TODaaS: Transport On-Demand as a Service

João Pinheiro
Instituto Superior Técnico
Lisbon, Portugal
joao_pinheiro_@live.com.pt

ABSTRACT

The increasingly competitive market of mobility services in cities, together with European regulations are motivating Public Transport providers to offer Demand-Responsive Transport Services. In this work we describe this kind of mobility solution, comparing it with other known solutions. Then, we make a literature review on similar projects, and algorithmic approaches to accomplish a service like this. Finally, we present a software solution for a demand-responsive public transport service, with its requirements validations, and a small study on its routing algorithm, making use of some literature examples.

Author Keywords

Demand-Responsive Transport; Mobility; Software; VRP; DARP; Genetic Algorithms.

INTRODUCTION

The present thesis refers to a project to build a software solution for transport providers, in order to provide a demand-responsive transport (DRT) service. It has the main purpose to allow existent providers to offer this new type of service.

DRT is a subgroup of Public Transport, where routes and schedules are flexible to users' needs.

This kind of transport is commonly used in areas with lower demand, which do not justify a regular service. In higher demand areas, it is an emergent solution to provide a more flexible service at a fairly low cost to users (less than driver services, like electronic hailing or taxis), while reducing operative costs for transport service providers as well.

Providers have a fleet of vehicles, operating in a defined geographic area. The vehicles are used to fulfill requests within this operating area. Then, a passenger should be able to request a trip from a departure point, to an arrival point. They should specify the time window they want to depart, and the time window they want to arrive. After submitting and confirming, they should be able to check the status of their trip: the vehicle and driver attribution, and the estimated route (with the estimated time to depart and to arrive).

A passenger should always make the desired trip in a single vehicle route, with no connections.

Objectives and Scope

Due to its academic nature, the work will only regard the back-end part of the solution. User interfaces, ticketing and payments will not be addressed. They are complex by themselves, and only a professional team with more man-hour resources could accomplish such a work with these many features and requirements.

The main objectives to accomplish in this thesis are:

- Research and Analyze previous and relevant work done in demand-responsive road transport services
- Develop and test the modules to accomplish the service features
- Evaluate the performance of the service against current methods
- Use the service for small studies in the public transport field

The work started from an initial contact with a company, from which it was not possible to define a concrete business model for the service. As such, there is the intention of building it to be adaptable to many different providers, which might have different business models.

Motivations for this work come from two main reasons: (1) the advantages of demand-responsiveness, and (2) the need of adapting to a more competitive market, as new regulations make public transport concessions shorter, and new technology-backed providers, like Uber and Lyft are penetrating the mobility market.

Demand-responsiveness allows providers to optimize their resources to the current needs, avoiding empty vehicles most of the time. It replaces exigent, expensive, and slow demand studies.

New European regulations for concession contracts [1] aims to increase market competitiveness, which affects public transport provides. They now face smaller companies' competition.

As such, providers can benefit from DRT services, as a smart strategy to innovate their business.

STATE OF ART

This work is ultimately a project, fact that emphasizes the need of two types of analysis: one in a conceptual side, and one in a technical side. For the conceptual parts, we have investigated the main concepts and how they were applied in other projects. Then, another investigation is centered on the algorithmic techniques and approaches to the problem. Finally, we studied applications and properties of genetic algorithms, like operators, representations and evaluation functions.

Mobility Solutions

Urban mobility demands are the most frequent mobility demands. With the growing numbers of people, comes the growing of urban mobility demands. As changing infrastructure or the space shape, to reduce congestion and distances, is a very expensive investment, the need of mobility solutions rises as an alternative solution to problems like traffic, pollution, excessive budgets on transport services, or the lack of quality of those.

Mainly, the mobility solutions explored for this work range from the cheap, yet inflexible regular public transportation services (regular bus, subway and tram, etc.), to the expensive and flexible driver services (as taxis or electronic hailing platforms, like Uber). Some of these driver services are trying to offer shared rides in order to lower the price - carpooling solutions. Despite having similar advantages to demand-responsive transport, carpooling is not the same. While in carpooling passengers are matched with drivers beforehand, a demand-responsive public transport must offer the possibility for passengers to request a trip with immediate effect. This requires sophisticated algorithms to optimize matches for faster and shorter routes. Also, public transport services are linked to public authorities, which ensure service quality, while driver services are not, being the driver the main responsible for the quality of the service.

