Dynamic two-level optimization for ride-sourcing vehicle dispatch

Minru Wang*[1] and Nikolas Geroliminis[2]

[1]PhD Candidate, LUTS, Ecole Polytechnique Fédérale de Lausanne, Switzerland
[2]Full Professor, LUTS, Ecole Polytechnique Fédérale de Lausanne, Switzerland

Short summary

While a ride-sourcing platform can dynamically dispatch a nearby driver to serve the nearest request, in the long-term, it is valuable to perform batch matching and additionally leverage spatial and temporal demand knowledge. We propose a two-level framework that incorporates future spatial demand and supply information to improve performance. The upper level predicts the aggregated demand and supply in the near future, and solves a flow maximization problem to optimize request and vehicle flows. The lower level solves an Integer Quadratic Programming problem to minimize the total cost of vehicle dispatch decisions at a shorter interval, where deviations from upper layer flow targets penalized in the objective function. This paper reports performances with varying upper-level prediction horizons and lower-level deviation penalties. Results demonstrate that during high-demand periods, the proposed two-level optimization consistently improves the service level compared to a dispatch policy of matching with the nearest vehicle.
Keywords: Ride-sourcing, shared mobility, optimization, simulation, maximum flow problem

1 Introduction

In the operation of ride-sourcing platforms, efficient vehicle resource allocation in a dynamic setting is a well-known challenge. While dynamic ride-sourcing matching problems can be solved as a Dial-a-Ride Problem with some success in works such as Cordeau & Laporte (2007), Simonetto et al. (2019), and Alonso-Mora et al. (2017), the mathematical models are typically formulated as bi-partite matching problems that are time-consuming or require heuristic algorithms to solve. Secondly, when ride-pooling is concerned where two passengers may share part of their journey together, it is common to reoptimize existing solutions to obtain marginal solution improvement. Instead of rashly reoptimizing matches over all requests in a large network, a strategic approach is to incorporate knowledge on the spatial demand and supply to make informed decisions while reducing the problem scope. For example, Pelzer et al. (2015) and Yao & Bekhor (2021) both show that appropriate partitioning methods can be used to preprocess potential matches, hence improving matching performance.

To devise a dynamic matching framework for serving pool and solo trips, we propose an optimization approach that monitors spatial demand and supply evolution, and decompose the challenging task of dynamically matching solo and pool rides into a 2-level optimization problem.

The upper level is formulated as a resource allocation problem and takes inspiration from studies that examine the shareability of specific request pairs (Santi et al., 2014). We extend existing methods to consider the shareability at the granularity of regions where requests are located in. We formulate a classical maximum flow problem (Ahuja et al., 2001; Hillier & Lieberman, 2001) to precompute the shareability for trips originating in different clusters of the network. Then, given the number of available vehicles in each region, the proposed model matches the requests with the available vehicles without always assigning vehicles from the same region or the nearest vehicle from request origins.

As for the lower level, we implement an optimization model that is motivated by the literature (Alonso-Mora et al., 2017), and incorporate max-flow targets from the upper level as part of a multi-objective optimization to help the platform make more informed matching decisions.

The remainder of this paper is organized as follows. Section 2 presents the hierarchical structure of the proposed solution, where we formulate the upper-level flow maximization problem and the lower-level batch optimization model, as well as their integration. Section 3 introduces a case study with simulated taxi trip data, and shows how the two-level optimization outperforms a baseline

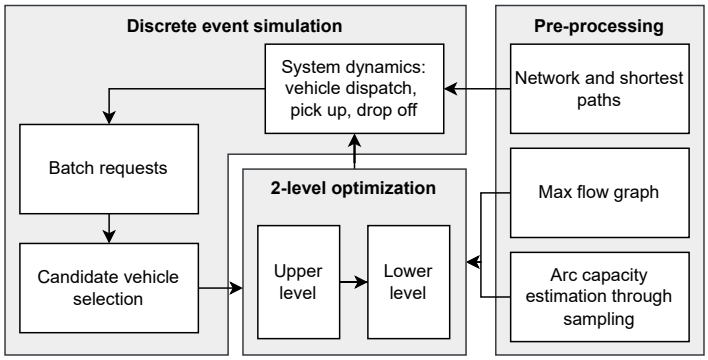

Figure 1: Proposed 2-level matching optimization implemented in a ride-sourcing platform simulation

algorithm that is not hierarchical in nature. Section 4 concludes with a summary of main findings and areas for future research.

