International Journal of Foundations of Computer Science
Vol. 15 No. 1 (2004) 127–141

World Scientific
www.worldscientific.com

ON THE k-TRUCK SCHEDULING PROBLEM *

WEIMIN MA†
School of Management, Xian Jiaotong University
Xian, Shaanxi, P. R. China

YINFENG XU
School of Management, Xian Jiaotong University
Xian, Shaanxi, P. R. China

JANE YOU
Department of Computing, Hong Kong Polytechnic University
Kowloon, Hong Kong, P. R. China

JAMES LIU
Department of Computing, Hong Kong Polytechnic University
Kowloon, Hong Kong, P. R. China

and

KANLIANG WANG
School of Management, Xian Jiaotong University
Xian, Shaanxi, P. R. China

Received 15 October 2002
Accepted 5 May 2003
Communicated by Oscar H. Ibarra

ABSTRACT

*The authors would like to acknowledge the support of Central Research Grant GV-975 of the Hong Kong Polytechnic University and Research Grant from NSF of China. No.19731001.
†School of Management, Xian Jiaotong University, Xian, Shaanxi, P. R. China.

In this paper, some results concerning the k-truck problem are produced. Firstly, the algorithms and their complexity concerning the off-line k-truck problem are discussed. Following that, a lower bound of competitive ratio $(1+\theta)\cdot k/(\theta\cdot k+2)$ for the on-line k-truck problem is given, where θ is the ratio of cost of the loaded truck and the empty truck on the same distance, and a relevant lower bound for the on-line k-taxi problem followed naturally. Thirdly, based on the *Position Maintaining Strategy* (PMS), some new results which are slightly better than those of [11] for general cases are obtained. For example, a c-competitive or $(c/\theta+1/\theta+1)$-competitive algorithm for the on-line k-truck problem depending on the value of θ, where c is the competitive ratio of some algorithm to a relevant k-server problem, is developed. The *Partial-Greedy Algorithm* (PG) is used as well to solve this problem on a line with n nodes and is proved to be a $(1+(n-k)/\theta)$-competitive algorithm for this case. Finally, the concepts of the on-line k-truck problem are extended to obtain a new variant: *Deeper On-line k-Truck Problem* (DTP). We claim that results of PMS for the STP (Standard Truck Problem) hold for the DTP.

Keywords: k-Truck problem; On-line algorithm; Competitive ratio; Position Maintaining Strategy; Partial-Greedy Algorithm.

1. Introduction

Foundations. It is natural and necessary to assume that problem data are given for handling many traditional optimization problems in different disciplines, including *Operations Research, Mathematics, Computer Science, Management, Economics* and so on. In many aspects of real life, however, many ongoing decision-making activities, such as currency exchange, stock transaction or mortgage financing, must be carried out in an on-line fashion, with no definite knowledge of future events, and decisions once made cannot be changed. Furthermore, that knowledge often influences the decision result in a fatal way. Facing with this lack of knowledge, players of these decision-making games often have two choices. One is to use models based on assumptions about the future distribution of relevant quantities such as exchange rates or mortgage rates, and aim for acceptable results on the average. The other is to analyze the worst case scenario in the games and then make some decision. Unfortunately, these two approaches may give some on-line solutions that are far from the relevant optimal solutions.

An alternate approach in such situations is to use *competitive* analysis (first applied to on-line algorithms by Sleator and Tarjian in [14]). In this approach, the performance of an on-line strategy is measured against that of an optimal off-line strategy having full knowledge of future events. For some measure of cost or of profit, we try to minimize the worst-case ratio of on-line cost to optimal cost or of optimal profit to on-line profit; if this ratio is bounded for all event sequences, we deem the on-line strategy to be competitive and call the supremum of this ratio for profit problem and the infimum of this ratio for cost problem the competitive ratio of this on-line strategy. An advantage of this performance measure over the traditional average-case measure is that for most nontrivial decision-making activities it is extremely difficult to come up with an accurate probabilistic model.

An on-line algorithm receives the input incrementally, one piece at a time. In response to each input portion, the algorithm must generate output, not knowing the future input. In a competitive analysis an on-line algorithm A is compared to an optimal off-line algorithm OPT. An optimal off-line algorithm knows the entire

input sequence in advance and can process it optimally. Given an input sequence I, let $C_A(I)$ and $C_{OPT}(I)$ denote the costs incurred by A and OPT in processing I, respectively. Algorithm A is called α-competitive if there exists a constant α and β, such that

$$C_{\mathrm{A}}(I) \leq \alpha \cdot C_{\mathrm{OPT}}(I) + \beta \tag{1}$$

for all input sequences I. An analogous definition can be given for on-line maximization problems. We note that a competitive algorithm must perform well on all input sequences.