On-Demand Services

There are already some projects for DRT platforms, and even some up and running services. In this section those will be described and discussed.

One of these is Kutsuplus [2], a pilot project ran in Helsinki, Finland, led by a partnership between Split Finland Ltd. (earlier Ajelo Ltd.), and the Helsinki Regional Transport Authority (HSL), from 2012 to 2015.

Kutsuplus passengers could use a mobile application to request trips between so-called virtual stops - specific points designated for passengers' pickup and delivery.

By not picking up passengers from custom locations - fixed virtual stops only – it was possible to minimize error in time estimations. This allows a major reduction in the possible paths for algorithms to optimize, and to have a fixed map, where only the times of travel between stops changes. The final report estimates that major savings were achieved, due to the more efficient use of the vehicles. These savings are, however, result of an enormous initial investment. Environmental impacts, along with profitability, are only achieved when the service is scaled up to the hundreds of vehicles, following the same report.

Split Finland ran another DRT solution in Washington D.C., leading to a very competitive environment, and then to a saturated market in the hailing business. As E-Hailing platforms (Uber and Lyft) applied discounts to their trips, Split lost its competitive premise (cheaper than hailing, more flexible than regular public transport), and discontinued activity.

MOIA [3], a company from the Volkswagen Group, enters the stage here, by buying Split Finland. Their vision is to provide ride services that will reduce the number of private cars in cities, reducing traffic, accident rates and pollution. All of this, while providing a flexible mobility solution.

Their business model is based in comfort of the ride, with a innovative vehicle design, aiming to add more personal space to each of the six seats. The vehicles are electric-powered, as another innovation towards environmental sustainability.

Finally, Via [4] is a company that offers the technology for shared rides services to transport providers. They partner with these providers by licensing their platform - a software solution, just like the one we built in this work.

This helps them to innovate onto more flexible solutions, without the big expense of building the technology themselves.

In summary, Kutsuplus was a pioneer in DRT services, and its evolution through time gives good insights and experience. MOIA is the result of this trial and error process. And Via has a similar software solution to the one we built in this thesis.

VRP

The Vehicle Routing Problem (VRP) [5], is the chosen representation for the problem our algorithm aims to solve, in its more specific variation of the Dial-a-ride problem (DARP) [6], being also an Open VRP (OVRP) [7], meaning that the vehicles do not have to return to a central depot after concluding a route, remaining in the streets serving requests through all their availability. This is a complex problem (NP-Hard), where one wants to obtain the most optimized solution possible, in the given time. Obtaining solutions for these problems is trivial. But we are only interested in Feasible solutions – solutions that respect all the problem constraints. In the DARP case, all passengers must be picked up and delivered within the defined time windows for the effect, the vehicles' capacity cannot be exceeded, both pickup and delivery points of a request must be in the same vehicle route, and the pickup points must be visited before the respective delivery point.

Several solution methods were investigated for this type of problem, mainly Exact Methods, 2-Phase Algorithms, Heuristics, Constructive Methods, and Metaheuristics; being the last ones the most interesting in terms of performance. We then chose to build our software solution using a Genetic Algorithm, as they offer a good variety of operators and representations to adapt to the details of every problem.

Genetic Algorithms

As for our Genetic Algorithm, we have investigated some different representations for our DNA – a sequence that

translates to a final solution for the problem – like a binary matrix, a sequence of indices, or a sequence of vehicle-request pairs. We chose the last one, due to its easiness of operators' application (we guarantee all points of each request are given to the same vehicle - one of the feasibility constraints).

As for genetic operators, we chose the Partially Matched Crossover (PMX) [8], a Single Gene Swap Mutation [9], and a Rank-Based Roulette Wheel Selection [10]. Our evaluation function (fitness) is based on the feasibility quality of a solution (measured from 0 to 1), plus the fitness value (which is given by $\frac{1}{1 + total\_distance}$). Total distance is the total covered distance by all vehicles in the given solution.

In order to reduce the search space, it is suggested [11] to distribute requests by the fleet of vehicles, and then perform a local search to optimize each route as a TSP. This way, the solution space is reduced to only the possible combinations of request attributions to the vehicles, and order does not matter anymore for the global search representation, being only relevant in the local searches.

