On the Online Dial-A-Ride Problem with Time-Windows

Fanglei Yi, Yinfeng Xu and Chunlin Xin
School of Management, Xi'an Jiaotong University,
Xi'an, ShaanXi 710049, P. R. of China
fangleiyi@163.com

Abstract

In this paper the first results on the Online Dial-A-Ride Problem with Time-Windows (ODARPTW for short) are presented. Requests for rides appearing over time consist of two points in a metric space, a *source* and a *destination*. Servers transport objects of requests from sources to destinations. Each request specifies a deadline. If a request is not be served by its deadline, it will be called off. The goal is to plan the motion of servers in an online way so that the maximum number of requests is met by their deadlines. We perform competitive analysis of two deterministic strategies for the problem with a single server in two cases separately, where the server has unit capacity and where the server has infinite capacity. The competitive ratios of the strategies are obtained. We also prove a lower bound on the competitive ratio of any deterministic algorithm of $\frac{2-T}{2T}$ for a server with unit capacity and of $\frac{2-T}{2T}\lceil\frac{1}{T}\rceil$ for a server with infinite capacity, where T denotes the diameter of the metric space.
Keywords : Logistics, Vehicle Scheduling, On-Line Algorithms, Competitive Analysis

1 Introduction

In dial-a-ride problems (DARP for short) servers are traveling in some metric space to serve requests for rides. Each ride is characterized by two points in the metric space, a *source*, the starting point of the ride, and a *destination*, the ending point of the ride. The problem is to design routes for the servers through the metric space, such that all requested rides are made and some optimality criterion is met. A common characteristic of almost all the approaches to the study of the problem is the off-line point of view. The input is known completely beforehand. However, in many routing and scheduling applications the instance only becomes known in an online fashion. In other words, the input of the problem is communicated in successive steps.

In a natural setting of dial-a-ride problems requests for rides are presented over time while the servers are enroute serving other rides, making the problem an online optimization problem. Examples of such problems in practice are taxi and minibus services, courier services, and elevators.

In this paper we consider a class of variations of online DARP in which there is a time-window on each of the requests: a server moves from point to point in

a metric space. Time is continuous and at any moment a request can arrive at a point in the space, requiring the server to carry the objects to the destination. Each request also specifies a deadline. If a request is to be served, the server must reach the point where the request originated during its time-window (the time between the request's arrival and its deadline). The goal of the algorithm is to serve as many incoming requests as possible by their deadlines. The online algorithm for online DARP neither has information about the release time of the last request nor has the total number of requests. It must determine the behavior of the server at a certain moment t of time as a function of all the requests released up to time t (and the current time t). In contrast, an off-line algorithm has information about all requests in the whole sequence already at time 0.

The ODARPTW and in general vehicle routing and scheduling problems have been widely studied for more than three decades (see[13] for a survey on the subject). We are inspired by recent exciting results in online routing and scheduling problems[4-6,10]. Most previous researches on online routing problems focused on the objectives of minimizing the makespan [4,5,7], the weighted sum of completion times [4,12], and the maximum/average flow time [11,13]. In the paper [6, 15, 16], results on the online k-taxi scheduling problem have been presented, in which a request consists of two points (a *source* and a *destination*) on a graph or in a metric space. Subsequently, a similar problem, online k-truck scheduling problem has been studied in [10]. Both of them (online k-taxi/truck scheduling) assumed that k servers (taxies or trucks) are all free when a new service request occurs, and the goal is to minimize the total distance traveled by servers. [4] studied the online DARP in which calls for rides come in while the server is traveling. The authors also considered two different cases, where the server has infinite capacity and where the server has finite capacity. Lipmann et al. [14] studied the online DARP under a restricted information model in which the information about the destination will be released (becomes known) while visiting the source. All of these previous work assumed that the requests could wait for any length of time until the server completed them. Results on online routing and scheduling problems, in which a certain job will be called off after waiting for a certain period of time, were presented in [8] firstly, studying the dynamic traveling repair problem, a degenerate form of ODRPTW presented in this paper. Subsequently, Krumke et al. [9] studied the similar problem in special case (such as in uniform metric space). We will pay our attention to the more general case that the requests for rides which have time-windows consist of two points (a *source* and a *destination*) in a general metric space. The goal is to serve as many requests as possible within their time-windows. We will give upper bounds for the competitive ratio of two algorithms. And several lower bounds for any deterministic algorithm will be shown in this paper.

