Variable Neighborhood Search for the Set Orienteering Problem and its application to other Orienteering Problem variants

Robert Pěnička[*], Jan Faigl, Martin Saska

Czech Technical University, Faculty of Electrical Engineering, Technicka 2, 166 27, Prague, Czech Republic

Abstract

This paper addresses the recently proposed generalization of the Orienteering Problem (OP), referred to as the Set Orienteering Problem (SOP). The OP stands to find a tour over a subset of customers, each with an associated profit, such that the profit collected from the visited customers is maximized and the tour length is within the given limited budget. In the SOP, the customers are grouped in clusters, and the profit associated with each cluster is collected by visiting at least one of the customers in the respective cluster. Similarly to the OP, the SOP limits the tour cost by a given budget constraint, and therefore, only a subset of clusters can usually be served. We propose to employ the Variable Neighborhood Search (VNS) metaheuristic for solving the SOP. In addition, a novel Integer Linear Programming (ILP) formulation of the SOP is proposed to find the optimal solution for small and medium-sized problems. Furthermore, we show other OP variants that can be addressed as the SOP, i.e., the Orienteering Problem with Neighborhoods (OPN) and the Dubins Orienteering Problem (DOP). While the OPN extends the OP by collecting a profit within the neighborhood radius of each customer, the DOP uses airplane-like smooth trajectories to connect

[*]Corresponding author

Email addresses: `penicrob@fel.cvut.cz` (Robert Pěnička), `faiglj@fel.cvut.cz` (Jan Faigl), `saskam1@fel.cvut.cz` (Martin Saska)

individual customers. The presented computational results indicate the feasibility of the proposed VNS algorithm and ILP formulation, by improving the solutions of several existing SOP benchmark instances and requiring significantly lower computational time than the existing approaches.

Keywords: Routing, Orienteering Problem, Variable Neighborhood Search

1. INTRODUCTION

The Set Orienteering Problem (SOP) belongs to a large class of routing problems with profits, where the objective is to find a tour that maximizes the collected profit for a given budget, or minimizes the tour length while ensuring at least a predefined profit, or the objective function combines profit maximization with tour length minimization (Feillet et al., 2005).

One of the well-studied routing problems with profits is the Orienteering Problem (OP), which was introduced into operational research by Tsiligirides (1984). The OP stands to find a tour with a limited length that maximizes the profit collected from a visited subset of the given nodes using predefined starting and ending depot locations. The OP thus combines two well-known combinatorial optimization problems, the Knapsack Problem (KP) and the Traveling Salesman Problem (TSP). While the Knapsack part of the problem addresses the maximization of the collected profit by selecting the subset of customers to be visited within the budget, the TSP part finds the sequence to visit selected customers and minimize the tour length in order to fit it within the budget.

The OP has multiple variants and generalizations, as it is shown in surveys by Vansteenwegen et al. (2011) and Gunawan et al. (2016). Among others, the recently introduced SOP proposed by Archetti et al. (2018) is a generalization of the OP where the customers are grouped in clusters.

The profit is associated with the individual clusters, and it is collected by visiting at least one customer in the respective cluster. The SOP has been introduced together with a Mixed-Integer Programming (MIP) formulation and a matheuristic solution algorithm.

Like the OP, the SOP is a combination of the Knapsack problem and the Generalized Traveling Salesman Problem (GTSP), studied in Laporte & Nobert (1983). In the GTSP, the customers are grouped in clusters as in the SOP, and the objective is to minimize the tour length for visiting at least one customer in each cluster. Therefore, the GTSP is an extension of the TSP in the same way as the SOP extends the OP.

The profit collection from clusters in the OP has been previously addressed by the Clustered Orienteering Problem (COP) in Angelelli et al. (2014). However, in the COP, all customers within the respective cluster have to be visited to collect the profit. The problem is solved by means of branch-and-cut and tabu search algorithms.

The Correlated Orienteering Problem (CorOP) by Yu et al. (2016) is also related to the SOP. In the CorOP, the profit collected from customers contains a part of the profit of neighboring customers based on the mutual spatial correlation between the visited customers and the neighboring customers. The CorOP thus creates spatially correlated clusters of the customers, where the profit consists of the individual visited customers together with the distance-weighted profit of the neighboring customers. Therefore, the CorOP can be seen as a hybrid combination of the COP and the SOP, as the profit gained from visiting individual customers consists of the portion of the otherwise unvisited neighboring customer's profit. However, the profit can be increased by visiting more customers within the cluster. The therein presented the exact solution of the CorOP is based on the Mixed-Integer

Quadratic Programming Yu et al. (2016).

The applications of the SOP, originally introduced by Archetti et al. (2018), are in mass distribution, where the carrier chooses to serve only one customer within a cluster of customers that are afterward served by internal distribution within the cluster. However, applications of the SOP far exceed the application originally outlined. In fact, the SOP can be used for any applications of the GTSP, discussed in Laporte et al. (1996), where the salesman has a limited budget, and cluster profits can be used for prioritization. The travel guide problem is an example of such an application, where the guide aims to maximize the profit from visiting attractions in a limited time, but only one attraction of each kind (cluster) would bring the profit.

Similarly, the SOP can be used for a generalization of the traveling salesman with profit-rated customers and a limited budget, where multiple modes of transport are allowed, but with constrained transport changes. Each cluster then consists of multiple departure modes of transport from the individual customer, and the objective is to maximize the profit while using the most suitable transport option to fit within the time budget.

The SOP can also be applied to several other variants of the OP. The Orienteering Problem with Neighborhoods (OPN) is a generalization of the OP, introduced in Faigl et al. (2016), where the profit from each customer can be collected anywhere within the circular neighborhood of the customer location. The OPN can be addressed, like the SOP, by creating clusters of position samples on a circle around the original customer's locations (see Section 4.1). An application example of the OPN is in sensory network information retrieval, where standalone sensor units displaced throughout the environment can wirelessly communicate within a close distance radius, as discussed in Li et al. (2009). To maximize the profit from information collected within a given time, the data collecting vehicle can save travel costs by retrieving the measured information without visiting the precise position of the sensors.

The Dubins Orienteering Problem (DOP), proposed in Pěnička et al. (2017a), can be used for planning package delivery by dropping from an airplane. The DOP addresses the generalization of the OP for the Dubins vehicle model, introduced by Dubins (1957), where the modeled airplane cannot feasibly travel the tour created by the Euclidean OP with sharp turns between the target locations. The Dubins vehicle has to travel between the locations using a curvature-constrained path with a minimum turning radius. By sampling the heading angle of the Dubins vehicle at the target locations, the DOP can be addressed as the SOP presented here (see Section 4.1).

The contributions of this paper are as follows. We introduce a novel ILP formulation of the SOP to find the optimal solution of small to medium-sized problems within less computational time than the existing MIP formulation. The definition of the SOP extends the existing definition by allowing different starting and ending depot clusters, both of which can have multiple nodes. We propose an algorithm for the SOP based on the Variable Neighborhood Search (VNS) metaheuristic, and we show that its computational times are about one order of magnitude shorter for small and medium-sized problems than the existing tabu search solution. The best-known solutions of several benchmark instances are improved by the proposed VNS algorithm for the SOP further denoted as VNS-SOP. Furthermore, we employed the studied SOP in a solution of other OP variants, such as the DOP and the OPN, both newly addressed as the sampling based SOP that is solved optimally. For reuse by the community and to accelerate research on SOP-related problems, both the ILP-based and VNS-SOP algorithms for the SOP are published as open-source software, together with benchmark datasets for comparison.

The remainder of this paper is organized as follows. The description and the formulation of the problem are presented in the next section. The VNS-SOP algorithm is introduced in Section 3. The computational results are presented in Section 4, and final conclusions are outlined in Section 5.

2. Problem description and formulation

The Set Orienteering Problem is a generalization of the OP where customers are grouped in clusters, and the objective is to find a tour with a predefined starting cluster and ending cluster, a restricted budget, and such that the tour maximizes the profit collected from the visited clusters. A cluster is visited when at least one customer belonging to the cluster has been visited. The herein presented SOP formulation builds upon the existing formulation by Archetti et al. (2018) that is extended by considering possibly two different starting and ending clusters, both with the possibility of having multiple nodes instead of a single depot cluster with one node, as in the original formulation.

