Journal of Combinatorial Optimization (2022) 44:2143–2166
https://doi.org/10.1007/s10878-020-00585-1

Online traveling salesman problem with time cost and non-zealous server

Tengyu Wu[1] · **Lin He**[2] · **Haiyan Yu**[3]

Published online: 25 May 2020

Abstract
Considering that the time of meeting the demands is very important for emergency vehicle and emergency vehicle can't reject any request, we introduce a more realistic cost form into online traveling salesman problem(OL-TSP). When a new request arrives, if the salesman can't serve the request immediately, per-unit-time cost will be generated. The goal is to minimize server's total costs(travel makespan plus the per-unit-time costs). We consider the server is a non-zealous server and show that neither deterministic nor randomized online algorithms can achieve constant competitive ratio for OL-TSP on general metric space. While on truncated line segment and uniform metric space, we prove lower bounds, and present competitive online algorithms. Especially for the case with uniform metric space, we prove an optimal Greedy algorithm.

Keywords Online algorithm · Emergency vehicle routing · Traveling salesman problem · Per-unit-time cost

1 Introduction

Consider the following traveling salesman problem: any request cannot be rejected, and if the salesman can't serve the request immediately, there occurs an extra cost which increases with the waiting time for the salesman. This problem conforms with reality such as in emergency vehicle rescue. In order to save more lives and transfer the

✉ Tengyu Wu
fly200205@163.com

[1] Chongqing smart post Engineering Technology Research Center, School of Economics and Management, Chongqing University of Posts and Telecommunications, Chongqing, China

[2] School of Economics and Management, Chongqing University of Posts and Telecommunications, Chongqing, China

[3] Insititute of Chongqing Port Logistics and Shipping, School of Economics and Management, Chongqing Jiaotong University, Chongqing, China

emergency materials to affected sites, the emergency vehicle can't reject any request. Zhou et al. (2017) consider that for the emergency vehicle, the sooner it meets the demands of the affected area, the better. For affected population, after sending out signal for help, they always want to be served immediately. A longer waiting time may cause more dissatisfaction and victims. For injured people, a longer waiting time may be life-threatening which can be seen an extra cost for the emergency vehicle. For example, in the earthquake rescue, the time of the rescue is really important for the affected population, one day after the earthquake, the survival rate is nearly 90 percent, and after two or three days, the survival rate may be 70 to 80 percent. Moreover, in order to cut the cost and improve efficiency, the emergency vehicles want to minimize the final time which servers are back at the origin plus the costs associated with the requests which can't be served immediately. The situations mentioned above are very similar to traveling salesman problem with per-unit-time cost.

For the time cost in TSP, Campbell and Thomas (2008) study the probabilistic TSP with deadline, they consider a per-unit-time charge of violation charge for violating the deadline. The cost for per-unit-time charge is represented by a customer-dependent λ_i (which is the same with the parameter B in our model). For instance, FedEx Custom Critical refunds varying percentages of the cost of a shipment based on how late the shipment is delivered (FedEx 2005). For additional examples, see (Charnsirisakskul et al. 2004) and (Slotnick and Sobel 2005). Our model is the online version of Campbell and Campbell and Thomas (2008)'s model when deadline is equal to 0 without considering the probabilistic case.

In this paper, we consider the online version of the problem. The requests arrive over-time, and the server has to decide how to move without any information of the future requests. There are a few researches concerning OL-TSP with the extra cost, but the cost form is different from the cost form in our model, the cost is generated if the server rejects the request or misses the deadline, and the server can't serve it. Ausiello et al. (2008) introduce the costs for unserved requests into online quota TSP, and call it online prize-collecting TSP, the objective of the server is to collect a given amount of quota while minimizing the sum of time to complete the service and the costs. Jaillet and Lu (2011) consider the situation that the server only serves the accepted requests, there is a cost for rejecting a request, and the objective is to minimize the time to serve all accepted requests plus the sum of penalties associated with the rejected requests. Wen et al. (2015) introduce the deadline into the OL-TSP with service flexibility, the server can choose whether to serve or not when a new request arrives, by rejecting the request or missing its deadline, costs will be generated, the objective is to minimize server's total costs. Yu et al. (2014) study the Online Quota Traveling Salesman Problem, and present optimal deterministic algorithms for each variant defined on a general space, a real line, or a half-line. Above literatures neither consider the cost which increases with the waiting time nor consider the situation that the server can't reject the request.

The similar online problem is OL-TRP(on line Traveling Repairman Problem) which is an extension of OL-TSP, Krumke et al. (2003) study the OL-TRP, and the objective is to minimize the latency. Irani et al. (2004) study the dynamic OL-TRP, and the objective is to serve maximum requests before the deadline. The most related research is the Net-latency OL-TRP, Allulli et al. (2008) study the Net-latency OL-TRP,

the objective function is to minimize the net latency (which is similar to the cost form in our model). They show when the lookahead information $\Delta > 2D$(D is the diameter of the network), there exits a competitive online algorithm. Especially on the line segment, they give an online algorithm with lookahead information, and the competitive ratio depends on the lookahead information. Different from the model(Allulli et al. 2008), we consider not only the cost but also the makespan(the earliest time for the server to go back to the origin after serving all the requests), give the lower bound and a competitive online algorithm on the truncated line segment without any lookahead information. Our result is an extension and improvement of their work.

Compared with previous research, we have several contributions: (1) We study the TSP with per-unit-time cost by considering an online version which matches several real-life scenarios. (2) We try to minimize the total cost of traveling time and the time cost as an objective instead of only one item. (3) We consider the non-zealous algorithm. Previous research only considers the zealous algorithm (Allulli et al. 2008), where the direction of online server changes only if a new request becomes known, or the server is either in the origin or at a request that has just been served (Blom et al. 2001). Taking realities into consideration, we consider non-zealous algorithm (Lipmann 2003), where the online server can change its directions at any time and at any place. The main contribution is the analysis of OL-TSP with per-unit-time cost. The results are the general extension of OL-TSP by considering the per-unit-time cost form. In previous research, they consider the scenario that the server does not serve the request if violating the deadline or the server rejects the request which generates the cost, but in our research, we consider the delayed service which leads the costs, this means that the later the service, the larger the cost. We show that neither deterministic nor randomized online algorithms can achieve constant competitive ratios on general metric spaces. While on the truncated line segment, we give a lower bound and an *Observe and Move Algorithm* whose competitive ratio is related to the length of the line. While on uniform metric space, we give a lower bound and a *Greedy Algorithm*, and prove that the *Greedy Algorithm* is optimal.

The rest of the paper is structured as follows. In Sect.2, we introduce online traveling salesman problem with per-unit-time cost and non-zealous server. In Sect.3, we show that there are no constant competitive ratios for either deterministic or randomized online algorithms on general metric spaces. In Sect.4, we analyze the problem on the truncated line segment, give a lower bound, and an *Observe and Move Algorithm* with its competitive ratio. In Sect.5, we analyze the problem on the uniform metric space, give a lower bound, and a *Greedy Algorithm* with its competitive ratio. Final conclusions are given in Sect.6.

2 Preliminaries

Given a general metric space M. Requests arrive over-time and request R_i is denoted as $R_i = (r_i, l_i)$, $i \in N$, $N = \{1, 2, 3 \cdots, n\}$, where n is the number of request. r_i represents the request's release time, and l_i represents the position of R_i. Denote s_i as the service time (the moment that the server arrives at l_i), and p_i is the cost that the server can't serve the request R_i immediately which equals to $B(s_i - r_i)$, where

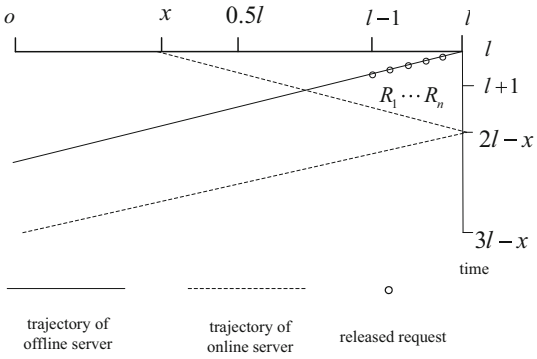

Fig. 1 Trajectory of online and offline server

$B(B \geq 1)$ is a cost rate. The problem starts at time 0 (initial state), and the server is at the origin. The server can only be idle or move at unit speed. The server must be idle at the origin after serving all the requests(final state). The earliest time for the server to reach the final state is called makespan. The objective of the problem is to minimize the total cost, i.e. makespan plus the total cost $\sum_{i=1}^{n} p_i$.

