A Data-thrifty Approach to Routing Optimization

Thomas Fayolle, Renaud de Landtsheer, Gustavo Ospina and Fabian Germeau
CETIC Research Centre, Charleroi, Belgium

Keywords: Local Search, PDPTW, Data, Routing Optimization, OscaR, Cbls.

Abstract: Solving a routing optimisation problem often requires to know the travel times between each pair of points of the problem. Usually, when solving a routing optimisation problem, the travel time is assumed to be constant. However, in real life problems, it can vary a lot due to traffic jams, especially near big cities. Some map data providers can provide accurate travel time estimations, but the data are expensive and querying such a data provider is time consuming. In this paper, we present a method to solve the Pickup and Delivery Problem with Time Windows (PDPTW) that reduces the number of calls to paid data providers while preserving the quality of the solution.

1 INTRODUCTION

Given a set of pickup points and, for each pickup point, an associated delivery point, solving the Pickup and Delivery Problem (PDP) (Toth and Vigo, 2014) consists in finding the shortest path that serves all the pickup and the delivery points. The pickup point and its associated delivery point must be served in that order by the same vehicle. In the Pickup and Delivery Problem with Time Windows (PDPTW), a time window can be associated to each pickup point and each delivery point: the point must be served before the latest time of the time window and it can be reached before the earliest time of the time window, but in that case the vehicle must wait until the earliest time before leaving.

Solving the PDPTW requires to know the travel times between each pair of points. In the classical version of the problem, the travel times are often given as a matrix of constant values representing the travel times between each pair of points. However, in real life problems, travel times may significantly change depending on the time of the day, especially near big cities. This problem is not new and has been defined in the most general form of Vehicle Routing Problem by (Malandraki and Daskin, 1992). This paper and many others that propose solutions for Time Dependent Vehicle Routing Problem variants assume that the time-dependent travel time is known when the optimization starts.

Unfortunately, the data representing the travel throughout the day is a very large histogram. The easiest solution is to buy those data from a map provider, as main map data providers include accurate travel times that take traffic into account. The problem with map providers is that they are financially expensive and computationally slow. For instance, getting a 100x100 data matrix for every 30 minutes of the day costs hundreds of euros and takes more than an hour. On the other side, free data providers (*e.g.* OpenStreetMap (OpenStreetMap, 2004)) combined with source free path finding algorithms (*e.g.* (Luxen and Vetter, 2011) or (GraphHopper, 2018)) can provide travel times without taking traffic into account.

In this paper, we introduce a method that uses approximated data and a local search engine in order to solve a PDPTW while reducing the amount of data that is needed to perform the optimization. The approach is to repeatedly perform optimization on approximate data that are both cheap and fast to obtain. This optimization leads to a route and high-quality data is queried only for the travels belonging to this route. The approximate data is then corrected with a high-quality data that takes traffic into account. The process is started again until the generated route only contains high quality data. The method has been applied to a real life PDPTW problem with real life data. In this example, it was able to significantly reduce the amount of data used while preserving the quality of the solution, the downside being that the optimization –so far– seems much slower with our approach.

Fayolle, T., De Landtsheer, R., Ospina, G. and Germeau, F.
A Data-thrifty Approach to Routing Optimization.
DOI: 10.5220/0010308504330437
In *Proceedings of the 10th International Conference on Operations Research and Enterprise Systems (ICORES 2021)*, pages 433-437
ISBN: 978-989-758-485-5

2 RELATED WORKS

In a survey on vehicle routing problems for city logistics, (Cattaruzza et al., 2017) identify four main challenges in the optimization of vehicle routes in urban area. One of the challenges is the optimization with time-dependent travel times. They conclude that "considering time-dependent travel times is essential when solving VRP in cities" and that it should "become a must-have attribute in the future".

Effectively, (Donati et al., 2008) compute solutions to a PDP using constant travel time on one hand and time-dependent travel time on the other hand. Then they evaluated the solution computed with constant travel time in the context of time dependent travel time. They show that approximating traffic-dependent travel time through constant travel time delivers a solution roughly 8% worse (up to 12%) than if the problem was solved with accurate data.

