LII Simpósio Brasileiro de Pesquisa Operacional
João Pessoa - PB, 3 a 5 de novembro de 2020

Multi-depot multiple TSP with pickup and delivery: an VNS approach

Brenner H. O. Rios
UNICAMP
Institute of Computing
brenner@students.ic.unicamp.br

Eduardo C. Xavier
UNICAMP
Institute of Computing
ecx@ic.unicamp.br

Flávio K. Miyazawa
UNICAMP
Institute of Computing
fkm@ic.unicamp.br

Pedro Amorim
University of Porto
Faculty of Engineering
amorim.pedro@fe.up.pt

ABSTRACT

We present a natural variation of the multi-depot multiple Travelling Salesman Problem (MmTSP) which considers pickup and delivery. Hence the notation of our problem: MmTSPPD. This problem must meet requests for transportation between customers and suppliers satisfying precedence. The objective is to find a solution with a minimum total travelling distance. A mixed integer programming model is presented for the MmTSPPD. To solve the MmTSPPD we propose a variable neighborhood search (VNS) that uses the variable neighborhood descent (VND) as local search procedure. We evaluate the performance of this heuristic on a data set adapted from TSPLIB instances. The results show that the VNS proposed is efficient and effective to solve MmTSPPD.

KEYWORDS. Vehicle routing. Muti-depot TSP. Pickup and delivery. VNS. MmTSPPD.

Paper topics. Combinatorial optimization, Transportation, Vehicle routing.

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings

1. Introduction

The diversity of applications has motivated the study of several variants of TSP. One of its more challenging variants is the multi-depot multiple TSP problem (MmTSP), because the number of possible solutions grows exponentially with the size of the instance, specifically with the number of depots. The MmTSP is a generalization of the TSP, where every traveler must visit exactly a single city. When it is considered a single depot the problem is called single depot mTSP. If several depots are considered, the problem is called MmTSP. There are two variants of the MmTSP in relation to the travelers' destination: fixed destination and non-fixed destination. In the first, travelers are required to return to their original depot, and in the second variant, no. Bektas [2006] presents an mTSP review and explores connections with VRPs.

The TSP variant, which will be discussed, is the version of MmTSP with pickup and delivery (MmTSPPD). The MmTSPPD is closely related to the steiner multi cycle problem (SMCP), recently introduced by Pereira et al. [2018]. The SMCP arises in the scenario where a company has to periodically exchange goods between two different locations, and different companies can collaborate to create a route that visits all its pairs of locations sharing the total cost of the route [Pereira et al., 2018]. The MmTSPPD can be seen as a version of the SMCP with depots. The MmTSPPD is also closely related to the problem proposed by Dridi et al. [2019]. In this work, the authors introduced the multi-depots pick-up and delivery problem with time windows and multi-vehicles. The principle of MmTSPPD is to design an optimal route for a fleet of vehicles, each one located in different depots. The optimal route allows serving a set of pickup and delivery points geographically dispersed. The number of vehicles is equal to the number of depots. Each vehicle must start and end the tour in its assigned depot. Vehicles must visit once and only once each node. In Figure 1 shows the tour of three vehicles belonging each one to a different depot.

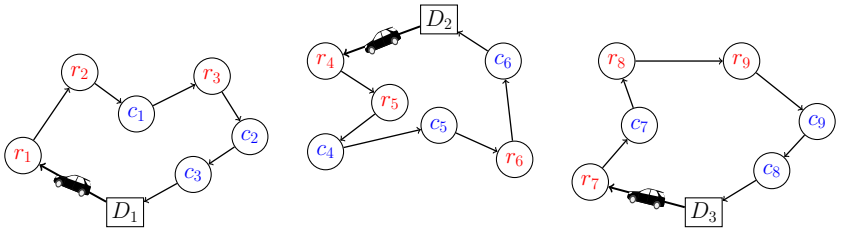


Figure 1: Example of a solution for the MmTSPPD with three vehicles. Each vertex r_i represents a pickup point while c_i represents its corresponding delivery point.

