Rendezvous on street networks

Malika Meghjani and Gregory Dudek

***Abstract*— We present an algorithm for finding a distance optimal rendezvous location with respect to both initial and target locations of two mobile agents. These agents can be humans or robots, who need to meet and split while performing a collaborative task. Our aim is to embed the meeting process within a background activity such that the agents travel through the same rendezvous location while taking the shortest paths to their respective target locations. We analyze this problem in a street network scenario where the agents are given their individual scheduled routes to complete with an underlying common goal. The total number of path combinations that the agents need to evaluate for the shortest path increases rapidly with the number of waypoints along their routes. We address this computational cost by proposing a combination of Euclidean and street network distances for a trade-off between the number of queries and a distance optimal solution.**

I. Introduction

We examine the two agent rendezvous problem in urban environments where mobility constraints must be taken into consideration. The key problem is to allow agents to execute self-selected trajectories defined by a set of waypoints, while also allowing them to meet (i.e. rendezvous) at some point during their travels. We propose an energy efficient rendezvous algorithm from our prior work [1] that combines this meeting process with the background activity which is represented by the sequential visitation of the waypoints. Examples of this kind of optimization are often encountered in everyday life, when a person would like to meet a friend on their way from office to home, or more hypothetically in the future where industrial robots are organizing the warehouse and would like to meet each other for load balancing, or when automated taxis need to load balance passengers.

II. Rendezvous Problem

We consider the following scenario for analyzing our rendezvous problem. Two agents, A_1 and A_2, are assigned two paths, $U(t)$ and $V(t)$ respectively, to follow as part of their task. Let $U(t) = \{u_1, u_2, ..., u_n\}$ and $V(t) = \{v_1, v_2, ..., v_n\}$ be the discrete representation, as sequences of waypoints, of these two paths. Let $R = \{r_1, r_2, ...r_n\}$ be the set of potential rendezvous locations. These locations are the midpoints of the waypoints that define the two paths. The agents are initially located at u_1 and v_1 and their desired target locations are u_n and v_n. We design our problem such that the agents depart from their original paths at locations u_k and v_k, at the same time, attempt to meet at a rendezvous point r_i and return to their respective routes at some waypoints, u'_k and v'_k. The total distance traveled by the two agents from their source to target locations is required to be minimized with respect to the rendezvous point. A graphical illustration of our problem is given in Figure 2.

The authors are with the School of Computer Science, McGill University, Montréal, Québec, Canada. email:{malika, dudek}@cim.mcgill.ca

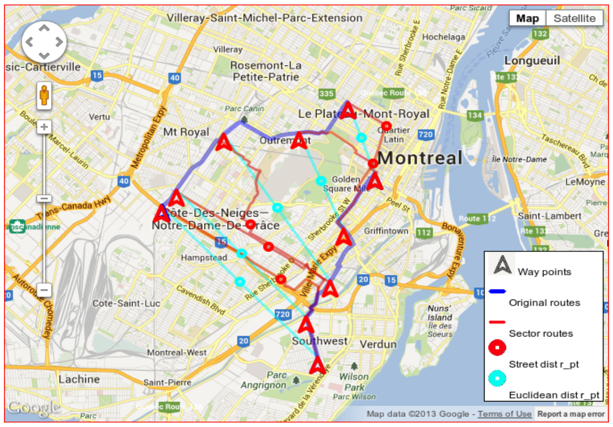

Fig. 1: An instance of rendezvous planner using Google maps API ©.

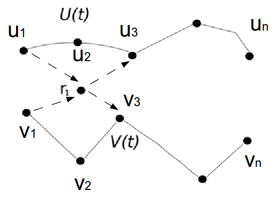

Fig. 2: $U(t)$ and $V(t)$ are the original paths of the agents with $\{u_1, v_1\}$ and $\{u_n, v_n\}$ as the source and target locations. In this example r_1 is the rendezvous location corresponding to the shortest path: $\{u_1, r_1, u_3, ..., u_n\}$ for agent A_1 where the waypoint u_2 is skipped.

For any particular agent, given a rendezvous location r_i, and a task path that consists of n waypoints, the number of possible paths that visit the rendezvous point is $\frac{n(n-1)}{2}$. This is explained by exhaustively considering all possible combinations of the rendezvous points and number of waypoints that the agent skips during its route. The agent can either choose to visit all of its waypoints and there will be $(n-1)$ possible paths to choose from or consider skipping one waypoint with $(n-2)$ options and so on until skipping all the waypoints with only one possible path from the source u_1 to rendezvous point r_i and returning to the original path at final waypoint u_n. Hence, the total number of paths considering one agent and all the potential rendezvous locations is $O(n^3) = \frac{n(n-1)}{2} * |R|$.

