On-demand Ride-sharing Services with Meeting Points

Sevket Gökay[1,2], Andreas Heuvels[2] and Karl-Heinz Krempels[1,2]
[1]*Informatik 5 (Information Systems), RWTH Aachen University, Aachen, Germany*
[2]*CSCW Mobility, Fraunhofer FIT, Aachen, Germany*

Keywords: Demand-Responsive Transport, Dial-a-Ride Problem with Time Windows, On-demand Ride-sharing, Spatial Clustering.

Abstract: On-demand ride-sharing services propose an alternative transportation mode to public and private transportation. They have similarities with private transportation, since the customers have the convenience of travelling from and to any desired location while defining the departure (or arrival) time. They resemble public transportation in multiple customers sharing a vehicle with similar journeys. This work proposes an approach to improve the throughput of on-demand ride-sharing services by introducing meeting points. The idea bases on combining a vehicle's nearby location visits (whether for pick-up or drop-off) into one, if temporal and spatial constraints are held, in order to reduce the vehicle detour costs. It, by design, diminishes customer convenience, since walking legs are introduced and departure/arrival times might deviate from what is desired. The trade-offs are evaluated by running two simulations, one without and one with meeting points. The results indicate that even a small customer inconvenience can yield significant increase in the number of satisfied trip requests without increasing vehicle costs.

1 INTRODUCTION

Technological advancements change our lives daily. They can make us rethink existing solutions, e. g. transportation services, and enable new approaches to realize them. Since smartphones with Global Positioning System (GPS) and Internet capabilities are becoming ubiquitous, on-demand ride-sharing services are gaining in popularity. Some recent reports analyse their effects and usage patterns: Research analysis in (Transportation Research Board, 2016) states that ride-sharing services are mostly used during night time when the coverage of public transportation is either poor or unavailable. This shows that ride-sharing services are often used to complement public transportation instead of competing with it. However, a more recent analysis in (Schaller, 2018) argues that ride-sharing actually increases traffic because more users switch from non-auto modes, and that the capacity of the vehicle is not used to its fullest because, even in shared rides, there are parts of the ride involving only one passenger. The *ride-sharing* aspect of these services is still in its infancy, since they are mostly seen as an alternative to the taxi service. For example, according to the analysis in (Schaller, 2018), the most recent data suggests that the percentage of shared trips compared to all trips is only 22% by Lyft and 23% by Uber in New York City.

Motivation

Public transportation services like bus services can achieve high throughput (i. e. bringing a large number of people from *A* to *B* in a period of time) because they operate with fixed routes, timetables and bus stops. This architecture neglects the customer satisfaction aspects since it enforces *always* walking to/from a station and waiting time at a station. On the other side of the spectrum, on-demand transportation services (e. g. taxi or ride-sharing) operate without fixed routes and timetables, and with arbitrary pick-up and drop-off locations. However, ride-sharing services might suffer from too frequent stops of a vehicle and suboptimal routes due to many small detours, especially when they utilize bigger vehicles (e. g. minibus, bus), where the likelihood of a large number of concurrent passengers is high.

In such contexts, the quality of service and throughput could be improved by introducing *meeting points* where pick-up and drop-off events in close proximity (w. r. t. time and location) can be bundled together. This, in our perspective, strikes a

Gökay, S., Heuvels, A. and Krempels, K.
On-demand Ride-sharing Services with Meeting Points.
DOI: 10.5220/0007709101170125
In *Proceedings of the 5th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2019)*, pages 117-125
ISBN: 978-989-758-374-2

happy medium between traditional public transportation (fixed stations) and classical ride-sharing services (no stations). Moreover, our work introduces a new concept, namely *location flexibility*, to the Dial-a-Ride Problem (DARP)[1], since DARP variants already leverage temporal flexibility – by making use of *time windows*[2] – but not location flexibility.

Our starting point is an online multi-vehicle DARP solution, where trip requests are processed as they appear in real-time without knowledge about the future (online) and are either assigned to one vehicle in the fleet (multi-vehicle) or rejected because they do not satisfy the constraints. It builds on the works of (Gökay et al., 2018; Tsubouchi et al., 2010) and uses insertion heuristics to solve the scheduling problem.

The paper is organized as follows. Section 2 provides an overview of the research field and similar practical solutions. Section 3 illustrates the taken approach in detail. Subsequently, Section 4 highlights the key points about the implementation, presents the evaluation methodology and discusses the results. Finally, Section 5 concludes the paper.

2 RELATED WORK

In (Stiglic et al., 2015), meeting points are used to bundle nearby requests if possible. In contrast to our approach, the authors rely on a preset of meeting points and suggest to use coffee shops or gas stations for example, depending on the local legal circumstances. For this study, a travel demand model for the metropolitan Atlanta region is used, which is then divided into multiple zones, each with a fixed amount of randomly generated meeting points. By requiring customers to walk and gather within a certain vicinity, the number of stops for a vehicle as well as the overall driven distance could be reduced.

The approach in (Li et al., 2018) aims to improve a ride-sharing system based on time windows by adding the concept of meeting points. To solve the mixed integer linear program that models the problem, a tabu-search based meta-heuristic algorithm is implemented. The results are then compared to the optimum obtained through CPLEX. While using small fleets with 3-5 vehicles, the faster heuristic yields results close to the optimum, while meeting points can reduce mileage by 2.7%–3.8%.

[1]DARP is the underlying problem definition that ride-sharing services aim to solve.

[2]Location visits (i. e. pick-up and drop-off events) have to occur within given time windows.