We propose an VNS that uses the VND heuristic as local search to deal with the MmTSPPD. We denote the proposed algorithm as Generalized VNS (GVNS). The GNVS is based on the VNS heuristics presented by Carrabs et al. [2007] for the Pickup and Delivery Traveling Salesman Problem with LIFO Loading. To evaluate its performance we compare it with the classical VNS introduced by Mladenović e Hansen [1997]. The TSP is a special case of VRP. There are several heuristic approaches to solving VRPs. State of the art solutions include: particle swarm optimization approach [Dridi et al., 2019], VNS [Polacek et al., 2004], adaptive large neighbourhood search [Laporte et al., 2010], ILS [Subramanian et al., 2010], genetic approach [Mendoza e Villegas, 2012], tabu search [Gendreau et al., 1996] and hybrid heuristic with exact methods [Rios et al., 2017]. The remainder of this paper is organized as follows. Section 2 presents the description and mathematical formulation of the MmTSPPD. Section 4 introduces the VNS proposed and presents the GVNS heuristic proposed for the problem. Section 5 and 6 presents results and conclusions of the work, respectively.

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings

2. MmTSPPD

In this study, MmTSPPD is defined as follows. Let $G = (V, E)$ be a complete undirected graph, where $V = \{v_1, \ldots, v_n\}$ is the vertex set and $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$ is the edge set. With each edge (v_i, v_j), it is associated a non-negative cost or distance d_{ij}. It is assumed that d_{ij} satisfies the triangle inequality. A subset of vertices $D = \{v_1, \ldots, v_m\}$ represents the depots, and the remaining vertices $V' = \{v_{m+1}, \ldots, v_n\}$ corresponds to pickup and delivery points. Let $w = |V'|/2$, then w vertices are pickup points and w vertices are delivery points. Each pickup point v_i is associated with a unique delivery point v_{i+w}, and vice versa, for $m + 1 \leq i \leq m + w$. There are m identical vehicles of unlimited capacity such that each one is located in a single depot. Each vehicle leaves its depot, serves a subset of pickup and delivery vertices and returns to its depot, forming a cycle. The problem consists in determining a set of m vehicle cycles of minimal total cost considering the following conditions: 1) each cycle starts and ends at the corresponding vehicles depot; 2) each $v \in V'$ is visited exactly once by one vehicle and 3) each pair of pickup and delivery points, e.g $\{v_i, v_{i+w}\}$ for $m + 1 \leq i \leq m + w$, must belong to the same cycle.

We adapt the mathematical formulation proposed by Kara e Bektas [2006] for the deterministic static version of MmTSPPD. In this formulation we assume G is a complete symmetric directed graph. The parameters and variables of the formulation are defined in Table 1.

D	Set of depots, $\{v_1, ..., v_m\}$.
V'	Set of nodes (pickup and delivery), $\{v_{m+1}, ..., v_n\}$
H^+	Set of pickup nodes, $\|H^+\| = w$.
u_i^k	A positive integer variable that indicates the order location i is visited by vehicle k, and $u_i^k = 0$ if i is not visited by k, $i \in V'$, $k \in D$.
d_{ij}	Euclidean distance between node i and j.
x_{ij}^k	If the vehicle originated from depot k travel along arc (i, j), then $x_{ij}^k = 1$. Otherwise $x_{ij}^k = 0$.

Table 1: Parameters and decision variables for the MmTSPPD

$$\text{minimize} \sum_{k \in D} \sum_{j \in V'} (d_{kj} x_{kj}^k + d_{jk} x_{jk}^k) + \sum_{k \in D} \sum_{i \in V'} \sum_{j \in V'} d_{ij} x_{ij}^k \tag{1}$$

$$s.t. \sum_{j \in V'} x_{kj}^k = 1, \quad k \in D, \tag{2}$$

$$\sum_{k \in D} x_{kj}^k + \sum_{k \in D} \sum_{i \in V'} x_{ij}^k = 1, \quad \forall j \in V', \tag{3}$$