The SOP can be defined on a directed graph $G = (V, A)$ with a set of vertices $V = \{v_0, \ldots, v_m\}$ and a set of arcs $A = \{a_{ij}\}$. For each pair of vertices v_i, v_j, there exists an arc a_{ij} with cost c_{ij}. The vertices are clustered into disjoint sets $s_0, \ldots, s_n$, with $S = \{s_0, \ldots, s_n\}$, $s_i \cap s_j = \emptyset$ for $i \neq j$, $0 \leq i, j \leq n$, and each vertex $v_{i=0,\ldots,m}$ is associated with exactly one set in S. All sets $s_{i=0,\ldots,n}$ have the associated profit $p_{i=0,\ldots,n}$ for visiting at least one vertex within the set. The starting set s_0 and the ending set s_n are for simplicity the first and the last sets, respectively, both associated with zero profit ($p_0 = p_n = 0$). The objective is to find a tour that maximizes the collected profit P such that its cost does not exceed the given budget $\mathrm{T}_{\max}$. Assuming that the triangle inequality holds for the arc costs, an optimal tour always includes one vertex per visited cluster (see Archetti et al. (2018)).

For instances with a common depot, as in the original SOP formulation Archetti et al. (2018), an additional copy of such a depot can be used as the ending set. Furthermore, the proposed formulation allows multiple vertices in both starting set ($|s_0| \geq 1$) and ending set ($|s_n| \geq 1$).

Any solution of the SOP can be described by a permutation Σ_k of set indexes, according to which the tour visits the individual sets $\Sigma_k = (\sigma_1, \ldots, \sigma_k)$ with $0 \leq \sigma_i \leq n$, $\sigma_i \neq \sigma_j$ for $i \neq j$ and $\sigma_1 = 0$, $\sigma_k = n$. Beside determining the permutation of the sets, the SOP also requires the vertices in the respective visited clusters to be found. The vertices are represented by their respective indexes $\Pi_k = (\pi_1, \ldots, \pi_k)$, $0 \leq \pi_i \leq m$ and $v_{\pi_i} \in s_{\sigma_i}$ for $i \in (1, \ldots, k)$. Using the above notation, the SOP can be defined as follows:

$$
\begin{aligned}
\underset{\Sigma_k, \Pi_k, k}{\mathit{maximize}}\, \mathrm{P} = \sum_{i=1}^{k} p_{\sigma_i} & \\
\text{s.t.}\ \sum_{i=2}^{k} c_{\pi_{i-1}, \pi_i} \leq \mathrm{T}_{\max}\ , & \\
v_{\pi_i} \in s_{\sigma_i} \quad \forall i = 1, \ldots, k\ , & \\
\sigma_1 = 0\ ,\ \sigma_k = n\ . &
\end{aligned}
\tag{1}
$$

The SOP can also be formulated as an Integer Linear Program. Unlike the MIP proposed by Archetti et al. (2018) for the SOP, the proposed formulation does not contain the binary variables of the vertices, and requires only two variable types. Furthermore, the proposed model uses subtour elimination constraints (SECs) like the COP formulation (see Angelelli et al. (2014)), instead of the connectivity cuts of the previous MIP formulation.

The decision variables used in the proposed ILP are:

- y_i: binary variable equal to 1 if at least one customer is visited in the set s_i and 0 otherwise;
- x_{ij}: binary variable equal to 1 if arc a_{ij} is traversed and 0 otherwise.

The proposed ILP formulation of the SOP is:

$$\textit{maximize} \sum_{s_i \in S} p_i y_i, \tag{2}$$

$$\text{s.t.} \sum_{a_{ij} \in A} c_{ij} x_{ij} \leq \mathrm{T}_{\max}, \tag{3}$$

$$\sum_{v_i \in V \setminus \{s_q\}} x_{ij} = \sum_{v_i \in V \setminus \{s_q\}} x_{ji} \qquad \forall s_q \in S \setminus \{s_0, s_n\}, \forall v_j \in s_q, \tag{4}$$

$$\sum_{v_i \in V \setminus \{s_q\}} \sum_{v_j \in s_q} x_{ij} = y_q \qquad \forall s_q \in S \setminus \{s_0\}, \tag{5}$$

$$\sum_{v_i \in V \setminus \{s_q\}} \sum_{v_j \in s_q} x_{ji} = y_q \qquad \forall s_q \in S \setminus \{s_n\}, \tag{6}$$

$$\sum_{v_i \in U} \sum_{v_j \in U} x_{ij} \leq \sum_{s_q \in U \setminus \{s_t\}} y_q \qquad \forall U \subset S \setminus \{s_0, s_n\}, \forall s_t \in U, \tag{7}$$

$$y_0 = 1, y_n = 1, \tag{8}$$

$$y_q \in \{0,1\}, \quad s_q \in S, \qquad x_{ij} \in \{0,1\}, \quad a_{ij} \in A. \tag{9}$$

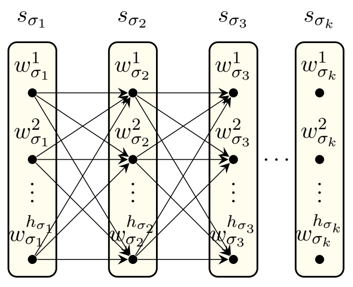
The objective function (2) calls for the maximization of the collected profit. The budget constraint (3) limits the total length of the arcs that are used. Constraints (4) ensure that each vertex, except for those in the starting and ending clusters, has the same number of entering and leaving arcs. Each visited cluster, except for the starting cluster, must have an entering arc. This is ensured by constraints (5). Similarly, constraints (6) ensure that one leaving arc must be selected for all visited clusters different from s_n. Constraints (7) are the SECs. Constraints (8) ensure that both the starting cluster and the ending cluster are visited. Finally, constraints (9) define the domains of the variables.

Both (1) and (2)-(9) aim at finding a permutation of a subset of the clusters and the vertices to visit inside the selected clusters at the same time. However, for the VNS-SOP algorithm, the problem can be partially separated into: (i) selection of the clusters to visit; (ii) determination of the order of visits to the selected clusters; and (iii) selection of the vertices to visit in the chosen clusters. For a given permutation of clusters Σ_k, the solution of (i) and (ii), the subproblem (iii) of selecting individual vertices Π_k within clusters can be addressed as finding the shortest path in a graph of the visited clusters, see Fig. 1a.

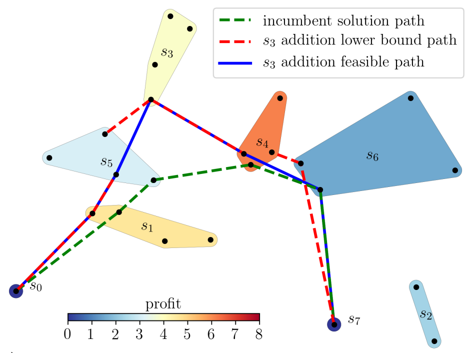
3. Variable neighborhood search algorithm for the SOP

The designed heuristic solution of the Set Orienteering Problem is based on the Variable Neighborhood Search metaheuristic proposed by Mladenović & Hansen (1997) for combinatorial optimization. The metaheuristic uses a greedy initial solution that minimizes the distance per additional profit gained by visiting a new, previously not visited cluster. Afterward, the VNS tries to improve the currently best incumbent solution by a set of predefined neighborhood operators. The VNS metaheuristic was introduced for the OP by Sevkli & Sevilgen (2006), and similar neighborhood operators have been further used for initial solutions of the DOP in Pěnička et al. (2017a) and the OPN in Pěnička et al. (2017b).

