Journal of Global Optimization (2006) 34: 489–502
DOI 10.1007/s10898-005-2849-1

Competitive Analysis for the On-line Truck Transportation Problem

WEIMIN MA[1,2], JAMES N.K. LIU[2], GUOQING CHEN[1] and JANE YOU[2]
[1] *School of Economics and Management, Tsinghua University, Beijing 100084, P. R. China (e-mail: mawm@em.tsinghua.edu.cn)*
[2] *Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong (e-mails: csnkliu@comp.polyu.edu.hk; csyjia@comp.polyu.edu.hk)*

(Received: 9 March 2004; accepted: 16 August 2005)

Abstract. In this paper, the on-line k-truck transportation problem (k-OLTTP) whose objects are to be transported between the vertices of a given graph on which there are k mobile trucks to be scheduled is proposed. It is motivated by the research concerning on-line k-truck problem and on-line transportation problem. The goal is to minimize the makespan which is consumed to complete some on-line request sequence. Some preliminary knowledge is introduced and the model of k-OLTTP is established firstly. Two versions of a special case of k-OLTTP, namely 1-OLTTP, have been studied and some results are obtained. For the first version, Open-1-OLTTP, a lower bound of competitive ratio 2 is presented and two optimal on-line algorithms, Reschedule Strategy (RS) and Lay Over Strategy (LOS) respectively, are analyzed. For the second version, Close-1-OLTTP, a lower bound of competitive ratio $\frac{1}{2}+\frac{1}{2}\cdot\sqrt{1+\frac{4}{\theta}}$, where θ is the ratio between the time consumed by the loaded truck and the empty truck to travel the same distance, is also developed and on-line algorithms RS and LOS are proved to have competitive ratio 2. Finally, some interesting remarks concerning OLTTP and its future research are discussed.

Key words: competitive analysis, k-truck transportation problem, on-line algorithms.

1. Introduction

In traditional combinatorial optimization, transportation problems in which objects are to be transported between given sources and destinations in a metric space are always studied under the assumption that the complete input for an instance is available for an algorithm to compute a solution. However, in many cases this off-line optimization does not reflect the real-world situation. In this paper we consider the following new on-line truck transportation problem (OLTTP): Objects are to be transported between the vertices of a given graph. A request consists of the objects to be transported and the corresponding source and target vertex of the transportation request. The requests arrive on-line and must be handled by a truck which starts and ends its work at a designated origin vertex and moves along the paths in the graph. The truck picks up and drops objects at their starting points and destinations. We assume that neither the release time of

the last request nor the number of requests is known in advance. The goal of OLTTP is to come up with a transportation schedule for the truck in order to complete the whole request sequence as quickly as possible.

Our investigations of the OLTTP were originally motivated by the research results of the on-line k-truck problem [1], which is a generalization of the famous k-server problem, and on-line transportation problem [2], which is developed from the performance analysis of a large distribution center of Herlitz AG, Berlin [3]. The main differences between above problems and the OLTTP studied in this paper are as follows:

- In the on-line k-truck problem, the goal of optimization is to minimize the cost of all of the trucks and therefore the request consists of source and destination vertices. However, the goal of OLTTP is to minimize the completion time (makespan) and the relevant request involves three parameters: source and destination vertices and the request release time.
- In [2], the truck takes the same time on a given distance regardless of whether it is loaded or empty. In this paper, a more realistic assumption is that the time that a loaded truck takes to cover a certain distance is θ times the time taken by an empty truck, for some $\theta \geqslant 1$.

The k-truck problem was proposed in [1]. In that paper, the authors established the relevant model and gave some results concerning the competitive ratio of certain on-line algorithms. The 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 relevant to the k-server problem. More results concerning on-line k-truck problem and its variants were presented in [4]. In [2], the on-line transportation problems concerning the scheduling of elevators were well investigated and some relevant results were obtained. Some algorithms which were presented in [2] are employed to deal with the OLTTP problem. Different competitive ratios concerning the algorithms are obtained. The lower bounds of the competitive ratios of two different variants of the OLTTP are also shown. From another point of view, because the cases in that paper were the relevant special cases (just let $\theta = 1$), the relevant lower bounds of the competitive ratios are improved in this paper. In [5–7], some analogies of transportation problems and various shop scheduling problems were well studied. The main difference between this present paper and these previous works is that they use different models, e.g., existing the parameter θ in the model of OLTTP.

Over the past two decades, on-line problems and their competitive analysis have been the object of considerable interest. The systematic study of on-line problems started when Sleator and Tarjan [8] suggested comparing an on-line algorithm with an optimal off-line algorithm and Karlin, Manasse, Rudolph and Sleator [9] coined the term competitive analysis.

