TONČI CARIĆ, Ph.D.[1]
(Corresponding author)
E-mail: tonci.caric@fpz.hr
JURAJ FOSIN, Ph.D.[2]
E-mail: juraj.fosin@mireo.hr
[1] University of Zagreb
Faculty of Transport and Traffic Sciences
Vukelićeva 4, 10000 Zagreb, Croatia
[2] Mireo d.d.
Buzinski prilaz 32, 10010 Zagreb, Croatia

Intelligent Transport Systems (ITS)
Original Scientific Paper
Submitted: 19 June 2019
Accepted: 21 Nov. 2019

USING CONGESTION ZONES FOR SOLVING THE TIME DEPENDENT VEHICLE ROUTING PROBLEM

ABSTRACT

This paper provides a framework for solving the Time Dependent Vehicle Routing Problem (TDVRP) by using historical data. The data are used to predict travel times during certain times of the day and derive zones of congestion that can be used by optimization algorithms. A combination of well-known algorithms was adapted to the time dependent setting and used to solve the real-world problems. The adapted algorithm outperforms the best-known results for TDVRP benchmarks. The proposed framework was applied to a real-world problem and results show a reduction in time delays in serving customers compared to the time independent case.

KEY WORDS

vehicle routing problem; time dependent travel times; congestion; city logistics;

1. INTRODUCTION

When attempting to minimize the resources needed for delivery by a fleet of vehicles, it is necessary to have some measure of travel cost and time needed to optimally traverse from one delivery point to another. Travel times depend significantly on external factors, mostly periodic congestion, but also weather, possible accidents occurring within the network, roadworks, etc. Due to the dynamic nature of traffic, using a constant travel speed for all periods of the day or for any day of the week gives a poor prediction of actual conditions. The problem is that most of these conditions are irregular and therefore cannot be used in a static environment where the conditions have to be known in advance in order to plan the routes.

In urban settings, congestion is a regular phenomenon and this regularity makes it possible to use historical data to compute congestion parameters, such as the time of occurrence and the intensity, and automatically predict areas where congestion will occur. It is possible to reduce 99% of delays when serving customers if one uses historical data on congestion [1]. While it is obvious that any potential travel planner would benefit from such data, it is still rarely used, both in planning the shortest routes and when dealing with fleet management [2]. One of the reasons is that a substantial amount of historical data is required to predict travel times precisely, while another is the rising complexity of algorithms needed to account for changing travel times.

This paper focuses on solving the Time Dependent Vehicle Routing Problem (TDVRP). Solutions to this problem involve working with information about changes in travel times due to congestion. These changes can be calculated by computing actual travel times, which depend on the vehicle departure time, or by computing coefficients to represent the slowdown intensity.

The main contribution of this paper is a framework for solving the TDVRP by using historical data that include predictions of travel times during certain times of day and from which zones of congestion for use in optimization algorithms can be derived. Our framework was applied to a real-world problem. Results showed a reduction in time delays in serving customers in comparison to a case where a standard industry solution was used. An algorithm capable of dealing with vehicle routing in a time dependent setting is proposed. To achieve this, two methods of travel time computation are compared with the best available estimates. This comparison shows that it is possible to reduce the amount of computation without notably sacrificing precision. The proposed algorithm fares well in a time dependent setting and improves upon the best published benchmark results on time dependent benchmark instances.

This paper is organized as follows. First, the paper provides an overview of previous research and then defines the problem. Next, algorithms used in this research are described, and a method of speeding up the pre-processing time of distance and travel time matrices is outlined. It is shown that this method performs well without sacrificing the quality of the results obtained. Finally, it is shown that two-phase heuristic adapted from [3] performs well even in a time dependent setting, improving published benchmark results. The proposed heuristics is used to solve a real-world problem. In conclusion, possible research directions for future studies are described.

2. LITERATURE REVIEW

This section provides an overview of the literature regarding solutions to the Time Dependent Vehicle Routing Problem (TDVRP) and congestion in the urban environment. First, an overview of relevant Vehicle Routing Problem (VRP) topics is given. Later in the section, literature concerning the time dependent variant is reviewed.

The VRP is a generalization of the well-known Travelling Salesman Problem (TSP) and is therefore also NP-hard. It was first formulated as the "truck dispatching problem" in [3]. Typically, the goal is to reduce the number of vehicles, the total distance covered by vehicles, the travel time, or some combination of these goals, although some alternative goals (such as achieving a uniform number of customers per vehicle) are not uncommon.

The VRP is an extensively studied problem with many well-defined sub-problems and variants [4, 5]. The most common variants of the problem are the Capacitated Vehicle Routing Problem (CVRP) and the Vehicle Routing Problem with Time Windows (VRPTW). The CVRP includes demands on the quantity of goods to be delivered to each customer and adds a limit to the amount of goods a vehicle can transport. The VRPTW usually includes the CVRP and adds time windows to each customer, defining when they are available for service.

Although the literature on the VRP is considerable, only a small portion refers to the vehicle routing problem with time dependent travel times [1]. For an exhaustive list of current state-of-the-art work on the broader scope of time dependent problems, the reader is referred to [2]. Some of the most important studies in this area are described below.

In 2003, Ichoua, Gendreau and Potvin proposed a stepwise speed model called (IGP model) [6]. They did not assume a constant speed over the entire length of a link, which is very important for solving TDVRP because the model guarantees that if a vehicle leaves customer i for customer j at a given time, any identical vehicle leaving a customer i for customer j at a later time will arrive later at customer j (i.e., the no-passing or first-in-first-out [FIFO] property). In the same paper Ichoua et al. [6] presented a parallel taboo search algorithm that penalizes late arrivals. Another interesting solution for TDVRP was described by Fleischmann et al. [7] who presented a general framework for the implementation of time-varying travel times in various vehicle-routing algorithms as well as TDVRP. A step forward was taken by Hashimoto et al. [8] who proposed an iterated local search algorithm that took into account both time dependent travel times and time dependent service times, as well as travel costs. It is also worth mentioning that there is a modern approach to solving TDVRP, using an ant colony algorithm, presented by Donati et al. [9].

Ehmke et al. [10] presented an interesting paper dealing with congestion and routing in which they focused on the adaptation of a TSP and VRPTW algorithm to a time dependent setting. In addition, Lecluyse et al. [11] noticed a correlation between time and spatial propagation of congestion; they devised a test with 32 customers and used circles with time dependent sizes to denote the spread of congestion time and the spatial propagation of congestion.

The VRPTW problem is a well-studied problem and Solomon benchmarks are commonly used for comparing algorithms. In [12] the well-known Solomon benchmarks are described. The benchmark consists of 56 problems, grouped into three categories by customer grouping (random, clustered, or a combination of both) and two categories by time-window length and customer demand. Later, Gehring and Homberger [13] described the generalization of Solomon benchmarks with 200, 400, 600, 800, and 1,000 customers, grouped by the same criteria.

