Yugoslav Journal of Operations Research
23 (2013), Number 1, 19-30
DOI:10.2298/YJOR120530015M

AN EFFICIENT GENERAL VARIABLE NEIGHBORHOOD SEARCH FOR LARGE TRAVELLING SALESMAN PROBLEM WITH TIME WINDOWS

Nenad MLADENOVIĆ
School of Mathematics, Brunel University-West London, UK,
Nenad.Mladenovic@brunel.ac.uk
Raca TODOSIJEVIĆ
Mathematical Institute, Serbian Academy of Science and Arts, Serbia,
racatodosijevic@gmail.com
Dragan UROŠEVIĆ
Mathematical Institute, Serbian Academy of Science and Arts, Serbia,
Dragan.Urosevic@mi.sanu.rs

Received: May 2012 / Accepted: August 2012

Abstract: General Variable Neighborhood Search (GVNS) is shown to be a powerful and robust methodology for solving travelling salesman and vehicle routing problems. However, its efficient implementation may play a significant role in solving large size instances. In this paper we suggest new GVNS heuristic for solving Travelling salesman problem with time windows. It uses different set of neighborhoods, new feasibility checking procedure and a more efficient data structure than the recent GVNS method that can be considered as a state-of-the-art heuristic. As a result, our GVNS is much faster and more effective than the previous GVNS. It is able to improve 14 out of 25 best known solutions for large test instances from the literature.

Keywords: Travelling Salesman Problem, Time windows, Variable Neighborhood Search.

MSC: 90C59, 90B06.

1. INTRODUCTION

The Travelling Salesman Problem with Time Windows (TSPTW) is a variant of the well-known Travelling Salesman Problem (TSP). Suppose that a depot, a set of customers, service time (i.e., the time that must be spent at the customer), and a time window (i.e. its ready time and due date) are given. The TSPTW problem consists of finding a minimum cost tour starting and ending at a given depot, after each customer is visited only once before its due date. The travelling salesman is allowed to arrive to the customer before its ready time, but has to wait. Obviously, there are tours which do not allow the travelling salesman to respect due dates of all customers. All such tours, we call infeasible, and the others are feasible tours (solutions). The cost of a tour is the total distance travelled.

Graph *G = (V, A)* is given, where *V = {1, 2,…, n}.* Let 0 denotes a depot and let $A = \{(i, j) : i, j \in V \cup \{0\}\}$ be the set of arcs between customers. The travelling cost from *i* to *j* is represented by c_{ij} , which includes both the service time of a customer *i* and the time needed to travel from *i* to *j*. Each customer *i* has an associated time window $[a_i, b_i]$ where a_i and b_i represent the ready time and the due date, respectively. So, the TSPTW can be stated, mathematically, as a problem of finding a Hamiltonian tour that starts and ends at the depot, satisfying all time windows constraints and minimizing the total distance traveled.

TSPTW is *NP*-hard problem since it is a special case of the well-known Travelling Salesman Problem, which is *NP*-hard. So, there is need for a heuristic able to solve efficiently realistic instances in the reasonable amount of time. In that direction, some steps have been already made. Carlton and Barnes [3] use a tabu-search heuristic with a static penalty function, using infeasible solutions in the search. Gendreau et al. [8] propose an insertion heuristic based on GENIUS heuristic [7], which gradually builds the route in construction phase and improves it in a post-optimization phase (based on successive removal and reinsertion of nodes). Calvo [2] solves an assignment problem with an ad hoc objective function and builds a feasible tour merging all such found sub-tours into a main tour; then a 3-opt local search procedure is applied to improve the initial feasible solution. Ohlmann and Thomas [15] use a variant of simulated annealing, called compressed annealing, which relaxes the time windows constraints by integrating a variable penalty method within a stochastic search procedure. Two new heuristics were proposed in 2010, by Blum et al. [12] and by Urrutia et al. [5]. In this paper, we compare the results of these two heuristics with ours, since they can be considered as the current state-of-the-art heuristics for TSPTW.

Blum et al. [12] proposed a hybrid method combining ant colony optimization with beam search. In general, Beam-ACO algorithms heavily rely on accurate and computationally inexpensive bounding information for differentiating between partial solutions. Urrutia et al. [5] proposed a two-stage VNS based heuristic. In the first stage, a feasible solution is constructed by using Variable neighborhood search, where the linear integer objective function is represented as an infeasibility measure. In the second stage the heuristic improves the feasible solution with a GVNS heuristic.

In this paper we propose new two-stage VNS based heuristic for solving the TSPTW problem. In the first stage, we use the same VNS as [5] to obtain feasible initial solution. In the second stage, we use new GVNS to improve the initial solution obtained in the previous stage. Our GVNS is more effective and more efficient than both state-of-the-art heuristics. Moreover, several new best known solutions are reported.

The rest of the paper is organized as follows. In Section 2, we describe implementation of our new GVNS heuristic, and in Section 3, we present computational results. Finally, in Section 4, we give some concluding remarks.

2. GVNS FOR TSPTW

General VNS is a variant of VNS where Variable neighborhood descent (VND) local search is used within basic VNS scheme (for the recent surveys on VNS see [9, 10]). Let us denote the solution of TSPTW as $x = (0, x_i, ..., x_n)$, i.e., let x be an order of clients in TSP tour that starts at depot 0.

