XIII Balkan Conference on Operational Research

Vehicle Scheduling Problem in Sugar Beet Transportation: A General Variable Neighborhood Search Approach

ANA ANOKIĆ[1], ZORICA STANIMIROVIĆ[2], TATJANA DAVIDOVIĆ[3], ĐORĐE STAKIĆ[4]

[1] Independent researcher, anokicana@gmail.com

[2] University of Belgrade, Faculty of Mathematics, zoricast@matf.bg.ac.rs

[3] Mathematical Institute, Belgrade, tanjad@mi.sanu.ac.rs

[4] University of Belgrade, Faculty of Mathematics, djordjes@matf.bg.ac.rs

Abstract: *A variant of Vehicle Scheduling Problem (VSP) arising from the sugar beet transportation in a sugar factory in Serbia is presented. The objective of the considered VSP is to minimize the required transportation time under problem-specific constraints. The problem is first formulated as a Mixed Integer Quadratically Constrained Program (MIQCP) and then transformed to a Mixed Integer Linear Program (MILP). The proposed MILP model was used within the framework of CPLEX solver, which produced optimal solutions only for small-size problem instances. Therefore, a General Variable Neighborhood Search (GVNS) is designed to solve problem instances of larger dimensions. GVNS is evaluated and compared against CPLEX on the set of real-life and generated problem instances. Obtained computational results show that GVNS is a promising solution approach to VSP, as it is able to reach high-quality (mostly optimal) solutions within very short running times.*

Keywords: *Vehicle Scheduling Problem, Mixed Integer Quadratically Constrained Programming, Mixed Integer Linear Programming, General Variable Neighborhood Search*

1. INTRODUCTION

This study considers a variant of Vehicle Scheduling Problem (VSP) that arises from a real-life problem of optimizing sugar beet transportation in a sugar factory in Serbia. The low price of sugar beet on the market has two main consequences regarding the organization of transport. First, transportation costs represent significant percentage of total production costs and therefore, savings in this early production stage are of great importance. Second, a company needs to keep individual farmers as suppliers of raw material for a longer period. For this reason, a company usually organizes the transport of sugar beet from each of many producers to the factory on its own expense by renting vehicles and hiring workers for transporting the goods, including loading and unloading. Therefore, from the company's point of view, it is necessary to have an efficient transport organization, which will satisfy all problem-specific constraints with minimum expenditure of time and money.

In the literature, there are several studies dealing with organizing the transport of agriculture raw materials. Sethanan and Pitakaso (2016) considered a vehicle routing problem (VRP) for raw milk collection, with the goal to minimize the sum of fuel costs and the costs of cleaning raw milk tanks on vehicles. The considered problem was solved by five solution approaches based on differential evolution algorithm. The problem of collecting olive oil in Tunisia was studied by Lahyani et al. (2015) and formulated as a Multi-Compartment Vehicle Routing Problem. The authors proposed an integer linear programming formulation and used branch-and-cut algorithm to provide optimal solutions. A case study of Australian sugar mill was presented by Higgins (2006), dealing with vehicle scheduling problem in sugar cane transportation. Two metaheuristic methods, Tabu search and Variable neighborhood search, were proposed as solution approaches to the VSP considered in Higgins (2006). Milan et al. (2006) integrated the problems of rail and road sugar cane transportation in Cuba, in order to reduce the total transportation costs. As the resulting problem showed to be difficult to solve, Milan et al. (2006) first solved a smaller-size subproblem by HyperLINDO solver and than used the obtained solution to construct a feasible solution of the integrated problem. Thuankaewsing et al. (2015) investigated the problem of maximizing the estimated sugar cane yield in Thai sugar cane industry, assuming fair benefits for all producers. The considered problem was solved by a tabu search heuristic proposed in Thuankaewsing et al. (2015). A review of applications of vehicle scheduling problems and vehicle routing problems together with solution approaches can be found in Bunte and Kliewer (2009).

Note that the variant of VSP considered in this study differs from the ones considered in Higgins (2006), Milan et al. (2006), and Thuankaewsing et al. (2015), due to the differences between sugar cane and sugar beet related to the sustainability in the open, the type of vehicles used for transport, and the available resources of the sugar factory. The considered VSP is first formulated as a Mixed Integer Quadratically Constrained Program

(MIQCP), with the objective and specific constrains that arise from the real-life situation. The proposed MIQCP is further reformulated as a Mixed Integer Linear Program (MILP) and tested on real-life and generated problem instances within the framework of the commercial CPLEX solver. As CPLEX provides optimal solutions only for small-size instances, a variant of Variable Neighborhood Search (VNS) metaheuristic, known as General Variable Neighborhood Search (GVNS) is designed to solve problem instances of larger dimensions. The choice of VNS method is motivated by its successful applications to various vehicle routing and scheduling problems, such as: inventory routing and scheduling problem in Liu and Chen (2012), location routing scheduling problem in Macedo et al. (2015), Vehicle Routing Problem with multiple trips in Cheikh et al. (2015), multiple trips VRP with backhauls in Wassan et al. (2017), Heterogeneous Fleet VRP in Bula et al. (2016), etc. Constructive elements of the proposed GVNS implementation are adapted to the problem characteristics. GVNS metaheuristic is tested and compared against the CPLEX solver on the set of real-life instances obtained from a sugar company in Serbia, as well as on the set of generated instances. The obtained computational results clearly indicate the potential of GVNS for the considered variant of VSP.

