Dynamic Traveling Repair Problem with an Arbitrary Time Window

Yossi Azar(✉) and Adi Vardi(✉)

School of Computer Science, Tel-Aviv University, 69978 Tel-Aviv, Israel
azar@tau.ac.il, adi.vardi@gmail.com

Abstract. We consider the online Dynamic Traveling Repair Problem (DTRP) with an arbitrary size time window. In this problem we receive a sequence of requests for service at nodes in a metric space and a time window for each request. The goal is to maximize the number of requests served during their time window. The time to traverse between two points is equal to the distance. Serving a request requires unit time. Irani et al., SODA 2002 considered the special case of a fixed size time window. In contrast, we consider the general case of an arbitrary size time window. We characterize the competitive ratio for each metric space separately. The competitive ratio depends on the relation between the minimum laxity (the minimum length of a time window) and the diameter of the metric space. Specifically, there exists a constant competitive algorithm only when the laxity is larger than the diameter. In addition, we characterize the rate of convergence of the competitive ratio, which approaches 1, as the laxity increases. Specifically, we provide matching lower and upper bounds. These bounds depend on the ratio between the laxity and the optimal TSP solution of the metric space (the minimum distance to traverse all nodes). An application of our result improves the previously known lower bound for colored packets with transition costs and matches the known upper bound. In proving our lower bounds we use an embedding with some special properties.

1 Introduction

Consider an employee in the Google IT division. He is responsible for replacing malfunctioning disks in Google's huge computer farms. During his shift he receives requests to replace disks at some points in time. Each request is associated with a deadline. If the disk will not be replaced before the deadline, there is a high probability that the performance of the Search Engine will experience a significant hit. Replacing a disk takes unit time (service time). However, before the employee can replace it, he must travel from his current location to the location of the disk. The goal is to maximize the number of disks replaced before their deadline. What path should the employee take and how should the path change with new requests? Irani et al., SODA 2002 [15,18] called this online

Supported in part by the Israel Science Foundation (grant No. 1506/16), by the Israeli Centers of Research Excellence (I-CORE) program, (Center No. 4/11) and by the Blavatnik Fund.

K. Jansen and M. Mastrolilli (Eds.): WAOA 2016, LNCS 10138, pp. 14–26, 2017.
DOI: 10.1007/978-3-319-51741-4_2

problem the Dynamic Traveling Repair Problem (DTRP). They considered the special case of a fixed size time window, where the window of a request is the period between its release time and its deadline. In contrast, we consider the general case of an arbitrary size time window. In this paper we characterize the competitive ratio for each metric space separately. We determine whether the competitive ratio is constant or not depending on the minimum laxity (the minimum length of a time window) and the diameter of the metric space (the maximum distance between nodes in the metric space). In addition, we consider the case where the laxity is large compared to the optimal TSP solution of the metric space (the minimum distance to traverse all nodes). Specifically, we provide matching lower and upper bounds for these cases. These bounds depend on the ratio between the laxity and the optimal TSP solution of the metric space.

We note that even when the service time is not negligible, our problem can be reduced to TSP with time windows and zero service time [5] by changing the metric space. However, our competitive ratio depends on the properties of the metric space and the reduction might change the parameters of the metric space significantly. Hence, it might influence a crucial parameter which determines the competitive ratio. Therefore, we take service time into account in our model. Moreover, in our main result, where the laxity is larger than the optimal TSP solution of the metric space, without service time it is easy to design a 1-competitive algorithm by traveling over an optimal TSP solution periodically.

Offline Problem. Note that in the offline case (i.e., when the sequence is known in advance), if the service time is negligible compared to the minimum positive distance between nodes (or 0) then the problem becomes TSP (or vehicle routing) with time windows and zero service time [5]. Moreover, if in addition all deadlines are the same and all release times are zero then the problem reduces to the (offline) orienteering problem [1,3,14]. Vehicle routing problem (with time windows and zero service time) has been extensively studied both in computer science and the operations research literature, see [11,12,19–22]. For an arbitrary metric space Bansal et al. [5] showed an $O(\log^2 n)$-approximation (for certain cases a better approximation can be achieved [8]). Constant factor approximations have been presented for the case of points on a line [6,17,23]. For the orienteering problem, i.e., all release times are zero, all deadlines are the same, and the service time is zero, there are constant factor approximation algorithms [5,7,9,10]. A restricted online version of the Vehicle Routing problem (without deadlines) was considered in [2,13,16].

Application for Packet Scheduling. Another motivation for our problem is the Colored Packets with Deadlines and Metric Space Transition Cost problem. In this setting we are given a sequence of incoming colored packets. Each colored packet is of unit size and has a deadline. There is a reconfiguration cost (setup cost) to switch between colors (the cost depends on the colors). The goal is to find a schedule that maximizes the number of packets that are transmitted before the deadline. Note that for one color the earliest deadline first (EDF) strategy is known to achieve an optimal throughput. The unit cost color has been considered in [4]. In particular, when we apply our results to the uniform metric space we improve the previous lower bound and match the known upper bound.

