Journal of Global Optimization **21:** 15–25, 2001.
© 2001 *Kluwer Academic Publishers. Printed in the Netherlands.*

On-line k-Truck Problem and Its Competitive Algorithms ⋆

WEIMIN MA[1,2], YINFENG XU[1] and KANLIANG WANG[1]
[1]*The School of Management Xi'an Jiaotong University, Xi'an, Shaanxi. P.R. China. 710049;* [2]*The Dept. of the Computing The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong (e-mail: cswmma@comp.polyu.edu.hk)*

(Received 2 November 2000; accepted in revised form 10 April 2001)

Abstract. In this paper, based on the *Position Maintaining Strategy* (PMS for short), on-line scheduling of k-truck problem, which is a generalization of the famous k-server problem, is originally presented by our team. We proposed several competitive algorithms applicable under different conditions for solving the on-line k-truck problem. First, a competitive algorithm with competitive ratio $2k + 1/\theta$ is given for any $\theta \geq 1$. Following that, if $\theta \geq (c + 1)/(c - 1)$ holds, then there must exist a $(2k - 1)$-competitive algorithm for k-truck problem, where c is the competitive ratio of the on-line algorithm about the relevant k-server problem. And then a greedy algorithm with competitive ratio $1 + \lambda/\theta$, where lambda is a parameter related to the structure property of a given graph, is given. Finally, competitive algorithms with ratios $1 + 1/\theta$ are given for two special families of graphs.

Key words: PMS, On-line k-truck problem, Competitive algorithm, Competitive ratio

1. Introduction

The k-truck problem can be stated as follows. We are given a metric space M, and k trucks which move among the points of M, each occupying one point of M. Repeatedly, a request (a pair of points $x, y \in M$) appears. To serve a request, an empty truck must first move to x and then move to y with goods from x. How do we minimize the total cost of all trucks? Let's first consider following problems:

(1) Given a service request sequence, how can we schedule trucks so as to minimize the cost?
(2) If the service request is received one by one in the process of service without any knowledge of the future requests, how to minimize the cost as possible as we can?

The k-truck problem aims at minimizing the cost of all trucks. Because the cost of the trucks with goods is different from that without goods on same distance, the total distance cannot be considered as the objective to be optimized. For simplicity, we assume that the cost of the truck with goods is θ times of that without goods on the same distance. Then we can take the $(1 + \theta)$ times of the empty loaded

⋆ The authors would like to acknowledge the support of research grant from National Natural Science Foundation of China. Grant No.19731001.

distance as the objective. Problem (1) is an off-line problem, and (2) is an on-line problem. The difference between them depends on whether the known service request sequence is total or part/problem. The problem (1) can be solved with the dynamic programming, but the problem (2) is difficult to resolve. We must serve the request only based on the information of the previous requests: the decision must be made on-line as we have no information about the future requests.

The k-truck problem is a generalization of the k-taxi problem [6] and well-known k-server problem [1]. In the k-server problem, each request contains only a point x in M. We must move one server to (server) x. For the k-server problem, several on-line algorithms have been previously proposed [1–6]. The k-taxi problem, as a special example of the k-truck problem, is presented in [6]. The authors gave a good strategy, namely, the *Position Maintaining Strategy* (PMS), to deal with the on-line k-taxi problem. For k-taxi problem, some competitive algorithms that have good competitive ratio were given. The strategy is also well used in dealing with the on-line k-elevator problem [7].

Let $\mathcal{M}$ be a class of metric spaces. We call an on-line strategy c-competitive for $\mathcal{M}$, if for every metric space in $\mathcal{M}$ and for every request sequence, the total cost incurred by that on-line strategy is at most c times the optimal off-line cost of serving the same request sequence. A strategy is competitive for $\mathcal{M}$ if it is c-competitive for $\mathcal{M}$, for some c. Note that if a strategy is competitive then the respective definitions hold for all metric space. For more positive and negative results for on-line algorithms one can refer to [3].

In this paper, we first establish the mathematical model of the k-truck problem in Section 2. And then the close relationship among k-truck problem, k-server problem, and k-taxi problem is given. With the PMS, several competitive algorithms that have good ratios are shown in the Section 3. Finally some concluding remarks are given in Section 4.