$$x_{kj}^k + \sum_{i \in V'} x_{ij}^k = x_{jk}^k + \sum_{i \in V'} x_{ji}^k, \quad \forall k \in D, j \in V', \tag{4}$$

$$\sum_{j \in V'} x_{kj}^k = \sum_{j \in V'} x_{jk}^k, \quad k \in D, \tag{5}$$

$$u_i^k \leq n(\sum_{j \in V'} x_{ij}^k + x_{ik}^k), \quad i \in V', \quad k \in D, \tag{6}$$

$$x_{ki}^k \leq u_i^k, \;\; i \in V', \;\; k \in D, \tag{7}$$

$$u_i^k + 1 \leq u_j^k + (1 - x_{ij}^k)n, \;\; i, j \in V', \;\; k \in D, \tag{8}$$

$$u_i^k + 1 \leq u_{i+w}^k + (1 - \sum_{j \in V'} x_{ij}^k)n, \;\; i \in H^+, \;\; k \in D, \tag{9}$$

$$x_{ij}^k \in \{0, 1\}, \;\; i, j \in V, \tag{10}$$

$$u_i^k \in \mathcal{Z}^+, \;\; i \in V, \;\; k \in D, \tag{11}$$

In this formulation, constraint (2) ensures that exactly one vehicle depart from each depot $k \in D$. Constraint (3) ensures that each node is visited exactly once. Route continuity for pickup and delivery nodes and depot nodes are represented by constraints (4) and (5), respectively. Constraints (6) assures that the order of client i is 0 if it is not in the route of vehicle k. Constraints (7) impose that if i is the first vertex visited in route k, than its order in this route is at least 1. Constraint (8) is a subtour elimination constraint, since if j is visited after i in route k, then the visit order of j must be larger than the one of i in this route. Constraint (9), ensures that each pick-up node (i) must be visited before the corresponding delivery node $(i + w)$. Finally we have the integrality constraints (10) and (11) of the variables in the model.

3. Basic operators

We describe six basic operators, *Shift(1,0), Swap(1,1), 2-opt, Mix-Shift, Double-Swap* and *Depot Exchange*. These six operators rely on simple moves which preserve the feasibility.

We denote with N to the set of operators used by VND. All operators are exhaustively executed. We divide these operators into three groups: *inter-tour, intra-tour* and *inter&intra-tour operators*. The *inter-tour* operators are: *Shift(1,0)* and *Swap(1,1)*. The *intra-tour* neighborhood operator is: 2-opt. Finally, Mix-shift(1,0) is the only operator that is *inter&intra-tour* operator. In the case of *inter-route* operators, to reduce the computational cost, each vertex removed of a route it can only be inserted before or after of its p closest neighbors of the other routes.

Shift(1,0) – $N^{(1)}$ – A vertex v is removed from a route t_1 to be inserted into route t_2, then the pair vertex of v is moved to the best position in route t_2 that does not break the constraints of the problem. This operator runs in $O(n)$. Figure 2 presents an example of the this operator.

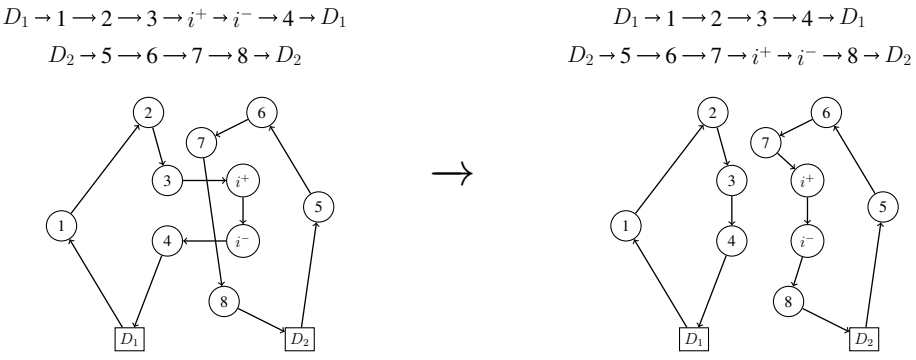


Figure 2: An example of Shift(1,1) operator

Swap(1,1) – $N^{(2)}$ – Exchange between a vertex v from a route t_1 and vertex u from a route t_2. The pair of vertex of v is moved to the best position of route t_2 that does not break the constraints of the problem. Analogously, the pair of vertex u is moved. This operator runs in $O(n)$. Figure 3 presents an example of the this operator.

