821.15

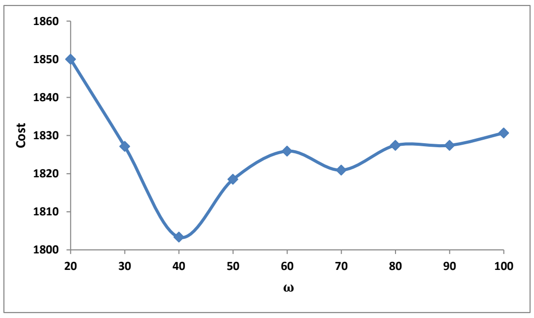


Figure 4: Relationship between ω and the average cost of the obtained solutions

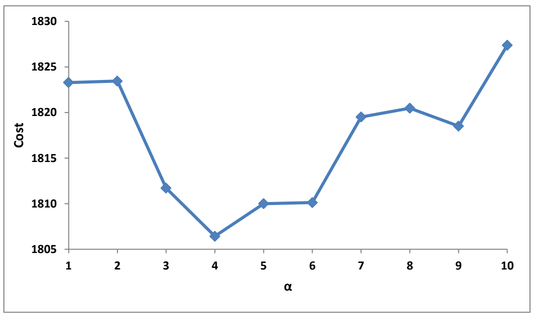


Figure 5: Relationship between α and the average cost of the obtained solutions

Parameter	**Description**	**Values**	**#**	**best setting**
I	Number of times the IPM_kd algorithm is restarted	10	1	**10**
P	Number of times the *Perturbation heuristic* is applied	10	1	**10**
ω	Maximum percentage number of routes in best solution found so far to be destroyed	20,30,40, . . . ,90,100%	9	**40%**
α	Number of closer neighbour vertices to be considered in the Repair heuristic	1,2,3,. . . ,9,10	10	**4**

Table 3: Heuristic parameters

If k increases, the cost of the k-th alternative solution increases, as well as the computational time needed to generate an additional dissimilar alternative solution.

If T_S increases, the cost of the k-th solution decreases, as well the computational time (see figure 6.1). In fact, the higher the threshold, the less constrained the problem and the fewer solutions will be discarded by the heuristic. If T_S is set equal to zero, the kd-VRP problem is reduced to find k disjoint alternative solutions. If T_S assumes values close to 1, the k alternative solutions obtained will share a high number of edges and the cost of the k-th alternative solution solution will tend to approach the cost of the best known VRP solution.

An exploration of the influence of the penalty factor β on both the quality of the solutions and the running times of the algorithm yields the following results. Smaller values of β encourage alternative solutions with lower cost to be generated. However, given that edges which are already used may still appear, the similarity between the alternative solutions may be close to T_s. On the other hand, high values of β discourage already selected edges, favouring dissimilarity at the expense of constructing alternative solutions with higher cost. The lower the penalty, the longer the search for feasible solutions and the better the quality of the solutions. As shown in Figure 6.1, if the value assigned to β increases, the computational time decreases, while the quality of the solution worsens. A good compromise between the quality of the solutions and the computational time might be to choose values of β between 0.10 and 0.20. Our experiments show that when β assumes values equal to 0.10 (or 0.20), the solution gets worse by 0.01 (or 0.02) if compared with the solutions obtained when β is equal to 0.05. However when β is equal

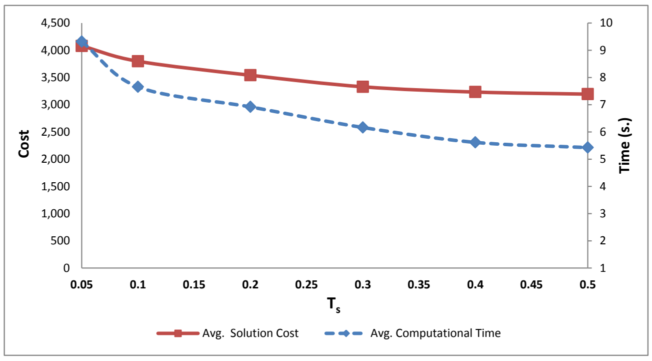


Figure 6: Relationship between the average solutions obtained and the average computational time in relation to different values of the similarity threshold T_S, while $k = 3$

to 0.10 or 0.20 the savings in computational times are respectively 0.08 and 0.14 lower than the computational time needed for β equal to 0.05.