In both the SOP initialization procedure and the VNS-SOP algorithm itself, the solution of the SOP is represented only by a sequence of clusters Σ_k. For a given sequence Σ_k with k clusters, the resulting path can be found using a shortest path search in a search graph that is visualized in Fig. 1a as a path connecting the starting cluster $s_{\sigma_1} = s_0$ with the ending cluster $s_{\sigma_k} = s_n$. The graph contains only the arcs between adjacent clusters of Σ_k, and therefore, the shortest path contains exactly one vertex in each cluster of Σ_k and defines the vertices used for a given sequence. The shortest path is found by a dynamic programming breadth-first search storing the shortest path from the starting cluster s_{σ_1} to all vertices in s_{σ_l} iteratively for $l = 2, \ldots, k$ by using already-stored shortest paths to vertices in the preceding cluster and the corresponding arcs connecting adjacent clusters. The shortest path over Σ_k is then defined as the shortest path found among the vertices of ending cluster s_{σ_k}. The proposed algorithm therefore operates with the sequence of clusters and internally calculates the vertices Π_k within the visited clusters such that the overall path length is minimized.



(a) Σ_k with vertices $\{w^1_{\sigma_l}, \ldots, w^{h_{\sigma_l}}_{\sigma_l}\} \in s_{\sigma_l}, \forall l \in (1, \ldots, k)$.



(b) Example of the cluster addition lower bound.

Figure 1: Graph of cluster sequence Σ_k in (a) and the cluster addition lower bound in (b).

The proposed VNS algorithm for the SOP, including the greedy initialization, consists largely of simple cluster sequence modifications, where a single cluster is added, moved or removed from an existing cluster sequence Σ_k. In the case of the cluster addition, the evaluation of the resulting path length requires only to calculate the connection from the previous to the following cluster in the sequence. However, the shortest path from the starting cluster to each vertex in the preceding cluster and also the shortest path from each vertex in the subsequent cluster to the ending cluster has to be known. Therefore, we propose to employ dynamic programming technique to store the shortest paths for each vertex (in the current cluster sequence Σ_k) from the starting and ending clusters to quickly evaluate the simple modifications without searching the shortest path in the whole graph.

The proposed VNS algorithm is further time optimized by using fast denial of the simple sequence modifications that produce solutions with over-budget length. For a typical SOP near-optimal solution, the total path length is close to the budget limit $\mathrm{T}_{\mathrm{max}}$, and almost all modifications, such as cluster addition or movement, produce an over-budget solution. To quickly determine such cases, the lower bound distance between each cluster pair, i.e., the minimal-length arc between the cluster pair, is found and stored before the initial solution of the SOP is created. Then, e.g., for adding the cluster s_3 between the clusters s_5 and s_4, as shown in Fig. 1b, the lower bound is first tested to be within the budget while using the minimal-length arcs from s_5 to s_3 and from s_3 to s_4. The lower bound path further consists of the shortest paths to cluster s_5 from the starting cluster and the shortest path to the ending cluster from s_4, both found as the shortest distance stored by the dynamic programming technique for an incumbent solution among the vertices in s_5 and s_4, respectively. The proposed lower bound can be unfeasible, as it might use different vertices in the cluster being added, the previous and following clusters. However, the feasible solution cannot be shorter, and finding the lower bound is of low complexity, e.g., $\mathcal{O}(|s_5|+|s_4|)$. Thus, simple cluster operations can be found to produce over-budget solutions without searching the vertices to be used in the previous s_5, the newly added s_3, and the following cluster s_4 in the cluster sequence of the feasible solution with the complexity of, e.g., $\mathcal{O}(|s_5||s_3| + |s_3||s_4|)$.

Since both the proposed ILP-based and VNS-SOP solution algorithms for the SOP employ the greedy construction of the initial solution, the procedure is described first, followed by the introduction of the VNS metaheuristic for the SOP.

3.1. Initial solution construction procedure

The proposed construction procedure of the initial solution uses a greedy approach that minimizes the additional length of the path per additional profit. The initially empty sequence of clusters Σ_2 contains only the predefined starting and ending clusters, and the path uses the shortest arc between these depot clusters. Then, in each step of the construction procedure, a non-visited cluster s_i^* and a position j^* for $1 < j^* < k$ within the current sequence Σ_k is found, using the rule

$$s_i^*, j^* = \underset{i \notin \Sigma_k, i \in \Sigma_{k+1}, 1<j<k+1, \sigma_j = i}{\operatorname{argmin}} \frac{\mathcal{L}(\Sigma_{k+1}) - \mathcal{L}(\Sigma_k)}{p_i}. \tag{10}$$

The selection rule (10) uses the difference of the lengths $\mathcal{L}(\Sigma_k)$ and $\mathcal{L}(\Sigma_{k+1})$ of the shortest paths over the cluster sequences Σ_k and Σ_{k+1}, respectively. This requires an evaluation of multiple simple addition operations, and therefore, the cluster distance lower bounds and the dynamic programming technique with storing of the shortest distance to the terminating clusters are used. The initialization procedure terminates as soon as the budget limit does not allow any other non-visited cluster to be added.

3.2. Variable Neighborhood Search algorithm

The Variable Neighborhood Search algorithm consists of two main procedures, the *shake* procedure and the *local search* procedure, which iteratively try to improve the best found incumbent solution. The shake procedure uses random changes of the incumbent solution to get away from a possible local maximum. The local search procedure then extensively searches around the randomly created solution to find a possibly better solution than the actual incumbent. The VNS thus optimizes the incumbent solution using a combination of the shake and local search procedures with the predefined operators in variably large solution space neighborhoods.

In order to uniquely represent any solution of the SOP inside the VNS, the solution is represented by a vector $u = (s_{\sigma_2}, \ldots, s_{\sigma_{k-1}}, s_{\sigma_{k+1}}, \ldots, s_{\sigma_n})$ of all clusters, including the not visited clusters, with the exception of the depot clusters s_{σ_1} and s_{σ_k}. Only the first $k-2$ clusters $(s_{\sigma_2}, \ldots, s_{\sigma_{k-1}})$ are feasibly visited between the starting cluster $s_{\sigma_1} = s_0$ and the ending cluster $s_{\sigma_k} = s_n$ within the $\mathrm{T}_{\max}$ budget limit forming the solution sequence Σ_k. The individual visited vertices Π_k in the respective visited clusters Σ_k are always calculated using the breadth-first search for the shortest path over Σ_k in the graph Fig. 1a. The number of visited clusters in the solution vector u is maximized, i.e., we select the largest k possible for the given u and $\mathrm{T}_{\max}$ such that the ending cluster $s_{\sigma_k} = s_n$ is reached within the budget.

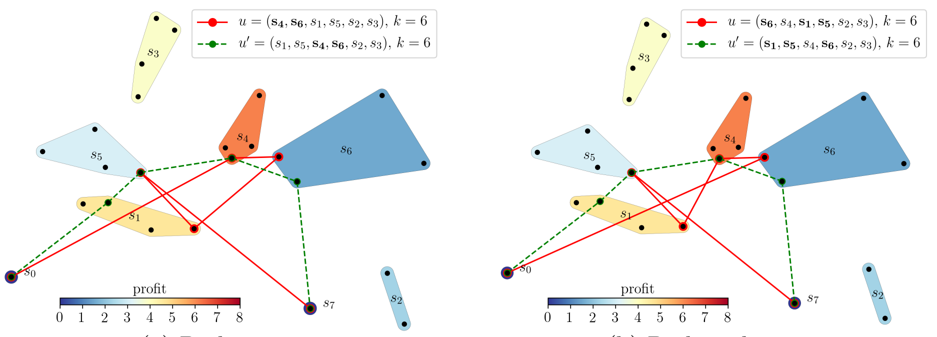
VNS-SOP algorithm is summarized in Alg. 1. It starts with the construction of the initial solution and then tries to improve the solution until the stopping criteria are met. A combination of the maximal computational time together with the limited number of iterations and the number of iterations without improvement is used as the stopping rule. In each iteration, the shake procedure is applied, followed by the local search procedure varying the neighborhood operators based on the variable l (with $1 \leq l \leq l_{max} = 2$). When the profit $P(u'')$ of the solution u'' found by the local search exceeds the profit of the incumbent solution $P(u)$, and its length $\mathcal{L}(u'')$ is within the budget, the incumbent solution is changed to the newly-found solution. The algorithm applies all neighborhood operators during a single iteration and thus increases the size of the examined solution space neighborhood.

