On-Line Algorithms for the Dynamic Traveling Repair Problem

Sandy Irani* Xiangwen Lu † Amelia Regan ‡

Abstract

We consider the dynamic traveling repair problem (DTRP) in which requests with deadlines arrive through time on points in a metric space. Servers move from point to point at constant speed. The goal is to plan the motion of servers so that the maximum number of requests are met by their deadline. We consider a restricted version of the problem in which there is a single server and the length of time between the arrival of a request and its deadline is uniform. We give upper bounds for the competitive ratio of two very natural algorithms as well as several lower bounds for any deterministic algorithm. Most of the result in this paper are expressed a function of β, the diameter of the metric space.

1 Introduction

In the dynamic traveling repair problem (DTRP), servers move from point to point in a metric space. The speed of each server is constant, so the time it takes to travel from one point to another is proportional to the distance between the two points. Time is continuous and at any moment a request for service can arrive at a point in the space. Each job also specifies a deadline. If a job is to be serviced, a server must reach the point where the request originated by its deadline. The goal of the algorithm is to service as many incoming requests as possible by their deadline. We consider *online* algorithms which learn about a new request only when it arrives into the system. The DTRP has many practical applications. Examples include routing utility and other repair fleets as well as routing of tanker trucks [3, 4].

In this paper, we consider the following restrictions to the DTRP problem:

- **Single Server:** We address the problem where there is only one server.
- **Window sizes are uniform:** We restrict our attention to the case where the time between a job's arrival and its deadline is a fixed number W for all jobs. For the remainder of this paper, will normalize all values so that the window length is 1.
- **Notification time:** One can consider variations of this problem depending on when the online algorithm is required to commit to accepting or rejecting a particular request. We adopt this as a parameter of the problem. Thus, an online algorithm must decide whether to accept or reject a request within time α of its arrival. We call α the *notification deadline*. If $\alpha = 0$, the server must decide immediately upon the arrival of a job into the system whether it will be served. If $\alpha = 1$, the decision is made only when the request is served or its deadline has passed.

  Most of the results in this paper focus on the case where $\alpha = 1$, although there is one lower bound which pertains to the general case. The situation in which the notification deadline is 1, is in fact reasonable operationally. Many cable TV or other repair fleets accept all requests and subcontract, sometimes at the last minute, requests they are unable to service with their primary fleet. Local truckload trucking operations, which can be modeled as a DRTP with asymmetric travel costs, typically operate in the same way. A primary carrier accepts all requests for service and then subcontracts to local owner operators if it is unable to service requests within their time windows.
- **Bounded diameter:** As we shall see in the results below, it is necessary to consider metric spaces of bounded diameter. We assume that the diameter of the metric space is bounded by a constant β. Our results will be expressed as a function of β.
- **Service time:** In our results below, we assume that once the server arrives at the location of the request, the request is served instantly. A natural extension of this problem would be to have a service time required for each request. All our results hold if there is a small, fixed service time except for

*Information and Computer Science, UC Irvine, 92697, irani@ics.uci.edu. Supported in part by NSF grants CCR-0105498 and CCR-9625844 and by ONR Award N00014-00-1-0617.

†Institute of Transportation Studies, UC Irvine, 92697, xlu@uci.edu. Partially supported by NSF Grant 9875675.

‡Department of Civil and Environmental Engineering, UC Irvine, 92697, aregan@uci.edu. Partially supported by NSF Grant 9875675.

Theorem 6.1 which requires the fact that the service time is zero.

A word about metric spaces: some previous work on related problems ([2]) focus on continuous metric spaces in which the path between any two points is continuous, formed by points in the metric space. In particular, this implies that the server can do a U-turn at any point in time. Our results hold for the more general case where the path between two points may not be comprised of points in the metric space and the server, once embarked on an edge between two point, must traverse the edge before changing directions. In this case, β indicates the longest time that it takes the server to reach any point in the metric space maximized over all possibilities for its current location and destination.