2 The Model

Let $\mathcal{M} = (X, d)$ be a metric space with n points which is induced by an undirected unweighted graph $G = (V, E)$ with $V = X$, i.e., for each pair of points from the metric space $\mathcal{M}$ we have $d(x, y)$ that equals the shortest path length in G between vertices x and y. We also assume that $d(x, y) \leq d(x, z) + d(z, y)$

for all $x, y, z \in X$. An instance of the basic online DARP in the metric space $\mathcal{M}$ consists of a sequence $R = (r_1, r_2, \cdots, r_m)$ of *requests*. Each request is a triple $r_i = (t_i, a_i, b_i) \in \mathcal{R} \times X \times X$ with the following meaning: t_i, a real number, is the time that request r_i is released; $a_i \in X$ and $b_i \in X$ are the source and destination, respectively, between which the object corresponding to request r_i is to be transported. It is assumed that the sequence $R = (r_1, r_2, \cdots, r_m)$ of requests is given in order of non-decreasing release times, that is, $0 \leq t_1 \leq t_2 \leq \cdots \leq t_m$. A request is said to be *accepted request* if the corresponding object is picked up by the server at source, and a request is said to be *completed request* if the corresponding object is transported to the destination. We do not allow *preemption*: it is not allowed to drop an accepted request at any other place than its destination. This means, once a request is accepted, it will not be called off.

In this paper, we consider the following restrictions to ODARPTW: a) The speed of the server is constant 1. This means that, the time it takes to travel from one point to another is exactly the distance between the two points; b) The window sizes for all requests are uniform. We will normalize all values so that the window length is 1 for the remainder of this paper; c) The diameter of the metric space is bounded by a constant T, which is the maximum time required to travel between the two farthest points in the metric space.

We evaluate the quality of online algorithms by competitive analysis [1-3], which has become a standard yardstick to measure the performance. In competitive analysis, the performance of an online algorithm is compared to the performance of the optimal off-line algorithm, which knows about all future jobs. An algorithm A for online DARP with time-windows is called α-*competitive* if for any instance R the number of request completed by A is at least $1/\alpha$ times the number of request completed by an optimal off-line algorithm OPT.

3 Online Algorithms and Competitive Analysis

In this section we propose two algorithms, REPLAN and SMARTCHOICE, for ODARPTW, which are similar to BATCH algorithm and Double-Gain algorithm in [8]. And the performance guarantees of the two algorithms for the problem in a general metric space are shown in this section.

3.1 The REPLAN algorithm (RE for short)

We will divide each step into *basic plan*. We shall think of basic plan as taking a certain interval of time to serve a certain subset of requests planed at the beginning of the interval. The length of time intervals is 1/2. At the beginning of each interval, the server stops and replans: it computes a optimal schedule with 1/2 length starting at the current position of the server, which can completes the maximum number of requests which arrived in the previous interval. Then it continues to use the new basic plan.

Theorem 1 *Provided that the server is unit-capacity* ($z = 1$)*, for any metric space with* $T < 1/2$*, RE is* $4(\frac{1+2T}{1-2T})$*-competitive.*

Proof. The server has capacity 1, i.e., it can carry at most 1 object at a time. Once the serve has accepted a request, it is not allowed to accept another request until the *accepted request* is completed. Let R_i be the set of all requests arriving during interval i, and let $R_{i,j}$ be the subset of R_i that OPT completed during

interval j. Obviously, $R_{i,j} = \emptyset$ for all $j \notin \{i, i+1, i+2, i+3\}$. Let $B_{OPT}(R)$ denote the number of requests completed by OPT on a request sequence R. Suppose the last request arrives in m^{th} interval, then

$$B_{OPT}(R) = \sum_{i=1}^{m} (|R_{i,i}| + |R_{i,i+1}| + |R_{i,i+2}| + |R_{i,i+3}|)$$

where $R_{i,i} \cup R_{i,i+1} \cup R_{i,i+2} \cup R_{i,i+3} \subseteq R_i$

When RE makes the basic plan at the beginning of interval i, it takes care of the set R_{i-1}, finds a new schedule and follows it. One of options for RE is to pick the largest of $R_{i-1,i-1}, R_{i-1,i}$, $R_{i-1,i+1}$ and $R_{i-1,i+2}$. We distinguish between two cases depending on the state of OPT's server at the beginning of an interval.