Algorithm 1: Variable Neighborhood Search for the SOP

```
Input  : $S$ - customer sets, $\mathrm{T}_{\max}$ - maximal allowed budget
Output: $u$ - solution path
1  $u \leftarrow$ createInitialSolution($S$,$\mathrm{T}_{\max}$)
2  while stopping conditions not met do
3      $l \leftarrow 1$
4      while $l \leq l_{max}$ do
5          $u' \leftarrow$ shake($u$, $l$)
6          $u'' \leftarrow$ local_search($u'$, $l$)
7          if $\mathcal{L}(u'') \leq \mathrm{T}_{\max}$ and $P(u'') > P(u)$ then
8              $u \leftarrow u''$
9              $l \leftarrow 1$
10         else
11             $l \leftarrow l + 1$
```

Shake procedure

The shake procedure of the employed VNS randomly changes the actual incumbent solution to get away from the possible local maximum. It consists of two random operators that modify the solution vector u. By changing u, the operators can change the order of traversing the clusters defining Σ_k and can also add some previously not visited clusters σ_i, $i > k$ to Σ_k. The **Path move** operator and the **Path exchange** operator move or exchange large parts of the solution vector u, and thus create a new solution u' away from the original incumbent. A detail description of the operators follows, and an example of the operators is shown in Fig. 2.


$u = (\mathbf{s_4}, \mathbf{s_6}, s_1, s_5, s_2, s_3)$, $k = 6$
$u' = (s_1, s_5, \mathbf{s_4}, \mathbf{s_6}, s_2, s_3)$, $k = 6$
$u = (\mathbf{s_6}, s_4, \mathbf{s_1}, \mathbf{s_5}, s_2, s_3)$, $k = 6$
$u' = (\mathbf{s_1}, \mathbf{s_5}, s_4, \mathbf{s_6}, s_2, s_3)$, $k = 6$
s_0
s_1
s_2
s_3
s_4
s_5
s_6
s_7
profit
0 1 2 3 4 5 6 7 8

(a) Path move (b) Path exchange

Figure 2: Examples of the shake operators that (a) move the clusters (s_4, s_6) after s_5; and (b) an exchange of (s_6) with (s_1, s_5).

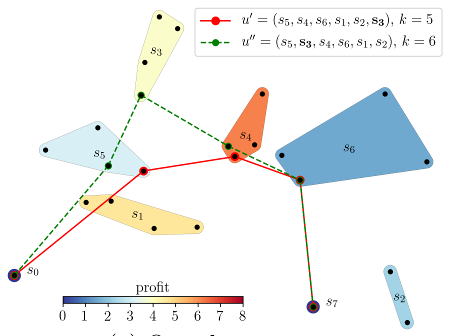
- The **Path move** $(l = 1)$ operator randomly selects a part of the solution vector u and moves it to a randomly selected position. This can be done by selecting three random positions inside u, e.g., $1 < i_1 < i_2 < i_3 \leq n$, $i_{1\dots3} \neq k$, and moving the sequence of clusters $s_{\sigma_j}, i_1 \leq j \leq i_2$ further in u after the cluster $s_{\sigma_{i_3}}$. Alternatively, with the same probability as moving the cluster sequence further in u, a cluster sequence $s_{\sigma_j}, i_2 \leq j \leq i_3$ is moved before $s_{\sigma_{i_1}}$.
- The **Path exchange** $(l = 2)$ operator exchanges two randomly selected non-overlapping parts of the solution vector. Similarly to the path move, the path exchange can be implemented using four randomly selected positions within u, e.g., $1 < i_1 < i_2 < i_3 < i_4 \leq n$, $i_{1\dots4} \neq k$. Afterwards, the cluster sequence $s_{\sigma_j}, i_1 \leq j \leq i_2$ is exchanged with the sequence $s_{\sigma_h}, i_3 \leq h \leq i_4$.

Local search procedure

The VNS local search procedure is used for an extensive search around the randomly created solution vector u' produced by the shake procedure. A close neighborhood of the solution u' is searched using the operators **One cluster move** and **One cluster exchange** to find a better solution.

The implemented local search procedure originates from the randomized variant of the VNS (RVNS) in which the local search operators are applied randomly to the solution vector instead of being applied according to deterministic rules as in the regular VNS. Both operators test simple modifications of the solution vector u', where only one (One cluster move) or two (One cluster exchange) clusters are moved within u'. Each operator tries n^2 such random modifications, and only those not worsening the quality of the solution are applied to u' before examining further modifications.

Each operator thus implements a hill climbing paradigm guaranteeing that no decrease occurs in the solution quality.


$u' = (s_5, s_4, s_6, s_1, s_2, \mathbf{s_3})$, $k = 5$
$u'' = (s_5, \mathbf{s_3}, s_4, s_6, s_1, s_2)$, $k = 6$
s_0
s_1
s_2
s_3
s_4
s_5
s_6
s_7
profit
0 1 2 3 4 5 6 7 8
(a) One cluster move

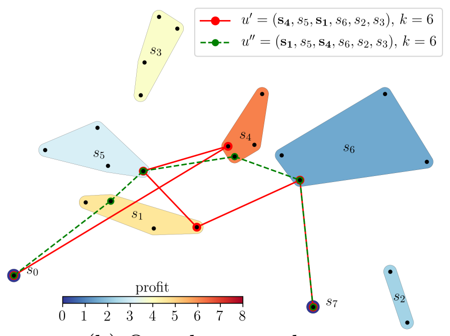

$u' = (\mathbf{s_4}, s_5, \mathbf{s_1}, s_6, s_2, s_3)$, $k = 6$
$u'' = (\mathbf{s_1}, s_5, \mathbf{s_4}, s_6, s_2, s_3)$, $k = 6$
s_0
s_1
s_2
s_3
s_4
s_5
s_6
s_7
profit
0 1 2 3 4 5 6 7 8
(b) One cluster exchange

Figure 3: Example of cluster sequence modifications made by the local search operator.

The two local search operators examine numerous cluster sequence modifications to improve the solution. By employing the dynamic programming technique, with storing the shortest paths inside the solution u', the evaluation of n^2 such modifications is significantly speeded up. Furthermore, each modification of this type is examined in advance to check whether its lower bound does not produce a solution with an over-budget length. The local search operators illustrated in Fig. 3 are as follows:

- The **One cluster move** ($l = 1$) operator repeatedly tries modifications where one random cluster within the solution vector is moved into a different randomly selected position. The modification can be realized by selecting two random positions $1 < i_1 < i_2 \leq n$, $i_{1,2} \neq k$, within the solution vector u'. Afterwards, one cluster is moved either $s_{\sigma_{i_1}}$ after $s_{\sigma_{i_2}}$ or $s_{\sigma_{i_2}}$ before $s_{\sigma_{i_1}}$ with the equal probability. Only modifications not decreasing the solution quality are applied to u' before examining further modifications.

- The **One cluster exchange** ($l = 2$) is similar to the previous local search operator; however, instead of moving one cluster, it exchanges two randomly selected distinct clusters within the solution vector. Using two random indexes $1 < i_1 < i_2 \leq n$, $i_{1,2} \neq k$, a single modification of this operator is made by exchanging clusters s_{i_1} and s_{i_2} in u'. The operator examines n^2 such exchange modifications and always applies only those that do not decrease the solution quality.

4. Computational tests

The proposed VNS-SOP algorithm and the novel ILP formulation have been evaluated on the existing SOP benchmark instances. Furthermore, the methods are tested on the instances of the OP with Neighborhoods (OPN) and the Dubins OP (DOP) addressed as a sampling-based SOP.

Both VNS-SOP and the ILP-based solution algorithms are implemented in C++, and the computational experiments have been performed on a standard PC equipped with an Intel Core i7, clocked at 3.40 GHz, and 16GB of RAM, by using a single core for each run. The ILP formulation (2)-(9) proposed to find the optimal solution of the SOP is solved by means of CPLEX 12.6.1. The subtour elimination constraints (7) are dynamically added to the formulation when found to be violated. The greedy construction procedure used for VNS-SOP is also used for creating an initial feasible solution for the CPLEX solver when addressing the ILP formulation (2)-(9). The maximal computational time for the CPLEX solver has been set to 9 hours.

The stopping condition of VNS-SOP is a combination of the following three criteria: a) the maximum of 2000 iterations, b) 1000 iterations without an improvement, and c) the maximum computational time of 20 minutes. Each problem instance has been solved 20 times to obtain valid statistical results of VNS-SOP.

In the following section, we describe the benchmark datasets that are used. Then, we present the computational results obtained by applying VNS-SOP and by solving the ILP formulation (2)-(9) by means of the CPLEX solver on the GTSP dataset instances used for the SOP in Archetti et al. (2018). Finally, the computational results of the OPN and the DOP are presented, both addressed as a sampling-based SOP.

4.1. Test instances

The evaluated benchmark instances can be categorized into three types. The first is based on the dataset created for the GTSP by Fischetti et al. (1997). The other two datasets are based on the benchmark instances created by Tsiligirides (1984) for the OP that are used to generate test instances for the OPN and DOP with a predefined number of samples forming the clusters for the SOP from the original OP nodes. The OP datasets for the OPN and DOP are $100\times$ scaled and use the rounded up distances between nodes instead of the exact Euclidean distances. The benchmark datasets are available online together with the implementations of the proposed methods[1].

GTSP dataset

The GTSP dataset has been previously used for evaluating the performance of the matheuristic based on tabu search for the SOP (MASOP) proposed in Archetti et al. (2018), and it is therefore used for comparison with the SOP solvers proposed here. To modify the GTSP dataset for the SOP with a single depot, the authors of MASOP removed the first node in each dataset instance from its original GTSP cluster and added the node to a new depot cluster s_0. The budget limit $\mathrm{T}_{\max}$ for individual instances is generated using the ω ratio of the GTSP*, the best known cost of the

[1]`https://github.com/ctu-mrs/vns-sop`

GTSP solution taken from Fischetti et al. (1997) [2]. Two types of cluster profit $\mathsf{p}_g \in \{g_1, g_2\}$ are considered in the dataset. The first, g_1, uses the cluster profit $p_i = |s_i|$ equal to the number of nodes in the respective cluster. The second type, g_2, uses the pseudo-random profit of each node j, with the exception of the depot node $j = 0$ with $p_0 = 0$, equal to $1 + (7141j)mod(100)$ with the consequent cluster profit summed from its respective nodes[3]. As we consider predefined starting and ending clusters in our SOP formulation, the original single depot cluster is duplicated and is used as both a starting cluster and an ending cluster.

OPN dataset

The second benchmark instances use the Set 2 dataset with 21 nodes originally proposed by Tsiligirides (1984) for the OP. In the OPN proposed by Faigl et al. (2016), the profit of individual nodes can be collected within a circular neighborhood of each node with predefined neighborhood radius δ. The solution can be addressed using a sampling-based approach, where for each original node (except the starting and ending nodes) in the dataset, a cluster with o_n equidistantly sampled nodes is created on the δ-radius circle. For all nonterminating original OP nodes with the position $v_i = (x_i, y_i), i \in (2, \ldots, n-1)$, the newly created clusters s_i have o_n sampled neighborhood nodes with the positions $(x_{i,j}, y_{i,j}), i \in (2, \ldots, n-1), j \in (0, \ldots o_n - 1)$. The

[2]In Archetti et al. (2018), the solutions with $\omega = 1$ are expected to collect all clusters within the dataset instance. However, this is not feasible due to the newly created depot cluster, which necessarily adds a travel cost compared to the GTSP* solution. Furthermore, the SOP dataset uses rounded up 'CEIL_2D' edge costs, which is reasonable for the budget limited SOP, but it further increases the length of the shortest cycle over all clusters. The original GTSP dataset uses rounding to the nearest integer value 'EUC_2D'.

[3]The originally proposed g_2 rule for the SOP in Archetti et al. (2018) and previously also used for the COP in Angelelli et al. (2014) uses the profit formula $1 + (7141j + 73)mod(100)$. However, the dataset and its results presented for the SOP match with the formula $1 + (7141j)mod(100)$.

positions of the neighborhood nodes are determined as:

$$(x_{i,j}, y_{i,j}) = (x_i, y_i) + \delta \left(\cos\left(\frac{2j\pi}{o_n}\right), \sin\left(\frac{2j\pi}{o_n}\right)\right). \tag{11}$$

The neighborhood radius used for generating the dataset is $\delta = 50$. Both terminating clusters contain only the original nodes. The created SOP dataset for the OPN thus consists of 21 clusters with $2 + 19o_n$ nodes. The dataset does not contain overlapping clusters although the original OPN can have overlapping δ-radius circles. The SOP dataset for the OPN is approximation of the original instances where more samples o_n better approximates the instances at the cost of the increased number of nodes.

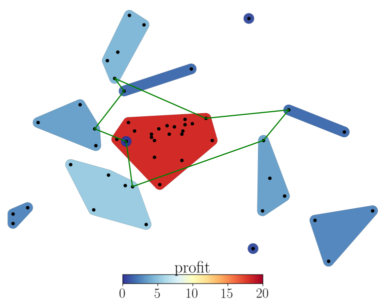
DOP dataset

The second shown variant of the OP solvable as the SOP is the DOP introduced in Pěnička et al. (2017a). In the DOP, the airplane-like vehicle is approximated by the Dubins vehicle model proposed in Dubins (1957). A solution of the OP contains straight line segments between nodes with sharp turns, which are not feasible for the Dubins vehicle. In the DOP, the aerial vehicle has to turn with a given turning radius ρ. Dubins showed that for a curvature-constrained vehicle of this type, the optimal length maneuver between two locations with initial and final heading angles is one of the six possible Dubins maneuvers which satisfy the triangular inequality. The Dubins vehicle state $q = (x, y, \theta)$ can be described by its position in the plane $(x, y) \in \mathbb{R}^2$ and its heading angle $\theta \in S^1$, i.e., its state q belongs to the special Euclidean group $q \in SE(2)$. To solve the DOP, we have to consider the heading angle at each node to connect the consecutive Dubins maneuvers between nodes feasibly, and thus the selection of the heading angles is a part of the optimization due to their influence to the length of the respective Dubins maneuvers. Similarly to the OPN dataset, a sampling-based approach with heading angles at the given nodes can approximate the original DOP by creating clusters of the SOP. For all original nodes v_i, $i \in (1, \dots, n)$, the created clusters s_i contain o_h nodes with equidistantly sampled heading angle $\theta_{i,j}$ for $j \in (0, \dots o_h - 1)$. The individual nodes $q_{i,j}$ representing the Dubins vehicle states are

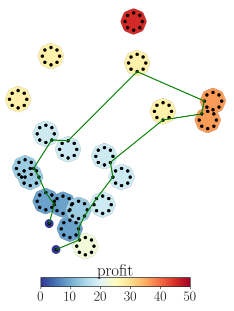
$$q_{i,j} = (x_{i,j}, y_{i,j}, \theta_{i,j}) = \left(x_i, y_i, \frac{2j\pi}{o_h} \right). \tag{12}$$

The minimal turning radius used in the created dataset is $\rho = 50$. The dataset for the DOP consists of asymmetric SOP instances, as the Dubins maneuver has a different length when the initial and final vehicle states of the maneuver are exchanged.

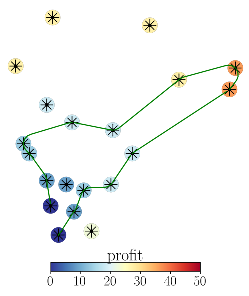
An example of the found solutions of the SOP on the GTSP, OPN and DOP test instances is shown in Fig. 4. Figure 4a shows the solution on the GTSP 11berlin52 dataset for $\omega = 0.6$ and $\mathsf{p}_g = g_1$. Figure 4b and 4c are example solutions of the OPN and DOP both with $\mathrm{T}_{\max} = 3000$ on the Set 2 dataset, using $o_n = 8, \delta = 50$ in the case of the OPN and $o_h = 8, \rho = 50$ for the DOP.