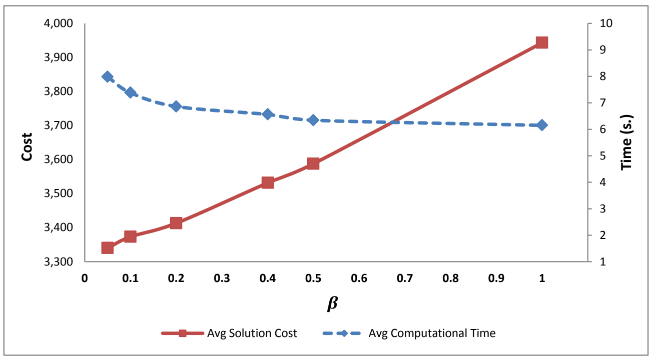


Figure 7: Relationship between the average solutions obtained and the average computational time in relation to different values of the β, while $k = 3$

In the third step of our computational experiments, using the best parameter configuration for the IPM_*k*d, we solved the benchmark instances described in Table 2. For each instance we executed 15 runs fixing the problem parameters as follows:

- The similarity threshold $T_S = 0.20$ (and thus the k alternative solutions, contained in S, must be different from each other by at least 80%);
- The penalty value $\beta = 0.05$.

The results obtained are summarized in Table 4, where for each value of k we report the percentage gap between the best solution obtained after 15 runs and the best known VRP solution (column % BestGap); the percentage gap between the average cost of the solutions obtained during 15 runs and the best known VRP solution (column % AvgGap); the average computational time in seconds (column AvgTime).

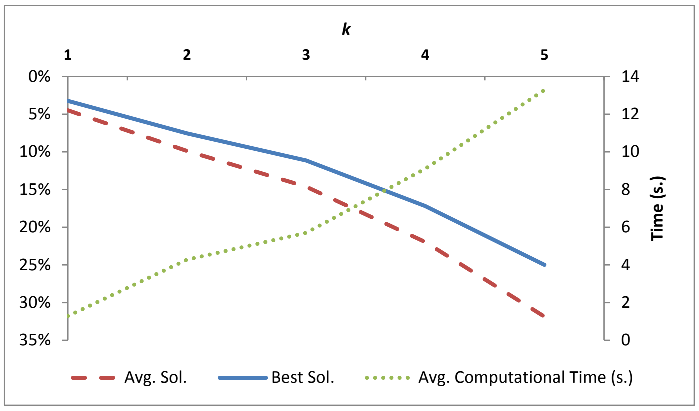


Figure 8: Average computational time, best and average percentage gap from the best known VRP solutions in relation to different values of k

As expected, when k increases the computational time grows approximately as a linear function (see Figure 6.1). On average, when k increases by one (e.g., from $k = 3$ to $k = 4$), the computational time needed to solve the kd-VRP grows by 46%. The relationship between the cost of the kd-VRP and k is shown in figure 6.1. As expected, for kd-VRP instances with a smaller number of vertices, the gap between the cost of the k-th alternative solution and the best known VRP solution is higher than in the case of bigger instances. In fact, when the number of vertices increases, the number of potential edges ($E = V \times V$) increase even more. Hence the possibility to select non shared edges, with a relative low cost, is much higher for larger instances.

However, considering all the 51 benchmark instances, the average percentage gap from the best known VRP solutions is only 25% when 5 alternative solutions are generated that differ by 80% from each other. The algorithm also shows a good level of robustness since the average difference between the costs of the best and the average solutions remains limited to 3.74%. The average computational times also seem encouraging, and the average time needed to solve an instance in the benchmark set, when $k = 5$ and $T_s = 0.20$, is below 14 seconds.

Table 4: Results of the benchmark instances for different values of $k = 2, 3, 4, 5$ by imposing $\beta = 0.05$ and $T_s = 0.20$

	$k = 2$			$k = 3$			$k = 4$			$k = 5$		
id	**% BestGap**	**% AvgGap**	**AvgTime**	**% BestGap**	**% AvgGap**	**AvgTime**	**% BestGap**	**% AvgGap**	**AvgTime**	**% BestGap**	**% AvgGap**	**AvgTime**
A-n53-k7	3.80%	10.08%	0.203	13.84%	19.02%	0.271	20.05%	30.27%	1.262	28.07%	50.91%	1.887
A-n65-k9	7.79%	11.37%	0.215	11.05%	20.57%	0.286	20.90%	30.61%	2.062	24.64%	43.53%	2.920
A-n80-k10	5.83%	7.68%	0.260	7.72%	12.57%	0.347	16.71%	22.30%	1.151	35.95%	45.35%	1.640
A-n69-k9	5.69%	8.32%	0.216	9.36%	12.85%	0.288	13.79%	18.99%	0.543	16.77%	23.16%	1.378
B-n56-k7	6.13%	8.23%	0.197	9.05%	12.90%	0.263	13.14%	16.98%	0.861	17.01%	23.55%	1.354
B-n68-k9	4.54%	5.45%	0.147	6.76%	8.07%	0.196	10.61%	13.06%	1.499	24.70%	33.55%	2.695
B-n78-k10	5.70%	7.74%	0.148	7.97%	11.30%	0.198	13.65%	25.63%	1.555	23.88%	43.87%	4.190
B-n63-k10	5.56%	8.56%	0.194	9.03%	12.15%	0.259	14.23%	19.12%	1.049	20.00%	30.29%	5.382
vrpnc1	11.25%	17.44%	0.132	16.76%	26.91%	0.176	29.28%	38.01%	0.845	39.03%	47.88%	1.295
vrpnc2	6.63%	11.10%	0.189	11.20%	19.13%	0.251	24.16%	31.10%	0.996	41.09%	49.40%	1.649
vrpnc3	4.70%	8.34%	0.217	10.11%	14.77%	0.290	17.18%	23.32%	1.012	21.77%	28.01%	1.521
vrpnc4	7.00%	10.14%	0.409	11.92%	15.18%	0.546	15.05%	23.73%	1.779	21.43%	28.37%	2.077
vrpnc5	8.44%	12.60%	0.883	12.43%	15.84%	1.177	15.46%	18.49%	2.869	20.48%	24.19%	3.816
vrpnc11	1.64%	2.67%	0.181	3.49%	5.28%	0.241	5.71%	8.14%	0.944	9.04%	11.10%	1.420
vrpnc12	6.76%	7.34%	0.179	14.49%	16.79%	0.239	23.15%	24.94%	2.555	31.75%	33.77%	3.363
F-n45-k4	4.34%	5.31%	0.398	5.81%	9.34%	0.531	8.78%	13.16%	0.834	13.24%	17.04%	3.271
F-n72-k4	16.43%	17.94%	0.245	19.79%	26.21%	0.326	27.26%	30.90%	0.896	27.54%	35.44%	4.349
F-n135-k7	7.20%	8.29%	0.528	8.30%	10.18%	0.704	10.23%	12.35%	1.239	11.60%	14.07%	4.699
tai75a	4.31%	5.66%	0.198	7.87%	10.36%	0.264	14.30%	22.33%	1.425	32.13%	48.25%	2.622
tai75b	3.11%	4.30%	0.878	5.09%	7.22%	1.171	9.09%	11.67%	1.606	12.58%	15.90%	5.510
tai75c	7.90%	10.23%	0.253	10.87%	15.06%	0.338	15.58%	20.34%	0.968	21.07%	26.45%	0.997
tai75d	5.06%	6.94%	1.840	10.14%	11.39%	2.453	35.39%	43.18%	2.857	61.60%	89.70%	6.182
tai100a	4.87%	5.88%	0.291	9.16%	12.27%	0.389	15.75%	21.28%	1.243	28.07%	40.64%	1.620
tai100b	4.10%	7.71%	0.269	9.59%	14.38%	0.359	12.41%	19.66%	1.594	22.58%	29.81%	2.257
tai100c	5.27%	6.39%	1.051	6.25%	9.00%	1.401	9.22%	10.93%	1.681	12.90%	18.26%	1.896