Related Work. Over the past two decades, *on-line problems* and their *competitive analysis* have received considerable interest. S. Albers and S. Leonardi [1] carried out a comprehensive survey of this area. On-line problems had been investigated already in the seventies and early eighties, but an extensive, systematic study started only when Sleator and Tarjian [14] suggested comparing an on-line algorithm to an optimal off-line algorithm and Karlin, Manasse, Rudolph and Sleator [8] coined the term competitive analysis. In the late eighties and early nineties, three basic on-line problems were studied extensively, namely paging, the k-server problem and metrical task systems. The k-server problem, introduced by Manasse et al. [12], generalizes paging as well as more general caching problems. The problem consists of scheduling the motion of the k mobile servers that reside on the points of a metric space S. The metrical task system, introduced by Borodin et al. [4], can model a wide class of on-line problems. An on-line algorithm deals with events that require immediate response. Future events are unknown when the current event is dealt with. The task system [4], the k-server problem [13], and on-line/off-line games [3] all attempt to model on-line problems and algorithms. During the past several years, apart from the three basic problems, many on-line problems have been investigated, covering areas such as data structures, distributed data management, scheduling and load balancing, routing, robotics, financial games, graph theory, and a number of problems arising in computer systems. The k-taxi problem, as a special example of the k-truck problem, is presented in [17]. The authors gave a good strategy, namely *Position Maintaining Strategy* (PMS), to deal with the on-line k-taxi problem. Some competitive algorithms that have good competitive ratios were given. The strategy is also well used in dealing with the on-line k-elevator problem [19]. In the paper [11], for k-truck problem, the authors established the relevant model and gave some results concerning the competitive ratio of some on-line algorithms. The best result in the paper is that there exists a c-competitive algorithm for the on-line k-truck problems, where c is the competitive ratio of some algorithm to a relevant k-server problem.

In [7] a survey concerning the adversary method is made. The adversary method for deriving lower bounds on the competitive ratio has been implicitly used by Woodall [16] in the analysis of the so-called Bay Restaurant Problem. Kierstead and Trotter [9] applied the adversary method in their investigation of on-line interval graph coloring. Yao [18] formulated a theorem to prove the impossibility of an on-line bin-packing algorithm with a competitive ratio strictly better than 3/2. This

seems to be the first result stated on the class of all on-line algorithms for a certain optimization problem, thus exploiting the distinction between on-line and off-line algorithms.

Our Contribution. In this paper, some new results concerning the k-truck problem are shown. We first discussed the algorithms and its complexity concerning the off-line k-truck problem, which is very important to study the relevant on-line k-truck problem. Following that, by constructing a special request sequence, a lower bound of competitive ratio $(1+\theta)\cdot k/(\theta\cdot k+2)$ for the on-line k-truck problem is given, and a relevant lower bound for the on-line k-taxi problem is obtained naturally. Especially, based on the *Position Maintaining Strategy* (PMS), we get some new results which are slightly better than those of paper [11] for the general cases, namely, there is a c-competitive algorithm if $\theta > (c+1)/(c-1)$ or a $(c/\theta+1/\theta+1)$-competitive algorithm if $1 \leq \theta \leq (c+1)/(c-1)$ for the on-line k-truck problem. In addition, we also use the *Partial-Greedy Algorithm* (PG) to solve this problem on a line and prove that PG is a $(1+(n-k)/\theta)$-competitive algorithm for this case. Finally, we extend the concepts of the on-line k-truck problem to obtain a new variant: the Deeper On-line k-truck Problem (DTP). For this variant of k-truck problem, we illustrate that all the results of PMS for the STP (Standard Truck Problem) still hold.

2. Preliminaries

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 to minimize the total cost of all trucks? Consider the following two cases,

(i) *Given a service request sequence, how can we schedule trucks so as to minimize the cost?*

(ii) *If the service request is received one by one in the process of service without any knowledge of the future requests, how can we minimize the cost as much as we can?*

Problem (i) is an off-line problem, and (ii) is an on-line problem. The difference between them depends on whether the known service request sequence is all or nothing. Problem (i) can be solved easily with dynamic programming, but problem (ii) 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. Obviously, the k-truck problem aims at minimizing the cost of all trucks. Because the cost of trucks with goods is different from that of trucks without goods on the same distance, the total distance cannot be considered as the objective to be optimized. For simplicity, we assume that the cost of a truck with goods is θ times that of one without goods on the same distance.

The Model. Let $G = (V, E)$ denote an edge weighted graph with n vertices and the weights of edges satisfying 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 that there are some goods at vertex a that must be moved to vertex b (for simplicity, we assume that the weight of the goods is same all the time). A service request sequence R consists of some service request in turn, namely $R = (r_1, ..., r_m)$, where $r_i = (a_i, b_i), a_i, b_i \in V$. The 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,

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

For a known sequence $R = (r_1, ..., r_m)$, let $C_{\text{OPT}}(R)$ be the optimal total cost after finishing it. For a new service request r_i, if scheduling algorithm A can schedule without information regarding 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

$$C_{\text{A}}(R) \leq \alpha \cdot C_{\text{OPT}}(R) + \beta \tag{2}$$

then for any possible R, A is called a competitive algorithm, where $C_{\text{A}}(R)$ is the total cost with algorithm A to satisfy sequence R.