The k-server problem, introduced by Manasse et al. [10], generalizes paging as well as more general caching problems. The metrical task system, introduced by Borodin et al. [11], can model a wide class of on-line problems. In past years, many on-line problems were investigated in application 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 [12–15].

2. Preliminaries of OLTTP

The k-OLTTP problem can be stated as follows. Given a metric space M, there are k trucks which move among the points of M in order to serve requests. Repeatedly, a request (a pair of points $x, y \in M$ and a releasing time t) appears. To serve a request, an empty truck must first move to x and then move to y with objects from x no earlier than time t. The goal is to minimize the total time taken (makespan) of all trucks for some unknown request sequence. In order to present the difference between the on-line and off-line versions of the k-OLTTP, let us first consider the following problems:

(1) Given a service request sequence, how can we schedule trucks so as to minimize the makespan?
(2) 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 relevant time as much as we can?

Obviously, problem (1) can be solved easily because the request sequence, namely, the complete information required to make a decision making, is given in advance. In problem (2), however, in order to minimize the relevant time, the decision maker must make a less informed decision. Upon receiving a new request, a decision maker must adjust a schedule in the absence of knowledge of future requests. We refer to problem (1) as an *off-line* problem and to problem (2) as an *on-line* problem with the difference being whether the service request sequence is known in advance.

The k-OLTTP problem aims at minimizing the makespan of all request sequences. Because over the same distance the time taken by trucks carrying objects is different from that of trucks that are not carrying objects, we cannot regard the total distance as the objective of optimization. For simplicity, we assume that over the same distance the time taken by a truck carrying objects is θ times that of a truck that is not carrying objects.

The Model. Let $G = (V, E)$ denote an edge weighted graph with n vertices, where V is a metric space consisting of n vertices, and E is the set of all weighted edges. For all $x, y, z \in V$, we assume that the weight of edge

(x, y) is denoted by $d(x, y)$ which actually means the shortest path between x and y. The weights are symmetric, i.e., $d(x, y) = d(y, x)$ and the weights of edges satisfy the triangle inequality, i.e., $d(x, y) + d(x, z) \geqslant d(y, z)$. In addition, for any point s on the edge (u, v) at distance $d(s, u)$ from u and distance $d(s, v) = d(u, v) - d(s, u)$ from v, its distance from any other point x on the graph G is defined by $\min\{d(s, u) + d(u, x), d(s, v) + d(v, x)\}$. We assume that k trucks occupying a distinguished origin vertex $o \in V$ at time $t = 0$ can be scheduled to complete the whole service. The ith request is defined as a triplet $r_i = (t_i, a_i, b_i)$, where $a_i, b_i \in V$ imply that there are some objects on vertex a_i that must be moved to vertex b_i. For simplicity, we assume that the weight of the objects is the same all the time. t_i denotes the release time which indicates the earliest starting time that the request can be dealt with i is a natural number. A service request sequence R is defined as an ordered list of service requests, namely $R = (r_1, \ldots, r_m)$. The on-line k-OLTTP scheduling problem is to devise some algorithms so as to minimize the makespan under the condition that the requests arrive in an on-line fashion.

All discussion is based on the following assumptions:

(1) The graph G is connected;
(2) All trucks move at two different constant speed units whether empty or loaded;
(3) Over the same distance, the time taken by a loaded truck is θ times that of an empty truck, and $\theta \geqslant 1$;
(4) Preemption is forbidden: once the truck has picked up the object, it may not drop it at any place other than its destination.

For a known sequence $R = (r_1, \ldots, r_m)$, let $C_{\mathrm{OPT}}(R)$ be the optimal makespan to complete all requests of R. For a new service request, if algorithm A can schedule without prior 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

$$C_{\mathrm{A}}(R) \leqslant \alpha \cdot C_{\mathrm{OPT}}(R) + \beta,$$

then for any possible R, A is called a α-competitive algorithm and α is called the competitive ratio [16], where $C_{\mathrm{A}}(R)$ is the total time taken with algorithm A to satisfy sequence R.

There are two different versions of OLTTP: (a) the trucks need not return to the origin vertex after having served the last request of the sequence; (b) the trucks must return to the origin vertex. The first version is called the Open-k-OLTTP and the second version is called the Close-k-OLTTP. These terms were coined in [2]. This paper addresses only cases

where $k=1$ and obtains some results. Cases where $k>1$ remain for further investigation.

3. Open OLTTP with One Truck

This section presents some results concerning the Open-1-OLTTP: a lower bound and some competitive algorithms with a good competitive ratio.