2. PROBLEM DESCRIPTION AND MATHEMATICAL FORMULATIONS

In the harvesting season of the sugar beet, a precise transportation plan has to be made on the daily basis. For each day, the list of locations with the amounts of goods to be transported is prepared in advance. As the quantities of collected goods at each location significantly exceed the capacity of vehicles, each location has to be visited several times in order to be emptied. It is assumed that all vehicles have same technical characteristics (i.e., average velocity, capacity). The factory area is the starting and the finishing point for each tour, meaning that each vehicle serves only one location in a tour before it returns to the factory. Tours with more than one location are not allowed, as it is important to prevent a bad quality mixture of sugar beet from different locations. Therefore, it is possible that last vehicle that serves some location is not fully loaded. When a vehicle loaded with sugar beet arrives to the factory area, a certain time is needed for unloading the sugar beet including the analysis of samples. After the unloading is finished, a vehicle can start a new tour. It is assumed that there is enough space and labor at each location and in the factory area to complete the loading and unloading of unlimited number of vehicles at the same time.

Having in mind specific sustainability of sugar beet, it is important that the collected quantities of goods do not stand in the open for too long, otherwise they will lose quality. In the case that on some location the goods are standing longer than a predetermined number of days, this location is considered as an urgent one. It is important that urgent locations are emptied during the working day. The total amount of goods transported to the factory within the working day should not be lower than a predetermined constant, which ensures continuous work of factory machines. Once the factory machines start to work, they should not be stopped, because the starting process is very expensive.

By taking into account all problem specific constrains mentioned above, the goal of the considered VSP is to find the optimal set of vehicle schedules which minimizes the maximum working time among all vehicles, i.e., the moment of time when all vehicles finish their tours. In our problem, a vehicle schedule is defined as the array of tours and the corresponding departure times from the factory.

In order to present mathematical formulations of the considered VSP, the following notation is introduced:

J : The set of locations; I : The set of vehicles; K : The set of tours;
n : The total number of locations; m : The total number of vehicles;
k_{max} : The maximum number of tours a vehicle can make during working day;
c_j : The quantity of goods collected at location $j \in J$;
d_j : The distance between the factory and location $j \in J$;
C : Capacity of vehicle; D : Daily factory needs; v : The average speed of a vehicle;
u : The average time that a vehicle spends in factory area between two tours, (the time needed for unloading with analyzing the samples);
w : The average time needed for loading a vehicle;
S_j : The time needed for serving location $j \in J$, calculated as the sum of driving time in both directions and loading and unloading time, i.e., $S_j = 2\frac{d_j}{v} + u + w$;
t_j: The number of days that the goods are kept in the open at location $j \in J$;
t_0: The maximum number of days that the goods can stay in the open without losing quality;
U : The set of urgent locations, i.e., $U = \{j \in J : t_j > t_0\}$;
T_j : Binary value assigned to location $j \in J$, defined as $T_j = 1$ if $j \in U$, and $T_j = 0$ otherwise;
ε: Small positive constant, satisfying $0 < \varepsilon < c_j/C$ for each $j \in J$;
t_{start}: Starting time; t_{end}: The end of working day.

Note that vehicles make different number of tours during the working day, depending on the visited locations. However, the number of tours for each vehicle is limited by k_{max}. In order to equalize the number of tours for all vehicles, we introduce the concept of *virtual tour*. The duration of a virtual tour is equal to zero, as it is assumed that during this tour a vehicle stays in the factory area. Therefore, by adding virtual tours to vehicles (if necessary), the objective function value remains unchanged. On the other hand, by setting the number of tours for each vehicle to k_{max}, we are able to simplify problem formulations without affecting its characteristics.

The following decision variables are used in mathematical formulations:

- Binary variables x_{ik}^j are equal to 1 if a vehicle $i \in I$ visits location $j \in J$ in the tour $k \in K$, and 0 otherwise. If a vehicle $i \in I$ has virtual tour $k \in K$, then $\sum_{j \in J} x_{ik}^j = 0$ holds;
- Real variables t_{ik} represent the departure time of a vehicle $i \in I$ from the factory in the tour $k \in K$;
- Binary variables y_j are set to 0 if location $j \in J$ is emptied, and 1 otherwise. The role of variables y_j is to keep the track on the total amount of goods delivered to the factory from location $j \in J$. If $y_j = 1$, the corresponding amount is obtained by multiplying the vehicle capacity C with the number of tours visiting location j (by all vehicles). Otherwise, in case that $y_j = 0$, whole amount of goods c_j collected at $j \in J$ is transported to the factory;
- Real variable T stands for the objective function value, i.e., the very last moment of time when all vehicles finish their last tours.