If there is no limit for the R and θ, the on-line truck problem is called P. In problem P, for any $r_i = (a_i, b_i)$, if $a_i \neq b_i$ and $\theta > 1$ hold, the problem is called $P1$. In problem P, if there is no limit for any $r_i = (a_i, b_i)$, but if $\theta = 1$, the problem is $P2$. In problem P, for any $r_i = (a_i, b_i)$, if $d(a_i, b_i) > 0$, namely $a_i \neq b_i$, and $\theta = 1$, the problem is called $P3$. In problem P, for any $r_i = (a_i, b_i)$, if $d(a_i, b_i) = 0$, namely $a_i = b_i$, the problem is called $P4$.

Several Lemmas. For problem $P4$, Koutsoupias and Papadimitriou [10] showed that there exists an on-line algorithm for the k-server problem with competitive ratio $2k - 1$.

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

For the k-truck problem, [11] presented the following two lemmas.

Lemma 2 *Letting OPT be an optimal off-line algorithm for a request sequence* $R = (r_1, ..., r_m)$, *then we have* $C_{\text{OPT}}(R) \geq C_{\text{OPT}}(\sigma) + \sum_{i=1}^{m} (\theta - 1) \cdot d(a_i, b_i)$, *where* $\sigma = ((a_1, a_1), ..., (a_m, a_m))$ *and* $r_i = (a_i, b_i)$.

Lemma 3 *For any algorithm A for a request sequence* $R = (r_1, ..., r_m)$, *we have* $C_{\text{A}}(R) \geq \sum_{i=1}^{m} \theta \cdot d(a_i, b_i)$, *and* $C_{\text{OPT}}(R) \geq \sum_{i=1}^{m} \theta \cdot d(a_i, b_i)$.

In [6], the following lemma is given.

Lemma 4 *There exists an on-line algorithm for the k-server problem on a line with the competitive ratio k.*

Position Maintaining Strategy (PMS) In [17], the PMS was proposed to obtain several good research results for the k-taxi problem.

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.

3. Off-line Problem

In this section, two solutions for the off-line k-truck problem are discussed.One is *Dynamic Programming* (DP) and the other is *Minimum Cost Maximum Flow* (MCMF) in an acyclic network. First of all, we need to define the configuration as follows:

Definition (Configuration) *On the metric space M, a possible position of k trucks is called a configuration. That is, a configuration is a k-multiset whose elements consist of at least one and at most k points of space M. Here, the multiset is a set that may contain an elements multiple times.*

Obviously, the configuration is different from that for the famous k-server problem for which the configuration is defined as the multiset with exactly k different points of metric space M. Note that in our "special" set, the element can occur more than one times. Normally, in a set, the element can not be repeated, in other words, all elements are different.

3.1. Dynamic Programming (DP) Solution

In [13], a DP solution was given for the famous k-server problem. Similarly, we can develop a DP solution for the k-truck problem. However, because there are some differences between the k-server problem and k-truck problem, the relevant DP solutions are also somewhat different. First of all we introduce a lemma which is a standard result in combinatorics of multisets and with which the number of possible configurations k-trucks on a given graph with n nodes is shown.

Lemma 5 *On a given graph G with n nodes, the number of possible configurations of all k trucks is $\binom{n+k-1}{n-1}$, where $k \leq n$.*

Let $C_{\text{OPT}}(R, S)$ denote the cost of the minimum cost algorithm that handles request sequence R and ends up in configuration S. As in paper [13], we can compute this function recursively as follows, assuming that the trucks are initially in configuration S_0,

$$C_{\text{OPT}}(\varepsilon, S) = \begin{cases} 0 & \text{if } S = S_0 \\ \text{undefine} & \text{otherwise} \end{cases} \tag{3}$$

$$C_{\text{OPT}}(Rr_i, S) = \begin{cases} \min_T(C_{\text{OPT}}(R, T) + d(T, \theta \cdot (a_i, b_i), S)) & \text{if } b_i \text{ is covered in } S \\ \text{undefine} & \text{otherwise} \end{cases} \tag{4}$$

where $d(T, \theta \cdot (a_i, b_i), S)$ is the cost of transition from configuration T to configuration S and the last operation of transition is $a_i \to b_i$ (satisfying the request r_i at cost $\theta \cdot (a_i, b_i)$), T and S denote the configurations at time $i-1$ and time i, respectively, and ε denotes the empty request sequence.

Theorem 1 *The above optimal off-line algorithm for the k-truck problem can give an optimal solution with time proportional to $m \cdot \binom{n+k-1}{n-1}^2$, where m is the length*

of the request sequence (the number of requests).

Proof. Let $|R| = m$, we can develop a table-building method according to the above discussion. Build a table with $|R| + 1$ rows, each of which implies a subsequence of request sequence R, and $\binom{n+k-1}{n-1}$ columns each of which denotes a possible configuration of trucks. Namely, the entry in row i and column j is $C_{\text{OPT}}(R_i, S_j)$, where R_i is the subsequence of R of length i. Each row of the table can be built from the previous one within time $\binom{n+k-1}{n-1}^2$. Furthermore, only $|R| = m$ rows need these computations. The proof is completed. □

Besides, the above table-building procedure can produce an optimal sequence of moves. See the relevant discussion in [13].