tai100d	5.67%	8.62%	0.406	9.29%	12.40%	0.541	13.50%	18.17%	1.207	22.10%	29.08%	1.395
tai150a	5.29%	7.75%	5.956	6.87%	10.60%	7.941	11.40%	24.43%	8.049	33.54%	36.52%	14.240
tai150b	5.36%	6.48%	0.342	6.37%	7.62%	0.456	7.48%	9.06%	1.238	9.96%	10.30%	4.459
tai150c	3.99%	5.40%	3.972	4.73%	7.30%	5.296	8.00%	9.34%	7.202	8.51%	10.61%	11.790
tai150d	4.55%	5.62%	0.424	5.91%	7.76%	0.566	8.22%	10.61%	1.518	12.32%	14.88%	5.630
tai385	11.02%	13.50%	21.506	13.60%	21.84%	28.675	18.37%	35.16%	59.787	20.87%	51.13%	79.656
kelly1	9.63%	12.47%	2.447	10.35%	13.41%	3.262	12.46%	14.82%	4.265	13.92%	16.31%	6.809
kelly2	3.63%	4.35%	4.416	3.64%	4.92%	5.887	5.15%	5.90%	13.117	6.39%	7.48%	16.874
kelly3	9.46%	11.57%	8.513	10.33%	13.09%	11.351	12.14%	16.56%	15.114	16.16%	16.86%	16.808
kelly4	10.27%	10.80%	12.596	15.55%	15.87%	16.795	19.17%	23.58%	21.105	20.78%	24.73%	23.716
kelly5	21.86%	25.70%	2.849	27.72%	34.56%	3.798	36.10%	40.34%	4.152	41.74%	46.72%	4.646
kelly6	12.48%	16.16%	4.492	19.80%	23.49%	5.990	22.42%	26.37%	7.601	26.40%	28.85%	12.622
kelly7	7.35%	8.99%	21.122	11.34%	14.15%	28.163	15.01%	16.60%	38.110	18.08%	20.28%	46.688
kelly8	8.22%	9.60%	5.149	9.49%	12.16%	6.866	16.50%	16.50%	10.990	21.15%	21.81%	16.884
kelly9	7.81%	15.01%	4.860	9.53%	11.13%	6.480	12.98%	14.31%	7.875	27.93%	27.93%	10.070
kelly10	8.79%	10.22%	6.210	12.20%	14.54%	8.280	14.80%	19.60%	11.583	25.76%	32.22%	16.085
kelly11	9.39%	9.68%	8.414	10.52%	13.85%	11.219	15.43%	17.36%	14.782	18.16%	23.43%	21.414
kelly12	9.68%	10.10%	8.887	10.47%	12.34%	11.850	12.82%	16.74%	20.183	22.54%	23.59%	28.781
kelly13	10.40%	12.56%	7.383	23.61%	29.16%	9.845	42.86%	46.87%	12.647	54.81%	60.06%	16.944
kelly14	12.45%	17.17%	10.694	24.73%	27.95%	14.259	43.06%	46.87%	20.332	59.24%	64.50%	26.512
kelly15	12.27%	14.12%	14.698	21.10%	23.88%	19.597	38.36%	43.07%	31.002	53.94%	60.79%	41.390
kelly16	11.48%	14.16%	20.614	17.57%	21.46%	27.485	32.99%	37.54%	45.737	43.54%	52.99%	67.365
kelly17	6.08%	7.82%	4.780	8.40%	9.91%	6.374	11.42%	14.27%	7.659	20.19%	23.28%	10.548
kelly18	7.11%	9.49%	5.990	9.02%	11.38%	7.987	14.19%	17.38%	11.339	22.42%	27.03%	18.145
kelly19	6.75%	9.41%	7.687	7.69%	10.67%	10.250	8.11%	13.18%	12.831	19.91%	24.03%	28.955
kelly20	10.23%	11.22%	13.998	10.38%	11.25%	18.665	13.20%	14.87%	39.319	15.30%	17.28%	59.077
Average	**7.55%**	**9.88%**	**4.277**	**11.14%**	**14.62%**	**5.703**	**17.20%**	**21.96%**	**9.099**	**24.99%**	**31.87%**	**13.284**

