A Partition Based Match Making Algorithm for Taxi Sharing

Jiajian Xiao
TUM CREATE Ltd.
1 CREATE Way, #10-02 CREATE Tower
Singapore 138602
jiajian.xiao@tum-create.edu.sg

Heiko Aydt
TUM CREATE Ltd.
1 CREATE Way, #10-02 CREATE Tower
Singapore 138602
heiko.aydt@tum-create.edu.sg

Michael Lees
School of Computer Engineering
Nanyang Technological University
Nanyang Avenue, N4-02a-32
Singapore 639798
mhlees@ntu.edu.sg

Alois Knoll
Department of Informatics
Technische Universität München
Boltzmannstrasse 3
D-85748 Garching bei München, Germany
knoll@in.tum.de

***Abstract*— Taxi sharing can help to improve the utilisation of taxis. Passengers for sharing are always chosen with some objectives in mind, for example, to minimise the inconvenience caused due to de-tours when picking up or dropping off other passengers. In this paper, we describe a method that optimises the match making process in order to minimise the inconvenience imposed on passengers. This method is based on the idea of dividing the road network into several partitions so that a certain quality of service requirement regarding inconvenience is satisfied. More precisely, this can be considered as a constrained optimisation problem. We describe a procedure how to decide on optimal parameters for the partitioning algorithm. In addition, we analyse the theoretical maximum sharing potential of commuters in Singapore using a simulation-based approach.**

I. Introduction

As of 2013 there are more than 28000 taxis operating day and night in Singapore (see Land Transport Authority (LTA)[1]). Despite the high population of taxis in Singapore, many people still find it difficult to get a taxi, in particular during peak hours. As a result, passengers may have some complaints about the service standards[2]. One reason for this kind of problem may be inefficient utilisation of the taxi capacities.

One possible solution to this problem is taxi sharing. In very simple terms, taxi sharing means that several passengers share the same taxi. This may make sense if the passengers travel in the same direction. Either passengers have the same origin and/or destination or they may be picked-up or dropped-off along the way. In addition to improving the usage of taxis, depending on the pricing scheme, it may also bring economical advantages to both, the driver and the passengers. However, passengers will also have to tolerate the inconvenience caused by de-tours that may be necessary in order to pick-up or drop-off other passengers.

[1] www.lta.gov.sg/content/dam/ltaweb/corp/PublicationsResearch/files/FactsandFigures/taxi_info_2013.pdf

[2] http://news.asiaone.com/News/AsiaOne%2BNews/Singapore/Story/A1Story20111210-315524.html

Passengers may not be willing to sacrifice comfort or convenience if they can have a taxi for themselves. However, this situation may look different during peak hours when passengers face the choice between sharing a taxi or not getting a taxi at all. During times of high demand, taxi sharing may be a viable solution in order to improve an important transportation bottle-neck. While issues such as discomfort due to the presence of strangers cannot easily be addressed from an operations perspective, other issues, e.g., the inconvenience caused by de-tours can be improved by optimising the taxi operations.

The selection of passengers for sharing a taxi should not be chosen randomly. They can be selected with a certain objective in mind. For example, one objective could be to minimise the inconvenience caused due to de-tours. This process of matching compatible passengers is further referred to as match making. In general, taxi sharing can be considered as a special case of ride sharing as the only notable difference is that the driver himself is not a passenger and typically not the owner of the vehicle.

Efforts have been made to address the match making problem in the context of taxi sharing and ride sharing applications (see Section II for more details). In this paper, we present a novel approach that allows users to optimise the match making process while minimising the inconvenience. More specifically, our algorithm is based on the concept of partitioning the road network into regions that satisfy certain inconvenience constraints. Match making is then done based on these partitions.

The remainder of this paper is structured as follows. In Section II we discuss existing work. In Section III we describe our partitioning and match making algorithms in greater detail. In Section IV we show how to optimise the parameters of our partitioning algorithm given the specific road network of Singapore. In Section V we further analyse the theoretical sharing potential in Singapore. In Section VI we present our conclusions and discuss future work.

II. Related Work

There are different taxi sharing models which lead to different strategies for conducting match making. Yan et al. [12] classify them into three models based on the deviation of passengers' routes. These models are 1) the many origins to one destination (MOOD) model, 2) the one origin to many destinations (OOMD) model, and the 3) many origins to many destinations (MOMD) model. In addition to these three models, we introduce a fourth basic model which is 4) the one origin and one destination (OOOD) model.