Using the above notation and decision variables, the considered VSP can be formulated as a Mixed Integer Quadratically Constrained Program (MIQCP) as follows:

$$\min \quad T \tag{1}$$

subject to

$$\sum_{j \in J} x_{ik}^j \leq 1, \qquad \forall i \in I,\ \forall k \in K \tag{2}$$

$$\sum_{i \in I} \sum_{k \in K} C x_{ik}^j - C + \varepsilon \leq c_j, \qquad \forall j \in J \tag{3}$$

$$T_j \leq 1 - y_j, \qquad \forall j \in J \tag{4}$$

$$\sum_{j \in J} (1 - y_j) c_j + \sum_{i \in I} \sum_{j \in J} \sum_{k \in K} C y_j x_{ik}^j \geq D, \tag{5}$$

$$\sum_{i \in I} \sum_{k \in K} x_{ik}^j \geq (c_j / C) \cdot (1 - y_j), \qquad \forall j \in J \tag{6}$$

$$\sum_{i \in I} \sum_{k \in K} y_j x_{ik}^j + \varepsilon \leq c_j / C, \qquad \forall j \in J \tag{7}$$

$$t_{ik} + \sum_{j \in J} S_j x_{ik}^j \leq t_{i,k+1}, \qquad \forall i \in I,\ \forall k \in K \setminus \{k_{max}\} \tag{8}$$

$$t_{i,k_{max}} + \sum_{j \in J} S_j x_{ik_{max}}^j \leq T, \qquad \forall i \in I \tag{9}$$

$$x_{ik}^j \in \{0,1\}, \qquad \forall i \in I,\ \forall j \in J,\ \forall k \in K \tag{10}$$

$$y_j \in \{0,1\}, \qquad \forall j \in J \tag{11}$$

$$t_{start} \leq t_{ik} \leq t_{end}, \qquad \forall i \in I,\ \forall k \in K \tag{12}$$

The objective function (1) together with constraints (9) minimizes the moment of time when all vehicles finish their last tours. Each vehicle in each tour serves only one location or make a virtual tour, which is ensured by constraints (2). Constraints (3) provide that the total amount of goods transported from a location $j \in J$ is not greater than the total amount c_j collected at the same location. As the last vehicle visiting a location $j \in J$ may not be full with goods, it is necessary to subtract C on the left hand side of constraints (3). The role of small positive constant ε that is added to the left hand side of constraints (3) is to provide a strict inequality between the quantities of goods transported from a location $j \in J$ diminished by vehicle capacity C and the quantities c_j. A strict inequality prevents a vehicle to make additional unnecessary tour to location $j \in J$ in the case when c_j/C is an integer.

All goods collected at an urgent location $j \in J$ must be transported during the working day, which is ensured by constraints (4). If $t_j \leq t_0$ holds, the value of T_j is set to 0, and the constraint (4) is satisfied regardless of the transported amount of goods. Otherwise, if $t_j > t_0$, T_j is equal to 1, meaning that location $j \in J$ is urgent and it must be emptied during the working day, which further implies that variable y_j takes the value of 0. Constraint (5) provides that the total amount of goods transported to the factory is at least D. The transported amount is obtained by summing the amount of goods delivered to the factory from emptied locations (the first sum on the

left hand side of (5)) and the quantities of goods transported from locations that are still not emptied (the second sum on the left hand side of (5)).

If a location $j \in J$ is emptied, variable y_j is equal to 0 and the constraints (6) provide that location $j \in J$ is visited at least c_j/C times, which is ensured by constraints (7). In the case that location $j \in J$ is not emptied, variable y_j is set to 1, and in this case, constraints (6) are obviously satisfied. The role of constraints (8) is to provide that a vehicle can not start a new tour before finishing the previous one. More precisely, the starting time of each tour must not be less than the starting time of the previous one, increased by the serving time for a location visited in the previous tour. If a tour $k \in K \setminus \{k_{max}\}$ of a vehicle $i \in I$ is virtual, the sum on the left hand side of constraints (8) is equal to zero. The upper limit on the finishing time for each vehicle is set by the constraints (9). Finally, constraints (10)-(12) indicate the type of decision variables x_{ik}^j, y_j, and t_{ik}, respectively.

