Fast and Efficient Rendezvous in Street Networks

Malika Meghjani, Sandeep Manjanna and Gregory Dudek

***Abstract*— We address the problem of rendezvous between two agents in urban street networks. Specifically, we consider the case where the agents have variable speeds and they need to schedule a rendezvous or a meeting under uncertainty in their travel times. Examples of such a scenario range from everyday life where two people would like to coordinate a meeting while going from office to home; to a futuristic case where automated taxis would like to meet each other for load balancing passengers. The scheduling for such scenarios can easily become challenging with uncertainties such as delayed departures, road blocks due to construction or traffic congestion. Any solution for such a task is required to minimize the waiting time and the planning overhead. In this paper, we propose an algorithm that optimizes the total travel time and the waiting time for two agents to complete their respective paths from start to rendezvous and from rendezvous to goal locations subject to delays along their paths. We validate our approach with a street network database which has a cost associated with every query made to the database server. Thus our algorithm intelligently optimizes for rendezvous trajectories that effectively mitigate the scourge of traffic delays, while simultaneously limiting the number of queries through careful analysis of the informative value of each potential query.**

I. Introduction

Traffic delays have long been a focus of frustration for urban commuters and have lead to countless solutions such as traffic-only radio stations and traffic-aware GPS guidance systems. We consider a robotic planning problem, of rendezvous between pairs of agents in urban streets, where traffic delays also play a key role. Examples of such scenarios are often experienced by both humans scheduling a meeting along their desired paths and robots physically coming together to exchange high bandwidth information or load balancing along their designated routes. A robot's rendezvous planning algorithm can access updated traffic information on-the-fly, thus allowing optimized routes to be considered, however this comes at a cost in terms of communication and planning time. In this paper, we propose an algorithm that intelligently optimizes for rendezvous trajectories that effectively mitigate the scourge of traffic delays, while simultaneously limiting the number of queries through careful analysis of the informative value of each potential query.

In our problem formulation, we consider two agents that are given their respective start and destination locations and they are expected to meet while reaching their destination in the shortest time possible with minimal uncertainty in their travel times as well as reducing the communication cost. This combined optimization problem requires the agents to plan a rendezvous while considering spatial, temporal and communication constraints. The examples of spatial constraints are one-way streets and inaccessible rendezvous locations such as in middle of a creek whereas the temporal constraints may refer to acceptable waiting times by the two agents at the selected rendezvous location. The communication constraints can be represented by the number of queries allowed to a database server which provides routing information of street networks. The desire to minimize the number of queries is important since the access to a centralized server can be both slow and expensive. In addition to these constraints, there are complications due to temporal uncertainties such as delayed departures from the starting locations of the agents, unknown road blocks due to construction-work, or unpredicted traffic congestion.

We model the uncertainty in travel time by considering different possible combinations of agents' speeds with expected delay times proportional to the agents' path lengths. This model also accounts for the temporal constraints on the waiting time while selecting the optimal cost rendezvous location. The spatial constraints are satisfied by applying our scheduling algorithm on a real street network database as illustrated by the example in Fig. 1. Lastly, we optimize the

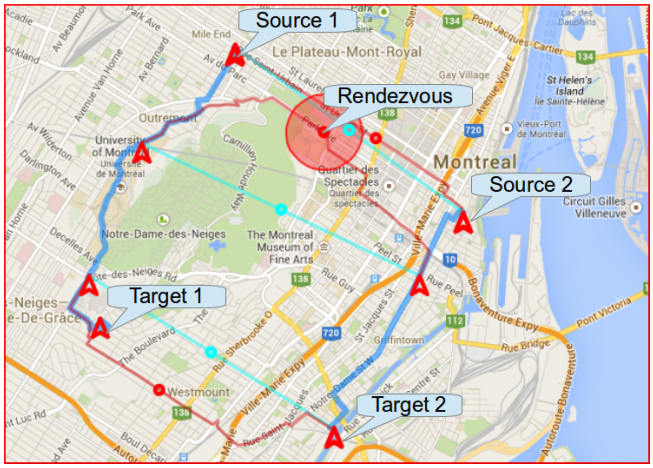


Fig. 1: Our rendezvous planner using Google maps API ©. The individual paths of the two agents are represented by the blue solid lines. Red arrow heads illustrate the waypoints. Small cyan and red circles indicate potential rendezvous points which are computed using Euclidean metric and actual travel times on the road network, respectively. The optimized rendezvous location using our cost function is shown by the center of the big red circular disk.

The authors are with the Centre for Intelligent Machines, McGill University, Montréal, Québec, Canada.
email:{malika, msandeep, dudek}@cim.mcgill.ca

communication constraint by proposing a Hybrid algorithm to lower the number of queries.

Given these constraints, our goal is to minimize the weighted sum of travel and waiting time at the rendezvous location, along with optimizing the number of queries made to the database server. A detailed problem formulation and our proposed approach are presented in Section III and Section IV, respectively. Experimental results on the street network database is presented in Section V.