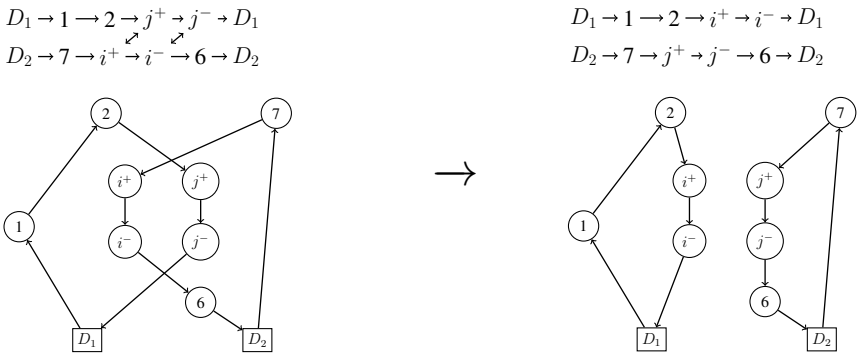


Figure 3: An example of Swap(1,1) operator

In case of improvement of the current solution, we will initiate an intensification process on each route. The objective is to decrease the cost of each route to reduce the value of the current solution. Therefore, the following neighborhoods are explored.

2-opt – $N^{(3)}$ – Two nonadjacent arcs are removed and other two are added to form a new route. We only consider movements that do not break the constraints of the problem. This operator runs in $O(n^3)$, see Carrabs et al. [2007].

Mix-Shift(1,0) – $N^{(6)}$ – This operator is similar to the Shift(1,0) operator with the difference that here is allowed movement within its own route. This operator runs in $O(n)$.

A set P of two perturbation mechanisms were adopted. Every time the *Shake* function is called one of the following operators is randomly selected, see Section 4.

Double-Swap – $P^{(1)}$ – Two Swap(1,1) operators are performed in sequence randomly. This operator runs in $O(n)$.

Depot Exchange – $P^{(2)}$ – The depot exchange operator select two depots at random, then exchange their positions. This operator runs in $O(1)$. Figure 4 presents an example of the this operator.

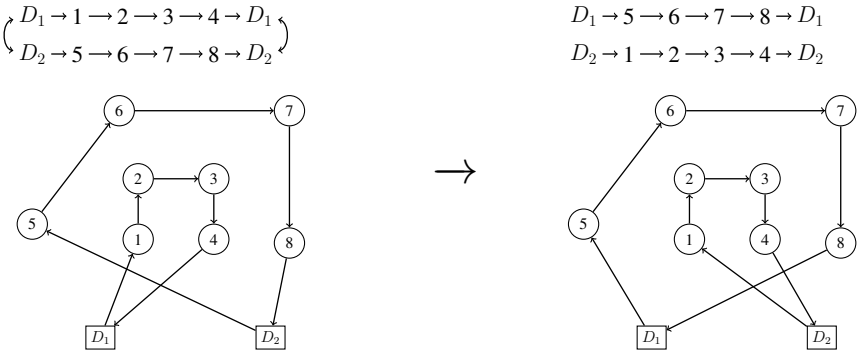


Figure 4: An example of Depots Exchange operator

4. VNS

Variable neighborhood search (VNS) Mladenović e Hansen [1997] systematically changes neighborhood structures during the search for an optimal (or near-optimal) solution Hansen et al. [2016]. The idea is to apply a perturbation to current neighborhood operator at a local minimum Carrabs et al. [2007]. VNS generally consists of three main steps: the shaking step, the local search step, and the neighborhood change step Hansen et al. [2016]. The local search step aims at finding a local optimal solution in a neighborhood. The shaking step aims at escaping from a local optimal solution as well as generating a new initial solution for the local search step Pongchairerks [2016]. Several enhancements of the original method have since been proposed in Hansen e Mladenović [2005]. Our GVNS is summarized in Algorithm 1, where s^* corresponds to the best solution and $k_{\max}$ is the maximum number of neighborhoods.