The first case is that the sever of OPT is empty at the beginning of any interval. This means, each of $R_{i-1,i-1}, R_{i-1,i}$, $R_{i-1,i+1}$ and $R_{i-1,i+2}$ can be served (accepted and completed) by OPT during a single interval. That is to say, there is a tour of length at most $1/2$ that covers all of $R_{i-1,i-1}$. The same is true for $R_{i-1,i}$,$R_{i-1,i+1}$ and $R_{i-1,i+2}$. Let $R_{max} = \{R' : |R'| = max(|R_{i-1,i-1}|, |R_{i-1,i}|, |R_{i-1,i+1}|, |R_{i-1,i+2}|)\}$ be the set of requests which is the largest of $R_{i-1,i-1}, R_{i-1,i}$,$R_{i-1,i+1}$ and $R_{i-1,i+2}$. We denote by l_{max} the tour, which covers R_{max}. Now RE's server can take some time at most T to reach an optimal starting location on the route l_{max}, then it can take a remaining time of $1/2 - T$ to serve the most requests in R_{max} starting from the optimal location. Since it needs time $1/2$ to serve R_{max}, the server of RE will serve at least $(1-2T)|R_{max}|$ requests during the time of $1/2 - T$. Thus, the number of requests which RE's server can serve during interval i is at least $(1/2 - T)|R_{max}|$. Denoting the total number that RE can serve on the request sequence R by $B_{RE}(R)$, we have

$$\begin{aligned} B_{RE}(R) &\geq (1-2T) \sum_{i=2}^{m+1} max\big(|R_{i-1,i-1}|, |R_{i-1,i}|, |R_{i-1,i+1}|, |R_{i-1,i+2}|\big) \\ &\geq \Big(\frac{1-2T}{4}\Big) \sum_{i=1}^{m} \big(|R_{i,i}| + |R_{i,i+1}| + |R_{i-1,i+2}| + |R_{i-1,i+2}|\big) \\ &= \frac{1-2T}{4} B_{OPT}(R) \end{aligned}$$

The second case is that the OPT's server is currently carrying an object for a request r_e. Since T is the longest time for the server to travel between any two points, r_e can be completed within time T. This means that there is a tour of length $1/2 + T$, denoted by l_{max}, that can serve all the requests in R_{max}. So, RE's server can take time no more than T to pick its optimal starting location on the route l_{max}, and has a remaining time of $1/2 - T$ to serve requests from R_{max}. There will be at least $\frac{1-2T}{1+2T}|R_{max}|$ requests completed by RE's server

during interval i. So we have

$$\begin{aligned} B_{RE}(R) &\geq \Big(\frac{1-2T}{1+2T}\Big) \sum_{i=2}^{m+1} max\big(\,|R_{i-1,i-1}|\,,|R_{i-1,i}|\,,|R_{i-1,i+1}|\,,|R_{i-1,i+2}|\,\big) \\ &\geq \frac{1}{4}\Big(\frac{1-2T}{1+2T}\Big) \sum_{i=1}^{m} \big(\,|R_{i,i}| + |R_{i,i+1}| + |R_{i-1,i+2}| + |R_{i-1,i+2}|\,\big) \\ &= \frac{1}{4}\Big(\frac{1-2T}{1+2T}\Big) B_{OPT}(R) \end{aligned}$$

This completes the proof the fact that RE is $4(\frac{1+2T}{1-2T})$ -*competitive* for unit-capacity in any $T < 1/2$ metric space.

3.2 The SMARTCHOICE Algorithm (SM for short)

The time is divided into intervals of length Δ. At the beginning of every interval (at time $i\Delta, i = 1, 2, \cdots$), SM will consider all yet unserved requests (including those that are currently carried by the server). We denote by $R^A(i\Delta)$ the set of all unserved requests at time $i\Delta$. SM then finds a new optimal route from its current position that can cover the maximum number of requests from $R^A(i\Delta)$ by their deadlines. Let $R^C(i\Delta)$ represents the set of outstanding requests that have yet to be served on SM's current route at time $i\Delta$, and $R^N(i\Delta)$ will denote the set of requests which are served on the new route for time $i\Delta$. At any beginning of intervals, SM's server will switch to the new route if the number of requests gained is at least λ times the number of requests lost by the switch. That is, $\big|R^N(i\Delta) - R^C(i\Delta)\big| \geq \lambda \big|R^C(i\Delta) - R^N(i\Delta)\big|$ for some $\lambda > 1$. All new requests that arrive during the interval are temporarily ignored until the next beginning of interval.