II. Related Work

The theoretical foundation for the rendezvous problem originated in game theory where it was first applied to simple and abstract environments such as straight lines and random graphs [1], [2]. The problem evolved with technology and created interesting applications in everyday life and robotics. One of the first robotics application for rendezvous was proposed by Roy *et al.* [3] for multi-robot exploration and mapping of unknown indoor environments. Our previous work [4] [5], is motivated by their rendezvous strategies for exploring office-like environments and random graphs with multiple agents without any prior knowledge. Specifically, we define a cost efficient ranking criteria for combining exploration with rendezvous. The problem discussed in this paper however, deals with a known world and plans for an offline rendezvous.

A common approach for multi-agent problems is to optimize the rendezvous locations only with respect to the start locations without considering any concurrent task that must be achieved or the desired end locations of the agents. An example for this case is the energy efficient rendezvous algorithm, proposed in [6] for a team of heterogeneous robots. Their work aims to minimize the total cost of traveling to the rendezvous location by proposing a heuristic in which the local heading of individual robots is iteratively computed, based on the start locations of other robots. This solution was empirically shown to be near optimal by comparing it against the globally optimal solution.

Another application of using multi-agent rendezvous for charging mobile robots was addressed in [7]. The aim of this work was to plan routes for the charger ground robots given the trajectory of the UAV worker robots. This problem was formulated as a directed acyclic graph with vertex partitions containing sets of charging points where rendezvous can potentially occur for each worker robot. The authors proposed a heuristic strategy that involves transforming the graph problem to the traveling salesman problem and then solving a mixed integer linear program using a heuristic solver. Though, these multi-agent rendezvous algorithms are energy efficient they do not account for communication cost or travel time uncertainty.

In the context of route optimization on large Geographical Information Systems (GIS), one of the notable result is presented by Bast *et al.* in [8], where path planning and data access are relevant constraints (without considering the rendezvous process itself). Similarly, [9], and [10] worked on the combined problem of route optimization and rendezvous in large GIS systems. However, their methods neither consider the route optimization post-rendezvous process nor minimize the uncertainty in travel time.

Yan *et al.* [9], discuss an optimal meeting point algorithm for street networks, applicable in the scenario where a tourist bus is deciding on an optimal location to pick up passengers who are at different locations. The optimal meeting point is then selected based on the minimum sum distance criterion. The desired target locations of the passengers are however, not taken into consideration. In contrast, Papadias *et al.* [10] consider the target locations of the agents while optimizing the meeting point. An example would be the case where a group of friends would like to meet for dinner and each one has a different restaurant preference. The proposed algorithm then selects a rendezvous location which minimizes the sum of distances to all the points from a given set of target locations. This problem has some similarity to our work, presented in this paper, with a key difference that our potential meeting locations are not the same as the target locations of the agents.

III. Problem Description and Assumptions

We consider the offline problem of finding the optimal rendezvous location for two asynchronous agents moving in a graph defined by city streets. Each agent has an independent preferred trajectory and we seek a rendezvous location that both agents can visit (where the first to arrive waits for the second). Our approach is based on our previous work [11], where we discretize the agent trajectories into segments of uniform length, and select the corresponding preliminary potential rendezvous locations that are fair with respect to the travel time for each agent. We presume traffic information (i.e. travel time) is collected and managed by a central database and we can obtain this data over a selected road segment at some cost per query. Hence, our optimization criterion is to minimize the weighted sum of travel and waiting time at the rendezvous location, and the number of queries to the database.

We formally describe our problem with two agents A_1 and A_2 who are assigned paths U and V, respectively. These paths are discretized into uniform time segments separated by n waypoints, such that, $U = \{u_1, u_2, ..., u_n\}$ and $V = \{v_1, v_2, ..., v_n\}$ as illustrated in Fig. 2. The potential rendezvous points are selected as the locations where the expected waiting time for the two agents is zero, given that the agents depart from and return to waypoints synchronously. The set of these points is represented by $R = \{r_1, r_2, ...r_n\}$. The paths joining the waypoints to potential rendezvous points are labeled as the *bridge paths* and are denoted by b. The agents are allowed to leave and rejoin their trajectories at any waypoint. This implies that the agents are not bound to their original routes (U, V). Thus our formulation of the problem facilitates the agents to have different speeds.