SOLUTION PROPOSAL AND ARCHITECTURE

The main service features are: (a) providing automatic routes calculation for incoming passenger requests (for providers), and (b) submitting a passenger request, and checking its status (for passengers). The service aims to connect passengers who need to get from A to B, to providers that can perform such a trip.

In order to better adapt to different business models from different providers, we divided the solution into four main modules:

1. Requests
2. Resources
3. Maps
4. Algorithm

The architecture is visible in Figure 1.

All the modules were implemented using the C# language, and the .NET Core Framework. The two Web API modules make use of SQL Server databases to persist data.

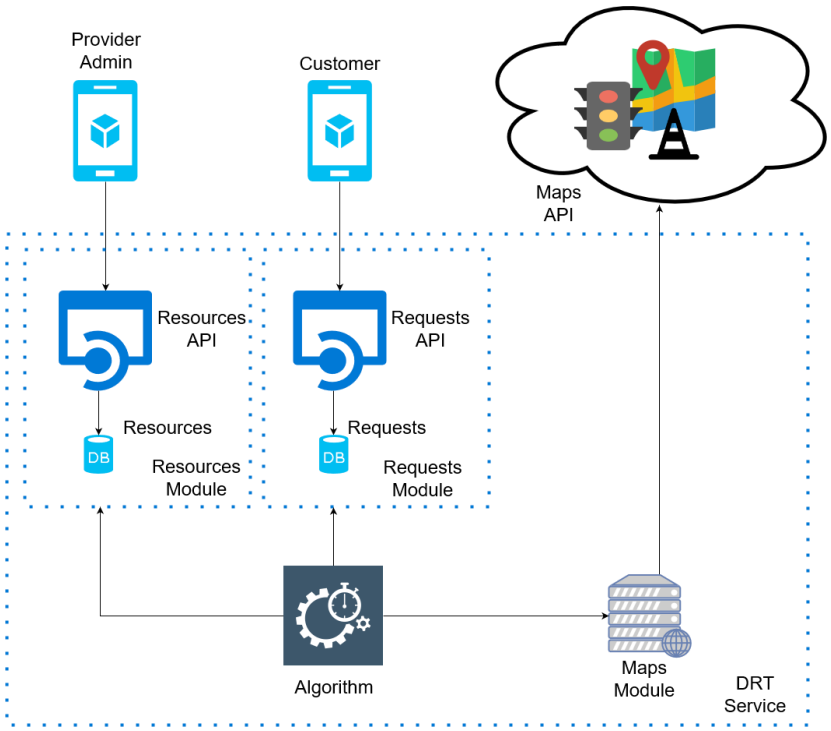

Figure 1. The DRT Service Architecture Design.

Before a provider can start fulfilling requests, they should use the Resources module (a Web API) to input and update their data, such as the area(s) of activity, the fleet, the drivers, and the schedule/availability of these. To do this, the provider should use their back-office interface (shown by the Provider Admin application in Figure 1).

A provider can have multiple, non-overlapping activity areas, and multiple providers can be active in the same geographic area. These areas are modelled and validated, using an Oblate Spheroid representation of the Earth, and intersections are calculated by solving the Vincenty's Inverse Problem [12]. For distances shorter than 10Km, we use a flat approach, as the error margin does not justify heavier calculations. We also use the Ray-casting algorithm

[13] to check if a point is inside an area (represented by a polygon).

Additionally, providers should inform the resources module of the vehicles' position regularly (either from their back-office, or from an external system that calls our Resources module). This information will be relevant for the algorithm.

Then, passengers of that provider can submit trip requests to the Requests module (another Web API). A trip request must include the following information elements:

- Pickup point (latitude and longitude coordinates)
- Pickup time window (two dates, for minimum and maximum pickup)
- Delivery point (latitude and longitude coordinates)
- Delivery time window (two dates, for minimum and maximum pickup)
- Number of passengers (for capacity and occupation calculations, defaults to 1)

Periodically, a job runs, verifying if there are any unserved requests. If there are, it gets the available provider fleet from the Resources module, then makes use of the Maps module to create a graph – an internal representation of the world for routing, and finally uses the Algorithm module, passing the last two as inputs, together with the requests. The result of the algorithm is a Solution – a set of routes to the available vehicles fleet – to satisfy the current requests.