Lemma 1 *Let $\Delta = 1/2 - T$ and R_{i-1} denotes the set of requests arriving during interval $[(i-1)\Delta, i\Delta)$. Provided that the server is unit-capacity ($z = 1$), during any interval of length Δ, the maximum number of requests that OPT's server can serve is at most $\lambda \big|R^C(i\Delta)\big|$.*

In the proof of lemma 1, we can also distinguish between two cases depending on the state of OPT's server at the beginning of an interval. For more details of the proof, please refer to the literature [8].

Since the time-window of request is 1 and every request is completed within time T of its accepted, any request in R_{i-1} will completed in the interval $[(i-1)\Delta, i\Delta + 1 + T)$. According to lemma 1, during any interval of length Δ, the maximum number of requests that OPT's server can serve is at most $\lambda \big|R^C(i\Delta)\big|$, the OPT's server can serve at most $\lambda \Big\lceil \frac{\Delta+1+T}{\Delta} \Big\rceil \big|R^C(i\Delta)\big|$ requests of R_{i-1} at any time. This means the total number of requests completed by OPT is at most

$$B_{OPT}(R) \leq \lambda \Big\lceil \frac{\Delta + 1 + T}{\Delta} \Big\rceil \sum_i \big|R^C(i\Delta)\big| \tag{1}$$

Now we will evaluate the number of requests that SM's server can complete. For each request that ever appears in SM's current route, consider a continuous interval of time in which it is in the current route. Each of intervals can be depicted a line on a horizontal time axis with two endpoints. $\big|R^C(i\Delta)\big|$ is the number of lines that contain point $i\Delta$. Since every request expires or is

completed within time $1+T$ of its arrival, each line can cover at most $\left\lceil \frac{1+T}{\Delta} \right\rceil$ interval endings. This means that

$$\sum_i \left| R^C(i\Delta) \right| \leq \left\lceil \frac{1+T}{\Delta} \right\rceil I \tag{2}$$

where I is the total number of lines.

The number of the requests served by SM can be expressed by the inequation

$$B_{SM}(R) \geq I(\lambda - 1)/\lambda \tag{3}$$

The proof of inequation (3) is similar to paper [8].

Theorem 2 *Provided that the server is unit-capacity ($z = 1$), for any metric space with $T < 1/2$, SM is $8\left\lceil \frac{3}{1-2T} \right\rceil \left\lceil \frac{1+T}{1-2T} \right\rceil$-competitive.*

Proof. According to inequation (2) and (3), we have

$$\sum_i \left| R^C(i\Delta) \right| \leq \left\lceil \frac{1+T}{\Delta} \right\rceil I \leq \frac{\lambda}{\lambda - 1} \left\lceil \frac{1+T}{\Delta} \right\rceil B_{SM}(R)$$

Thus, from (1) we get

$$B_{OPT}(R) \leq \frac{\lambda^2}{\lambda - 1} \left\lceil \frac{1+\Delta+T}{\Delta} \right\rceil \left\lceil \frac{1+T}{\Delta} \right\rceil B_{SM}(R)$$

Note that $\Delta = 1/2 - T$, we obtain

$$\frac{2\lambda^2}{\lambda - 1} \left\lceil \frac{3}{1-2T} \right\rceil \left\lceil \frac{1+T}{1-2T} \right\rceil B_{SM}(R) \geq B_{OPT}(R)$$

The optimal choice for $\lambda > 1$ is 2, then $\frac{2\lambda^2}{\lambda - 1} = 8$. This completes the proof.

When we consider the case that the server has infinite capacity ($z = \infty$), we will obtain the following corollaries.

Corollary 1 *Provided that the server has infinite capacity ($z = \infty$), for any metric space with $T < 1/2$, RE is $\frac{3}{1-2T}$ -competitive.*

Corollary 2 *Provided that the server has infinite capacity ($z = \infty$), for any metric space with $T < 1$, SM is $4\left\lceil \frac{2}{1-T} \right\rceil \left(\left\lceil \frac{2}{1-T} \right\rceil + 1 \right)$-competitive.*

Since the server has infinite capacity, it can accept several service requests in succession, then complete them in the copestone. So no matter OPT or online algorithms, RE and SM, will firstly pick up objects from the requests' sources as many as possible until the last request is disappear (expires or be served), then the serve carries accepted objects to their destinations. The proofs of the two corollaries above are similar to the proof of theorem 1 and theorem 2, respectively.