3.2. Minimum Cost Maximum Flow (MCMF) Solution

In [5], MCMF was used to resolve the off-line k-server problem. As discussed in that paper, we can use this method to handle the case of the off-line k-truck problem. The main difference is how to construct the relevant network on which the MCMF is used to get the optimal solution of the off-line k-truck problem. Our objective is to find an optimal strategy to serve a sequence of m requests with k trucks, if the request sequence is given in advance. Assume that the k-trucks initially occupy one point, the origin. And denote the i-th request by the binary-tuple (a_i, b_i). If there are m requests, the inputs to our problem are the superdiagonal entries of an $(m+1) \times (m+1)$ matrix, whose $(0, j)$ entry is the sum of cost from the original to the location of j-request start a_j (empty) and then to the request destination b_j (with the goods), $j = 1, 2, ..., m$, and whose (i, j) entry is the sum of cost from the location of i-request destination to the location of j-request start and then to the relevant destination with goods, $j, 1 \leq i < j \leq m$.

Theorem 2 *There is an $O(km^2)$-time off-line algorithm to find an optimal schedule for k trucks to serve a sequence of m requests (whether or not the triangle inequality holds).*

Proof. We can resolve the off-line problem (with or without triangle inequality) by reducing it to the problem of finding a minimum cost flow of maximum value in an acyclic network. Suppose that there are k trucks $t_1, ..., t_k$ and m requests $r_1, ..., r_m$, where $r_i = (a_i, b_i)$, and $i = 1, ..., m$, we can build the following $(2 + k + 3m)$-node acyclic network: the vertex set is $V = \{s, s_1, ..., s_k, a_1, b_1, b'_1, ..., a_m, b_m, b'_m, t\}$. In that vertex set, nodes s and t are the source and sink, respectively. Each arc of our network has a capacity one. There is an arc of cost 0 from s to each s_i, an arc of cost 0 form each b'_i to t, as well as an arc to t from each s_i, of cost 0. From each s_i, there is an arc to a_j of cost equal to the distance from the origin to the location of a_j. From each a_j, there is only an arc to b_j of cost equal to $\theta \cdot d(a_i, b_i)$. For $i < j$, there is an arc from b'_i to a_j of cost equal to the distance between b_i to a_j. Moreover, from b_i to b'_i there is an arc of cost $-K$, where K is an extremely large positive real. The constructing of the network is completed.

It is easy to know that the value of the maximum flow in this network is k. Using minimum-cost augmentation [15], we can find an integral min-cost flow of value k in time $O(km^2)$, because all capacities are integral and the network is acyclic. An

integral $s \to t$ flow of value k can be decomposed into k arc-disjoint $s \to t$ paths, the ith one passing through s_i. Obviously, this flow saturates all of the (b_i, b_i') arcs, and hence corresponds to an optimal schedule for serving the requests, the ith server serving exactly those requests contained in the $s \to t$ path that passes through s_i, because $-K$ is so small. □

4. A Lower Bound

In this section we will give a lower bound of competitive ratio for the k-truck problem on a symmetric metric space. In other words, any general on-line algorithm for this problem, either a deterministic or a randomized algorithm, must have a competitive factor of at least $(\theta+1)\cdot k/(\theta\cdot k+2)$. In fact, we have actually proven a slightly more general lower bound on the competitive ratio. Suppose we wish to compare an on-line algorithm with k servers to an off-line one with $h \leq k$ servers. Naturally, the factor decreases when the on-line algorithm gets more servers than the off-line algorithm. We get the lower bound as $(\theta+1)\cdot k/((\theta+2)\cdot k-2h+2)$. A similar approach was taken in [13], where the lower bound and matching upper bound are given for the traditional k-server problem.

Theorem 3 *Let A be an on-line algorithm for the symmetric k-truck problem on a graph G with at least k nodes. Then, for any $1 \leq h \leq k$, there exists request sequences $R_1, R_2, \ldots$ such that: (1) For all i, R_i is an initial subsequence of R_{i+1}, and $C_A(R_i) < C_A(R_{i+1})$; (2) There exists an h-truck algorithm B (which may start with its trucks anywhere) such that for all i, $C_A(R_i) > (\theta+1)\cdot k\cdot C_B(R_i)/((\theta+2)\cdot k-2h+2)$.*

Proof. Without loss of generality, assume A is an on-line algorithm and that the k trucks start out at different nodes. Let H be a subgraph of G of size $k+2$, induced by the k initial positions of A's trucks and two other vertexes. Define R, A's nemesis sequence on H, such that R_i is the subsequence of R by the time i, $R(i)$ and $R(i-1)$ are the two unique vertexes in H not covered by A and a request $r_i = (R(i), R(i-1))$ occurs at time i, for all $i \geq 1$. Obviously, if the length (the number of the requests) of the R is t, then denote R by R_t and the cost of the algorithm A on the R by $C_A(R_t)$. Then

$$\begin{aligned} C_A(R_t) &= \sum_{i=1}^{t} (d(R(i+1), R(i)) + \theta \cdot d(R(i), R(i-1))) \\ &= (1+\theta)\cdot \sum_{i=1}^{t-1} d(R(i+1), R(i)) + d(R(i+1), R(i)) + \theta \cdot d(R(1), R(0)) \quad (5) \end{aligned}$$

because at each step R requests the node just vacated by A.