3 Lower bounds for OL-TSP with per-unit-time cost

3.1 Deterministic case

Theorem 1 *No deterministic online algorithm can achieve a constant competitive ratio, even on the positive half line.*

Proof Consider the positive half line. At time $l(l \geq 6)$, we consider the different positions of the online server, suppose the online server is at point x.

Case 1. When $x \leq 0.5l$, the adversary releases n requests denoted as $R_{1\leq i\leq n} = (l + \frac{i}{n}, l - \frac{i}{n})$. After serving all the n requests, the online server and the offline server would be back to the origin(see Fig. 1).

As shown in Fig. 1, the cost of the offline server is equal to $c_{opt} = 2l$. At time $l + 1$, the online server arrives at point $x + 1$, and the cost for each request in $R_{1\leq i\leq n}$ is at least equal to $l - x - 2 > 1$, the cost of the online server is equal to $c_{online} > 3l - x + n(l - x - 2)B > 2.5l + nB$. We have $\frac{c_{online}}{c_{opt}} > \frac{2.5l+nB}{2l} \to \infty$ for $n \to \infty$.

Case 2. When $x > 0.5l$, the offline server arrives at point l at time l, the adversary releases n requests denoted as $R_{1\leq i\leq n} = (l + \frac{i}{n}, 1 + \frac{i}{n})$. After serving all the n requests, the online server and the offline server would go back to the origin. The cost of the offline server is equal to $c_{opt} = l + 2$, and the cost of the online server is equal to $c_{online} > l + x + n(x - 2)B > 1.5l + nB$. We have $\frac{c_{online}}{c_{opt}} > \frac{1.5l+nB}{l+2} \to \infty$ for $n \to \infty$.

3.2 Randomized case

Since the deterministic online algorithms perform badly against the adversary, the randomized online algorithms seem interesting. In this section, we only consider the randomness of online server's position. For the randomness of the requests's release time, see Simroth and Souza (2009)

Theorem 2 *No randomized online algorithm can achieve a constant competitive ratio, even on the positive half line.*

Proof Consider the positive half line. At time 0, the offline server moves towards point $l(l \geq 4)$. Because the online server is a non-zealous server, the online server may stay any position between 0 and l, at time l. The probabilities are the same for online server to stay at these positions. The position of the online server would be uniformly distributed in the segment $[0, l]$, the offline server arrives at point l. The adversary releases n requests denoted as $R_{1\leq i\leq n} = (l + \frac{i}{n}, l - \frac{i}{n})$. After serving all the n requests, the online server and the offline server must go back to the origin. The offline server can serve all the requests without any extra cost. The online server's cost is correlated with the position of the online server at time l. Suppose the online server would generate $c_{online}(1)$ if it is located during $[0, l-3]$. At time $l+1$, the online server is located during $[1, l-2]$, the cost of each request in $R_{1\leq i\leq n}$ is at least equal to 1 for the online server. Thus we have $c_{online} > c_{online}(1) > \frac{l-3}{l}(\int_0^l (l-x)dx + l) + n\frac{l-3}{l}B$, and $\frac{c_{online}}{c_{opt}} \to \infty$ for $n \to \infty$.

4 OL-TSP with per-unit-time cost on truncated line segment

Deterministic and randomized online algorithms can't perform well against the offline adversary. It is interesting to explore the performance of online algorithms on restricted metric spaces. In this section, similar to Wen et al. (2012) and Gutiérrez et al. (2006), we consider the truncated line segment $[-L, L]$ with unit distance between nodes, and $L \geq 3$ as an integer. The requests can only be released at nodes, and every node has only one unserved request at a time.

4.1 A lower bound

Theorem 3 *The lower bound for OL-TSP with per-unit-time cost on truncated line segment is* $1 + 0.75LB$*(*L *is the half length of the truncated line,* B *is a cost rate).*

Proof Because the adversary is an omnipotent one which can release the requests at the positions of the offline server, and this would decrease the cost of the offline server. The enlarged distance between the online server and the offline server would increase the cost of the online server. With all the situations mentioned above, the adversary would first release request sequence as follows. The classification of the case analysis in the proof of lower bound is based on the position of the online server and the strategy of the online server after the newly released request.

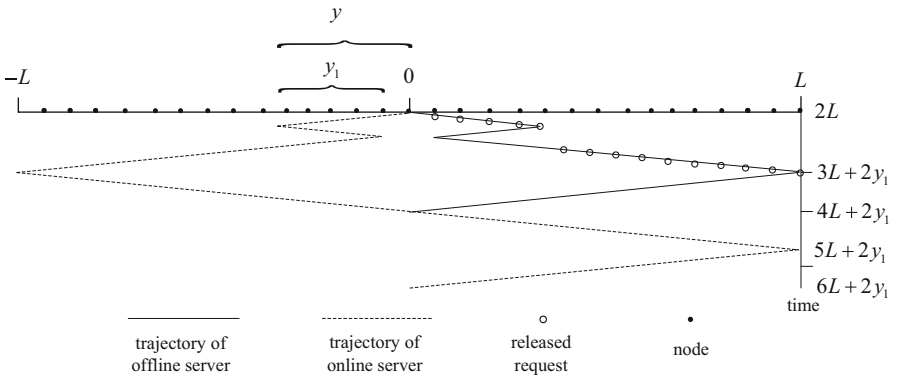

Fig. 2 Trajectory of online and offline server

At time L, there are two cases for the online server: arrive at the extreme nodes(L or $-L$) and arrive at other positions.

Case 1. The online server arrives at the extreme nodes(L *or* $-L$).

Without loss of generality suppose the online server arrives at node L at time L, the offline server arrives at node $-L$. The adversary would release L requests on the negative half line, denoted as $R_{1\leq i\leq L} = (L + i - 1, -L + i - 1)$. The online server must move towards $-L$, then the online server and the offline server meet at the origin. If the offline server is idle at the origin, the cost of the offline server is equal to $c_{offline} = 2L$, the cost of the online server is equal to $c_{online} = 4L + (L^2 + L)B$, the cost ratio is equal to $\frac{c_{online}}{c_{offline}} = 2 + \frac{(L+1)B}{2}$. Now we consider the offline server starts to move again at time $2L$, the offline server moves straightly to L, the adversary releases requests at the offline server's positions. There are two cases for the online server, in these two cases, we only consider the added cost.

Case 1.1 The online server goes straightly to node $-L$, *then turns around.*

When the offline server is idle at the origin again at time $4L$, the offline server's cost increases by $2L$, the online server's cost increases by $2L + 2L^2B$, the cost ratio is equal to $\frac{2L+2L^2B}{2L} = 1 + LB$.

Case 1.2 The online server travels a distance $y(y < L)$, *then turns around.*

Once the offline server finds that the online server turns around, the offline server turns around too. The adversary releases requests at the offline server's positions without unserved requests for the online server. There are two cases for the online server after turning around:

Case 1.2.1 After turning around, the online server travels a distance $y_1(y_1 \leq y)$ *then turns around to node* $-L$.

The online server doesn't serve any requests on the right side of the origin in this case (See Fig. 2).

When the offline server is idle at the origin again, the offline server's cost increases by $2L+2y_1$. For the online server's cost, we notice that the requests between node $-y$ and node $-L$ increases by $2(L-y)y_1$, the online server's cost increases by $2L+2y_1 + (2y_1(L-y)+2y_1y+2L^2)B$, the cost ratio is equal to $\frac{2L+2y_1+(2y_1(L-y)+2y_1y+2L^2)B}{2L+2y_1} = 1 + LB$.(See "Appendix" A for details)

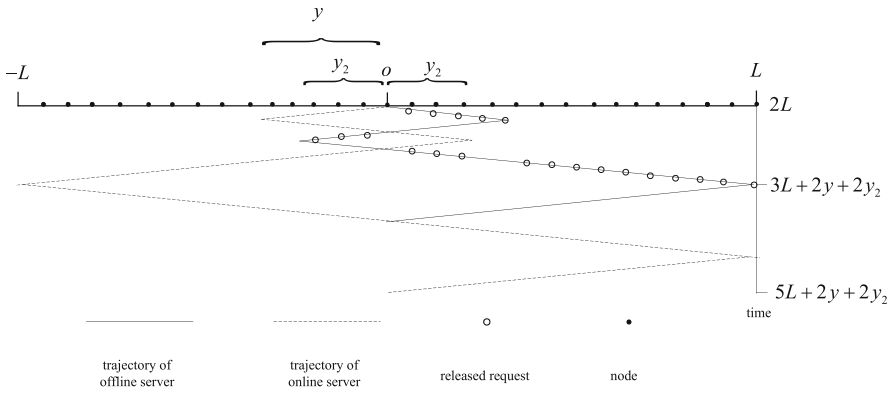

Fig. 3 Trajectory of online and offline server

Case 1.2.2 After turning around, the online server travels a distance $y + y_2$*, then turns around to the origin.*

The online server serves $min\{y, y_2\}$ requests on the right side of the origin in this case. When the offline server is idle at the origin again, the offline server's cost increases by $2L + 2y + 2y_2$.