Building an initial solution. Building an initial feasible solution is also a NP-hard problem. We start with the solution obtained as in the procedure proposed in [5]. It is a VNS based procedure that relocates customers of a random solution (minimizing its infeasibility) until a feasible solution is obtained. We also tried out different usual initialization strategies, but they did not show better performances than the one from [5].

Neighborhood structures. The most common moves performed on a TSP solution are 2-opt moves and OR-opt moves. A 2-opt move breaks down two edges of a current solution, and makes two new edges by inverting the part of a solution in such a way that the resulting solution is still a tour. One variant of 2-opt move is so-called 1-opt move which is applicable on four consecutive customers, i.e. x_1, x_2, x_3, and x_4, in such a way that edges x_1, x_2 and x_3, x_4, are broken down and the edge x_2, x_3 is inverted. On the other hand, OR-opt move relocates a chain of consecutive customers without inverting any part of a solution. If a chain contains k customers, we call such move OR-opt-k move. If a chain of k consecutive customers is moved backward, that move will be called backward OR-opt-k. Similarly, if a chain is moved forward, the move will be called forward OR-opt-k.

Maintaining feasibility. Previously described moves can be performed on each feasible solution of TSPTW problem since TSPTW is a variant of TSP. However, we must be careful because some moves can lead to infeasible solutions. So, it is important to check whether the move yields feasible or infeasible solution. For that purpose, we build an array g where g_i denotes maximal value for which arrival time at a node *i*, i.e. β_i, could be increased so that the feasibility on the final part of a tour, which starts at the node i, is kept. Elements of the array g are evaluated starting with the depot and moving backward through the tour. If we suppose that node j precedes node *i*, than g_j is calculated in the following way:

$$g_j = \min\{ g_i + \max\{0, a_j - \beta_j\}, b_j - \beta_j \} \tag{1}$$

where $g_0 = b_0 - \beta_0$. If we want to check the feasibility of a move, we have to recalculate arrival time to each customer in the move, as well as the arrival time to the first and the last customer according to the resulting tour if the move should be performed. If all these arrival times do not violate time windows and arrival time to the last customer, i.e. *i*, is increased for value less or equal to the value of g_i, then new solution is feasible, otherwise it is infeasible.

Variable neighborhood descent for the TSPTW. In our VND procedure we use the following neighborhood structures respectively: 1-opt, backward OR-opt-2, forward OR-opt-2, backward OR-opt-1, forward OR-opt-1, 2-opt. Since each move may be considered as a relocation of a customer, we decide to explore neighborhood structures by moving some customer backward or forward depending on the examined neighborhood structure. However, the search for an improvement by moving customer *i* in some neighborhood structure is stopped as soon as we find a infeasible move (move which does not keep feasibility of solution) which reduces the value of the objective function.

Each of these neighborhood structures is explored by using the best improvement search strategy. However, the search for an improvement of a current solution is continued in the next neighborhood structure regardless the improvement is found or not in a previous neighborhood structure. The whole search procedure is repeated until an improvement of a current solution can be found in some neighborhood structure.

Shaking procedure. Shaking procedure is a function named $Shake(x,k)$ which performs k random feasible OR-opt-1 moves on a given solution x.

Pseudo-code. The steps of our GVNS heuristic for solving TSPTW are given in Algorithm1.

Algorithm 1: GVNS for solving TSPTW.

```
Function GVNS();
x ← Construct Feasible Solution by VNS;
repeat
    k ← 1;
    while k ≤ k_max do
        x' ← Shake(x,k)            ;
        x'' ← SeqVND(x')           ;
        k ← k + 1;
        if x'' is better then X then
            x ← x''; k ← 1;
        end
    end
until t ≤ t_max;
```

GVNS contains 2 parameters: maximum time allowed in the search $(t_{\max})$ and the largest distance from the incumbent solution $x(t_{\max})$. GVNS terminates when the given total running time $t_{\max}$ elapses. In the inner loop, the incumbent solution x moves until no improvement is detected in the neighborhood with the largest distance from it.

3. NUMERICAL RESULTS

The proposed method is coded in C++ and run on a 2.53GHz processor. Note that our computer has similar characteristics as those used in [5] (2.4GHz processor) and in [12] (2.66GHz processor). The GVNS parameter $k_{\max}$ has been set to 30 for all test instances, whereas the parameter $t_{\max}$ has been adjusted to the particular instance. In this section we compare our GVNS with two state of the art heuristics for TSPTW. GVNS heuristic proposed in [5], we denote with GVNS-1.

3.1. GVNS versus GVNS-1

The comparison between GVNS and GVNS-1 is performed on the same benchmark test instances used in [5], where GVNS-1 was proposed. All test problems are grouped in sets of five test instances. The number of customers and the maximum range of time windows in each test instance can be deduced from the name of the test case to which that instance belong. For example, all five test instances in the test case 'n400w500' have 400 customers with maximum range of time window equal to 500. As in [5], each instance is run 30 times (starting from a different initial solution) and average results are reported. In other words, for each test instance, we calculate the average value of the objective function, average time and standard deviation σ. The obtained results are compared with those obtained with GVNS-1.

Test instances proposed by Urrutia et al. [5]. The GVNS proposed in this paper has been tested on instances introduced by Urrutia et al [5]. Values of VNS parameters k_{max} and $t_{\max}$ are set to 30 and 30 seconds, respectively. According to the obtained results (Table 1), our GVNS offers 14 new best known solutions, reducing the average computational time, in comparison with the GVNS-1, for about 50%. It should be noted that the proposed GVNS heuristic has not found the best known solution only for test case n300w200. However, the average value obtained by our GVNS on all test cases, is better than the average value obtained by the GVNS-1 (compare 12142.71 with 12149.66).