2. The Model of the k-Truck Problem

Let $G = (V, E)$ denote an edge weighted graph with n vertices and the weights of edges satisfy the triangle inequality, where V is a metric space consisting of n vertices, and E is the set of all weighted edges. We assume that the weight of edge (x, y) is denoted by $d(x, y)$ and the weights are symmetric, i.e., for all $x, y, d(x, y) = d(y, x)$. We assume that k trucks occupy a k-vertexes which is a subset of V. A service request $r = (a, b)$, $a, b \in V$, implies there are some goods on vertex a that must be moved to vertex b (for simplicity, we assume that the weight of goods is same all the time). A service request sequence R consists of some service request in turn, namely $R = (r_1, r_2, \dots, r_m)$ where $r_i = (a_i, b_i)$, $a_i, b_i \in V$. On-line k-truck scheduling problem is to decide to move which truck when a new service request occurs on the basis that we have no information about future possible requests.

All discussion is based on the following essential assumptions:

i) Graph G is connected;
ii) When a new service request occurs, k trucks are all free;
iii) All trucks have same load weight and the cost of the truck with goods is θ times that of without goods on same distance, and $\theta \geq 1$.

For a known sequence $R = (r_1, r_2, \dots , r_m)$, let $C_{opt}(R)$ be the optimal total cost after finishing it. For every new service request, if scheduling algorithm A can schedule without information of the sequence next to r_i, we call A an on-line algorithm. For on-line algorithm A, if there are constants α and β satisfying:

$$C_A(R) \leqslant \alpha C_{opt}(R) + \beta.$$

Then for any possible R, A is called a competitive algorithm, where $C_A(R)$ is the total cost with algorithm A to satisfy the sequence R.

For a request $r_i = (a_i, b_i)$, the procedure scheduling of a truck to a_i is called empty load. And that from a_i to b_i is heavy load. If there is no limit for the R and θ, the on-line truck problem is called P. In problem P, if for any $r_i = (a_i, b_i)$, $d(a_i, b_i) > 0$ and $\theta > 1$ holds, the problem is called P_1. Problem P_1 is also called the normal k-truck problem. In problem P, if there is no limit for any $r_i = (a_i, b_i)$ but $\theta = 1$, the problem is P_2. Problem P_2 is also called k-taxi problem. In P_2, if $d(a_i, b_i)$>0, namely $a_i = b_i$, the problem is called P_3. Problem P_3 is also called normal k-taxi problem. In problem P, if $d(a_i, b_i) = 0$, namely $a_i = b_i$, it is called P_4. Problem P_4 is also called k-server problem.

3. Several Results of the Competitive Ratios

In this section, several competitive algorithms and their competitive ratios are presented. And the interrelationship among the k-truck, k-taxi and k-server problems is given.

3.1. PROBLEM P_4: k-SERVER PROBLEM

The famous on-line k-server problem is presented as a special case of the on-line truck problem in 1990 [1]. There are many results on the competitive ratio for the k-server problem. Koutsoupias and Papadimitriou showed that there exists an on-line algorithm for the k-server problem with competitive ratio $2k - 1$[3]. Because k-server problem is a special case of the k-truck problem, the researches concerning the on-line truck problem must discover the relationship between them. Here we first give following lemma [3].

LEMMA 3.1 *There exists an on-line algorithm for the k-server problem with the competitive ratio $2k - 1$.*

3.2. POSITION MAINTAINING STRATEGY

In the paper [6], the PMS was proposed and was used to get several good research results for the k-taxi problem. For our investigation of k-truck problem, we outline the PMS with k-truck opinion as follows.

Position Maintaining Strategy is defined as follows. For the present request $r_i = (a_i, b_i)$, after a_i is reached, the truck reaching a_i must move from a_i to b_i with the goods to complete r_i. When the service for r_i is finished, the PMS moves the truck at b_i back to a_I(empty) before the next request arrives.

LEMMA 3.2 *Let Opt be an optimal algorithm for an request sequence $R = (r_1, r_2, ..., r_m)$, then we have*

$$C_{opt}(R) \geq C_{opt}(\sigma) + \sum_{i=1}^{m} (\theta - 1) \cdot d(a_i, b_i)$$

where $\sigma = ((a_1, a_1), (a_2, a_2), ..., (a_m, a_m))$.