2 Methodology

In this paper, we highlight the value of leveraging near-future incomplete demand and supply information to improve batch optimization performance. Figure 1 illustrates the hierarchical structure of the proposed method, namely how the optimization framework is incorporated in the ride-sourcing platform, and how preprocessing steps support real-time matching decisions. The next two subsections will provide specifications of the upper and lower level optimization models, and how the upper level flow maximization model communicates with the lower level batch optimization.

Upper level: flow maximization model

Equations (1) - (4) present the objective function and constraints to be solved for each upcoming prediction horizon with index k, prior to each time the ride-sourcing platform performs batch matching. The duration of the prediction horizon should be as long as the duration of the batching interval. The terms "prediction horizon" and "time step" are used interchangeably to refer to the upper level flow maximization problem.

$$\max_{x_{ij}(k)} \quad F(k) \tag{1}$$

$$\text{subject to} \quad \sum_{\{j:(i,j\in E)\}} x_{ij}(k) - \sum_{\{j:(j,i\in E)\}} x_{ji}(k) = \begin{cases} F(k) & \text{for } i = S_1 \\ 0 & \text{for } i \notin \{S_1, S_2\} \\ -F(k) & \text{for } i = S_2 \end{cases} \tag{2}$$

$$0 \leq x_{ij}(k) \leq u_{ij}(k) \quad \text{for all } (i,j) \in E \tag{3}$$

$$x_{ij}(k) \in Z \quad \text{for all } (i,j) \in E \tag{4}$$

The objective $F(k)$ is the total flow through the graph for time interval k which is to be maximized. The decision variables $x_{ij}(k)$ are flows along each arc $(i,j) \forall i,j \in C$. Constraints in Equation (2) characterize flow conservation, which should be equal to the value of the objective function at the source S_1 and sink S_2, and zero elsewhere. Constraints (3) require non-negative flow through any arc which cannot exceed the arc capacity $u_{ij}(k)$; arc capacities will be elaborated later in this section. Finally, constraints (4) require the program to provide integer solutions. An advantage of this formulation is the guaranteed existence of an optimal solution, which could be non-unique.
The set of vertices V, interchangeably referred to as nodes, follows $V = S \cup M^0 \cup M^1 \cup \Lambda$, and consists of: the set of source and sink $S = \{S_1, S_2\}$, the set of idle vehicles $M^0 = \{M_1^0, M_2^0, ..., M_n^0\}$ located in each of the n regions with $|M^0| = n$, the set of single occupancy vehicles $M^1 = \{M_{1,1}^1, M_{1,2}^1, ..., M_{i,j}^1\}$ with current region in $i \in C$ and destination region $j \in C$, and the set of trip requests $\Lambda = \{\Lambda_{1,1}, \Lambda_{1,2}, ..., \Lambda_{i,j}\}$ that have origins and destination respectively in regions $i, j \in C$. Although both solo and two-passenger pool trips are included in this paper, it is worth

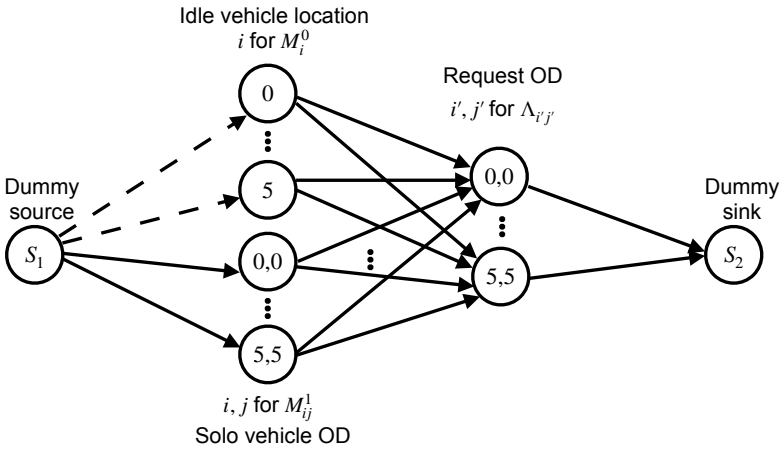

Figure 2: Illustration for the graph of flow maximization in a network with 6 regions

noting that such a formulation can adapt well to different sharing scenarios, as the arcs are specified separately for solo and pool trips.