There are a number of commercial products that implement the OOOD model. These applications include mobile phone apps like split-it![3], GoMyWay[4], and shareTransport[5], as well as website-based services such as taxi.de[6]. Most of these operate in a way that users can either create new rides by providing their details (origin, destination, time of departure) and wait for others to join or they can view available trips and choose one to join. Any neutral meeting point should be negotiated before the ride starts. The limitation of these implementations is that users conduct the match making almost manually. In the vHike system, Stach [8] improves the idea by introducing a ride sharing system with enhanced security and privacy. When a new ride is created, besides publishing it on a website, it also sends information to a small range of people via Bluetooth so that users near the origin can be matched.

As for examples of an OOMD model, Chen [1] presents a system in Vehicular Ad-hoc Network (VANET), which optimises the fuel-saving according to the road network and current traffic situation. One highlight of the system is that it conducts match making based on road network partitioning. The road network is divided into sub areas in the form of grids and the system matches passengers not only travelling to the same destination sub area, but also to all sub areas which share the borders with the destination sub area.

We are not aware of pure implementations of the MOOD model. However, there is some work which combines OOMD and MOOD models. For example, Tao [9] implements a taxi-sharing system based on an Intelligent Transportation Systems (ITS). The match making phase is done by enumerating all possible combinations from one origin to all destinations. The combination that leads to the shortest path will be chosen as the final result. The efficiency of the algorithm may be an issue since the complexity of the enumeration and the calculation of shortest path are relatively high. Among all the models, MOMD can be considered the most general. It can also deal with all the situations occurring in the other models. There are several approaches regarding MOMD model.

Gidofalvi and Pedersen [4] use a mathematical approach to solve the match making problem. Given a set of sharing requests (containing expiration time given by the passenger), a maximum taxi-share size k and a minimum saving requirement, the algorithm of Gidofalvi and Pedersen aims at finding combinations of passengers in order to achieve the maximum financial benefit. The algorithm first tries to find the best combination of sharing partners in a group of k for every request. In this case *best* is defined as sharing the longest common route. For all these combinations, the algorithm further tries to find the best one with maximum overall saving. However, taxi fares are approximated by using the linear distance between 2 locations. This approach may lead to sub-optimal results in practice. This is because cost typically depends not only on the distance but also the time travelled which is highly traffic-dependent. The algorithm may contain some inaccuracies, since the taxi fares were estimated by the linear distance between 2 location points.

d'Orey [3] presents a taxi-sharing algorithm with two parts, namely a customer algorithm and a taxi algorithm. The customer algorithm receives user-defined parameters such as ride time or distance from users and sends the information to a selected range of taxis. The taxi algorithm calculates the optimal route with minimum distance by enumerating possible combinations of passengers with different constraints. This includes capability constraints and precedence constraints. The optimal route is returned to the customer algorithm which chooses the one with highest rank value. Geisberger et al. [4] invent a fast detour computation algorithm to minimize the de-tour. Once a new sharing request is generated, it will be compared to all existing requests to find the combination which has the smallest de-tour. They apply a modified version of the routing algorithm so that each new sharing request incurs a small routing calculation. This algorithm focuses on efficiently enumerating all the combinations and finding the optimal solution. However, if the search space is large, exhaustively searching through all possible solutions may become a bottleneck.

Zeng et al. [13] view the problem from a different perspective, which is closely related to our own approach. Their model is based on partitioning the road network into a grid at different levels of granularity ($1km \times 1km$ tiles, $2km \times 2km$ tiles, $5km \times 5km$ tiles, $10km \times 10km$ tiles). The granularity is selected for every single sharing case. There is a rigid route (rigid here means the trip time, origin and destination are fixed, but detours are still allowed) of the car driver and he wants to share this trip. This route is thus expressed in a sequence of tiles which is referred to as a corridor. For match making, only those passengers are considered whose origin and destination lie within the corridor of an existing trip. By partitioning, the algorithm efficiently reduces the size of search space. However choosing an approximate granularity remains an open issue. If the granularity is too big, the search space is still too large, while if too small, interesting matches may be filtered out.

As our match making algorithm is based on partitioning the road network, it is necessary to examine the related work regarding partitioning algorithms. Generally partitioning is conducted with a certain objective, such as balancing the number of vertices, edges or work load, for hierarchical computation, for example.

[3] http://www.split-it.sg/
[4] http://www.gomywayapp.com/
[5] http://www.sharetransport.sg/
[6] http://www.taxi.de/

As for balancing, Möhring et al. [7] compare several partitioning schemes from the point of speeding up Dijkstra algorithm. These schemes include partitioning the road network into grid, Quad-tree, Kd-tree and METIS [6]. These methods put emphasis on balancing the number of vertices or edges within one partition. Wei et al. [10] partition the network for parallel traffic simulation. Their partitioning method aims at balancing the computation work load for each partition. Gonzalez et al. [5] partition the road network for their hierarchical path computation algorithm. Roads are classified according to their width (highways, main roads, neighbourhood roads, etc.). Highways first divide the road network into areas, and main roads further divide the areas into sub-areas.