Proof. For any request sequence R, the scheduling procedure of the serving it must finish the sequence σ, and at same time $\sum_{i=1}^{m} \theta d(a_i, b_i)$ is the necessary cost because it is the total cost of all loaded trucks. Furthermore, the overlap of the cost to finish the sequence σ and the cost of and $\sum_{i=1}^{m} \theta d(a_i, b_i)$ is at most $\sum_{i=1}^{m} d(a_i, b_i)$.

Thus any algorithm which completes the sequence R must at least take the cost at the sum of $C_{opt}(\sigma)$ and $\sum_{i=1}^{m} (\theta - 1) d(a_i, b_i)$. The inequality holds.

3.3. THE RESULTS FOR k-TRUCK PROBLEM P

With the above Lemma and PMS, the close relationship between k-server and k-truck problem can be shown below.

THEOREM 3.1 *For a given graph G if there is a c-competitive algorithm for the k-server problem on G, then there is a $(c + 1 + 1/\theta)$-competitive algorithm A for the k-truck problem on G, where θ has the same meaning as above and $\theta \geq 1$.*

Proof. For any $R = (r_1, r_2, \ldots, r_m)$, considering k-server problem's request $\sigma = (a_1, a_2, \ldots, a_m)$, let A_σ be a c-competitive algorithm for on-line k-server problem on graph G. We design the relevant algorithm A for on-line k-truck problem as follows.

For current service request $r_i = (a_i, b_i)$ we first schedule a truck to a_i using algorithm A_σ, then complete the r_i with PMS. Thus the algorithm A's total cost is,

$$\begin{aligned} C_A(R) &= \sum_{i=1}^{m} C_A(\gamma_i) \\ &= \sum_{i=1}^{m} (C_{A_\sigma}(a_i) + (\theta + 1) \cdot d(a_i, b_i)) \qquad (3.1) \\ &= C_{A_\sigma}(\sigma) + \left(1 + \frac{1}{\theta}\right) \cdot \sum_{i=1}^{m} \theta \cdot d(a_i, b_i), \end{aligned}$$

where θ is defined above and $\theta \geq 1$. From Lemma 2 and algorithm A_σ, we have,

$$\begin{aligned} C_{A_\sigma}(\sigma) &\leq c \cdot C_{opt}(\sigma) + \beta \\ &\leq c \cdot \left(C_{opt}(R) - \sum_{i=1}^{m} ((\theta - 1) \cdot d(a_i, b_i))\right) + \beta \qquad (3.2) \\ &\leq c \cdot C_{opt}(R) + \beta \end{aligned}$$

and

$$\sum_{i=1}^{m} (\theta \cdot d(a_i, b_i)) \leq C_{opt}(R). \qquad (3.3)$$

Combining the (3.1), (3.2) and (3.3), we get,

$$C_A(R) \leq \left(c + 1 + \frac{1}{\theta}\right) \cdot C_{opt}(R) + \beta,$$

where c and β are some constants.

For the k-taxi problem, as a special case of the k-truck problem, we can let $\theta = 1$ and then get an $(c + 2)$-competitive algorithm, which was given in [6].

From Theorem 3.1 and Lemma 3.1, the following Corollary holds.

COROLLARY 3.1 *For a given graph G, there exists a $(2k + 1/\theta)$-competitive algorithm for k-truck problem on G, where θ is a constant and $\theta \geq 1$.*

Based on the above discussion, we have no restrictions except for $\theta \geq 1$, in fact if we further restrict θ, we can improve the result as follows.

THEOREM 3.2 *For a given graph G, if there is a c-competitive ($c \geq 1$) on-line algorithm for the k-server problem on G and $\theta \geq (c + 1)/(c - 1)$, then there is a c-competitive algorithm A for the k-truck problem on G, where θ is defined above.*

Proof. Similar to theorem 3.1, we can design an algorithm A' such that,

$$C_{A'}(R) = C_{A_\sigma}(\sigma) + \left(1 + \frac{1}{\theta}\right) \cdot \sum_{i=1}^{m} \theta \cdot d(a_i, b_i) \tag{3.4}$$

and

$$C_{A_\sigma}(\sigma) \leq c \cdot \left(C_{opt}(R) - \sum_{i=1}^{m} (\theta - 1) \cdot d(a_i, b_i)\right) + \beta \tag{3.5}$$

where θ is defined as above and $\theta \geq (c+1)/(c-1)$.