4 Lower Bounds

In this section we derive lower bounds on the competitive ratio of any deterministic online algorithm for serving the requests in the versions of the problem. The results are obtained by considering the optimal algorithm as an adversary that specifies the request sequence in a way that the online algorithm performs

badly.

Theorem 3 *Provided that the server has unit capacity* ($z = 1$)*, for any* $T < 1/2$*, there is a metric space with diameter* T *in which no deterministic online algorithm can obtain a competitive ratio less than* $\frac{2-T}{2T}$*.*

Proof. Let Q be a large enough integer, and k is an even number. Consider a metric space with Qk points that consists of Q groups which are denoted by g_q for $q = 1, 2, \cdots, Q$. Each group consists of two point sets, each with $k/2$ points. Let one set P_q^+ represent the set of pickup points and the other P_q^- be the set of delivery points in group g_q. Therefore, $\mid P_q^+ \mid = \mid P_q^- \mid = k/2$, $\mid P_q^+ \cup P_q^- \mid = k$ for any $q \in \{1, 2, \cdots, Q\}$. The distance between any two points is $1 - T$ units of length from the same group and $\frac{T}{2}$ units of length from different groups. Let $\tau = T$. At every time $t_i = i\tau (i = 0, 1, \ldots)$, the adversary will release the requests on the different points in the metric space depending on the state of the online algorithm's server at time t_i as follows:

1) If the online algorithm's server is working on one point in a certain group g_c or moving between two points which are belong to the same group g_c at time t_i, then a request will be released on each point in set $P_{q'}^+$ for every $q' \in \{1, 2, \cdots, Q\}/\{c\}$, requiring the server to carry the objects to the corresponding destinations in set $P_{q'}^-$.

2) If the online algorithm's server is moving between two points which are belong to two different groups, g_c and $g_{c'}$, then a request will be released on each point in set $P_{q'}^+$ for every $q' \in \{1, 2, \cdots, Q\}/\{c, c'\}$, requiring the server to carry the objects to the corresponding destinations in set $P_{q'}^-$. We notice that the points of the *source* and the *destination* of a certain request are in a same group.

The number of requests which the online algorithm's server can server that arrive at time $t_i (i = 0, 1, \ldots)$ is at most 1. The online server has to go to another group at some time after serving a request from the *current group* (where it is *currently* located) since no request will be released on any point in it. And the optimal time for the online server to leave for another group, denoted g_f, is at time $t_i + \varepsilon$, just after the adversary releases a request on each pickup point in every group (except for the *current group*), where $\varepsilon > 0$, an arbitrarily small number. When the online server arrives at one pickup point in group g_f at time $t_i + 1 - T + \varepsilon$, all the requests released *before* time t_i will be called off. There are only $k/2$ outstanding requests released at time t_i in group g_f. The online server will take $\frac{T}{2}$ units of time to server one request, ending at time $t_i + 1 - \frac{T}{2} + \varepsilon$. After that, if the online server tries to serve one more request at another point, all the outstanding requests will be called off when it arrives at the point since it has to spend at least $T/2$ units of time moving to the nearest pickup point from its current position. Thus, it will only be able to serve one of the requests that are released at time t_i. The total time that the online server needs to server one request is $1 - T + \frac{T}{2} = 1 - \frac{T}{2}$.

On the other hand, the adversary will arrange to make its server stay at a group and serve the outstanding requests if the online server does not arrive at that group or is on the way to that group. Otherwise, the adversary's server will leave for a new group to continue to work. Note that it is always in the adversary's best interest to select the group that will be the last one which will be visited by the online server after the adversary's server leaves. So we can think of the adversary's server as staying at one group all the while as long as

the number of groups gets large enough. The adversary's server will arrange to arrive at a pickup point in the group at time t_i when the new requests are released. And it will continue to go to another pickup point in the same group for the next request just after the server completes one request. As a result, the adversary can serve one request within T units of time. Note that the online server needs at least $1 - \frac{T}{2}$ units of time to serve a request. So the competitive ratio is $\frac{2-T}{2T}$. Thus, we can say that no deterministic online algorithm can achieve a competitive ratio less than $\frac{2-T}{2T}$. The proof of theorem 3 is completed.