Our goal is to find a rendezvous point $r^* \in R$ that minimizes the total expected travel and waiting time given

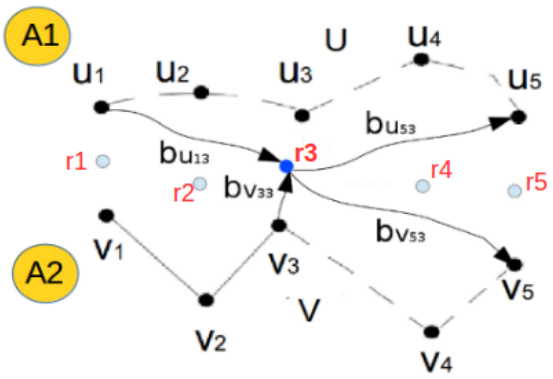


Fig. 2: Paths executed by two agents where agent A_2 is faster than agent A_1 and A_1 skipped all the waypoints.

all the path combinations.

$$r^* = \underset{r_i}{\text{argmin}} \quad T(r_i, j_1, j_2) \tag{1}$$

where, $T(r_i, j_1, j_2)$ provides a set of shortest expected travel times, given a rendezvous point r_i, $i \in \{1, n\}$ and number of skipped waypoints, $j_1, j_2 \in \{0, (n-2)\}$ for the agents A_1 and A_2, respectively. Each of these shortest expected travel times is represented by t^*_{i,j_1,j_2}.

$$t^*_{i,j_1,j_2} = \min_{k_1,k_2} \quad \overline{t_{i,k_1,k_2}} \tag{2}$$

$$\overline{t_{i,k_1,k_2}} = t(u_1, u_{k_1}) + t(u_{k_1}, r_i) + t(r_i, u_{j_1+k_1+1}) + t(u_{j_1+k_1+1}, u_n) + t(v_1, v_{k_2}) + t(v_{k_2}, r_i) + t(r_i, v_{j_2+k_2+1}) + t(v_{j_2+k_2+1}, v_n) \tag{3}$$

where, $t(u_i, u_{i+1})$ denotes the expected travel time between the nodes u_i and u_{i+1}. $k_1 \in \{1, (n-1-j_1)\}$ and $k_2 \in \{1, (n-1-j_2)\}$ are indices for the waypoints in U and V at which the agents A_1 and A_2 leave their respective paths to rendezvous as shown in Fig. 3.

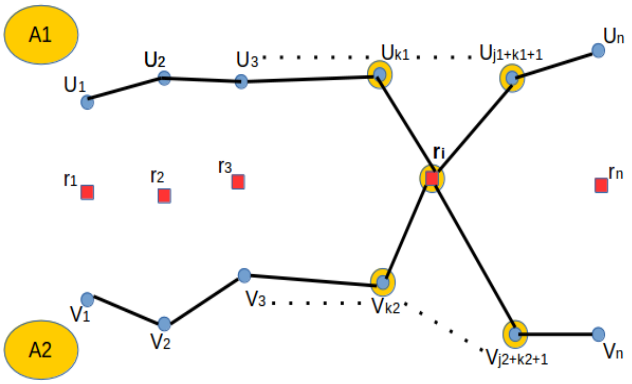


Fig. 3: Agent $A1$ leaves its path at the waypoint U_{k_1} to go for rendezvous at r_i and continues to its path after skipping j_1 waypoints. Agent $A2$ leaves its path at the waypoint V_{k_2} to go for rendezvous at r_i and continues to its path after skipping j_2 waypoints.

In order to obtain an estimate of waiting time for the two agents, we need to account for the travel time and the uncertainty in travel time of the agents. Specifically their travel duration in traffic conditions. Since we do not have access to the actual travel durations in traffic conditions, we model the travel time as a Gaussian random variable:

$$t_{i,k_1,k_2} \sim \eta(\overline{t_{i,k_1,k_2}}, \hat{\sigma}^2_{i,k_1,k_2}) \tag{4}$$

where, $\hat{\sigma}^2_{i,k_1,k_2}$ is an estimate of the variance in the travel time. In our formulation, we select the variance to be proportional to the path length because the uncertainty in travel time increases with the path length. This also allows us to account for delayed departures.

The waiting time is then given by the difference in travel times of the agents between their respective start locations and the rendezvous point. Given that the travel times are Gaussian random variables, the waiting time (t_w) is also a Gaussian with mean as the difference of the mean travel times of the two agents and variance as sum of the variances, as given in Eq. 5.

$$t_w \sim \eta(|\bar{t_1} - \bar{t_2}|, (\hat{\sigma}^2_1 + \hat{\sigma}^2_2)) \tag{5}$$

where,

$$|\bar{t_1} - \bar{t_2}| = |(t(u_1, u_{k_1}) + t(u_{k_1}, r_i)) - (t(v_1, v_{k_2}) + t(v_{k_2}, r_i))|,$$

$$\hat{\sigma}^2_1 + \hat{\sigma}^2_2 = (\hat{\sigma^2_1}(u_1, u_{k_1}) + \hat{\sigma^2_1}(u_{k_1}, r_i)) + (\hat{\sigma^2_2}(v_1, v_{k_2}) + \hat{\sigma^2_2}(v_{k_2}, r_i)).$$