3.1. A LOWER BOUND OF COMPETITIVE RATIO FOR OPEN-1-OLTTP

For Open-1-OLTTP, we will present a lower bound to illustrate how an on-line algorithm performs compared with an optimal off-line algorithm. We have the following theorem.

THEOREM 3.1. No deterministic algorithm for Open-1-OLTTP can achieve a competitive ratio $c<2$.

Proof. The underlying graph $G=(V, E)$ for the instance of OLTTP consists of a real line with the origin being the vertex o. We denote any vertex by v_i which satisfies that $d(o, v_i)=i$.

Suppose that A is a deterministic on-line algorithm with competitive ratio c. We can choose the number n so large that $\theta<(n-2)/3$ and $2 \geqslant 2-(4\cdot\theta+1)/(n-1+\theta)>c$. Otherwise, the proof will be trivial.

Now the off-line optimal player constructs a request sequence as follows. At time $t_0=0$, the algorithm A is faced with the first request $r_0=(0, v_0, v_{-1})$. Thus we can claim that at time $t_1=n-1$ the on-line truck cannot be strictly to the right, e.g., on the path from $v_{2\theta}$ to $+\infty$ but not on $v_{2\theta}$. Otherwise, if the truck were to the right, say at distance $\delta>0$ to the right of vertex $v_{2\theta}$, then we could add the request $r_1=(n-1, v_{-(n-\theta)}, v_{-(n-\theta+1)})$. The on-line truck would then need a period of at least $n+2\cdot\theta+\delta$ to satisfy this request. This would result in a total time of $2\cdot(n-1)+2\cdot\theta+\delta+1$. On the other hand, the off-line truck could serve r_0 starting at time t_0 and then continue to move to the left until it reached vertex $v_{-(n-\theta)}$ at time $n-1$ ready to serve the new request r_1. Thus the off-line truck needs time $n-1+\theta$ to serve the requests. We have

$$\frac{C_{\mathrm{A}}(r_0, r_1)}{C_{\mathrm{OPT}}(r_0, r_1)}=\frac{2\cdot(n-1)+2\cdot\theta+\delta+1}{n-1+\theta}>2$$

which means that A would not be 2-competitive.

We have seen that at time $t_1=n-1$ the on-line truck is to the left of vertex $v_{2\theta}$. We now add the request $r_1'=(n-1, v_{n-2-\theta}, v_{n-1-\theta})$. To serve this request needs time at least $n-2-2\cdot\theta$ and then the total time of on-line

service is at least $2 \cdot n - 3 - 2 \cdot \theta$. On the other hand, the off-line truck serves the sequence (r_0, r_1') by handling r_0 at time t_0 and then immediately moving to vertex $v_{n-2-\theta}$ and reaches it at time $n-1$ and is ready to serve r_1'. Thus, we have that $C_{\mathrm{OPT}}(r_0, r_1') = n - 1 + \theta$ and then

$$\frac{C_{\mathrm{A}}(r_0, r_1')}{C_{\mathrm{OPT}}(r_0, r_1')} = \frac{2 \cdot n - 3 - 2 \cdot \theta}{n - 1 + \theta} = 2 - \frac{4 \cdot \theta + 1}{n - 1 + \theta} > c.$$

This contradicts the assumption that A is c-competitive with $c < 2$. □

3.2. TWO OPTIMAL ON-LINE ALGORITHMS FOR OPEN-1-OLTTP

For a request sequence R and a point x, let $L(t, x, R)$ denote the shortest time taken. This starts at the point x at time t and serves all requests of R. Obviously, its value is the time difference between its completion time and the start time t. Intuitively, for $t' > t$, $L(t', x, R) \leqslant L(t, x, R)$. Moreover, $C_{\mathrm{OPT}}(R) = L(0, o, R)$ and thus $C_{\mathrm{OPT}}(R) \geqslant L(t, o, R)$ for any time $t \geqslant 0$. In addition, according to the definition of the request, the optimal off-line truck OPT cannot start to deal with the last request $r_m = (t_m, a_m, b_m)$ from R before this request is released. Therefore we get $C_{\mathrm{OPT}}(R) \geqslant t_m + \theta \cdot d(a_m, b_m)$. Finally, for any $t \geqslant 0$ the following inequality holds,

$$C_{\mathrm{OPT}}(R) \geqslant \max\{L(t, o, R), t_m + \theta \cdot d(a_m, b_m)\}.$$

In order to prove the main theorems, we need to prove the following Lemmas first.