When $y_2 \leq y$, the online server serves y_2 requests on the right side of the origin(See Fig. 3). The adversary releases y_2 new requests on the left side of the origin. The online server's cost increases by $2L+2y+2y_2+(2yy_2+2{y_2}^2+2(L-y_2)(y+y_2)+2L^2)B$(See "Appendix" B for details), the cost ratio is equal to

$$\frac{2L+2y+2y_2+(2yy_2+2{y_2}^2+2(L-y_2)(y+y_2)+2L^2)B}{2L+2y+2y_2} = 1 + LB.$$

When $y_2 > y$, the online server serves y requests on the right side of the origin. The adversary releases y new requests on the left side of the origin. The online server's cost increases by $2L + 2y + 2y_2 + (2(y + y_2)(L - y) + 2y^2 + 2yy_2 + 2L^2)B$(See "Appendix" 1 for details), the cost ratio is equal to

$$\frac{2L+2y+2y_2+(2(y+y_2)(L-y)+2y^2+2yy_2+2L^2)B}{2L+2y+2y_2} = 1 + LB$$

According to above analysis, the strategy of turning around for the online server wouldn't decrease the cost ratio, so the online server wouldn't turn around until arriving at node $-L$. We denote the situation (the offline server moves to an extreme node, then goes back to the origin) as a cycle. For one cycle, the cost of the offline server would increase by $2L$, and the cost of the online server would increase by $2L + 2L^2B$, after n cycles, the cost ratio is equal to

$\frac{c_{online}}{c_{offline}} = \frac{4L+(L^2+L)B+n(2L+2L^2B)}{2L+2nL}$, when $n \to \infty$, $\frac{c_{online}}{c_{offline}} = 1 + LB > 2 + \frac{(L+1)B}{2}$, $L \geq 3$, $B \geq 1$.

Case 2. The online server arrives at other positions

Without loss of generality suppose the online server is on the negative half line or at the origin at time L, the offline server arrives at L. We consider the parity of L.

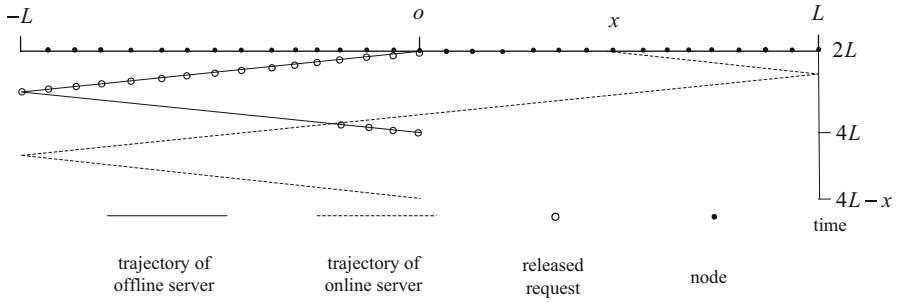

Fig. 4 Trajectory of online and offline server

When L is even, the adversary releases $2L + 1$ requests denoted as $R_{1\le i\le L} = (L + i - 1, L - i + 1)$. The offline server would arrive at node $-L$ at time $3L$. If at time $3L$, the online server dozen't serve the request at node L, the offline server waits at node $-L$ until online server arrives at node L. Notice that this is case 1, and the cost ratio is equal to $1 + LB$. If at time $3L$, the online server has served the request at node L which means that at time $2L$, the online server's position is $x(0 < x \le L)$. The online server's strategy is to serve the requests on the positive half line firstly, then turn around to serve the requests on the negative half line(otherwise the online server can't serve the request at node L before time $3L$). We consider the cycles which means that whenever the offline server arrives at the origin, the offline server would move to the extreme node again which is farther from the online server(we notice that the online server and the offline server don't meet at the origin in this case). The adversary releases the requests at the offline server's positions. For the online server, similar to above analysis, the online server must arrive at the extreme node which has an unserved request before the offline server arrives at another extreme node(See Fig. 4), otherwise the case will become case 1.

For each cycle, offline server increases by $2L$, and that of the online server increases by $2L + (2L^2 - 0.5x^2 + 2L - x)B$. When $x = L$, the online server's added cost can achieve its minimum value, which is equal to $2L + (1.5L^2 + L)B$. After n cycles, the cost ratio is equal to $\frac{c_{online}}{c_{offline}} = \frac{3L+c+n(2L+(1.5L^2+L)B)}{2L+2nL}$, where c is the cost of the first L requests. When $n \to \infty$, $\frac{c_{online}}{c_{offline}} = 1 + (0.75L + 0.5)B$, $L \ge 3$, $B \ge 1$.

When L is odd, similar to above analysis, the cost ratio is equal to $\frac{c_{online}}{c_{offline}} = 1 + (0.75L + 0.75)B$.

In the above cases, we don't consider the "waiting" strategy for the online server. In case 1, if the online server adopts "waiting" strategy, so does the offline server, the cost ratio is still the same. In case 2, if the online server adopts "waiting" strategy, the case is similar with case 1 which will increase the ratio.

In conclusion of Case 1 and Case 2, we can get the lower bound for OL-TSP with per-unit-time cost on truncated line segment as $1 + 0.75LB$.

4.2 Observe and move algorithm and its competitive ratio

The main purpose of this section is to design OM algorithm and get the competitive ratio. In order to describe OM algorithm clearly, we prove the best way of releasing the requests for the adversary (Lemma 1). According to Lemma 1, we define the *Released request sequence*. Based on the definition of the *Released request sequence*, we give the description of *Observe and Move algorithm*. The main idea of OM algorithm is that through observing the released request sequence, the online server will decide to keep his direction or turn around. In order to get the competitive ratio, we prove that after the occurring of *Meeting situation 2*, the offline server must go to the extreme node (L or $-L$) or go to the origin and idle (Lemma 2). At last, we prove that no matter how the adversary releases the requests, *Meeting situation 2* must happen. We get the competitive ratio.

Lemma 1 *Once the online server and the offline server are not at the same position, the cost ratio of online algorithm A won't decrease if the adversary only releases the requests at the positions of the offline server.*

Proof After the separation of the online server and the offline server, define a request sequence set $S = \{S_i, i = 1...k\}$, in S the adversary only releases the requests at the positions of the offline server, and every request sequence in S begins at the time when offline server leaves the origin, and ends at the time when the offline server goes back to the origin for the second time.

For S_m, the offline server's cost is equal to y_m, and the online server's cost is equal to z_m. Suppose the adversary only releases a request R_j at node j which isn't at the position of the offline server, the release time is assumed to be t unit time earlier than the arrival time, the offline server's cost is equal to $y_m + Bt$.

Now we consider the online server's cost. If based on algorithm A, the online server's trajectory is the same with the trajectory when the adversary releases the request sequence S_m, there are two cases for the online server:

Case 1. The position of node j is between the meeting position of the two servers and the online server, the online server's cost is equal to $z_m - Bt$, and the cost ratio is equal to $\frac{z_m - Bt}{y_m + Bt} < \frac{z_m}{y_m}$;

Case 2. The position of node j is at other positions except the positions in case 1, the online server's cost is equal to $z_m + Bt$, and the cost ratio is equal to $\frac{z_m + Bt}{y_m + Bt} < \frac{z_m}{y_m}$.

If based on algorithm A, the online server's trajectory isn't the same with the trajectory when the adversary releases the request sequence S_m(this may also change the trajectory of the offline server), after the offline server serves all the requests, we can find that the request sequence S_m turns into a new request sequence S_l in S(for the unserved requests of the online server, the adversary can't release new requests on the nodes, the online server and the offline server will meet at different points which would lead to a different released request sequence). For the new request sequence S_l, we can analyze it as above.