(a) 11berlin52



(b) OPN Set 2



(c) DOP Set 2

Figure 4: Example solutions of the SOP on selected dataset instances.

4.2. Computational results on the GTSP dataset

The proposed ILP formulation and VNS-SOP have been evaluated on the GTSP dataset instances, and have been compared with the existing MIP formulation and the matheuristic based on tabu search (MASOP), both proposed by Archetti et al. (2018).

The results shown in Table 1 concern small instances with up to 76 nodes and 16 clusters. Both cluster profit types $\mathsf{p}_g \in \{g_1, g_2\}$ are considered together with various ω ratios of the GTSP* solution and the corresponding budget limit $\mathrm{T}_{\mathrm{max}}$. For each method are reported the collected profit P and the computational time T in seconds. The collected profit of VNS-SOP has been identical in all runs, and the reported computational time is the average from 20 runs.

Table 1: Comparison with exisitng methods on small GTSP dataset instances.

instance	p_g	ω	$\mathbf{T_{max}}$	MIP		MASOP		ILP		VNS-SOP	
				P	**T**	**P**	**T**	**P**	**T**	**P**	**T**
11berlin52	g_1	0.4	1616	37	47.07	37	1.75	37	1.08	37	0.11
11berlin52	g_2	0.4	1616	1829	65.96	1829	1.70	1829	1.18	1829	0.11
11berlin52	g_1	0.6	2424	43	777.88	43	2.40	43	4.24	43	0.16
11berlin52	g_2	0.6	2424	2190	1532.91	2190	2.64	2190	1.34	2190	0.15
11berlin52	g_1	0.8	3232	47	2648.04	47	7.17	47	4.63	47	0.19
11berlin52	g_2	0.8	3232	2384	3833.50	2384	6.61	2384	7.67	2384	0.19
11eil51	g_1	0.4	69	24	39.72	24	1.85	24	2.54	24	0.09
11eil51	g_2	0.4	69	1279	40.13	1279	1.97	1279	2.81	1279	0.09
11eil51	g_1	0.6	104	39	34.64	39	5.13	39	1.67	39	0.14
11eil51	g_2	0.6	104	1911	204.65	1911	4.74	1911	3.01	1911	0.14
11eil51	g_1	0.8	139	43	1586.67	43	2.30	43	16.51	43	0.18
11eil51	g_2	0.8	139	2114	1520.67	2114	1.93	2114	40.32	2114	0.20
14st70	g_1	0.4	126	33	9666.29	33	4.43	33	16.65	33	0.14
14st70	g_2	0.4	126	1672	4396.77	1672	4.35	1672	28.50	1672	0.15
14st70	g_1	0.8	252	65	18227.23	65	8.80	65	959.59	65	0.31
14st70	g_2	0.8	252	3355	30851.18	3355	7.89	3355	228.84	3355	0.33
16eil76	g_1	0.4	83	40	4987.09	40	3.88	40	86.18	40	0.19
16eil76	g_2	0.4	83	2223	4939.08	2223	4.73	2223	37.55	2223	0.20
16eil76	g_1	0.6	125	59	29565.85	59	2.40	59	64.31	59	0.31
16eil76	g_2	0.6	125	3119	21127.41	3119	6.28	3119	108.75	3119	0.32

Table 1 shows that solving to optimality the ILP formulation (2)-(9) requires significantly less computational time than solving the MIP formulation proposed in Archetti et al. (2018). Furthermore, the computational times of VNS-SOP are about one order of magnitude lower than those of MASOP, while the solution value is optimal for all the runs. We recall that in Archetti et al. (2018) the MIP formulation was solved by means of CPLEX 12.6 and the experiments were carried on a standard PC equipped with Intel Core i7 clocked at 2.80 GHz. Thus, the computational time improvement obtained can be only partially justified by the newer version of the CPLEX solver and the better PC used to perform our experiments. The significantly lower computational times suggest that solving the ILP formulation (2)-(9) and computing solutions by VNS-SOP are both themselves less computationally demanding. The ILP model (2)-(9) has fewer variables (no vertex variables) than the MIP formulation. Furthermore, the different SECs are added only when found to be violated, which can save the insertion of all SECs (especially when the lower bound is set to the CPLEX solver using the greedy initial feasible solution).

The results shown in Table 2 compare the performance of the proposed algorithms against that of MASOP for the budget ratio $\omega = 1$ on large instances with up to 1084 nodes. The large instances cannot be solved optimally using the ILP formulation within the given computational time, except four cases. For both profit types, the table shows the solution value P and the computational time T for the solution computed by MASOP. The computational results of VNS-SOP are reported with the maximal P and the average P_{avg} solution values, and also with the average computational time T. The results computed by the CPLEX solver when addressing the ILP formulation (2)-(9) are shown with the maximally achieved solution values during the optimization and with the percentage gap (or the computational time in seconds for the four optimal solutions found). The profits computed by VNS-SOP appear in bold or underlined when found to be larger or smaller, respectively, than those computed by MASOP.

Regarding the results presented in Table 2, VNS-SOP requires less com-

Table 2: Comparison on large GTSP dataset instances of the SOP with $\omega = 1$.