This behavior is illustrated in Figure 2.

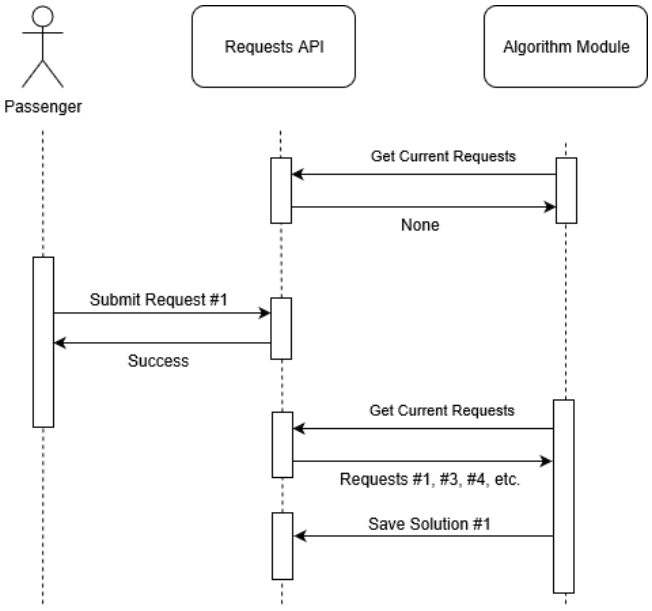

Figure 2. Flow of storing a request, and routes calculation.

The job executes an instance of this process per each provider area, in separate worker tasks.

Requests are Unprocessed when they are stored. Then, when a passenger is presented with a route proposal (the attributed route from the calculated solution), they should either Confirm, or Cancel. If they Confirm, the request Starts when the vehicle reaches the pickup point (and the passenger enters the vehicle), and finally Finishes when the vehicle drops the passenger at its delivery point. If a passenger does not Finish the trip or is not present when the vehicle reaches the pickup point, the request is considered Abandoned. Passengers can provide a reason for abandonment, as feedback is important for providers to offer the best quality of service possible.

Figure 3 shows the possible request states and transitions. Thicker borders indicate Final States.

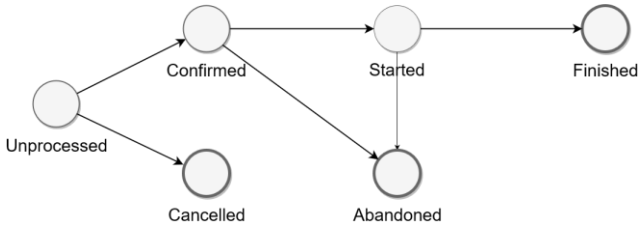

Figure 3. Diagram of a Request Lifecycle.

Passengers are also able to check the status of their request, being presented with their route, from the current best solution. This route can change over time, if new solutions are calculated (with different results), and changes do not impact current requests' constraints.

When a passenger finishes a request, they can Rate it, providing an evaluation to the vehicle, to the driver, and to the overall trip experience. This will be an important feature for providers, as a good way of obtaining feedback.

Reducing Trip Stops

In order to broad the offer of possible business model a provider can adopt, it is possible for a provider to either provide a threshold distance, for stops clustering, or a set of fixed stops (just like Kutsuplus has done).

If they adopt a fixed stops approach, the passenger requests' points must coincide exactly with these stops.

Otherwise, the threshold distance will be used as the maximum cluster size on a Hierarchical Clustering [14] process, that condenses close enough points into a single stop. This allows providers to save precious travel time, and reduce passengers' waiting times too.

After defining the overall architecture and features, let's now jump into each module in more detail.

Requests

This API has endpoints to Create, Update, and Retrieve (both individually and in filtered collections) models. Creations are made through the HTTP POST verb, Updates through the PUT verb, and retrievals through the GET verb.

There are no deletions, as it is not possible to delete a request once it is made. It is only possible to cancel or

abandon it. Solutions are managed internally by the job, and can never be manipulated by users, only consulted.

Figure 4 illustrates this module's models and their relationships.