In order to reformulate the proposed MIQCP model (1)-(12) to a linear program, it is necessary to transform the product of binary variables y_j and x_{ik}^j that occurs in constraints (5) and (7). We introduce binary variables r_{ik}^j, $i \in I, j \in J, k \in K$ to replace the product $y_j x_{ik}^j$. These variables satisfy the following constraints:

$$r_{ik}^j - (y_j + x_{ik}^j)/2 \leq 0 \qquad \forall i \in I,\ \forall j \in J,\ \forall k \in K, \tag{13}$$

$$r_{ik}^j - y_j - x_{ik}^j + 1 \geq 0 \qquad \forall i \in I,\ \forall j \in J,\ \forall k \in K, \tag{14}$$

$$r_{ik}^j \in \{0,1\} \qquad \forall i \in I,\ \forall j \in J,\ \forall k \in K. \tag{15}$$

Therefore, constraint (5) is replaced by constraint (5'), while constraints (7) are replaced by constraints (7'):

$$\sum_{j \in J}(1-y_j)c_j + \sum_{i \in I}\sum_{j \in J}\sum_{k \in K} C r_{ik}^j \geq C, \tag{5'}$$

$$\sum_{i \in I}\sum_{k \in K} r_{ik}^j + \varepsilon \leq c_j/C \qquad \forall j \in J. \tag{7'}$$

Finally, MIQCP model (1)–(12) is transformed to the MILP model (1)-(4), (5'), (6), (7'), (8)-(15).

3. General Variable Neighborhood Search for the VSP in sugar beet transportation

Variable Neighborhood Search (VNS) is a well-known metaheuristic method based on systematic change of neighborhoods within a possibly randomised local search algorithm. Since the early work on VNS by Mladenović and Hansen (1997), different variants of this metaheuristic have been proposed in the literature: Basic VNS (BVNS), Reduced VNS (RVNS), Variable Neighborhood Descent (VND), General VNS (GVNS), Skewed VNS (SVNS), Variable Neighborhood Decomposition Search (VNDS), Primal-dual VNS, Parallel VNS, etc. An overview of VNS based methods and applications can be found in Hansen et al. (2010) or Hansen and Mladenović (2014). In this section, we describe the GVNS implementation proposed as a solution approach to the considered VSP.

Solution representation and objective function calculation. A solution of the problem is represented as an integer matrix S that consists of m rows of length k_{max}. The i-th row of S corresponds to vehicle $i \in I$, and contains integers $s_{ik} \in \{0,1,\ldots,n\}$, where s_{ik} denotes the index of a location visited by vehicle i in tour $k \in K$. If s_{ik} is 0, it means that the k-th tour of vehicle i is virtual. For simplicity, virtual tours (if they exist) are always placed at the end of rows in the solution matrix. Objective function of a solution is calculated as follows. For each vehicle $i \in I$, values t_{ik} representing departure times in each tour $k \in K$ are calculated based on the values s_{ik} in the solution matrix S. As all vehicles start their first tour at the same time (i.e., at the beginning of the working day), t_{i1} is set to starting time t_{start}. The departure time t_{ik} of a vehicle $i \in I$ in a tour $k \in \{2,\ldots,k_{max}\}$, is equal to $t_{i,k-1}$ increased by the time needed to serve the location visited in the tour $k-1$. The finishing time for each vehicle is obtained as the departure time in its last non-virtual tour increased by the time required to serve visited location. Finally, the objective function value $T = f(S)$ is calculated as the maximum of finishing times among all vehicles, i.e., $T = \max_{i \in I} t_f(i)$, where $t_f(i) = t_{ik_{max}} + \sum_{j \in J} S_j x_{ik_{max}}^j$. Note that all t_{ik}, for k being virtual tour, are mutually equal and represent the finishing time of the last non-virtual tour of vehicle i. In the case that $T > t_{end}$, the corresponding solution is infeasible and it is discarded.

Generating initial solution. Initial solution for GVNS is generated in a greedy way. First, urgent locations are identified, as these locations have high priority and they must be served during the working day. If the quantity of goods collected at urgent locations can not be transported by the given set of vehicles within the working day, no feasible solution exists for the given data. Otherwise, the procedure continues with sorting the locations to be served according to these two criteria: urgency and distance from the factory. The sorted list of locations is

structured as follows: at the beginning of the list are urgent locations sorted in non-decreasing order in respect to their distances from the factory, followed by non-urgent locations that are also sorted in non-decreasing order according to their distances from the factory. Initially, the elements of initial solution's matrix S are set to 0. One by one location from the sorted list is taken to fill in the first column of S (corresponding to the first tours of vehicles), then the second column (corresponding to the second tours of vehicles), etc., until all urgent locations are emptied and the factory needs are satisfied. Note that a vehicle may start its last tour to a non-empty location only if this tour can be finished until the end of working day, otherwise, this location is visited by the next vehicle. After the solution matrix S is filled in, one by one row of S is considered and locations in each row are sorted in non-increasing order in respect to their distances from the factory. In this way, it is ensured that a vehicle will first make longer tours and then the shorter ones. As described above, the values of corresponding departure times t_{ik}, $i \in I, k \in K$ are calculated based on the values s_{ik} in the sorted matrix of initial solution.