In their paper, (Sun et al., 2018) solve PDPTW with time-dependent travel times using a branch and price algorithm that give an exact solution. Like many papers that address a VRP with time-dependent travel times, they assume that the travel time function is known at the beginning of the route optimization.

To the best of our knowledge, we do not know any paper that solves a Vehicle Routing Problem while facing the lack of accurate data.

3 OPTIMIZATION WITH (UNDER-)APPROXIMATED DATA

This section presents our approach. Section 3.1 presents the requirements that shall be fulfilled by the method, section 3.2 presents an overview of the method and sections 3.3 and 3.4 discuss the effect of data enrichment methods and stop conditions on the approach.

3.1 Requirements

The method aims at solving a routing optimization problem while respecting two main requirements: (1) the method shall reduce the amount of traffic-aware data compared to an approach requiring the complete information prior to the optimization and (2) the method shall find a solution that is as good as the solution found when knowing all the data.

The method presented in section 3.2 repeatedly uses a routing optimization engine. In the example detailed in section 4, the local search engine OscaR.cbls (OscaR Team, 2012) has been used. Thus, getting the optimal solution is not guaranteed, but requirement (2) means that the search method presented in this paper should not give a worse solution than a solution obtained with the same routing optimization engine and all the needed data.

3.2 Overview of the Method

The whole process of the search method is depicted in Figure 1.

The search starts with an under-approximated time matrix. Such a matrix is obtained using a free map provider and open source path finding algorithm. The travel times obtained using this method do not take traffic data into account. We assume that the real travel time is greater or equal to the travel time computed this way. The method is based on the assumption that the use of an under-approximated time matrix guarantees the requirement (2). The idea behind this assumption is that an under-approximated travel time makes time windows constraints more permissive; any solution of the problem that is acceptable wrt. the strong constraints with a real travel time data shall be accepted with the under-approximated time. One of the purpose of the experimentation presented in section 4 is to test if this claim is reasonable.

The under-approximated time matrix is used in a local search engine to obtain a solution, which is computed using approximated travel time. In order to have a more accurate solution, the travel times of this solution are updated using a high-quality data. After the update, the solution has significantly changed, and may even not be feasible (for example because some time window constraint is not respected anymore). However, updating the solution gives us more information about the travel times. The new information can be used to enrich the knowledge about the travel times. Several solutions for the enrichment process are presented in section 3.3.

After enriching the knowledge, a new search can be launched. Since now we have more data, this search will find either a different solution or a solution with accurate travel times (or more likely an hybrid solution with both different and similar parts with accurate travel times). The new solution is updated using a paid data provider, the knowledge is enriched, and a new search begin.

The process is repeated until the solution obtained from the local search engine is satisfying. The stopping criterion of the search is discussed in section 3.4.

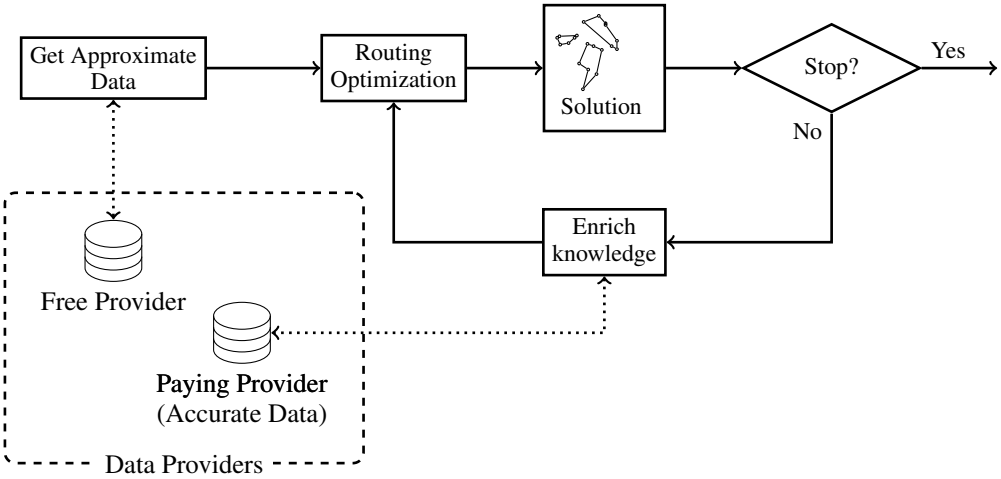

Figure 1: Solving the PDPTW with approximated travel times.