Theorem 4 *Provided that the server has infinite capacity ($z = \infty$), for any $T < 1/2$, there is a metric space with diameter T in which no deterministic online algorithm can obtain a competitive ratio less than $\frac{2-T}{2T}\lceil\frac{1}{T}\rceil$.*

Proof. The proof is similar to the proof of theorem 3. The server has infinite capacity, so when it arrives at a pickup point every time, it can pick up all the outstanding requests on that point. As long as the number of pickup points in a group is at least $\lceil\frac{2}{T}\rceil$, the adversary can serve $\lceil\frac{1}{T}\rceil$ requests within T units of time. Still, the online server will only be able to serve one request within $1 - \frac{T}{2}$ units of time with the same reasoning. So the competitive ratio is $\frac{2-T}{2T}\lceil\frac{1}{T}\rceil$. Therefore, the theorem holds.

5 Conclusions

Online dial-a-ride problems are occurring in a wide variety of practical settings and cover not only physical rides but transportation means[1]. We consider a class of variations of this kind of problems, in which there is a uniform time-window on each of the requests. Two cases are considered separately, where $z = 1$ and where $z = \infty$. It will be interesting to extend the results to the case of non-uniform windows. Another interesting direction is to consider the case when the server has limited capacity $1 < z \leq C$, where C is a constant.

Acknowledgements

The work was partly supported by the NSF of Chian (NO.70525004,70471035).

References

[1] Manasse M.S., McGeoch L.A., and Sleator D.D., Competitive algorithms for server problems, *Journal of Algorithms*, 11(1990), 208–230.

[2] David S.B., Borodin A., A new measure for the study of the on-line algorithm,*Algorithmica*, 11(1994), 73–91.

[3] Alon N., Karp R.M., Peleg D., et al., A graph-theoretic game and its application to the k- server problem,*SIAM. J. Comput.* 24(1995),78–100.

[4] Feuerstein E. and Stougie L., On-line single server dial-a-ride problems, *Theoretical Computer Science*,268(2001), 91–105.

[5] Ascheuer N., Krumke S.O., and Rambau J., Online dial-a-ride problems: Minimizing the completion time, *Lecture Notes in Computer Science*, (2000), 639–650.

[6] Xu Y.F., Wang K.L., Scheduling for on-line taxi problem and competitive algorithms,*Journal of Xi'an Jiao Tong University*,31 (1997), 56–61.

[7] Ausiello G., Feuerstein E., Leonardi S., Stougie L., and Talamo M., Algorithms for the on-line traveling salesman,*Algorithmica*,29(2001), 560–581.

[8] Irani S., Lu X., and Regan A., On-line algorithms for the dynamic traveling repair problem,*In Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, (2002), 517–524.

[9] Krumke S.O., Megow N., and Vredeveld T., How to whack moles, *Lecture Notes in Computer Science*,2909(2004), 192–205.

[10] Ma W.M., XU Y.F., and Wang K.L., On-line k-truck problem and its competitive algorithm,*Journal of Global Optimization*,21(2001), 15–25.

[11] Hauptmeier D. Krumke S.O., and Rambau J., The online dial-a-ride problem under reasonable load,*Lecture Notes in Computer Science*, (2000),125–136.

[12] Krumke S.O., de Paepe W.E., Poensgen D., and Stougie L., News from the online traveling repairman, *Theoretical Computer Science*,295(2003), 279–294.

[13] Krumke S.O., Laura L., Lipmann M., Marchetti-Spaccamela et al., Nonabusiveness helps: An O(1)-competitive algorithm for minimizing the maximum flow time in the online traveling salesman problem, *Lecture Notes in Computer Science*, (2002), 200–214.

[14] Lipmann M.,Lu X.,de Paepe W.E. and Sitters R.A., On-Line Dial-a-Ride Problems under a Restricted Information Model,*Lecture Notes in Computer Science*,2461(2002), 674–685.

[15] Xu, Y.F., Wang K.L. and Zhu B., On the k-taxi problem,*Journal of Information*,2(1999), 429–434.

[16] Xu, Y.F., Wang K.L. and Ding J.H., On-line k-taxi scheduling on a constrained graph and its competitive algorithm, *Journal of System Engineering(P.R. China)*,4(1999)