Neighborhood structures. The proposed GVNS uses four types of neighborhoods, denoted as N^I, N^{II}, N^{III}, and N^{IV}. N^I-neighbor of solution S is obtained by exchanging a tour to a non-urgent location with a tour to a non-emptied location. Neighborhood N^{II} consists of all solutions S' obtained by replacing virtual tour of a vehicle with a tour to a non-emptied location. N^{III}-neighbor S' of solution S is obtained by exchanging a tour to a non-urgent location with virtual tour in the list of tours of vehicle that has the longest working time. Finally, neighborhood N^{IV} is obtained when a pair of vehicles in S exchanges non-virtual tours. Note that moves in neighborhoods N^{II} and N^{IV} preserve the feasibility of solutions. On the other hand, the transformation that defines neighborhoods N^I and N^{III} may violate feasibility, as the factory needs in the newly obtained solution may not be satisfied. However, the next move within the same neighborhood may produce a feasible solution. It can be proved that the above described neighborhoods are correctly defined for the considered VSP.

Algorithm 1 The proposed GVNS for VSP

```
procedure GVNS(Problem Data, r_max, t_max)
    Generate initial solution S;
    repeat
        λ ← 1; r ← 1;
        while (λ ≤ 2) do
            if r ≤ r_max then
                S' ← ShakeN^I(S, r):                    //Shaking in N^I
            else
                S' ← ShakeN^II(S):                      //Shaking in N^II
            if S' is feasible then
                S'' ← VND(S'):                          //Local Search-VND
                if f(S'') < f(S) then                   //Move or Not step
                    S ← S''; r ← 1; λ ← 1;
                else  r ← r + 1;
                if r > r_max then λ ← λ + 1;
    until SessionTime ≥ t_max
```

GVNS implementation. The structure of the proposed GVNS is presented by Algorithm 1. GVNS starts with generating initial solution S in a greedy way. Each GVNS iteration starts by setting λ and r to 1 and continues with performing the three main steps *Shaking*, *Local Search*, and *Move or Not* within the neighborhood change loop while $\lambda \leq 2$. GVNS iterations are repeated until time limit t_{max} is reached. Shaking step is performed

Algorithm 2 VND

```
procedure VND(Problem Data, S')
    S'' ← S';
    while (Improvement) do
        S' ← Sort_1(S');
        λ ← 1;
        while (λ ≤ 2) do
            if λ = 1 then
                Find the best neighbor S'' ∈ N^III(S');     //Local Search in N^III
            else
                Find the best neighbor S'' ∈ N^IV(S');      //Local Search in N^IV
            if f(S'') < f(S') then                          //Move or Not step
                S' ← S''; λ ← 1;
            else  λ ← λ + 1;
    return (S');
```

within neighborhoods N^I and N^{II}. Neighborhood N^I of order r, $r = 1,\ldots,r_{max}$ used in this step is obtained by repeating r times random move that defines this neighborhood. Note that only non-urgent locations in the vehicle's list of tours are allowed to be replaced by non-emptied locations, and therefore, all urgent locations will remain served and the delivered quantities from each location will not exceed the prepared amounts. As this move can change the amount of goods delivered to the factory, it may affect the feasibility of the solution. For this reason, only feasible solution S' produced by Shaking step in neighborhood N^I is further passed to Local Search, otherwise, this step is repeated for the same value of r. When the value of r exceeds r_{max}, Shaking step continues in neighborhood N^{II} by performing an exchange of virtual tour with a tour to the non-emptied location, resulting in a feasible solution. Neighborhood change in Shaking step is regulated by parameter λ.

Instead of standard Local Search, the proposed GVNS uses VND (presented in Algorithm 2) that explores neighborhoods N^{III} and N^{IV}. First, solution S' is transformed as follows: vehicles are sorted in non-increasing order according to their finishing times, and for each vehicle, the locations are sorted in non-increasing order in respect to their distances from the factory. Note that a permutation of tours for a vehicle in a solution does not affect the objective function value, as finishing time of vehicle remains unchanged. Neighborhood N^{III} of the solution S' is first explored by performing a procedure that tries to replace one by one tour to the non-urgent location of the first vehicle (the one with a longest working time) by a virtual tour. The obtained solution is evaluated only if daily factory needs are satisfied.

Neighborhood structure N^{IV} is explored as follows. The search starts from the first ($i_1 = 1$) and the last ($i_2 = m$) vehicle in the sorted list, and tries to exchange the locations from the first tour ($k_1 = 1$) of vehicle i_1 and the last tour ($k_2 = k_{max}$) of vehicle i_2 (i.e., the longest and the shortest tour of vehicles i_1 and i_2, respectively). If this move leads to a decrease of the maximal finishing time among vehicles i_1 and i_2, the exchange of their tours k_1 and k_2 is performed. In the newly obtained solution, the tours of the two considered vehicles i_1 and i_2 are sorted in non-increasing order according to the distances of the visited locations from the factory. Finally, the list of vehicles is arranged in non-increasing order according to their finishing times. If the exchange of tours k_1 and k_2 produces no improvement of the maximal finishing time among vehicles i_1 and i_2, the Local Search continues through the sorted list of tours in vehicle i_2 and tries to exchange, one by one, its tour k_2, $k_2 < k_{max}$ with the tour k_1 of vehicle i_1. These attempts are performed until an improvement is found or the first tour of vehicle i_2 is reached without an improvement. In the latter case, the search goes forward through the sorted tours of vehicle i_1 trying to exchange, one by one, tour k_1, $k_1 \leq k_{max}$ with the tour k_2 of vehicle i_2 until an improvement is reached or k_1 reaches k_{max} without an improvement. If no improvement is found by exchanging tours of vehicles i_1 and i_2, the described steps are repeated with vehicle i_1 and vehicles $i_2 - 1, i_2 - 2, \ldots, i_1 + 1$.