LEMMA 3.2. For any request sequence $R = \{r_1, \ldots, r_m\}$, any request $r_i = (t_i, a_i, b_i)$ from R and any time $t \geqslant t_m$, the following inequality holds,

$$L(t, b_i, R \backslash r_i) \leqslant L(t, o, R) - \theta \cdot d(a_i, b_i) + d(e, o),$$

where e is the end point occupied by the truck on the path related to $L(t, o, R)$ and $R \backslash r_i$ denotes the request sequence excluding r_i.

Proof. Let S^* denote an optimal schedule which starts at the origin o at time t and deals with all requests in R. According to definitions above, $C_{S^*}(R) = L(t, o, R)$ holds. Obviously, to prove the lemma, it is enough to construct another schedule S which starts at b_i no earlier than time t and serves all requests in $R \backslash r_i$ and the inequality $C_{\mathrm{S}}(R \backslash r_i) \leqslant L(t, o, R) - \theta \cdot d(a_i, b_i) + d(e, o)$ holds.

Apparently, different schedules may deal with different requests in different orders. Assume S^* deals with the requests in the order $r_{j1}, \ldots, r_{jm}$ such that $r_i = r_{jk}$. Then construct schedule S which starts at b_i at time t and deals with the request sequence $R \backslash r_i$ in the order

$$r_{j(k+1)}, \ldots, r_{jm}, r_{j1}, \ldots, r_{j(k-1)}.$$

Note that the most possible on additional consuming time is $d(e, o)$.

LEMMA 3.3. For any request sequence $R = \{r_1, \ldots, r_m\}$, let $R_{\geqslant t_S}$ denote the subsequence of R in which the release time of the every request is after t_S. Then the following inequality holds

$$t_S + L(t_S, o, R_{\geqslant t_S}) \leqslant C_{\mathrm{OPT}}(R) + d_{\max},$$

where $d_{\max} = \max\{d(x, y)\}$, for any $x, y \in V$.

Proof. For any off-line optimal algorithm OPT, according to the definition of $R_{\geqslant t_S}$, all requests in the algorithm must be dealt with after t_S. Let the request $r_f = (t_f, a_f, b_f)$ be the first request of $R_{\geqslant t_S}$ that is deal with. Then we have $t_S \leqslant t_f$ and

$$t_S + L(t_S, o, R_{\geqslant t_S}) \leqslant t_f + L(t_f, a_f, R_{\geqslant t_S}) + d(o, a_f) \leqslant C_{\mathrm{OPT}}(R) + d_{\max}.$$

In [2], the authors presented the on-line algorithms Reschedule and Lay Over (which they called Replan and Ignore) strategies for some special cases of OLTTP. In fact, these algorithms are also competitive for the OLTTP. Their algorithms are as follows.

Reschedule Strategy (RS): When a new request arrives, the truck always reformulates a new shortest schedule after completing the current request (if it is dealing with one). Otherwise it continues to perform the current schedule. The new shortest schedule takes account of all remaining requests and starts at the current position and then either stops (for Open-OLTTP) or returns to the origin vertex (for Close-1-OLTTP).

Note the difference between *schedule* and *request* in this algorithm. The *schedule* can be adjusted if a new request arrives. The process of serving a request cannot, however, be interrupted. As for the competitive ratio of the algorithm Reschedule Strategy, we get the following theorem.

THEOREM 3.4. For Open-1-OLTTP, the algorithm RS is 2-competitive.

Proof. Let $R = \{r_1, \ldots, r_m\}$ be a request sequence and the latest request becomes known at time t_m. Apparently the following two cases need to be considered.

Case 1. The truck is empty at time t_m. A new optimal schedule is computed which starts at the truck's current position, denoted by $s(t_m)$, and which deals with all remaining requests. For the new schedule the following inequality holds

$$L(t_m, s(t_m), R) \leqslant d(o, s(t_m)) + L(t_m, o, R).$$

Therefore,

$$\begin{aligned} C_{\mathrm{RS}}(R) &\leqslant t_m + d(o, s(t_m)) + L(t_m, o, R) \\ &\leqslant t_m + d(o, s(t_m)) + C_{\mathrm{OPT}}(R) \\ &\leqslant 2 \cdot C_{\mathrm{OPT}}(R) + d_{\max}. \end{aligned}$$