III. Algorithms

In this section we introduce our partitioning and match making algorithms. The partitioning algorithm divides the road network into a number of partitions. The partitions are generated based on objectives and constraints regarding inconvenience caused due to de-tours. Based on the partitioned road network, a passenger's route can then be described as a sequence of partitions that have to be passed through in order to reach the destination. Sequences of partitions are further referred to as corridors. The match making algorithm compares the corridor of one passenger with the corridor of another passenger. If one corridor is found to be a subset of the other, then these two passengers can possibly share a ride.

The approach described here is similar to the one described by Zeng et al. [13]. However, in contrast to their work, our method comes with a clearly defined procedure to determine the size of the partitions. In addition, partitions are not limited to rectangular shapes with a few pre-defined sizes. Instead, the shape of a partition takes the geographical features of the road network into consideration and allows for arbitrarily shaped partitions. Our approach is thus more flexible as it customises the partitions to the given road network. In this paper we use Singapore as an example. However, due to the flexibility of our approach, it can be applied to other cities as well.

A. Partitioning algorithm

Roads in a road network can be classified into different categories. Typically they are distinguished between various categories of major and minor roads. We consider three categories: a) highways, b) major roads, and c) minor roads. Major roads typically divide a city into several partitions. A partition is thus defined by a set of minor roads surrounded by major roads. This is illustrated in Figure 1, which shows how a partition with a set of minor roads is surrounded by a set of major roads.

Definition of partitions using major roads is not always precise as there may be multiple major roads that could be used as boundaries for a partition. For example, the partition illustrated in Figure 1 is surrounded by multiple, parallel major roads on either side. Pre-processing of the road network data could be done in such a way that close-by major roads are combined in order to make partition definition more precise. However, even when pre-processing the road network data, there may still be cases where partitions cannot clearly be defined. We thus use a different approach that is based on 1) defining a buffer zone around the boundaries, 2) enlarging the size of partitions, and 3) fixing problems with remaining roads. This 3-step approach is explained in greater detail in the remainder of this section.

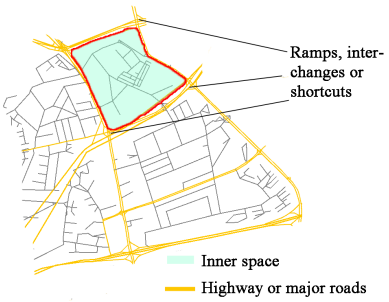

Fig. 1. Example of a partition where a set of smaller roads is surrounded by major roads that define the partition boundaries.

a) Step 1 – Basic Partitioning: The first step is concerned with the generation of bounding shapes for all major roads using buffer and union operations. The buffer operation generates a bounding shape with a round end cap around every single major road. The width of the bounding shape depends on a parameter α as illustrated in Figure 2.

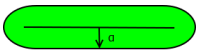

Fig. 2. A buffer operation results in a bounding shape with a width defined by α.

The second operation combines overlapping bounding shapes by performing a union operation as illustrated in Figure 3.

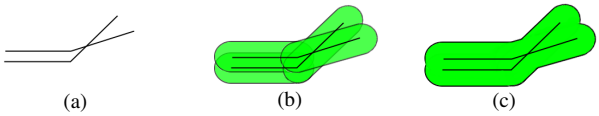

Fig. 3. Illustration of the buffer and union operation. Given a set of major roads (a), the buffer operation is applied which leads to a set of individual bounding shapes (b). These bounding shapes are then combined by the union operation in order to obtain a single bounding shape that contains all major roads (c).

Based on a given road network, buffer and union operations can be applied in order to obtain a buffer zone which effectively divides the road network into partitions. More precisely, the partition is determined by the area enclosed by the buffer zone. This is illustrated in Figure 4.

b) Step 2 – Partition Scaling: After Step 1 we obtain a number of disjoint partitions which exclude all major roads. This is not desirable and partitions are thus scaled according to a parameter β until adjacent partitions overlap[7]. This is

[7]The value of β defines the extent of the overlap.