Algorithm 1: Gereralized VNS

$s \leftarrow$ FindInitialSolution() ;
$s_1 \leftarrow$ LocalSearch(s);
$s^* \leftarrow s_1$; $k \leftarrow 1$;
while $k \leq k_{\max}$ **do**
  $s_2 \leftarrow$ Shake(s_1,k);
  $s_3 \leftarrow$ LocalSearch(s_2);
  $s_1, k \leftarrow$ AcceptanceCriterion(s^*, s_3)
$s^* \leftarrow$ LocalSearch(s^*);
return s^*

4.1. Initial solution generation

The method employed for building a feasible initial solution is based in the work of Kuo and Wang Kuo e Wang [2012], so it is generated by following two steps. The first step is called *nodes assignment*. Each pickup and delivery point is assignment to one of the depots. After all points have been assigned to depots, the second step, called *nodes sequencing*, decides the service sequence of the pickup and delivery points. The details of these two steps are introduced in the following paragraphs.

4.1.1. Nodes assignment

This step assigns nodes to the depot which is conveniently close to them, so we assign nodes to the depot which is nearest them. In the same way as in Kuo e Wang [2012], in order to make the initial solution more flexible, the assignments of pickup an delivery points to depots are based on a probability. Suppose that $d(D_a, r_i, c_i)$ indicates the sum of the distances between pickup point r_i and depot D_a ($d(D_a, r_i)$), and the distance between the delivery point c_i and depot D_a ($d(D_a, c_i)$). Let $\overline{d(D, r_i, c_i)}$ the average distance between the pair of pickup and delivery $\{r_i, c_i\}$ and all depots. The probability of the set $\{r_i, c_i\}$ being assigned to depot D_a, is calculated by Eq. (12).

$$P(D_a, r_i, c_i) = \frac{\max\left\{\overline{d(D, r_i, c_i)} - d(D_a, r_i, c_i), 0\right\}}{\sum_{a=1}^{|D|} \max\left\{\overline{d(D, r_i, c_i)} - d(D_a, r_i, c_i), 0\right\}} \tag{12}$$

Based on the probability calculated using Eq. (12), a pickup and delivery pair may still be assigned to depot which is not the nearest to it.

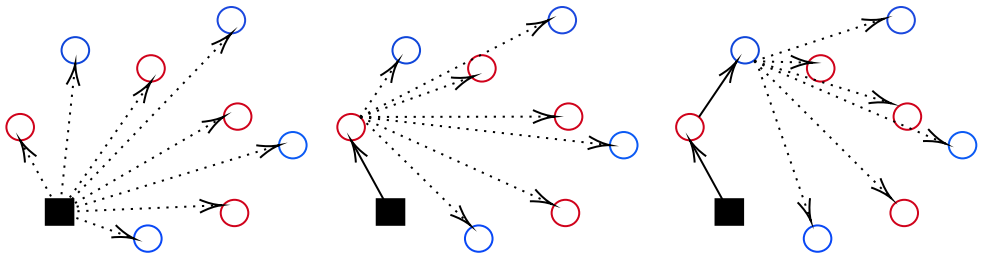


Figure 5: Example of nodes sequencing in a route.

4.1.2. Nodes sequencing

Once we have assigned the pickup and delivery points for each depot, we sequence the nodes to create cycles. Let D_i be an arbitrary depot, v_i a pickup or delivery point associated with D_i and t_i a tour containing D_i. The procedure begins by looking for a node v_i nearest to the last node of t_i, which is currently D_i, such that no MmTSPPD constraint is broken if v_i is appended to t_i. Then v_i is appended to t_i and the procedure is repeated until all the pickup and delivery points are inserted into t_i. The Figure 5 shows an example of sequencing of a depot and four pairs of pickup and delivery (red nodes are pickups points, blue nodes are delivery points and black square is a depot).