We select a truncated Gaussian distribution to account for the above uncertainty in travel time for simplicity of the analysis. Alternatively, it can be replaced with one of the several statistical distributions presented in the related work on traffic flow modeling [12]. It is interesting to note that in the aforementioned paper, Aron *et al.*, empirically show that mixtures of Gaussian were the best fit to their real data with traffic congestion. However, a Gaussian distribution is unrealistic, because it assigns finite probabilities even to negative travel times, albeit only with very small cumulative probability. Although the difference in travel times is itself a Gaussian with possibly-negative values, the resulting waiting time is the absolute value of this Gaussian, which is strictly a positive quantity. Thus, we transform the Gaussian distribution, representing the waiting time in Eq. 5, to a half-normal distribution, as given below,

$$t_w \sim |\bar{t_1} - \bar{t_2}| + \eta(0, \hat{\sigma}^2_w) \tag{6}$$

$$t_w \sim |\bar{t_1} - \bar{t_2}| + 2\eta_h\left(\hat{\sigma_w}\sqrt{\frac{2}{\pi}}, \quad \hat{\sigma}^2_w\right) \tag{7}$$

where, $\hat{\sigma}^2_w = \hat{\sigma}^2_1 + \hat{\sigma}^2_2$ and η_h is a half-normal distribution (Eq. 7). We finally derive the expected waiting time, $\overline{t_w}$ as,

$$\overline{t_w} = |\bar{t_1} - \bar{t_2}| + 2\hat{\sigma_w}\sqrt{\frac{2}{\pi}} \tag{8}$$

We propose a heuristic cost function to account for both travel time and waiting time. The cost c_{i,k_1,k_2}, of a path where agents A_1 and A_2 depart from waypoints k_1 and k_2 respectively, to reach a rendezvous location

r_i (as in Fig. 3), is given as the sum of total expected travel time $\overline{t_{i,k_1,k_2}}$ (Eq. 4) and expected waiting time (Eq. 8).

$$c_{i,k_1,k_2} = \overline{t_{i,k_1,k_2}} + |\bar{t_1} - \bar{t_2}| + 2\hat{\sigma_w}\sqrt{\frac{2}{\pi}} \tag{9}$$

Therefore, we aim to minimize the cost which is a probabilistic bound on the total expected travel time, and includes the expected waiting time.

IV. Proposed Approach

Our proposed approach requires two pairs of source and target locations to represent the start and end points for the two agents. We query for the shortest path between these pairs of points to obtain the two paths U and V. These paths are truncated to be of the same length and divided into segments of uniform time for synchronizing the travel times of the agents. The midpoints in travel time of the paths joining the corresponding waypoints of the two agents are selected as the potential rendezvous points. These midpoints represent the locations where the expected waiting time for two synchronized agents is zero, given their mean velocities. For the asynchronous agents, the expected waiting time is greater than zero and hence we account this factor while calculating the cost optimal rendezvous location.

In order to find the minimum cost path using Eq. 9, we need to account for all possible combinations of waypoints that the agents can skip. The minimum number of skips, considering one agent and one rendezvous point, is zero which provides $(n-1)$ possible paths. The maximum number of skips are $(n-2)$ which can be traversed in only one way i.e. navigating from the source location to the rendezvous point and returning to the path at the target location. The total number of possible paths for one agent and one rendezvous point is $\frac{n(n-1)}{2}$. A graphical representation of all possible paths for 5 waypoints, one agent and one rendezvous point is presented in Fig. 4. Similarly, the total number of paths

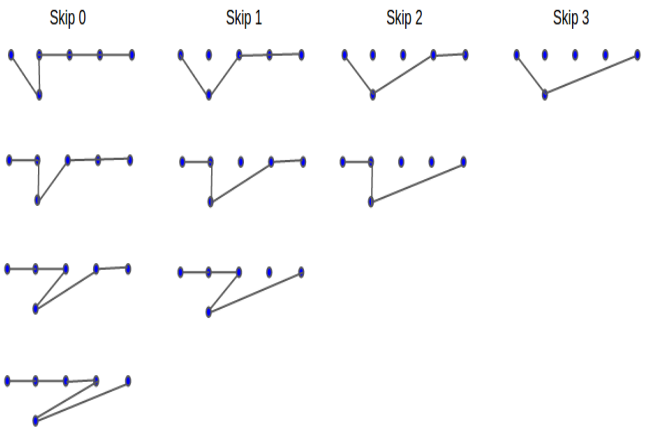


Fig. 4: An illustration of Exhaustive path combinations for one agent and one rendezvous point.

for two synchronous agents having the same speed and $|R|$ rendezvous points is $\frac{n(n-1)}{2}|R|$. For two asynchronous agents, the total number of paths increases to $\frac{n^2(n-1)^2}{4}|R|$, since we need to consider all possible combinations of skips for the two agents. In our problem formulation, there are as many rendezvous points as the number of waypoints, i.e., $|R| = n$. Hence, we need $O(n^5)$ queries to consider all the possible paths.

Querying for all the $O(n^5)$ paths to find the minimum cost path can greatly increase the communication cost. Hence, we propose a *Hybrid* algorithm to efficiently find the minimum cost path with the smallest travel time and waiting time. We compare this minimum cost with the cost of minimum travel time path obtained using two extreme methods, namely,

- Euclidean algorithm and
- Exhaustive algorithm

For the Euclidean algorithm, all paths are compared in Euclidean metric to find the shortest travel time path. The path lengths are measured in Euclidean distance. The travel times corresponding to these paths are obtained using Euclidean distance and speed estimates of the two agents. Similarly for the Exhaustive algorithm, all paths are compared with respect to the actual travel times which are obtained by querying to the street network database server.