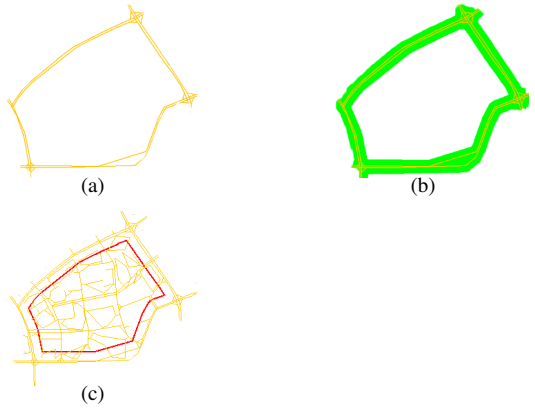

Fig. 4. After the buffer and union operations are applied to a given area with major roads (a), a buffer zone can be obtained (b). The inner ring of this buffer zone is then used to define the boundaries of the partition (c).

necessary in order to improve the success rate of matching compatible passengers that reside at the edges of adjacent partitions. As explained earlier, match making is only possible between passengers whose corridors are subsets of each other. Therefore, without overlapping partitions, passengers that are close-by but residing in two different partitions would never be matched. Figure 5 illustrates the scaling of partitions depending on parameter β.

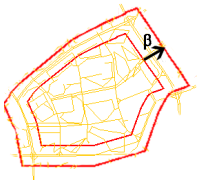

Fig. 5. A partitions is scaled according to parameter β.

As a result, passengers which reside in the outer regions of two adjacent partitions have a higher chance of being successfully matched.

c) Step 3 – Problem Fixing: Our partitioning algorithm is based on geographical operations. As a result, the outcome depends on the geographical characteristics of the road network. It works well if all the partitions are of similar size. However, this is hardly the case with a real-world road network, where there can be partitions of varying sizes. Depending on the road network it is thus possible that small partitions are omitted if the α value is too high. This case is illustrated in Figure 6 where two different values for α result in the loss of a partition.

Another problem that has to be handled occurs in the proximity of the border of the road network. It is possible that minor roads at the border of the network are not included in any partition as illustrated in Figure 7.

Both problems can be addressed in the same way. Instead of applying the buffer operation only on major roads, we apply it to all minor roads which are not already included in a partition. If the buffer of such a road intersects with an existing partition it is merged into this partition using the union operation. In case the buffer does not overlap with

Fig. 6. Illustration of how a partition, indicated in red, is defined by four major roads and their corresponding buffer zones, indicated in green (a). Depending on the α value, i.e., the size of the buffer zones, the size of the partition is affected. If the α value is too large, the partition may disappear altogether (b).

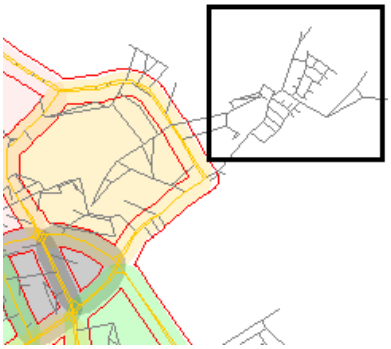

Fig. 7. Example of minor roads that are not included in any partition.

any of the existing partitions, a new partition is created. Other roads which are overlapping with this new partition are merged into it using the union operation.

B. Match Making Algorithm

Based on the partitioned road network, a passenger's route can be expressed as a corridor. Our match making algorithm works as follows. If, at a any point of time, two passengers reside in the same partition, further referred to as origin partition, their corridors are compared in order to see whether one is a subset of the other. If so, then the passengers can possibly be matched. There are in general two ways in which a passenger can reside in a partition. Either the passenger is already on his way in a taxi, which is currently passing through the partition, or the passenger is looking for a taxi but has not found one yet. Match making is possible if at least one of the two passengers is still looking for a taxi. This means that two passengers that are already on their way in separate taxis are not matched even if they are passing through the same partition at the same time.

IV. Partition Optimisation

Taxi sharing imposes a certain degree of inconvenience on passengers due to having to pick-up or drop-off other passengers. For example, consider the case where a passenger wants to travel from point A to point B. On the way a second passenger is picked-up at point C and dropped-off at point D before the taxi reaches its ultimate destination, point B. The additional time required to pick up the second passenger is causing inconvenience I_P and the additional time to drop off the second passenger is causing inconvenience I_D. This

example is illustrated in Figure 8. Depending on the origin and destination of the two passengers, it is possible that I_P and I_D may be zero (e.g., in in cases where both passengers travel from or to the same location). Also, it is possible that more than two passengers share a ride, in which case there is even more inconvenience imposed on the passengers.