Table I: A comparison of distances and number of queries for Euclidean, smart and hybrid methods

Cities	Euclidean distance (km)	Real distance (km)	Hybrid distance (km)	No. of queries Total	Smart	Hybrid	Saved	Euclidean rpt accessible?
Montreal	2.8	3.4	3.4	27	9	4	5	No (Park)
New York	2.9	3.3	3.3	27	9	6	3	Yes
San Francisco	6.0	7.4	7.4	125	25	17	8	Yes
Paris	7.3	8.0	8.0	64	16	6	10	Yes
Singapore	12.9	15.0	15.0	512	64	50	14	Yes
Hong Kong	6.9	11.5	11.5	125	25	31	-6	No (Forest)
Tokyo	8.3	9.2	9.2	125	25	5	20	Yes
Sydney	7.9	9.0	9.0	125	25	11	14	No (Harbor)
Hyderabad	5.5	6.9	6.9	64	16	9	7	No (Lake)
Mexico City	19.2	21.4	21.4	729	81	38	43	Yes
Average	7.9	9.5	9.5	192.3	29.5	17.7	40%	—

We would like to find the rendezvous point r^* that minimizes the total path length among all the combinations for the rendezvous points r_i where $i \in \{1, n\}$ and number of skips $j \in \{0, (n-1)\}$ i.e.

$$r^* = \underset{r_i}{\operatorname{argmin}} \quad D(r_i, j) \tag{1}$$

where $D(r_i, j)$ is a function that returns $p^*_{i,j}$ which is the minimum path length for a given rendezvous point r_i and number of skips j.

$$p^*_{i,j} = \min_k \quad p_{i,k} \tag{2}$$

$$p_{i,k} = b_{i,k} + b_{i,(j+k+1)} + d(u_1, u_k) + d(u_{(j+k+1)}, u_n) \tag{3}$$

where $k \in \{0, (n-1-j)\}$ and d is the distance function. The value $b_{i,k}$ represents the *bridge distance* between rendezvous point r_i and u_k in the k^{th} option for j skips.

III. Rendezvous Algorithm

We analyzed the offline rendezvous problem on street networks using the Google maps API [2]. Given the original routes divided into segments and the potential rendezvous locations, we exhaustively list $O(n^3)$ combinations of possible paths for one agent at a time. Since the number of combinations increase very fast with number of waypoints, we propose a smart query reduction method. According to this method we only require $O(n^2)$ queries to the server to enumerate all the $O(n^3)$ path lengths for one agent. Specifically, we query only the *bridge distances* and combine them with the segment lengths from the original routes to obtain the total path length by parts for all the combinations.

A further reduction in the number of queries can be achieved by considering an inherent property of the street networks. On the street networks, the Euclidean distances provide a lower bound on the street network distances. We leverage this fact and propose a hybrid combination of Euclidean and street network distance measures for calculating the path lengths. Specifically, we obtain a list of Euclidean distances for all the *bridge paths* (paths towards and away from rendezvous point) and arrange them in ascending order. The *bridge path* corresponding to the shortest path in Euclidean distance is re-evaluated for street network distance. If the street network distance is still the minimum in the list of *bridge path* lengths then the rendezvous location along this path is selected. Otherwise, the process is repeated until the shortest *bridge paths* are in street network distances. This process can reduce the number of queries to as low as 2 for two *bridge distances* ($b_{i,k}$, $b_{i,(j+k+1)}$) in the best case and less than twice the number of smart queries in worst case.

IV. Experimental Results

We evaluated our algorithm on urban street network maps from 10 different cities around the world. The results are compared based on Euclidean distances, street network distances and hybrid combination of Euclidean and street network distances. The criteria for comparison are the shortest path length from source to target location passing through rendezvous location, number of queries made to the server and whether the suggested Euclidean distance based rendezvous location is accessible. A summary of the results is given in Table I.

We observed that the Euclidean based measures were a close approximation to the street network path lengths. However, the Euclidean distance based measurements can return rendezvous locations which may not be accessible whereas real-street distance measurements require several queries to the server. Hence, we use our hybrid algorithm which minimizes the total number of queries and provides accessible rendezvous locations with the same length of the path as obtained by smart queries. The average fraction of queries saved using the hybrid algorithm in 10 scenarios, that we illustrated is 40%.

V. Discussions and Conclusions

This work provides a proof of concept for a novel two agent rendezvous problem in street networks which bridges the gap between pure theory based simulations and real applications using our web application framework. In addition, it also addresses the issue of reducing expensive query cost to the server for finding the shortest path by combining Euclidean and street network distance measurements. Our experimental results suggest that the Euclidean based measurements can provide a good first approximation of the street network distances hence making the planning cost very reasonable.

References

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

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