We don't consider that the offline server would stay at some point without moving(except staying at the origin at the final state). Because the adversary can't release

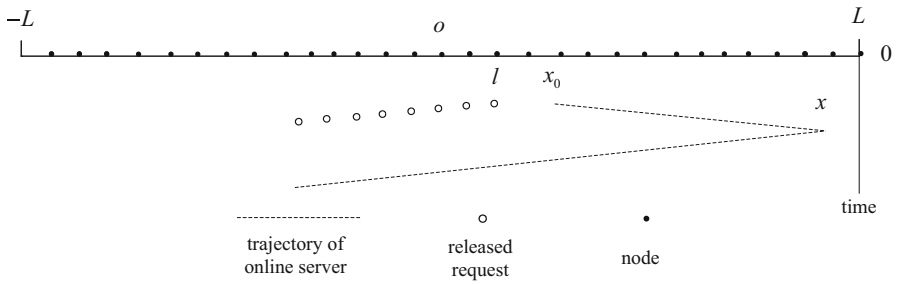

Fig. 5 Released request sequence 1

any new request which would lead to a higher offline server's cost and a lower online server's cost according to Lemma 1 and the hypothesis.

4.2.1 Observe and move algorithm

According to Lemma 1, after the separation of the online server and the offline server, the adversary must release the requests at the positions of the offline server. In the proof of the lower bound, the adversary also releases the requests at the positions of the offline server. When the online server serves these requests, the cost of each request for the online server is the same. For the online server, turning around before serving these requests is no good. This kind of released request must be considered specially. In order to describe the Observe and Move Algorithm clearly, the definitions of two released request sequences is given firstly:

Released request sequence 1: For a given request sequence S, the online server must serve it, if the cost of each request in S for the online server is the same, which equals to $2\,|x - 0.5(x_0 + l)|$(where x is the position where the online server turns around, l is the position of the first released request in S, x_0 is the position of online server when the first request in S is released), we call this sequence S *Released request sequence 1*(Especially, only one request can be seen as *Released request sequence 1*)(see Fig. 5).

Released request sequence 2: The other released request sequences except *released request sequence 1* are denoted as *Released request sequence 2*.

We give the *OM Algorithm* in the following table.

The main idea of the algorithm is that the online server observes the released request sequence, and then he makes the decisions (keeping the direction or turning around). If the online server is serving the *released request sequence 1*, no matter the direction of the online server, he must serve all the requests of the *released request sequence 1* then turns around. If the online server isn't serving the *released request sequence 1*, when the online server is moving towards the extreme node, the newly released requests behind the online server satisfy *released request sequence 1*, the online server won't turn around until serving all the requests in front of the online server, if not, turn around immediately; when the online server is moving towards the origin, the strategy is quite different. Once the newly request has been released behind the online server, the online server turns around immediately.

Observe and Move Algorithm

1. Stay still until there are newly released requests, choose the optimal route that generates the smallest cost. After online server leaves the origin, once a new request is released, go to step 2.
2. If online server is moving towards the extreme node L (or $-L$) or arriving at origin, go to step 3; else if online server is moving towards the origin from $-L$(or L) or arriving at the extreme node L (or $-L$), go to step 4.
3. The online server observes the request sequence in front of the online server first.
   - 3.1. If the request sequence in front of the online server contains *released request sequence 1*, the online server goes straightly to the rightmost (or to the leftmost) unserved request of the *released request sequence 1* then go to step 5.
   - 3.2. Else the online server observes the request sequence behind the online server.
     - 3.2.1. If the request sequence behind the online server contains *released request sequence 1*, the online server keeps his direction, serve all the requests in front of the online server then turns around, then go to step 5.
     - 3.2.2. Else the online server turns around immediately to serve all the requests on the left(or right) side of the online server, then go to step 2.
4. The online server observes the request sequence in front of the online server first.
   - 4.1. If the request sequence in front of the online server contains *released request sequence 1*, the online server goes straightly to the rightmost (or to the leftmost) unserved request of the *released request sequence 1*, then go to step 5.
   - 4.2. Else the online server observes the request sequence behind the online server.
     - 4.2.1. If any request is released behind the online server, the online server turns around immediately to serve all the requests on the left(or right) side of the online server, then go to step 5.
     - 4.2.2. Else the online server keeps his direction until a new request is released or the requests have been served, then go to step 5.
5. The online server observes whether there are unserved requests.
   - 5.1. If there are no unserved requests, the online server goes back to the origin. During the travel when a new request is released, go to step 2.
   - 5.2. Else if there are unserved requests, go to step 2.

4.2.2 Competitive ratio

Before we give the proof of the competitive ratio, we give the definitions of two meeting situation:

Meeting situation 1: Online server moves towards the origin, and offline server moves towards the extreme node(L or $-L$). When two servers meet at some point on the truncated line, we call it *Meeting situation 1*.

Meeting situation 2: Online server moves towards the extreme node(L or $-L$), and offline server moves towards the origin. When two servers meet at some point on the truncated line, we call it *Meeting situation 2*.

Especially, when two servers meet at the origin, we define this meeting as *Meeting situation 2*. The classification of the case analysis in Lemma 2 is based on the *Meeting situation 2*'s occurrence.

Lemma 2 *After Meeting situation 2 occurs, the offline server must go straightly to the origin and idle or keep his direction to the extreme node, and this won't decrease the cost ratio of OM algorithm.*

Proof Without loss of generality, suppose that the *Meeting situation 2* occurs at node $-x$(for simplicity of the analysis, suppose x is an integer and we suppose all the

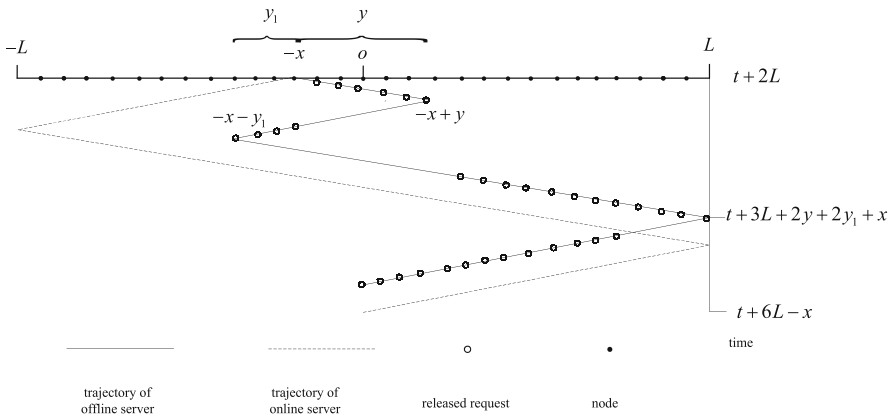

Fig. 6 Trajectory of online and offline server

parameters are integer in the following part). Before this meeting, the offline server wouldn't turn around until the offline server arrives at the extreme node (L or $-L$). According to Lemma 1 and OM algorithm, the last meeting of the two servers occurs at node x at time t, it was also *Meeting situation 2*. Now we only consider two servers' cost after time $t + 2L$. Based on the different movement of the offline server, we consider three cases:

Case 1. The offline server goes straightly to the origin or node L.

When the offline server goes straightly to node L, then turns around back to the origin, according to Lemma 1 and OM algorithm, the online server and the offline server will meet at node $L - x$. The cost ratio is equal to $\rho_0 = \frac{4L-x+2(L-x)(2L+x)B}{2L+x}$.

When the offline server goes straightly to the origin, the online server and the offline server won't meet again. According to Lemma 1 and OM algorithm, the cost ratio is equal to $\rho = \frac{2(L-x)LB+2L-x}{x}$.

We will compare the cost ratio of the following cases with ρ or ρ_0. If the value of ρ or ρ_0 is larger, we can prove Lemma 2.

Case 2. The offline server travels a distance of y, then turns around, before the meeting happens, turns around again.

If two servers don't meet for the first turning around of the offline server(the two servers will eventually meet), the value of y must satisfy $1 \leq y < L-x$, if $y \geq L-x$, the offline server must turn around on the right side of node $-x$. The adversary can't release any new request which would decrease the cost ratio.

The offline server turns around to node $-x - y_1 (0 \leq y_1 \leq L - x - y - 1)$(this assures that two servers wouldn't meet at the negative half line), then goes straightly to node L, goes back to the origin and stays still. According to Lemma 1, the adversary releases requests at the positions of the offline server, but the adversary can't release requests at the nodes which online server hasn't served. The cost of the offline server is equal to $c_{offline} = 2y + 2y_1 + x + 2L$. For the online server's movements, we can see Fig. 6.

As shown in Fig. 6, the released requests between node $-x$ and node $-x+y$ for the online server satisfy *Released request sequence 1*, the cost for each request is equal to $2(L-x)B$; the released requests between node $-x+y$ and node L for the online server also satisfy *Released request sequence 1*, the cost for each request is equal to $2(L-x-y-y_1)B$. According to OM algorithm, the online server would go straightly to node $-L$, turn around to node L, then go back to the origin. The cost of the online server is equal to $c_{online} = 4L - x + (2(L-x)(L-x+y) + 2(L-x-y)(y_1+1) + 2(L-x-y-y_1)(L+2x+y_1) - ({y_1}^2 + y_1))B$.