At the start of each time step k, the maximum flow problem depicted in Figure 2 is solved to determine the optimal interregional vehicle flow over the prediction horizon. For notational convenience, the time index k may be omitted when unambiguously referring to flows within a specific time step. To accomplish this, the operator updates the arc capacities associated with vehicle supplies and trip demands in different regions of the network. The next subsection explains the capacity estimation for empty vehicles and those with one passenger available for potential pooling.

Arcs $(i, j) \in E$ between two vertices $i, j \in V$ have finite capacity $u_{ij}(k)$ which is related to the spatial distributions of vehicles and trip requests. From left to right in Figure 2:

- $(S_1, i) \forall i \in M^0 \cap M^1$ represent the vehicle supply: dotted arcs have capacity of $m_i^0(k)$, represent the predicted number of idle vehicles in region i to potentially serve a nearby solo trip request over time step k. Solid arcs, whose capacities are $m_{ij}^1(k)$, correspond to the predicted number of single occupancy vehicles over time step k for serving pool trips of passengers whose origins are located in region i and destinations in region j;
- $(i, j) \forall i \in M^0 \cap M^1, j \in \Lambda$ indicate the service capacity for solo or pool trips traveling between all possible origin and destination region pairs, and constitute a subset of the total number of available vehicles, namely, (1) the capacity for matching solo trips is $\alpha_{ij}^0(v(k)) \cdot m_i^0(k)$ for arcs from $i \in M^0$ to $j \in \Lambda$. $\alpha_{ij}^0(v(k))$ is estimated based on the network geometry and the average speed $v(k)$ at the start of time step k; (2) the capacity for matching pool trips is $\alpha_{ij,i'j'}^1 \cdot m_{ij}^1$, where i is the current region that the vehicle is located in, j is the destination region of the passenger currently traveling inside the vehicle, and $i', j' \in C$ are the origin and destination regions of the incoming request to be matched. The estimation of this quantity is similar to those for solo trips, and additionally involves detour constraints for those in a pooled trip.
- $(i, S_2) \forall i \in \Lambda$ relate to the trip demand, where $\lambda_{ij}(k)$ denotes the number of passenger requests expected to enter the system in the time interval k with origin in region i and destination in region j.

Detailed experimental settings and results from the estimation of arc capacities and vehicle numbers are omitted in this short paper.

Lower level: batch matching with max-flow target

At the lower level, requests are matched in batches for either solo trips or pooled trips; at predefined batching intervals, the platform creates a batch containing requests that enter during the same time period, and solve an integer quadratic programming problem to match these requests with a selected set of available vehicles with minimum assignment costs.

To encourage the platform to attain flow-maximization targets from the upper level, we additionally introduce a penalty term in the objective function to account for deviations in arc flow from the max-flow targets, as well as related constraints to translate vehicle-request matching into arc flows in the upper level.

The problem of maximizing the number of requests served while minimizing deviation from the max-flow targets is written as a multi-objective optimization in Equation (5) subject to constraints in Equations (6) - (8). The vehicle-trip assignment aims to minimize the objective function cost which consists of three components, namely the total passenger waiting and detour time, the total cost of request rejection, and the total cost of deviation from max-flow targets.
In the objective function, the first two components constitute a classical bipartite vehicle-request matching problem for the current batch, and the third component quantifies the cost of the proposed batch assignment evaluated relative to predicted performances over a longer time horizon.
While the bipartite vehicle-request matching can be solved as an integer combinatorial optimization problem, the nomenclature and formulation in this paper is similar to that in Alonso-Mora et al. (2017) with vehicle capacity of two requests. Let $\mathcal{R} = \{r_1, ..., r_{|\mathcal{R}|}\}$ denote the set of requests in the batch, which can form trips in a stand-alone fashion or be combined with another compatible request. By enumerating solo trips and grouping two pool requests that would satisfy the waiting time and maximum detour into a trip T_i, we have a set of trips $\mathcal{T} = \{T_1, ..., T_{|\mathcal{T}|}\}$. The set of vehicles is denoted as $\mathcal{W} = \{W_1, ..., W_{|\mathcal{W}|}\}$. The assignment between trips and vehicles forms a set of binary decision variables $\epsilon_{ij} \in \mathcal{E},\ \forall i \in \mathcal{T}, \forall j \in \mathcal{W}$, where $\epsilon_{ij} = 1$ if the trip i is matched with vehicle j. The decision variables are subject to classical constraints that each vehicle can serve at most one trip, which can contain one or two requests, and a trip is either served or rejected.
Pool trips can be formed in two ways, either by grouping together two new requests in the current batch, or matching one new request with an existing solo trip. Respective decision variables for trips and vehicles are created to enable this functionality. On the other hand, once a vehicle-trip match is established, the platform cannot reassign them in subsequent intervals. In addition, a request can be pooled with at most one other request; in other words, the platform does not support back-to-back pooling after dropping off a previous passenger.
In addition, in this work, we create constraints to group trip-vehicle flows ϵ_{ij} based on the zones where vehicles are located in, as well as the request origin and destination zones; the solution is converted into $y_{ml},\ \forall m \in M^0 \cap M^1, l \in \Lambda$. Then, the deviation between the aggregated solution from y_{ml} and the max-flow target flows adjusted for the batch duration x_{ml} are included in the objective function as the sum of squared differences.
Omitting the time step index without loss of generality, the objective function is written as follows:

$$\min_{\epsilon_{ij}} \quad \sum_{i \in \mathcal{T},\ j \in \mathcal{W}} c_{ij} \cdot \epsilon_{ij} + \sum_{k \in \mathcal{R}} c_{ko} \cdot \chi_k + \sum_{m \in M^0 \cap M^1,\ l \in \Lambda} c_d \cdot (x_{ml} - y_{ml})^2 \tag{5}$$

$$\text{subject to} \quad \sum_{i \in \mathcal{I}^T_{W=j}} \epsilon_{ij} \leq 1 \qquad \forall w_j \in \mathcal{W} \tag{6}$$

$$\sum_{i \in \mathcal{I}^T_{R=k}} \sum_{i \in \mathcal{I}^W_{T=i}} \epsilon_{ij} + \chi_k = 1 \qquad \forall r_k \in \mathcal{R} \tag{7}$$

$$y_{ml} = \sum_{i \in m,\ j \in l} \epsilon_{ij} \qquad \forall m \in M^0 \cap M^1, l \in \Lambda \tag{8}$$

In the objective function (5), c_{ij} is the cost of serving a trip-vehicle pair, which in this problem translates into the sum of the total waiting time and any detour time for all passengers involved in the request; c_{ko} is a large penalty term for when a request cannot be served; c_d is a penalty term related to any deviation from the max-flow target solutions adjusted for the current batch step, namely x_{ml}. Section 3 will report results from a grid search on values of δ and c_d. $\mathcal{I}^T_{W=j}$ denotes the set of trips that can be served by vehicle j. Constraints in (8) map the trip-vehicle solutions to corresponding arc flows defined in the max flow solution structure. The integer quadratic programming problem is set up in Python and solved using commercial solver Gurobi version 10.0.3.

3 Results

In this section, we evaluate the performance of the proposed two-level optimization framework under different δ and c_d, and illustrate, with the best parameters found, how the platform outperforms a well-known benchmark of matching with the nearest available vehicle.

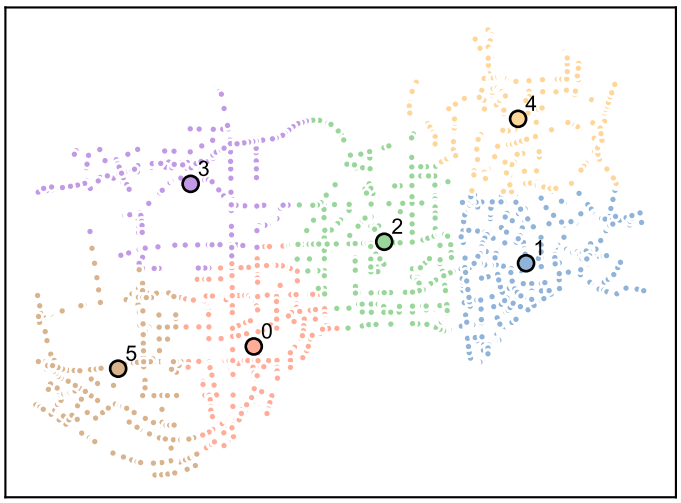

Figure 3: K-means clustering of nodes in the study area in Shenzhen into six regions