3.3 Enriching the Knowledge

When a solution is obtained after a local search, the data are enriched with a paying provider, in order to have the accurate travel times for that solution. The new data obtained from the paying provider can be used to enrich the knowledge. The most naive idea to enrich the knowledge is to store the data received from the paying provider in order to use the real data the next time the travel time is needed in the same hour. In less naive versions, the data can be used to get a better approximation of the travel time on other points of the problem: we can propose landmark-based and learning-based approaches.

Algorithms like landmarks (Goldberg and Harrelson, 2003) can further improve the quality of the approximate data by performing data extrapolations. It uses the triangle inequality to deduce sound under-approximate travel time out of known high-quality travel times. This idea is that given the distance from a landmark L to point A and a point B, we know that $(A \to B) \geq (L \to A) - (L \to B)$ so $(L \to A) - (L \to B)$ is an under approximation of $A \to B$. Since the landmark generates under-approximated values, they should preserve the quality of the solution. On the other side, the under-approximation may be really low (0 if L is equidistant from A and B)

Data learning methods could be used too to get a better approximation than the travel time without traffic data. Since the search is started with a better approximation, it should take a smaller number of iterations and thus less time to arrive to a solution that is computed with real travel time, but learning methods do not guarantee to provide sound under-approximated values and the impact on the quality of the solution should be leveraged.

3.4 Stop Condition

The stop condition can affect the computation time and the quality of the solution. The strongest stop condition will be to verify that the solution only relies on exact travel time data. Weaker conditions could be used, like checking that the solution is still feasible after updating the travel times. As usual when using meta-heuristics, there is a trade-off between computation time and solution quality. This trade-off usually depends on the problem specific situation where the tool has to be implemented.

4 PRELIMINARY RESULTS

In order to verify the claim that the method presented in section 3 fulfills the requirements introduced in the same section, it has been applied on a PDPTW problem. In this section, we present the problem on which it has been applied and the preliminary results obtained.

4.1 Benchmark Settings

The PDPTW on which the method has been applied contains real GPS positions around Brussels, Belgium. It is a shared cab problem, which has about 300 nodes within 98 different geographic locations. Among the 300 nodes, 14 are vehicle depots; each vehicle has a capacity that should not be exceeded. Since it is a pickup and delivery problem, the pickup node and the delivery shall be served in that order (the pickup node before the delivery node) and shall be served by the same vehicle.

	Statistics (mean on 10 runs)				
	Objective	Time[1]	Iteration	Free Provider calls	Paying Provider calls
Entire Matrix	104 051 534	5'02"	-	9 604	451 388
Request when needed	104 429 079	3'28"	-	9 604	18 483
Search with approximation	101 848 245	49'12"	10,9	9 604	662

[1] Since the API is simulated, this time does not take into account the time required to get the data from the map provider

Figure 2: Benchmark results.

A time window is associated to each node. The latest time of the time window is a strong constraint: the solution cannot be accepted if the node is served after the latest time of the time window. The earliest time of the time window is not a constraint that can be used to reject a solution. However, if a vehicle reaches a node before the earliest time of the time window, it shall not leave the node before the earliest time of the time window. The objective function is the length of the route (in meters) plus a penalty for the unrouted nodes.

4.2 The Results

The main purpose of the example described here is to verify if the method presented in section 3 reduces the number of calls to a paid data provider and does not deteriorate the value of the objective function. A real travel time function matrix has been acquired to a paid map provider. For each pair of points in the 98 points of the problem, the travel time is asked for each 30 minutes in the clock of the day (this means we have 98*98*48 = 460 992 travel times). As a cost and computation time indication, getting the travel times costed about 200€ and took about 20 hours. The use of asynchronous calls to the API could reduce this time to a time between 1 or 2 hours.