1.1 Our Results

Denote by σ the sequence of requests. The window of request i is $[r_i, d_i]$, where r_i is the release time of the request and d_i is the deadline of the request. Let $L = \min_{i \in \sigma}\{d_i - r_i\} \geq 1$ be the minimum laxity of the requests (the minimum length of a time window). Note that the laxity has to be at least 1 since the service time equals 1. Denote by $\Delta(G)$ the diameter of the metric space G, i.e., the largest distance between two nodes. Denote by $TSP(G)$ the weight of a minimal TSP solution in the metric space G and $MST(G)$ the weight of a minimal spanning tree.

In this paper we characterize when it is possible to achieve a $\Theta(1)$ competitive algorithm for the Dynamic Traveling Repair Problem with an arbitrary time window, and when the best competitive algorithm is unbounded. Moreover, we characterize the rate of convergence of the competitive ratio, which approaches 1 as the laxity increases. Specifically, we provide matching lower and upper bounds depending on the ratio between the laxity and the optimal TSP solution of the metric space.

It is also interesting to mention that in many cases the competitive ratio of an algorithm is computed as the supremum over all metric spaces while lower bounds are proved for one specific metric space. In contrast, we prove more refined results. Specifically, we show an upper bound and a lower bound for each metric space separately. Hence, one cannot design a better competitive algorithm for the specific metric space that one encounters in the real specific instance. Hence, even for specific metric spaces, we show it is impossible to do better.

We consider three cases. The last two cases are done for completeness of the result while the first case is our main result.

- **Case A:** $L > TSP(G)$. Let $\delta = TSP(G)/L < 1$. We show a strictly larger than 1 lower bound. Specifically, if $\delta \leq \frac{1}{256}$ we provide a lower bound of $1 + \Omega\left(\sqrt{\delta}\right)$ as well as a matching upper bound of $1 + O\left(\sqrt{\delta}\right)$.
  We note that without service time it is easy to design 1-competitive algorithm by traveling over an optimal TSP solution periodically. Recall that there is a reduction from the service time model to a model without service time that seems to contradict the lower bound (see [5]). However, the reduction modifies the metric space and hence increases δ such that δ is not smaller than $\frac{1}{256}$.
- **Case B:** $3\Delta(G) < L \leq TSP(G)$. We design a $O(1)$-competitive algorithm and a 1.00054 lower bound.
- **Case C:** $L < \Delta(G)/2$. For any metric space the competitive ratio of any deterministic online algorithm is unbounded (easily proved). For randomized algorithms the competitive ratio depends on the metric space. For example, for a metric space which consists of 2 points one can easily show a 4-competitive algorithm even for $L = 0$. In contrast, in a uniform metric space the competitive ratio is at least $|V|$ where V is the number of nodes in the metric space, even for $L < \Delta(G)$.
  Note that in the remaining cases, i.e., $\Delta(G)/2 \leq L \leq 3\Delta(G)$, the question of whether there exists a constant competitive algorithm depends on the metric space for both deterministic and randomized algorithms. Specifically, for

deterministic algorithms where $L = \Delta(G)$ it is easy to prove that there is no constant competitive algorithm for the uniform metric space. In contrast, there is a constant competitive algorithm for the line metric space. As mentioned above, for randomized algorithms the bound depends on the number of nodes in the metric space for a given diameter.

Application. For the uniform metric space (when all distances are unit size), our problem is equivalent to the Colored Packets with Deadlines problem. In this case our result improves the lower bound of [4]. Specifically, we improve their $1 + \Omega(\delta)$ lower bound to $1 + \Omega(\sqrt{\delta})$ and match their upper bound for the uniform metric space.

Embedding Result. One of the techniques that we use for the lower bound is the following embedding. Let $w(\mathrm{S})$ denote the weight of the star metric S (i.e., the sum of the weights of the edges of S). We prove that for any given metric space G on nodes V and for any vertex $v_0 \in V$ there exists a star metric S with leaves V and an embedding $f : \mathrm{G} \to \mathrm{S}$ from G to S (f depends on v_0) such that:

1. $w(\mathrm{S}) = \mathrm{MST}(\mathrm{G})$.
2. The weight of every Steiner tree in S that contains v_0 is not larger than the weight of the Steiner tree on the same nodes in G.

Note that this embedding is different from the usual embedding since we do not refer specifically to distances between vertices. Typically, an embedding is used to prove an upper bound by simplifying the metric space. In contrast, our embedding is used to prove a lower bound.

In order to prove the lower bound we first establish it for the star metric, and then extend it to general metric spaces. Note that a lower bound for a sub-graph is not a lower bound for the original graph. For example, a lower bound for an MST of a metric space G is not a lower bound for G since the algorithm may use additional edges to reduce the transition time.