Table 1: Test cases proposed by Urrutia et al. [5]

Test case	GVNS			Time GVNS		GVNS-1			Time GVNS-1	
	min.value	av.value	σ	av.sec.	σ	min.value	av.value	σ	av.sec.	σ
n200w100	10019.6	10020.4	0.8	0.0	0.0	10019.6	10019.6	0.1	4.8	0.3
n200w200	9252.0	9254.2	11.4	0.1	0.0	9252.0	9254.1	7.2	5.8	0.2
n200w300	**8022.8**	8023.1	0.3	10.0	3.3	8026.4	8034.3	4.5	7.2	0.2
n200w400	**7062.4**	7072.4	19.3	11.8	3.7	7067.2	7079.3	4.4	8.7	0.4
n200w500	**6466.2**	6472.7	11.4	13.8	4.2	6466.4	6474.0	5.1	10.0	0.3
n250w100	12633.0	12633.0	0.0	0.0	0.0	12633.0	12633.0	0.0	9.9	0.2
n250w200	11310.4	11314.0	5.0	0.3	0.1	11310.4	11310.7	0.7	11.9	0.4
n250w300	10230.4	10231.0	3.4	3.7	1.9	10230.4	10235.1	2.8	14.9	0.6
n250w400	**8896.2**	8897.9	5.3	37.7	7.7	8899.2	8908.5	4.1	18.9	0.7
n250w500	**8069.8**	8083.5	13.2	42.2	8.1	8082.4	8082.4	6.7	20.7	0.9
n300w100	15041.2	15041.2	0.0	0.0	0.0	15041.2	15041.2	0.0	21.2	0.7
n300w200	13851.4	13857.6	14.9	0.6	0.2	13846.8	13853.1	2.3	23.7	0.6
n300w300	**11477.2**	11478.8	2.7	10.9	3.4	11477.6	11488.5	5.2	37.0	3.8
n300w400	**10402.8**	10419.6	25.5	30.0	6.0	10413.0	10437.4	12.9	31.7	1.2
n300w500	**9842.2**	9849.2	7.9	49.5	6.3	9861.8	9876.7	8.9	35.4	1.1
n350w100	17494.0	17494.0	0.0	0.0	0.0	17494.0	17494.0	0.0	41.0	2.5
n350w200	15672.0	15672.0	0.0	1.7	0.9	15672.0	15672.2	0.6	47.3	2.1
n350w300	**13648.8**	13660.8	17.8	13.2	3.8	13650.2	13654.1	1.7	54.9	2.2
n350w400	**12083.2**	12090.6	9.5	46.8	7.9	12099.0	12119.6	8.9	60.2	2.8
n350w500	**11347.8**	11360.6	17.7	59.0	7.5	11365.8	11388.2	12.0	57.8	1.2
n400w100	19454.8	19454.8	0.0	0.0	0.0	19454.8	19454.8	0.0	57.1	0.6
n400w200	18439.8	18442.6	5.1	1.8	0.4	18439.8	18439.9	0.6	66.9	1.9
n400w300	**15871.8**	15875.8	8.5	28.8	4.8	15873.4	15879.1	3.0	93.6	7.9
n400w400	**14079.4**	14112.0	24.4	54.9	6.9	14115.4	14145.5	12.9	96.2	3.9
n400w500	**12716.6**	12755.8	26.9	77.5	7.8	12747.6	12766.2	9.7	109.3	4.4
Average	12135.43	12142.71	9.25	19.78	3.40	12141.58	12149.66	4.57	37.84	1.64

Test instances proposed by Ohlmann and Thomas [15]. The proposed GVNS with t_{max} set to 30 seconds has been, also, tested on five test cases proposed by Olhmann and Thomas [15]. The obtained results (Table 2) show that the proposed GVNS is able to find best known solutions on all test cases, consuming less mean computational time in comparison to the computational time of the GVNS-1. Moreover, the mean value found by the proposed GVNS on all test cases is better than that obtained by the GVNS-1.

Table 2: Test instances proposed by Ohlmann and Thomas [15]

Test case	GVNS			Time GVNS		GVNS-1			Time GVNS-1	
	min.value	av.value	σ	av.sec.	σ	min.value	av.value	σ	av.sec.	σ
n150w120	722.0	722.1	0.6	11.2	3.6	722.0	722.3	0.4	11.8	0.3
n150w140	693.8	693.9	0.4	18.7	4.4	693.8	694.8	0.5	13.3	0.5
n150w160	671.0	672.6	2.9	13.4	5.0	671.0	671.2	0.3	15.0	0.8
n200w120	803.6	803.9	0.3	11.5	3.6	803.6	803.9	0.1	30.3	2.0
n200w140	798.0	798.7	1.6	24.7	5.9	798.0	799.5	1.1	38.0	1.1
Average	737.68	738.24	1.16	15.91	4.51	737.68	738.34	0.48	21.68	0.94

Test instances proposed by Gendreau et al. [8]. On all test instances proposed by Gendreau et al. [8], we have run the GVNS with the t_{max} set to 10 seconds. The obtained computational results are presented in Table 3. According to these results, the proposed GVNS offers one new best known solution (test case n100w100), while on all other instances, it gives the same minimal values as the GVNS-1. Similarly as on the previous test cases, our GVNS is more than two times faster than the GVNS-1 (compare 1.11 seconds with 2.45 seconds).