4.2. Local Search

The local search is based on the VND heuristics introduced by Mladenović and Hansen Mladenović e Hansen [1997]. In the variable neighborhood descent method a change of neighborhoods is performed in a deterministic way. The proposed VND is presented in Algorithm 2

Algorithm 2: VND

```
Let r be the number of neighborhoods structures;
k := 1; current neighborhood;
while k ≤ r do
    Find the best neighbor s' of s ∈ N^k;
    if f(s') < f(s) then
        s := s';
        k := 1;
        intensification in the modified routes;
        s' := 2 − opt(s);
        if f(s' ≤ f(s) then
            s := s'
    else k := k + 1;
```

Given the versatility to search solutions of the *Mix-Shift(1,0)* operator, and based on neighborhood structure proposed in Gendreau et al. [1996], we present the Random Mix-Shift heuristic (Algorithm 3). In this heuristic a randomly selected vertex v is removed, and inserted immediately before or after one of its p closest neighbors. The vertex u (pair of v, e.g. $\{v, u\}$) is randomly inserted into the same route of v, without breaking the constraints of the problem. This heuristic is used as a refinement mechanism for the initial solution. To avoid unnecessary calls to local search

heuristics, we chose to use local search only if s' is *promising*. A solution s' is *promising* if it has the potential to become the new best solution, specifically, if it is $\alpha\%$ worse than solution s.

Algorithm 4. shows the pseudocode of our GVNS heuristic.

Algorithm 3: Random Mix-Shift

```
for k := 1 ..., MaxIterShift do
    v := SelectRandomVertex(s) ;
    s' := Mix-Shift(s, v) ;
    if f(s') < f(s*) then
        s := s'; s* := s'; k := 1 ;
    else
        if f(s') < α f(s*) then
            s := s' ;
        else
            s := s*;
```

Algorithm 4: GVNS heuristic

```
s := FindInitialSolution(seed);
s_i := RandomMixShift(s) ;
diversification := 0 ;
for iter := 1 to MaxIter do
    perturbation phase ;
    s'_i := Shake(s_i) ;
    if f(s_j) < f(s*) then
        s* := s_j ;
        iter := 0;
    s_k := VND(s_j);
    if f(s_k) < f(s*) then
        s* := s_k;
        iter := 0;
    else
        s_k :=AcceptanceCriteria(s*, s_k);
    s_i := s_k;
    diversification phase ;
    if iter = MaxIter and diversification = 0 then
        iter := 0;
        diversification := 1;
        s'_i := Shake(s_i);
```

4.3. Shaking and Acceptance criterion

The shaking phase is implemented with *Double-Swap* and *Depot Exchange* operators. The last part of the heuristic concerns the acceptance criterion, it is based on the work of Carrabs et al. [2007]. Here we have to decide if the solution produced by VND will be accepted as a staring solution for the next iteration. Logically, if the solution found is better than the best solution obtained so far, we accept the new solution. Otherwise we accept the new solution only if

$f(\text{CurrentSol}) < \alpha f(\text{BestSol})$ (f is the cost function). The idea in this formula is not to move too far from the optimal local solution. If the above condition is not satisfied the heuristic restarts from the best solution.

When we change the VND heuristic for 2-opt and remove *RandomMixShift* heuristic we get the classic VNS heuristic.

5. Computational experiments

We conducted experiments using a data set derived from five TSPLIB instances (*ulysses16, bayg29, eil51* and *st70*). For each of these instances, n vertices in the interval of [2, 10] were randomly selected to be depots. A random matching was performed among the other vertices to create pickup and delivery pairs. We generate 20 test instances.