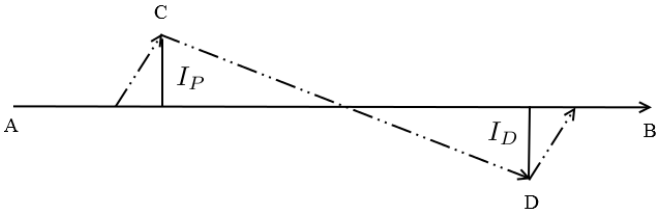

Fig. 8. Example of picking up and dropping off another passenger

From a practical point of view, a taxi operator is assumed to be interested in imposing an upper limit L to the maximum amount of inconvenience I_{max} caused to passengers. The question then becomes how to minimise the average inconvenience I_{avg}, while not exceeding the upper limit. Match making is done based on the notion of partitions and in general, if the partitions are larger, more matches are possible. However, the average inconvenience may be higher due to the size of the partition. Choosing an optimal size for partitions, and thus choosing optimal values for α and β, is crucial. We therefore consider the following optimisation problem:

$$\begin{aligned} \underset{\alpha,\beta}{\text{minimise}} \quad & I_{avg}(\alpha,\beta) \\ \text{subject to} \quad & I_{max}(\alpha,\beta) \leq L. \end{aligned} \tag{1}$$

A solution to this optimisation problem is defined by a pair of values for α and β. In order to evaluate the fitness of a given solution, the inconvenience for each partition has to be determined first. Recall passengers are only matched if they are in the same partition. In the case of taxi sharing, a passenger A would thus have to accept a certain inconvenience in order to pick-up another passenger B in the same partition. The typical inconvenience I_k of a partition k, depends not only on the size of the partition but also on how good different locations in the partition are connected. In addition, the typical speed on the roads in the partition also need to be considered.

For a partition k, we determine the average inconvenience $I_{avg,k}$ and maximum inconvenience $I_{max,k}$ by sampling 30 random origin/destination pairs in this partition. For each pair, the fastest route is determined using a standard Dijkstra algorithm [2]. The weights for the underlying graph used for routing reflect the typical speed for a road link. This may also reflect traffic conditions based on historical or real-time data. Given the total number of partitions n, the overall average inconvenience I_{avg} and maximum inconvenience I_{max} can thus be defined as follows:

$$I_{avg} = \frac{\sum_{k=1}^{n} I_{avg,k}}{n} \tag{2}$$

$$I_{max} = \frac{\sum_{k=1}^{n} I_{max,k}}{n} \tag{3}$$

The first parameter α has a value range of $[55, 550]$ meters. The upper bound for α was chosen based on experience of Singapore and considering the issue of loss of partitions due to large α values. The number of partitions decreases with increasing values for α as illustrated in Figure 9. The resulting coverage of the road network by partitions is negatively affected for large α values.

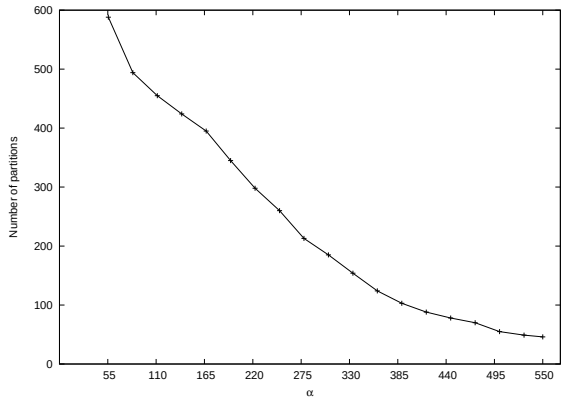

Fig. 9. Number of partitions with increasing α.

As explained above, β is a scaling parameter used to enlarge partitions in order to achieve overlaps between adjacent partitions. Before scaling, the distance between adjacent partitions is $2 \times \alpha$ (this is an artifact of the buffer operation used to generate partitions). Therefore, in order to achieve an overlap between adjacent partitions β values need to be choosen such that $\beta > \alpha$. We define $\beta = \alpha + \gamma$ where $\gamma \geq 0$ indicates the degree to which a partition overlaps into adjacent partitions. For $\gamma = 0$ there is no overlap. The value range of γ is $[0, 275]$. In Section III-A, we explain that the purpose of Step 2 in our partitioning algorithm is to increase sharing opportunities between passengers in adjacent partitions. However, with increasing γ, the overlap will become too large which is not desirable as it effectively enlarges partitions, resulting in increased inconvenience. The upper bound of γ was thus chosen based on experience.

For the experiments, a resolution of 27.5 meters was chosen for α and γ. This results in a total of 20 discrete values for α and 10 discrete values for β. The resulting search space has thus a size of 200 possible (α,β) combinations. Given the relatively small size of the search space, an exhaustive search can be performed to evaluate all possible (α,β) combinations. The results for I_{avg} and I_{max} are illustrated in Figure 11 and Figure 12, respectively. In general, the lowest inconvenience can be achieved by small values for γ. However, as we will explain in Section V, small values for γ lead to poor sharing potential. Intuitively this makes sense, as $\gamma = 0$ means that

Fig. 10. Large α values lead to bad coverage of the road network. In this example, $\alpha = 550$ meters has been used and the resulting partitions are indicated in red.