Instead of iteratively querying to the database server for the actual travel distance and time for all the $O(n^5)$ paths, we propose a *smart* querying method. This method requires cached queries for only the *bridge* paths. These bridge paths can be added to the required segments from the paths U and V to reconstruct the desired path by parts. Since there are n waypoints and $|R|$ rendezvous points, the total number of bridge paths and hence, the total number of *smart* queries for two agents are only $O(n^2)$.

We further minimize the number of *smart* queries using our *Hybrid* algorithm. The assumption of this algorithm is that the Euclidean distances are good first approximations of the real street distances, as presented in our previous work [11]. Given this assumption, we sort all the *bridge* paths in ascending order of travel time using the Euclidean length of the path and estimated average speed of the agents. We then replace the shortest travel time in Euclidean space with actual travel time by querying the database. This process is performed iteratively, until we receive the shortest path containing all parts in actual travel times.

Since we allow the two agents to have variable speeds, we need to optimize the cost with respect to individual rendezvous points such that the two agents select the same rendezvous location. We achieve this by iteratively applying our algorithm to each set of paths having the same rendezvous point. In addition, we pre-sort the rendezvous points in ascending order of their cost in Euclidean space. A summary of our proposed Hybrid algorithm is presented in Algo. 1.

The best case result of this algorithm returns the path corresponding to the least cost with 4 queries for the 4 *bridge* paths and in the worst case the number of queries can be four times the number of *smart* queries, hence, $O(n^2)$. For calculating the travel time in Euclidean metric, we do not require any queries to the server and so the number of queries

are zero. A comparison of the number of Exhaustive, Smart, Hybrid and Euclidean queries is presented in Table I.

Queries			
Exhaustive	Smart	Hybrid	Euclidean
$O(n^5)$	$O(n^2)$	$O(n^2)$	0

TABLE I: Number of queries

1) **Input**:
   - Endpoint selection:$\{u_1, u_n, v_1, v_n\}$,
   - Path generation: $U(t) = \{u_1, u_2, ..., u_n\}$ and $V(t) = \{v_1, v_2, ..., v_n\}$
2) **Rendezvous selection**: generation of possible rendezvous points $R = \{r_1, r_2, ..., r_n\}$
3) **Query synthesis**:
   - Find all the costs in Euclidean space using Eq. 9: $C = \{C_1, C_2, ..., C_n\}$ where, C_i is the set of costs for all paths through R_i.
   - $Sort(C_i)$ in ascending cost order, $\forall i$.
   - $Sort(C)$ based on the minimum in each set C_i.
   - Assign, $c_{min} = min(C_1)$
   - For each rendezvous point r_i
     - if $(c_{min} > min(C_i))$ then $c_{min} = min(C_i)$, $r_{min} = r_i$
     - if c_{min} is not expressed as street network cost then make it so.
     - else c_{min} is the minimum cost in street network space for the set C_i.
       * Add, c_{min} to the set C_{i+1}
       * Next r_i
   - $c^* = c_{min}$, $r^* = r_{min}$
   - The minimum cost is c^* And, the minimum cost rendezvous location is r^*.
4) **Path execution**

Algorithm 1: Hybrid algorithm

V. Experimental Results

We validate the proposed Hybrid algorithm (Algo. 1) using our web application interface which we developed with Google Maps API © [13]. Our algorithm was applied to the Google street network database for 8 cities around the world. The source and target locations are randomly selected around the city centers for 100 trials per city. We recorded the travel time, waiting time, minimum cost and the number of queries for Euclidean, Exhaustive and Hybrid algorithms.

We compare the cost of the path suggested by our Hybrid algorithm to the cost of the shortest travel time path provided by the Euclidean and Exhaustive algorithms in Table II. It can be observed that the Hybrid algorithm provides an intermediate cost between the underestimated Euclidean cost and the cost corresponding to the actual shortest travel time path provided by the Exhaustive algorithm. The fractional cost improvement achieved by using Hybrid algorithm over the Exhaustive algorithm is presented in the column: "%Reduced". The cost reduced is typically small indicating that the Hybrid algorithm finds paths similar to those produced by the Exhaustive algorithm but with a much more efficient data-access policy. In our experiments, the Hybrid algorithm found the same solution as the Exhaustive algorithm in 47% trials while saving 64% of queries over Smart queries on an average. This indicates that the Hybrid algorithm is efficient in finding the low cost trajectories with minimal database queries.