Table 3: Test instances proposed by Gendreau [8]

Test case	GVNS			Time GVNS		GVNS-1			Time GVNS-1	
	min.value	av.value	σ	av.sec.	σ	min.value	av.value	σ	av.sec.	σ
n20w120	265.6	265.6	0.0	0.0	0.0	265.6	265.6	0.0	0.3	0.0
n20w140	232.8	232.8	0.0	0.0	0.0	232.8	232.8	0.0	0.3	0.0
n20w160	218.2	218.2	0.0	0.0	0.0	218.2	218.2	0.0	0.3	0.0
n20w180	236.6	236.6	0.0	0.0	0.0	236.6	236.6	0.0	0.4	0.0
n20w200	241.0	241.0	0.0	0.0	0.0	241.0	241.0	0.0	0.4	0.0
n40w120	377.8	377.8	0.0	0.0	0.0	377.8	377.8	0.0	0.8	0.0
n40w140	364.4	364.4	0.0	0.0	0.0	364.4	364.4	0.0	0.8	0.0
n40w160	326.8	326.8	0.0	0.0	0.0	326.8	326.8	0.0	0.9	0.0
n40w180	330.4	330.5	0.9	2.2	1.2	330.4	331.3	0.8	1.0	0.0
n40w200	313.8	313.8	0.3	3.6	1.2	313.8	314.3	0.4	1.0	0.1
n60w120	451.0	451.0	0.0	0.3	0.2	451.0	451.0	0.1	1.5	0.1
n60w140	452.0	452.0	0.0	0.1	0.0	452.0	452.1	0.2	1.7	0.1
n60w160	464.0	464.6	0.2	0.0	0.0	464.0	464.5	0.2	1.7	0.0
n60w180	421.2	421.2	0.0	0.4	0.2	421.2	421.2	0.1	2.2	0.1
n60w200	427.4	427.4	0.0	0.3	0.1	427.4	427.4	0.0	2.4	0.1
n80w100	578.6	578.6	0.0	0.7	0.4	578.6	578.7	0.2	2.3	0.1
n80w120	541.4	541.4	0.1	1.3	0.8	541.4	541.4	0.0	2.7	0.1
n80w140	506.0	506.3	0.6	1.4	0.5	506.0	506.3	0.2	3.2	0.3
n80w160	504.8	505.1	1.2	1.5	1.1	504.8	505.5	0.7	3.3	0.1
n80w180	500.6	500.9	2.3	3.3	1.0	500.6	501.2	0.9	3.7	0.1
n80w200	481.8	481.8	0.0	0.4	0.2	481.4	481.8	0.1	4.2	0.2
n100w80	666.4	666.4	0.0	0.4	0.2	666.4	666.6	0.2	3.1	0.2
n100w100	640.6	641.0	1.5	2.8	1.1	642.0	642.1	0.1	3.7	0.1
n100w120	597.2	597.5	0.5	5.6	1.7	597.2	597.5	0.3	4.1	0.2
n100w140	548.4	548.4	0.0	0.2	0.0	548.4	548.4	0.0	4.4	0.2
n100w160	555.0	555.0	0.0	1.1	0.3	555.0	555.0	0.1	5.1	0.2
n100w180	561.6	561.6	0.0	1.2	0.6	561.6	561.6	0.0	6.3	0.3
n100w200	550.2	550.6	3.97	4.0	1.0	550.2	551.0	1.2	6.8	0.3
Average	441.27	441.37	0.41	1.11	0.42	441.31	441.50	0.20	2.45	0.10

Test instances proposed by Dumas [6]. According to the results obtained by our GVNS, with t_{max} set to 10 seconds, our GVNS manifests similar behavior as the GVNS-1 regarding the quality of the obtained solution. However, our GVNS is more than four times faster.

Table 4: Test instances proposed by Dumas [6]