If the best found solution S'' in the Local Search step is better than S', solution S' is replaced with S'' and the search continues within the neighborhood N^{III}, otherwise λ is set to $\lambda + 1$ (Move or Not step). The termination criterion for our VND is completed local search in both neighborhoods without improvement. Note that the size of neighborhoods N^{III} and N^{IV} used in VND step is $O(mk_{max})$ and $O(\binom{m}{2}k_{max}^2)$, respectively. The sorting of tours for each vehicle according to distances from the factory requires $O(mk_{max}^2)$ operations, while sorting the vehicles in respect to their finishing times is performed in $O(m^2)$ steps. Therefore, the overall worst-case complexity of our VND is $O(\binom{m}{2}k_{max}^2 + mk_{max} + mk_{max}^2 + m^2)$.

4. Experimental analysis

All computational experiments presented in this section were carried out on an Intel Core i7-2600 processor on 3.40GHz with 12GB RAM memory under Linux operating system. The experiments were performed on real-life and generated data set. Real-life instances are obtained from the considered sugar factory in Serbia. This data set includes 40 problem instances with up to 15 locations, 40 vehicles and the maximum number of 20 tours during the working day. Generated problem instances involve up to 1000 locations, 400 vehicles, and the maximum number of 400 tours during the working day. This data set includes 28 test examples, which are generated following the structure of real-life instances.

Commercial CPLEX 12.6.2 MIP solver is used to solve MILP model described in section 2. on small and medium-size instances to optimality (if possible). The time limit imposed on CPLEX run is set to 5 hours. The proposed GVNS algorithm is evaluated on all real-life and generated instances. On each instance, GVNS is run 30 times. The values of GVNS stopping criterion parameter t_{max} is set to $t_{max} = 1$ second for small-size real-life instances, $t_{max} = 10$ seconds for medium-size real-life instances, and $t_{max} = 100$ seconds for generated instances. Regarding the parameter r_{max}, eight formulas expressing r_{max} as a function of k_{max} are evaluated through preliminary computational experiments on the subset of real-life and generated test instances. Based on the obtained results, the value of r_{max} is set to $r_{max} = k_{max}/2 + 3$.

Computational results obtained on all considered instances are presented in Table 1. The left part of Table 1

Table 1: Computational results on small-size real-life instances solved to optimality by CPLEX 12.6.2