When $y = 1, y_1 = 0$, $\frac{c_{online}}{c_{offline}}$ can get the maximum value, $\rho_0 - \frac{c_{online}}{c_{offline}} > 0$(See "Appendix" C for details).

Case 3. The Offline server travels a distance of y, then turns around, and two servers meet.

After traveling a distance of $y(1 \le y \le L+x-1)$, the offline server turns around and meets with the online server at node $-L+y$. Based on the values of $-L+y$, we consider three cases:

Case 3.1. The online server and the offline server meets on the left side of node $-x$(including node $-x$).

If the online server and the offline server meets on the left side of the node $-x$, the value of y must satisfy $y \le L-x$.

Suppose the offline server would move a distance of $z(1 \le z \le y)$ after the meeting, then turn around back to the origin. According to OM algorithm, online server would go towards node $-L$, then turn around. If two servers would meet for the second time, the position of the meeting is node $y-x-z$. There is $y-x-z \ge -L+y$, this means that the position of the meeting is on the right side of node $-L+y$. Based on the different values of $y-x-z$, we consider two cases:

Case 3.1.1. The online server and the offline server meet at negative half line.

If the online server and the offline server meet at negative half line, the value of y satisfy $y < x+z$.

The meeting is *Meeting situation 2*(when *Meeting situation 2* occurs again, we don't consider that the offline server would turn around again until the offline server arrives at the origin or the extreme node). The offline server would go straightly back to the origin after the meeting, the cost of the offline server is equal to $c_{offline} = 2L + 2z - x$. According to OM algorithm, we can get the trajectory of the online server and the offline server(see Fig. 7).

As shown in Fig. 7, the released requests between node $-x+1$ and node $-x+y$, the released requests between node $-L+y+1$ and node $-L+y-z$, and the released requests between node $y-x-z+1$ and the origin for online server all satisfy *Released request sequence 1*, and the cost for each request is equal to $2(L-x)B$. According to OM algorithm, the online server would go straightly to node $-L$ at time $t+2L$, turn around to node $-x+y$, and turn around to node $-L+y-z$, then go back to the origin. The cost of the online server is equal to $c_{online} = 4L - 3x + 2z + (4(L-x)z + (L-x-y)^2 + (L-x-y) + (L-x-z)^2 + (L-x-z) + 2L^2 - 2xL)B$.

Let $f(y,z) = \rho - \frac{c_{online}}{c_{offline}}$, $f(y,z)$ is an increasing function of y and z. We can get $f(y,z) > 0$(See "Appendix" D for details).

Case 3.1.2. The online server and the offline server don't meet at negative half line.

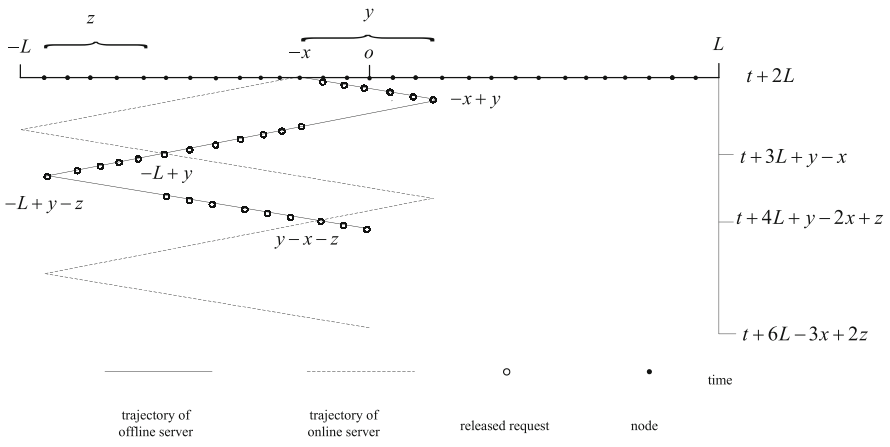

Fig. 7 Trajectory of online and offline server

If the online server and the offline server don't meet at negative half line, the value of y must satisfy $y \geq x + z$.

The online server and the offline server wouldn't meet at the positive half line, because the offline server has stayed at the origin. According to Lemma 1 and OM algorithm, the cost of the offline server is equal to $c_{offline} = 2L + 2z - x$, the cost of the online server is equal to $c_{online} = 4L - 3x + 2z + (2(L - x)(L - x + y + z) + (L - x - y)^2 + (L - x - y) + (L - y)^2 + (L - y))B$. Similar to the analysis in Case 3.1.1, we can get $\rho > \frac{c_{online}}{c_{offline}}$.

Case 3.2 The online server and the offline server meet between the node $-x$ and the origin(including the origin).

If the online server and the offline server meet between the node $-x$ and the origin(including the origin), the value of y must satisfy $L - x < y \leq L$.

Suppose the offline server would move a distance of $z(1 \leq z \leq y)$ after the meeting, then turn around back to the origin. The analysis of this case is similar to the Case 3.1, no matter what values y and z are, we can get $\rho > \frac{c_{online}}{c_{offline}}$.

Case 3.3 The online server and the offline server meet at the positive half line.

If the online server and the offline server meet at the positive half line, the value of y must satisfy $y > L$.

After the meeting, the offline server would go back to the origin. For the online server, according to OM algorithm and Lemma 1, the online server would go straightly to node $-L$, turn around, move straightly to node $-x+y$, then go back to the origin(See Fig. 8).

The cost of the offline server is equal to $c_{offline} = 2y - x$, the cost of the online server is equal to $c_{online} = 2L + 2y - 3x + 2(L - x)(2y - L)B$.

Let $f\ (y) = \frac{c_{online}}{c_{offline}}$, $f\ (y)$ is an increasing function of y. When $y = L+x-1$, $f\ (y)$ can get the maximum value, $\rho_0 - f\ (L + x - 1) = f\ (L + x) - f\ (L + x - 1) > 0$.

In conclusion of the above cases, we can prove Lemma 2.

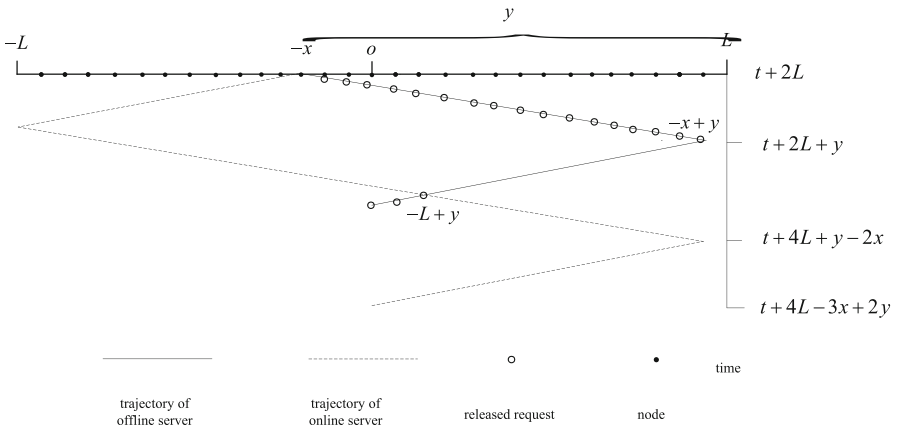

Fig. 8 Trajectory of online and offline server

Theorem 4 *The competitive ratio for Observe and Move Algorithm on the truncated line segment is* $1 + LB$.

Proof The adversary always wants to separate the online server and the offline server. According to OM algorithm, there are two cases to separate the online server and the offline server: 1. At time 0, the adversary releases a request at node x. 2. At time 0, the adversary doesn't release any request, offline server moves towards extreme node(L or $-L$). We show that no matter how the adversary releases the requests, *Meeting situation 2* must occur.

Case 1. Without loss of generality, suppose $x > 0$, according to OM algorithm, the online server serves the request immediately. For the offline server, when the online server leaves the origin, staying at the origin seems meaningless(Because this only leads to a higher cost of the offline server, and a lower cost of the online server).

Case 1.1. The offline server moves towards node $-L$.

After traveling a distance of $y(y > 0)$, the offline server turns around. According to Lemma 1, the adversary would release requests at positions of the offline server. For the online server, these y requests satisfy *Released request sequence 1*. According to OM algorithm, the online server would serve the request on node x firstly, then turn around to serve these y requests. If $y \geq x$, two servers would meet at negative half line or at the origin again, and this meeting is *Meeting situation 2*; If $y < x$, two servers would meet at node $x - y > 0$, the offline server must serve the request at node x. Suppose the offline server travels a distance of $z(z \geq 0)$ after serving the request on node x, then turns around to the origin. Two servers would meet at $0.5z$ again, this meeting is *Meeting situation 2*.