With this data, we simulated the map provider API. The number of calls to the simulated provider is recorded during the search. The search with approximation is compared with two different methods. In the first method, the entire data matrix is asked to the provider. The problem is solved using the entire matrix. In the second method, the map provider is called each time a travel time is needed. The search with approximation is used without a smart enrichment method: when the travel time is requested to the map provider, it is stored; in further local searches, the real travel time for a given departure time is used if it is already known and the travel time at midnight is used otherwise. The stop condition is strong: the solution is considered satisfying if the arrival times at each point of the solution are exactly the same before getting the real times and after getting the real times. In other words, the solution is considered as satisfying if it has been calculated only with real travel times. For each method, the solution is found using the Very Large-Scale Neighborhood (VLSN) (Mouthuy et al., 2011) implemented in the local search framework OscaR.cbls.

The results are shown in the table in Figure 2. The figures in this table are the average figures of 10 runs. The first column contains the value of the objective function for each method. The value of the objective function is almost identical for the first two methods, and is a bit better with the search with approximation. As local search engines may fall into a local minimum, one of the easiest meta-heuristics used to escape local minima is to restart the search after perturbing the solution. This is what is (involuntarily) done in the search with approximation, and it could explain why the objective is better.

The second column contains the run time for every method. The third column contains the number of time the loop of Figure 1 is run. The search with approximation takes significantly more time than the two other methods. It is explained by the fact that it requires more than 10 iterations of local search on average where the two other methods only require one local search. However, the API of the map provider is simulated, and it does not consider the time required to request and receive the data from the map provider.

The last column contains the number of API calls. The calls are separated between the calls for the travel time at midnight and the calls for the travel time at other time of the day. The calls at midnight are considered as free: they can be acquired from a free data provider. This number is identical, no matter the method. But the number of calls to a paid provider is significantly reduced when using the search with approximation: only 3.58% of the calls needed for the second method are needed for the search with approximation. On the provider from which we got the data for this benchmark, it represents a cost of about 0.25€, instead of about 6.7€ for the second method and about 200€ for the first method.

5 CONCLUSION AND FUTURE WORK

In this paper, we presented a method to solve a PDPTW with approximated data that reduces the need for accurate data from a paid provider. The application of the method on an example shows that it allows to significantly reduce the need for data and that the solution obtained is not worse than a solution obtained with a full access to the data, even though this result should be consolidated by giving the same search time to both method.

The development of the method is an ongoing work and a there is more to be done. Firstly, the run time is quite high. This is due to the number of iterations that are needed to obtain a solution computed with only exact travel times. In the example presented in this paper, when the travel time is not known, it is approximated by the time without traffic information, which can be a bad approximation. Using a better approximation should reduce the number of iterations. Approximation using landmarks and learning methods are mentioned in the paper and should be experimented.

Secondly, the method has been tested with a simulated API which makes the presented example rather theoretical. The access to the data is immediate and there is no network access problem. The method should be tested with a real API in order to leverage the impact of getting data from a real web service.

Finally, the results presented here have been acquired with one example and one data set. This data set is a particular set of points and does not allow to compare the results with state-of-the-art results. More examples should be tested in order to validate the conclusions of the paper.

ACKNOWLEDGEMENTS

This research was supported by the SAMOBIGrow CWALITY research project from the Walloon Region of Belgium (nr. 1910032).

REFERENCES

Cattaruzza, D., Absi, N., Feillet, D., and González-Feliu, J. (2017). Vehicle routing problems for city logistics. *EURO Journal on Transportation and Logistics*, 6(1):51 – 79.

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

Goldberg, A. and Harrelson, C. (2003). Computing the shortest path: A* search meets graph theory. *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*.

GraphHopper (2018). https://www.graphhopper.com.

Luxen, D. and Vetter, C. (2011). Real-time routing with OpenStreetMap data. In *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, pages 513–516.

Malandraki, C. and Daskin, M. S. (1992). Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science*, 26(3):185–200.

Mouthuy, S., Hentenryck, P. V., and Deville, Y. (2011). Constraint-based very large-scale neighborhood search. *Constraints*, 17:87–122.

OpenStreetMap (2004). https://www.openstreetmap.org.

OscaR Team (2012). OscaR : Operational Research in Scala. Available under the LGPL licence from https://bitbucket.org/oscarlib/oscar.

Sun, P., Veelenturf, L. P., Hewitt, M., and Van Woensel, T. (2018). The time-dependent pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 116:1 – 24.

Toth, P. and Vigo, D., editors (2014). *Vehicle Routing*, volume 18 of *MOS-SIAM Series on Optimization*. SIAM.