We use competitive analysis to evaluate online algorithms, which has now become the most widely accepted means of evaluating an online algorithm. In competitive analysis, the performance of an online algorithm is compared to the performance of the optimal offline algorithm which knows about all future jobs. In the problem addressed in this paper, the algorithms are seeking to maximize their *benefit* for a particular sequence of request arrivals which is the number of requests satisfied by their deadline. Let $B_A(\sigma)$ denote the benefit obtained by algorithm A on request sequence σ. Let OPT denote the optimal online algorithm. The algorithm A is said to be c-competitive if there is a constant d such that for every sequence σ,

$$c \cdot B_A(\sigma) + d \geq B_{OPT}(\sigma).$$

The *competitive ratio* of the algorithm A, denoted c_A, is the infimum over c such that A is c-competitive.

2 Related Work

The dynamic traveling repair problem (DTRP) falls into a more general class of problems known as stochastic vehicle routing. This class includes the dynamic traveling salesman problem (DTSP), the probabilistic traveling salesman problem (PTSP), probabilistic traveling salesman location problem and other problems such as the probabilistic shortest path problem and the dynamic vehicle allocation problem. Powell, Jaillet and Odoni provide an excellent review of these problems [8]. In the DTSP, customer locations are known in advance and service requests arrive according to a Poisson point process at each node. The objective is to determine a dispatching strategy which minimizes customers' expected waiting time. This problem is discussed by Psarafits [9]. In the PTSP, an a priori tour must be constructed for a network in which each node has a given probability of requiring a visit [6]. The DTRP is also usually considered under the assumption that the input is generated by a known probability distribution. In this paper, we analyze algorithms for the DTRP using competitive analysis in which the input sequence can be arbitrary.

From the online algorithms literature, this problem most closely resembles the On-Line Traveling Salesman Problem (OLTSP) studied by Ausiello *et al.*[2]. In their version of the problem, requests do not have deadlines and the goal is to minimize the total completion time of all requests. The completion time of a request is the length of time that passes after the request is released until it is served. They give a 2.5-competitive algorithm for all *continuous* metric spaces in which the server can turn around at any point in time and return to the last point visited. They also consider a variant of the problem called the *Homing*-OLTSP in which the server is required to return to its departure point before serving another request.

Other related work includes online graph exploration problems in which all nodes of a graph must be visited and the edges of the graph are revealed only when the server has visited an incident node [5]. In our problem, the entire graph (or metric space) is known in advance and the locations of requests are revealed on an on-line manner. Another related problem is the k-server problem [7]. In the k-server problem, it is assumed that servers can travel instantaneously to their destination, although the goal is to minimize the total distance traveled by all servers. Also, in the k-server problem, requests must be visited in the order in which they arrive but in the problem addressed here, the server can choose the order in which to satisfy the requests.

3 Summary of Results

We present two very natural algorithms for the DTRP and prove upper bounds for their competitive ratios. All of our results are given as a function of β which is the maximum time required to travel between the two farthest points in the metric space. The particular value of β for that metric space determines which bound is stronger.

3.1 The BATCH Algorithm

The first algorithm is called the BATCH algorithm. Time is divided into intervals of length 1/2. At the beginning of an interval, the jobs which arrived in the previous interval are considered. The algorithm then determines the maximum number of requests from this set it can reach from its current position in the next interval of 1/2 and follows that path. Thus, the set of requests considered in a given interval of 1/2 are those requests which arrived in the previous interval and expire in the following interval. The algorithm must

solve an instance of the prize-collecting TSP problem with a fixed starting position and arbitrary ending location for every interval. (See [1] for a discussion of the prize-collecting TSP). BATCH requires that the notification deadline (α) be at least 1/2. We prove that BATCH is $(3/(1-2\beta))$-competitive. We also give a lower bound for BATCH which shows that our analysis is tight.

3.2 The Algorithm Double-Gain

An appealing feature of Double-Gain (DG) is that it is well defined for the more general version of DTRP with non-uniform windows. Our proof for the competitiveness of DG, however, only applies for uniform window lengths.