Case 2. The truck is currently serving a request $r = (t, a, b)$. The time needed to complete the current request is $\theta \cdot d(s(t_m), b)$. Then a shortest schedule starting at b and serving all remaining requests is computed which has a length at most $L(t_m, b, R \backslash r)$. Thus in Case 2

$$\begin{aligned} C_{\mathrm{RS}}(R) &\leqslant t_m + \theta \cdot d(s(t_m), b) + L(t_m, b, R \backslash r) \\ &\leqslant t_m + \theta \cdot d(s(t_m), b) + L(t_m, o, R) - \theta \cdot d(a, b) + d(e, o) \\ &\leqslant t_m - \theta \cdot d(a, s(t_m)) + d(e, o) + C_{\mathrm{OPT}}(R) \\ &\leqslant t_m + C_{\mathrm{OPT}}(R) + d(e, o) \\ &\leqslant 2 \cdot C_{\mathrm{OPT}}(R) + d_{\max}. \end{aligned}$$

The second step of this inequality holds for lemma 3.2. For Cases 1 and 2, let $d_{\max} = \beta$, then we have

$$C_{\mathrm{RS}}(R) \leqslant 2 \cdot C_{\mathrm{OPT}}(R) + \beta.$$

Now let us see another optimal competitive algorithm for the Open-1-OLTTP.

Lay Over Strategy (LOS) [2]: The truck always continues to operate according to the current optimal schedule and lays over all requests that arrive before it completes all the tasks on the current schedule. The truck then follows a new optimal schedule that deals with all remaining requests and performs it immediately to start new circulation.

THEOREM 3.5. For Open-1-OLTTP, the LOS is a competitive algorithm with competitive ratio 2.

Proof. Two cases need to be considered at t_m when the last request r_m becomes known.

Case 1. If the truck is currently idle, it then completes the last task on its schedule no later than

$$\begin{aligned} C_{\mathrm{LOS}}(R) &\leqslant t_m + d(s(t_m), a_m) + \theta \cdot d(a_m, b_m) \\ &\leqslant C_{\mathrm{OPT}}(R) + d(s(t_m), a_m) \\ &\leqslant C_{\mathrm{OPT}}(R) + d_{\max}. \end{aligned}$$

Case 2. Suppose the truck is following a current schedule S for a subsequence R_S of R at time t_m. Assume S will start at time t_S at vertex x and end at vertex y. The set of requests that are served by Lay Over in its last schedule is denoted by $R_{\geqslant t_S}$. We have

$$\begin{aligned} C_{\mathrm{LOS}}(R) &\leqslant t_S + L(t_S, x, R_S) + L(t_m, y, R_{\geqslant t_S}) \\ &\leqslant t_S + L(t_S, o, R_{\geqslant t_S}) + d(o, y) + L(t_S, o, R_S) + d(o, x) \\ &\leqslant 2 \cdot C_{\mathrm{OPT}}(R) + d(o, y) + d(o, x) + d_{\max} \\ &\leqslant 2 \cdot C_{\mathrm{OPT}}(R) + 3 \cdot d_{\max}. \end{aligned}$$

For the above two cases, let $\beta = 3 \cdot d_{\max}$. Then we get

$$C_{\mathrm{RS}}(R) \leqslant 2 \cdot C_{\mathrm{OPT}}(R) + \beta.$$

4. Close OLTTP with One Truck

This section presents some results concerning the Close-1-OLTTP: a lower bound and some competitive algorithms with a good competitive ratio.

4.1. A LOWER BOUND OF COMPETITIVE RATIO FOR CLOSE-1-OLTTP

For Close-1-OLTTP, a lower bound will be presented to illustrate how an on-line algorithm performs compared with the optimal off-line algorithm. We have the following theorem.

THEOREM 4.1. No deterministic algorithm for Close-1-OLTTP can achieve a competitive ratio $c < \frac{1}{2} + \frac{1}{2} \cdot \sqrt{1 + \frac{4}{\theta}}$.

Proof. The underlying graph $G = (V, E)$ for the instance of OLTTP consists of a positive real line with the origin being the vertex o. We denote any vertex which satisfies $d(o, v_i) = i$ by v_i.

Suppose that A is a deterministic on-line algorithm with a competitive ratio c and $c < \frac{1}{2} + \frac{1}{2} \cdot \sqrt{1 + \frac{4}{\theta}}$. Otherwise, there is nothing left to prove.