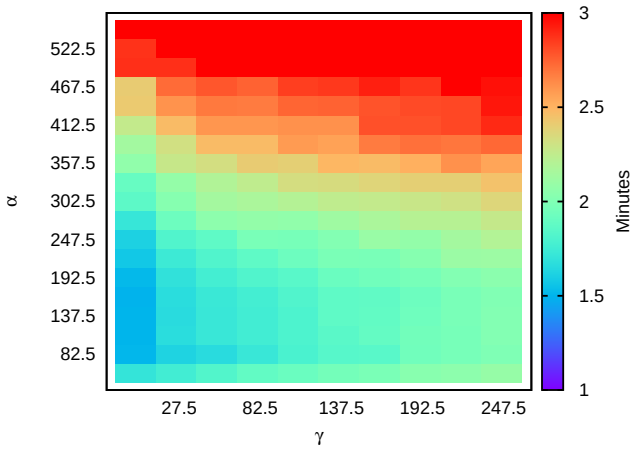

Fig. 11. Experimental results for the average inconvenience I_{avg} depending on α and γ.

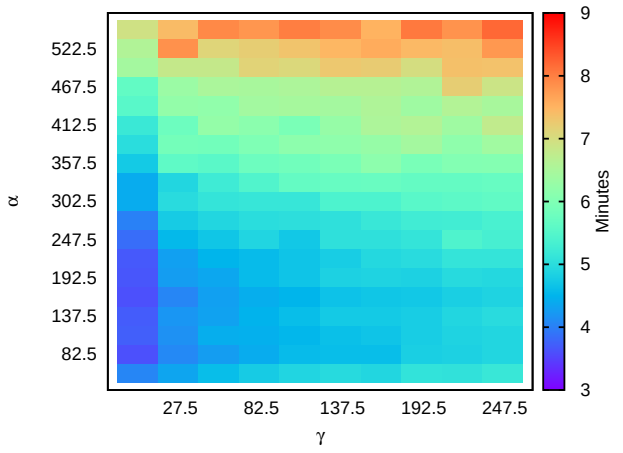

Fig. 12. Experimental results for the maximum inconvenience I_{max} depending on α and γ.

there are no overlaps of partitions, thus reducing sharing opportunities.

We analyse the percentage of partitions that satisfy the constraint that $I_{max}(\alpha, \beta) \leq L$ where the upper bound L has been set to 5 minutes. The result is illustrated in Figure 13. The best solution to the optimisation problem is $\alpha = 137.5$ and $\beta = 137.5$. With these settings, the partitioning algorithm generates a total of 424 partitions with $I_{avg} = 1.51$ minutes and $I_{max} = 3.77$ minutes. The average inconvenience for 99% of the partitions is below the given upper limit of 5 minutes which is considered acceptable for this example. This relaxation of the optimisation constraint is necessary due to the geographical features of Singapore which has a large water reservoir in the centre of the island. This results in a partition which never satisfies the constraint.

V. Sharing Potential

We evaluate our algorithm by considering the resulting sharing potential. If two passengers can be matched according to our match making algorithm, we consider them to have sharing potential. We use a simulation study to evaluate

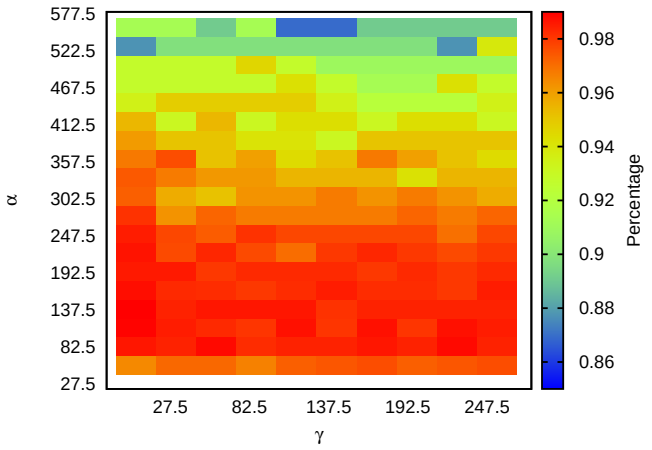

Fig. 13. Percentage of partitions in which the inconvenience is below the upper limit $L = 5$ minutes.

the degree of sharing potential within Singapore when using our match making algorithm. This is done through the application of our match making algorithm on the partitioned road network with the optimized parameters ($\alpha = 137.5, \beta = 137.5$). The simulation test is based on SEMSim [11], an agent-based simulation tool. This tool is used to simulate realistic travel patterns on the Singapore road network based on real data about the commuting patterns and traffic demand in Singapore. The data provides origin-destination pairs of typical traffic demand for a 24 hour period in Singapore. While this data is not necessarily the origin and destination pairs of taxis, we use this as an estimate of the percentage of passengers in Singapore which could potentially share taxis. The percentage of passengers which could potentially share taxis is further referred to as the *sharing potential ratio*.