The algorithms described above were coded in C++ and all experiments were run on a Linux operating system with 3 GB memory and Intel Core i5 2.54x4 Ghz processor. Computational times reported here are in CPU seconds on this machine. To evaluate our GVNS heuristic we compare it a classic VNS algorithm. Ten independent runs of the algorithms were performed for each test case. The number of iterations (*MaxIter*) was 100. The parameters α, $MaxIterShift$ and p were fixed to 1.05, 100 and 5, respectively. They were calibrated empirically after preliminary tests with different values.

Instance			**VNS**			**GVNS**		
Name	$\|V\|$	$\|D\|$	**Best**	**Avg.**	**Time**	**Best**	**Avg.**	**Time**
ulysses16a	16	2	**85,89**	86,22	0,08	**85,89**	86,09	2,81
ulysses16b	16	2	**85,35**	85,67	0,07	**85,35**	85,35	2,74
ulysses16c	16	2	**81,14**	81,14	0,07	**81,14**	81,14	2,97
ulysses16d	16	4	96,18	97,23	0,07	**84,11**	89,52	2,18
ulysses16e	16	4	95,32	105,02	0,07	**94,87**	98,89	2,36
bayg29a	29	3	12184,60	12283,70	4,93	**11968,50**	12098,80	5,25
bayg29b	29	9	19518,10	20325,10	2,88	**18445,9**	19218,5	3,71
bayg29c	29	9	20834,40	20835,70	2,90	**18545,8**	19923,6	3,90
bayg29d	29	3	**11583,80**	11619,90	5,15	**11583,80**	11603,00	5,62
bayg29e	29	5	13085,80	13096,10	4,81	**12881,40**	13025,80	5,04
eil51a	51	3	645,11	667,67	11,16	**640,71**	669,09	12,70
eil51b	51	7	785,84	823,30	8,99	**749,75**	802,89	9,61
eil51c	51	9	733,06	750,83	8,88	**713,41**	736,02	8,81
eil51d	51	7	**727,17**	755,55	9,71	739,64	764,91	11,01
eil51e	51	5	648,28	677,46	9,37	**646,661**	670,03	9,84
st70a	70	4	1219,14	1240,61	15,85	**1162,88**	1216,58	24,22
st70b	70	6	1375,93	1403,10	14,79	**1300,42**	1381,94	16,93
st70c	70	8	1454,73	1494,23	14,16	**1441,92**	1466,14	19,41
st70d	70	8	1563,47	1593,13	13,67	**1551,45**	1579,46	17,28
st70e	70	6	1289,08	1308,94	15,00	**1233,41**	1279,37	20,87

Table 2: Results of the GVNS and VNS to solve MmTSPPD.

Table 2 shows the results of the GVNS and the VNS. The best solutions are in boldface. The columns related with the instances show: the instance name, *Name*, the number of vertices in the graph, $|V|$, and the number of depots $|D|$. Columns *Avg.* and *Best* show the average and the best solution costs found by the algorithms in their ten independent executions, respectively.

	VNS			VNS	
Instance	**Best (%)**	**Avg. (%)**	**Instance**	**Best (%)**	**Avg. (%)**
ulysses16a	0,00	-0,14	eil51a	-0,69	0,21
ulysses16b	0,00	-0,38	eil51b	-4,81	-2,54
ulysses16c	0,00	0,00	eil51c	-2,75	-2,01
ulysses16d	-14,35	-8,61	eil51d	1,69	1,22
ulysses16e	-0,48	-6,21	eil51e	-0,25	-1,11
bayg29a	-1,81	-1,53	st70a	-4,84	-1,98
bayg29b	-5,81	-5,76	st70b	-5,81	-1,53
bayg29c	-12,34	-4,58	st70c	-0,89	-1,92
bayg29d	0,00	-0,15	st70d	-0,77	-0,87
bayg29e	-1,59	-0,54	st70e	-4,51	-2,31

Table 3: Percentage of improvement obtained by the GVNS algorithm.

Column *Time* presents the average processing time, in second, spend by each algorithm. The GVNS presented the best results for most instances.