Now the off-line optimal player constructs a request sequence as follows. At time $t_0 = 0$, the algorithm A receives two requests $r_1 = (0, o, v_n)$ and $r_2 = (0, v_n, o)$. We note that $C_{\mathrm{OPT}}(r_1, r_2) = 2n \cdot \theta$ and algorithm A must deal with request r_2 at some time T which satisfies the inequality $n \leqslant T \leqslant 2n \cdot \theta \cdot c - n \cdot \theta$. In addition, if $n \leqslant T < n \cdot \theta$ holds, request r_1, which takes at least $n \cdot \theta$, cannot be satisfied by algorithm A before time T. Therefore, $C_{\mathrm{A}}(r_1, r_2) \geqslant T + 2n \cdot \theta + n \geqslant 2n \cdot (\theta + 1)$. Then we have

$$\frac{C_{\mathrm{A}}(r_1, r_2)}{C_{\mathrm{OPT}}(r_1, r_2)} \geqslant \frac{2n \cdot (\theta + 1)}{2n \cdot \theta} = 1 + \frac{1}{\theta} \geqslant \frac{1}{2} + \frac{1}{2} \cdot \sqrt{1 + \frac{4}{\theta}},$$

which contradicts the assumption that A is c-competitive with $c < \frac{1}{2} + \frac{1}{2} \cdot \sqrt{1 + \frac{4}{\theta}}$. Now we can claim that $n \cdot \theta \leqslant T \leqslant 2n \cdot \theta \cdot c - n \cdot \theta$.

At time T the off-line player releases requests $r_3 = (T, v_n, v_{n+1})$ and $r_4 = (T, v_{n+1}, v_n)$. The on-line truck would then need time of at least $C_{\mathrm{A}}(r_1, r_2, r_3, r_4) \geqslant T + n \cdot \theta + 2\theta + 2n$. On the other hand, the off-line truck could deal with r_1 starting at time 0 and then stay at the vertex v_n until time T to deal with r_3, r_4 and finally, r_2 at a total cost of $C_{\mathrm{OPT}}(r_1, r_2, r_3, r_4) = T + n \cdot \theta + 2\theta$. We get the following inequality

$$\begin{aligned}
\frac{C_{\mathrm{A}}(r_1, r_2, r_3, r_4)}{C_{\mathrm{OPT}}(r_1, r_2, r_3, r_4)} &\geqslant \frac{T + n \cdot \theta + 2\theta + 2n}{T + n \cdot \theta + 2\theta} \\
&\geqslant \frac{(2n \cdot \theta \cdot c - n \cdot \theta) + n \cdot \theta + 2\theta + 2n}{(2n \cdot \theta \cdot c - n \cdot \theta) + n \cdot \theta + 2\theta} \quad (\text{by } T \leqslant 2n \cdot \theta \cdot c - n \cdot \theta) \\
&= \frac{n \cdot \theta \cdot c + \theta + n}{n \cdot \theta \cdot c + \theta},
\end{aligned}$$

namely, the following inequality holds

$$c \geqslant \frac{n \cdot \theta \cdot c + \theta + n}{n \cdot \theta \cdot c + \theta}.$$

Considering the case when $n \to \infty$ (maximize the right side of above inequality) and making some mathematical manipulations, we can get $c \geqslant \frac{1}{2} + \frac{1}{2} \cdot \sqrt{1 + \frac{4}{\theta}}$ and this contradicts the position that A is a c-competitive algorithm with competitive ratio $c < \frac{1}{2} + \frac{1}{2} \cdot \sqrt{1 + \frac{4}{\theta}}$. □

4.2. RS AND LOS ALGORITHMS FOR CLOSE-1-OLTTP

Similar to that in section 3, for a request sequence R and a point x of the Close-1-OLTTP, let $L^*(t, x, R)$ denote the shortest time taken which starts at the point x at time t and deals with all requests of R and ends at the origin vertex. Its value is the difference between its completion time and the start time t. Note that the difference of above definition for $L(t, x, R)$ is that we require the schedule to end at the origin. We also get the following results: for $t' > t$, we have that $L^*(t', x, R) \leqslant L^*(t, x, R)$; $C_{\mathrm{OPT}}(R) = L^*(0, o, R)$ and thus $C_{\mathrm{OPT}}(R) \geqslant L^*(t, o, R)$ for any time $t \geqslant 0$; and $C_{\mathrm{OPT}}(R) \geqslant t_m + \theta \cdot d(a_m, b_m) + d(b_m, o)$ for the optimal off-line truck OPT cannot start to deal with the last request $r_m = (t_m, a_m, b_m)$ from R before this request is released. Finally, for any $t \geqslant 0$ the following inequality holds:

$$C_{\mathrm{OPT}}(R) \geqslant \max\{L^*(t, o, R), t_m + \theta \cdot d(a_m, b_m) + d(b_m, o)\}.$$

We also have two lemmas similar to lemmas 3.2 and 3.3.