A comparison of the total number of queries required to find path combinations to the number of Smart and Hybrid queries, along with percentage of queries saved by the Hybrid algorithm over the smart queries is presented in Table II (last column). In general, the percentage of queries saved are correlated to how well the Euclidean algorithm represents the actual travel times. We calculated the number of Smart and Hybrid queries as a function of average number of waypoints $(n = 4.97)$, using Table II. We observed that on average, for 8 cities, the multiplicative factor for Hybrid queries (presented in Table III) is much lesser than the multiplicative factor for the smart queries $(0.75 < 2)$.

Queries		
Total	Smart	Hybrid
$O(n^5)$	$2n^2$	$\mathbf{0.75(n^2)}$

TABLE III: Number of queries from the experimental results

In order to analyze the average travel and waiting time for the minimum cost path provided by the Hybrid algorithm, we compare them individually to the actual shortest travel time path provided by the Exhaustive algorithm. Fig. 5, illustrates a comparison for these average travel times. It can be observed that the Hybrid algorithm finds paths which have travel times similar to the actual shortest travel time provided by the Exhaustive algorithm. The waiting time corresponding to these travel times is presented in Fig. 6. The Hybrid algorithm has shorter waiting time for all the cities when compared to the waiting time of the shortest travel time path provided by the Exhaustive algorithm.

The travel paths of the two agents along with the selected rendezvous locations provided by Hybrid (green blob) and Exhaustive (red blob) algorithms, are presented in Fig. 7. The blob size is proportional to the waiting time for the illustrated instance. The waiting time difference is significant for non-grid like cities, Fig. 7 (e.g. Montreal, Tokyo, Sydney and Hyderabad). Since in non-grid cities, the Euclidean metric fails to provide a good approximation of the street distances, the Hybrid algorithm ends up finding paths which have slightly higher travel times than the actual shortest travel time at the benefit of minimizing the waiting time. For the grid-like cities, Fig. 7 (e.g. New York, San Francisco) the Hybrid algorithm has good approximation of travel time from Euclidean algorithm and hence finds path similar to the actual shortest travel time with similar waiting times.

Cities	Cost				No. of Queries			
	Euclidean	Exhaustive	Hybrid	**%Reduced**	Total	Smart	Hybrid	**%Saved**
Montreal	48.68	59.06	57.49	**2.66**	928.08	55.10	17.68	**67.91**
New York	46.88	57.34	56.39	**1.66**	980.9	57.98	18.32	**68.40**
San Francisco	58.05	67.51	65.35	**3.20**	1574.28	70.92	23.74	**66.53**
Paris	43.6	51.64	50.94	**1.36**	712.51	50.70	16.00	**68.44**
Singapore	47.03	60.52	58.86	**2.73**	548.90	47.46	20.67	**56.45**
Tokyo	32.62	40.86	39.72	**2.79**	135.09	28.72	11.14	**61.21**
Sydney	45.69	53.40	51.77	**3.06**	780.31	53.94	18.78	**65.18**
Hyderabad	47.47	61.43	59.60	**2.98**	768.03	52.28	21.79	**58.32**
Average	**46.25**	**56.47**	**55.01**	**2.56**	**803.51**	**52.14**	**18.52**	**64.06**

TABLE II: Average minimum cost and the corresponding number of queries.

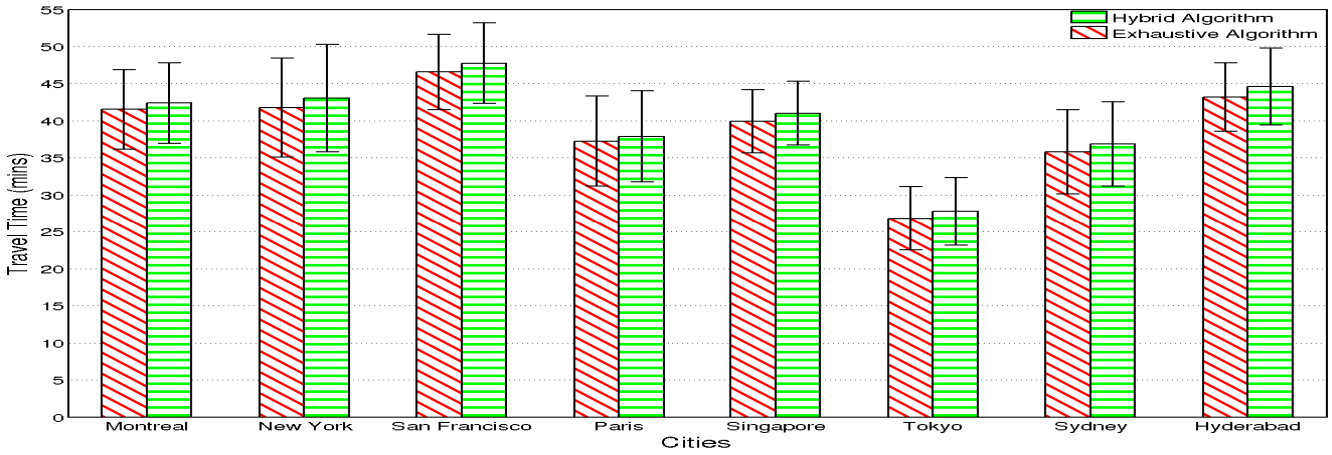


Fig. 5: Average travel time (100 trials).