Case 1.2. The offline server moves towards node L.

The offline server would stay at the origin first, after the leaving of the online server, the offline server starts to move. When the adversary begins to release requests, according to OM algorithm, the first released request for the online server satisfies *Released request sequence 1*, online server must serve the request at node x then makes the decision. At the same time, the offline server would go on keeping the direction

to extreme node L, or turn around. If the meeting occurs at negative half line, the meeting is *Meeting situation 2*; if the meeting occurs at positive half line, the meeting is *Meeting situation 1*, the following analysis is similar to Case 1.1, the next meeting must be *meeting situation 2*.

Case 2. Without loss of generality, at time 0, suppose the offline server moves towards node L, the online server stays at the origin. When the first request is released, according to Lemma 1 and OM algorithm, the online server would move to serve the request immediately and the offline server is on the right side of the online server. No matter how the adversary releases requests, two servers must meet at positive half line, and the meeting is *meeting situation 2*.

According to Lemma 2, after *Meeting situation 2* occurs, the offline server must go straightly to the origin and idle or keep his direction to the extreme node. Suppose the position of the meeting on the positive half line is node x. Now we consider the situation (the offline server arrives at L and $-L$ successively, then goes back to the origin) as a cycle, for each cycle, the cost of the offline server would increase by $4L$, and the cost of the online server would increase by $4L + 4(L^2 - x^2)B$. When $x = 0$, the cost of the online server would get maximum value.

Through above analysis, two servers meet at the origin is the worst case. We can describe the worst case as follows: At time 0, the adversary releases a request at node L, the online server must move towards to node L to serve the request, the offline server moves straightly to node $-L$. The adversary releases the requests at the positions of the offline server, this will last $n + 1$ cycles. The cost of the offline server is equal to $4L + 3LB + 4nL$, the cost of the online server is equal to $6L + LB + 2L^2B + 4nL + 4nL^2B$. The competitive ratio is $\frac{c_{online}}{c_{offline}} = \frac{6L+LB+2L^2B+4nL+4nL^2B}{4L+3LB+4nL}$, when $n \to \infty$, $\frac{c_{online}}{c_{offline}} = 1 + LB$.

We conclude that the competitive ratio of OM algorithm is $1 + LB$.

5 Uniform metric space

5.1 Lower bound on uniform metric space

In this section, we discuss the online TSP with per-unit-time cost on the uniform metric space, the uniform metric space is induced by a complete graph with unit edge weights. Besides the origin, there are n vertices in the metric space, and the request can only be released at the vertex.

Theorem 5 *No deterministic online algorithm can achieve a competitive ratio less than* $1 + B$.

Proof The offline server would visit the n vertices one by one, then return to the origin, the adversary would release the requests at the offline server's positions. The cost of the offline server is equal to $c_{offline} = n + 1$. We consider the different positions of the online server at time 1.

Case 1. The online server stays at the origin.

For the online server, when a new request is released, there are three options: 1. Serve the request immediately. 2. Go to another node. 3. Stay. The options of "stay" would lead to a higher cost for the online server. We only consider the two options:

1. Serve the request immediately. The cost of each request for the online server is equal to B. After serving all the n requests, the cost of the online server is equal to $c_{online} = n + 2 + nB$.
2. Go to another vertex. The worst case for the online server is that after visiting $n-1$ nodes, the online server has not served any request at all. The adversary would release the last request at time n. The penalties of n requests for the online server are larger than nB, the cost of the online server is equal to $c_{online} > 2n - 1 + nB$.

If the online server adopts the mixed strategy, i.e, for some requests, the online server would serve it immediately, for other requests, the online server would go to another vertex, the value of the cost in this case must larger than $n + 2 + nB$.

The cost ratio is equal to $\frac{c_{online}}{c_{offline}} = \frac{n+2+nB}{n+1}$, when $n \to \infty$, $\frac{c_{online}}{c_{offline}} = 1 + B$.

Case 2. The online server stays at other vertex.

Consider the symmetry of the network, this case is the same with case 1, the cost ratio is equal to $\frac{c_{online}}{c_{offline}} = 1 + B$.

Case 3. The online server stays between the origin and vertex j.

Vertex j would be any vertex in the uniform metric space, suppose at time 1, the distance between the online server and the origin(or vertex j) is $x(0 < x \leq 0.5)$. The online server still has two options as in case 1 when a new request is released. Similar to the analysis in case 1, the minimum value of the online server's cost is equal to $c_{online} = n + 2 + x + n(1 + x)B$. The cost ratio is equal to $\frac{c_{online}}{c_{offline}} = 1 + (1 + x)B$ when $n \to \infty$.

In conclusion of above cases, we can get the lower bound for online TSP with per-unit-time cost on the uniform metric space as $1 + B$.

5.2 *The greedy algorithm*

In this section, we give a greedy algorithm and analyze the competitiveness on the uniform metric space.

Greedy algorithm(GA): At any time t, define $S(t)$ as a set for all the released requests but not yet served by online server. The greedy server will always choose to serve the requests in $S(t)$ for the least cost. If set $S(t)$ is empty, the greedy server will go back to the origin.

Lemma 3 *The adversary only releases the requests at the positions of the offline server.*

Proof When the adversary only releases the requests at the positions of the offline server, the cost of the offline server is equal to y, the cost of the online server is equal to z. Suppose the adversary only releases one request R_j at node j before the offline server arrives at node j, the distance from the online server to node j at time r_j is $x(0 < x \leq 1)$. The cost of the offline server is equal to $c_{offline} \geq y + xB$. For the online server, at time r_j, there are two unserved requests. According to *GA*, there are two cases for the online server:

Case 1. The online server serves the request R_j later, the cost of the online server is equal to $z_1 = z + xB$.

Case 2. The online server serves the request R_j firstly, the cost of the online server is equal to z_2.

if $z_2 > z_1$, the cost ratio is equal to $\frac{z+xB}{y+xB} < \frac{z}{y}$; if $z_2 < z_1$, the cost ratio is equal to $\frac{z_2}{y+xB} < \frac{z+xB}{y+xB} < \frac{z}{y}$.

Theorem 6 *The competitive ratio of the Greedy algorithm is* $1 + B$.

Proof According to *GA*, the online server would serve the request immediately when a new request is released. According to Lemma 3, the adversary only releases the requests at the positions of the offline server. For the online server, there is only one unserved request at the same time. For each request, the cost is equal to B. If the adversary does'nt release request when the offline server arrives at the vertex which would decrease the online server's cost(the offline server's cost is the same). After serving all the n requests, the cost of the offline server is equal to $c_{offline} = n + 1$, the cost of the online server is equal to $c_{online} = n + 2 + nB$, when $n \to \infty$, the competitive ratio is $\frac{c_{online}}{c_{offline}} = 1 + B$.

6 Conclusion

In this paper, we consider the affected population's dissatisfaction in the emergency vehicle routing problem, introduce per-unit-time cost into the OL TSP. First, we find out that no deterministic and randomized online algorithms can achieve constant competitive ratio for OL-TSP with per-unit-time cost on general metric space even on the positive half line. Therefore, we propose the problem on more restricted metric spaces, such as the truncated line segment and the uniform metric space. While on truncated line segment, we give a lower bound, an *Observe and Move Algorithm* and the algorithm's competitive ratio. The algorithm is to observe the sequence of the released requests, accordingly the online server decides how to move. There are two interesting practical applications that can be solved by the proposed algorithm: (1)Emergency vehicle routing problem. After the natural disaster (such as earthquake, hurricane), the natural disaster would destruct the roads of the city, there would be only several main roads which can pass through in the city (especially the structure of the city is grid network). For each main road, we can see it as a truncated line segment; (2) The order picking problem. Imagine in a big ware house, under the assembly-line order picking system, every picker is responsible for one cross aisle which can be seen as a truncated line segment. The request in our model can be seen as the storage which would be picked by the picker, the objective function can measure the efficiency of the picker. While on uniform metric space, we give a lower bound and a *Greedy Algorithm* with its competitive ratio, and prove it optimal. Besides, introducing other factors(such as deadline) into this problem is also interesting.

Acknowledgements The authors would like to thank the anonymous referee for the constructive comments. This work is funded by Chongqing Social Science Planning Project(2016BS085),NSF of China (No. 71702016) and Humanities and Social Science Research Project of Chongqing Education Commission (No.17SKJ034).