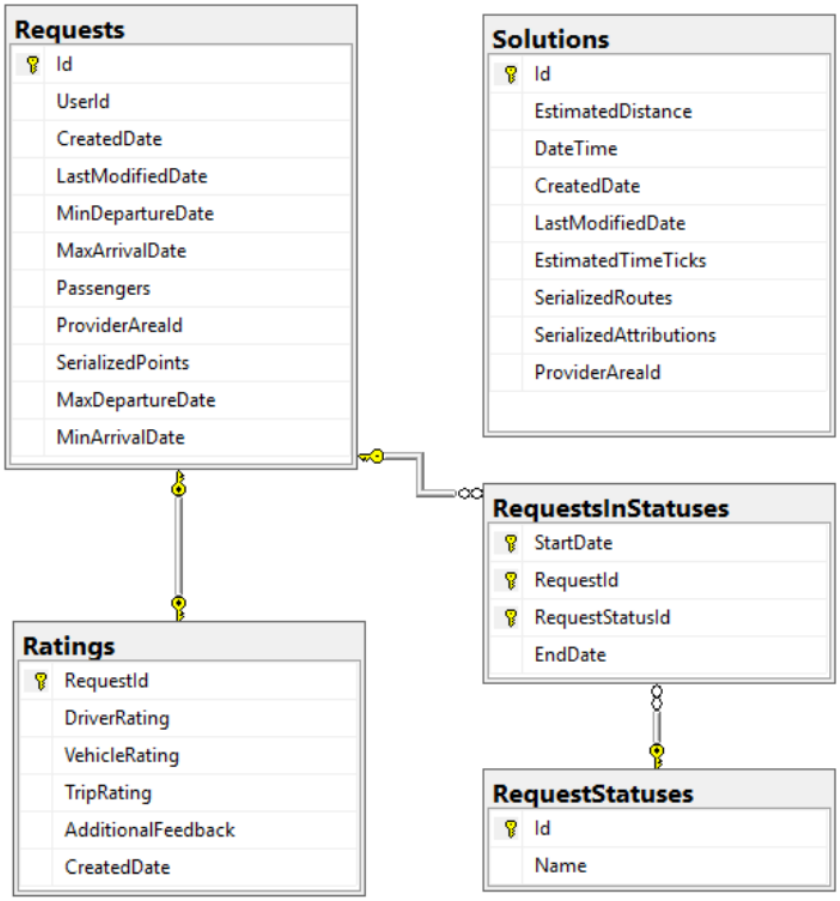

Figure 4. The Requests Module Models Schema.

Resources

A Web API made to store and manage providers' information. This information is composed by provider areas, vehicles, drivers, availabilities (an association between a vehicle and a driver, for a defined period of time), and vehicle tracking points (periodically sent positions of the vehicle over time).

The models and their relationships are shown in Figure 5.