Optimal Multi-Meeting-Point Route (OMMPR) queries, introduced in (Li et al., 2016), aim to find a short path between a start and end point within a road network while also minimizing detours for additional stops in between. These intermediate stops are not fixed to a location but limited by constraints on how much they can accommodate the original query to a meeting point. Calculating the OMMPR query is NP-hard and two solutions based on dynamic programming are proposed in the paper.

Uber introduced *Express POOL (Stock, 2018)* in early 2018, an addition to its taxi services that reduces detours by having the customer walk to/from a location nearby the start/endpoint of the initial request. The so-called *Express spots* change based on popular routes at the time of request. The application requires the customer to wait a few minutes in the beginning, such that possible co-passengers can be matched to the same ride. A higher success rate can be achieved by postponing every final decision as long as possible to increase the systems flexibility (Hawkins, 2018).

One aspect of our meeting points approach relies on determining hotspots based on historical data. We aim to identify hotspots as potential meeting points by applying *clustering*, a data mining technique. For this purpose, we investigated several clustering algorithms. Density-based spatial clustering of applications with noise (DBSCAN) is introduced in (Ester et al., 1996). Opposed to other common algorithms like *k*-means (MacQueen, 1967), which is known to be unfitting for spatial clustering (Murray and Grubesic, 2002; Grubesic et al., 2014), DBSCAN does not separate data into a fixed number of clusters. It tries to identify concentrations of data points by analysing the density and by grouping those points that are in a sufficiently dense neighbourhood. But the algorithm has a problem identifying clusters if the data point density varies strongly in different regions. Ordering Points To Identify the Clustering Structure (OPTICS), proposed in (Ankerst et al., 1999), counters this weakness by taking points of other nearby clusters into account.

3 APPROACH

Our input model for requests contains the number of passengers, the desired pick-up (or drop-off) time, the pick-up and drop-off locations, a *slack time*[3], a *time flexibility* and a *location flexibility*. *Process request* in Figure 1 converts this input into a fully-fledged *trip*

[3]The duration to expand the pick-up and drop-off times into time windows.

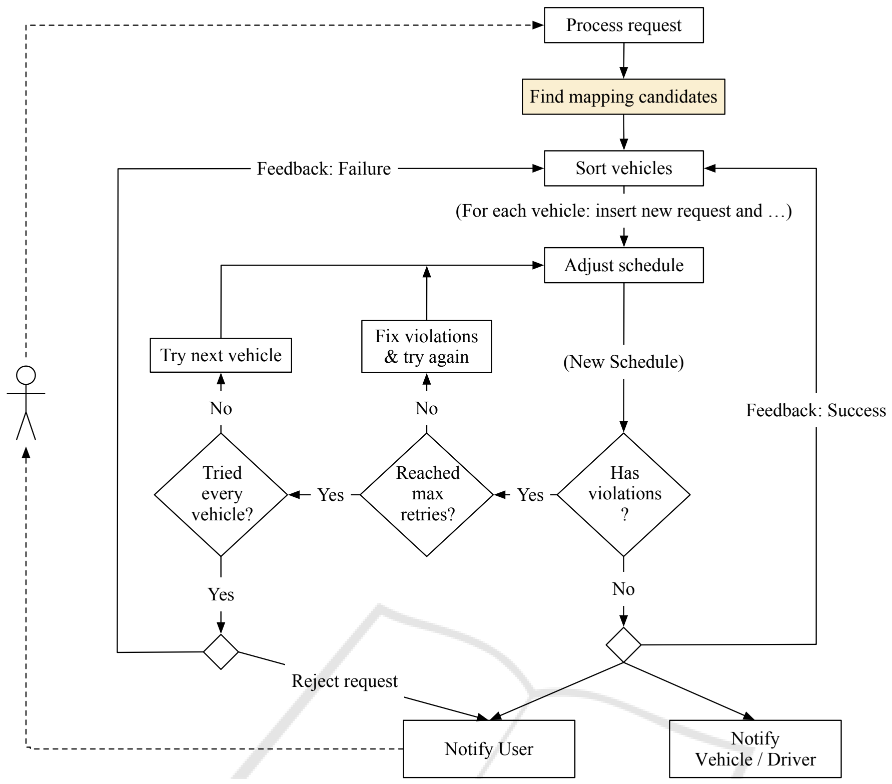

Figure 1: Decision structure of the approach. Finding mapping candidates is the main contribution of this work.

request model that contains the number of passengers, the pick-up and drop-off events, the time and location flexibilities. Each event contains a location, a time window[4] and an actual time within the time window. The calculation of time windows is done according to the scheme in (Tsubouchi et al., 2010). Time flexibility describes the duration we can increase the time window by in order to define the temporal search space when looking for events in close proximity. Similarly, location flexibility describes the maximum walking path length from the location and consequently the spatial search space.

In our online scenario, we consider already accepted trip requests to be bound to their location (i. e. their pick-up and drop-off locations cannot be moved). Since we aim to respond to trip requests in a matter of seconds, changing the locations afterwards (e. g. if as a result of future requests, more optimal meeting points are possible) and notifying the user about an unexpected walking leg would deteriorate the user experience. In this regard, our approach *maps* the pick-up and drop-off events of a new request to those of accepted requests in a vehicle schedule, if their spatio-temporal distance respects specified location and time flexibility thresholds, or we process the request *as is*. The time window and location properties of the new request's events are practically overriden with the values of the accepted one. This principle, however, raises the importance of the events of initial requests since their quality is decisive whether and how the future request events can be mapped. To counteract this situation, we introduce *hotspots*: Analyse historical request data to determine spatio-temporal hotspots (i. e. sort of *virtual stations that can move over time*). This process is described in Section 3.1. A hotspot, similar to a pick-up or drop-off event, has a location and time window (duration of its *hotness*) and is therefore treated as such. We try mapping a request event at first to events in a vehicle schedule and then to hotspots.