Finally, in [14] the time dependent variants of the Solomon benchmarks are provided. These benchmarks add 12 groups of coefficients to each problem, making a total of 672 benchmark problems, varying according to the time and intensity of speed drops, as well as time windows and the geographical distribution of customers. The article also provides route construction and improvement heuristics to provide initial benchmark results for comparison.

To solve the real-world problems, a method for computing the quickest or shortest paths between customers is also required. Finding the shortest path on the road network is achieved by solving the Shortest Path Problem (SPP) on the corresponding graph. The solution to the problem is a path with a minimal sum of edge weights. This problem was researched in depth and there are various methods for generating solutions [15, 16].

When solving a TDVRP, the expected vehicle speeds change based on the vehicle departure time from a customer, meaning a single weight per edge is not sufficient, as it is in the VRPTW variant. The variant of SPP which allows variation in edge weights based on time is called a Time Dependent SPP (TDSPP). Research regarding TDSPP does exist, e.g. [17], but it is scarce.

The added complexity of computing time dependent travel times notably increases the amount of resources needed to solve the real-world TDVRP problems, which has resulted in a smaller number of papers on TDVRP, as mentioned previously in this section.

3. PROBLEM DEFINITION

In this section, a simple Vehicle Routing Problem with Time Windows (VRPTW) is described first, and then some properties of the time dependent variant (TDVRP) are discussed.

Transport networks are commonly represented as mathematical graphs $G=(V,A)$, where the set of vertices $V=\{v_0,\dots,v_{n+1}\}$ represents n customers $(v_1,\dots,v_n)$ and one depot, which is represented with two vertices (v_0,v_{n+1}) for the beginning and ending of each route [6]. The binary decision variable x_{ij}^k indicates whether vehicle k travels between customers i and j [12]. Set $A=\{(v_i,v_j):i\neq j \wedge i,j\in V\}$ is a set of edges. Each edge (v_i,v_j) is associated with distance $d_{ij}\geq 0$ and travel time $t_{ij}(b_i)\geq 0$ where b_i is the departure time from customer i. A fleet of vehicles starts their routes at a single warehouse. Each vehicle travels to a set of customers (vertices) to deliver their cargo and returns to the warehouse when finished. Every customer is served only once with only one vehicle [4]. The set of vehicles is denoted as K and the set of customers as $C=\{v_1,\dots,v_n\}$. There are constraints on the vehicles that have to be respected in order for the route to be considered feasible. First, the vehicles have a limited capacity q_{max} for $k=1,\dots,K$, while each customer has a fixed demand q_i for $i=1,\dots,n$, which in turn means a single vehicle can only deliver to a limited number of customers [3]. Furthermore, each customer i has a time window $[e_i, l_i]$ associated with it (e_i and l_i being early and late time, respectively) and service time s_i. The arrival time of the vehicle at the i-th customer on its route is denoted by a_i. If a vehicle arrives too early (before the time window of the customer opens), it has to wait for the time window to open. The departure time from customer i is denoted by b_i. Every customer has to be visited exactly once in total, but the depot is visited twice by each vehicle (once at the start and once at the end of the route). The distance between two customers c_i and c_j is denoted by d_{ij} and the corresponding travel time is denoted by t_{ij}. If a vehicle arrives at a customer, it must also depart from that customer [5]. Each vehicle leaves [7] from and returns to the depot exactly once [8]. Customer service times must satisfy time window start [9] and ending times [10]. Travel time between customers must include customer service time. [11].

The Time Dependent Vehicle Routing Problem can be described as a constraint solving problem, see for example [14]. The TDVRP is formulated as follows. The primary objective is to minimize the number of routes (*1*).

$$Minimize \sum_{k\in K}\sum_{j\in C} x_{0j}^k \tag{1}$$

The secondary objective (*2*) is the minimization of the total time or distance where c_d and c_t stand for costs per distance travelled/route duration. The y_{ij}^k stands for real number decision variable which indicates service start time for customer i served by vehicle k [13].

$$Minimize\ c_d \sum_{k\in K}\sum_{(i,j)\in A} d_{ij}^k x_{ij}^k + c_t \sum_{k\in K}\sum_{j\in C} \left(y_{n+1}^k - y_0^k\right) x_{0j}^k \tag{2}$$

The constraints are:

$$\sum_{i\in C} q_i \sum_{j\in V} x_{ij}^k \leq q_{\max}, \quad \forall k\in K \tag{3}$$

$$\sum_{k\in K}\sum_{j\in V} x_{ij}^k = 1, \quad \forall i\in C \tag{4}$$

$$\sum_{i\in V} x_{il}^k - \sum_{j\in V} x_{lj}^k = 1, \ \forall l \in C, \ \forall k \in K \quad (5)$$

$$x_{i0}^k = 0, \ x_{n+1,i}^k = 0, \ \forall i \in C, \ \forall k \in K \quad (6)$$

$$\sum_{j\in V} x_{0j}^k = 1, \ \forall k \in K \quad (7)$$

$$\sum_{j\in V} x_{j,n+1}^k = 1, \ \forall k \in K \quad (8)$$

$$e_i \sum_{j\in V} x_{i,j}^k \leq y_i^k, \ \forall i \in V, \ \forall k \in K \quad (9)$$

$$l_i \sum_{j\in V} x_{i,j}^k \geq y_i^k, \ \forall i \in V, \ \forall k \in K \quad (10)$$

$$x_{ij}^k\left(y_i^k + s_i + t_{ij}\left(y_i^k + s_i\right)\right) \leq y_j^k \ \ \forall (i,j) \in A, \ \forall k \in K \quad (11)$$

$$x_{ij}^k \in \{0,1\}, \ \forall (i,j) \in A, \ \forall k \in K \quad (12)$$

$$y_i^k \in \Re, \ \forall i \in V, \ \forall k \in K \quad (13)$$

There are two ways of handling the case of a vehicle arriving at a customer location before the time window opens or after it closes. The VRPTW with soft time windows penalizes arrivals outside the time window in the cost function and raises the cost of such routes. On the other hand, the VRPTW with hard time windows allows early arrivals, but disallows late arrivals and considers such routes as infeasible, disregarding them completely as possible solutions. If a vehicle route satisfies the capacity and the time window constraints, then the route is feasible; otherwise, if one of the constraints is not met, the route is infeasible. In this paper, the VRPTW with hard time windows is considered.