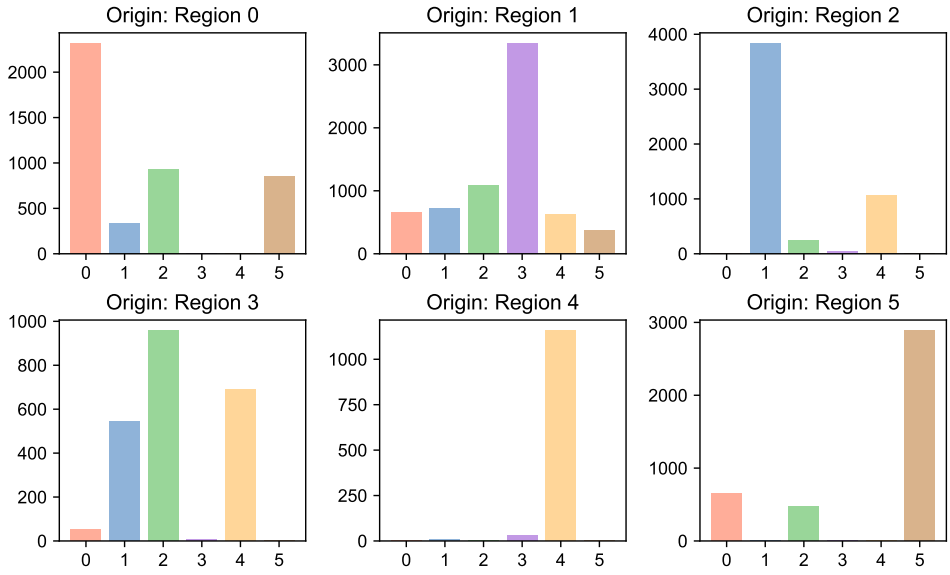

Figure 4: Total trip demand from one region (subplot title) to other regions (x-axis of each subplot) over three hours

Case study

The proposed matching framework is tested using a simulated taxi dataset of Shenzhen, China created by Beojone & Geroliminis (2021) and later extended by Zhu et al. (2022). Although the reference studies are not necessary to appreciate the contribution of this paper, interested readers can find descriptions for the study area and the traffic dynamics calibration in Beojone & Geroliminis (2021), and synthetic trip data generation procedure in Zhu et al. (2022). The data consists of 159'162 pre-generated trips with trip start times over three hours, where each request has a start time, an origin node and a destination node. The city network is simplified into 1'858 intersections connected by 2'013 street segments over an area of approximately 72 km^2.
The nodes in the study area are clustered into $n = 6$ regions by k-means clustering, as shown in Figure 3. The three-hour trip demand generated in Zhu et al. (2022) is illustrated in Figure 4, where spatial demand asymmetry can be observed. As additionally illustrated in Figure 7 by the grey demand curve, the network experiences peak demand hour during the second hour, where trip generation rate is double that during the first and third hours.

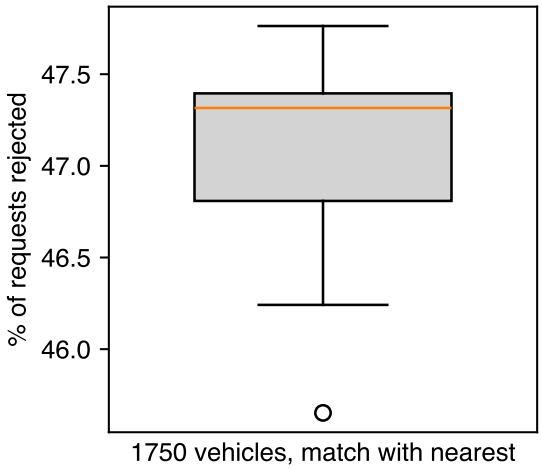

Figure 5: Rejection fraction box plot from 10 simulations of nearest available matching policy

Performance baseline: nearest available vehicle

We first introduce the performance baseline for a fleet size of 1750 vehicles. As shown in Figure 5, 10 simulation runs with the nearest available policy lead to average rejection of 47.4%. This is a reasonable policy for a ride-sourcing service due to its simplicity and how it mimics taxi drivers's cruise for customer behaviour. Therefore, the remainder of this short paper will compare the 2-level optimization with the nearest available baseline.