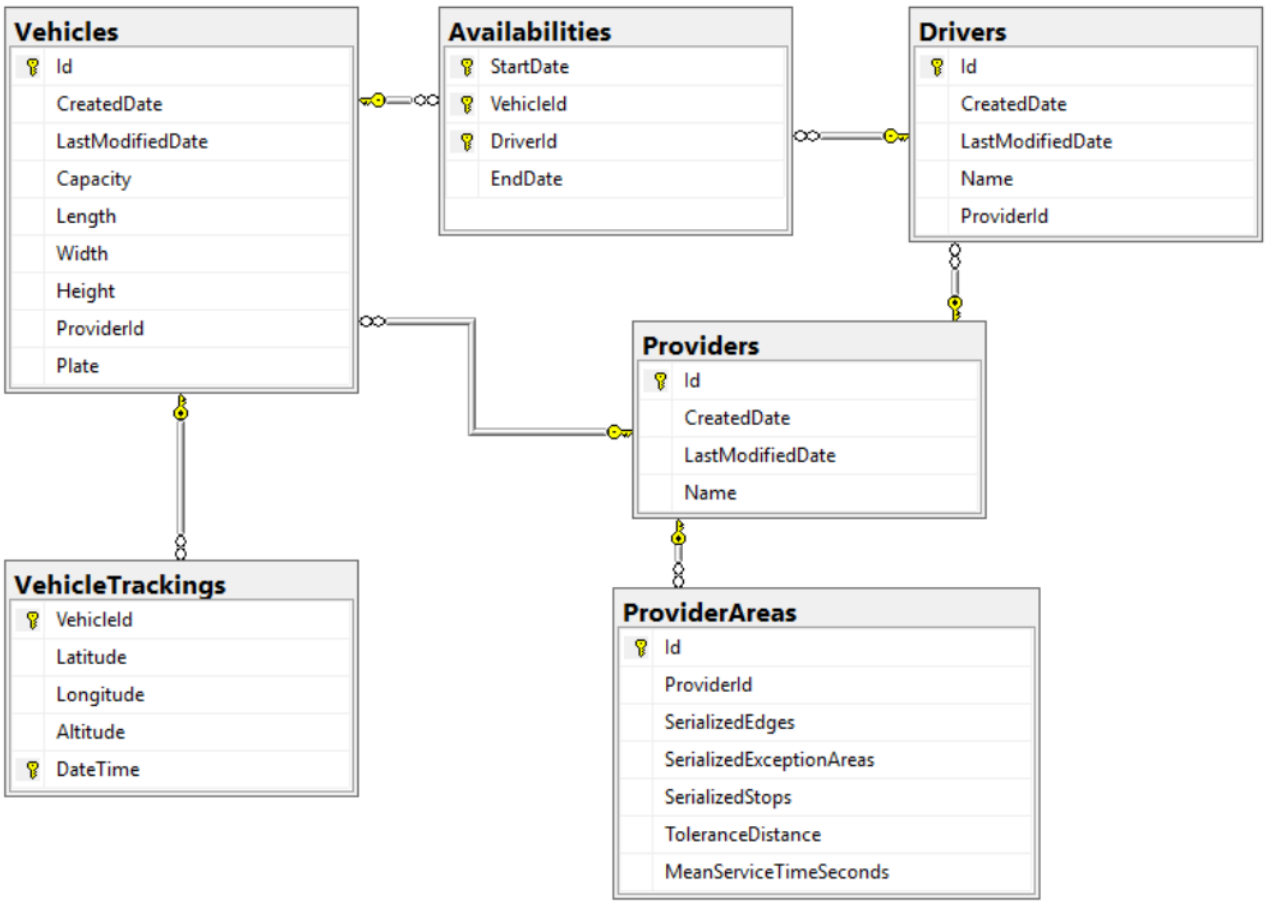

Figure 4. The Resources Module Models Schema.

Maps

This is the simplest of all modules, containing only two connectors to two external maps API services: HERE [15], and Openrouteservice [16]. After trying both, we chose HERE, as it offers better time estimates between points than Openrouteservice.

Algorithm

As stated before, this module contains only a Genetic Algorithm to solve a DARP. Yet, it is the most complex module of all, being rich in many different genetic operators tried, and containing also another Genetic Algorithm to solve local TSP instances. The combination of the DARP and TSP solvers are able to reduce drastically the solution space, and to reach better and feasible solutions faster than our previous algorithm. The chosen genetic operators are mentioned in the State of Art, being the same (adapted to the different problem representations) for both solvers. The local TSP is represented by a sequence of stop indices, while the DARP has a set (unordered) of pairs vehicle-request, just to distribute requests to the local TSP instances.

Users, Authentication and Authorization

There are two types of users in our solution: passengers, and administrators. As the names suggest, passengers can only use the Requests API, properly authenticated and authorized via JSON Web Tokens (JWT). Administrators can use both API modules, being also able to manage all providers' information. The authentication and authorization mechanisms are the same for both API modules (only authenticated users can perform requests, except for registering, and some requests can only be made by an administrator user). All requests are sent via the HTTPS protocol.

EVALUATION

In order to evaluate this work, we have divided it into two main groups: (1) the amount and quality of features (making use of module and integrated tests), and (2) service behavior studies, making use of some well-known test instances (Cordeau and Laporte (2003) [17] DARP instances, Christofides (1979) [18], and Fisher (1994) [19] OVRP instances). We also created a few instances of our own, from a small subset of the Lisbon Metro trips. The studies compared the service performance over different problems (DARP vs. OVRP), and the performance of our service against the Lisbon Metro.