Let S be any h-element subset of H containing $R(1)$ but not $R(0)$. We can define an off-line h-truck algorithm $A(S)$ as follows: the trucks finally occupy the vertices in set S. To process a request $r_i = (R(i), R(i-1))$, the following rule is applied: If S contains $R(i)$, move the truck at node $R(i)$ to $R(i-1)$ with goods to satisfy the request, and update the S to reflect this change. Otherwise move the truck at node

$R(i-2)$ to $R(i)$ without goods and then to $R(i-1)$ with goods, also to satisfy the request, and update S to reflect this change.

It is easy to see that for all $i > 1$, the set S contains $R(i-2)$ and does not contain $R(i-1)$ when step i begins. The following observation is the key to the rest of the proof: if we run the above algorithm starting with distinct equal-sized sets S and T, then S and T never become equal, for the reason described in the following paragraph.

Suppose that S and T differ before $R(i)$ is processed. We shall show that the versions of S and T created by processing $R(i)$, as described above, also differ. If both S and T contain $R(i)$, they both move the truck at node $R(i)$ to node $R(i-1)$, at which there is exactly not any truck. The other nodes have no changes, so S and T are still different and both S and T contain $R(i-1)$. If exactly one of S or T contains $R(i)$, then after the request exactly one of them contains $R(i-1)$, so they still differ. If neither of them contains $R(i)$, then both change by dropping $R(i-2)$ and adding $R(i-1)$, so the symmetric difference of S and T remains the same (non-empty).

Let us consider simultaneously running an ensemble of algorithms $A(S)$, starting from each h-element subset S of H containing $R(1)$ but not $R(0)$. There are $\binom{k}{h-1}$ such sets. Since no two sets ever become equal, the number of sets remains constant. After processing $R(i)$, the collection of subsets consists of all the h element subsets of H which contain $R(i-1)$.

By our choice of starting configuration, step 1 just costs $\theta \cdot d(R(1), R(0))$. At step i (for $i \geq 2$), each of these algorithms either moves the truck at node $R(i)$ to $R(i-1)$ (if S contains $R(i)$), at cost $\theta \cdot d(R(i), R(i-1))$, or moves the truck at node $R(i-2)$ to $R(i)$ and then to $R(i-1)$ (if S does not contain $R(i)$), at cost $d(R(i-2), R(i)) + \theta \cdot d(R(i), R(i-1))$. Of the $\binom{k}{h-1}$ algorithms being run, $\binom{k-1}{h-1}$ of them (the ones which contain $R(i-2)$ but not contain either $R(i)$) incur the cost of $d(R(i-2), R(i)) + \theta \cdot d(R(i), R(i-1))$. The remaining $\binom{k-1}{h-2}$ of algorithms incur the cost of $\theta \cdot d(R(i), R(i-1))$. Thus, for step i, the total cost incurred by all of the algorithms is

$$\binom{k}{h-1} \cdot \theta \cdot d(R(i), R(i-1)) + \binom{k-1}{h-1} \cdot d(R(i-2), R(i)). \tag{6}$$

The total cost of running all of these algorithms up to and including R(t) is

$$\sum_{i=1}^{t} \binom{k}{h-1} \cdot \theta \cdot d(R(i), R(i-1)) + \sum_{i=2}^{t} \binom{k-1}{h-1} \cdot d(R(i-2), R(i)) \tag{7}$$

Thus the expected cost of one of these algorithms chosen at random is

$$C_{\text{EXP}}(R_t) = \theta \cdot \sum_{i=1}^{t} d(R(i), R(i-1)) + \frac{\binom{k-1}{h-1}}{\binom{k}{h-1}} \cdot \sum_{i=2}^{t} d(R(i-2), R(i))$$

$$\leq \frac{(\theta+2)k - 2h + 2}{k} \cdot \sum_{i=1}^{t-1} d(R(i), R(i+1)) +$$

$$\frac{(\theta+1)k-h+1}{k}\cdot d(R(1),R(0))-\frac{k-h+1}{k}\cdot d(R(t-1),R(t)) \tag{8}$$

This inequality holds for the triangle inequality and expending of the binomial coefficients.

Recall that the cost to A for the same steps was

$$C_{\mathrm{A}}(R_t)=(1+\theta)\cdot\sum_{i=1}^{t-1}d(R(i+1),R(i))+d(R(i+1),R(i))+\theta\cdot d(R(1),R(0)) \tag{9}$$

Because the distances are symmetric, the two summations of the $C_{\mathrm{EXP}}(R_t)$ and $C_{\mathrm{A}}(R_t)$ are identical, except that both of the costs include some extra terms, which are bounded as a constant. Therefore, after some mathematical manipulation (e.g., let $t\to\infty$), we obtain

$$\frac{C_{\mathrm{A}}(R_t)}{C_{\mathrm{EPT}}(R_t)}\geq\frac{(\theta+1)\cdot k}{(\theta+2)\cdot k-2h+2} \tag{10}$$

Finally, there must be some initial set whose performance is often no worse than the average of the costs. Let S be this set, and $A(S)$ be the algorithm starting from this set. Let R_i be an initial subsequence of R, for which $A(S)$ does no worse than average. □

Corollary 1 *For any symmetric k-truck problem, there is no c-competitive algorithm for* $c<(\theta+1)\cdot k/(\theta\cdot k+2)$.