Grid search for best δ and c_d

By obtaining information on the vehicle and request distribution in the near future, the platform can make matching decisions that better distribute available vehicles in the long term. Simulations with prediction horizons δ in the range of [1, 19] minutes and deviation penalty c_d in the range of [1, 19] are tested at increments of 2 units. For the lower level, the batch size for matching requests is 100, or as soon as 1 minute has elapsed.
A simulation with each parameter pair is run 4 times with different random seeds to account for small variations across demand patterns. The parameters tested and the resulting service levels measured as "fraction of requests rejected" are reported in Figure 6. From the top left subfigure, it can be noted that $\delta = 1$ minute is equivalent to the duration of a batching interval; in other words, matching decisions with this upper level setting does not consider any future information; and the resulting rejection fraction is more scattered and higher compared to other parameter settings with longer prediction horizons. Furthermore, it can be observed that the rejection fraction is slightly lower at smaller c_d values. Finally, all the parameters tested here outperform the nearest available policy. We proceed with $\delta = 11$, $c_d = 1$ for the remainder of the analysis in this short paper, as it is shown to produce the best service level for a three-hour simulation.

Service level per 10-minute interval

As pointed out earlier, the platform experiences a peak-hour demand during the second hour. Let us examine how the proposed framework performs during the peak hour and how it recovers from the increased demand. We compare this analysis with the "nearest vehicle" policy in Figure 7. Each data point corresponds to the number of accumulated requests in a 10-minute interval; it can be verified that during off-peak hours, the demand is approximately 100 trips/minute, and during peak hours 200 trips/minute. During the first hour, both matching policies perform equally well under low demand, but when the trip demand spikes during the peak hour, more requests are being rejected with the nearest available policy than the 2-level policy, as shown in the blue region in the graph. Finally, the 2-level policy can recover faster from peak demand, at as early as the 14th interval, whereas "nearest available" only recovers at interval 7.

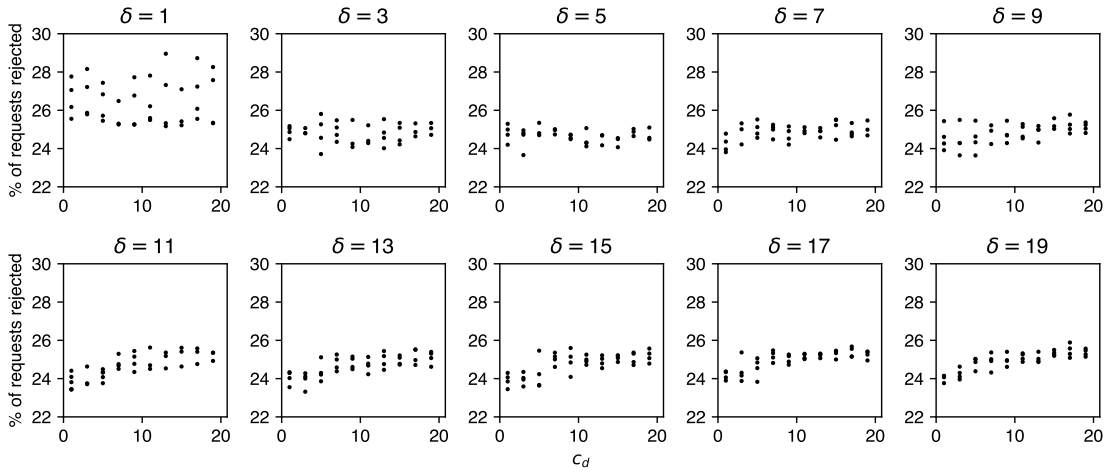

Figure 6: Fraction of rejected requests for different prediction horizons δ (in minutes, each value is a subplot) and penalty on deviation from upper level target c_d (horizontal axis in every subplot)

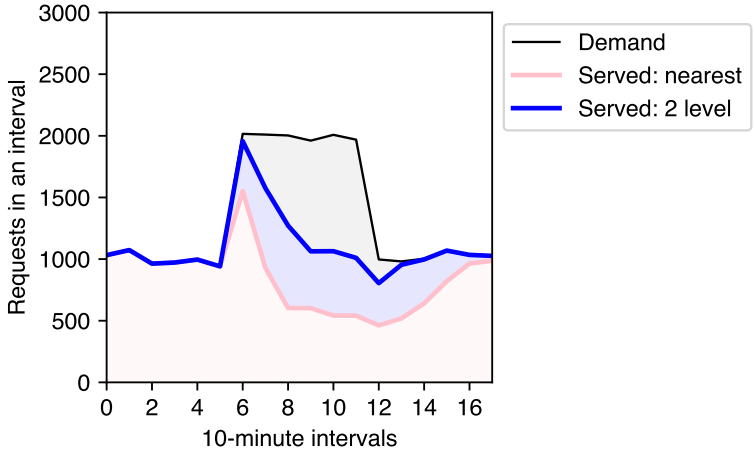

Figure 7: Demand and served request in 10-minute intervals. Nearest: match a request with nearest vehicle; 2-level: proposed matching optimization.