instance	g_1							g_2						
	MASOP		VNS-SOP			ILP		MASOP		VNS-SOP			ILP	
	P	**T**	**P**	$\mathbf{P_{avg}}$	**T**	**P**	**T/gap**	**P**	**T**	**P**	$\mathbf{P_{avg}}$	**T**	**P**	**T/gap**
16pr76	74	8.6	74	74.0	0.6	74	*1.4%*	3765	10.6	3765	3765.0	0.6	3765	26567.7
20kroA100	96	10.5	96	96.0	0.8	96	*3.1%*	4868	11.5	4868	4868.0	0.8	4868	*2.9%*
20kroB100	98	13.4	98	98.0	0.8	98	*1.0%*	4916	10.7	4916	4916.0	0.9	4916	*1.9%*
20kroC100	97	10.8	97	96.1	0.9	97	*2.1%*	4882	11.2	4882	4869.2	1.1	4882	*2.6%*
20kroD100	96	10.3	96	96.0	1.0	96	*3.1%*	4838	8.9	4838	4838.0	1.1	4838	*3.5%*
20kroE100	96	9.1	96	96.0	0.9	96	15536.1	4887	9.9	4887	4887.0	1.1	4887	18938.5
20rat99	93	7.6	93	92.9	1.2	85	*15.3%*	4721	8.1	4721	4721.0	1.0	4483	*11.7%*
20rd100	97	10.4	97	97.0	1.3	97	*2.1%*	4929	9.6	4929	4929.0	1.3	4929	*1.6%*
21eil101	97	8.8	**98**	97.8	1.2	98	*1.0%*	4953	20.1	**4993**	4957.0	1.1	4948	*2.1%*
21lin105	102	8.1	102	102.0	1.1	102	*2.0%*	5157	8.4	5157	5157.0	1.1	5101	*2.5%*
22pr107	101	8.6	101	101.0	0.6	101	*5.0%*	5109	8.3	5109	5105.4	0.7	5104	*5.1%*
25pr124	121	11.3	121	121.0	1.3	114	*7.9%*	6173	11.9	6173	6170.2	1.5	6159	*1.2%*
26bier127	125	16.0	125	125.0	1.8	125	*0.8%*	6314	16.2	6314	6314.0	1.8	6314	4967.7
26ch130	127	10.2	127	126.7	1.9	126	*2.4%*	6412	9.7	6412	6382.3	2.2	6412	*1.4%*
28pr136	134	10.2	134	134.0	2.1	134	*0.7%*	6841	12.2	6841	6831.4	1.8	6808	*0.6%*
29pr144	141	17.4	141	141.0	1.7	139	*2.9%*	7195	22.0	7195	7157.3	1.6	7137	*1.5%*
30ch150	144	9.6	**147**	146.7	2.1	134	*11.2%*	7315	12.3	**7394**	7378.2	1.9	6750	*11.6%*
30kroA150	145	11.3	145	144.7	2.2	140	*6.4%*	7361	13.8	7361	7356.6	2.3	7145	*5.4%*
30kroB150	148	15.2	148	148.0	2.5	148	*0.7%*	7445	15.1	7445	7445.0	2.6	7355	*2.4%*
31pr152	147	18.2	147	145.6	1.7	137	*10.2%*	7422	17.8	7422	7355.6	1.9	6545	*17.0%*
32u159	157	15.5	157	155.1	2.2	143	*10.5%*	7991	22.9	**8011**	7965.2	2.8	7666	*4.8%*
39rat195	189	13.0	189	188.9	5.2	164	*18.3%*	9558	11.0	9558	9546.3	4.7	8438	*16.9%*
40d198	196	36.8	196	195.2	5.7	171	*15.2%*	9934	25.7	**9938**	9926.3	6.7	8628	*15.9%*
40kroa200	198	22.8	198	198.0	3.7	189	*5.3%*	10010	24.5	10010	9976.0	3.6	9577	*5.0%*
40krob200	198	19.9	198	198.0	5.0	192	*3.6%*	9990	28.6	9990	9982.7	5.8	9869	*1.9%*
45ts225	221	34.7	221	220.8	7.3	185	*21.1%*	11187	26.2	**11225**	11158.4	7.8	9767	*15.8%*
45tsp225	219	16.7	**220**	219.1	6.1	186	*20.4%*	11103	16.4	**11124**	11063.9	7.1	9615	*17.6%*
46pr226	224	26.9	224	224.0	3.6	222	*1.4%*	11368	26.4	11368	11358.1	4.6	11222	*1.4%*
53gil262	258	27.2	258	254.2	8.3	215	*21.4%*	13050	25.9	13050	13003.6	7.8	10957	*20.4%*
53pr264	262	34.0	262	262.0	7.1	230	*14.3%*	13277	36.9	13277	13277.0	7.4	13277	*0.2%*
56a280	273	33.5	273	270.8	10.4	212	*31.6%*	13971	37.0	13971	13834.9	9.9	11996	*18.2%*
60pr299	296	31.3	296	295.0	12.0	270	*10.4%*	15005	36.8	15005	14974.4	12.1	12138	*24.5%*
64lin318	315	43.4	**316**	313.8	10.6	295	*7.5%*	16013	77.3	16013	15948.0	11.2	15170	*5.7%*
80rd400	397	76.7	**398**	394.3	28.2	342	*16.7%*	20055	48.1	**20140**	19942.2	29.4	17617	*14.4%*
84fl417	415	103.5	415	414.4	18.0	399	*4.3%*	21030	114.7	21030	20956.0	18.1	19766	*6.5%*
88pr439	437	158.0	437	432.4	33.9	415	*5.5%*	22110	132.8	22110	22032.2	33.1	21058	*5.3%*
89pcb442	440	129.5	440	438.2	38.6	361	*22.2%*	22300	95.0	22300	22116.3	36.2	19456	*14.7%*
99d493	490	120.8	490	487.1	67.3	462	*6.5%*	24827	153.1	**24840**	24708.3	66.3	23545	*5.6%*
115rat575	562	91.2	**563**	555.0	76.5	459	*25.1%*	28497	65.9	28361	28043.5	75.6	23192	*25.2%*
115u574	571	204.5	571	569.9	80.3	509	*12.6%*	28888	212.7	28888	28866.5	72.2	26118	*10.9%*
131p654	652	356.0	652	650.6	45.9	640	*2.0%*	32991	360.1	32950	32894.9	44.2	32450	*1.7%*
132d657	649	126.1	649	642.1	108.9	551	*19.1%*	32974	155.9	**33022**	32901.6	100.3	29198	*13.7%*
145u724	716	99.6	**717**	708.6	176.6	564	*28.2%*	36288	116.7	**36316**	35964.8	171.4	29195	*25.1%*
157rat783	767	279.2	760	750.4	225.7	618	*26.5%*	38953	145.5	38487	37999.8	233.1	31279	*26.3%*
201pr1002	994	304.9	994	981.8	480.4	877	*14.1%*	50453	992.7	50172	49760.9	534.7	45314	*11.6%*
212u1060	1057	873.5	1057	1056.3	679.9	950	*11.5%*	53450	798.5	53437	53391.8	641.8	48151	*11.2%*
217vm1084	1078	489.5	1070	1059.7	832.0	942	*15.0%*	54642	655.5	54363	53744.2	733.7	48955	*11.8%*

putational time than MASOP for almost all instances with up to 654 nodes. VNS-SOP does not find the best known result for two instances with g_1 profit and for six instances with g_2. For both profit types, the unachieved best solutions occur for the largest instances with 493 nodes and more, where also the computational time is the same as, or larger than, the time required by MASOP. However, VNS-SOP improved the best known solutions for seven g_1 instances and 10 instances with profit type g_2. The high VNS-

SOP computational time for the largest instances is most probably caused by the graph search used for finding the optimal selection of the cluster nodes for the particular cluster sequence that is being examined for possible improvement. In the case of a large number of clusters in a sequence, as for the largest SOP instances with $\omega = 1$, the maintenance of the graph with the shortest path from the starting cluster and the ending cluster to each vertex of the current solution requires a significant amount of time in each VNS-SOP iteration. The four ILP optimal solutions that were found show that for $\omega = 1$, the paths do not visit all the clusters in the dataset instance, and VNS-SOP finds solutions with the same optimal value.

A comparison of the results shown in Tables 1 and 2 shows that VNS-SOP robustly finds high quality solutions, and in most cases significantly faster than MASOP. For several instances, VNS-SOP does not find the best known solution, but it improves the solution of a larger number of dataset instances. Furthermore, solving the ILP formulation (2)-(9) by means of the CPLEX solver requires a fraction of the computational time needed to solve the MIP formulation in Archetti et al. (2018).

4.3. Application of the SOP to other OP variants

VNS-SOP and the ILP formulation (2)-(9) are further tested when used to solve the OPN and the DOP. Both problems can be addressed as the SOP by sampling either the neighborhood positions in the OPN or the heading angles in the DOP. In both cases, the resulting problem is an approximation of the original one, i.e., its solution space is a subset of the solution space associated with the corresponding original problem. A similar VNS-based algorithm has previously been used by the authors for solving the DOP in Pěnička et al. (2017a); however, it was not addressed as the SOP studied here, nor solved using the ILP.

Orienteering Problem with Neighborhoods

The performance of the proposed methods is tested for the OPN dataset instances derived from the Tsiligirides (1984) Set 2 (see Section 4.1-OPN dataset). The results are shown in Table 3 for various budget limit T_{max} and number of neighborhood samples $o_n \in \{4, 8, 12\}$. The different values of o_n lead to instances with 78, 154, and 228 nodes, respectively. The maximal collected profit P and the average computational times T are reported for both VNS-SOP and the solutions found by the CPLEX solver when addressing the ILP formulation. For the instances with a medium budget and a large number of samples, the optimal solutions of the ILP formulation are not found within the given computational time, and the optimization gap is reported instead.

Table 3: Computational results of the OPN solved as the SOP.

$\mathbf{T_{max}}$	$o_n = 4$				$o_n = 8$				$o_n = 12$			
	ILP		VNS-SOP		ILP		VNS-SOP		ILP		VNS-SOP	
	P	**T/gap**	**P**	**T**	**P**	**T/gap**	**P**	**T**	**P**	**T/gap**	**P**	**T**
1500	180	3.3	180	0.2	180	96.8	180	0.4	180	1710.5	180	0.7
2000	230	8.6	230	0.3	230	63.5	230	0.6	230	242.9	230	1.0
2300	230	10.9	230	0.4	240	1798.1	240	0.8	240	14971.0	240	1.2
2500	260	39.5	260	0.4	260	6462.2	260	0.8	260	*10.7%*	260	1.2
2700	280	99.6	280	0.5	290	12165.4	290	0.8	270	*16.3%*	290	1.3
3000	320	432.4	320	0.4	320	*7.0%*	340	1.0	320	*15.6%*	340	1.5
3200	360	261.3	360	0.5	370	*2.7%*	370	0.9	360	*25.0%*	370	1.4
3500	410	708.3	410	0.6	410	*9.8%*	430	1.0	390	*15.4%*	430	1.5
3800	450	35.0	450	0.4	450	4314.8	450	0.9	430	*4.7%*	450	1.6
4000	450	6.9	450	0.4	450	6.2	450	1.0	450	8.3	450	1.6
4500	450	0.2	450	0.5	450	321.7	450	1.1	450	284.8	450	1.8

VNS-SOP finds the same optimal solutions in all cases where the optimal solution of the ILP formulation is found. In other cases, VNS-SOP achieves either the same results as, or better results than, the solutions found when addressing the ILP. The maximal computational time of VNS-SOP is 1.8 s, while the optimal solution of the ILP formulation is found within maximally 14 971.0 s. The selected numbers of samples o_n demonstrate how the solution quality improves when the OPN is better approximated using more samples.