Table 3 shows percentage improvements of the best and average solutions obtained by the GVNS, against the VNS. Negative values mean that GVNS was better that the VNS. The formula $(GVNS - VNS)/GVNS$ is used to generate the values presented in Table 3. Except for *eil51d*, the results show that the GVNS was superior to the VNS for all instances.

6. Conclusion

This article described a practical TSP involving multiple depots, and pickup and delivery (MmTSPPD). In order to dealt with MmTSPPD, an algorithm based on the VNS metaheuristic, which uses a VND heuristic as local search procedure was proposed. We use four local search operators, *Shift(1,0), Swap(1,1), 2-opt, and Mix-Shift*. Also, we use two perturbation mechanisms, *Double-Swap and Depot-exchange*. We propose a heuristic based on *Mix-shift* operator to refine the initial solution of the GVNS. The GVNS was compare with the classic VNS. We report the results for 20 instances. Except for *eil51d*, the results show that the GVNS was superior to the VNS for all instances. Our approach can be used as benchmark for future research in this area. The MmTSPPD can be further generalized to handle more practical constraints, e.g., limited capacity vehicles, time windows and stochastic demands.

Acknowledgment

This work is financed by the CNPq (Proc. 314366/2018-0, 425340/2016-3, 304856/2017-7), FAPESP (Proc. 2015/11937-9, 2016/01860-1, 2018/08879-5, 2019/26159-2), ERDF – European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia within project POCI-01-0145-FEDER-031908.

References

Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219.

Carrabs, F., Cordeau, J.-F., e Laporte, G. (2007). Variable neighborhood search for the pickup and delivery traveling salesman problem with LIFO loading. *INFORMS Journal on Computing*, 19 (4):618–632.

Dridi, I. H., Alaïa, E. B., Borne, P., e Bouchriha, H. (2019). Optimisation of the multi-depots pick-up and delivery problems with time windows and multi-vehicles using PSO algorithm. *International Journal of Production Research*, p. 1–14.

Gendreau, M., Laporte, G., e Séguin, R. (1996). A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. *Operations Research*, 44(3):469–477.

Hansen, P. e Mladenović, N. (2005). Variable neighborhood search. In *Search Methodologies*, p. 211–238. Springer US.

Hansen, P., Mladenović, N., Todosijević, R., e Hanafi, S. (2016). Variable neighborhood search: basics and variants. *EURO Journal on Computational Optimization*, 5(3):423–454.

Kara, I. e Bektas, T. (2006). Integer linear programming formulations of multiple salesman problems and its variations. *European Journal of Operational Research*, 174(3):1449–1458.

Kuo, Y. e Wang, C.-C. (2012). A variable neighborhood search for the multi-depot vehicle routing problem with loading cost. *Expert Systems with Applications*, 39(8):6949–6954.

Laporte, G., Musmanno, R., e Vocaturo, F. (2010). An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Science*, 44 (1):125–135.

Mendoza, J. E. e Villegas, J. G. (2012). A multi-space sampling heuristic for the vehicle routing problem with stochastic demands. *Optimization Letters*, 7(7):1503–1516.

Mladenović, N. e Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

Pereira, V. N. G., San Felice, M. C., Hokama, P. H. D. B., e Xavier, E. C. (2018). The steiner multi cycle problem with applications to a collaborative truckload problem.

Polacek, M., Hartl, R. F., Doerner, K., e Reimann, M. (2004). A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of Heuristics*, 10(6): 613–627.

Pongchairerks, P. (2016). Forward VNS, reverse VNS, and multi-VNS algorithms for job-shop scheduling problem. *Modelling and Simulation in Engineering*, 2016:1–15.

Rios, B. H. O., Goldbarg, E. F. G., e Quesquen, G. Y. O. (2017). A hybrid metaheuristic using a corrected formulation for the traveling car renter salesman problem. In *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE.

Subramanian, A., Drummond, L., Bentes, C., Ochi, L., e Farias, R. (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 37(11):1899–1911.

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings