Solving the Time Dependent Vehicle Routing Problem Using Real-world Speed Profiles

Lovro Rožić, Juraj Fosin, Tonči Carić
Faculty of Traffic and Transport Sciences
University of Zagreb
Vukelićeva 4, 10 000 Zagreb, Croatia
{lovro.rozic, juraj.fosin, tonci.caric}@fpz.hr

Abstract. *Vehicle routing problem finds routes to serve a set of customers. It belongs to the field of intelligent transport systems and logistics. Significant savings can be achieved in real-world scenarios. The mathematical interpretation of the vehicle routing problem is an NP-hard optimization problem. Due to the computational complexity, various heuristics are used to solve the problem within a reasonable processing time. Previous research had been focused mostly on static variants, with constant edge weights represented by expected speed, which results in a too rough approximation of a dynamic traffic environment. The proposed research will take into account the time dependent aspects of the traffic environment. Edge weights will be time dependent functions acquired by analysis of historic GPS paths of vehicles. The proposed method will solve two complex problems: finding a time dependent shortest path in a graph, and solving the time dependent vehicle routing problem.*

Keywords. Vehicle routing problem, Time dependent vehicle routing problem, Iterative local search, Time dependent travel time

1 Introduction

Vehicle routing problem (VRP) needs to find an optimal set of routes for a fleet of vehicles to visit a given set of customers. The problem was introduced in 1959 to solve a real-world problem of gasoline delivery to service stations [1]. This problem usually includes a variant where each customer has a demand defined, and vehicles are constrained with a limited capacity. This is the most studied variant and is named the Capacitated Vehicle Routing Problem (CVRP) [2]. More formally, a given set of homogeneous vehicles with defined capacity has to serve a given set of customers so that every route starts and finishes at the depot. Each customer has to be visited exactly once by a single vehicle. An important extension of a basic model for real-world application is the Vehicle Routing Problem With Time Windows (VRPTW), where earliest and latest possible arrival time at customers are defined. Service at each customer can begin only in a specified time window and lasts for a given service time [4].

With the introduction of time windows prediction of travel times between customers becomes crucial in terms of robustness of calculated routes in a real-world environment. In the VRPTW model, travel time predictions are based on constant expected traveling speed for each road link in a calculated route. This means that travel time from customer i to customer j is the same regardless of the time of day or the day of the week when a vehicle leaves customer i. The VRP variant that tries to give a solution accounting for the dynamic nature of a traffic network is the Time Dependent Vehicle Routing Problem (TDVRP).

Travel time in urban areas can significantly vary depending on the time of the day. The differences in travel time can occur, for example, if congestions appear along the route during rush hours. Most of congestions are caused by daily migrations of workers and they occur on a regular basis. Thus, it can be beneficial for an algorithm to take them into account when optimizing solutions. Traffic congestions can be avoided by selecting alternative routes or by rearranging customer sequences in routes. Kok et al. [3] report that 99% of late arrivals to customers can be eliminated if one accounts for traffic congestions. Moreover, they report that about 87% of the extra duty time caused by traffic congestion can be eliminated by smart congestion avoidance strategies.

TDVRP will be modeled as a modified VRPTW problem. Let $G = (V, E)$ be a connected digraph consisting of a set V of $n + 1$ nodes that represent customers, and a set E of arcs with non-negative weights d_{ij} and with associated travel times, t_{ij}. Node 0 represents the depot, while other nodes represent customers. Each customer must be serviced exactly once by one vehicle. Each route starts and ends at the depot. Service in each node i starts in a specified time window (t_{e_i}, t_{l_i}). Service time t_s is also defined. If a vehicle arrives to customer i before opening of the time window it has to wait until t_{e_i} to start with service. Each vehicle v has a capacity Q, and each customer has a non-negative demand q_i. Solution is feasible if each

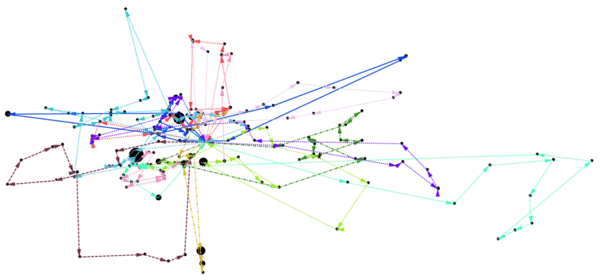

Figure 1: Illustration of VRPTW

customer is visited exactly once by one vehicle, if each vehicle used has enough capacity to service customers in its route and if vehicles arrive at customers in their route before t_{l_i} is closed.

Although scientific literature on the CVRP and VRPTW is exhaustive, the TDVRP variant received very little attention [3]. The reason is that the dynamic VRP is much harder to model and to solve [5].

The time dependent VRP was first formulated by Malandraki [6] using a mixed integer linear programming formulation. Ichoua et al. solved TDVRP with a parallel Tabu Search algorithm. Fleischmann et al. developed an algorithm that solves various variants of basic VRP model, among others, one is TD-VRP. Hashimoto et al. [7] developed an iterated local search (ILS) algorithm. Van Wonsel et al. [5] solved TDVRP without time windows using the Tabu search algorithm. An algorithm based on the ant colonies was presented by Donati et al. [8]. Figliozzi [9] introduced benchmark instances based on standard Solomon benchmarks with the addition of coefficients that modify travel time between nodes in given time intervals. Ehmke et al. [10] modified TSP and VRPTW algorithms to take into account time dependent travel times.

This paper is organized as follows. Section 2 presents a method to obtain time dependent travel times and a time dependent shortest path algorithm is briefly described. Section 3 presents an iterated local search algorithm to solve the time dependent vehicle routing problem. Section 4 describes test instances and gives results for both benchmarks and the examined real-world problem. In the same section a brief discussion of obtained computational results is given. Section 5 concludes the paper.

2 Time dependent travel times

2.1 Speed profiles

When solving real world routing problems, information about traffic patterns of various roads is crucial for accurate results. As the vehicles travel across some part of the road network the conditions on the road change. Some changes, such as congestions or seasonal changes, are periodical and predictable, which

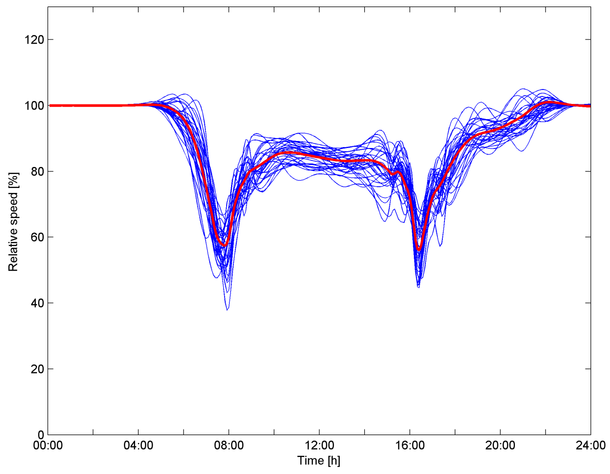

Figure 2: An example of a computed cluster with corresponding elements.

allows the usage of historical data for predicting future travel times. In this work, historical GPS records of vehicles were used to develop speed profiles for the road network of Croatia. The method used is similar to the method provided in [13], with some changes and improvements.

The GPS data used to create the speed profiles was collected by around 4000 taxis and delivery vehicles travelling on the road network of Croatia during a five year period (from August 2009 to September 2014). The data was provided by the partner company Mireo Inc. which also did the map matching of the GPS path, as well as providing a map of Croatia with a traffic layer. The data was stored chronologically for each vehicle, so that it was possible to reconstruct the routes of tracked vehicles. The provided map of Croatia is divided into a network of road-links, where a link is most commonly a road segment between two intersections, at most 100 meters long, and is the smallest unit of the network on which all computations are done. The roads of Croatia consist of a total of $450,000$ links. For two way segments both directions belong to a single link but are processed separately.

To account for the daily changes in traffic patterns each hour was divided into five minute intervals, with special handling of the intervals between 22:00 and 5:30. Each day of the week was processed separately to account for differences between them (especially between workdays and weekends). Before computing the profiles, the free-flow speed of each link was computed as a reference speed. The free-flow speed is defined as the average speed of passenger cars in conditions of low traffic flow rates. As no explicit information about free-flow speeds was available, the free-flow speed for each link was approximated using night time speeds, as there are no congestions at night and traffic flows freely. The night intervals (22:00 - 5:30) were all given free-flow speeds.

The speeds for the rest of the day were computed

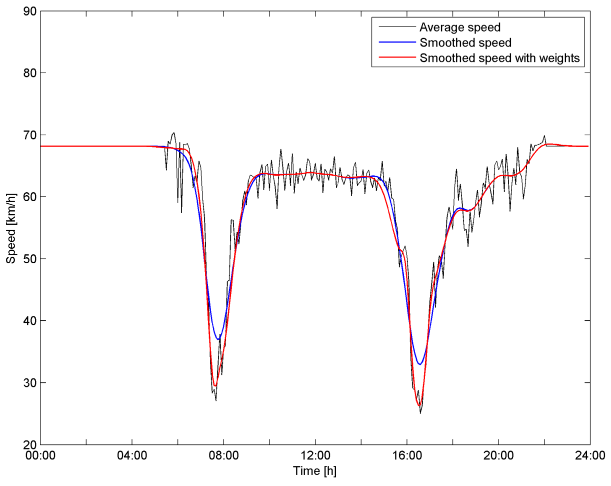

Figure 3: An example profile showing differences between weighted and unweighted smoothing.

separately for every interval. The GPS signals were recorded approximately every 100 meters or every 5 minutes, which produced a relatively sparse dataset. Even though the speeds recorded by GPS devices are generally very accurate, due to the sparsity of the dataset the recorded speeds were unusable and real speeds had to be extrapolated from the recorded distance covered and the time difference between two GPS records. The method used here is the same as described in [13], but instead of using arithmetic mean (also called time mean speed) the harmonic mean (also called space mean speed) was used. The reason for this is that, during congestions, the standard deviation of recorded speeds is large, which makes the time mean speed a poor predictor as it gives equal weight to both high and low speeds. Space mean speed, on the other hand, puts more weight on lower speeds and so gives a more realistic prediction of travel speed.

The method in short is the following: for each link in a vehicle route an entry point and an exit point are determined. The entry point is the first record which appears on the given link, and the exit point is the first point in the same route appearing on the next link in the route. The speed is then determined using the elapsed time and distance between the entry and exit points. This method includes the time needed to traverse the intersection, where most of the speed drops on a link appear. By using only data which appears on a single link one would discard such important information. Finally, the computed speed is added to the intervals between the link entry and exit time and, once all the records are processed, the space mean speed of each interval is computed and stored as a relative slowdown coefficient with regard to the free-flow speed. If there is insufficient data, the intervals and days are combined as in [13].

Next, the resulting data are smoothed with a weighted smoothing spline. Weights were added when speed drops were higher than 25%, relative to the intensity of the drop, in order to counter the tendency of the smoothing spline to pull data points towards the regression line, giving optimistic travel speed predictions as a result. The comparison of raw, smoothed and weighted profiles is shown in figure 3.

Finally, the resulting $680,000$ profiles were clustered into $3,230$ clusters using k-means to reduce storage space. One cluster showing both morning and evening congestions is shown in figure 2.

2.2 Shortest Path Problem

When solving a real world problem, Euclidean distances are a poor approximation of actual distances between customers, as there are rarely direct routes from one customer to another. Instead, the road network is represented by a digraph $G(V,E)$, with the set of nodes V corresponding to intersections and set of edges E representing the roads between them.

In the time-independent case, the edges of the graph are given constant weights W_{uv}, which correspond to some cost of travelling from customer u to customer v. Although widely used, the time independent Shortest Path Problem (SPP) is unable to account for daily changes in travel times and speeds, as the cost of travelling between two customers is always constant.

On the other hand, in the time-dependent case the constant weights are replaced by functions $W_{uv} : \mathbb{R} \to \mathbb{R}_{\geq 0}$ as described in [14]. The functions depend on time as a parameter, allowing for precise approximations of traffic conditions dependent on the time of day. This model allows the SPP algorithm to account for congestions and dynamically choose the best path according to the time dependent shortest path criteria. In this work, when solving the standard benchmarks the travel time is used as a criterion, but when solving the real-world problem both distance and travel time are used to find optimal routes.

The algorithm most commonly used for finding the shortest path in a graph is the well known Dijkstra algorithm [15], or one of its numerous modifications. The modifications deal mostly with preprocessing to make the search space as small as possible while not sacrificing precision [14].

Finally, in order to use real parameters to compute route lengths and durations, the distances and travel times were precomputed and stored as matrices. Each interval was assigned two matrices, one for distance and one for travel times. Night time intervals all used the same matrices as it was expected that there are no significant changes in traffic behaviour during the night, while the rest of the intervals each had a separate distance and time matrix computed. Once the matrices are computed and loaded, in order to retrieve distances and travel times for a pair of customers one also has to supply the interval during which the trip begins. The procedure then is simply to retrieve the matrix corresponding to the given interval and the element of the matrix corresponding to the given customer indices.

3 Iterative local search algorithm

Iterative local search iteratively builds a sequence of solutions generated by the embedded heuristic, leading to far better solutions than if one were to use repeated random trials of that heuristic [11]. A VRPTW variant of the ILS algorithm was modified to solve TDVRP. ILS was chosen as it is simple, easy to implement, robust and highly effective. Pseudo code of the algorithm is given with (algorithm 1).

Algorithm 1 Iterative local search

```
1:  $s \leftarrow InitialSolution()$
2:  $s \leftarrow LocalSearch(s)$
3:  $s' \leftarrow s$
4:  while $not$ Terminate do
5:      $s' \leftarrow Perturbate(s)$
6:      $s'' \leftarrow LocalSearch(s')$
7:      if $f(s'') < f(s)$ then
8:          $s \leftarrow s''$
9:      end if
10: end while
```

ILS first generates an initial solution (line 1), which is then improved by a local search procedure (line 2). The algorithm then iteratively tries to escape from local optima by perturbing (line 5) and improving (line 6) the current solution until the termination criteria are met (line 4). If the found solution is better according to the acceptance criteria (line 7), the current solution is set as a best solution found so far.

3.1 Initial solution

The initial solution is calculated by a Solomon I1 VRPTW algorithm [12], modified to take into account the variable travel times occurring in TDVRP. A route is first initialized by a seed customer, after which unrouted customers are added to the route until no possible insertions are found. If not all customers are visited and there are no feasible insertions into existing routes, a new vehicle is initialized. This procedure is repeated until all customers are visited.

3.2 Local search

For a local search procedure, the most important part is selecting move mechanisms which explore neighboring solutions. These moves are often called the local search operators. For this algorithm, we have selected one operator that tries to improve a single route and three operators which change two routes. Since our goal was to implement a fast ILS algorithm, we selected simple operators. The relocate operator tries to change order of the customers in a single route to obtain a better solution. Since we tried to solve the real-world problem with narrow time windows, we decided not to implement the 2-opt operator. Of the operators which try to improve the solution by changing two routes, we selected the Relocate, Exchange and 2-opt* operators. For more details on local search procedures we refer to the survey of Bräysy and Gendrau [4].

The operators were modified to work in a time-dependent environment. When a route is changed, the modified operators check its feasibility, from the position of the changed customer along all subsequent customers by evaluating their new arrival times. This modification of operators is required as a change in the arrival time of a customer affects all subsequent customers due to time-dependent travel times.

The acceptance strategy for the operators is best accept, which searches all possible feasible moves and selects the best one.

3.3 Perturbation

After the local search gets stuck in a local optimum, a perturbation procedure tries to find unexplored neighborhoods so that a further local search can potentially find an overall better solution. The implemented perturbation is essentially a random move in a neighborhood. A number of randomized Relocate and Exchange moves which accept worse solutions (while still retaining feasibility) are used to escape current local optima.

3.4 Acceptance criterion

The acceptance strategy can help guide local search to find solutions that have some desired properties. As mentioned in the introduction, VRPTW has two objectives: first to minimize the number of routes, and second to minimize total distance. When running the ILS algorithm we use two acceptance criteria: in the first half of the running time we try to minimize total time spent on the road, while in the second half we minimize the total distance. The assumption is that if we minimize the total time needed for delivery to be made, there will be more free space in a route to squeeze in customers from other routes. That way, the Relocate operator can empty routes containing a small number of customers. Although this is a simple approach and not a state-of-the-art route minimization technique, some routes are nevertheless reduced.

4 Results

All computational test were conducted on a workstation with Intel Xeon E5-1607v3 processor and 8GB of random access memory, running a $64-$bit Microsoft's Windows 7 operating system. The algorithm was implemented in *C++* programing language and compiled

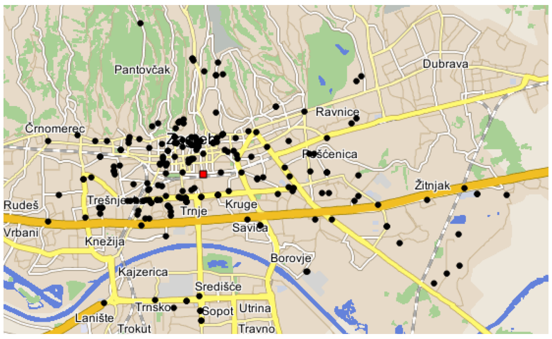

Figure 4: Real-world problem displayed on digital map

with *Visual C++* 11.0 compiler. The graphical user interface was developed as a Windows form using *.NET 4.5* framework.

The main idea of the work was applying the ILS algorithm to a real-world problem using speed profiles computed from historical GPS data. To benchmark our algorithm we also applied it to the Solomon benchmark problems with time-dependent travel times, provided in [9].

4.1 Real-world problem

For a test on a real-world problem data was acquired from a national postal service for a delivery to a large customer in the area of the capital city of the Croatia. Figure 4 shows the problem on the digital map, where customers are represented as circles in black color and the depot as a red square. Routes were recorded with GPS loggers, so that a precise comparison can be made. The problem has 225 customers served by 16 vehicles. Delivery occurs in the morning, and each customer has a time window and a service time defined. Customers on average have time windows opened for 66 minutes. Customer demand q is expressed as a quantity of large postal bags to be delivered. Delivery is made by a homogeneous fleet of vehicles with defined capacity Q.

Analysis of GPS tracks recorded by vehicles driving the actual routes used in practice showed that the total route length was 240 kilometers. The minimum number of vehicles was computed by taking capacity constraints into account, giving a lower bound of 12 vehicles in order to feasibly solve the problem. The algorithm described in section 3 was set for 110 iterations. Matrices were calculated for each 5 minute interval in a day, a total of 198 matrices during daytime (5:30-22:00) and one matrix representing night. Average runtime of the described ILS algorithm was 198.3 seconds, while the calculation of 199 matrices lasted approximately 30 hours. After the end of the first phase of the algorithm, a solution with 12 vehicles was obtained for all of the 30 runs, resulting in a 25% decrease in used vehicle number compared to the number actually used. In the second phase, the minimization of the total length occurs. In the best solution found, vehicles travel total of 219 kilometers, the average total was 228, and worst was 238. Even the worst solution is better than routes used in practice (240 kilometers). Figure 5 shows the best solution found. The depot is shown as a blue circle, and customers are shown as a black circles where the diameter depends on a customers demand q. For better visibility of routes in the solution, the digital map is not shown, but routes are drawn on real road links.

4.2 Standard benchmarks

The described ILS algorithm was tested on the benchmarks provided in [9] in order to assess its general usability. The benchmarks attempt to simulate real world conditions by changing vehicle speeds depending on the time a vehicle starts travelling from one customer to another. The base for the instances are the well known Solomon benchmarks [12] with modified travel times. The time window of the depot is split into five time intervals, and each interval is given a coefficient describing the speed at the corresponding interval. Due to the relatively tight constraints on time windows in the Solomon benchmarks, to keep the problem feasible speeds are always increased by multiplying them by some coefficient. By setting all coefficients to 1, the original Solomon benchmarks can be obtained.

The author of [9] provided several sets of instances, all attempting to simulate various conditions in an urban network. There are four sets of different coefficients added to time intervals $[0, 0.2t_{l_0})$, $[0.2t_{l_0}, 0.4t_{l_0})$, $[0.4t_{l_0}, 0.6t_{l_0})$, $[0.6t_{l_0}, 0.8t_{l_0})$, $[0.8t_{l_0}, l_0]$, where t_{l_0} corresponds to the closing time of the depot.

For example, one of the variants called TD1d has coefficients $[1.00, 1.00, 1.05, 1.60, 1.60]$ respectively, modelling the case when congestions start early in the morning and subside towards the end of the workday, or put simply, when the morning speeds are low and later speeds are high. The variants TD2d and TD3d cover situations where the speed rise is more pronounced by using higher speed coefficients. Other variants cover different situations, as explained in much detail in [9].

Each instance was run for 110 iterations for a total of eight times. The averaged results of running ILS on the described benchmarks are shown in tables 1 - 4. The average execution time for all instances was 26 seconds, where the problem types C1, R1 and RC1 performed much faster (17 seconds on average) than the C2, R2 and RC2 (43 seconds on average), due to fact that the operators work with much larger routes when solving the latter instances.

In total, the algorithm in [9] was better when reducing the number of vehicles (12.12% on average). However, as was shown in subsection 4.1, the main focus was distance and travel time reduction as the vehicle limit of the real-world problem is easily reached. The

Table 1: Results for standard TDVRP benchmarks set A

	R1		R2		C1		C2		RC1		RC2	
	No.	Tr. time	No.	Tr. time	No.	Tr. time	No.	Tr. time	No.	Tr. time	No.	Tr. time
TD1												
Figliozzi	11.67	1080	2.82	990	10.00	729	3.00	563	11.38	1164	3.25	1177
ILS	13.22	1046.08	3.64	819.98	10.00	732.91	3.08	540.50	12.44	1154.45	4.25	929.10
Δ[%]	13.28	−3.14	29.08	−17.17	0.00	0.54	2.67	−4.00	9.31	−0.82	30.77	−21.06
TD2												
Figliozzi	10.75	897	2.55	861	10.00	644	3.00	495	10.50	989	2.88	993
ILS	11.33	767.859	3.18	593.01	10.01	624.20	3.14	440.53	11.11	846.05	3.69	697.11
Δ[%]	5.40	−14.40	24.71	−31.13	0.10	−3.07	4.67	−11.00	5.81	−14.45	28.13	−29.80
TD3												
Figliozzi	9.92	793	2.27	774	10.00	608	3.00	485	10.00	860	2.75	867
ILS	10.91	670.36	3.18	508.35	10.04	576.329	3.34	406.96	10.76	724.26	3.25	618.93
Δ[%]	9.98	−15.47	40.09	−34.32	0.40	−5.21	11.33	−16.09	7.60	−15.78	18.18	−28.61

Table 2: Results for standard TDVRP benchmarks set B

	R1		R2		C1		C2		RC1		RC2	
	No.	Tr. time	No.	Tr. time	No.	Tr. time	No.	Tr. time	No.	Tr. time	No.	Tr. time
TD1												
Figliozzi	12.42	1064	3.00	1027	10.00	732	3.00	545	12.13	1180	3.38	1200
ILS	13.11	1053.58	3.55	782.27	10.00	760.88	3.02	502.25	12.73	1233.36	4.19	933.78
Δ[%]	5.56	−0.98	18.33	−23.83	0.00	3.95	0.67	−7.84	4.95	4.52	23.96	−22.19
TD2												
Figliozzi	11.50	905	2.73	893	10.00	650	3.00	467	11.25	1010	3.25	1053
ILS	12.74	921.05	3.36	661.72	10.00	689.62	3.05	438.55	12.5	1070.07	4.00	803.08
Δ[%]	10.78	1.77	23.08	−25.90	0.00	6.10	1.67	−6.09	11.11	5.95	23.08	−23.73
TD3												
Figliozzi	11.42	808	2.73	831	10.00	584	3.00	446	11.00	916	3.00	981
ILS	12.71	843.61	3.27	587.36	10.01	651.70	3.19	407.09	12.33	985.04	3.88	736.81
Δ[%]	11.30	4.41	19.78	−29.32	0.10	11.59	6.33	−8.72	12.09	7.54	29.33	−24.89

Table 3: Results for standard TDVRP benchmarks set C

	R1		R2		C1		C2		RC1		RC2	
	No.	Tr. time	No.	Tr. time	No.	Tr. time	No.	Tr. time	No.	Tr. time	No.	Tr. time
TD1												
Figliozzi	11.67	1066	2.73	1003	10.00	697	3.00	573	11.50	1186	3.25	1147
ILS	12.08	968.37	3.27	733.37	10.00	743.29	3.18	507.35	11.81	1120.01	4.00	861.41
Δ[%]	3.51	−9.16	19.78	−26.88	0.00	6.64	6.00	−11.46	2.70	−5.56	23.08	−24.90
TD2												
Figliozzi	10.83	881	2.55	843	10.00	618	3.00	483	10.75	1012	2.75	1027
ILS	11.32	810.34	3.18	611.26	10.25	674.08	3.23	444.68	11.63	936.51	3.36	761.86
Δ[%]	4.52	−8.02	24.71	−27.49	2.50	9.07	7.67	−7.93	8.19	−7.46	22.18	−25.82
TD3												
Figliozzi	10.17	801	2.36	760	10.00	565	3.00	451	10.13	904	2.75	886
ILS	10.77	715.99	3	544.67	10.35	633.88	3.30	409.21	11.27	823.40	3.38	665.28
Δ[%]	5.90	−10.61	27.12	−28.33	3.50	12.19	10.00	−9.27	11.25	−8.92	22.91	−24.91

Table 4: Results for standard TDVRP benchmarks set D

	R1		R2		C1		C2		RC1		RC2	
	No.	Tr. time	No.	Tr. time	No.	Tr. time	No.	Tr. time	No.	Tr. time	No.	Tr. time
TD1												
Figliozzi	12.25	1114	3.00	1045	10.00	731	3.00	552	12.00	1192	3.38	1192
ILS	13.17	1037.41	3.73	799.91	10.00	727.26	3.02	501.35	12.75	1168.04	4.13	922.62
Δ[%]	7.51	−6.88	24.33	−23.45	0.00	−0.51	0.67	−9.18	6.25	−2.01	22.19	−22.60
TD2												
Figliozzi	11.58	943	2.73	915	10.00	652	3.00	494	11.25	1035	3.25	1053
ILS	12.78	915.19	3.53	710.31	10.00	657.75	3.02	437.03	12.19	1015.00	4.29	804.44
Δ[%]	10.36	−2.95	29.30	−22.37	0.00	0.88	0.67	−11.53	8.36	−1.93	32.00	−23.60
TD3												
Figliozzi	11.08	871	2.64	864	10.00	612	3.00	461	10.75	964	3.25	975
ILS	12.66	843.89	3.49	655.64	10.03	617.83	3.00	402.32	12.08	927.04	4.00	741.01
Δ[%]	14.26	−3.11	32.20	−24.12	0.30	0.95	0.00	−12.73	12.37	−3.83	23.08	−24.00

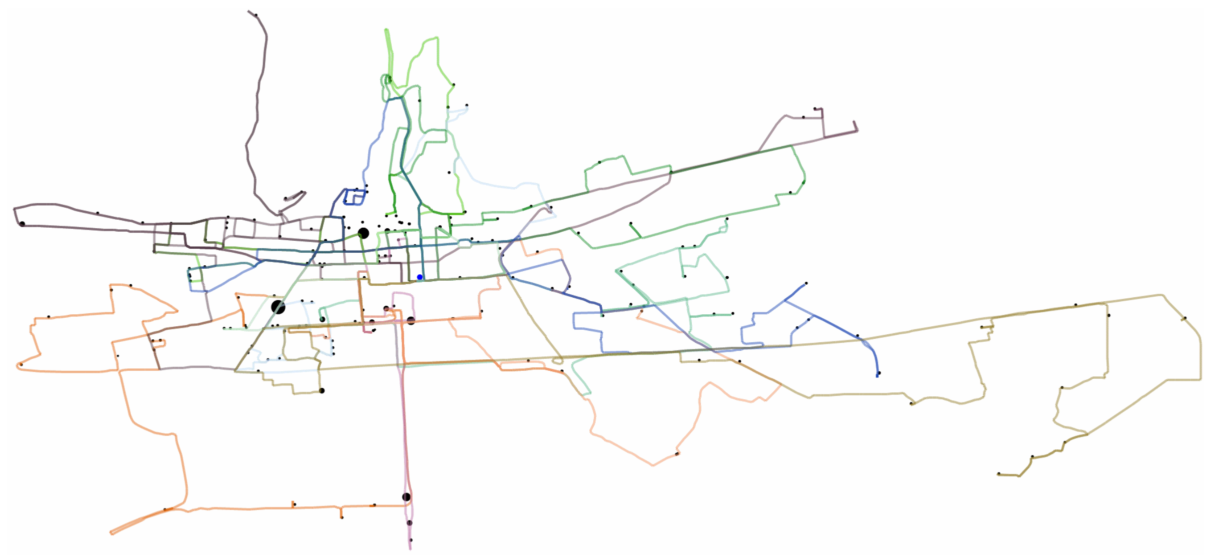

Figure 5: Best solution for tested real world problem

algorithms show similar ability to reduce travel times. When both algorithms reach the same number of vehicles (less than 2% difference), the average difference in travel times is 1.7%. The best results of our algorithm are achieved for the C2 type problems, where we reached 10% less total travel time compared with [9], but with only 4.36% higher number of vehicles.

5 Conclusion

The TDVRP is a modification of VRPTW which attempts to better approximate travel times between customers. In this paper we present speed profile, as a way to model dynamic properties of a real-world traffic network and incorporate them in the calculation of travel times in TDVRP. There were 680, 000 speed profiles calculated and grouped by k-means algorithm into 3, 230 speed profiles used by a modified, time dependent Dijkstra algorithm.

The time dependent Solomon I1 heuristic was implemented as a starting point for the time dependent ILS algorithm. The time-dependent ILS works in two phases. First, an attempt is made to reduce the number of vehicles needed by reducing the total travel time of the solution. In the second phase, total distance is minimized. We approach the TDVRP in two ways. Only one set of standard benchmarks is currently available for the TDVRP and they define coefficients that modify travel time. For a real-world problem presented in subsection 4.1 we have used another approach, more suitable for the given problem. First, in the preprocessing phase, we calculated 199 distance and travel time matrices for every 5 minute interval during daytime. When distance or travel time is needed for some pair of the customers, ILS algorithm looks up those matrices.

Results for standard benchmarks are also given. Since the proposed ILS algorithm has only a basic route reduction ability, it uses 12.12% more vehicles compared to the algorithm published by Figliozzi. On the other hand, travel time is lower by 11.03%. We find that the ILS algorithm is suitable for the presented real-world problem, since the lower bound with regard to capacity is 12 vehicles which was obtained for every one of the 30 runs. That result is 25% lower compared to the actual number of vehicles used for the delivery. Total distance calculated is 219 kilometers, compared to 240 measured in practice.

For future work, adding specialized method to minimize the number of vehicles is planed. Possibility to reduce search space and consequently reduce preprocessing time of matrices will also be considered.

References

[1] Dantzig, G. B.; Ramser, J. H. The Truck Dispatching Problem. *Management Science*, 6(1), 80-91, 1959.

[2] Irnich, S.; Toth, P.; Vigo, D. Chapter 1: The Family of Vehicle Routing Problems. In *Vehicle Routing: Problems, Methods and Applications*, 1-33, 2014.

[3] Kok, A. L.; Hans, E. W.; Schutten, J. M. J. Vehicle routing under time-dependent travel times: The impact of congestion avoidance. *Computers & Operations Research*, 39(5), 910-918, 2012.

[4] Braysy, O.; Gendreau, M. Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. *Transportation Science*, 1(39), 104-118, 2005.

[5] Van Woensel, T.; Kerbache, L.; Peremans, H.; Vandaele, N. Vehicle routing with dynamic travel times:

A queueing approach. *European Journal of Operational Research*, 186(3), 990-1007, 2008.

[6] Malandraki, C. Time Dependent Vehicle Routing Problems: Formulations, Solution Algorithms and Computational Experiments, *PhD thesis*, Northwestern University, Evanston, USA, 1989.

[7] Hashimoto, H.; Yagiura, M.; Ibaraki, T. An iterated local search algorithm for the time-dependent vehicle routing problem with time windows, *Discrete Optimization*, 5, 434-456, 2008.

[8] Donati, A. V.; Montemanni, R.; Casagrande, N.; Rizzoli, A. E.; Gambardella, L. M. Time dependent vehicle routing problem with a multi ant colony system, *European Journal of Operational Research*, 185(3), 1174-1191, 2008.

[9] Figliozzi, M. A. The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics, *Transportation Research Part E*, 48(3), 616-636, 2012.

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

[11] Hoos, H.; Stüzle, T. Stochastic Local Search: Foundations & Applications. *Morgan Kaufmann Publishers*, San Francisco, CA, USA, 2004.

[12] Solomon, M. M. Algorithms for the Vehicle Routing Problem with Time Windows. *Transportation Science*, 29(2), 156-166, 1995.

[13] Erdelić, T.; Vrbančić, S.; Rožić, L. A model of speed profiles for urban road networks using G-means clustering. *Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 1081-1086, 2015.

[14] Geisberger, R.; Sanders, P. Engineering Time-Dependent Many-to-Many Shortest Paths Computation. *10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems, ATMOS*, 74-87, 2010.

[15] Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1), 269-271, 1959.