Combining the (4) and (5), we get,

$$\begin{aligned} C_{A'}(R) &\leq c \cdot C_{opt}(R) + \left[1 + \frac{1}{\theta} - c \cdot \left(\frac{\theta - 1}{\theta}\right)\right] \sum_{i=1}^{m} \theta \cdot d(a_i, b_i) + \beta \\ &\leq c \cdot C_{opt}(R) + \beta. \end{aligned}$$

The second inequality holds for that $\theta \geq (c+1)/(c-1)$ and then $1 + 1/\theta - c \cdot (1 - 1/\theta) \leq 0$, where c and β are constants.

From Theorem 3.2 and Lemma 3.1, the following Corollary holds.

COROLLARY 3.2 *For a given graph G, if $\theta \geq (c+1)/(c-1)$ holds, then there exists a $(2k-1)$-competitive algorithm for k-truck problem on G.*

Although the k-taxi problem is the special example of the k-truck problem, because the translation needs $\theta = 1$ and the above corollary holds if $\theta \geq (c+1)/(c-1) > 1$, the result of above the $(2k-1)$ ratio does not holds for the k-taxi problem.

3.4. ON THE CONSTRAINT GRAPHS

An extreme cases of the k-truck problem occurred when the number of trucks is either equal to the number of the vertices of G or the number of G minus one. For the k-server problem, when $k = n$ the constant cost is enough to handle any request sequence. For the case of $k = n - 1$, there exists a $(n-1)$-competitive algorithm [1]. For k-truck problem, we can easily use PMS to handle the case of $k = n$ with competitive ratio $1 + 1/\theta$. And for the case of $k = n - 1$, if each request $r_i = (a_i, b_i)$ satisfies $d(a_i, b_i) > 0$, then using PMS we can also obtain a $(1 + 1/\theta)$-competitive algorithm. We give following lemma that can be used to get above results.

LEMMA 3.3 *For any algorithm A for an request sequence $R = (r_1, r_2, \dots, r_m)$, we have*

$$C_A(R) \geq \sum_{i=1}^{m} \theta \cdot d(a_i, b_i)$$

and,

$$C_{opt}(R) \geq \sum_{i=1}^{m} \theta \cdot d(a_i, b_i).$$

Proof. For any request sequence R, $\sum_{i=1}^{m} \theta \cdot d(a_i, b_i)$ is the necessary cost because it is the cost of the heavy load. Namely, any algorithm that completes the sequence R must at least take the cost $\sum_{i=1}^{m} \theta \cdot d(a_i, b_i)$. The inequalities hold.

Using Lemma 3.3 and PMS, we can obtain the following theorem concerning the two cases.

THEOREM 3.3 *For a given graph G with n vertices, if $k = n$, then there is an on-line algorithm for the k-truck problem on G with a competitive ratio $1 + 1/\theta$ and if each request $r_i = (a_i, b_i)$ satisfies $d(a_i, b_i) > 0$, and if $k = n - 1$, then with PMS one can obtain a $(1 + 1/\theta)$-competitive algorithm.*

Proof. We assume that on the each vertex there is at most one truck. Otherwise, we can precondition the truck locations such that each vertex has at most one truck. Furthermore, the cost of this precondition is at most a constant $(n - 1)d$, where $d = max_{x,y \in v} d(x, y)$.

For the case $k = n$, we design the relevant on-line k-truck problem algorithm A_1 as follows. For any $R = (r_1, r_2, \dots, r_m)$, considering the current service request $r_i = (a_i, b_i)$,

If $a_i = b_i$, all trucks need not any scheduling and the cost is 0.

If $a_i \neq b_i$, because there are trucks on both a_i and b_i, we can schedule the truck on a_i to carry goods to b_i and the truck on b_i to move to a_i without empty load. Then the cost to finish r_i is $(1 + \theta) \cdot d(a_i, b_i)$, at same time there still is one and only one truck on each vertex.

For the on-line algorithm A$_1$, we have,

$$
\begin{aligned}
C_{A_1}(R) &= \sum_{i=1}^{m} C_{A_1}(r_i) + \beta \\
&\leq (\theta + 1) \sum_{i=1}^{m} d(a_i, b_i) + \beta \\
&= \left(1 + \frac{1}{\theta}\right) \sum_{i=1}^{m} \theta \cdot d(a_i, b_i) + \beta \\
&\leq \left(1 + \frac{1}{\theta}\right) C_{opt}(R) + \beta
\end{aligned}
$$

where β is the cost of precondition and $\beta \leq (n-1) \cdot d$, where $d = max_{x,y \in v} d(x, y)$.