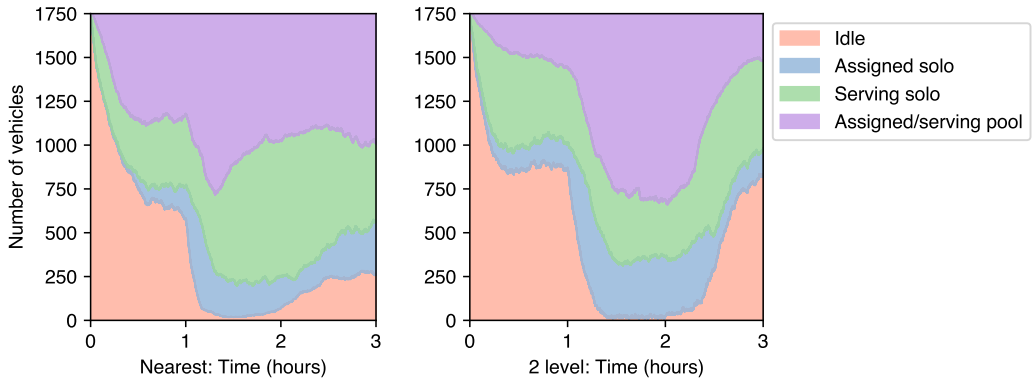

Figure 8: Evolution of vehicles status over a 3-hour simulation for two matching policies. Left: nearest; right: 2 level.

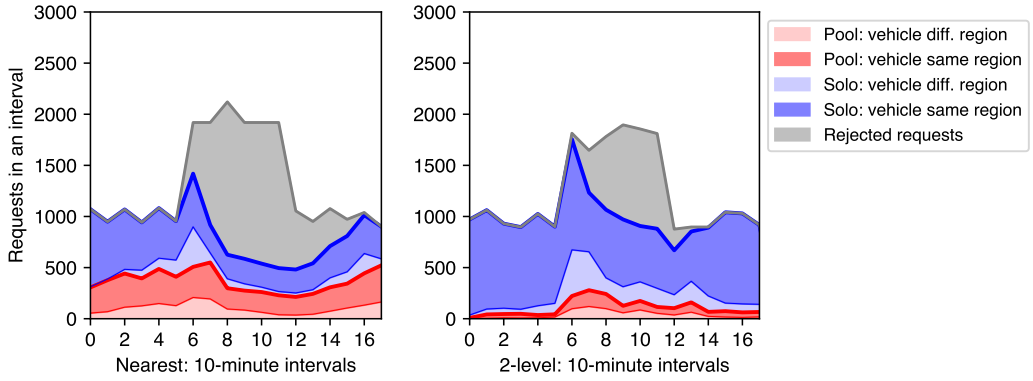

Figure 9: Requests that are matched with vehicles in different statuses and regions. The regions are those illustrated in Figure 3. In the legend: Pool: a request matched with a vehicle who has an existing passenger. Solo: a request matched with a vehicle who is empty."Diff. region": at the time of matching, the matched vehicle is located in a region different from where the request originates. "Same region": at the time of matching, the matched vehicle is located in an area same as where the request originates.

Evolution of vehicle status

By tracking the status of all 1750 ride-sourcing vehicles throughout two simulations in Figure 7, where "nearest available" is shown in the left subfigure and "2 level" on the right, one notices a higher proportion of vehicles serving shared rides in the left subfigure, owing to the nature of the matching mechanism, which assigns a request as long as the nearest vehicle has an available seat, which may lead to inefficiencies related to detours and spatial imbalances of available vehicles, ie. idle vehicles located in places that are far away from request origins.