The state of an algorithm can be defined by its current location and the set of outstanding requests which have not yet expired. The maximum number of outstanding requests which can be served given that the algorithm is in state S at time t is well defined. The route which achieves this maximum will be called the *new* route for time t. $N(t)$ will denote the set of requests which are served on the new route for time t. We assume that the the number of outstanding requests will be small enough that the algorithm can determine $N(t)$ optimally. At any given moment in which there are outstanding requests in the system, the algorithm has a route which it is currently executing which we will call the *current* route. If the algorithm finishes its current route at time t, it begins immediately executing the new route for time t. If there are no outstanding requests in the system which the algorithm can reach, the current route for the algorithm is simply to stay put. Let $C(t)$ denote the set of outstanding requests which have yet to be served on the algorithm's current route at time t. At any given point in time t, the server will switch to the new route at time t if the following condition holds:

$$|N(t) - C(t)| \geq 2(|C(t) - N(t)|).$$

That is, the algorithm switches routes if the number of requests gained is at least twice the number of requests lost by the switch. Note that the proof only requires that the algorithm find one route that which acieves $|N(t)|$ at any given time t, although there may be more than one which achieves the maximum.

We prove that the competitive ratio of Double-Gain is at most

$$4\left\lceil \frac{2}{1-\beta} \right\rceil \left(\left\lceil \frac{2}{1-\beta} \right\rceil + 1 \right).$$

3.3 Lower Bounds

We first show that for any notification deadline α, and any $\beta \leq \alpha$, there is a metric space with diameter β for which no competitive algorithm exists. We then prove three lower bounds which hold for any deterministic online algorithm and any value for α. Each lower bound is a function of β. Thus, the value of β determines which lower bound is strongest. We prove that there is no algorithm which will achieve a competitive ratio on all metric spaces lower than the following bound:

$$\max\left\{2, \frac{1}{1-\beta}, \frac{\beta}{(1-\beta)^2}\right\}.$$

The proof for the last bound of $\beta/(1-\beta)^2$ holds only when $\beta > 1/2$. However, if $\beta \leq 1/2$, the lower bound of 2 will be stronger anyway. Note that this bound shows that the bound proven for Double-Gain is within a constant factor of the best possible competitive ratio.

4 Bounds for the BATCH Algorithm

THEOREM 4.1. *BATCH is $3/(1-2\beta)$-competitive.*

Proof. Consider a sequence of arrivals σ and the schedule which OPT chooses for σ. Time is divided into intervals of length 1/2. A *batch* is a set of jobs which arrives in one of these intervals. Number the batches from 1 to m. Batch m is the last batch in σ in which any jobs arrive. Let $S_{i,j}$ be the set of jobs which arrive during interval i and which OPT schedules during interval j. Let $s_{i,j} = |S_{i,j}|$. Clearly, if $i < j-2$ or $i > j$, then $S_{i,j} = \emptyset$.

$$B_{OPT}(\sigma) = \sum_{j=1}^{m+2} (s_{j-2,j} + s_{j-1,j} + s_{j,j}).$$

Suppose that BATCH is at location q at the beginning of batch i. It will consider the set of requests S_{i-1} which arrived during batch $i-1$ and find the tour which reaches the most requests from S_{i-1}, starting at q and lasting for at most 1/2. Note that $S_{i-1,i-1} \cup S_{i-1,i} \cup S_{i-1,i+1} \subseteq S_{i-1}$. Also note that the optimal can serve each of $S_{i-1,i-1}$, $S_{i-1,i}$ and $S_{i-1,i+1}$ during a single interval. This means that there is a tour of length at most 1/2 which covers all of $S_{i-1,i-1}$, given that the starting position can be chosen arbitrarily. The same is true for $S_{i-1,i}$ and $S_{i-1,i+1}$.