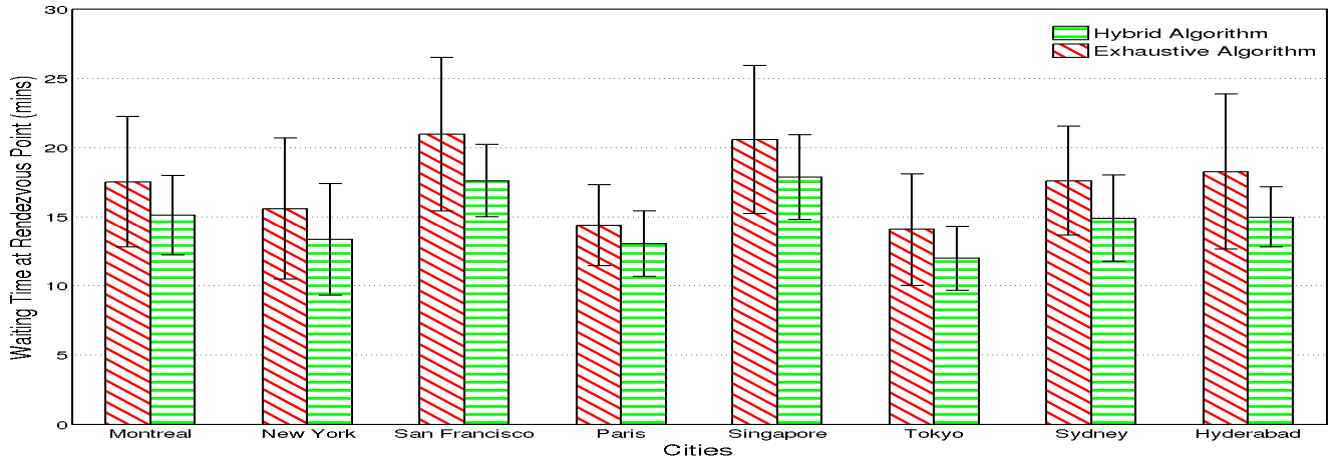
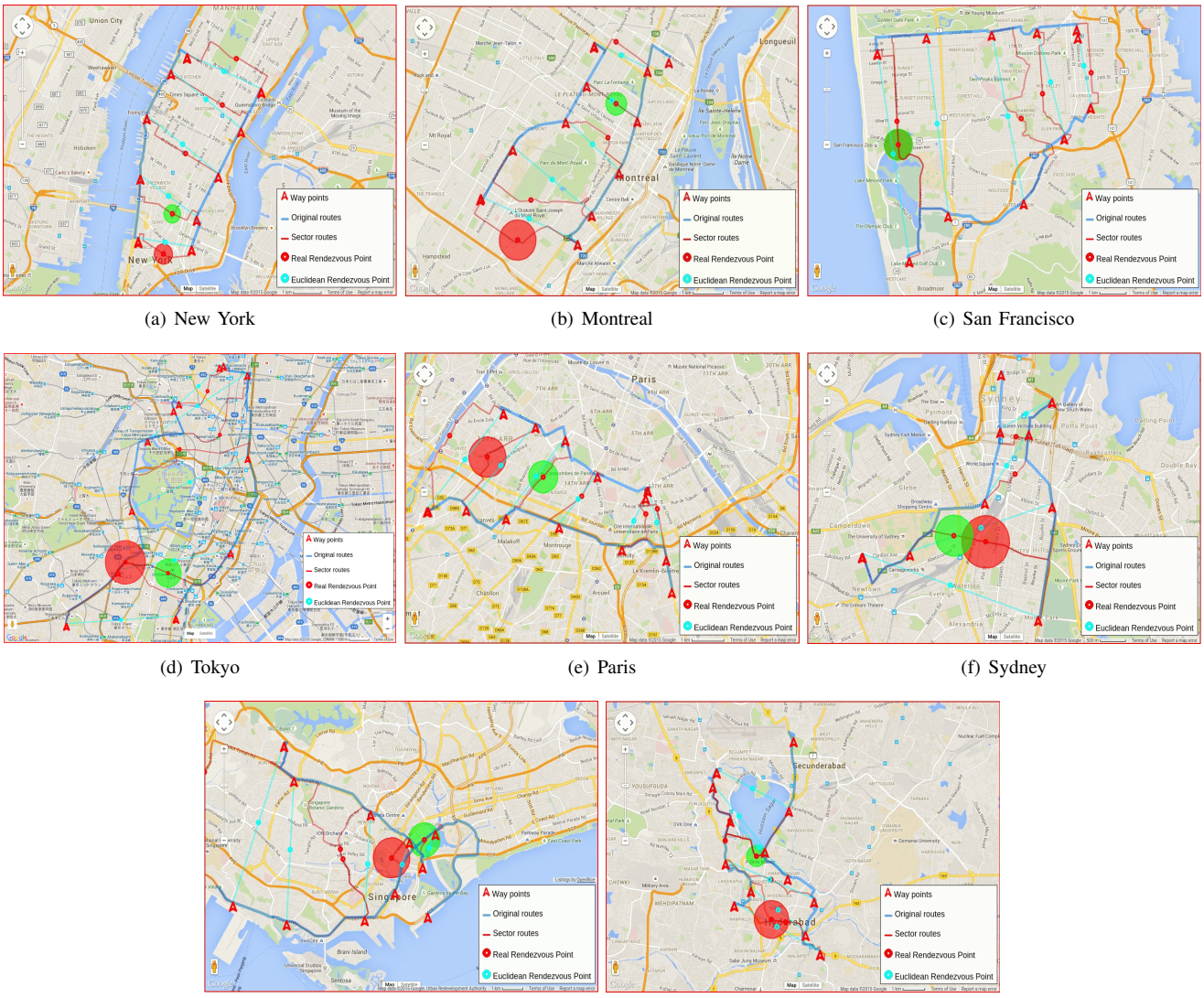