Appendix A. Case 1.2.1 in Theorem 3

The added cost of the online server includes four parts:

1. The added penalties of the requests between node $-y-1$ and node $-L$. For each of $L-y$ requests, the increased cost is equal to $2y_1B$. The increased costs of the $L-y$ requests are equal to $2(L-y)y_1B$;
2. The costs of the requests between node 1 and node y. y requests are released during the time interval $[2L+1, 2L+y]$ one by one, the requests are served during the time interval $[4L+2y_1+1, 4L+2y_1+y]$ by online server. The cost is equal to $(2L+2y_1)B$ for each request. The costs of the y requests are equal to $2(L+y_1)yB$;
3. The costs of the requests between node $y+1$ and node L. $L-y$ requests are released during the time interval $[2L+2y_1+y+1, 3L+2y_1]$ one by one, the $L-y$ requests are served during the time interval $[4L+2y_1+y+1, 5L+2y_1]$ by online server. The cost is equal to $2LB$ for each request. The costs of the $L-y$ requests is equal to $2(L-y)LB$;
4. The added traveling time of the online server. The increased traveling time of the online server is equal to $2L+2y_1$.

The total added cost of the online server is equal to $2L+2y_1+2(L-y)y_1B+2(L+y_1)yB+2(L-y)LB = 2L+2y_1+(2L^2+2Ly_1)B$.

Appendix B. Case 1.2.2 in Theorem 3

When $y_2 \le y$, the added cost of the online server includes seven parts:

1. The added costs of the requests between node $-y-1$ and node $-L$. For each of $L-y$ requests, the increased cost is equal to $2(y+y_2)B$. The increased costs of the $L-y$ requests are equal to $2(L-y)(y+y_2)B$;
2. The costs of the requests between node 1 and node y_2. y_2 requests are released during the time interval $[2L+1, 2L+y_2]$ one by one, the y_2 requests are served during the time interval $[2L+2y+1, 2L+2y+y_2]$ by online server. The cost is equal to $2yB$ for each request. The costs of the y_2 requests are equal to $2yy_2B$;
3. The costs of the requests between node y_2+1 and node y. $y-y_2$ requests are released during the time interval $[2L+2y_2+1, 2L+y]$ one by one, the $y-y_2$ requests are served during the time interval $[4L+3y_2+2y+1, 4L+2y_2+3y]$ by online server. The cost is equal to $2(L+y+y_2)B$ for each request. The costs of the $y-y_2$ requests are equal to $2(L+y+y_2)(y-y_2)B$;
4. The costs of the requests between node -1 and node $-y_2$. y_2 requests are released during the time interval $[2L+2y+1, 2L+2y+y_2]$ one by one, the y_2 requests

are served during the time interval $[2L + 2y + 2y_2 + 1, 2L + 2y + 3y_2]$ by online server. The cost is equal to $2y_2B$ for each request. The costs of the y_2 requests are equal to $2y_2^2B$;

5. The costs of the requests between node 1 and node y_2. y_2 requests are released during the time interval $[2L + 2y + 2y_2 + 1, 2L + 2y + 3y_2]$ one by one, the requests are served during the time interval $[4L + 2y + 2y_2 + 1, 4L + 2y + 3y_2]$ by online server. The cost is equal to $2LB$ for each request. The costs of the y_2 requests are equal to $2y_2LB$;
6. The costs of the requests between node $y+1$ and node L. $L-y$ requests are released during the time interval $[2L+3y+2y_2+1, 3L+2y+2y_2]$ one by one, the $L-y$ requests are served during the time interval $[4L + 3y + 2y_2 + 1, 5L + 2y + 2y_2]$ by online server. The cost is equal to $2LB$ for each request. The costs of the $L-y$ requests are equal to $2(L-y)LB$;
7. The added traveling time of the online server. The increased traveling time of the online server is equal to $2L(L + y + y_2)$.

The total added cost of the online server is equal to $2L + 2y + 2y_2 + (2(L-y)(y + y_2)B + 2yy_2B + 2(L + y + y_2)(y - y_2)B + 2y_2{}^2B + 2y_2LB + 2(L-y)LB) = 2(L + y + y_2) + 2(L + y + y_2)LB$.

When $y_2 > y$, the added cost of the online server includes five parts:

1. The added costs of the requests between node $-y-1$ and node $-L$. For each of $L-y$ requests, the increased cost is equal to $2(y + y_2)B$. The increased costs of the $L-y$ requests are equal to $2(L-y)(y + y_2)B$;
2. The costs of the requests between node 1 and node y. y requests are released during the time interval $[2L + 1, 2L + y]$ one by one, the y_2 requests are served during the time interval $[2L + 2y + 1, 2L + 3y]$ by online server. The cost is equal to $2yB$ for each request. The costs of the y requests are equal to $2y^2B$;
3. The costs of the requests between node -1 and node $-y$. y requests are released during the time interval $[2L + 2y + 1, 2L + 3y]$ one by one, the y requests are served during the time interval $[2L + 2y + 2y_2 + 1, 2L + 3y + 2y_2]$ by online server. The cost is equal to $2y_2B$ for each request. The costs of the y requests are equal to $2yy_2B$;
4. The costs of the requests between node 1 and node L. L requests are released during the time interval $[2L + 2y + 2y_2 + 1, 3L + 2y + 2y_2]$ one by one, the L requests are served during the time interval $[4L + 2y + 2y_2 + 1, 5L + 2y + 2y_2]$ by online server. The cost is equal to $2LB$ for each request. The costs of the L requests are equal to $2L^2B$;
5. The added traveling time of the online server. The increased traveling time of the online server is equal to $2L(L + y + y_2)$.

The total added cost of the online server is equal to $2L + 2y + 2y_2 + (2(L-y)(y + y_2)B + 2y^2B + 2yy_2B + 2L^2B) = 2(L + y + y_2) + 2(L + y + y_2)LB$.

Appendix C. Case 2 in Lemma 2

The cost of the online server after time $t + 2L$ includes six parts:

1. The costs of the requests between node $-x-1$ and node $-L$. $L-x$ requests are released during the time interval $[t+2x+1, t+x+L]$ one by one, the $L-x$ requests are served during the time interval $[t+2L+1, t-x+3L]$ by online server. The cost is equal to $2(L-x)B$ for each request. The costs of the $L-x$ requests are equal to $2(L-x)^2B$;
2. The costs of the requests between node $-x+1$ and node $-x+y$. y requests are released during the time interval $[t+2L+1, t+2L+y]$ one by one, the y requests are served during the time interval $[t+4L-2x+1, t+4L-2x+y]$ by online server. The cost is equal to $2(L-x)B$ for each request. The costs of the y requests are equal to $2(L-x)yB$;
3. The costs of the requests between node $-x$ and node $-x-y_1$. y_1+1 requests are released during the time interval $[t+2L+2y, t+2L+2y+y_1]$ one by one, the y_1+1 requests are served during the time interval $[t+4L-2x-y_1, t+4L-2x]$ by online server. The costs of these requests are equal to $2(L-x-y)B$, $2(L-x-y-1)B, \cdots\cdots 2(L-x-y-y_1)B$ respectively which forms an arithmetic sequence. The costs of the y_1+1 requests are equal to $(2(L-x-y)(y_1+1)-({y_1}^2+y_1))B$;
4. The costs of the requests between node $-x+y+1$ and node L. $L+x-y$ requests are released during the time interval $[t+2L+3y+2y_1+1, t+3L+2y+2y_1+x]$ one by one, the $L+x-y$ requests are served during the time interval $[t+4L+y-2x+1, t+5L-x]$ by online server. The cost is equal to $2(L-y-y_1-x)B$ for each request. The costs of the $L+x-y$ requests are equal to $2(L-y-y_1-x)(L+x-y)B$;
5. The costs of the requests between node $x+y+y_1-1$ and the origin. $x+y+y_1$ requests are released during the time interval $[t+4L+y+y_1+1, t+4L+2y+2y_1+x]$ one by one, the $x+y+y_1$ requests are served during the time interval $[t+6L-y-y_1-2x+1, t+6L-x]$ by online server. The cost is equal to $2(L-y-y_1-x)B$ for each request. The costs of the $x+y+y_1$ requests are equal to $2(L-y-y_1-x)(x+y+y_1)B$;
6. The traveling time of the online server. The traveling time of the online server is equal to $4L-x$.

The total cost of the online server is equal to $c_{online} = 4L-x+(2(L-x)(L-x+y)+2(L-x-y)(y_1+1)-({y_1}^2+y_1)+2(L-y-y_1-x)(L+2x+y_1))B$.

As shown in Fig. 6, the cost of the offline server is equal to $c_{offline} = 2y+2y_1+x+2L$.