7 Conclusions

In this paper we have presented a new combinatorial optimization problem, including a mathematical formulation, the aim of which is to generate a set of k alternative solutions of a single vehicle routing problem instance, in such a way that each alternative solution differs from all the others by at least a given threshold. A min-max objective function minimizes the cost of the worst solution in this set.

The kd-VRP is applicable in several practical situations. In the cash-in-transit sector, the kd-VRP can be used to define a set of alternative routes to pick up cash and valuables, something which is often required by law. Other applications can be found in the fuel distribution or in the transportation of dangerous goods, in which all customer must be served and for specific reasons (e.g., accidents, unavailability of some route edges, security reasons), several alternative routes need to be generated.

To solve this problem, we have developed an iterative method (IPM_kd) based on a similar method for the k dissimilar shortest paths problem. A distance metric between alternative solutions has also been defined. The IPM_kd metaheuristic was tested using 51 VRP benchmark instances varying the number of alternative solutions that need to be generated. The results obtained are encouraging. In a limited computational time we were able to obtain solutions of good quality. In particular, for $k = 5$ and a maximum similarity threshold $T_s = 0.20$, the VRP alternative presenting the highest cost is, on average, only 25% worse than the best known solution of the original VRP problem.

In the future, we plan to investigate extensions of the kd-VRP, e.g., by including additional real life constraints such as time windows, route length restrictions, and precedence relations between vertices. Also the comparison to other algorithms, and the development of exact methods and bounds for this problem, present promising research avenues.

References

R.K. Ahuja, K. Mehlhorn, J. Orlin, and R.E. Tarjan. Faster algorithms for the shortest path problem. *Journal of the ACM*, 37(2):213–223, 1990.

V. Akgün, E. Erkut, and R. Batta. On finding dissimilar paths. *European Journal of Operational Research*, 121(2):232–246, 2000.

P. Augerat, J.M. Belenguer, E. Benavent, A. Corberán, D. Naddef, and G. Rinaldi. Computational results with a branch and cut code for the capacitated vehicle routing problem. Technical report, Research Report 949-M, Universite Joseph Fourier, Grenoble, France, 1998.

J. A. Azevedo, M. E. O. S. Costa, J. J. E. R. S. Madeira, and E. Q. V. Martins. Algorithm for the ranking of shortest paths. *European Journal of Operational Research*, 69(1):97–106, 1993.

T. Barra, B. Perez, and J. Anez. Multidimensional path search and assignment. In *21st PTRC Summmer Annual Conference*, 1993.

N. Christofides, A. Mingozzi, P. Toth, and C Sandi. The vehicle routing problem. In *Combinatorial Optimization*, pages 315–338. John Wiley and Sons, Chichester, 1979.

J.B.J.M. De Kort. A branch-and-bound algorithm for symmetric 2-peripatetic salesman problems. *European Journal of Operational Research*, 70(2):229–243, 1993.

P. Dell'Olmo, M. Gentili, and A. Scozzari. On finding dissimilar Pareto-optimal paths. *European Journal of Operational Research*, 162(1):70–82, 2005.

L. Di Puglia Pugliese and F. Guerriero. Dynamic programming approaches to solve the shortest path problem with forbidden paths. *Optimization Methods and Software*, 28(2):221–255, 2013.

E. Duchenne, G. Laporte, and F. Semet. The undirected m-peripatetic salesman problem: Polyhedral results and new algorithms. *Operations Research*, 55(5):949–965, 2007.

D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3): 216–229, 2004.

M. L. Fisher. Optimal solution of vehicle routing problems using minimum k-trees. *Operations Research*, 42(4):626–642, 1994.

Bruce L Golden, Edward A Wasil, James P Kelly, and I-Ming Chao. The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. In *Fleet management and logistics*, pages 33–56. Springer, 1998.

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

Z. Gotthilf and M. Lewenstein. Improved algorithms for the k simple shortest paths and the replacement paths problems. *Information Processing Letters*, 109(7):352–355, 2009.