Corollary 2 *For any symmetric k-taxi problem, there is no c-competitive algorithm for* $c<2k/(k+2)$.

5. Competitive Ratios

In this section, some new results of competitive ratio for the on-line k-truck problem are given. All results here are the improvements on those of [11].

5.1. Position Maintaining Strategy Solution

In [11], with the PMS, the case under which $\theta>(c+1)/(c-1)$ was studied, and a c-competitive algorithm was found to exist for the k-truck problem. In fact, we can get a somewhat better result for general cases.

Theorem 4 *For the on-line k-truck problem and a given graph G, if there is a c-competitive on-line algorithm for the k-server problem on G, then: (1) If* $\theta>(c+1)/(c-1)$, *then PMS is a c-competitive algorithm; (2) If* $1\leq\theta\leq(c+1)/(c-1)$, *then PMS is a* $(c/\theta+1/\theta+1)$*-competitive algorithm.*

Proof. For any $R=(r_1,...,r_m)$, where $r_i=(a_i,b_i)$, considering the k-server problem's request sequence $\sigma=(a_1,...,a_m)$, let A_σ be a c-competitive algorithm for the on-line k-server problem on graph G to satisfy the sequence. We design algorithm A as follows. For current service request $r_i=(a_i,b_i)$, first schedule a truck to a_i using algorithm A_σ, then complete the r_i with PMS. Thus total cost of A is

$$C_{\mathrm{A}}(R) = \sum_{i=1}^{m} C_{\mathrm{A}}(r_i)$$
$$= \sum_{i=1}^{m} [C_{\mathrm{A}}(a_i) + (\theta+1) \cdot d(a_i, b_i)]$$
$$= C_{A_\sigma}(\sigma) + (1 + 1/\theta) \cdot \sum_{i=1}^{m} \theta \cdot d(a_i, b_i) \tag{11}$$

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

$$C_{A_\sigma}(\sigma) \leq c \cdot C_{\mathrm{OPT}}(\sigma) + \beta$$
$$\leq c \cdot [C_{\mathrm{OPT}}(R) - \sum_{i=1}^{m} (\theta - 1) \cdot d(a_i, b_i)] + \beta \tag{12}$$

Then we get

$$C_{\mathrm{A}}(R) \leq c \cdot C_{\mathrm{OPT}}(R) + [1 + 1/\theta - c \cdot (\theta - 1)/\theta] \cdot \sum_{i=1}^{m} \theta \cdot d(a_i, b_i)] + \beta \tag{13}$$

Obviously,if $\theta > (c+1)/(c-1)$, we get $C_{\mathrm{A}}(R) \leq c \cdot C_{\mathrm{OPT}}(R) + \beta$; if $1 \leq \theta \leq (c+1)/(c-1)$, and with lemma 3, $C_{\mathrm{OPT}}(R) \geq \sum_{i=1}^{m} \theta \cdot d(a_i, b_i)$,we have $C_{\mathrm{A}}(R) \leq (c/\theta + 1/\theta + 1) \cdot C_{\mathrm{OPT}}(R) + \beta$, where c and β are some constants. □

In [11], for on-line k-truck problem in the case of $1 \leq \theta \leq (c+1)/(c-1)$, it has been proved that the PMS is $(c + 1/\theta + 1)$-competitive. Obviously, for this case, the result here is somewhat better than that of [11]. Furthermore, for the case of $\theta > (c+1)/(c-1)$ for the k-truck problem, although we just get the c-competitive algorithm, in fact, if $\theta \gg (c+1)/(c-1)$, the performance of PMS is better than what we have proved. For the k-taxi problem, which is a special case of the k-truck problem, we can let $\theta = 1$ and then get a $(c+2)$-competitive algorithm, which was given in [17].

Combining Theorem 4 and Lemma 1, the following corollary holds.

Corollary 3 *For the on-line k-truck problem on a given graph G, if* $\theta > (c+1)/(c-1)$, *holds, then there exists a* $(2k-1)$*-competitive algorithm; if* $1 \leq \theta \leq (c+1)/(c-1)$, *then there exists a* $(2k/\theta + 1)$*-competitive algorithm.*

5.2. Comparison of Two Algorithms

In [11], an algorithm B, here we called it the PG, is given for the problem $P1$. The competitive ratio of algorithm B is $1 + \lambda/\theta$, where $\lambda = d_{\max}/d_{\min}$, $d_{\max} = \max d(v_i, v_j)$, and $d_{\min} = \min d(v_i, v_j), i \neq j, v_i, v_j \in V$. We denote the PMS algorithm of subsection 5.1 by algorithm A. We may be confronted with the problem of choosing one algorithm from A and B in different contexts.