We generate agents according to the time stamp and frequency of the real world data, each agent then moves from the origin to the destination along the shortest path. Once the simulation starts, every generated agent in the simulation is regarded as a passenger who will travel by taxi. Given the current position and the route of each agent, our match making algorithm can be applied. Each passenger can at most be matched once with another passenger. We don't match passengers when on highways as the picking up or dropping off of passengers on highways is illegal. Obviously, during peak hour demand there can significant levels of traffic, while late at night the amount of traffic will be lower. Therefore the sharing potential ratio is collected over a period of 24 hours in half hour intervals.

In order to determine the sharing ratio, we consider the total number of agents N that are created during a time period t_v. Among these, N_m agents are matched according to our match making algorithm. We can then define the sharing potential ratio as R_{t_v}:

$$R_{t_v} = \frac{N_m}{N} \tag{4}$$

As the trip generation is stochastic it also necessary to repeat the simulation q times, so that an average sharing potential ratio can be obtained. In our experiments $q = 20$ and $t_v = 0.5$ hours. The results are illustrated in Figure 14.

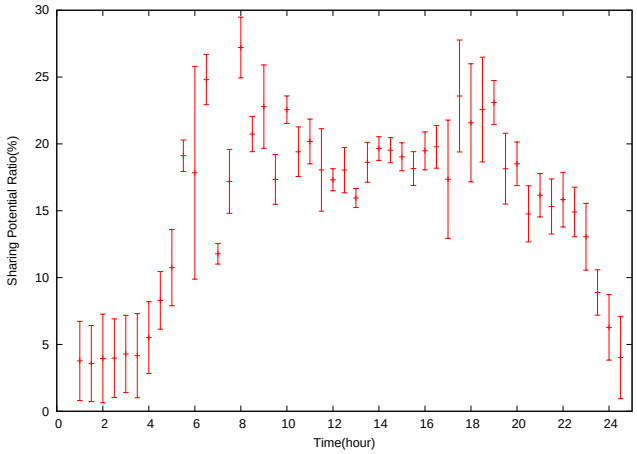

Fig. 14. Sharing potential for $\alpha = 137.5$ and $\beta = 137.5$.

The overall average sharing potential ratio is 15.56%, with the total number of agents in each simulation around 800,000. That means every day nearly 125,000 passengers could potentially share taxis given our match making algorithm. The sharing potential ratio during morning peak hours (7:00 to 9:30) is 19.94%. The total number of agents during morning peak hour is about 205,000, so that means almost 41,000 passengers could share their trips. There is also a good opportunity for sharing around midnight (23:00 to 0:30), with the average sharing potential ratio during this time at 9.4% and the total number of agents around 8,400. In general, the sharing potential depends highly on γ. The relationship between α, γ, and the sharing ratio R_{t_v} is illustrated in Figure 15.

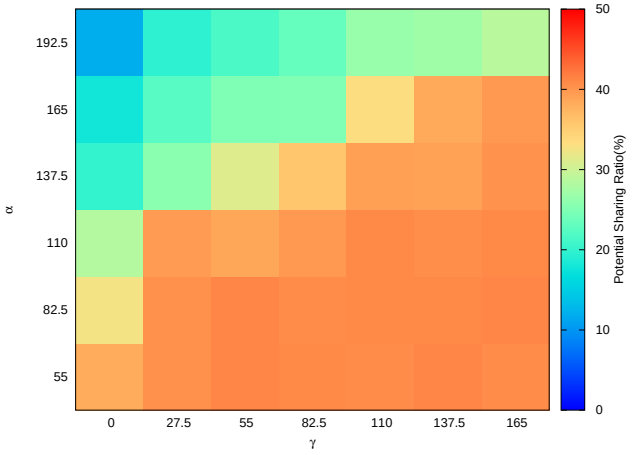

Fig. 15. Evaluation result for sharing potential depending on α and γ.. The results indicate that the sharing potential increases with the degree of partition overlapping (specified by γ).

Our results indicate that there is a trade-off between increasing the sharing potential (see Figure 14) and decreasing the average inconvenience I_{avg} (see Figure 11). However, even for larger values of γ the quality-of-service

constraint $I_{max} \leq L$ where $L = 5$ minutes can be satisfied. This means that there is room for improving the sharing potential by increasing γ at the expense of increasing the average inconvenience I_{avg} while not violating the quality of service constraint. From a practical perspective, a taxi sharing operator would have to decide on a reasonable trade-off between minimising the inconvenience and maximising the sharing potential.