For the case of $k = n - 1$, we design the relevant on-line k-truck problem algorithm A$_2$ as follows. For any $R = (r_1, r_2, \dots, r_m)$, considering the current service request $r_i = (a_i, b_i)$ satisfies $d(a_i, b_i) > 0$,

(1) If there are trucks on both a_i and b_i, we can schedule the truck on a_i to carry goods to b_i and the truck on b_i to move to a_i without empty load. Then the cost to finish r_i is $(1 + \theta) \cdot d(a_i, b_i)$, at same time there still is one and only one truck on each vertex.

(2) If there is a truck on a_i but not on b_i, we can schedule the truck on a_i to b_i with heavy load. The cost is $\theta \cdot d(a_i, b_i)$. Moreover, there is not any vertex on which there are more than one trucks.

(3) If there is a truck on b_i but not on a_i, we can schedule the truck on b_i to a_i with empty load and then go back b_i with heavy load from a_i. The cost is $(1+\theta) \cdot d(a_i, b_i)$. Moreover, there is no vertex on which there are more than one truck.

Similar to the first case, we can prove that the on-line algorithm A_2 is a $(1 + 1/\theta)$-competitive algorithm.

3.5. SCHEDULING k-TRUCK ON A SPECIAL GRAPH

In this section we consider scheduling of k trucks on a special graph. Let $d_{\max} = \max d(v_i, v_j)$, $d_{\min} = \min d(v_i, v_j)$, $i \neq j$, $v_i, v_j \in V$, and let

$$
\lambda = \frac{d_{\max}}{d_{\min}}
$$

We study the k-truck problem with following constrains,

(a) there is a constant C such that $\lambda \leq C$,

(b) for each request $r_i = (a_i, b_i)$ satisfies $d(a_i, b_i) > 0$, and

(c) with respect to the present truck locations, there is at most one truck located at a vertex.

With all of the above constraints, we present the following on-line algorithm B to solve the k-truck problem. Let $r_i = (a_i, b_i)$ be the present request. Considering the scheduling of the following cases, we can give algorithm B as follows,

(1) If there is a truck at a_i and also there is a truck at b_i, then B moves the truck at a a_i to b_i complete the request and at the same time B moves the truck at b_i to a_i with empty load. The cost of B for the r_i is $(1+\theta)\cdot d(a_i, b_i)$ and at present no vertex has more than one truck.

(2) If there is a truck at a_i and no truck at b_i, then B moves the truck at a_i to b_i complete the request. The cost of B for the r_i is $\theta \cdot d(a_i, b_i)$ and at present no vertex has more than one truck.

(3) If there is no truck at a_i and there is a truck at b_i, then B moves the truck at b_i to a_i first without load, and after that moves from a_i to b_i to complete the request complete the request. The cost of B for the r_i is $(1+\theta)\cdot d(a_i, b_i)$ and at present no vertex has more than one truck.

(4) If there is no truck at a_i and b_i, then B moves the truck which is the closest to a_i (suppose that the truck is locate at c_i) to a_i with empty load and then moves to b_i to complete the request. The cost of B for the r_i is $d(c_i, a_i) + \theta \cdot d(a_i, b_i)$ and again no vertex has more than one truck.

According to the above algorithm B, we can get the result as follows.

THEOREM 3.4 *Under the assumption (a), (b) and (c) specified at the beginning of this section, scheduling algorithm B for the k-truck problem achieves competitive ratio* $1 + \lambda/\theta$.

Proof. We have four possible cases for dealing with a request. For cases (1), (2) and (3), the cost of B is at most $(1+\theta)$ times the optimal cost for any request. For case (4), the extra cost is $d(c_i, a_i)$. Since c_i is the closest occupied vertex to a_i, we have $d(c_i, a_i) \le d_{max}$. Let $C_B(R)$ denote the cost of B for R, then we have

$$C_B(R) \le \sum_{i=1}^{m} \{\max[d(b_i, a_i), d(c_i, a_i)] + \theta \cdot d(a_i, b_i)\} + \beta$$

where β is the cost for preconditioning the truck such that each vertex has at most one truck and it is bounded by a constant related with G.