2 The Model

We formally model the Dynamic Traveling Repair Problem with an arbitrary time window as follows. Let $G = (V, w)$ be a given metric space where V is a set of n nodes and w is a distance function. Let $s \in V$ be a given initial node. We are given an online sequence of requests for service. Each request is characterized by a pair $([r_i, d_i], v_i)$, where $r_i \in N_+$ and $d_i \in N_+$ are the respective arrival time and deadline of the request, and $v_i \in V$ is a node in the metric space G. The time to traverse from node v_i to node v_j is $w(v_i, v_j)$. For simplicity we assume that $w(v_i, v_j)$ is integral. Serving a request at some node requires unit size service time. The goal is to serve as many requests as possible within their time windows $[r_i, d_i]$, starting from node s.

Note that when all r_i are equal to 0 and all d_i are equal to B and the service time is negligible the problem reduces to the well-known orienteering problem with budget B and a prize for each node which is equal to the number of requests at this node. That is, finding a path of total distance at most B that maximizes the prize of all visited nodes.

Let ALG(σ), OPT(σ) denote the respective throughput of the online, optimal offline algorithms with respect to a sequence σ. We consider a maximization problem and hence $\inf_\sigma \mathrm{OPT}(\sigma)/\mathrm{ALG}(\sigma) \geq 1$.

3 Lower Bounds

3.1 Lower Bound for a Small Diameter Laxity Ratio (Case A and B)

In this section we consider Cases A and B. Let $\delta = TSP(G)/L$. If $\delta < 1$ (Case A), we show a strictly larger than 1 lower bound. Specifically, if $\delta \leq \frac{1}{256}$ we provide a lower bound of $1 + \Omega\left(\sqrt{\delta}\right)$. If $\delta > 1$ (Case B) we can use requests with a laxity of $256TSP(\mathrm{G})$ (i.e., $\delta = \frac{1}{256}$), and obtain a lower bound of 1.00054. Therefore, from now on we only consider Case A.

Lower Bound for a Star Metric. In this section we consider the case where the traveling time between nodes is represented by a star metric. This is also equivalent to the case where the traveling time from node i is w_i.

The general idea is that the adversary creates many requests with a large deadline at node v_0 at each time unit, and also blocks of fewer requests with close deadlines at other nodes. Any online algorithm must choose between serving many requests with a large deadline or traveling between many nodes and serving requests with close deadlines.

Recall that $w(\mathrm{S})$ denotes the weight of the star metric S (i.e., the sum of the weights of the edges of S). Let w_i denote the weight of the edge incident to vertex v_i. We define $F = \sqrt{w(\mathrm{S})L}$. Let $\delta = \frac{\mathrm{TSP(G)}}{L} = \frac{2w(\mathrm{S})}{L}$.

Theorem 1. *No deterministic or randomized online algorithm can achieve a competitive ratio better than* $1+\Omega\left(\sqrt{\delta}\right)$ *for any given star metric* S *when* $\delta \leq \frac{1}{256}$. *Otherwise, if* $\delta > \frac{1}{256}$, *the bound becomes* 1.00054.

Proof. Let S be a given star metric with nodes $V = \{v_0, \ldots, v_{n-1}\}$. We will construct a sequence $\sigma(\mathrm{S}, \mathrm{ALG})$ such that:

$$\frac{\mathrm{OPT}(\sigma)}{E(\mathrm{ALG}(\sigma))} \geq min\left\{ \frac{3-\delta}{3-\frac{1}{8}\left(\sqrt{\delta/2}\right)}, \frac{3}{3-\frac{1}{4}\left(\sqrt{\delta/2}\right)}, \frac{3}{3-\frac{1}{48}\left(\sqrt{\delta/2}\right)} \right\}$$

Note that we can assume, without loss of generality, that $\delta \leq \frac{1}{256}$, since otherwise one may use requests with a laxity of $256w(\mathrm{S})$ (i.e., $\delta = \frac{1}{256}$), and obtain a lower bound of 1.00054. Let $v_0 \in V$ be a type A node and the rest of the nodes type B. Let type A requests and type B requests refer to requests at a type A node and type B node, respectively. We begin by describing the sequence $\sigma(\mathrm{S}, \mathrm{ALG})$.

Sequence Structure: Recall that each request is characterized by a pair $([r_i, d_i], v_i)$, where $r_i \in N_+$ and $d_i \in N_+$ are the respective arrival time and deadline of the request, and v_i is a node in S. There are up to $N = \frac{L}{3F} = \frac{1}{3}\sqrt{\frac{L}{w(\mathrm{S})}}$

blocks, where each block consists of $3F$ time units. Let $t_i = 1 + 3(i-1)F$ denote the beginning time of block i. For each block i, where $1 \leq i \leq N$, F requests located at various nodes arrive at the beginning of the block. Specifically, $\frac{w_j}{w(\mathrm{S})-w_0}F$ type B requests $([t_i, L+t_i], v_j)$, for each $1 \leq j \leq n-1$, are released. A type A request $([t, 3L], v_0)$ is released at each time unit t in each block. Once the adversary stops the blocks, additional requests arrive (we call this the final event). The exact sequence is defined as follows:

1. $i \leftarrow 1$.
2. Add block i.
3. If with probability at least $1/4$ there are at least $F/2$ unserved type B requests at the end of block i (denoted by Condition 1), then L requests $([t_{i+1}, L + t_{i+1}], v_1)$ are released and the sequence is terminated. Clearly, t_{i+1} is the time of the final event. Denote this by Termination Case 1.
4. Else, if with probability at least $1/4$, at most $2F$ requests are served during block i (denoted by Condition 2), then $3L$ requests $([t_{i+1}, 3L], v_0)$ are released and the sequence is terminated. Clearly, t_{i+1} is the time of the final event. Denote this by Termination Case 2.
5. Else, if $i = N$ (there are N blocks, none of which satisfy Conditions 1 or 2) then $3L$ requests $([L+1, 3L], v_0)$ are released, and the sequence is terminated. Clearly, $L+1$ is the time of the final event. Denote this by Termination Case 3.
6. Else ($i < N$) then $i \leftarrow i+1$, Goto 2.

We make the following **observations:** (i) Each block consists of $3F$ time units. Hence, if ALG served at most $2F$ requests during a block, there must have been at least F idle time units. (ii) There are up to $\frac{1}{3}\sqrt{\frac{L}{w(\mathrm{S})}}$ blocks and each block consists of $3\sqrt{w(\mathrm{S})L}$ time units. Hence, the time of the final event is at most $L+1$. (iii) Exactly one type A request arrives at each time-slot until the final event. Hence, at most L type A requests arrive before (not including) the final event. (iv) During each block, exactly F type B requests arrive, which sum up to at most $L/3$ type B requests before (not including) the final event.

Now we can analyze the competitive ratio of $\sigma(\mathrm{S}, \mathrm{ALG})$. Consider the following possible sequences (according to the termination type):

1. Termination Case 1: Let Y denote the number of requests in the sequence. According to the observations, the sequence consists of at most L type A requests, and at most $\frac{4}{3}L$ type B requests ($L/3$ until the final event and L at the final event). Hence, $Y \leq L + \frac{4}{3}L \leq 3L$.
   - **We bound the performance of ALG:** At time t_{i+1} there is a probability of at least $1/4$ that ALG has $L + F/2$ unserved type B requests. Since type B requests have a laxity of L, ALG can serve at most $L+1$ of them, and must drop at least $F/2 - 1$. The expected number of served requests is

$$E(\mathrm{ALG}(\sigma)) \leq Y - \frac{1}{4}(F/2 - 1) = Y - \frac{1}{8}F + 1/4.$$

   - **We bound the performance of an algorithm OPT′**: OPT′ serves the requests in three stages:

- **Type B requests that arrive before the final event:** Recall that all type B requests in a block arrive at once in the beginning of the block. In each block OPT' first serves all requests at node v_1, then all requests at node v_2, and so on. It is clear that OPT' needs at most $F + 2w(\mathrm{S})$ time units to serve the requests (F for serving and $2w(\mathrm{S})$ for traveling). OPT' serves the requests starting from the beginning of the block. Recall that $L \geq 256w(\mathrm{S})$ and $F = \sqrt{w(\mathrm{S})L}$. Therefore $2F \geq 512w(\mathrm{S})$. Since the block's size is $3F$, there are enough time units. Moreover, since $L \geq 256w(\mathrm{S})$, $L \geq 16\sqrt{w(\mathrm{S})L} = 16F > F + 2w(\mathrm{S})$. Hence, all requests can be served before their deadline.
- **Type B requests that arrive during the final event:** The L requests $([t_{i+1},\ L + t_{i+1}], v_1)$ that arrive during the final release time are served by OPT' consecutively from time t_{i+1}. OPT' can serve L requests, except for one travel phase, and hence may lose at most $2w(\mathrm{S})$ requests. According to our observations, the time of the final event t_{i+1} is at most $L + 1$. Hence, OPT' serves all type B requests until time unit $2L$.
- **Type A requests:** OPT' serves the L type A requests consecutively from time unit $2L+1$. Since the deadlines are $3L$, OPT' serves all type A requests.

We conclude that $\mathrm{OPT}(\sigma) \geq \mathrm{OPT}'(\sigma) \geq Y - 2w(\mathrm{S})$.

The competitive ratio is

$$\frac{\mathrm{OPT}(\sigma)}{E(\mathrm{ALG}(\sigma))} \geq \frac{Y-2w(\mathrm{S})}{Y-\frac{1}{8}F+1/4} \geq \frac{3L-2w(\mathrm{S})}{3L-\frac{1}{8}F+1/4} \geq \frac{3L-2w(\mathrm{S})}{3L-\frac{1}{8}\left(\sqrt{w(\mathrm{S})L}\right)+1/4} = 1 + \Omega\left(\sqrt{\delta}\right).$$

Here the second inequality holds since $Y \leq 3L$, the number is above 1 and the numerator and the denominator increase by the same value.