Instance	**CPLEX 12.6.2**		**GVNS**			**Instance**	**CPLEX 12.6.2**	**GVNS**		
$T_{n,m,k_{max}}$	*opt. sol.*	*t(s)*	*best*	$t(s)$	*gap(%)*	$T_{n,m,k_{max}}$	*best*	*best*	$t(s)$	*gap(%)*
$T_{3,2,4}$	**16.874**	0.09	**opt**	0.000	0.000	$T_{5,5,10}$	**17.781**	**17.781**	0.000	0.000
$T_{3,3,3}$	**13.727**	0.13	**opt**	0.000	0.000	$T_{5,10,10}$	**17.014**	**17.014**	0.079	0.005
$T_{3,3,4}$	**16.303**	0.20	**opt**	0.000	0.000	$T_{5,20,20}$	24.648	**24.591**	0.001	0.000
$T_{3,3,5}$	**17.164**	0.09	**opt**	0.000	0.000	$T_{8,40,10}$	26.291	**26.201**	2.075	0.065
$T_{3,4,2}$	**10.580**	0.04	**opt**	0.000	0.000	$T_{10,10,10}$	**16.243**	**16.243**	0.007	0.000
$T_{3,4,3}$	**13.727**	0.17	**opt**	0.000	0.000	$T_{15,20,15}$	26.214	**25.990**	0.255	0.061
$T_{3,4,4}$	**16.303**	0.08	**opt**	0.000	0.000	$T^r_{10,20,20}$	58.600	**58.486**	5.144	0.062
$T_{3,5,2}$	**12.294**	0.17	**opt**	0.000	0.000	$T^r_{10,30,15}$	/	**49.093**	0.489	0.000
$T_{4,2,4}$	**13.446**	0.05	**opt**	0.000	0.000	$T^r_{10,50,10}$	36.896	**36.667**	5.785	0.002
$T_{4,3,2}$	**10.580**	0.04	**opt**	0.000	0.000	$T^r_{15,30,25}$	/	**63.580**	7.020	0.001
$T_{4,4,2}$	**11.151**	0.07	**opt**	0.000	0.000	$T^r_{15,40,20}$	/	**53.453**	0.256	0.003
$T_{4,4,3}$	**14.299**	0.12	**opt**	0.000	0.000	$T^r_{20,30,10}$	/	**22.214**	0.007	0.000
$T_{4,4,4}$	**16.874**	0.57	**opt**	0.000	0.000	$T^r_{20,40,20}$	/	**41.163**	0.129	0.000
$T_{4,5,7}$	**15.287**	307.93	**opt**	0.172	0.037	$T^r_{25,50,15}$	/	**36.060**	1.508	0.001
$T_{5,2,4}$	**13.156**	0.06	**opt**	0.000	0.000	$T^r_{30,30,25}$	/	**46.674**	8.405	0.008
$T_{5,3,2}$	**10.580**	0.03	**opt**	0.000	0.000	$T^r_{30,60,15}$	/	**29.493**	1.840	0.006
$T_{5,3,3}$	**13.156**	0.44	**opt**	0.000	0.000	$T^r_{40,60,55}$	/	**103.680**	19.199	0.035
$T_{5,3,4}$	**16.589**	0.36	**opt**	0.000	0.000	$T^r_{45,65,70}$	/	**109.407**	16.911	0.047
$T_{5,4,2}$	**10.866**	0.10	**opt**	0.000	0.000	$T^r_{50,70,70}$	/	**116.669**	7.056	0.044
$T_{5,4,4}$	**17.446**	0.65	**opt**	0.000	0.000	$T^r_{55,65,65}$	/	**153.287**	12.581	0.012
$T_{5,4,5}$	**18.879**	0.21	**opt**	0.000	0.000	$T^r_{60,80,80}$	/	**126.453**	12.420	0.004
$T_{5,5,2}$	**11.437**	0.22	**opt**	0.000	0.000	$T^r_{70,60,60}$	/	**171.906**	28.357	0.017
$T_{5,5,5}$	**18.879**	40.74	**opt**	0.000	0.000	$T^r_{80,65,65}$	/	**181.413**	28.391	0.025
$T_{6,2,4}$	**13.160**	0.08	**opt**	0.000	0.000	$T^r_{90,80,80}$	/	**158.011**	31.732	0.016
$T_{6,3,2}$	**10.294**	0.14	**opt**	0.000	0.000	$T^r_{100,85,85}$	/	**184.046**	34.671	0.009
$T_{6,3,3}$	**12.870**	0.48	**opt**	0.000	0.000	$T^r_{120,90,90}$	/	**216.260**	38.895	0.007
$T_{6,4,2}$	**10.009**	0.13	**opt**	0.000	0.000	$T^r_{150,100,100}$	/	**240.303**	36.428	0.027
$T_{6,5,2}$	**10.866**	0.18	**opt**	0.000	0.000	$T^r_{300,120,120}$	/	**254.463**	56.403	0.031
$T_{6,6,6}$	**16.569**	258.00	**opt**	0.184	0.158	$T^r_{400,150,150}$	/	**373.000**	81.898	0.042
$T_{6,7,6}$	**11.479**	53.00	**opt**	0.000	0.000	$T^r_{500,200,200}$	/	**370.121**	95.446	0.034
$T_{7,3,3}$	**12.299**	0.36	**opt**	0.000	0.000	$T^r_{600,220,220}$	/	**428.366**	93.223	0.025
$T_{7,5,2}$	**10.580**	0.29	**opt**	0.000	0.000	$T^r_{800,300,300}$	/	**417.210**	83.698	0.009
$T_{7,5,6}$	**16.140**	24.43	**opt**	0.172	0.085	$T^r_{900,350,350}$	/	**650.036**	10.868	0.008
$T_{8,6,5}$	**14.246**	269.84	**opt**	0.049	0.000	$T^r_{1000,400,400}$	/	**948.847**	80.059	0.002
Average	**13.170**	28.22	**opt**	0.017	0.008	**Average**	/	**141.788**	23.566	0.018

contains experimental results on small-size real-life instances, while results on 6 medium-size real-life and all generated instances are presented on the right part of Table 1. Instance's name is presented in column $T_{n,m,k_{max}}$. Columns related to CPLEX contain the objective function value of the optimal solution *opt. sol.* and the corresponding runtime *t(s)*. In the case when CPLEX reached only feasible solution, the column $t(s)$ is omitted, as CPLEX run until time or memory limit was reached. The first column related to GVNS presents the objective function value of the best GVNS solution, with mark *opt* when it coincides with *opt. sol*. Remaining columns related to GVNS contain the average time *avg. t(s)* in which GVNS reaches *best* solution, the average percentage gap *avg. gap (%)* of the GVNS solution with respect to *opt. sol.* or *best*, all calculated through 30 runs. The last row of Table 1, denoted with **Average** contains average values of the presented results. The objective function values of the optimal or best-known solutions are bolded in columns *best*, as well as the best values in the last row **Average**.