Requests and locations of matched vehicles

Figure 9 shows the proportion of requests being served by vehicles that are located in close proximity, i.e., in the same region, or farther away, i.e., in different regions. Again, in the left subplot where the nearest available vehicle is assigned, the proportion of bright red shading indicates that a higher number of pool trips are matched with vehicles located in the same region. One can draw the conclusion that although pooling can be seen as desirable to increase the vehicle capacity utilization, the pair of pool passengers should be selected carefully, and excessive pooling may deteriorate the platform's service level.

4 Conclusion

In this short paper, we show that the proposed two-level matching optimization framework can outperform a reasonable benchmark of matching with the nearest available vehicle. The advantage of the formulation lies in that it spatially aggregates ride-sourcing trip demand over a short horizon, and provide an optimal allocation of solo and pool trips for origins and destinations distributed over different regions. At the lower level, compliance with upper level targets is desirable, and deviations from the targets are penalized.

For a system that is characterized by spatial demand asymmetry and peak hour demand, the proposed framework with the best grid-search parameters can greatly improve service level, as it optimizes the matching with pool and solo trips by selecting vehicles in strategic locations. Nevertheless, the parameters may be further optimized and validated with more simulation runs and sensitivity analysis. In the future, additional analysis with trip demand that exhibits different spatial and temporal asymmetry can provide useful insight.

Furthermore, when pooling decisions are assigned in the batching process, it is currently assumed that passengers will accept the assigned option. For future directions for this work, one can incorporate passenger preference in the lower level; in the case that passengers are unwilling to participate in pooling, the flow maximization targets can serve as operational guidelines for the platform to reassign requests to different vehicles. This is a promising and important area for future research.

Acknowledgements

This paper is partially funded by DIT4TraM "Distributed Intelligence & Technology for Traffic & Mobility Management" project from the European Union's Horizon 2020 research and innovation programme under Grant Agreement 953783.

References

Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (2001). Maximum flow problem. In C. A. Floudas & P. M. Pardalos (Eds.), *Encyclopedia of Optimization* (pp. 1339–1350). Boston, MA: Springer US. Retrieved 2023-07-31, from https://doi.org/10.1007/0-306-48332-7_275 doi: 10.1007/0-306-48332-7_275

Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., & Rus, D. (2017). On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, *114*(3), 462–467. Retrieved from https://www.pnas.org/content/114/3/462 doi: 10.1073/pnas.1611675114

Beojone, C. V., & Geroliminis, N. (2021). On the inefficiency of ride-sourcing services towards urban congestion. *Transportation Research Part C: Emerging Technologies*, *124*, 102890. Retrieved from http://www.sciencedirect.com/science/article/pii/S0968090X20307907 doi: 10.1016/j.trc.2020.102890

Cordeau, J.-F., & Laporte, G. (2007). The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, *153*(1), 29–46. Retrieved from https://doi.org/10.1007/s10479-007-0170-8 doi: 10.1007/s10479-007-0170-8

Hillier, F. S., & Lieberman, G. J. (2001). *Introduction to operations research* (Seventh ed.). New York, NY, USA: McGraw-Hill.

Pelzer, D., Xiao, J., Zehe, D., Lees, M. H., Knoll, A. C., & Aydt, H. (2015). A partition-based match making algorithm for dynamic ridesharing. *IEEE Transactions on Intelligent Transportation Systems*, *16*(5), 2587-2598. doi: 10.1109/TITS.2015.2413453

Santi, P., Resta, G., Szell, M., Sobolevsky, S., Strogatz, S. H., & Ratti, C. (2014). Quantifying the benefits of vehicle pooling with shareability networks. *Proceedings of the National Academy of Sciences*, *111*(37), 13290–13294. Retrieved from https://www.pnas.org/content/111/37/13290 doi: 10.1073/pnas.1403657111

Simonetto, A., Monteil, J., & Gambella, C. (2019). Real-time city-scale ridesharing via linear assignment problems. *Transportation Research Part C: Emerging Technologies*, *101*, 208-232. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0968090X18302882` doi: https://doi.org/10.1016/j.trc.2019.01.019

Yao, R., & Bekhor, S. (2021). A dynamic tree algorithm for peer-to-peer ridesharing matching. *Networks and Spatial Economics*, *21*(4), 801–837.

Zhu, P., Sirmatel, I. I., Trecate, G. F., & Geroliminis, N. (2022). Idle-vehicle rebalancing coverage control for ride-sourcing systems. In *2022 european control conference (ecc)* (p. 1970-1975). doi: 10.23919/ECC55457.2022.9838069