2. Termination Case 2: The sequence consists of more than $3L$ type A requests, and all deadlines are at most $3L$.
   - **We bound the performance of ALG:** The probability that ALG was idle for F time units is at least $1/4$. Hence, the expected number of served requests is $E(\mathrm{ALG}(\sigma)) \leq 3L - \frac{1}{4}F$.
   - **We bound the performance of OPT′**: At each time unit until the final event, OPT' serves the type A request that arrived at that particular time unit. Consequently, from the final event until time unit $3L$, OPT' serves the type A requests that arrived at the final event. Therefore, OPT' serves $3L$ type A requests, and so $\mathrm{OPT}(\sigma) \geq \mathrm{OPT}'(\sigma) \geq 3L$.

   The competitive ratio is

$$\frac{\mathrm{OPT}(\sigma)}{E(\mathrm{ALG}(\sigma))} \geq \frac{3L}{3L - \frac{1}{4}F} = \frac{3L}{3L - \frac{1}{4}\left(\sqrt{w(\mathrm{S})L}\right)} = 1 + \Omega\left(\sqrt{\delta}\right).$$

3. Termination Case 3: the sequence consists of $3L$ type A requests, and all deadlines are at most $3L$.
   - **We bound the performance of ALG:** Let U_i be the event that the number of unserved type B requests at the end of block i is less than

$F/2$. If U_i occurs, then let j_k, $1 \leq k \leq r$, be the type B nodes visited by ALG in block i. At least $F/2$ requests that arrived in this block have to be served (recall that F type B requests arrive at the beginning of each block). Therefore,

$$\frac{w_{j_1}}{w(\mathrm{S})-w_0}F + \frac{w_{j_2}}{w(\mathrm{S})-w_0}F + \cdots + \frac{w_{j_r}}{w(\mathrm{S})-w_0}F \geq F/2,$$

and so

$$w_{j_1} + w_{j_2} + \cdots + w_{j_r} \geq \frac{w(\mathrm{S})-w_0}{2}.$$

Let E_i be the event that more than $2F$ requests are served during block i. If event U_{i-1} and E_i occur, then there are at most $3F/2$ unserved type B requests in the beginning of block i (F arrived at the beginning of the block and there are at most $F/2$ from the previous block) but more than $2F$ requests were served. Therefore, at least one type A request was served during the block. Combining the results, if U_i, U_{i-1}, and E_i occur then:

- During block i at least $(w(\mathrm{S})-w_0)/2$ time units were used for traveling between type B nodes.
- A Type A request was served during the block.

A block i is called *good* if the events U_i, U_{i-1}, and E_i occur. For any two (consecutive) good blocks the traveling cost is at least $(w(S)-w_0)/2+w_0 \geq w(S)/2$. Since none of the blocks satisfy Condition 1 or 2, it follows that for all i such that $\frac{1}{3}\sqrt{\frac{L}{w(\mathrm{S})}} \geq i \geq 1$ we have: $\Pr[U_i] \geq 3/4, \Pr[U_{i-1}] \geq 3/4$, and $\Pr[E_i] \geq 3/4$. Therefore:

$$\begin{aligned}\Pr[U_i \cap U_{i-1} \cap E_i] &= 1 - \Pr[\neg(U_i \cap U_{i-1} \cap E_i)] \\ &= 1 - \Pr[\neg U_i \cup \neg U_{i-1} \cup \neg E_i] \geq 1 - 1/4 - 1/4 - 1/4 = 1/4.\end{aligned}$$

The sequence consists of $\frac{1}{3}\sqrt{\frac{L}{w(\mathrm{S})}}$ blocks. Therefore, the expected number of good blocks is $\frac{1}{4} \cdot \frac{1}{3}\sqrt{\frac{L}{w(\mathrm{S})}} = \frac{1}{12}\sqrt{\frac{L}{w(\mathrm{S})}}$ and of disjoint pairs of blocks is $\frac{1}{24}\sqrt{\frac{L}{w(\mathrm{S})}}$. Consequently, the expected number of lost requests is at least $\frac{1}{24}\sqrt{\frac{L}{w(\mathrm{S})}}\frac{w(\mathrm{S})}{2}$ and of served requests is:

$$E(\mathrm{ALG}(\sigma)) \leq 3L - \frac{1}{48}w(\mathrm{S})\sqrt{\frac{L}{w(S)}} = 3L - \frac{1}{48}\left(\sqrt{w(\mathrm{S})L}\right).$$

– **We bound the performance of OPT′**: At each time unit until the final event, OPT′ serves the type A request that arrived at the same time unit. Consequently, from the final event until time unit $3L$, OPT′ serves the type A requests that arrived at the final event. Therefore, OPT′ serves 3 L type A requests, and so $\mathrm{OPT} \geq \mathrm{OPT}' \geq 3L$.

The competitive ratio is

$$\frac{\mathrm{OPT}(\sigma)}{E(\mathrm{ALG}(\sigma))} \geq \frac{3L}{3L - \frac{1}{48}\left(\sqrt{w(\mathrm{S})L}\right)} = 1 + \Omega\left(\sqrt{\delta}\right).$$

Note that in all 3 cases we get $1 + \Omega\left(\sqrt{\delta}\right)$. This completes the proof. ■

The following straightforward corollary improves the lower bound of $1 + \Omega\left(C/L\right)$ from [4]. Recall that n is the number of nodes in the metric space.