Test case	GVNS			Time GVNS		GVNS-1			Time GVNS-1	
	min.value	av.value	σ	av.sec.	σ	min.value	av.value	σ	av.sec.	Σ
n20w20	361.2	361.2	0.0	0.0	0.0	361.2	361.2	0.0	0.2	0.0
n20w40	316.0	316.0	0.0	0.0	0.0	316.0	316.0	0.0	0.2	0.0
n20w60	309.8	309.8	0.0	0.0	0.0	309.8	309.8	0.0	0.2	0.0
n20w80	311.0	311.0	0.0	0.0	0.0	311.0	311.0	0.0	0.3	0.0
n20w100	275.2	275.2	0.0	0.0	0.0	275.2	275.2	0.0	0.3	0.0
n40w20	486.6	486.6	0.0	0.0	0.0	486.6	486.6	0.0	0.3	0.0
n40w40	461.0	461.0	0.0	0.0	0.0	461.0	461.0	0.0	0.4	0.0
n40w60	416.4	416.4	0.0	0.0	0.0	416.4	416.4	0.0	0.5	0.0
n40w80	399.8	399.8	0.0	1.2	0.6	399.8	399.9	0.4	0.5	0.0
n40w100	377.0	377.0	0.0	0.0	0.0	377.0	377.0	0.2	0.6	0.0
n60w20	581.6	581.6	0.0	0.0	0.0	581.6	581.6	0.0	0.6	0.0
n60w40	590.2	590.6	1.9	0.1	0.0	590.2	590.2	0.0	0.8	0.0
n60w60	560.0	560.0	0.0	0.0	0.0	560.0	560.0	0.0	0.9	0.0
n60w80	508.0	508.0	0.0	0.2	0.2	508.0	508.1	0.2	1.2	0.0
n60w100	514.8	514.8	0.0	0.1	0.1	514.8	514.8	0.0	1.3	0.0
n80w20	676.6	676.6	0.0	0.0	0.0	676.6	676.6	0.0	0.9	0.0
n80w40	630.0	630.0	0.0	0.1	0.0	630.0	630.0	0.0	1.3	0.0
n80w60	606.4	606.7	1.2	1.0	0.6	606.4	606.4	0.1	1.8	0.1
n80w80	593.8	593.9	0.2	0.6	0.8	593.8	593.8	0.1	2.1	0.1
n100w20	757.6	757.6	0.0	0.0	0.0	757.6	757.6	0.0	1.4	0.0
n100w40	701.8	701.8	0.0	0.1	0.0	701.8	701.8	0.0	1.9	0.1
n100w60	696.6	696.6	0.0	0.1	0.1	696.6	696.6	0.0	2.7	0.1
n150w120	868.4	868.4	0.0	0.1	0.1	868.4	868.4	0.0	3.6	0.3
n150w140	834.8	834.8	0.0	0.4	0.4	834.8	834.8	0.0	5.3	0.3
n150w60	818.6	818.6	0.00	1.9	0.8	818.6	818.6	0.1	7.4	0.7
n200w20	1009.0	1009.1	0.20	2.0	1.2	1009.0	1009.1	0.1	8.5	0.5
n200w40	984.2	984.2	0.21	5.2	1.4	984.2	984.2	0.1	12.6	0.8
Average	579.50	579.53	0.14	0.50	0.23	579.50	579.51	0.05	2.14	0.11

3.2. GVNS versus Beam-ACO [12]

The proposed GVNS is also tested on test instances that were not used for testing GVNS proposed in [5]. The time limit, t_{max}, for GVNS was set to 60 seconds (as it was time limit for Beam-ACO [12]). The proposed GVNS was run 15 times on each test instance. The obtained results are compared with those obtained by recently proposed Beam-ACO algorithm [12]. Comparison of results is done regarding mean relative percentage deviation (RPD) as well as standard deviation of RPD (σ RPD) n 15 runs.

Test instances proposed by Ascheuer [1]. The proposed GVNS is tested on asymmetric test instances proposed by Ascheur [1]. According to the results, it succeeded to find six new best known solutions, and only on two instances did not succeed to find best known solutions. In comparison with Beam-ACO algorithm, the proposed GVNS is more efficient in finding good solution in a reasonable amount of time. The mean value per instance found by our GVNS is less than that found by Beam-ACO algorithm. Also, the mean time per instance of GVNS is less than that of Beam-ACO.

Table 5: Test instances proposed by Ascheuer [1]