Dubins Orienteering Problem

The DOP solved as a sampling-based SOP is evaluated on instances also derived from the Tsiligirides (1984) Set 2 (see Section 4.1-DOP dataset). For the DOP, the heading angles of the airplane-like Dubins vehicle are sampled into $o_h \in \{4, 8, 12\}$ values, creating datasets with 84, 168, and 252 nodes, respectively. Such SOP instances have starting and ending clusters with o_h nodes and the heading angles in these clusters have to be found during optimization. Table 4 reports on the achieved results of the solution methods for various budget limit $\mathrm{T_{max}}$ and number of heading samples o_h.

Table 4: Computational results of the DOP solved as the SOP.

	$o_h = 4$				$o_h = 8$				$o_h = 12$			
$\mathbf{T_{max}}$	ILP		VNS-SOP		ILP		VNS-SOP		ILP		VNS-SOP	
	P	**T**	**P**	**T**	**P**	**T**	**P**	**T**	**P**	**T**	**P**	**T**
1500	115	2.97	115	0.21	120	8.18	120	0.39	120	20.62	120	0.62
2000	175	1.62	175	0.26	190	6.14	190	0.50	190	14.34	190	0.88
2300	190	1.96	190	0.31	200	9.14	200	0.61	200	17.76	200	1.00
2500	205	2.76	205	0.45	220	8.45	220	0.70	220	13.23	220	1.12
2700	215	3.14	215	0.39	230	5.67	230	0.70	230	17.25	230	1.18
3000	240	9.05	240	0.48	255	12.15	255	0.87	255	33.35	255	1.48
3200	265	4.11	265	0.50	280	12.06	280	0.90	290	51.76	290	1.31
3500	295	5.55	295	0.41	315	18.57	315	0.74	315	30.02	310	1.26
3800	330	9.17	330	0.47	345	23.15	345	1.10	345	46.18	345	1.91
4000	360	3.75	360	0.54	375	14.71	375	1.08	375	50.30	375	1.74
4500	415	3.54	415	0.59	430	14.29	430	0.93	440	21.31	440	1.85

According to the presented results, the optimal solution is found by the CPLEX solver when addressing the ILP formulation for all the tested instances with the maximally required computational time 51.76 s. VNS-SOP finds solutions that are optimal in all instances with the exception of one case with $\mathrm{T_{max}} = 3500, o_h = 12$. The maximal computational time of VNS-SOP is 1.91 s. Similarly to the OPN, the DOP is better approximated using more samples o_h, as can be seen for $o_h = 4$ and $o_h = 12$. We can also observe that the optimal solution when addressing the ILP formulation is found much faster in the case of the asymmetric DOP than for the OPN. The branch-and-cut algorithm used by the CPLEX solver thus performs better for the DOP with a large difference in the lengths of the Dubins maneuvers between samples of the heading angle connecting the same clusters.

The results in both Table 3 and Table 4 show that the SOP can be successfully used for solving the sampled OPN and DOP. VNS-SOP can find high-quality solutions within 1.91 seconds for problems with up to 252 nodes. Furthermore, the solution of the ILP formulation, found with very low computational time in the case of the DOP, indicates that VNS-SOP can achieve the optimal solution of the sampled DOP and OPN for almost all evaluated instances.

5. CONCLUSIONS

In this paper, we introduce a Variable Neighborhood Search (VNS) metaheuristic and a novel Integer Linear Programming (ILP) formulation for the Set Orienteering Problem (SOP). The SOP is a generalization of the OP where customers are grouped in clusters, and the objective is to find a tour with a predefined starting cluster and ending cluster, a restricted budget, and such that the tour maximizes the profit collected from clusters with at least one visited customer. The VNS algorithm for the SOP (VNS-SOP) robustly provides high-quality solutions and improves the solution of several benchmark instances. The computational times of finding the SOP solution using both the novel ILP formulation and VNS-SOP are significantly lower than those of the existing approaches, especially in low to medium-size test instances. Furthermore, we show other variants of the Orienteering Problem that can be addressed as the SOP using a sampling-based approach. The Orienteering Problem with Neighborhoods, with profit collection within the neighborhood radius of each customer, and the Dubins Orienteering Problem for an airplane-like vehicle constrained by the minimum turning radius, can both be addressed as the studied SOP. The implementation of VNS-SOP and of the ILP formulation in the CPLEX solver are published as open-source software together with the dataset instances that have been used.

ACKNOWLEDGMENT

The work has been supported by the Czech Science Foundation under project No. 16-24206S and project No. 17-16900Y, and by OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics". The support of CTU in Prague grant No. SGS17/187/13 is gratefully acknowledged. Finally, we thank the anonymous reviewers whose comments and suggestions helped to significantly improve this manuscript.

References

Angelelli, E., Archetti, C., & Vindigni, M. (2014). The clustered orienteering problem. *European Journal of Operational Research*, *238*, 404–414.

Archetti, C., Carrabs, F., & Cerulli, R. (2018). The set orienteering problem. *European Journal of Operational Research*, *267*, 264–272.

Dubins, L. E. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, *79*, 497–516.

Faigl, J., Pěnička, R., & Best, G. (2016). Self-organizing map-based solution for the orienteering problem with neighborhoods. In *IEEE International Conference on Systems, Man, and Cybernetics* (pp. 1315–1321).

Feillet, D., Dejax, P., & Gendreau, M. (2005). Traveling salesman problems with profits. *Transportation Science*, *39*, 188–205.

Fischetti, M., Gonzlez, J. J. S., & Toth, P. (1997). A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, *45*, 378–394.

Gunawan, A., Lau, H. C., & Vansteenwegen, P. (2016). Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, *255*, 315–332.

Laporte, G., Asef-Vaziri, A., & Sriskandarajah, C. (1996). Some applications of the generalized travelling salesman problem. *The Journal of the Operational Research Society*, *47*, 1461–1467.

Laporte, G., & Nobert, Y. (1983). Generalized travelling salesman problem through n sets of nodes: An integer programming approach. *INFOR: Information Systems and Operational Research*, *21*, 61–75.

Li, J., Andrew, L., Foh, C., Zukerman, M., & Chen, H.-H. (2009). Connectivity, coverage and placement in wireless sensor networks. *Sensors*, *9*, 7664–7693.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, *24*, 1097–1100.

Pěnička, R., Faigl, J., Váňa, P., & Saska, M. (2017a). Dubins orienteering problem. *IEEE Robotics and Automation Letters*, *2*, 1210–1217.

Pěnička, R., Faigl, J., Váňa, P., & Saska, M. (2017b). Dubins orienteering problem with neighborhoods. In *International Conference on Unmanned Aircraft Systems* (pp. 1555–1562).

Sevkli, Z., & Sevilgen, F. E. (2006). Variable neighborhood search for the orienteering problem. In *21th International Symposium on Computer and Information Sciences* (pp. 134–143).

Tsiligirides, T. (1984). Heuristic methods applied to orienteering. *The Journal of the Operational Research Society*, *35*, 797–809.

Vansteenwegen, P., Souffriau, W., & Oudheusden, D. V. (2011). The orienteering problem: A survey. *European Journal of Operational Research*, *209*, 1–10.

Yu, J., Schwager, M., & Rus, D. (2016). Correlated orienteering problem and its application to persistent monitoring tasks. *IEEE Transactions on Robotics*, *32*, 1106–1118.