Finally, our approach can be summarized as follows (i. e. *Find mapping candidates* in Figure 1 contains the following steps):

1. For the pick-up and drop-off events of a new trip request, find all *feasible* events in the vehicle schedule and *feasible* hotspots (See Section 3.2). This gives us all the mapping candidates for both events separately. If there is no candidate for one event, then the original event of the request is used.

[4]It denotes the earliest and latest times that this event can occur by applying the slack time

2. Combine all pick-up and drop-off candidates that can be served by the same vehicle.
3. Order the pairs of pick-up and drop-off candidates by their ascending spatio-temporal distance to the original events (See Section 3.3). The candidates from vehicle schedule have priority over hotspots.
4. Return the first pair that does not violate customer constraints (See Section 3.4).

The resulting pair of events can now be treated as an ordinary trip request without any additional information and processed further (starting with *Sort vehicles* in Figure 1).

3.1 How to Determine Hotspots?

Existing solutions for finding meeting points follow mostly one of the following approaches:

- Random generation/selection from data (Stiglic et al., 2015)
- Crowdsourcing (Hansen et al., 2010)
- Certain locations (e. g. parking place, fuel station, street intersection) with convenience features (e. g. illumination, parking quality) (Czioska et al., 2017)

However, these predominantly concentrate on meeting points between a driver and passenger. In contrast, our approach aims to group pick-up and drop-off events of passengers. It employs a data mining technique to analyze historical data (i. e. demand) to derive patterns and presumes that future demand will be similar. The process of determining hotspots is automated and requires no manual intervention or additional interpretation.

After evaluating various clustering algorithms and implementations with trips extracted from New York City taxi trip data (2010–2013)[5], we decided to use OPTICS with ξ[6] extraction. The dataset contains taxi trips with the start and end times of the trip and pick-up and drop-off locations. We split a trip into two data points (pick-up and drop-off), where each data point contains a time and location. We partition the period of time, for which we want to identify clusters, by a predefined duration parameter *bucket size* into smaller *time buckets*. This has two main advantages: Better control over temporal validity of hotspots (i. e. hotspots remain hot as long as the time bucket) and improved clustering runtime because of dimension reduction of the distance function. Subsequently, we assign the data points to their time buckets and run OPTICS for each time bucket. Since we are handling the time dimension externally, the distance function in OPTICS addresses only the spatial distance and uses haversine formula.

The output of OPTICS is a hierarchical cluster structure with larger high-level clusters containing smaller more dense clusters. In order to determine the hierarchy level, we walk the hierarchy top-down and for each cluster at each level we calculate the weighted center and determine whether there are points outside of a predefined distance threshold, *decide level radius*. If it is the case, we continue the hierarchy walk. Otherwise, we find the closest location on the street network for this cluster's weighted center and decide that it is a hotspot. The results are stored as key-value pairs, where each time bucket contains a set of hotspots.

3.2 Feasibility Constraints for Candidates

For an event or hotspot to be considered as a candidate, the following constraints must be satisfied:

1. Location constraint: A walking path between the locations of original event and candidate must exist and its length should not be greater than the location flexibility.
2. Time constraints:
   - If the user specified the pick-up time and the candidate is for pick-up, the candidate's time window must start after the user's.
   - If the user specified the drop-off time and the candidate is for drop-off, the candidate's time window must not end after the user's.
   - Candidate's time window is allowed to be completely within the user's (Figure 2 (a_1)).
   - If user's time window is completely within the candidate's, then the total exceed ($e_1 + e_2$) must not be greater than the time flexibility (Figure 2 (a_2)).
   - If time windows of candidate and event only partially overlap, the exceed e must not be greater than the time flexibility (Figure 2 (b_1), (b_2)).
   - If there is no overlap, the distance e from candidate's farthest point of time window to event's closest point of time window, must not be greater than the time flexibility (Figure 2 (c_1), (c_2)).

To find event candidates, we walk over the vehicle schedules and mark feasible events. Similarly,

[5]https://databank.illinois.edu/datasets/IDB-9610843

[6]A steepness threshold to classify clusters by relative density change.

to find hotspots candidates, we find corresponding time buckets and mark feasible hotspots for each time bucket.

3.3 Criteria for Ordering

For each event-candidate arrangement, we calculate the following terms:

l_d The distance between the locations of the event and candidate.

d_e Total duration of event's time window.

d_o Overlapping part of event's and candidate's time window (Figure 2 (a$_1$), (a$_2$), (b$_1$), (b$_2$)).

d_{ib} Indent before: Time offset how much the event's start is *before* the candidate's (Figure 2 (b$_1$), (c$_1$)).

d_{ia} Indent after: Time offset how much the event's end is *after* the candidate's (Figure 2 (b$_2$), (c$_2$)).

d_{eb} Excess before: Time offset how much the candidate's start is *before* the event's (Figure 2 (a$_2$), (b$_2$)).

d_{ea} Excess after: Time offset how much the candidate's end is *after* the event's (Figure 2 (a$_2$), (b$_1$)).

Then, we order the candidates in ascending order by calculating the following cost function:

$$\Delta_C = l_d * (ratio_{overlap} + ratio_{before} + ratio_{after})$$

where

$$ratio_{overlap} = 1 - \frac{d_o}{d_e}$$
$$ratio_{before} = \frac{d_{eb} + d_{ib}}{d_e}$$
$$ratio_{after} = \frac{d_{ea} + d_{ia}}{d_e}$$

Ordering in this fashion allows us to prioritize candidates that have smaller spatio-temporal distances to the original event.