Fig. 6: Average waiting time (100 trials).

VI. Conclusions and Discussions

This paper addresses the rendezvous problem with real agents in real street networks. We proposed a fast and efficient rendezvous algorithm that accounts for expected delays of two agents along their original routes. Our algorithm minimizes the weighted sum of travel and waiting times while reducing the number of database queries, thus making it fast and efficient. We also presented an uncertainty model that guarantees a probabilistic bound on the rendezvous time for two agents. Our experimental results support our hypothesis that with a small increase in the expected travel time, we can achieve reduction in expected waiting time. In addition, we generalized the problem formulation presented in our previous work for synchronous agents [11] to agents with variable speeds. This increased the total number of possible paths and hence the number of queries quadratically. We presented a Hybrid algorithm to reduce the number of queries to the server. In conclusion, we achieved cost optimal rendezvous location with minimum expected travel time, reduced expected waiting time and the least number of queries to the street network database.



(a) New York (b) Montreal (c) San Francisco

(d) Tokyo (e) Paris (f) Sydney

(g) Singapore (h) Hyderabad

Fig. 7: Rendezvous locations selected by Hybrid (green) and Exhaustive (red) algorithms. The blob size represents the waiting time. Examples of rendezvous locations with similar waiting times (a,c,e,g) and high variation in waiting times (b,d,f,h) between Hybrid and Exhaustive algorithms. Google maps API ©

REFERENCES

[1] S. Alpern and S. Gal. The theory of search games and rendezvous. pages 165–178, 2003.

[2] A. Dessmark, P. Fraigniaud, and A. Pelc. Deterministic rendezvous in graphs. *Algorithms-ESA 2003*, pages 184–195, 2003.

[3] N. Roy and G. Dudek. Collaborative robot exploration and rendezvous: Algorithms, performance bounds and observations. *Autonomous Robots*, 11(2):117–136, 2001.

[4] M. Meghjani and G. Dudek. Combining multi-robot exploration and rendezvous. In *CRV '11: Proceedings of the 2011 Canadian Conference on Computer and Robot Vision*, pages 80–85. IEEE Computer Society, May 2011.

[5] M. Meghjani and G. Dudek. Multi-robot exploration and rendezvous on graphs. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages –. IEEE, 2012.

[6] P. Zebrowski, Y. Litus, and R.T. Vaughan. Energy efficient robot rendezvous. In *Computer and Robot Vision, 2007. CRV'07. Fourth Canadian Conference on*, pages 139–148. IEEE, 2007.

[7] Neil Mathew, Stephen L Smith, and Steven L Waslander. A graph-based approach to multi-robot rendezvous for recharging in persistent tasks. In *IEEE Int. Conf. on Robotics and Automation, Karlsruhe, Germany*, 2013.

[8] Hannah Bast, Erik Carlsson, Arno Eigenwillig, Robert Geisberger, Chris Harrelson, Veselin Raychev, and Fabien Viger. Fast routing in very large public transportation networks using transfer patterns. In *Algorithms–ESA 2010*, pages 290–301. Springer, 2010.

[9] D. Yan, Z. Zhao, and W. Ng. Efficient algorithms for finding optimal meeting point on road networks. *Proceedings of the VLDB Endowment*, 4(11), 2011.

[10] D. Papadias, Q. Shen, Y. Tao, and K. Mouratidis. Group nearest neighbor queries. In *Data Engineering, 2004. Proceedings. 20th International Conference on*, pages 301–312. IEEE, 2004.

[11] M. Meghjani and G. Dudek. Multi-agent rendezvous on street networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, 2014.

[12] Maurice Aron, Neïla Bhouri, and Younes Guessous. Estimating travel time distribution for reliability analysis. In *Transport Research Arena (TRA) 5th Conference: Transport Solutions from Research to Deployment*, 2014.

[13] Gabriel Svennerberg. *Beginning Google Maps API 3*. Apress, 2010.