As it can be seen Table 1, the proposed GVNS method reached all optimal solutions on small-size instances solved to optimality by CPLEX. On these instances, the average running time of GVNS was 0.017 seconds, which is significantly shorter compared to the average running time of CPLEX (28.22 seconds). In addition, GVNS was very stable, as it produced solutions with low average gap calculated in respect to known optimal solutions (0.008%). Table 1 also shows that CPLEX solver produced feasible solutions only for 6 real-life and two generated instances within the given time limit of 5 hours. These feasible solutions are presented in a column *best* related to CPLEX. On the other hand, the proposed GVNS reached all upper bounds provided by CPLEX and in case of five instances ($T_{5,20,20}$, $T_{8,40,10}$, $T_{15,20,15}$, $T^r_{10,20,20}$, and $T^r_{10,50,10}$), the corresponding upper bounds were improved. For medium-size real-life and generated problem instances, GVNS was also stable in providing the best-known solutions, as its average gap was very low (0.018%). The average running time that GVNS needed to obtain its best solutions on medium and generated instances was 23.566 seconds.

5. Conclusion

A variant of Vehicle Scheduling Problem that arises from optimizing the transport of sugar beet in Serbia is considered. The problem is formulated as Mixed Integer Quadratically Constraint Program (MIQCP) that is further linearized, and the obtained MILP formulation was used within the framework of CPLEX 12.6.2 MIP solver. As optimal solutions are obtained only for small-size real-life problem instances, a General Variable Neighborhood Search (GVNS) metaheuristic is designed to solve problem instances of larger dimensions, Computational results on small-size real-life data set show that GVNS quickly reaches all optimal solutions obtained by CPLEX solver. For medium-size real-life and generated test examples, GVNS either reaches or improves upper bounds provided by CPLEX within the given time limit. Based on the presented results, it may be concluded that proposed GVNS represents a promising solution approach to the considered VSP. As future work, the proposed VSP model can be extended by including non-homogenous vehicles or more than one factory. In addition, the designed GVNS may be combined with other optimization methods to improve the obtained best solutions of the considered VSP or to solve its extensions.

Acknowledgement

This research was partially supported by Serbian Ministry of Education, Science and Technological Development under the grants nos. 174010 and 174033.

REFERENCES

Bula, G., Prodhon, C., Gonzalez, F. A., Afsar, H., and Velasco, N. (2016). Variable neighborhood search to solve the vehicle routing problem for hazardous materials transportation. *Journal of Hazardous Materials*, 324(part B):472–480.

Bunte, S. and Kliewer, N. (2009). An overview on vehicle scheduling models. *Public Transport*, 1(4):299–317.

Cheikh, M., Ratli, M., Mkaouar, O., and Jarboui, B. (2015). A variable neighborhood search algorithm for the vehicle routing problem with multiple trips. *Electronic Notes in Discrete Mathematics*, 47:277–284.

Hansen, P. and Mladenović, N. (2014). Variable neighborhood search. In Burke, E. K. and Graham, R. D., editors, *Search methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, pages 313–337. Springer-Verlag, New York.

Hansen, P., Mladenović, N., and Pérez, J. A. (2010). Variable neighbourhood search: methods and applications. *Ann. Oper. Res.*, 175(1):367–407.

Higgins, A. (2006). Scheduling of road vehicles in sugarcane transport: A case study at an Australian sugar mill. *European Journal of Operational Research*, 170(3):987–1000.

Lahyani, R., Coelho, L., Khemakhem, M., Laporte, G., and Semet, F. (2015). A multi-compartment vehicle routing problem arising in the collection of olive oil in Tunisia. *Omega*, 51:1–10.

Liu, S. and Chen, A. (2012). Variable neighborhood search for the inventory routing and scheduling problem in a supply chain. *Expert Systems with Applications*, 39(4):4149–4159.

Macedo, R., Alves, C., Hanafi, S., Jarboui, B., Mladenović, N., Ramos, B., and de Carvalho, J. (2015). Skewed general variable neighborhood search for the location routing scheduling problem. *Computers & Operations Research*, 61:143–152.

Milan, E., Fernandez, S., and Aragones, L. (2006). Sugar cane transportation in Cuba, a case study. *European Journal of Operational Research*, 174(1):374–386.

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

Sethanan, K. and Pitakaso, R. (2016). Differential evolution algorithms for scheduling raw milk transportation. *Computers and Electronics in Agriculture*, 121:245–259.

Thuankaewsing, S., Khamjan, S., Piewthongngam, K., and Pathumnakul, S. (2015). Harvest scheduling algorithm to equalize supplier benefits: A case study from the Thai sugar cane industry. *Computers and Electronics in Agriculture*, 110:42–55.

Wassan, N., Wassan, N., Nagy, G., and Salhi, S. (2017). The Multiple Trip Vehicle Routing Problem with Backhauls: Formulation and a Two-Level Variable Neighbourhood Search. *Computers & Operations Research*, 78:454–467.