3.4 Feasibility Constraints for Customers

After filtering out infeasible candidates by applying event-level constraints, we filter out infeasible pick-up and drop-off pairs, if one of the following customer-level constraints are violated:

- Pick-up and drop-off locations must be different
- Earliest drop-off time $-$ earliest pick-up time $\geq$ direct ride time

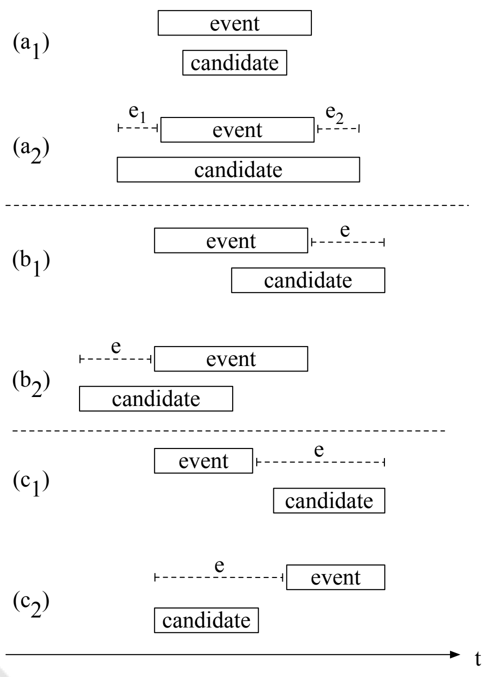

Figure 2: The various arrangement possibilities of an event and candidate w. r. t. time.

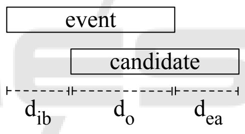

Figure 3: Exemplary event-candidate arrangement with indent before d_{ib}, overlap d_o and excess after d_{ea}.

- Actual pick-up time $<$ actual drop-off time
- Actual ride time $\geq$ direct ride time

Direct ride time is the duration of the shortest path from pick-up to drop-off (without detours), whereas *actual ride time* denotes the ride time within a vehicle schedule (with detours).

4 EVALUATION

This section describes the evaluation methodology, the data sets and presents the results while discussing key findings.

4.1 Implementation Overview

The system is developed as a standalone Java 8 application. Route calculation is handled by GraphHopper[7], which imports OpenStreetMap (OSM)[8] maps

[7]https://www.graphhopper.com/
[8]http://www.openstreetmap.org/

and builds the underlying graph to be used for routing. As for clustering, we opted for the ELKI[9] library, which provides an OPTICS implementation with ξ extraction among many others. We make use of parallel computing at all steps where it is possible.

4.2 Simulation Environment and Data Set Description

We evaluate our approach by simulating taxi trips extracted from New York City taxi trip data with two on-demand ride-sharing service implementations, one without and one with meeting points. We interpret each taxi trip as a trip request by setting the pick-up and drop-off location, the desired pick-up time to trip start time and number of passengers to one. The trip data is already sorted by trip start time and our simulations process the requests in the same order. A trip request is *satisfied*, if it can be assigned to a vehicle. We collect different measures in three groups:

- General: Success rate (ratio of number of satisfied requests to all requests in the dataset), request processing duration (how much time it takes until finding a vehicle assignment or until rejecting the request), number of mapped events
- Customer satisfaction: Ride delay (duration difference between actual and direct ride times), waiting time (duration difference between *desired* pick-up time and the time when the customer is actually picked up)
- Service/vehicle costs: Capacity utilization, driven distance per vehicle, number of shared rides

The OSM data used for New York City contains all information and changes up to 2017-04-09T15:01:34Z. We extract trips on Saturday May 11, 2013 from New York City taxi trip data. For the simulations to finish in reasonable time, we prepare three data sets:

1. Picking a time interval with *high density* of requests to represent peak demand (with pick-up times from 19:00 to 20:00 totaling to 30,884 trips).
2. Picking a time interval with *low density* of requests to represent off-peak demand (with pick-up times from 05:00 to 07:00 totaling to 10,388 trips).
3. Reduced number of trips by selecting *every 10th* (from 00:00 to 23:59 totaling to 52,552 trips). The aim is to keep the data distribution as close as possible to the whole day with less data points. Hereby, we assume not to introduce a selection bias.

[9]https://elki-project.github.io/

For clustering (i. e. hotspot detection), we use the day Saturday May 4, 2013 (i. e. a week before the simulation data set). The simulations are run inside a virtual machine configured with 8 vCPUs (Intel Xeon E5-2650 clocked at 2.20 GHz). Used Java Virtual Machine parameters are `-Xms2048m -Xmx4096m -XX:+UseG1GC`.

We evaluate the effect of different parameter values related to meeting points, while keeping the remaining configuration within each test category the same (i. e. number of vehicles, vehicle capacity, slack time) or using the default values for the remaining dimensions (e. g. when testing with various bucket size values, time flexibility is set to 5 min.). In all tests, we set the vehicle capacity to 10 and the slack time to 5 minutes. The number of vehicles is set to 1000 for *high density*, 250 for *low density* and 100 for *every 10th* data sets. These values are chosen in a way to prevent oversaturating the system with resources and in order to better observe the results of meeting points. In addition to already introduced concepts, we investigate the impact of *minPoints* (an OPTICS parameter that defines the number of points required to form a cluster) and *mapping mode*, where we intentionally disable mapping to either hotspots or already existing request events.