Now since β is the longest time it takes to travel between two points, its takes a fraction of 2β of one batch to travel between the two farthest points in the metric space. This means that BATCH can pick its optimal starting location, take time β to travel to that location and has a remaining time of $1/2 - \beta$ to service requests. Thus, one of the options is to pick which of $S_{i-1,i-1}$, $S_{i-1,i}$ and $S_{i-1,i+1}$ is the largest. Call it $S_{i-1,j}$. Consider the tour which covers all the requests in $S_{i-1,j}$ in time 1/2. Now pick the portion of the path

which is $(1/2-\beta)$ long and covers the most requests. This portion of the path will reach at least $(1-2\beta)s_{i-1,j}$ requests. BATCH has the option of traveling to the beginning of that portion of the path and visiting all the requests along that portion of the path. This means that the number of jobs which BATCH visits during interval i is at least

$$(1-2\beta)\max\{s_{i-1,i-1}, s_{i-1,i}, s_{i-1,i+1}\}$$

and

$$\begin{aligned} & B_{BATCH}(\sigma) \\ \geq & \sum_{i=2}^{m+1}(1-2\beta)\max\{s_{i-1,i-1}, s_{i-1,i}, s_{i-1,i+1}\} \\ \geq & \left(\frac{1-2\beta}{3}\right)\sum_{i=2}^{m+1}(s_{i-1,i-1}+s_{i-1,i}+s_{i-1,i+1}) \\ = & \left(\frac{1-2\beta}{3}\right)\sum_{j=1}^{m+2}(s_{j-2,j}+s_{j-1,j}+s_{j,j}) \\ = & \left(\frac{1-2\beta}{3}\right)B_{OPT}(\sigma). \end{aligned}$$

□

The following result shows that our analysis for BATCH is tight.

THEOREM 4.2. *There is a metric space $\mathcal{M}$ such that for any d and any constant $c < 3/(1-2\beta)$, there is a sequence of arrivals σ such that*

$$c \cdot B_{BATCH}(\sigma) + d \leq B_{OPT}(\sigma).$$

Proof. We will re-normalize the distances so that the distance which can be traversed in one batch is one. This means that a window length is 2. Pick a small constant δ such that $1/\delta$ is an integer. Since δ can be arbitrarily small, we can pick it so that $(1-2\beta)/\delta$ can be arbitrarily close to an integer. Thus, we will assume that $(1-2\beta)/\delta$ is an integer r and $1/\delta$ is an integer m. We will position $3m$ points in the metric space, each a distance of δ away from each other. Call this set A. There will be another set of points B which will consist of $r+1$ points which are also each a distance of δ away from each other. Each point from A is a distance of 2β away from any point in B. Note that since the window length of each job is 2, this means that the diameter of the space is β.

The points in A will be labeled $a_{i,j}$, where i is an integer in the range $0, 1, \ldots, m-1$ and $j \in \{0,1,2\}$ The points in B will be labeled $b_0, b_1, \ldots, b_r$. In the middle of every batch, a request will arrive on each point in B.

No jobs arrive on points in A during the first batch. Let ϵ be a small constant such that $\epsilon << \delta$. Requests will arrive during batch $b \geq 2$ on points in A is follows:

For b MOD $3 \equiv 0$: For each $i \in \{0, 1, \ldots, m-1\}$:

- A request arrives at $a_{i,0}$ at time $b + i\delta - \epsilon$.
- A request arrives at $a_{i,1}$ at time $b + i\delta$.
- A request arrives at $a_{i,2}$ at time $b + i\delta + \epsilon$.

For b MOD $3 \equiv 1$: For each $i \in \{0, 1, \ldots, m-1\}$:

- A request arrives at $a_{i,0}$ at time $b + i\delta + \epsilon$.
- A request arrives at $a_{i,1}$ at time $b + i\delta - \epsilon$.
- A request arrives at $a_{i,2}$ at time $b + i\delta$.

For b MOD $3 \equiv 2$: For each $i \in \{0, 1, \ldots, m-1\}$:

- A request arrives at $a_{i,0}$ at time $b + i\delta$.
- A request arrives at $a_{i,1}$ at time $b + i\delta + \epsilon$.
- A request arrives at $a_{i,2}$ at time $b + i\delta - \epsilon$.