Test case	Best known	GVNS				Time GVNS		Beam-ACO [12]		Time Beam-ACO [12]	
	value	min. value	av.value	mean RPD	σ RPD	av. sec	σ	mean RPD	σ RPD	av. sec	σ
rbg010a.tw	671	671	671.00	0.00	0.00	0	0	0.00	0.00	0	0
rbg016a.tw	938	938	938.00	0.00	0.00	0	0	0.00	0.00	0	0
rbg016b.tw	1304	1304	1304.00	0.00	0.00	0	0	0.00	0.00	0	0
rbg017.2	852	852	852.00	0.00	0.00	0	0	0.00	0.00	0	0
rbg017.tw	893	893	893.00	0.00	0.00	0	0	0.00	0.00	1	1
rbg017a	4296	4296	4296.00	0.00	0.00	0	0	0.00	0.00	0	0
rbg019a	1262	1262	1262.00	0.00	0.00	0	0	0.00	0.00	0	0
rbg019b	1866	1866	1866.00	0.00	0.00	0	0	0.00	0.00	0	0
rbg019c.tw	4536	4536	4536.00	0.00	0.00	0	0	0.00	0.00	0	0
rbg019d.tw	1356	1356	1356.00	0.00	0.00	0	0	0.00	0.00	0	0
rbg020a.tw	4689	4689	4689.00	0.00	0.00	0	0	0.00	0.00	0	0
rbg021	4536	4536	4536.00	0.00	0.00	0	0	0.00	0.00	0	0
rbg021.2.tw	4528	4528	4528.00	0.00	0.00	0	0	0.00	0.00	2	2
rbg021.3.tw	4528	4528	4528.00	0.00	0.00	0	0	0.00	0.00	9	8
rbg021.4.tw	4525	4525	4525.00	0.00	0.00	0	0	0.00	0.00	0	0
rbg021.5.tw	4515	4515	4515.00	0.00	0.00	0	0	0.02	0.02	13	19
rbg021.6	4480	4480	4480.00	0.00	0.00	0	0	0.00	0.00	8	6
rbg021.7.tw	4479	4479	4479.00	0.00	0.00	0	0	0.00	0.00	2	2
rbg021.8.tw	4478	4478	4478.00	0.00	0.00	0	0	0.00	0.00	1	1
rbg021.9	4478	4478	4478.00	0.00	0.00	0	0	0.00	0.00	1	1
rbg027a.tw	5091	5091	5091.00	0.00	0.00	0	0	0.00	0.00	0	0
rbg031a	1863	1863	1863.00	0.00	0.00	0	0	0.00	0.00	1	1
rbg033a.tw	2069	2069	2069.00	0.00	0.00	0	0	0.00	0.00	0	0
rbg034a	2220	2222	2222.00	0.00	0.00	0	0	0.09	0.00	2	2
rbg035a.2	2056	2056	2056.00	0.00	0.00	0	0	0.04	0.02	15	17
rbg035a.tw	2144	2144	2144.00	0.00	0.00	0	0	0.00	0.00	1	1
rbg038a	2480	2480	2480.00	0.00	0.00	0	0	0.00	0.00	6	8
rbg040a.tw	2378	2378	2378.00	0.00	0.00	0	0	0.02	0.03	15	16
rbg041a.tw	2598	2598	2598.00	0.00	0.00	9	11	0.06	0.06	34	15
rbg042a.tw	2772	2772	2772.93	0.03	0.04	14	21	0.16	0.07	24	16
rbg048a.tw	9387	9383	9383.00	-0.04	0.00	1	1	0.11	0.05	26	16
rbg049a	10019	10018	10018.50	0.00	0.01	6	15	0.05	0.04	26	17
rbg050a	2953	2953	2953.27	0.01	0.03	11	17	0.30	0.04	20	15
rbg050b	9863	9863	9863.00	0.00	0.00	6	8	0.05	0.04	28	15
rbg050c	10026	10024	10024.00	-0.02	0.00	2	3	0.07	0.04	40	17
rbg055a	3761	3761	3761.00	0.00	0.00	0	0	0.00	0.00	11	14
rbg067a	4625	4625	4625.00	0.00	0.00	0	0	0.00	0.02	15	13
rbg086a	8400	8400	8400.00	0.00	0.00	1	1	0.06	0.05	24	19
rbg092a	7158	7158	7158.00	0.00	0.00	5	6	0.05	0.03	18	15
rbg125a	7936	7936	7936.00	0.00	0.00	0	0	0.05	0.04	32	19
rbg132	8468	8468	8468.00	0.00	0.00	6	6	0.19	0.08	27	16
rbg132.2	8191	8191	8191.73	0.01	0.01	11	17	0.45	0.14	38	17
rbg152	10032	10032	10032.00	0.00	0.00	11	16	0.06	0.03	25	18
rbg152.3	9791	9788	9788.60	-0.02	0.01	16	18	0.15	0.06	35	15
rbg172a	10950	10950	10951.20	0.01	0.01	19	20	0.39	0.16	35	17
rbg193	12535	12535	12540.20	0.04	0.03	20	17	0.29	0.14	37	15
rbg193.2	12143	12138	12141.10	-0.02	0.02	19	15	0.51	0.10	37	16
rbg201a	12948	12948	12950.10	0.02	0.02	23	16	0.48	0.12	37	14
rbg233	14992	14993	15001.30	0.06	0.03	33	17	0.56	0.15	42	10
rbg233.2	14496	14494	14497.60	0.01	0.03	33	13	0.61	0.10	43	11
Average	5551.1	5550.82	5551.35	0.002	0.005	4.9	4.8	0.01	0.03	14.6	8.5

Test instances proposed by Potvin and Bengio [18]. On all test instances proposed by Potvin and Bengio [18], the proposed GVNS is able to find best known solutions. On the other hand, Beam-ACO algorithm is also able to find all best known solutions, but on these test instances Beam-ACO algorithm has less mean RPD than the proposed GVNS.

Regarding computational time, the proposed GVNS is much faster than Beam-ACO algorithm.

Table 6: Test instances proposed by Potvin and Bengio [18]

Test case	Best known value	GVNS				Time GVNS		Beam-ACO [12]		Time Beam-ACO [12]	
		min. value	av. value	mean RPD	σ RPD	av. sec	σ	mean RPD	σ RPD	av. sec	σ
rc_201.1	444.54	444.54	444.54	0.00	0.00	0	0	0.00	0.00	0	0
rc_201.2	711.54	711.54	711.54	0.00	0.00	0	0	0.00	0.00	0	0
rc_201.3	790.61	790.61	790.61	0.00	0.00	0	0	0.00	0.00	2	3
rc_201.4	793.64	793.64	793.64	0.00	0.00	0	0	0.00	0.00	0	0
rc_202.1	771.78	771.78	771.78	0.00	0.00	14	12	0.00	0.00	0	0
rc_202.2	304.14	304.14	304.14	0.00	0.00	0	0	0.00	0.00	0	0
rc_202.3	837.72	837.72	837.72	0.00	0.00	0	0	0.00	0.00	1	1
rc_202.4	793.03	793.03	793.03	0.00	0.00	0	0	0.00	0.00	0	0
rc_203.1	453.48	453.48	453.48	0.00	0.00	0	0	0.00	0.00	0	0
rc_203.2	784.16	784.16	784.16	0.00	0.00	0	0	0.00	0.00	0	0
rc_203.3	817.53	817.53	817.53	0.00	0.00	0	0	0.00	0.00	2	2
rc_203.4	314.29	314.29	314.29	0.00	0.00	0	0	0.00	0.00	0	0
rc_204.1	878.64	878.64	878.64	0.00	0.00	0	0	0.00	0.00	11	10
rc_204.2	662.16	662.16	662.16	0.00	0.00	0	0	0.00	0.00	8	7
rc_204.3	455.03	455.03	455.03	0.00	0.00	0	0	0.00	0.00	0	0
rc_205.1	343.21	343.21	343.21	0.00	0.00	0	0	0.00	0.00	0	0
rc_205.2	755.93	755.93	755.93	0.00	0.00	0	0	0.00	0.00	0	0
rc_205.3	825.06	825.06	825.06	0.00	0.00	0	0	0.00	0.00	1	1
rc_205.4	760.47	760.47	760.47	0.00	0.00	0	0	0.00	0.00	5	5
rc_206.1	117.85	117.85	117.85	0.00	0.00	0	0	0.00	0.00	0	0
rc_206.2	828.06	828.06	828.06	0.00	0.00	0	0	0.00	0.00	0	0
rc_206.3	574.42	574.42	574.42	0.00	0.00	0	0	0.00	0.00	1	1
rc_206.4	831.67	831.67	832.06	0.05	0.18	1	4	0.00	0.00	3	2
rc_207.1	732.68	732.68	732.68	0.00	0.00	0	0	0.00	0.00	0	0
rc_207.2	701.25	701.25	701.25	0.00	0.00	0	0	0.00	0.00	7	5
rc_207.3	682.40	682.40	682.40	0.00	0.00	0	0	0.00	0.00	1	1
rc_207.4	119.64	119.64	119.64	0.00	0.00	0	0	0.00	0.00	0	0
rc_208.1	789.25	789.25	791.86	0.33	0.28	1	2	0.30	0.29	19	21
rc_208.2	533.78	533.78	533.78	0.00	0.00	0	0	0.00	0.00	1	1
rc_208.3	634.44	634.44	634.44	0.00	0.00	0	0	0.00	0.00	12	11
Average	634.75	634.75	634.85	0.013	0.015	0.53	0.60	0.010	0.010	2.47	2.37