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

$$c_A = \begin{cases} 2k-1 & \text{if } \theta > (c+1)/(c-1) \\ 2k/\theta + 1 & \text{if } 1 \le \theta \le (c+1)/(c-1) \end{cases} \tag{14}$$

and $c_B = 1 + \lambda/\theta$. Letting $c_A = c_B$, we can get a k that makes the algorithm A and B equal as follows

$$k = \begin{cases} 1 + \lambda/(2\theta) & \text{if } \theta > (c+1)/(c-1) \\ \lambda/2 & \text{if } 1 \le \theta \le (c+1)/(c-1) \end{cases} \tag{15}$$

Obviously, the following theorem holds,

Theorem 5 *For on-line k-truck problem P1, denoting the PMS and PG algorithms by A and B, respectively, at the aspect of the competitive ratio: if* $\theta > (c+1)/(c-1)$ *holds, if* $k \le 1 + \lambda/(2\theta)$ *then A is better than B, and contrarily if* $k > 1 + \lambda/(2\theta)$ *then B is better than A; if* $1 \le \theta \le (c+1)/(c-1)$ *holds, if* $k \le \lambda/2$ *then A is better than B, and contrarily if* $k > \lambda/2$ *then B is better than A.*

5.3. Partial-Greedy Algorithm on A Special Line

Let $G = (V, E)$ for the instance of an on-line k-truck problem consisting of a line of n vertices with $n-1$ edges whose lengths are equal to one. More formally, we have that $V = \{v_i | i = 1, ..., n\}$ and $E = \{v_i v_{i+1} | i = 1, ..., n-1\}$. All edge-weights are equal to one. It is natural to assume that no vertex has more than one truck (otherwise, we can get at this situation at most cost of $k \cdot (k+1)/2$). This constant does not influence the discussion concerning the competitive algorithm. In fact, the problem considered here can be viewed as a generalization of the k-elevator problem [19]. In addition, we assume that $n \ge k+2$ holds (otherwise the fourth case of the following algorithm does not exist). In fact, in [11] the general cases of $n = k$ and $n = k+1$ were studied. For the k-truck problem on this special line, we give the following Partial-Greedy Algorithm which was so called because the Greedy Algorithm is used for some of the cases in this problem.

Partial-Greedy Algorithm. *For the current request* $r_i = (a_i, b_i)$ *from the request sequence* $R = (r_1, ..., r_m)$, *schedule the k-truck problem P1 on the above special line with the following rules:*

(1) If there is a truck at a_i *and also one at* b_i, *then PG moves the truck at* a_i *to* b_i *complete the request, and at the same time PG moves the truck at* b_i *to* a_i *with an empty load. The cost of PG 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 PG moves the truck at* a_i *to* b_i *to complete the request. The cost of PG 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 PG moves the truck at* b_i *to* a_i *first without a load, and after that moves it from* a_i *to* b_i *to complete*

the request. The cost of PG 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 PG moves the truck which is the closest to a_i (suppose that the truck is located at c_i) with an empty load and then moves to b_i to complete the request. The cost of PG 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.

Theorem 6 *PG is a $(1+(n-k)/\theta)$-competitive algorithm for the k-truck problem P1 on the above special line.*

Proof. For cases (1), (2) and (3), the cost of it PG 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 $d(c_i,a_i)\le(n-k)\cdot d(a_i,b_i)$. Let $C_{\mathrm{PG}}(R)$ denote the cost of algorithm PG for request sequence $R=(r_1,...,r_m)$, then we have

$$\begin{aligned} C_{\mathrm{PG}}(R) &= \sum_{i=1}^{m}\{max[d(b_i,a_i),d(c_i,a_i)]+\theta\cdot d(a_i,b_i)\}+\beta \\ &\le \sum_{i=1}^{m}\{(n-k)\cdot d(a_i,b_i)+\theta\cdot d(a_i,b_i)\}+\beta \\ &= (1+(n-k)/\theta)\cdot\sum_{i=1}^{m}\theta\cdot d(a_i,b_i)+\beta \\ &\le (1+(n-k)/\theta)\cdot C_{\mathrm{OPT}}(R) \end{aligned} \tag{16}$$

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. The last inequality holds for the lemma 3. □

Similar to subsection 5.2, combining the lemma 4 and the above theorem 6, we have the following theorem.

Theorem 7 *For on-line k-truck problem P1 on the special line, denoting the PMS and PG algorithms by A and B, respectively, at the aspect of the competitive ratio: if $\theta > (c+1)/(c-1)$ holds, if $k\le(n+\theta)/(\theta+1)$ then A is better than B, and contrarily if $k>(n+\theta)/(\theta+1)$ then B is better than A; if $1\le\theta\le(c+1)/(c-1)$ holds, if $k\le(n-1)/2$ then A is better than B, and contrarily if $k>(n-1)/2$ then B is better than A.*

6. Deeper On-line k-Truck Problem

We call the on-line k-truck problem studied in previous sections, the *Standard On-line k-truck problem* (STP). In this section we will discuss another interesting variant of it, the *Deeper On-line k-truck problem* (DTP). For each request $r_i=(a_i,b_i)$ of the STP, it is easy to see that when the on-line scheduler (or on-line decision-maker) decides which truck should be moved to serve the request, he knows all the information of the request; in other words, he knows both the start and destination nodes of the request. However, if he knows only the node on which the request occurs other than both the start and destination, namely, knowing a_i

but not knowing b_i, how does he schedule the k trucks in order to obtain a good competitive ratio? Obviously, the problem becomes difficult for the case with less information. We formulate DTP as follows:

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. However, only the node x of request occurring is known when the information of the request is received, and the destination node y will not be known until a truck has already been on the node of request occurring. To serve a request, an empty truck must first move to x and then move to y with goods from x. How to minimize the total cost of all trucks?