This sequence can continue for an arbitrary number of batches, say T. How does BATCH service the requests? During the second batch, it tries to service some of the requests that arrived during the first batch. Since these are all on points in B, BATCH will move to a point in B and in the best case, service all the $r+1$ requests which arrived. We will show that in each subsequent batch, BATCH will remain on points in B and will get $r+1$ requests per batch. Note that in each batch, exactly one request arrives on each point in A and B. If BATCH does not travel to set A, then it can get all $r+1$ requests which arrive in B. If it does travel to A, then it will spend 2β time in transit from B to A. That means it has a remaining 2β time to go from point to point (either in A or B). Since each request is point is δ away from every other point, it can get to at most $(1-2\beta)/\delta = r$ requests. Since the first option yields more requests satisfied, BATCH will stay in B and service the $r+1$ requests. This amounts to a total of $T(r+1)$ total requests satisfied over the course of the entire sequence.

The optimal algorithm, on the other hand, will forgo any requests that arrived in the first batch. Then for batch $b \geq 2$, it will visit each node $a_{0,j}, a_{1,j}, \ldots, a_{m-1,j}$ in turn where $j = b$ MOD 3. It arrives at node $a_{i,j}$ at time $b + i\delta$ where there are three requests waiting: one that arrived at time $(b-2)+i\delta+\epsilon$ (and expires at $b + i\delta + \epsilon$), one that arrived at time $(b-1)+i\delta$ (and expires at time $(b+1)+i\delta$) and one that arrived at time $b + i\delta - \epsilon$. Thus, after the third batch, the optimal algorithm will get $3m$ requests per batch. It will get $2m$ in batch 3 and m in batch 2. Thus, the total

number of requests satisfied will be $3m(T-2)$. Now pick constants d and $c < 3/(1-2\beta)$. Say $c = [3/(1-2\beta)]-\gamma$. We will show that T can be picked large enough and δ small enough so that

$$\begin{aligned} c \cdot B_{BATCH}(\sigma) + d &\leq B_{OPT}(\sigma) \\ \left[\frac{3}{1-2\beta} - \gamma\right] T(r+1) + d &\leq 3m(T-2) \\ \left[\frac{3}{1-2\beta} - \gamma\right] + \frac{d}{T\left(\frac{1-2\beta}{\delta}+1\right)} &\leq \frac{\frac{3}{\delta}(T-2)}{T\left(\frac{1-2\beta}{\delta}+1\right)} \\ \left[\frac{3}{1-2\beta} - \gamma\right] + \frac{d}{T\left(\frac{1-2\beta}{\delta}+1\right)} &\leq \left(\frac{3\left(1-\frac{2}{T}\right)}{1-2\beta+\delta}\right) \end{aligned}$$

As T grows and δ shrinks, the right hand side gets arbitrarily close to $3/(1-2\beta)$ and the left hand side gets arbitrarily close to $[3/(1-2\beta)]-\gamma$, so the inequality will hold. □

5 An Upper Bound for Double-Gain

THEOREM 5.1. *The competitive ratio of Double-Gain is at most*

$$4\left\lceil\frac{2}{1-\beta}\right\rceil\left(\left\lceil\frac{2}{1-\beta}\right\rceil+1\right).$$

Proof. Fix an input sequence σ. Partition time into periods of length $\phi = (1-\beta)/2$. The i^{th} period ends at time ϕi. At the end of each period, we will consider the number of outstanding requests in the Double-Gain's current route. Recall that at time t, the set of outstanding requests on the current route is denoted by $C(t)$. We will prove two facts.

1. $2\left\lceil\frac{1}{\phi}\right\rceil B_{DG}(\sigma) \geq \sum_i |C(\phi i)|$.
2. $B_{OPT}(\sigma) \leq 2\left\lceil\frac{1+\phi}{\phi}\right\rceil \sum_i |C(\phi i)|$.