For the first group, we can observable that a lot of the work done in this thesis is centered in offering providers a broad range of solutions, and all of the features desired for such a service are implemented. To be noted the level of detail on entities validations and definition, which is high, mainly when saving a provider area, or a trip request.

As for the behavioral studies section, we mainly have two observations: the service behaves as expected (we being able to observe the same behavior as in the reviewed literature), but it does not fulfill the demands of a real world use case (it still needs algorithmic improvement, the results are worse than the best known solutions). A good observation is that the algorithm performs at its best when facing a DARP problem, meaning it is well built for a passenger transport service scenario, just needs some more work on performance.

This service might be very useful for academic research in the transportation science field, and the architecture is built to easily adapt to different kinds of business (either by using a different navigation system, or by using a different algorithm for a different business model, with different constraints).

New studies could be done on the applicability of such a service to replace the Lisbon night buses, which work in fixed routes, just like daytime regular service. Such a study would have to be backed with information on the current usage of these buses.

CONCLUSIONS AND FUTURE WORK

Let's now resume all the work done, the learning involved, the conclusions reached, and how this work should continue improving in the future.

We think a very flexible architecture was achieved, making it possible in the future to adapt the service to different providers, or even to new types of service. This is a good asset for such a project, where the business model is not tightly defined.

The results obtained in the Evaluation, when comparing our service to the Lisbon Metro, lead us to the conclusion that such a service should focus their activity in less accessible zones, or in less crowded periods of the day. This way, operation costs of a regular service would benefit from the savings and be appealing enough for passengers.

Yet, this is not a final conclusion. More studies, with a better algorithmic solution could improve a lot the results, a very interesting work to be done in the future.

The problem constraints study has the main conclusion that different providers should have different algorithms, specially tuned for their business logic.

As for future work, we have three main groups of improvements:

- New features to be developed, or to improve even more the service architecture
- Algorithmic Improvements
- New Studies, to assert the possibility of implementing of such a service and how to do it

New Features

The new features are some security improvements, like enabling two-factor authentication, the possibility to link the user account with external logins (like Microsoft, Google or Facebook accounts), being able to login using these accounts; or adding a "Forgot my credentials" feature.

We also had some ideas on improving the service jobs, by having one that cancels unprocessed requests that don't get confirmed nor cancelled after a defined timeout.

Thirdly, it surged the idea of having a new endpoint on the Requests API, that allows a user to compare the estimated routes from the algorithm solutions, with the actual path made by the vehicle.

Then, we thought of, when storing a new request, running a lighter algorithm that takes the current vehicle routes, and tries to fit the new request in any of them, without breaking any constraints. If successful, the heavy algorithm does not have to run so often, raising the service adaptability to new requests, and users can preview their trips faster.

The next phases of this project should now also include user interfaces, in order to allow tests with real users.

Algorithm

On the algorithmic improvements, we propose Best-Cost Route Crossover (BCRC), proposed by Ombuki et. Al [20]. From the reviewed literature, we also suggest exploring Tabu Search, and a Gravitational Search Algorithm, as proposed by Hosseinabadi et. Al [21]. These new methods could be interesting for comparative studies, and possibly be combined for different providers, if futures studies conclude that none of them outperforms another for all situations.

New Studies

The first is in order to assert the possibility of using a service like this to replace the Lisbon night buses (or at least some of them). The goal would be to maximize vehicles' usage and minimize traveled distances. If possible, would be interesting that the service could also reduce passengers' waiting times.

Another interesting study would be to use a broader portion of the Lisbon metro data set, and check if the determined vehicle routes converge to the metro lines courses. This would assert the quality of their distribution on the city. There is a popular idea that their planning is not so good, and such a study could help with the discussion.

Yet another proposed study is to use Lisboa Aberta data sets [22], a municipal open data initiative, to generate real-world test scenarios, using the pedestrian densities (normalized) as probabilities of getting a request for that

location. The pedestrian densities are visible in the Potential Pedestrian Map (Mapa de Potencial Pedonal) [23].

ACKNOWLEDGMENTS

I would like to thank Professor Alberto Cunha for the support and guidance.

Also, to my family, friends, colleagues, and beloved Timmy, who were always there when times were rough, my best thanks.

Finally, thanks to Enlightenment.AI, especially to Miguel Marques, and Manuel Levi, whom experience added valuable ideas and insights.