We easily know that the results of the competitive ratio of the PMS still hold for the DTP but those of the PG algorithm do not hold for the DTP.

Theorem 8 *For the DTP on a given graph G, if there is a c-competitive on-line algorithm for the k-server problem on G, then: (1) If $\theta > (c+1)/(c-1)$, then PMS is a c-competitive algorithm; (2) If $1 \leq \theta \leq (c+1)/(c-1)$, then PMS is a $(c/\theta + 1/\theta + 1)$-competitive algorithm.*

7. Concluding Remarks

This paper presents some new results concerning the k-truck problem, which is a generalization of the famous k-server problem. Most of the results of this paper can be extended to the relevant cases of the k-taxi problem [17], since the latter is a special case (just let $\theta = 1$) of the former. For problems $P1$ and $P4$ in this paper, there are many theoretical problems that need to be studied further. Paper [2] studied the on-line elevator problem, in which the parameter of time of request occurring is considered. The objective is to minimizing the completion time. Obviously, combining the concepts of the k-truck problem with the concepts of transportation in [2] is an attractive research direction. In this paper, employing the method of adversary we get a lower bound of competitive ratio for the k-truck problem. However, the optimal lower bound of the competitive ratio for it is still open. Furthermore, whether there are some on-line algorithms that have better competitive ratios than those of the PMS or the PG needs further investigation. In addition, for the DTP only the results of the PMS hold. Whether there are some better on-line algorithms for the DTP or not is still open. 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

1. S. Albers and S. Leonardi, "Online algorithms. " *ACM Computing Surveys.* Vol.31-3 (1999).
2. N. Ascheuer, S. O. Krumke, and J. Pambau. "Competitive scheduling of elevators." *Technical report,* (ZIB Berlin, 1998).
3. S. Ben-David, A. Borodin, R. M. Karp, G. Tardos, and A. Wigderson. "On the power if randomization in on-line algorithms." *Proc. 22nd Annual ACM Symp. on*

Theory of Computing. 1990 pp. 379-386.

4. A. Borodin, N. Linial, M. Sake, "An optimal on-line algorithm for metrical task systems." *Journal of ACM.* 39(1992) 745-763.
5. M. Chrobak, H. Karloff, T. Payne, S. Vishwanathan, "New results on the server problem," *SIAM Journal on Discrete Mathematics.* 4 (1991) 172-181.
6. M. Chrobak, L. Larmore, "An optimal algorithm for the server problem on trees," *SIAM Journal of Computing.* 20(1991)144-148.
7. A. Fiat and G. J. Woeginger, *Online algorithms-the state of the art. Lecture Note in Computer Science* (Springer, Berlin 1998).
8. A. Karlin, M. Manasse, L. Rudlph and D. D. leator, "Competitive snoopy caching," *Algorithmica.* 3(1988) 79-119.
9. H. A. Kierstead and W. T. Trotter, "An extreme problem in recursive combinatorics." *Congressus Numerantium.* 33(1981) 143-153.
10. E. Koutsoupias, C. Papadimitriou, "On the k-server conjecture," *Journal of the ACM,* 42(5)(1995) 971–983.
11. W. M. Ma, Y. F. Xu, and K. L. Wang, "On-line k-truck problem and its competitive algorithm," *Journal of Global Optimization.* 21 (1)(2001) 15-25.
12. M. S. Manasse, L. A. McGeoch, and D. D. Sleator, "Competitive algorithms for on-line problems." *Proc. 20th Annual ACM Symp. on Theory of Computing.* 1988 pp.322-33.
13. M. S. Manasse, L. A. McGeoch, and D. D. Sleator, "Competitive algorithms for server problems," *Journal of Algorithms.*11(1990) 208-230.
14. D. D. Sleator, R. E. Tarjan, "Amortized efficiency of list update and paging rules," *Communication of the ACM.* 28 (1985) 202-208.
15. R. Tarjan, *Data Structures and Network Algorithms* (SIAM, Philadelphia, 1983). pp. 109-111.
16. D. R. Woodall, "The bay restaurant- a linear storage problem." *American Mathematical Monthly.* 81(1974) 240-246.
17. Y. F. Xu, K. L. Wang, and B. Zhu, "On the k-taxi problem," *Information.* Vol.2-4 (1999).
18. A. C. C. Yao. "New algorithm for bin packing." *J. Assoc. Comput.Mach.* 27(1980) 207-227.
19. B. A. Ying, Y. F. Xu, and Y. Zhu, "On-line k-elevator problem and competitive algorithm," *Navigation Computing Technology.* 31(2)(2001)47-50. (In Chinese).