In the time independent case, travel times t_{ij} are constant, as the travel time does not change when the departure time changes. In the time dependent case, the time window of the depot is called the *time horizon*. The time horizon is split uniformly into a number of smaller intervals $T_1,...,T_p$, each with an assigned travel time matrix $\boldsymbol{C}(T_k)$, $k=1,...,p$. The travel times are then functions of time intervals, $t_{ij}(T_k)$, or alternatively of departure times $t_{ij}(b_i)$ where b_i is the departure time from customer c_i that belongs to some interval T_k.

Since distance d_{ij} and travel times t_{ij} are known in advance, the speed when driving from customer c_i to c_j at interval T_k is defined to be the ratio $v_{ij}(T_k)=d_{ij}/t_{ij}(T_k)$.

Since travel times are dependent on departure times, a change in the departure time from a customer on the route can affect arrival times at subsequent customers, and with that the feasibility of the route. This makes it more computationally expensive to perform feasibility checks during computation.

When dealing with real problems, in addition to the travel time changing, the route itself can also change. This would mean that distance d_{ij} is a function of time as well. In this research the distance between two customers is considered to be a constant.

4. PROBLEM SOLUTION

In this section a solution for the TDVRP is proposed. The solution was tested on both benchmark problems described by Figliozzi [14] and real-world delivery problems on the road network in Croatia.

The primary objective when solving the VRP is to find a solution that uses a minimal number of vehicles/routes (*1*). The secondary objective is to find a solution where vehicles travel a minimal distance or a minimal amount of time (*2*). For that reason, modern state-of-the-art algorithms use a two-phase approach, consisting of one dedicated heuristic for each objective.

To generate an initial solution, constructive heuristics are usually used. A review of constructive methods for the VRPTW can be found in [18]. In this paper the initial solution was generated by using the Solomon I1 heuristic. A method to solve both VRPTW and TDVRP were proposed. In both cases the distance and travel time matrices were computed in a pre-processing procedure in order to speed up the execution of the algorithm. The algorithm itself is a two-phase heuristic, based on general idea of the Ejection Pool (EP) algorithm for reducing the number of vehicles and an Iterative Local Search (ILS) heuristic for route optimization.

Constructive heuristics are usually used to generate an initial solution. Our implementation was based on the idea of the Solomon I1 heuristic which initializes a new route with a seed customer and inserts unrouted customers into the current route as long as a customer can be feasibly inserted. A route is considered feasible if each constraint is satisfied, e.g. all customers are served within their time window. If unrouted customers exist, a new vehicle is initialized, and customers are iteratively added to its route. A seed customer is selected as the furthest one from the depot or the one that has to be served earliest. Since this heuristic is not computationally intensive, both variants are calculated, and the better one is set as initial solution. Further details about the cost function and parameters can be found in [12].

Algorithm 1 – Ejection pool algorithm

```
1: select shortest route from $V$ and remove it
2: insert removed customers into $EP$
3: initialize penalties for all customers, $p[i] \leftarrow 1$
4: while $EP \neq \emptyset$ and $t_{EP} < t_{max}$ do
5:    select and remove customer $u_{in}$ from $EP$ $(LIFO)$
6:    if $\mathcal{N}_{add}^{fe}(u_{in,\sigma}) \neq \emptyset$ then
7:          select $\sigma' \in \mathcal{N}_{add}^{fe}(u_{in,\sigma})$
8:          $\sigma \leftarrow \sigma'$
9:    else
10:         $p[u_{in}] \leftarrow p[u_{in}] + 1$
11:         select $\sigma' \in \mathcal{N}_{EJ}^{fe}(u_{in,\sigma})$ such that $P_{sum} \leftarrow p[u_{out}^{(1)}] + \cdots + p[u_{out}^{(k)}]$ is minimal
12:         $\sigma \leftarrow \sigma'$
13:         add removed customers $\{u_{out}^{(1)}, \cdots, u_{out}^{(k)}\}$ to $EP$
14:         $\sigma \leftarrow LocalSearch(\sigma)$
15:   end if
16: end while
```

The original EP algorithm was proposed by Nagata and Braysy [19]. *Algorithm 1* is an EP algorithm modified to account for time dependent travel speeds. The "squeeze" part of the algorithm was not included for simplicity and lower execution times. The procedure is briefly described here. The algorithm derives its name from the ejection pool, a stack (LIFO structure) of customers that at some point are not served by any vehicles. The algorithm works as follows: the vehicle serving the least number of customers is removed and its customers are added to the ejection pool (*Algorithm 1*, lines 1-2). One by one, the customers are removed from the top of the ejection pool (*Algorithm 1*, line 5) and an attempt is made to insert them where feasible into other routes (*Algorithm 1*, line 6). If the attempt is successful, the process continues with the next customer in the stack.

Each customer has an associated penalty number (*Algorithm 1*, line 3) that represents how difficult it is to insert that customer feasibly. If the feasible insertion attempt fails, the penalty for the customer being inserted is incremented (*Algorithm 1*, line 10). The algorithm continues by adding the customer into a route disregarding the feasibility of the resulting route (*Algorithm 1*, lines 11-13). The attempt to restore feasibility is then made by removing the customers around the newly inserted customer. When removing the customers, certain criteria have to be met: the resulting route is feasible and the sum of penalties of the removed customers is minimal.

The removed customers are pushed to the top of the ejection pool and the procedure continues iteratively until the pool is empty. If the algorithm is unable to remove all customers from the pool in the allotted time, the last feasible solution is restored. If the pool is successfully emptied, the procedure starts from the beginning using the newly generated solution as a starting point. When removing the customers, minimizing the penalty sum of the removed customers ensures that customers who are difficult to insert feasibly are not often in the ejection pool.

Algorithm 2 – Iterative local search

```
1: $\sigma' \leftarrow \sigma$
2: while not Finished do
3:    $\sigma' \leftarrow Perturb(\sigma)$
4:    $\sigma'' \leftarrow LocalSearch(\sigma')$
5:    if $f(\sigma'') < f(\sigma)$ then
6:          $\sigma \leftarrow \sigma''$
7:    end if
8: end while
```

After the EP method finishes, the feasible solution with the lowest number of vehicles is used as a starting point for the second phase of the algorithm. The total distance or total time is reduced by the iterative local search (ILS) method (*Algorithm 2*). The ILS tries to escape from local optima by using an escape procedure (*Algorithm 2*, line 3) and by finding a new solution with local search operators (*Algorithm 2*, line 4). The operators interchange one or more customers between two routes or change the order of customers inside one route. A total of five operators is used. One operator changes the position of one customer inside a single route (Relocate operator) and four operators work on two routes: Relocate, Exchange, 2-opt*, and Cross-exchange. For details on local search operators, refer to [18].

When local search gets stuck in local optima, a perturbation mechanism is applied so that a further local search can potentially find an overall better solution. The Ruin-and-Recreate method of Schrimpf et al. described in [20] is used to alter a large part of the current solution. Customers are randomly selected and removed from the solution and then reinserted by a heuristic procedure. Two insertion heuristics are used: greedy and regret. The greedy heuristic inserts unrouted customers into the current solution so that the route length is extended minimally. This approach often results in suboptimal results, because of its short-sighted nature of looking just one step ahead. A better approach is to look at further steps, which is the main idea of the Regret heuristic by Pisinger and Ropke [21]. It calculates the cost of insertion of other unrouted customers and adds customers who will be regretted most if not added to the solution now. For details, readers may refer to the original paper [21]. Although the Regret heuristic yields better results than the greedy approach, the greedy approach is still used because of its fast execution time.

Both computation phases are terminated after a defined period of time elapses. Some changes are required to adapt a time independent algorithm to a time dependent setting. TDVRP differs most from VRPTW in the pre-processing phase when distance and travel time matrices become significantly more complex to calculate. The algorithm itself is not particularly different, but the time needs to be calculated differently. Most notably, the route feasibility checking becomes harder to perform. A common way to speed up feasibility checking in VRPTW is to compute the latest possible arrival time for each customer in a route, such that arriving before the computed time does not impact the route feasibility. This is not directly usable in TDVRP due to the fact that a change in one part of the route affects the travel time for every subsequent customer.

Instead, the maximum driving speed between each customer pair was computed. This was used to compute "the latest optimistic arrival time" for each customer which denotes the latest time of arrival at a customer such that the route remains feasible in the hypothetical case when the vehicle drives the rest of the route at its maximum speed. A violation of the latest optimistic arrival time would certainly result in an infeasible route. The inverse, however, does not hold, and if the vehicle's arrival time does not violate the latest optimistic arrival time of a customer, the time window constraints of the subsequent customers still need to be checked.

5. TRAVEL TIMES

In this section, a method for generating speed profiles and ways of incorporating them into an algorithm for solving TDVRP problems is briefly described. In this research, speed profiles are functions mapping a road segment, time of day, and day of week to the expected speed of vehicles on that segment at the specified time. They are typically derived from historical data and show typical traffic behaviour [22]. Next, we describe the profiles themselves, give an overview of a method for automatically determining congestion zones, and explain their use in optimization algorithms.

5.1 Speed profiles

The speed profiles used in this research were created using historical data from roughly 4,900 vehicles on the Croatian roads during a five-year period (2009–2014). The profiles were generated for each road link, where a link is simply a segment of road between two intersections. The digital map of the Republic of Croatia contains a total number of 448,393 links with median link length of 88 metres. The GPS data set was relatively sparse so vehicle speeds recorded by GPS devices were unusable (the time span between consecutive records could be as long as five minutes). In order to extrapolate the speeds from the dataset, each route made by a vehicle was examined. For each link in the route, the time difference between the first GPS record appearing on that link and the first record on the next link in the same route was computed. As link lengths were known in advance, it was then possible to compute the speed. Finally, the mean speed was calculated using all speeds in the same five-minute interval on that link [23]. In order to solve the time dependent SPP, the time dependent Dijkstra algorithm was used. The resulting profiles were smoothed and clustered into 4,096 clusters. The clustering was done because the number of profiles significantly impacted the computation time [24].

5.2 Pre-processing

When using speed profiles to compute the distance for use in optimization algorithms, there are two possible approaches: using an SPP algorithm every time a TDVRP algorithm needs information about the travel time or distance or precomputing optimal routes for each pair of customers and each time interval and storing the results. Solving the SPP directly while running the TDVRP algorithm is not feasible in practice as the shortest path algorithms typically run in times of the order of milliseconds and are typically called in the order of a billion times per run. The computing time also increases with the distance between the start and end points. Therefore, storing distance and travel time matrices is unavoidable; however, this can be done in different ways.

For pre-processing, the Time-Dependent Contraction Hierarchies algorithm [24], developed by Mireo d.d., was used. The algorithm itself was implemented in the C++ programing language for best performance. The digital road map was produced and provided by the same industry partner.

The simplest way is to compute the travel times for each pair of customers and each time interval appearing in the time window of the depot. For example, if the speed profile resolution is five minutes (as it was in this research), 84 travel time and distance matrices are needed for a seven-hour workday. Each matrix holds the minimal time required to traverse from one customer to another. This can create significant resource requirements even when solving problems of medium size. Moreover, the precomputing time for these problems can be huge. For example, pre-processing distances and travel times for the problem presented in this research, consisting of 187 customers in an area around Zagreb, took 297 hours of processor time. The shortest path computed three million times with an average computing time of 350 milliseconds, due to the relatively large spatial dispersion of the customers.

Furthermore, when introducing a new problem, the whole problem needs to be pre-processed from scratch. Even when adding a single customer to an existing problem, the distances between that customer and all the other customers have to be computed in all time intervals. Obviously, the simplest way is also the most resource demanding one. Next, an alternative approach to pre-processing travel times while retaining information on congestions is presented.

5.3 Congestion zones

The resource demands in pre-processing can be made by isolating a series of zones where major speed drops are expected and then finding times when congestion appears, as well as an average speed drop for each zone and the corresponding time interval [25]. A single reference route can then be used for each customer pair, with travel time multiplied by a single coefficient corresponding to the zone and route start time. As the congestion zones are computed for a large area, only a single distance matrix is used for both distance and travel time. Minimal pre-processing is needed when the problem changes (i.e. a customer is added) or when computing a completely different problem within the same area because only a single distance and travel time matrix have to be updated. Moreover, there is no need to re-compute congestion zone coefficients.

By using speed profiles, as described earlier, it was possible to devise such zones. The process began by dividing the digital map of Croatia into a square grid, each square having a width and height of 500 metres. The whole map was covered by 36,161 squares in total. For each square a slowdown coefficient was computed based on the speed drops appearing in it. Once each square had an associated slowdown coefficient, the squares were grouped spatially using image processing algorithms like morphological closing [26]. By using this technique, explained in more detail in [24], eight slowdown zones were automatically identified in Croatia, including Croatia's four largest cities (Zagreb, Split, Rijeka, and Osijek). The Zagreb zone is shown in *Figure 1*. Each square is coloured based on the maximum slowdown appearing on the links inside the square. The green squares show areas of no congestion during the day, while the red squares indicate significant congestion appearing on the links inside the squares at one point in the day. The outlined polygon represents the overall congestion zone. The zone outline was determined by connecting many

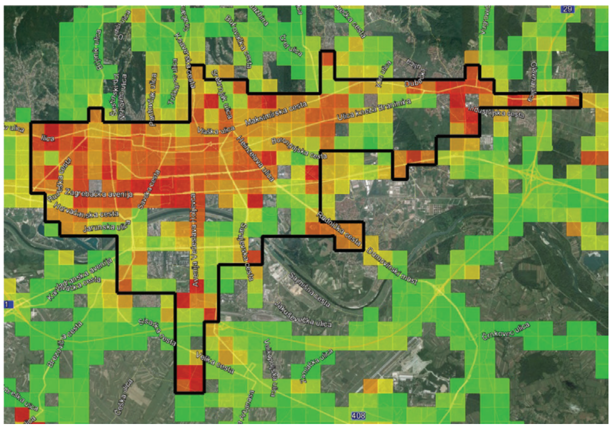


Figure 1 – An example of a developed congestion zone

smaller zones that appeared close to each other into a single large one. The outline roughly corresponds to the city limits.

Once the zones had been identified, the goal was to find the expected slowdowns inside the zones depending on the time of day. Random routes were generated with start and end points inside the zone. The departure times were randomly chosen from each of the 198 five-minute intervals, which were derived from the 990-minute daytime period (05:30 till 22:00). The midnight route duration was chosen as the referent route time. The intensity of slowdowns at a certain interval was expressed as a ratio of the travel time of the route at that interval and the travel time of the referent route. Finally, the slowdowns were grouped based on intensity and the time of appearance [25]. In total, seven daytime intervals and corresponding slowdown coefficients were developed for Zagreb. The intervals and intensities are given in *Table 1*.

Table 1 – Slowdown intervals and corresponding coefficients in Zagreb

Interval	Slowdown
05:30 – 06:45	0.9910
06:45 – 07:25	1.1790
07:25 – 08:20	1.3166
08:20 – 15:30	1.1808
15:30 – 17:05	1.3158
17:05 – 19:00	1.1624
19:00 – 22:00	1.0142

There is a possibility that a part of the shortest path between two customers lies outside the zone. This, however, does not invalidate the procedure, as such paths were included in calculations for the time and coefficients of intensity of congestion. The only fact we are interested in is the duration of the shortest path when two customers are inside the zone, not the path itself.

6. RESULTS

The results were obtained by solving three types of problems: time independent benchmark instances, time dependent benchmark instances, and a real-world delivery problem in Zagreb, the capital of Croatia. The same algorithm was used to solve the problems in all instances, as described in Section 4, with the only difference between benchmarks and the real-world problem being travel time computations, as described in Section 5. The goal function was always the same, primarily minimizing the number of vehicles and then minimizing the travel times. The results were computed on a single core of an Intel Xeon E5-2620v3 processor, compiled with the Visual C++ 11.0 compiler. The computation times were set to 15 minutes for EP and 5 minutes for ILS.

6.1 Benchmark problems

In this section, the EP_ILS algorithm is compared with the Iterative Route Construction and Improvement algorithm (IRCI) presented in [14].

The benchmark problems presented here were first presented by Figliozzi [14]. The research using the Figliozzi benchmark is scarce; however, it is crucial for the community dealing with similar problems to be able to compare the results on well-defined problems. To the authors' best knowledge, the original article [14] and papers [27] and [28] are the only studies which tested a procedure on all of the benchmark instances and reported some results. Because of that, a comparison of the EP_ILS method with other methods was first made on the time independent instances to better gauge its quality. The 56 Solomon VRPTW benchmark instances were used for comparison. The cumulative results are given in *Table 2*. The R, C, and RC columns stand for Random, Clustered, and Random-Clustered problem classes, respectively. The number after the class designation

Table 2 – Comparison of Solomon benchmark results

		R1	R2	C1	C2	RC1	RC2	CNV/CTD
Best published	Vehicle no.	11.92	2.73	10.00	3.00	11.50	3.25	405
	Distance	1,210	952	828	590	1,384	1,119	57,187
Figliozzi	Vehicle no.	12.58	3.00	10.00	3.00	12.12	3.38	422
	Distance	1,248	1,124	841	626	1,466	1,308	62,109
Rincon-Garcia et al.	Vehicle no.	12.00	2.80	10.00	3.00	11.60	3.25	408
	Distance	1,232	970	829	590	1,404	1,160	n/a
EP_ILS	Vehicle no.	12.00	2.91	10.00	3.00	11.50	3.25	408
	Distance	1,243	985	828	600	1,451	1,188	59,120

indicates the type of problem: type 1 uses smaller time windows and larger capacity constraints and type 2 larger time-windows and smaller capacity, resulting in the solutions of type 2 problems having much longer routes on average. The CNV/CTD column stands for Cumulative Number of Vehicles and Cumulative Travel Distance. The column shows the sum of vehicles and distances for all the solutions of the 56 problem instances.

The EP_ILS algorithm gives a total of 408 vehicles, whereas the best published results [29] give a total of 405. The IRCI procedure gives 422 vehicles, a difference of 3.4% compared to EP_ILS. This shows that the IRCI procedure has an inferior ability to reduce the number of vehicles in a time independent setting when compared to EP_ILS.

The time dependent problem instances used in [14] are an adaptation of Solomon benchmarks. The time horizon is split into five time intervals, $[0, 0.2l_0)$, $[0.2l_0, 0.4l_0)$, $[0.4l_0, 0.6l_0)$, $[0.6l_0, 0.8l_0)$, $[0.8l_0, l_0)$, where l_0 is the upper limit of the depot time window. Each interval is given a coefficient to represent travel speeds at certain times of the day. A speed value of 1.00 is the same as used in time independent Solomon benchmarks. It should be noted that the coefficients always increase the speed so that the problem remains feasible.

The coefficients are grouped into four categories, each category representing a different traffic scenario. For example, coefficients in group C describe the situation when there is no congestion at the beginning of the workday and the congestion builds up towards the end of the workday, while group D describes the opposite. Each group also has three levels of intensity. As an example, the coefficients from group D for all three intensity levels are listed below:

D1 = [1.00, 1.00, 1.05, 1.60, 1.60]
D2 = [1.00, 1.00, 1.50, 2.00, 2.00]
D3 = [1.00, 1.00, 1.75, 2.50, 2.50]

The similarity between the congestion zone coefficients presented in Section 5 and the coefficients used in the presented benchmarks is now obvious. While in [14] Figliozzi used speed-up coefficients, the congestion zone coefficients described here present slowdown coefficients. Computing travel times using the IGP model requires travel speeds. However, this does not present a significant change as the slowdown coefficients are directly inversely proportional to speed coefficients. The only other difference is the use of Euclidean distances to determine the congestion zone impact, as shown in Section 5.3.

The results were generated for the 56 problems and for each of the 12 groups of coefficients (672 instances in total). The results per problem type, compared to those from Figliozzi [14] and Rincon-Garcia [28], denoted as Fig and R-G respectively, are given in *Table 3*. The EP_ILS algorithm is denoted as EP_ILS. The primary objective of VRP variants is to minimize the total number of vehicles and this is the focus when considering the performance of

Table 3 – Comparison of results with Figliozzi [14] and Rincon-Garcia [28]

	Fig.	R-G	EP_ILS	Fig.	R-G	EP_ILS	Fig.	R-G	EP_ILS
	A1			A2			A3		
Vehicle no.	402	387	385	378	361	360	360	348	348
Distance	64,875	57,439	58,780	64,580	57,106	57,969	64,667	57,359	58,447
Travel tm.	53,643	46,703	47,322	45,847	39,505	39,573	41,198	35,105	34,984
	B1			B2			B3		
Vehicle no.	420	403	399	398	378	380	393	370	373
Distance	65,044	57,950	59,101	64,925	59,179	59,746	65,781	59,018	60,809
Travel tm.	54,053	47,892	48,293	46,773	41,878	41,441	42,837	37,480	37,195
	C1			C2			C3		
Vehicle no.	402	387	387	380	360	359	365	350	350
Distance	65,304	57,842	58,529	64,921	57,794	58,524	64,791	57,317	58,108
Travel tm.	53,346	47,051	47,318	45,583	40,599	40,548	40,985	36,005	35,780
	D1			D2			D3		
Vehicle no.	417	401	403	399	387	382	388	375	377
Distance	64,858	57,639	58,057	64,304	57,318	58,476	65,084	58,369	58,253
Travel tm.	54,930	48,841	49,071	47,905	42,466	43,074	44,466	39,473	39,392

Table 4 – Comparison of total results with Figliozzi [14] and Rincon-Garcia [28]

	EP_ILS	Figliozzi	Rincon-Garcia et al.
Vehicle no.	4,503	4,702 (4.42%)	4,507 (0.09%)
Distance	704,800	779,134 (10.55%)	694,329 (-1.51%)
Travel tm.	503,989	571,566 (13.41%)	502,999 (-0.2%)

algorithms. Cumulative distance or travel time can be compared only if the number of vehicles is the same. When designing the algorithm, we chose to minimize the travel time instead of distance because our intention was to solve real-world problems.

Both algorithms, R-G and EP_ILS, obtained significantly better results than those originally published by Figliozzi. The differences are the greatest in relation to the travel time, and smallest in relation to the number of vehicles used. It is obvious that the two algorithms outperform the Figliozzi algorithm that was the first one to solve the benchmarks selected. A comparison between R-G and EP_ILS is more interesting, and the winner is not obvious. Considering just the primary objective, vehicle minimization, EP_ILS obtains better results in more benchmark groups: A1, A2, B1, C2, and D2. The R-G algorithm finds solutions with a lower number of vehicles for B2, B3, D1, and D3. For other groups (A3, C1, and C3), both algorithms find solutions with the same number of vehicles. Just those three groups are sufficient to compare the relative performance of the algorithms in respect of the second objective, distance/travel time minimization. For cumulative distance, it is easy to find a winner: the R-G algorithm outperforms the EP_ILS in all groups of three benchmarks with the same number of vehicles. However, if the cumulative travel times are compared, EP_ILS obtains lower travel time in two groups (A3 and C3). As stated earlier, the EP_ILS algorithm was designed to minimize the travel time before distance, which could explain the observed results.

When analysing the results, it can be concluded that the EP_ILS algorithm performs slightly better in vehicle reduction and the R-G algorithm performs better in distance reduction. This is further highlighted in *Table 4*.

These results, coupled with the results against the Solomon benchmarks, indicate that our two-phase heuristic, which does well in a time independent setting, can work well even in a time dependent setting, if the cost function and the travel time computations are correspondingly modified.

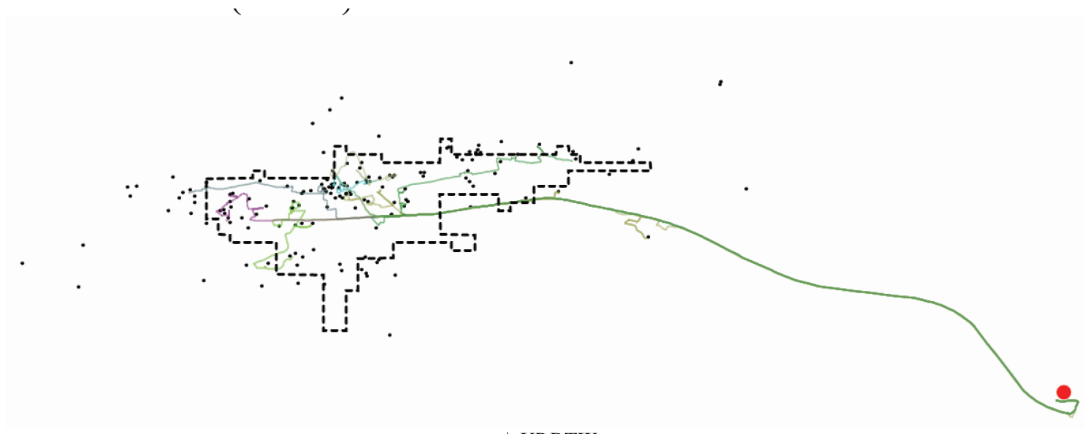
6.2 Real-world problem

The following problem comes from a large delivery company in Croatia, focusing on the capital (Zagreb) and its surrounding area. The depot is located in Ivanić Grad, a town located roughly 35 km from the centre of Zagreb. The problem consists of 156 customers, 53 of which are inside the zone defined by the procedure described in Section 5. The time windows are relatively wide, around six hours on average, allowing for many different approaches. The average service time for customers is 12 minutes. The time horizon lasts from 07:00 to 14:00, seven hours in total. The problem was solved using the algorithm described in Section 4, with two variants of travel time computation, using nighttime travel times and slowdown zones. The EP_ILS algorithm produced a solution with 7 vehicles in every run and every mode of travel time computation. Looking at the slowdown intervals in *Table 1*, it can be observed that the vehicles can travel inside the slowdown zone during three time intervals:
07:00 – 07:25 with slowdown coefficient 1.1790
07:25 – 08:20 with slowdown coefficient 1.3166
08:20 – 14:00 with slowdown coefficient 1.1808

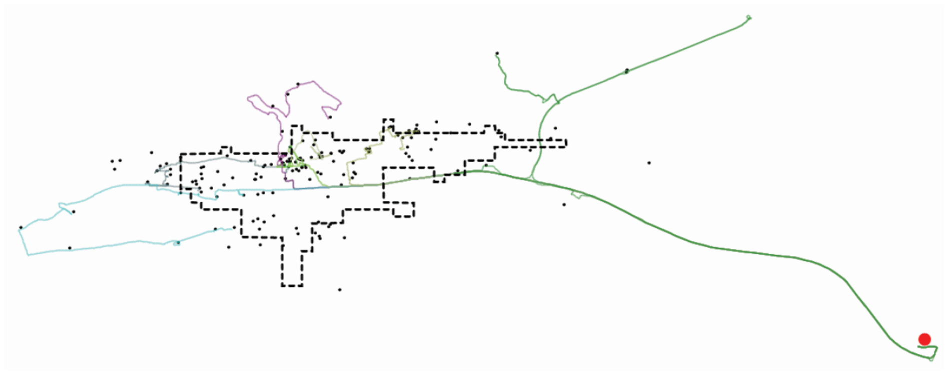
It is obvious that the second interval results in the slowest driving times and the biggest difference in travel times between the two modes of travel time computation. Consequently, there are two ways a congestion-aware vehicle could decrease the time spent in the congestion zone: by driving before the second interval starts or after it ends.

Figure 2 shows the problem and two different solutions. The black circles in the images represent customers. All vehicles started their routes in the red circle representing the depot. The route of each vehicle is shown in a different colour. A polygon that represents the congestion zone is represented by a dashed black line. Both images show estimations of vehicle positions and routes driven after two hours (at 9:00).

The upper image shows a solution obtained by solving the time independent variant (VRPTW) using nighttime speeds, while the lower image shows a solution obtained by using zone coefficients for travel time computations (TDVRPc). The imag-



a) VRPTW



b) TDVRPc

Figure 2 – Two approach solutions at 8:30

es show differences in approaching the slowdown zone when using the two travel time computation methods. The VRPTW generates solutions which disregard the congestion zone and therefore, many vehicles enter the congestion zone during the most congested period. On the other hand, TDVRPc attempts to avoid the zone, routing vehicles inside only after the problematic period has ended.

Once the solutions were obtained, each resulting route was interpreted in four different ways. First, the travel times were computed using the most precise method, that is, solving the TDSPP directly for each customer pair by using the developed speed profiles. Second, travel times were estimated using the nighttime speeds and congestion zone coefficients described in Section 5. Third, only nighttime speeds were used. Finally, speeds currently used by industrial digital map providers were used. The results of the latter three methods were compared with the first in order to measure the error in travel time estimation and to quantify the degree of lateness incurred.

TDSPP, ZC, NT and IN were used to stand for profile, zone coefficient, nighttime, and industrial travel time estimations, respectively. In order to consider the speed profiles as a basis or "ground truth", their validity needs to be established. This was done in [25] showing an average relative error of less than 5%.

Figure 3 depicts the vehicle arrival times on a single route. The x axis shows the number of the customer (position) in the route, and the y axis shows the arrival time at that customer. The graph shows absolute differences (errors) when comparing various methods of travel time estimation. The ZC travel times (blue) give the best estimations, with a dif-

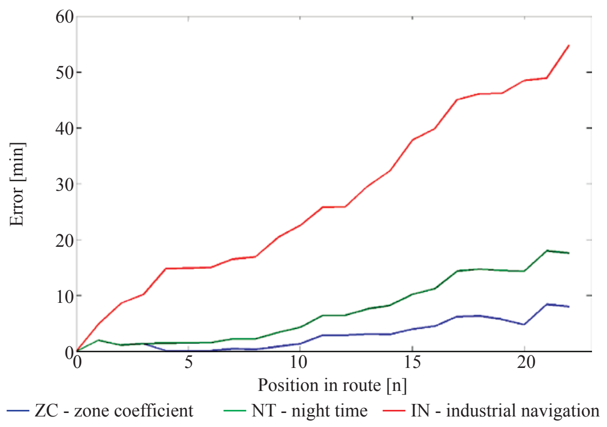


Figure 3 – Example of arrival times on one route using different travel times

ference of 185 seconds on average from times obtained from "ground truth" (TDSPP with speed profiles). The NT times (green) offer worse predictions (449 seconds difference on average); however, even the simplest method offered far better results than the industrial navigation (red), which gave a prediction with 1,705 seconds average difference. This suggests that even the simplest method based on historical data can outmatch the current time independent navigation methods when it comes to estimation precision.

Another way to compare the quality of solutions computed using the two methods is to compare the total average time the vehicles spend inside the zone for both methods. It is important to note that even in the best-case scenario, the vehicle has to spend some amount of time serving the customers inside the zone. To put the computed differences into a perspective, a lower bound had to be estimated. For this estimation the nighttime speeds and travel times were used and the lowest value out of 20 runs was used. The lower bound was estimated to be 14,164 seconds. This roughly means that spending more time in the congestion zone than the lower bound is in theory an unnecessary waste of time. Thus, the lower bound was used as a baseline to compare the two methods. When using congestion zone coefficients, the average time that the vehicles spent inside the congestion zone was 17,703.8 seconds, compared to the 18,996.19 seconds when using nighttime speeds. Regarding the lower bound, these amounted to differences of 3,539.8 and 4,832.19 seconds, respectively, a difference of 26.75%

7. CONCLUSION

Recurrent congestion is a regular phenomenon and the time of occurrence, as well as intensity, can be predicted. It is obvious that any fleet management planner would benefit from recurrent congestion prediction, but it is still rarely used because a large amount of historical data is required to predict such travel times, and the complexity of the algorithms needed to account for changing travel times is continually increasing. To reduce time delays in serving customers, algorithms for solving Time Dependent Vehicle Routing Problems (TDVRP) in the real-world logistics industry need information about variations in travel time due to congestion.

The method to speed up pre-processing of distance and travel time matrices, which is an unavoidable part of solving the real-world TDVRP problems, uses automatically generated zones to determine the slowdown periods and their intensities to determine the penalty the vehicle pays when driving along the congested routes. It was shown that this method largely reduces the time needed to pre-process the problems and compute distances between customers but at the same time minimally sacrifices the precision when compared to full pre-processing. We proposed TDVRP algorithms capable of dealing with vehicle routing in a time dependent setting. We showed that the proposed algorithm fares well in a time dependent setting, improving on the current best published results against time dependent benchmark instances. By solving the benchmark problems, it was shown that the quality of a heuristic when solving time independent problems determines its quality on time dependent problems to a large degree. The most interesting result that was achieved is that the combined Ejection Pool (EP) and Iterative Local Search (ILS) algorithms, in a real-world problem, avoid the congestion zones in peak hours, routing vehicles inside only after the problematic periods have ended.

Future work should include methods of speeding up the feasibility checks in a time dependent environment. The VRPTW solution quality suggests that better solutions can be found. Heuristics which explicitly account for changing travel times should also be considered.

ACKNOWLEDGEMENTS

The research was supported by the European Regional Development Fund under Grant KK.01.2.1.01.0120 and was also partially supported by other two sources: Croatian Science Foundation under Project IP-2018-01-8323 and Project KK.01.1.1.01.0009 (DATACROSS) within the activities of the Centre of Research Excellence for Data Science and Cooperative Systems supported by the Ministry of Science and Education of the Republic of Croatia.

Dr. sc. **TONČI CARIĆ**[1]
E-mail: tonci.caric@fpz.hr
Dr. sc. **JURAJ FOSIN**[2]
E-mail: juraj.fosin@mireo.hr
[1] Sveučilište u Zagrebu, Fakultet prometnih znanosti
Zavod za inteligentne transportne sustave
Vukelićeva 4, 10000 Zagreb, Hrvatska
[2] Mireo d.d.
Buzinski prilaz 32, 10010 Zagreb, Hrvatska

RJEŠAVANJE PROBLEMA VREMENESKI OVISNOG USMJERAVANJA VOZILA KORIŠTENJEM ZONA ZAGUŠENJA

SAŽETAK

U ovom radu prikazan je sustav za rješavanje vremenski ovisnog problema usmjeravanja vozila korištenjem povijesnih podatka. Podaci se koriste za predviđanja vremena putovanja u kritičnim vremenskim periodima dana kao i za određivanje zona zagušenja pripremljenih za uporabu u optimizacijskim algoritmima. Kombinacija dobro poznatih algoritama je prilagođena vremenski ovisnom modelu usmjeravanja vozila u svrhu rješavanja problema u praksi. Prilagođeni algoritmi daju bolje rezultate od trenutno najboljih za TDVRP ispitne zadatke. Predloženi sustav je primijenjen na stvarnim problemima, gdje pokazuje smanjenje kašnjenje pri posluživanju korisnika u odnosu na sustav koji ne uzima u obzir vremensku ovisnost trajanja putovanja o trenutku početka putovanja.

KLJUČNE RIJEČI

vremenski ovisan problem usmjeravanja vozila;
zagušenja; logistika u gradovima;

REFERENCES

[1] Kok AL, Hans EW, Schutten JMJ. Vehicle routing under time-dependent travel times: The impact of congestion avoidance. *Computers & Operations Research*. 2012;39(5): 910-8.

[2] Gendreau M, Ghiani G, Guerriero E. Time-dependent routing problems: A review. *Computers & Operations Research.* 2015;64: 189-97.

[3] Dantzig GB, Ramser JH. The truck dispatching problem. *Management Science*. 1959;6(1): 80-91.

[4] Eksioglu B, Vural AV, Reisman A. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*. 2009;57(4): 1472-83.

[5] Braekers K, Ramaekers K, van Nieuwenhuyse I. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*. 2016;99: 300-13.

[6] Ichoua S, Gendreau M, Potvin J-Y. Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research.* 2003;144: 379-96.

[7] Fleischmann B, Gietz M, Gnutzmann S. Time-varying travel times in vehicle routing. *Transportation Science*. 2004;38(2): 160-73.

[8] Hashimoto H, Yagiura M, Ibaraki T. An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. *Discrete Optimization*. 2008;5: 434-56.

[9] Donati AV, Montemanni R, Casagrande N, Rizzoli AE, Gambardella LM. Time dependent vehicle routing problem with a multi ant colony system. *European Journal of Operational Research.* 2008;185(3): 1174-91.

[10] Ehmke JF, Steinert A, Mattfeld DC. Advanced routing for city logistics service providers based on time-dependent travel times. *Journal of Computational Science.* 2012;3(4): 193-205.

[11] Lecluyse C, Sörensen K, Peremans H. A network-consistent time-dependent travel time layer for routing optimization problems. *European Journal of Operational Research.* 2013;226(3): 395-413.

[12] Solomon M. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*. 1987;35(2): 254-65.

[13] Gehring H, Homberger J. A Parallel Hybrid Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows. 1999. p. 57-64. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.617.1364&rep=rep1&type=pdf

[14] Figliozzi MA. The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics. *Transportation Research Part E: Logistics and Transportation Review.* 2012;48(3): 616-36.

[15] Dijkstra EW. A note on two problems in connexion with graphs. *Numerische Mathematik.* 1959;1(1): 269-71.

[16] Cherkassky BV, Goldberg AV, Radzik T. Shortest paths algorithms: Theory and experimental evaluation. *Mathematical Programming.* 1993;73(2): 129-74.

[17] Cooke KL, Halsey E. The shortest route through a network with time-dependent internodal transit times. *Journal of Mathematical Analysis and Applications.* 1966;14(3): 493-8.

[18] Braysy O, Gendreau M. Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. *Transportation Science*. 2005;39(1): 104-18.

[19] Nagata Y, Braysy O. A powerful route minimization heuristic for the vehicle routing problem with time windows. *Operations Research Letters*. 2009(37): 333-8.

[20] Schrimpf G, Schneider J, Stamm-Wilbrandt H, Dueck G. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*. 2000;159(2): 139-71.

[21] Pisinger D, Ropke S. A general heuristic for vehicle routing problem. *Computers & Operations Research.* 2007;34(8): 2403-35.

[22] Ehmke J. *Integration of information and optimization models for routing in city logistics*. S.l: Springer; 2014.

[23] Erdelić T, Ravlić M, Carić T. Travel time prediction using speed profiles for road network of Croatia. *2016 International Symposium ELMAR, 12-14 Sept. 2016, Zadar, Croatia*. IEEE; 2016. p. 97-100.

[24] Batz GV, Geisberger R, Neubauer S, Sanders P. Time-dependent contraction hierarchies and approximation. In: *Experimental Algorithms, Proceedings of the 9th International Symposium, SEA 2010, 20-22 May 2010, Ischia Island, Naples, Ital*y; 2010. p.166-177.

[25] Fosin J. *Time dependent vehicle routing problem solving method based on speed profiles*. PhD thesis. University of Zagreb, Faculty of Transport and Traffic Sciences; 2016.

[26] Gonzalez R, Woods R. *Digital Image Processing*. 4th ed. Pearson; 2017.

[27] Kumar SN, Panneerselvam R. A time-dependent vehicle routing problem with time windows for e-commerce supplier site pickups using genetic algorithm. *Intelligent*

Information Management. 2015;7: 181-94.

[28] Rincon-Garcia N, Waterson, B & Cherrett, T. A hybrid metaheuristic for the time-dependent vehicle routing problem with hard time windows. *International Journal of Industrial Engineering Computations*. 2017;8(1): 141-60.

[29] Desaulniers G, Madsen OBG, Ropke S. The vehicle routing problem with time windows. In: Toth P, Vigo D. (eds.) *Vehicle routing: problems, methods and applications*. 2nd ed. Philadelphia, Pennsylvania: Society for Industrial and Applied Mathematics; 2015. p. 119-159.