Test instances proposed by Pesant et al. [16]. On test instances proposed by Pesant et al. [16], both the proposed GVNS and Beam-ACO, in all 15 runs succeeded to find best known solutions on each test instance. However, the proposed GVNS need significantly less time than Beam-ACO algorithm to obtain those solutions.

Table 7. Test instances proposed by Pesant et al. [16]

Test case	Best known value	GVNS				Time GVNS		Beam-ACO [12]		Time Beam-ACO [12]	
		min. value	av. value	mean RPD	σ RPD	av. sec	σ	mean RPD	σ RPD	av. sec	σ
rc201.0	628.62	628.62	628.62	0.00	0.00	0	0	0.00	0.00	0	0
rc201.1	654.70	654.70	654.70	0.00	0.00	0	0	0.00	0.00	0	0
rc201.2	707.65	707.65	707.65	0.00	0.00	0	0	0.00	0.00	0	0
rc201.3	422.54	422.54	422.54	0.00	0.00	0	0	0.00	0.00	0	0
rc202.0	496.22	496.22	496.22	0.00	0.00	0	0	0.00	0.00	0	0
rc202.1	426.53	426.53	426.53	0.00	0.00	0	0	0.00	0.00	0	0
rc202.2	611.77	611.77	611.77	0.00	0.00	0	0	0.00	0.00	0	0
rc202.3	627.85	627.85	627.85	0.00	0.00	0	0	0.00	0.00	0	0
rc203.0	727.45	727.45	727.45	0.00	0.00	0	0	0.00	0.00	1	0
rc203.1	726.99	726.99	726.99	0.00	0.00	0	0	0.00	0.00	3	3
rc203.2	617.46	617.46	617.46	0.00	0.00	0	0	0.00	0.00	1	1
rc204.0	541.45	541.45	541.45	0.00	0.00	0	0	0.00	0.00	0	0
rc204.1	485.37	485.37	485.37	0.00	0.00	0	0	0.00	0.00	2	2
rc204.2	778.40	778.40	778.40	0.00	0.00	2	5	0.00	0.01	19	14
rc205.0	511.65	511.65	511.65	0.00	0.00	0	0	0.00	0.00	0	0
rc205.1	491.22	491.22	491.22	0.00	0.00	0	0	0.00	0.00	0	0
rc205.2	714.69	714.70	714.70	0.00	0.00	0	0	0.00	0.00	1	1
rc205.3	601.24	601.24	601.24	0.00	0.00	0	0	0.00	0.00	0	0
rc206.0	835.23	835.23	835.23	0.00	0.00	3	3	0.00	0.00	5	5
rc206.1	664.73	664.73	664.73	0.00	0.00	0	0	0.00	0.00	3	3
rc206.2	655.37	655.37	655.37	0.00	0.00	0	0	0.00	0.00	2	2
rc207.0	806.69	806.69	806.69	0.00	0.00	0	0	0.00	0.00	0	0
rc207.1	726.36	726.36	726.36	0.00	0.00	0	0	0.00	0.00	2	2
rc207.2	546.41	546.41	546.41	0.00	0.00	0	0	0.00	0.00	0	0
rc208.0	820.56	820.56	820.56	0.00	0.00	1	4	0.00	0.00	7	8
rc208.1	509.04	509.04	509.04	0.00	0.00	0	0	0.00	0.00	2	2
rc208.2	503.92	503.92	503.92	0.00	0.00	0	0	0.00	0.00	1	1
Average	623.71	623.71	623.71	0.00	0.00	0.22	0.44	0.00	0.00	1.81	1.63

Test instances proposed by Langevin et al. [11]. According to test results on test instances proposed by Langevin et al. [11], the proposed GVNS need less computational time to obtain best known solutions in comparison with Beam-ACO algorithm. Also, in all 15 runs, both heuristics are able to find best known solutions on each test instance.