The cost ratio is equal to $f(y_1) = \frac{c_{online}}{c_{offline}}$,

$$f'(y_1) = \frac{-2c_{online} + (2L-8x-4y-6y_1-1)Bc_{offline}}{{c_{offline}}^2}$$

.

Let $g(y_1) = -2c_{online} + (2L-8x-4y-6y_1-1)Bc_{offline}$, $g'(y_1) = -6Bc_{offline} < 0$.

Because $0 \le y_1 \le L-x-y-1$, when $y_1 = 0$, $g(y_1)$ can get the maximum value.

Let $h(x) = g(0) = -2(4L-x+2((L-x)(L-x+y)+(L-x-y)(L+2x+1))B)+(2y+x+2L)(2L-8x-4y-1)B$

$h'(x) = (-16L - 20x - 26y - 3)B + 2 < 0$, when $x = 0$, $h(x)$ can get the maximum value.

$h(0) = -8L + (-4L^2 - 4yL - 8y^2 - 6L + 2y)B < 0$, $g(0) \leq h(0) < 0$, $f'(y_1) < 0$, $f(y_1)$ is a decreasing function of y_1. When $y_1 = 0$, $\frac{c_{online}}{c_{offline}}$ can get the maximum value. Similar to the above analysis, when $y = 1$, $\frac{c_{online}}{c_{offline}}$ can get the maximum value.

When $y_1 = 0$, $y = 1$, $\frac{c_{online}}{c_{offline}} = \frac{4L-x+(2(L-x)(L+2x)+(L-7x-2))B}{2L+x+2}$

$$\rho_0 - \frac{c_{online}}{c_{offline}} = \frac{2(4L-x) + 8(3x+1)(2L+x)B}{(2L+x+2)(2L+x)} > 0$$

Appendix D. Case 3.1.1 in Lemma 2

The cost of the online server after time $t + 2L$ includes seven parts:

1. The costs of the requests between node $-x-1$ and node $-L$. $L-x$ requests are released during the time interval $[t+2x+1, t+x+L]$ one by one, the $L-x$ requests are served during the time interval $[t+2L+1, t-x+3L]$ by online server. The cost is equal to $2(L-x)B$ for each request. The costs of the $L-x$ requests are equal to $2(L-x)^2B$;
2. The costs of the requests between node $-x+1$ and node $-x+y$. y requests are released during the time interval $[t+2L+1, t+2L+y]$ one by one, the y requests are served during the time interval $[t+4L-2x+1, t+4L-2x+y]$ by online server. The cost is equal to $2(L-x)B$ for each request. The costs of the y requests are equal to $2(L-x)yB$;
3. The costs of the requests between node $-x$ and node $-L+y+1$. $L-x-y$ requests are released during the time interval $[t+2L+2y, t+3L+y-x-1]$ one by one, the $L-x-y$ requests are served during the time interval $[t+3L+y-x+1, t+4L-2x]$ by online server. The costs of these requests are equal to $2(L-x-y)B, 2(L-x-y-1)B, \cdots\cdots 2B$ respectively which forms an arithmetic sequence. The costs of the $L-x-y$ requests are equal to $((L-x-y)^2 + (L-x-y))B$;
4. The costs of the requests between node $-L+y-1$ and node $-L+y-z$. z requests are released during the time interval $[t+3L+y-x+1, t+3L+y-x+z]$ one by one, the z requests are served during the time interval $[t+5L+y-3x+1, t+5L+y-3x+z]$ by online server. The cost is equal to $2(L-x)B$ for each request. The costs of the z requests are equal to $2(L-x)zB$;
5. The costs of the requests between node $-L+y$ and node $y-x-z-1$. $L-x-z$ requests are released during the time interval $[t+3L+y-x+2z, t+4L+y-2x+z-1]$ one by one, the $L-x-z$ requests are served during the time interval $[t+4L+y-2x+z+1, t+5L-3x+y]$ by online server. The costs of these requests are equal to $2(L-x-z)B, 2(L-x-z-1)B, \cdots\cdots 2B$ respectively which forms an arithmetic sequence. The costs of the $L-x-z$ requests are equal to $((L-x-z)^2 + (L-x-z))B$;

6. The costs of the requests between node $y-x-z+1$ and the origin. $x+z-y$ requests are released during the time interval $[t+4L-2x+y+z+1, t+4L-x+2z]$ one by one, the $x+z-y$ requests are served during the time interval $[t+6L-4x+z+y+1, t+6L-3x+2z]$ by online server. The cost is equal to $2(L-x)B$ for each request. The costs of the $x+z-y$ requests are equal to $2(L-x)(x+z-y)B$;
7. The traveling time of the online server. The traveling time of the online server is equal to $4L-3x+2z$.

The total cost of the online server is equal to $c_{online} = 4L+3x-2z+(2(L-x)(2z+L)+(L-x-y)^2+(L-x-y)+(L-x-z)^2+(L-x-z))B$.

As shown in Fig. 7, the cost of the offline server is equal to $c_{offline} = 2L+2z-x$.

Let $f(y,z) = \rho - \frac{c_{online}}{c_{offline}} = \frac{A}{xc_{offline}}$, $f'_z(y,z) = \frac{-2xA+A'xc_{offline}}{xc_{offline}}$.

Let $g(z) = -2xA + A'xc_{offline}$, $g'(z) = -2xBc_{offline} < 0$.

Because $z \le y$, when $z = y$, $g(z)$ can get the minimum value.

$g'(y) = 2x(2L+2y-x)B > 0$, when $y = 1$, $g(y)$ can get the minimum value.

$ming(y) = g(1) > g(0) = x^2(4L^2+2x^2-6xL+6L-5x)B > 0$, we can get $g(z) > 0$.

$f(y,z)$ is an increasing function of z, when $z = 1$, $f(y,z)$ can get the minimum value. For $f(y,1)$, with the similar analysis, we can get $f(y,1)$ is an increasing function of y, and

$minf(y,z) = f(1,1) > f(0,0) = \frac{(4(L-x)^2L-2x((L-x)^2+(L-x))B+4(L-x)^2}{x(2L-x)} > 0$.

References

Allulli L, Ausiello G, Bonifaci V, Laura L (2008) On the power of lookahead in on-line server routing problems. Theor Comput Sci 408(2–3):116–128

Ausiello G, Bonifaci V, Laura L (2008) The online prize-collecting traveling salesman problem. Inf Process Lett 107(6):199–204

Blom M, Krumke SO, De Paepe WE, Stougie L (2001) The online TSP against fair adversaries. INFORMS J Comput 13(2):138–148

Campbell AM, Thomas BW (2008) Probabilistic traveling salesman problem with deadlines. Transp Sci 42(1):1–21

Charnsirisakskul K, Griffin PM, Keskinocak P (2004) Order selection and scheduling with leadtime flexibility. IEE Trans 36:697C707

FedEx, (2005) Rules/accessorial tariff via all motor routes naming Rules, Regulations and claims procedures applying on surface expedited services between points in North America (Except Mexico). http://customcritical.fedex.com/us/serviceinfo/documents/pdf/tarifffdcc101g.pdf?link=4, accessed on April 11

Gutiérrez S, Krumke S, Megow N, Vredeveld T (2006) How to whack moles. Theor Comput Sci 361(2–3):329–341

Irani S, Lu X, Regan A (2004) On-line algorithms for the dynamic traveling repair problem. J. Sched 7(3):243–258

Jaillet P, Lu X (2011) Online traveling salesman problems with service flexibility. Networks 58(2):137–146

Krumke SO, Paepe WD, Poensgen D, Stougie L (2003) News from the online traveling repairman. Theor Comput Sci 295:279–294

Lipmann M (2003) On-line routing. PhD Thesis, Technische Universiteit Eindhoven

Simroth A, Souza A (2009) On an online traveling repairman problem with flowtimes: worst-case and average-case analysis. COCOON, pp 168-177

Slotnick SA, Sobel MJ (2005) Manufacturing lead-time rules: customer retention versus tardiness costs. Eur J Oper Res 163:825–856

Wen X, Xu Y, Zhang H (2012) Online traveling salesman problem with deadline and advanced information. Comput Ind Eng 63(4):1048–1053

Wen X, Xu Y, Zhang H (2015) Online traveling salesman problem with deadlines and service flexibility. J Comb Optim 30(3):545–562

Yu W, Liu Z et al (2014) Optimal deterministic algorithms for some variants of online quota traveling salesman problem. Eur J Oper Res 238(3):735–740

Zhou Y, Liu J, Zhang Y, Gan X (2017) A multi-objective evolutionary algorithm for multi-period dynamic emergency resource scheduling problems. Transp Res E-Log 99(3):77–95