LEMMA 4.2. For Close-1-OLTTP and any request sequence $R=\{r_1, \ldots, r_m\}$, any request $r_i = (t_i, a_i, b_i)$ from R and for any time $t \geqslant t_m$, the following inequality holds:

$$L^*(t, b_i, R\backslash r_i) \leqslant L^*(t, o, R) - \theta \cdot d(a_i, b_i) + d(a_i, o),$$

where $R\backslash r_i$ denotes the request sequence excluding r_i.

Note that lemma 3.3 also holds for the case of Close-1-OLTTP. Now let us see how to prove that algorithms RS and LOS are also competitive for the Close-1-OLTTP.

THEOREM 4.3. For Close-1-OLTTP, the algorithm Reschedule Strategy is 2-competitive.

Proof. Let $R = \{r_1, \ldots, r_m\}$ be a request sequence and the latest request becomes known at time t_m. The following two cases need to be considered.

Case 1. The truck is empty at time t_m. In this case, a new optimal schedule is computed which starts at its current position, denoted by $s(t_m)$, deals with a11 remaining requests and returns to the origin vertex. For the new schedule the following inequality holds

$$L^*(t_m, s(t_m), R) \leqslant d(o, s(t_m)) + L^*(t_m, o, R).$$

Therefore,

$$\begin{aligned} C_{\mathrm{RS}}(R) &\leqslant t_m + d(o, s(t_m)) + L^*(t_m, o, R) \\ &\leqslant t_m + d(o, s(t_m)) + C_{\mathrm{OPT}}(R) \\ &\leqslant 2 \cdot C_{\mathrm{OPT}}(R) + d_{\max}, \end{aligned}$$

where $d_{\max} = \max\{d(x, y)\}$, for any $x, y \in V$.

Case 2. The truck is currently dealing with a request $r = (t, a, b)$. The time needed to complete the current request is $\theta \cdot d(s(t_m), b)$. Then a shortest schedule which starts at b and deals with all un-served requests and ends at origin vertex is computed. This schedule takes at most $L^*(t_m, b, R\backslash r)$. Thus in Case 2

$$\begin{aligned} C_{\mathrm{RS}}(R) &\leqslant t_m + \theta \cdot d(s(t_m), b) + L^*(t_m, b, R \backslash r) \\ &\leqslant t_m + \theta \cdot d(s(t_m), b) + L^*(t_m, o, R) - \theta \cdot d(a, b) + d(a, o) \\ &\leqslant t_m - \theta \cdot d(a, s(t_m)) + d(a, o) + C_{\mathrm{OPT}}(R) \\ &\leqslant t_m + C_{\mathrm{OPT}}(R) + d(a, o) \\ &\leqslant 2 \cdot C_{\mathrm{OPT}}(R) + d_{\max}. \end{aligned}$$

The second step of this inequality holds for lemma 4.2. For both cases, 1 and 2, let $d_{\max} = \beta$. Then we have

$$C_{\mathrm{RS}}(R) \leqslant 2 \cdot C_{\mathrm{OPT}}(R) + \beta.$$

THEOREM 4.4. For Close-1-OLTTP, the Lay Over Strategy is a strictly competitive algorithm with the competitive ratio 2.

Proof. Two cases need to be considered at t_m when the last request r_m becomes known again.

Case 1. If the truck is currently idle, it must stay at the origin vertex and it completes its last scheduled task no later than

$$\begin{aligned} C_{\mathrm{LOS}}(R) &\leqslant t_m + d(o, a_m) + \theta \cdot d(a_m, b_m) + d(b_m, o) \\ &\leqslant C_{\mathrm{OPT}}(R) + d(o, a_m) \end{aligned}$$

Case 2. If the truck is performing a current schedule S for a subsequence R_S of R at time t_m, and assume S will start at time t_S. Let $R_{\geqslant t_S}$ denote the set of requests that are dealt with by Lay Over in its last schedule. We have

$$\begin{aligned} C_{\mathrm{LOS}}(R) &\leqslant t_S + L^*(t_S, o, R_S) + L^*(t_m, o, R_{\geqslant t_S}) \\ &\leqslant C_{\mathrm{OPT}}(R) + L^*(t_S, o, R_{\geqslant t_S}) \\ &\leqslant 2 \cdot C_{\mathrm{OPT}}(R). \end{aligned}$$

Consider these cases together, since $d(o, a_m) \leqslant C_{\mathrm{OPT}}(R)$ the following inequality holds for any case

$$C_{\mathrm{RS}}(R) \leqslant 2 \cdot C_{\mathrm{OPT}}(R).$$

5. Concluding Remarks

In this paper we have studied two versions of the 1-OLTTP problem and employed the algorithms RS and LOS to get some surprisingly small competitive ratios. We have also obtained the relevant lower bound of their competitive ratios. From the lower bound of competitive ratio for Open-1-OLTTP, we know that the algorithms RS and LOS are optimal on-line

algorithms having a competitive ratio 2. It is very interesting to consider other variants of OLTTP. For instance, if the on-line player knows the time when the last request becomes known (but has no information about the other requests before they are released), a simple strategy (known as WAIT) can also give a competitive ratio 2. We can describe the WAIT Strategy as follows:

WAIT Strategy: The on-line truck will do nothing until the last request is released at time t_m. It will then deal with all requests from time t_m according to an optimal schedule.