4.3 Results and Findings

The simulation results are depicted in Table 1. Some of the key findings are as follows:

- The simulations with the *high density* dataset benefit from meeting points the most, since increasing density of requests increases the potential of events to be bundled.
- The most important factors that influence the results are *time* and *location flexibility*. As depicted in Figure 4, the bigger the flexibilities, the better the success rates.
- The choice of *bucket size* is important, since its value directly affects the number of clusters and how long they remain active. Smaller bucket sizes produce higher numbers of hotspots to which mapped events are distributed whereas we want to concentrate events around a small number of hotspots. The larger the bucket size, the less the number of hotspots which reduces the probability of mappings. As a result, too low or high values for bucket size are both detrimental to the improvement of quality of results.

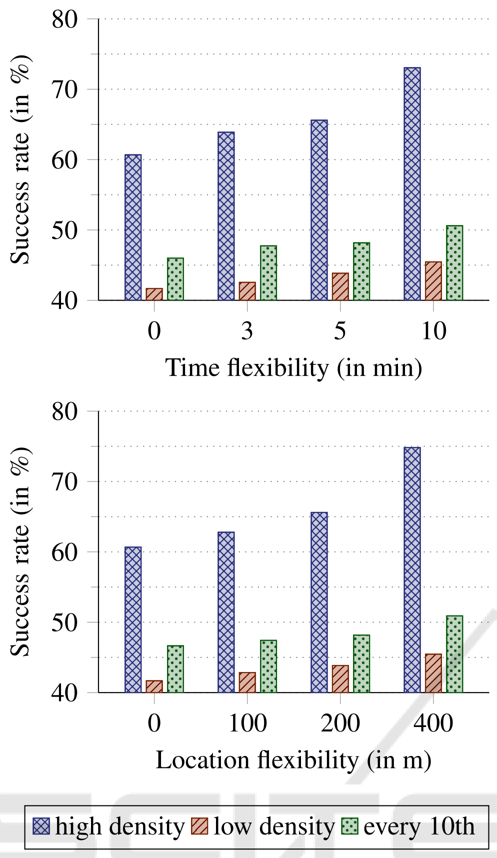

Figure 4: Success rates w. r. t. flexibility changes from Table 1. Time and location flexibilities determine whether an event can be mapped to another event or hotspot. Therefore, when increasing one flexibility (time or location) while keeping other parameters constant, we can observe better success rates (default time flexibility: 5, default location flexibility: 200). The category 0 functions as control group and denotes the success rates without meeting points.

- If *decide level radius* is set to lower values, our clustering post-processing stage becomes more selective and as a result there are less hotspots to map to.
- *minPoints* and ξ directly affect the outcome of OPTICS and therefore the number of clusters. Their values determine the number of clusters which impacts the number of maps to hotspots and the amount of success rate improvement.
- With *mapping mode* we observe that mapping to request events is the main factor that impacts the results and the addition of hotspots is an improvement of these results. Mapping *only* to hotspots is an improvement as well, but does not produce the same high impact. As it turns out, there are many *unnecessary* hotspots maps that do not improve the efficiency but decrease the customer convenience.
- Processing duration per request, capacity utilization and ride delay benefit, for the most part, from inclusion of meeting points.
- The percentage of shared rides decreases marginally and this is in direct correlation with time and location flexibility. With increasing flexibilities we can bundle more events and requests, which in return leave some of the *non-mappable* requests alone during some parts of the schedule. These, otherwise, would have shared the ride with bundled requests.
- It may appear from Table 1 that the driven distances per vehicle increases with the meeting points, but if we put it in perspective with the success rate improvement, the vehicles are actually driving less per satisfied request.

In general, we can conclude that the introduction of meeting points is beneficial regarding the success rate and service/vehicle costs, but comes at a price of customer inconvenience due to slightly increased waiting time and incurring walking legs to the pick-up and/or from drop-off location. In best case, we observe an 14.15% improvement of success rate, when location flexibility is set to 400 meters and time flexibility to 5 minutes (per event). In worst case, the increase of waiting time amounts to 1.3 minutes, when location flexibility is set to 200 meters and time flexibility to 10 minutes (per event).

5 CONCLUSION

The ride-sharing concept aims to assign multiple similar trips to the same vehicle in order to better utilize the resources. However, a naive realization of it might cause many nearby location visits of the same vehicle. This forces the vehicle to make small detours and/or frequent stops, which reduces the overall efficiency of the system. In this work, we investigated the possibility of eliminating small detours and their resulting consequences.

For a better user experience, we employ an *online* ride-sharing approach: The trip requests are not collected beforehand; but are unveiled in real-time, processed in a few seconds and the response about the vehicle assignment is given to the customer. Since the knowledge about the world in which we are operating is always changing, we can only rely on existing data about requests when determining flexible meeting points. Therefore, our solution is based on *mapping* the pick-up and drop-off events of a new request to those of accepted requests in a vehicle schedule, if their spatio-temporal distances are within the speci-

Table 1: Each cell contains a value that represents the result of the simulation with meeting points and another value within parentheses that expresses the difference of the this value from the result of the simulation without meeting points. The green (or red) cells indicate that the simulation with meeting points produced a better (or worse) result. In all tests, we set the vehicle capacity to 10 and the slack time to 5 minutes.