VI. Conclusions and Future work

In this paper, we have presented a match making algorithm for taxi sharing. The match making algorithm is based on partitioning of the road network. A 3-step partitioning algorithm is introduced to partition the road network. By optimising two parameters of the partitioning algorithm, the inconvenience for passengers in each partition is minimised. Finally we use a simulation and real world traffic data to asses the feasibility of the algorithm. The results of these experiments indicate that our algorithm could generate a sizeable sharing ratio. Currently there are a few issues with the partitioning algorithm which could be improved. For example, under certain conditions, partitions can be over-sized. This is particularly a problem with the Bukit Timah area in Singapore. In future work, we plan to improve the partitioning algorithm. Further, due to space constraints, a direct comparison between our approach and existing ones is out of scope. However, future work will be concerned with the evaluation of our approach, compared to existing ones.

As for the actual sharing rate in Singapore, a recent survey conducted by TUM CREATE has resulted in some insights about the acceptance of taxi sharing in Singapore. In general, taxi sharing is currently not very popular. This is mostly due to issues of privacy (sharing a taxi with strangers) and comfort. The partitioning algorithm proposed by us is explicitly considering the inconvenience in terms of detours and attempts to minimise inconvenience caused due to multiple pick-up/drop-off locations. Nevertheless the concerns regarding privacy are more difficult to address. Passengers usually do not mind sharing public buses with strangers. However, as the vehicle size gets smaller, privacy becomes increasingly an issue. In order to improve the acceptance of taxi sharing it may thus be worthwhile considering larger vehicles, such as vans. Despite the currently low acceptance, taxi sharing represents a viable option to alleviate the issue of supply bottlenecks during times of high demand. Increasing the acceptance of taxi sharing among the population, probably requires a serious support by taxi operators, optimised operations (e.g., by using the approach described in this paper), appropriate and transparent pricing that makes taxi sharing also economically attractive to the drivers and better marketing.

References

[1] Po-Yu Chen, Je-Wei Liu, and Wen-Tsuen Chen. A fuel-saving and pollution-reducing dynamic taxi-sharing protocol in vanets. In *Proceedings of the 72nd IEEE Vehicular Technology Conference Fall (VTC 2010-Fall)*, pages 1–5, 2010.

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

[3] P.M. d'Orey, R. Fernandes, and M. Ferreira. Empirical evaluation of a dynamic and distributed taxi-sharing system. In *Proceedings of the 15th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 140–146, 2012.

[4] Gyözö Gidofalvi and Torben Bach Pedersen. Cab-sharing: An effective, door-to-door, on-demand transportation service. In *Proceedings of the 6th European Congress on Intelligent Transport Systems and Services*, 2007.

[5] Hector Gonzalez, Jiawei Han, Xiaolei Li, Margaret Myslinska, and John Paul Sondag. Adaptive fastest path computation on a road network: a traffic mining approach. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB)*, pages 794–805, 2007.

[6] George Karypis and Vipin Kumar. Metis - unstructured graph partitioning and sparse matrix ordering system, version 2.0. Technical report, 1995.

[7] Rolf H. Möhring, Heiko Schilling, Birk Schütz, Dorothea Wagner, and Thomas Willhalm. Partitioning graphs to speed up dijkstras algorithm. *Experimental and Efficient Algorithms*, 3503:189–202, 2005.

[8] C. Stach. Saving time, money and the environment - vhike a dynamic ride-sharing service for mobile devices. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 352–355, 2011.

[9] Chi-Chung Tao. Dynamic taxi-sharing service using intelligent transportation system technologies. In *Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing (WiCom)*, pages 3209–3212, 2007.

[10] Dali Wei, Feng Chen, and Xinxin Sun. An improved road network partition algorithm for parallel microscopic traffic simulation. In *Proceedings of the 2010 International Conference on Mechanic Automation and Control Engineering (MACE)*, pages 2777–2782, 2010.

[11] Yadong Xu, Heiko Aydt, and Michael Lees. Semsim: A distributed architecture for multi-scale traffic simulation. In *Proceedings of the 2012 ACM/IEEE/SCS 26th Workshop on Principles of Advanced and Distributed Simulation (PADS)*, pages 178–180, 2012.

[12] Shangyao Yan, Chun-Ying Chen, and Yu-Fang Lin. A model with a heuristic algorithm for solving the long-term many-to-many car pooling problem. *IEEE Transactions on Intelligent Transportation Systems*, 12:1362–1373, 2011.

[13] Y. Zeng, M. Szczygiel, and B. Honary. Car-share: Making the right connection ride matching. In *Proceedings of 12th Annual Post Graduate Network Symposium (PGNet)*, 2011.