Corollary 1. *No deterministic or randomized online algorithm can achieve a competitive ratio better than* $1 + \Omega\left(\sqrt{n/L}\right)$ *when all traveling times takes one unit of time and* $L \geq 256n$. *Otherwise, if* $L < 256n$, *the bound becomes* 1.00054.

Proof. Let S be a star metric such that the weight of each edge is equal to $1/2$. Clearly, traveling between any two nodes requires one time unit and $w(\mathrm{S}) = n/2$. Applying Theorem 1, we obtain the lower bound of $1 + \Omega\left(\sqrt{n/L}\right)$ (note that in this case $\delta = n/L$). ■

Embedding of Metric Spaces. In this section we describe an embedding of a general metric space into a star metric with special properties. We begin by introducing some new definitions:

- We define $w(\mathrm{T}) = \sum_{e \in V} w(e)$ for a rooted tree $\mathrm{T} = (V, E)$, and let $P_{\mathrm{T}}(v)$ denote the parent of node v in a rooted tree T.
- Let S be a star metric with a center c. We define $w_{\mathrm{S}}(V) = \sum_{v \in V} w(c, v) = \sum_{v_i \in V} w_i$. It is clear that for a star S with leaves V, $w_{\mathrm{S}}(V) = w(\mathrm{S})$.
- Let $\mathrm{T}_{\mathrm{G}}(V)$ be the minimum weight connected tree that contains the set V (i.e., the minimum Steiner tree on these points) in the metric space G.

Recall that $MST(\mathrm{G})$ denotes the weight of the minimal spanning tree (MST) in the metric space G.

Theorem 2. *For any given metric space* G *on nodes* V *and for any vertex* $v_0 \in V$ *there exists a star metric* S *with leaves* V *and an embedding* $f : \mathrm{G} \to \mathrm{S}$ *from* G *to* S (f *depends on* v_0) *such that:*

1. **Property 1:** $w(\mathrm{S}) = MST(\mathrm{G})$.
2. **Property 2:** *For every* $V' \subseteq V$ *such that* $v_0 \in V'$, $w(\mathrm{T}_{\mathrm{G}}(V')) \geq w_{\mathrm{S}}(V')$.

Proof. We prove the theorem by describing a star metric that satisfies the required properties. Let G be a given metric space on nodes V with a vertex $v_0 \in V$. Let T be the MST for G created by applying Prims' algorithm with the root v_0. Let S be a star metric with leaves V such that for each $u \in V$, $w_u = w(u, P_{\mathrm{T}}(u))$. Clearly, $w_{v_0} = 0$. We prove that S and v_0 satisfy the theorem's properties:

Property 1: Clearly, $w(\mathrm{S}) = w(\mathrm{T})$, and since T is a MST for G, $w(\mathrm{S}) = w(\mathrm{T}) = \mathrm{MST(G)}$.

Property 2: Assume by a contradiction that there exists $V' = \{v_0, v_{i_1}, \ldots, v_{i_r-1}\} \subseteq V$ such that $w(\mathrm{T_G}(V')) < w_S(V') = \sum_{j=1}^{r-1} w(v_{i_j}, P_{\mathrm{T}}(v_{i_j}))$.

Let $V'' = \{v_0, v_{i_1}, \ldots, v_{i_r-1}, \ldots, v_{i_k}\}$ be the vertices of $T_{\mathrm{G}}(V')$ (note that $V' \subseteq V''$). Consider the following process. Let $T' = T_{\mathrm{G}}(V')$. Run Prim's algorithm from node v_0. Each time Prim's adds a new node not in T', we add Prim's edge to T'. Note that Prim starts from node $v_0 \in T_{\mathrm{G}}(V')$ and we add each node not in T'. Hence, when Prim's algorithm finishes, T' is a tree on nodes V. Moreover, T' is T where edges $\big(v_{i_1}, P_{\mathrm{T}}(v_{i_1})\big), \ldots, \big(v_{i_k}, P_{\mathrm{T}}(v_{i_k})\big)$ were replaced by the edges of $T_{\mathrm{G}}(V')$. Since we assumed that $w(\mathrm{T_G}(V')) < \sum_{j=1}^{r-1} w(v_{i_j}, P_{\mathrm{T}}(v_{i_j}))$, and clearly $\sum_{j=1}^{r-1} w(v_{i_j}, P_{\mathrm{T}}(v_{i_j})) \leq \sum_{j=1}^{k} w(v_{i_j}, P_{\mathrm{T}}(v_{i_j}))$, we have $w(T') < w(T)$. This is a contradiction since T is an MST. ■

Lower Bound for a General Metric Space. In this section we consider the case where the traveling time between nodes is represented by a metric space G. Note that a lower bound for a star metric space does not imply a lower bound for a general metric space. Recall that $\delta = \mathrm{TSP(G)}/L < 1$.