Test category	Test dimension	Test value	Success rate (%)	Avg. processing duration per request (in ms)	Shared rides (%)	Avg. capacity util. (%)	Avg. driven distance per vehicle (in km)	Avg. waiting time (in s)	Avg. ride delay (in s)	Event map counts	Avg. walk distance (in m)
high density (1000 vehicles)	time flexibility (in min) (default: 5)	3	63.89 (+3.21)	758.4 (−108.73)	99.44 (−0.56)	23.05 (+1.05)	56.4 (+0.3)	74 (+15)	124 (+2)	4338	61.41
		5	65.6 (+4.92)	700.65 (−166.48)	98.17 (−1.83)	22.55 (+0.55)	56.98 (+0.88)	98 (+39)	116 (−6)	6601	64.12
		10	73.05 (+12.37)	520.95 (−346.18)	96.44 (−3.56)	21.85 (−0.15)	58.93 (+2.83)	138 (+79)	107 (−15)	11 172	66.32
	location flexibility (in m) (default: 200)	100	62.79 (+2.11)	814.81 (−52.32)	99 (−1)	22.15 (+0.15)	56.81 (+0.71)	75 (+16)	120 (−2)	3172	29.09
		200	65.6 (+4.92)	700.65 (−166.48)	98.17 (−1.83)	22.55 (+0.55)	56.98 (+0.88)	98 (+39)	116 (−6)	6601	64.12
		400	74.83 (+14.15)	434.1 (−433.03)	98.28 (−1.72)	25.13 (+3.13)	57.32 (+1.22)	118 (+59)	119 (−3)	11 781	160.79
	bucket size (in min) (default: 5)	3	66.36 (+5.68)	659.59 (−207.54)	98.2 (−1.8)	22.34 (+0.34)	57.18 (+1.08)	90 (+31)	109 (−13)	5523	62.11
		5	65.6 (+4.92)	700.65 (−166.48)	98.17 (−1.83)	22.55 (+0.55)	56.98 (+0.88)	98 (+39)	116 (−6)	6601	64.12
		10	65.18 (+4.5)	814.92 (−52.21)	98.58 (−1.42)	24.48 (+2.48)	57.31 (+1.21)	77 (+18)	148 (+26)	5819	58.19
	decide level radius (in m) (default: 200)	100	65.47 (+4.79)	716.5 (−150.63)	98.54 (−1.46)	22.8 (+0.8)	57.12 (+1.02)	77 (+18)	114 (−8)	3956	59.58
		200	65.6 (+4.92)	700.65 (−166.48)	98.17 (−1.83)	22.55 (+0.55)	56.98 (+0.88)	98 (+39)	116 (−6)	6601	64.12
		400	65.84 (+5.16)	715.82 (−151.31)	98.62 (−1.38)	22.69 (+0.69)	57.1 (+1)	101 (+42)	122 (0)	7413	67.97
	minPoints (for OPTICS) (default: 20)	10	66.39 (+5.71)	702.21 (−164.92)	98.84 (−1.16)	23.09 (+1.09)	56.62 (+0.52)	117 (+58)	129 (+7)	9288	76.15
		20	65.6 (+4.92)	700.65 (−166.48)	98.17 (−1.83)	22.55 (+0.55)	56.98 (+0.88)	98 (+39)	116 (−6)	6601	64.12
		50	64.95 (+4.27)	732.29 (−134.84)	98.52 (−1.48)	22.68 (+0.68)	57.05 (+0.95)	75 (+16)	114 (−8)	3683	59.51
	ξ (for OPTICS) (default: 0.02)	0.02	65.6 (+4.92)	700.65 (−166.48)	98.17 (−1.83)	22.55 (+0.55)	56.98 (+0.88)	98 (+39)	116 (−6)	6601	64.12
		0.04	66.53 (+5.85)	679.72 (−187.41)	98.52 (−1.48)	22.73 (+0.73)	56.85 (+0.75)	92 (+33)	118 (−4)	6239	62.66
		0.08	65.47 (+4.79)	715.48 (−151.65)	98.56 (−1.44)	22.72 (+0.72)	57.02 (+0.92)	82 (+23)	114 (−8)	4695	60.12
	mapping mode (default: events & hotspots)	events only	65.18 (+4.5)	724.02 (−143.11)	98.53 (−1.47)	22.76 (+0.76)	57.04 (+0.94)	75 (+16)	114 (−8)	3607	59.18
		hotspots only	60.79 (+0.11)	862.4 (−4.73)	100 (0)	22.06 (+0.06)	56.32 (+0.22)	80 (+21)	126 (+4)	3171	62.16
		events & hotspots	65.6 (+4.92)	700.65 (−166.48)	98.17 (−1.83)	22.55 (+0.55)	56.98 (+0.88)	98 (+39)	116 (−6)	6601	64.12
low density (250 vehicles)	time flexibility (in min) (default: 5)	3	42.55 (+0.87)	199.3 (+0.31)	99.59 (−0.41)	14.71 (+0.2)	127.23 (+0.01)	104 (+5)	102 (−2)	235	51.49
		5	43.85 (+2.17)	196.92 (−2.07)	99.21 (−0.79)	14.9 (+0.39)	127.5 (+0.28)	106 (+7)	102 (−2)	399	53.28
		10	46.09 (+4.41)	185.96 (−13.03)	98.08 (−1.92)	15.02 (+0.51)	128.07 (+0.85)	114 (+15)	96 (−8)	726	54.26
	location flexibility (in m) (default: 200)	100	42.85 (+1.17)	198.13 (−0.86)	99.71 (−0.29)	14.94 (+0.43)	126.88 (−0.34)	102 (+3)	104 (0)	175	25.22
		200	43.85 (+2.17)	196.92 (−2.07)	99.21 (−0.79)	14.9 (+0.39)	127.5 (+0.28)	106 (+7)	102 (−2)	399	53.28
		400	45.46 (+3.78)	189.49 (−9.5)	98.81 (−1.19)	15.1 (+0.59)	128.5 (+1.28)	113 (+14)	97 (−7)	763	116.78
	bucket size (in min) (default: 5)	3	43.64 (+1.96)	197.93 (−1.06)	99.27 (−0.73)	14.94 (+0.43)	127.54 (+0.32)	106 (+7)	101 (−3)	365	53.21
		5	43.85 (+2.17)	196.92 (−2.07)	99.21 (−0.79)	14.9 (+0.39)	127.5 (+0.28)	106 (+7)	102 (−2)	399	53.28
		10	43.53 (+1.85)	196.8 (−2.19)	99.27 (−0.73)	14.89 (+0.38)	128.09 (+0.87)	106 (+7)	103 (−1)	372	52.11