For any request sequence R, $C_{\mathrm{WAIT}}(R) \leqslant t_m + L(t_m, o, R_S) \leqslant 2 \cdot C_{\mathrm{OPT}}(R)$ holds. Then competitive ratio of 2 is obtained. In addition, if the on-line player knows only the number of requests, then the player can also use the WAIT Strategy to get the competitive ratio 2.

For the OLTTP problem, further research could be of interest. For example, is there a better lower bound for the competitive ratio of Close-1-OLTTP? It should also be borne in mind that all the results presented in this paper operate with only one truck. Operating more than one truck would present further challenges.

Acknowledgements

The authors are grateful to support of the National Natural Science Foundation of China (70401006, 70231010), the China Postdoctoral Science Foundation (2003034014) and the Departmental Research Grant of The Hong Kong Polytechnic University (H-ZJ87).

References

1. Ma, W.M., Xu, Y.F. and Wang, K.L. (2001), On-line k-truck problem and its competitive algorithm, *Journal of Global Optimization*, 21(1), 15–25.
2. Ascheuer N., Krumke S.O. and Rambau J. (1999), The on-line transportation problem: competitive scheduling of elevators. Preprint No. 98/24, Konrad-Zuse für Informationstechnik, Berlin(ZIB).
3. Ascheuer, N., Grotschel, M., Krumke, S.O. and Rambau, J. (1998), Combinatorial online optimization. In: *Proc. of the International Conference of Operations Research (OR'98)*, Zurich Springer.
4. Ma, W.M., Xu, Y.F., You, J., Liu, J. and Wang, K.L. (2004), On the k-Truck scheduling problem, *International Journal of Foundations of Computer Science*, 15(1), 127–141.
5. Beck, J.C., Prosser P. and Selensky E. (2002), Graph transformations for the vehicle routing and job shop scheduling problems. *ICGT 2002*, *LNCS 2505*, 60–74.
6. Blaz̲ewicz, J., Domschke, W. and Pesch E. (1996), The job shop scheduling problem: conventional and new solution techniques, *European Journal of Operational Research*, 93, 1–33.
7. Chen, B. and Vestjens, A.P.A. (1997), Scheduling on identical machines: how good is LPT in an on-line setting? *Operations Research Letter*, 21, 165–169.

8. Sleator, D.D. and Tarjan, R.E. (1985), Amortized efficiency of list update and paging rules, *Communication of the ACM*, 28, 202–208.
9. Karlin, M.M. Rudlph, L. and Sleator, D.D. (1988), Competitive snoopy caching, *Algorithmica*, 3, 79–119.
10. Manasse, M.S., McGeoch, L.A. and Sleator, D.D. (1988), Competitive algorithms for on-line problems. In: *Proc. 20th Annual ACM Symp. on Theory of Computing*, 322–333.
11. Borodin, A., Linial, N. and Sake, M. (1992), An optimal on-line algorithm for metrical task systems, *Journal of ACM*, 39, 745–763.
12. Ben-David, S., Borodin, A., Karp, R.M., Tardos, G. and Wigderson, A. (1990), On the power of randomization in on-line algorithms. In: *Proc. of the 22nd Annual ACM Symp. on Theory of Computing*, 379–386.
13. Ma, W.M. You, J., Xu, Y.F., Liu, J. and Wang, K.L. (2002), On the on-line number of snacks problem, *Journal of Global Optimization*, 24(4), 449–462.
14. Albers, S. and Bals, H. (2003), Dynamic TCP acknowledgement: Penalizing long delays. In: *Proc. 14th ACM-SIAM Symposium on Discrete Algorithms*, 47–55.
15. Bartal, Y and Mendel, M. (2004), Randomized Algorithms for the k-Server Problem in Growth Rate Bounded Metric Spaces. In: *Proc. of the 15th ACM-SIAM Symp. on Discrete Algorithms*, January 2004.
16. Borodin, A. and El-Yaniv, R. (1998), *Online Computation and Competitive Analysis*, Cambridge University Press.