We use the embedding from Theorem 2 to prove a $1 + \Omega\big(\sqrt{\delta}\big)$ lower bound.

Theorem 3. *No deterministic or randomized online algorithm can achieve a competitive ratio better than* $1 + \Omega\big(\sqrt{\delta}\big)$ *for any given metric space* G*, when* $\delta \leq \frac{1}{256}$*. Otherwise, if* $\delta > \frac{1}{256}$*, the bound becomes* 1.00054*.*

3.2 Lower Bound for a Large Diameter Laxity Ratio (Case C)

In this section we consider the case where $L < \Delta(G)/2$ (recall that $\Delta(G)$ is the diameter and L is the laxity), and we show that the competitive ratio of any deterministic algorithm is unbounded.

Theorem 4. *No deterministic online algorithm can achieve a bounded competitive ratio for any metric space in which* $L < \Delta(G)/2$*.*

Proof. Let G be any metric space. Every $\Delta(G) + 1$ units of time we introduce a request with a laxity of L to a node which is at a distance of at least $\Delta(G)/2$ from the current location of the online algorithm (note that there is always such a node). It is clear that the algorithm can not serve any requests while OPT can serve all the requests. ■

4 Upper Bounds

4.1 Asymptotically Optimal Algorithm for Case A

In this section we design a deterministic online algorithm, for a general metric space. The algorithm achieves a competitive ratio of $1 + o(1)$ when the minimum

In each phase $\ell = 1, 2, \ldots$, do

- Beginning of the phase (at time $K(\ell - 1)$)
  - Decrease the deadline of each unserved request $([r, d], v)$ from d to $K\lfloor d/K \rfloor$.
  - Let R^ℓ be the collection of unserved requests such that their decreased deadline was not exceeded. Let S^ℓ be the K-length prefix of EDF (earliest deadline first) schedule (according to the modified deadline) of R^ℓ. Let $S^\ell_j \subseteq S^\ell$ denote the subset of requests at node v_j in S^ℓ.
    Let $v_{i_1}, v_{i_2}, ..., v_{i_n}$ denote the order of the nodes in the minimal TSP (or approximation).
  - ρ_ℓ consists of all requests of $S^\ell_{i_1}$ served consecutively, then all requests of $S^\ell_{i_2}$ served consecutively, and so on.
- During the phase (between time $K(\ell - 1)$ and time $K\ell$)
  - The requests are served according to ρ_ℓ (unserved requests in the suffix of ρ_ℓ, due to the end of the phase are dropped).

Fig. 1. Algorithm TSP-EDF.

laxity of the requests is asymptotically larger than the weight of the TSP (as shown in the previous sections, this is essential).

The algorithm is a natural extension of the BG algorithm from [4]. Our algorithm, which we call TSP-EDF, formally described in Fig. 1, works in phases of $K = \sqrt{\text{TSP(G)}L}$ time units. In each phase the algorithm serves requests node by node. The order of the nodes is determined by the minimum TSP or an approximation. The algorithm achieves a competitive ratio of $1 + O\left(\sqrt{\text{TSP(G)}/L}\right)$ for $L > 10\text{TSP(G)}$.

Theorem 5. *The algorithm* TSP-EDF *attains a competitive ratio of* $1 + O\left(\sqrt{\text{TSP(G)}/L}\right)$.

4.2 Constant Approximation Algorithm for Case B

In this section we design a deterministic online algorithm, for a general metric space where $L > 9\Delta(G)$ (recall that $\Delta(G)$ is the diameter of G). The algorithm achieves a constant competitive ratio. As shown in the previous section, no online algorithm can achieves a competitive ratio better 1.00054. A more precise analysis can replace $L > 9\Delta(G)$ with $L > (2+\epsilon)\Delta(G)$ for any $\epsilon > 0$ and the approximation becomes $O(\frac{1}{\epsilon})$.

The algorithm which we call ORIENT-WINDOW (Fig. 2) combines the following ideas.

- The algorithm works in phase of $K = 3\Delta(G)$. In each phase the algorithm serves only requests that arrived in the previous phases, and will not expired during the phase. Due to this perturbation we lose a constant factor.
- The decision which requests will be served in a phase ignore their deadlines. Due to this violation of EDF we lose a constant factor.

In each phase $\ell = 1, 2, \ldots$, do

- Beginning of the phase (at time $K(\ell - 1)$)
  - Decrease the deadline of each unserved request $([r, d], v)$ from d to $K\lfloor d/K \rfloor$.
  - Let R^ℓ be the collection of unserved requests such that their decreased deadline was not exceeded. Let $S_j^\ell \subseteq R^\ell$ denote the subset of requests at node v_j in R^ℓ.
  - Using a constant approximation algorithm solve the unrooted orienteering problem with budget $\Delta(G)$ where the prize of a node v_j is the number of requests in S_j^ℓ. Let $v_{i_1}, v_{i_2}, ..., v_{i_r}$ denote the order of the nodes in the solution.
  - ρ_ℓ consists of all requests of $S_{i_1}^\ell$ scheduled consecutively, then all requests of $S_{i_2}^\ell$ scheduled consecutively, and so on.
- During the phase (between time $K(\ell - 1)$ and time $K\ell$)
  - The requests are served according to ρ_ℓ (unserved requests in the suffix of ρ_ℓ, due to the end of the phase are dropped).