K. Helsgaun. *An Effective Heuristic Algorithm for the Traveling-Salesman Problem.* Number 81. Datalogiske Skrifter, Roskilde University, 1998.

K Helsgaun. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

K. Helsgaun. *An Effective Implementation of K-opt Moves for the Lin-Kernighan TSP Heuristic.* Number 109. Datalogiske Skrifter, Roskilde University, 2006.

J. Hershberger, M. Maxel, and S. Suri. Finding the k shortest simple paths: A new algorithm and its implementation. *Transactions on Algorithms*, 3(4):45–56, 2007.

P.E. Johnson, D.S. Joy, and D.B. Clarke. Highway 3.01, an enhancement routing model: program, description, methodology and revised user's manual. Technical report, Oak Ridge National Laboratories, 1992.

Kaggle. Traveling Santa Problem. http://www.kaggle.com/c/traveling-santa-problem, 14 December 2012.

J. Krarup. The peripatetic salesman and some related unsolved problems. In B. Roy, editor, *Combinatorial Programming: Methods and Applications*, volume 19 of *NATO Advanced Study Institutes Series*, pages 173–178. Springer Netherlands, 1975.

M. Kuby, X. Zhongyi, and X. Xiaondon. A minimax method for finding the k best 'differentiated' paths. *Geoghaphical Analysis*, 29(4):298–313, 1997.

V.I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics - Dokladyy*, 10:707–710, 1966.

S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973.

A. Løkketangen, J. Oppen, J. Oyola, and D.L. Woodruff. An attribute based similarity function for VRP decision support. *Decision Making in Manufacturing and Services*, 6(2):65–83, 2012.

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

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

S.U. Ngueveu, C. Prins, and R. Wolfler Calvo. A hybrid tabu search for the m-peripatetic vehicle routing problem. *Matheuristics*, 10:253–266, 2010a.

S.U. Ngueveu, C. Prins, and R. Wolfler Calvo. Lower and upper bounds for the m-peripatetic vehicle routing problem. *4OR*, 8(4):387–406, 2010b.

T. Nguyen. On the disjoint paths problem. *Operations Research Letters*, 35(1):10–16, 2007.

D. Park, S. L. Sharma, L. R. Rilett, and M. Chang. Identifying multiple reasonable alternative routes: Efficient vector labeling approach. *Transportation Research Record: Journal of the Transportation Research Board*, 1783(1):111–118, 2002.

C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers and Operations Research*, 31(12):1985 –2002, 2004.

R. D. Shier. On algorithms from finding the k shortest paths in a network. *Networks*, 9(3):195–214, 1979.

K. Sörensen. Route stability in vehicle routing decisions: a bi-objective approach using metaheuristics. *Central European Journal of Operations Research*, 14(2):193–207, 2006.

K. Sörensen. Distance measures based on the edit distance for permutation-type representations. *Journal of Heuristics*, 13(1):35–47, 2007.

Service Public Federal Interieur SPFI. Arrêté royal réglant certaines méthodes de surveillance et de protection du transport de valeurs et relatif aux spécificités techniques des véhicules de transport de valeurs. Moniteur Belge, 7 April 2003.

É. D. Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 23(8):661–676, 1993.

L. Talarico, K. Sörensen, and J. Springael. The risk-constrained cashi-in-transit vehicle routing problem with time window constraints. In Maartin Josef Geiger Andreas Fink, editor, *14th Workshop of the EURO Working Group "EU/ME : the metaheuristics community"*, pages 104–109, 2013.

K. Thyagarajan, R. Batta, M.H. Karwan, and R.J. Szczerba. Planning dissimilar paths for military units. *Military Operations Research*, 10(1):25–42, 2005.

P.H. Van Leeuwen and A. Volgenant. Solving symmetric vehicle routing problems asymmetrically. *European Journal of Operational Research*, 12(4):388–393, 1983.

S. Vanhove. *Alternative Routing Algorithms for Road Networks.* phd thesis, Ghent University, Department of Applied Mathematics and Informatics, 2012.

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

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

J. Yeonjeong and K. Dong-Kyu. Dissimilar alternative path search algorithm using a candidate path set. In N. Mansour, editor, *Search algorithms and applications*, pages 409–424, 2011.