	decide level radius (in m) (default: 200)	100	43.41 (+1.73)	198.4 (−0.59)	99.25 (−0.75)	14.84 (+0.33)	127.78 (+0.56)	106 (+7)	102 (−2)	341	50.47
		200	43.85 (+2.17)	196.92 (−2.07)	99.21 (−0.79)	14.9 (+0.39)	127.5 (+0.28)	106 (+7)	102 (−2)	399	53.28
		400	43.36 (+1.68)	194.76 (−4.23)	99.36 (−0.64)	14.9 (+0.39)	126.99 (−0.23)	105 (+6)	102 (−2)	384	52.99
	minPoints (for OPTICS) (default: 20)	10	43.99 (+2.31)	194.59 (−4.4)	99.41 (−0.59)	14.94 (+0.43)	127.34 (+0.12)	109 (+10)	100 (−4)	478	51.27
		20	43.85 (+2.17)	196.92 (−2.07)	99.21 (−0.79)	14.9 (+0.39)	127.5 (+0.28)	106 (+7)	102 (−2)	399	53.28
		50	43.41 (+1.73)	198.65 (−0.34)	99.25 (−0.75)	14.84 (+0.33)	127.78 (+0.56)	106 (+7)	102 (−2)	341	50.47
	ξ (for OPTICS) (default: 0.02)	0.02	43.85 (+2.17)	196.92 (−2.07)	99.21 (−0.79)	14.9 (+0.39)	127.5 (+0.28)	106 (+7)	102 (−2)	399	53.28
		0.04	43.61 (+1.93)	195.51 (−3.48)	99.27 (−0.73)	14.83 (+0.32)	127.45 (+0.23)	107 (+8)	101 (−3)	376	52.41
		0.08	43.74 (+2.06)	197.04 (−1.95)	99.27 (−0.73)	14.91 (+0.4)	127.97 (+0.75)	106 (+7)	102 (−2)	366	52.07
	mapping mode (default: events & hotspots)	events only	43.41 (+1.73)	198.48 (−0.51)	99.25 (−0.75)	14.84 (+0.33)	127.78 (+0.56)	106 (+7)	102 (−2)	341	50.47
		hotspots only	41.67 (−0.01)	198.73 (−0.26)	100 (0)	14.52 (+0.01)	127.14 (−0.08)	99 (0)	104 (0)	46	62.55
		events & hotspots	43.85 (+2.17)	196.92 (−2.07)	99.21 (−0.79)	14.9 (+0.39)	127.5 (+0.28)	106 (+7)	102 (−2)	399	53.28
every 10th (100 vehicles)	time flexibility (in min) (default: 5)	3	47.74 (+1.09)	93.64 (+4.54)	99.94 (−0.06)	14.74 (+0.16)	1239.44 (−5.01)	108 (+3)	102 (+1)	2040	58.68
		5	48.17 (+1.52)	93.4 (+4.3)	99.83 (−0.17)	14.86 (+0.28)	1235.86 (−8.59)	114 (+9)	103 (+2)	3236	60.26
		10	50.6 (+3.95)	90.91 (+1.81)	99.62 (−0.38)	14.67 (+0.09)	1225 (−19.45)	128 (+23)	101 (0)	5529	63.59
	location flexibility (in m) (default: 200)	100	47.42 (+0.77)	94.08 (+4.98)	99.95 (−0.05)	14.7 (+0.12)	1237.04 (−7.41)	109 (+4)	101 (0)	1397	28.46
		200	48.17 (+1.52)	93.4 (+4.3)	99.83 (−0.17)	14.86 (+0.28)	1235.86 (−8.59)	114 (+9)	103 (+2)	3236	60.26
		400	50.9 (+4.25)	89.25 (+0.15)	99.56 (−0.44)	15.46 (+0.88)	1227.51 (−16.94)	128 (+23)	104 (+3)	7102	145.56
	bucket size (in min) (default: 5)	3	47.3 (+0.65)	93.13 (+4.03)	99.84 (−0.16)	14.5 (−0.08)	1241.25 (−3.2)	112 (+7)	99 (−2)	1760	59.7
		5	48.17 (+1.52)	93.4 (+4.3)	99.83 (−0.17)	14.86 (+0.28)	1235.86 (−8.59)	114 (+9)	103 (+2)	3236	60.26
		10	47.78 (+1.13)	96.36 (+7.26)	99.88 (−0.12)	15.77 (+1.19)	1243.9 (−0.55)	107 (+2)	123 (+22)	2682	57.37
	decide level radius (in m) (default: 200)	100	47.62 (+0.97)	92.87 (+3.77)	99.87 (−0.13)	14.7 (+0.12)	1244.11 (−0.34)	108 (+3)	100 (−1)	1033	58.2
		200	48.17 (+1.52)	93.4 (+4.3)	99.83 (−0.17)	14.86 (+0.28)	1235.86 (−8.59)	114 (+9)	103 (+2)	3236	60.26
		400	48.59 (+1.94)	94.99 (+5.89)	99.86 (−0.14)	14.8 (+0.22)	1235.18 (−9.27)	123 (+18)	104 (+3)	5641	67.15
	minPoints (for OPTICS) (default: 20)	10	50.2 (+3.55)	96.04 (+6.94)	99.8 (−0.2)	15.23 (+0.65)	1222.68 (−21.77)	132 (+27)	110 (+9)	8259	72.71
		20	48.17 (+1.52)	93.4 (+4.3)	99.83 (−0.17)	14.86 (+0.28)	1235.86 (−8.59)	114 (+9)	103 (+2)	3236	60.26
		50	47.4 (+0.75)	93.3 (+4.2)	99.82 (−0.18)	14.76 (+0.18)	1242.66 (−1.79)	109 (+4)	101 (0)	897	58.15
	ξ (for OPTICS) (default: 0.02)	0.02	48.17 (+1.52)	93.4 (+4.3)	99.83 (−0.17)	14.86 (+0.28)	1235.86 (−8.59)	114 (+9)	103 (+2)	3236	60.26
		0.04	48.14 (+1.49)	93.04 (+3.94)	99.86 (−0.14)	14.82 (+0.24)	1239.52 (−4.93)	114 (+9)	103 (+2)	2920	58.88
		0.08	47.72 (+1.07)	92.99 (+3.89)	99.82 (−0.18)	14.77 (+0.19)	1241.57 (−2.88)	110 (+5)	102 (+1)	1951	56.03
	mapping mode (default: events & hotspots)	events only	47.32 (+0.67)	93.17 (+4.07)	99.86 (−0.14)	14.8 (+0.22)	1245.88 (+1.43)	108 (+3)	101 (0)	826	56.81
		hotspots only	47.41 (+0.76)	89.43 (+0.33)	100 (0)	14.67 (+0.09)	1231.08 (−13.37)	113 (+8)	103 (+2)	2410	60.34
		events & hotspots	48.17 (+1.52)	93.4 (+4.3)	99.83 (−0.17)	14.86 (+0.28)	1235.86 (−8.59)	114 (+9)	103 (+2)	3236	60.26