Table 7: Test instances proposed by Langevin et al. [11]

Test case	Best known value	GVNS				Time GVNS		Beam-ACO [12]		Time Beam-ACO [12]	
		min. value	av. value	mean RPD	σ RPD	av. sec	σ	mean RPD	σ RPD	av. sec	σ
n20w30	724.7	724.7	724.7	0.00	0.00	0	0	0.00	0.00	0	0
n20w40	721.5	721.5	721.5	0.00	0.00	0	0	0.00	0.00	0	0
n40w20	982.7	982.7	982.7	0.00	0.00	0	0	0.00	0.00	0	0
n40w40	951.8	951.8	951.8	0.00	0.00	0	0	0.00	0.00	0	0
n60w20	1215.7	1215.7	1215.7	0.00	0.00	0	0	0.00	0.00	0	1
n60w30	1183.2	1183.2	1183.2	0.00	0.00	0	0	0.00	0.00	0	0
n60w40	1160.7	1160.7	1160.7	0.00	0.00	0	0	0.00	0.00	3	2
Average	991.47	991.47	991.47	0.00	0.00	0	0	0.00	0.00	0.43	0.43

4. CONCLUSIONS

According to the numerical results reported in this paper, the proposed GVNS outperforms recently proposed GVNS based [5] and ACO - Beam heuristics [12] as currently being state-of- the-art heuristics for solving TSP with time windows constraints. Our method outperforms both mentioned heuristics with respect to the quality of solutions and CPU running time consumed. The efficiency and effectiveness of our implementation relies on the larger number of neighborhood structures examined, the new updating formula and the new efficient feasibility checking procedure. In some future work, this approach can be extended to similar TSP problems.

Acknowledgement. This work is supported by the Serbian Ministry for Education and Science, Grants ON174010 and III44006.

REFERENCES

[1] Ascheuer, N., Fischetti, M., and Grötschel, M., "Solving asymmetric travelling salesman problem with time windows by branch-and-cut", *Mathematical Programming*, 90 (2001) 475-506.
[2] Calvo, R.W., "A new heuristic for the travelling salesman problem with time windows", *Transportation Science*, 34 (2000) 113-124.
[3] Carlton, W.B., and Barnes, J.W., "Solving the travelling salesman problem with time windows using tabu search", *IEEE Transactions*, 28 (1996) 617-629.
[4] Cvetković, D., Cangalović, M., and Kovacevic-Vujcić, V., "Semidefinite relaxations of the traveling salesman problem", *Yugoslav Journal of Operations Research*, 9 (1999) 158-167.
[5] Da Silva, R.F., and Urrutia, S. "A General VNS heuristic for the travelling salesman problem with time windows", Discrete Optimization, 7 (2010) 203-211.
[6] Dumas, Y., Desrosiers, J., Gelinas, E., and Solomon, M., "An optimal algorithm for the travelling salesman problem with time windows", *Operations Research*, 43 (1995) 367-371.
[7] Gendreau, M., Hertz, A., and Laporte, G., "New insertion and postoptimization procedures for the travelling salesman problem", *Operations Research*, 40 (1992) 1086-1094.
[8] Gendreau, M., Hertz, A., Laporte, G., and Stan, M., "A generalized insertion heuristic for the travelling salesman problem with time windows", *Operation Research*, 46 (1998) 330-335.
[9] Hansen, P., Mladenović, N., and Moreno-Pérez, J.A., "Variable neighbourhood search: methods and applications (invited survey) ", *4OR*, 6 (2008) 319-360.
[10] Hansen, P., Mladenović, N., and Moreno-Pérez, J.A., "Variable neighbourhood search: methods and applications (invited survey) ", *Annals of Operation Research*, 175 (2010) 367-407.
[11] Langevin, A., Desrochers, M., Desrosiers, J., Gélinas, S., and Soumis, F., "A two-commodity flow formulation for the travelling salesman and makespan problems with time windows", *Networks*, 23 (1993) 631-640.
[12] Lopez-Ibanez, M., and Blum, C., "Beam-ACO for the travelling salesman problem with time windows", *Computers and Operations Research*, 37 (2010) 1570-1583.
[13] Mladenović, N., and Hansen, P., "Variable neighborhood search", *Computers and Operations Research*, 24 (1997) 1097-1100.
[14] Monnot, J., "Approximation results toward nearest neighbor heuristic", *Yugoslav Journal of Operations Research*, 12 (2002) 11-16.
[15] Ohlmann, J.W., and Thomas, B.W., "A compressed-annealing heuristic for the travelling salesman problem with time windows", *INFORMS Journal on Computing*, 19 (2007) 80-90.

[16] Pesant, G., Gendreau, M., Potvin, J.Y., and Rousseau, J.M., "An exact constraint logic programming algorithm for the travelling salesman problem with time windows", *Transportation Science*, 32 (1998) 12-29.
[17] Pop, P., and Popistar, C., "A new efficient transformation of the generalized vehicle routing problem into the classical vehicle routing problem", *Yugoslav Journal of Operations Research*, 21 (2011) 187-198.
[18] Potvin, J.Y., and Bengio, S., "The vehicle routing problem with time windows part II: genetic earch", *INFORMS Journal on Computing*, 8 (1996) 165-172.