Fig. 2. Algorithm ORIENT-WINDOW

– In each phase the algorithm serves requests node by node. The order of the nodes is determined by solving an orienteering problem. Since a constant approximation algorithm is known to the orienteering problem, we lose a constant factor.

Theorem 6. *The algorithm ORIENT-WINDOW attains a competitive ratio of* $O(1)$.

References

1. Arkin, E.M., Mitchell, J.S., Narasimhan, G.: Resource-constrained geometric network optimization. In: SOCG, pp. 307–316. ACM (1998)
2. Ausiello, G., Feuerstein, E., Leonardi, S., Stougie, L., Talamo, M.: Algorithms for the on-line travelling salesman. Eindhoven University of Technology, Department of Mathematics and Computing Sciences (1999)
3. Awerbuch, B., Azar, Y., Blum, A., Vempala, S.: New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen. SIAM J. Comput. **28**(1), 254–262 (1998)
4. Azar, Y., Feige, U., Gamzu, I., Moscibroda, T., Raghavendra, P.: Buffer management for colored packets with deadlines. In: SPAA 2009, pp. 319–327. ACM (2009)
5. Bansal, N., Blum, A., Chawla, S., Meyerson, A.: Approximation algorithms for Deadline-TSP and vehicle routing with time-windows. In: STOC, pp. 166–174. ACM (2004)
6. Bar-Yehuda, R., Even, G., Shahar, S.M.: On approximating a geometric prize-collecting traveling salesman problem with time windows. J. Algorithms **55**(1), 76–92 (2005)
7. Blum, A., Chawla, S., Karger, D.R., Lane, T., Meyerson, A., Minkoff, M.: Approximation algorithms for orienteering and discounted-reward TSP. In: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, pp. 46–55. IEEE (2003)
8. Chekuri, C., Korula, N.: Approximation algorithms for orienteering with time windows. arXiv preprint arXiv:0711.4825 (2007)

9. Chekuri, C., Korula, N., Pál, M.: Improved algorithms for orienteering and related problems. ACM Trans. Algorithms (TALG) **8**(3), 23 (2012)
10. Chekuri, C., Kumar, A.: Maximum coverage problem with group budget constraints and applications. In: Jansen, K., Khanna, S., Rolim, J.D.P., Ron, D. (eds.) APPROX/RANDOM -2004. LNCS, vol. 3122, pp. 72–83. Springer, Heidelberg (2004). doi:10.1007/978-3-540-27821-4_7
11. Desrochers, M., Desrosiers, J., Solomon, M.: A new optimization algorithm for the vehicle routing problem with time windows. Oper. Res. **40**(2), 342–354 (1992)
12. Desrochers, M., Lenstra, J.K., Savelsbergh, M.W., Soumis, F.: Vehicle routing with time windows: optimization and approximation. Veh. Routing: Methods Stud. **16**, 65–84 (1988)
13. Feuerstein, E., Stougie, L.: On-line single-server dial-a-ride problems. Theoret. Comput. Sci. **268**(1), 91–105 (2001)
14. Golden, B.L., Levy, L., Vohra, R.: The orienteering problem. Naval Res. Logist. **34**(3), 307–318 (1987)
15. Irani, S., Lu, X., Regan, A.: On-line algorithms for the dynamic traveling repair problem. J. Sched. **7**(3), 243–258 (2004)
16. Jaillet, P., Lu, X.: Online traveling salesman problems with rejection options. Networks **64**(2), 84–95 (2014)
17. Karuno, Y., Nagamochi, H.: 2-approximation algorithms for the multi-vehicle scheduling problem on a path with release and handling times. Discrete Appl. Math. **129**(2), 433–447 (2003)
18. Krumke, S.O., Megow, N., Vredeveld, T.: How to whack moles. In: Solis-Oba, R., Jansen, K. (eds.) WAOA 2003. LNCS, vol. 2909, pp. 192–205. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24592-6_15
19. Nagamochi, H., Ohnishi, T.: Approximating a vehicle scheduling problem with time windows and handling times. Theoret. Comput. Sci. **393**(1), 133–146 (2008)
20. Savelsbergh, M.W.: Local search in routing problems with time windows. Ann. Oper. Res. **4**(1), 285–305 (1985)
21. Tan, K.C., Lee, L.H., Zhu, Q., Ou, K.: Heuristic methods for vehicle routing problem with time windows. Artif. Intell. Eng. **15**(3), 281–295 (2001)
22. Thangiah, S.R.: Vehicle Routing with Time Windows Using Genetic Algorithms. Citeseer (1993)
23. Tsitsiklis, J.N.: Special cases of traveling salesman and repairman problems with time windows. Networks **22**(3), 263–282 (1992)