Since $a_i \ne b_i$ and $d(a_i, b_i) > 0$, we have

$$\frac{C_B(R)}{\sum_{i=1}^{m} d(a_i, b_i)} \le \theta + \frac{\sum_{i=1}^{m} \max[d(b_i, a_i), d(c_i, a_i)]}{\sum_{i=1}^{m} d(a_i, b_i)} + \frac{\beta}{\sum_{i=1}^{m} d(a_i, b_i)}$$

Since $d(c_i, a_i) \le d_{\max}$, and $d_{\min} \le d(a_i, b_i) \le d_{\max}$, we have

$$\frac{C_B(R)}{\sum_{i=1}^m d(a_i, b_i)} \le \theta + \frac{\sum_{i=1}^m d_{\max}}{\sum_{i=1}^m d_{\min}} + \frac{\beta}{\sum_{i=1}^m d(a_i, b_i)}$$
$$\le \theta + \lambda + \frac{\beta}{\sum_{i=1}^m d(a_i, b_i)}$$

which implies that,

$$C_B(R) \le \left(1 + \frac{\lambda}{\theta}\right) \cdot \theta \cdot \sum_{i=1}^m d(a_i, b_i) + \beta$$
$$\le \left(1 + \frac{\lambda}{\theta}\right) \cdot C_{opt}(R) + \beta$$

3.6. COMPARISON OF TWO ALGORITHMS FOR PROBLEM P_1

In sections 3.3 and 3.5, we gave two algorithms A' and B respectively, and both of which are competitive algorithms for P_1. We may confronted with the problem of choosing one algorithm from A' and B in different context.

The criterion, with which one can judge which on-line algorithm is better than the other, is the competitive ratio concerning relevant on-line algorithm. Respectively the competitive ratios of algorithms A' and B are

$c_{A'} = 2k - 1$, and

$c_B = 1 + \lambda/\theta.$

Let $c_{A'} = c_B$, we can get a k that makes the algorithm A and B equal,

$2k - 1 = 1 + \lambda/\theta.$

We get

$k = 1 + \lambda/(2\theta).$

Obviously, the following theorem holds.

THEOREM 3.5 *For problem P_1, if on-line algorithms A' and B at the aspect of the competitive ratio, if $\theta >= (c+1)/(c-1)$ holds, comparing with $k <= 1+\lambda/(2\theta)$ then A' is better than B, and contrarily if $k >= 1 + \lambda/(2\theta)$ then B is better than A'.*

4. Concluding Remarks

For the k-taxi problem, all results are suitable as long as $\theta = 1$, because the k-taxi problem is only a special example of k-truck problem. For problem P_1 and P_4

in this paper, there are many theoretical problems that need to be studied further. Another interesting problem related to the k-truck problem is that we can consider some other optimal criteria, such as minimizing the maximum waiting time or minimizing the sum of all empty move distances.

References

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

Ben David, S. and Borodin, A. (1994), A new measure for the study of the on-line algorithm, *Algorithmica* 11: 73–91.

Koutsoupias, E. and Papadimitriou, C. (1994) On the k-server conjecture, *STOC.* 507–511.

Alon, N., Karp, R.M. and Peleg, D. et al. (1995), A graph-theoretic game and its application to the k-server problem, *SIAM J.Comput.* 24(1): 78–100.

Du, D. Z. (1991), k-server problem and competitive algorithm, *Practice and Acquaintanceship of Mathematics*, 4: 36–40.

Xu, Y.F. and Wang, K.L. (1997), On-line k-taxi problem and competitive algorithm, *Journal of Xi'an Jiaotong University*, 1: 56–61.

Xu, Y.F., Wang, K.L. and Zhu, B. (1999) On the k-taxi problem, *Journal of Information*, Vol. 2, No. 4. 429–434.

Ma, W.M., Xu, Y.F. and Wang, K.L. (1999), On-line k-truck scheduling problem and its competitive strategies, *Journal of Northwest University (Natural Science, P.R. China)*, Vol. 29, No. 4: 254–258.

Xu, Y.F., Wang, K.L. and Ding, J.H. (1999), On-line k-taxi scheduling on a constrained graph and its competitive algorithm, *Journal of System Engineering* (P.R. China), No. 4..