To each and every one of you - Thank you, and best regards.

REFERENCES

1. Directive 2014/23/EU of the European Parliament and of the Council of 26 February 2014 on the award of concession contracts (OJ L 94, 26.2.2014, p. 1-64)
2. Rissanen, K.: Kutsuplus - Final Report. Helsinki Regional Transport Authority, Helsinki (2016)
3. MOIA, https://www.moia.io/. Last accessed 10 May 2019
4. Via, https://platform.ridewithvia.com/. Last accessed: 10 May 2019
5. Dantzig, G. B., Ramser, J. H.: The Truck Dispatching Problem. Management Science **6** (1), 80-91 (1959)
6. Savelsbergh, M. W. P., Sol, M.: The General Pickup and Delivery Problem. Transportation Science **29** (1), 17-29 (1995)
7. Li, F., Golden, B., Wasil, E.: The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. Computers & Operations Research **34** (10), 2918-2930, (2007)
8. Baldacci R., Battarra M., Vigo D.: Routing a Heterogeneous Fleet of Vehicles. In: Golden B., Raghavan S., Wasil E. (eds) The Vehicle Routing Problem: Latest Advances and New Challenges. Operations Research/Computer Science Interfaces, vol 43, pp. 3-27 Springer, Boston, MA (2008)
9. Potvin, J.-Y., Bengio, S.: The Vehicle Routing Problem with Time Windows - Part II: Genetic Search. INFORMS Journal on Computing **8**, 165-172 (1996)
10. Kumar, R.: Blending Roulette Wheel Selection & Rank Selection in Genetic Algorithms. International Journal of Machine Learning and Computing, **2** (4), 365-270 (2012)
11. Hamzaçebi, C.: Improving genetic algorithms' performance by local search for continuous function optimization. Applied Mathematics and Computation **196**(1), 309-317 (2008)
12. Vincenty, T.: Direct and Inverse Solutions of Geodesics on the Ellipsoid with Application of Nested Equations. Survey Review. **23** (176), 88-93 (1975)
13. Ray-casting algorithm | Rosetta Code, https://rosettacode.org/wiki/Ray-casting_algorithm. Last accessed 10 May 2019
14. Wikipedia | Hierarchical Clustering, https://en.wikipedia.org/wiki/Hierarchical_clustering. Last accessed: 10 May 2019
15. Routing API | HERE, https://developer.here.com/documentation/routing/topics/request-matrix-of-routes.html. Last accessed 10 May 2019
16. Matrix | Openrouteservice API, https://openrouteservice.org/dev/#/api-docs/matrix/. Last accessed 10 May 2019
17. Cordeau, J.-F., G. Laporte.: A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transportation Research Part B: Methodological **37** (6), 579-594 (2003)
18. Christofides, N., Mingozzi, A., Toth, P.: The vehicle routing problem. In: Combinatorial Optimization, pp. 315-338. Wiley, Chichester, UK (1979)
19. Fisher, M. L.: Optimal Solution of Vehicle Routing Problems Using Minimum K-trees. Operations Research **42** (4), 626-642 (1994)
20. Ombuki, B., Ross, B., Hanshar, F. T.: Multi-Objective Genetic Algorithms for Vehicle Routing Problem with Time Windows. Applied Intelligence **24** (1), 17-30 (2006)
21. Hosseinabadi, A.A.R., Kardgar, M., Shojafar, M., Shamshirband, S., Abraham A.: Gravitational Search Algorithm to Solve Open Vehicle Routing Problem. In: Snášel, V., Abraham, A., Krömer, P., Pant, M., Muda, A. (eds.) Innovations in Bio-Inspired Computing and Applications. Advances in Intelligent Systems and Computing, vol. 424, pp 93-103. Springer, Cham (2015).
22. Conjuntos de Dadoa | Lisboa Aberta, http://lisboaaberta.cm-lisboa.pt/index.php/pt/dados/conjuntos-de-dados. Last accessed: 10 May 2019
23. MAPPe - Mapa de Potencial Pedonal | Plano de Acessibilidade Pedonal, http://planoepap.maps.arcgis.com/apps/webappviewer/index.html?id=069c4784a209443ca98ede37e9fb5867. Last accessed: 10 May 2019