fied location and time flexibility thresholds. To reduce the importance of initial requests which might shape the schedule of a vehicle rather unfavourably, we analyse historical request data to determine *hotspots* in order to map initial request events to them, since they function as meeting points as well. For the evaluation, we extract trip requests from New York City taxi trip data and process them in two on-demand ride-sharing simulations (i. e. with and without meeting points). The results indicate that the addition of meeting points is particularly beneficial during peak times when demand is high and dense. We observe that even if passengers are willing to walk short distances, the overall efficiency increases significantly (with regards to success rate and vehicle costs). Decrease in customer convenience can be counterbalanced by service providers offering financial incentives (e. g. cheaper rides). We also identified some drawbacks in our clustering and hotspot decision approach. Since this work was only a starting point, addressing this issue and improving this aspect are part of future work.

REFERENCES

Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. (1999). OPTICS: Ordering Points to Identify the Clustering Structure. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, SIGMOD '99, pages 49–60, New York, NY, USA. ACM.

Czioska, P., Mattfeld, D. C., and Sester, M. (2017). GIS-based identification and assessment of suitable meeting point locations for ride-sharing. *Transportation Research Procedia*, 22:314 – 324. 19th EURO Working Group on Transportation Meeting, EWGT2016, 5-7 September 2016, Istanbul, Turkey.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press.

Gökay, S., Heuvels, A., and Krempels, K. (2018). Heuristics for Improving Trip-Vehicle Fitness in On-demand Ride-Sharing Systems. In *Proceedings of the 4th International Conference on Vehicle Technology and Intelligent Transport Systems, VEHITS 2018, Funchal, Madeira, Portugal, March 16-18, 2018.*, pages 323–334.

Grubesic, T. H., Wei, R., and Murray, A. T. (2014). Spatial clustering overview and comparison: Accuracy, sensitivity, and computational expense. *Annals of the Association of American Geographers*, 104(6):1134–1156.

Hansen, E., Gomm, M., Bullinger-Hoffmann, A., and Moeslein, K. (2010). A community-based toolkit for designing ride-sharing services: The case of a virtual network of ride access points in Germany. *International Journal of Innovation and Sustainable Development*, 5:80–99.

Hawkins, A. J. (2018 – accessed November 27, 2018). Uber Express Pool offers the cheapest fares yet in exchange for a little walking. Available at https://www.theverge.com/2018/2/21/17020484/uber-express-pool-launch-cities.

Li, R., Qin, L., Yu, J. X., and Mao, R. (2016). Optimal multi-meeting-point route search. *IEEE Transactions on Knowledge and Data Engineering*, 28(3):770–784.

Li, X., Hu, S., Fan, W., and Deng, K. (2018). Modeling an enhanced ridesharing system with meet points and time windows. *PLOS ONE*, 13(5):1–19.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif. University of California Press.

Murray, A. and Grubesic, A. (2002). Identifying non-hierarchical spatial clusters. *International Journal of Industrial Engineering : Theory Applications and Practice*, 9(1):86–95.

Schaller, B. (2018). *The New Automobility: Lyft, Uber and the Future of American Cities*. Schaller Consulting.

Stiglic, M., Agatz, N., Savelsbergh, M., and Gradisar, M. (2015). The Benefits of Meeting Points in Ride-sharing Systems. ERIM Report Series Research in Management ERS-2015-003-LIS, Erasmus Research Institute of Management (ERIM), ERIM is the joint research institute of the Rotterdam School of Management, Erasmus University and the Erasmus School of Economics (ESE) at Erasmus University Rotterdam.

Stock, E. (2018 – accessed November 27, 2018). Introducing Express POOL: Walk a little to save a lot. Available at https://www.uber.com/newsroom/expresspool/.

Transportation Research Board (2016). *Shared Mobility and the Transformation of Public Transit*. The National Academies Press, Washington, DC.

Tsubouchi, K., Yamato, H., and Hiekata, K. (2010). Innovative on-demand bus system in Japan. *IET Intelligent Transport Systems*, 4(4):270–279.