We start by proving the first fact. For each request that ever appears in Double-Gain's current route, consider a continuous interval of time in which it is in the current route. (Some requests may appear in the algorithm's current route, drop out and then reappear in the algorithm's current route. This is treated as two separate intervals). Since every request expires or is served within time 1 of its arrival, each interval can cover at most $\lceil 1/\phi \rceil$ interval endings. This means that

$$\sum_i |C(\phi i)| \leq \lceil 1/\phi \rceil I, \tag{5.1}$$

where I is the total number of intervals.

An interval ends either because the request was satisfied or because the algorithm chose a new route and drops the request from its current route. We call the former *satisfied* intervals and the latter *unsatisfied* intervals. Consider a point in time when the algorithm chooses a new route. Suppose that x unsatisfied intervals end at this moment. Then it must be the case that at least $2x$ intervals begin at this moment as well. (This is the requirement for choosing a new route). Thus, we can associate two intervals which begin at time t with each unsatisfied interval that ends at time t in such a way that no interval is associated with more than one preceding interval. Note that some intervals that begin at time t may not be associated with any preceding interval.

Now we will assign a weight to each interval. We start with each interval having a weight of one. We will move a pointer from left to right, starting with the leftmost left endpoint and ending with the rightmost right endpoint. The location of the pointer represents a point in time. As the pointer moves, we will maintain two properties. The first is that the weight of each interval is less than 2. The second is that any interval whose left endpoint is to the right of the pointer has a weight of 1.

Suppose that the pointer reaches the right endpoint of an unsatisfied interval $\mathcal{I}$ whose weight is w. Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be the two intervals associated with $\mathcal{I}$. We know that the weights of $\mathcal{I}_1$ and $\mathcal{I}_2$ are at most 1 since until now their left endpoints have been to the right of the pointer. Set the weight of $\mathcal{I}$ to 0 and add $w/2$ to the weight of each of $\mathcal{I}_1$ and $\mathcal{I}_2$. Since $w < 2$, the new weight of these intervals is $1 + w/2 < 2$. This ensures that property one is maintained. No interval whose left endpoint is to the right of the pointer has changed weight, so the second property still holds as well.

At the end of this process, the total weight which is equal to I, the number of intervals, has remained unchanged. The weight of each unsatisfied interval is 0. The weight of all intervals is at most 2. This means that the number of satisfied intervals is at least $I/2$. Since the number of satisfied intervals is exactly the benefit of Double-Gain, we have that

$$2B_{DG}(\sigma) \geq I.$$

Putting this together with Inequality 5.1, implies the first Fact.

Now we will establish Fact 2. We will bound the number of requests which the optimal can serve that arrive in a period of length ϕ. Suppose the period starts at time $t-\phi$ and lasts through time t. In determining $N(t)$, one possible option for Double-Gain is to consider only the jobs which arrive in the interval from $t-\phi$ to time t, spend time β moving to the optimal starting location and spend time ϕ serving as many requests

that can be served in an interval of length ϕ. Note that $2\phi + \beta = 1$, so none of the requests which arrived in the interval from $t-\phi$ to time t will have expired. Let N be the set of jobs that would be satisfied by this course of action. Note that $|N(t)| \geq |N|$. If Double-Gain does not change routes at time t, then $2|C(t) - N(t)| > |N(t) - C(t)|$ which implies that $2|C(t)| \geq |N(t)| \geq |N|$. If Double-Gain does change routes at time t, then the new route for time t becomes the current route for time t. This means that $|C(t)|$ is the maximum number of future requests that Double-Gain can satisfy among all outstanding requests, given its current location. Since N is the set of future requests which can be satisfied using a specific route, $|C(t)| \geq |N|$. In either case, we have that $2|C(t)|$ is at least the number of request gained by the optimal algorithm which is allowed to choose any set of requests which arrive in the interval from $t - \phi$ to time t, relocate to an optimal location and server as many requests which can be served in an interval of length ϕ.

Any job which arrives during the interval $[t-\phi, t)$ must be scheduled in the interval $[t-\phi, t+1)$. We have established that during any subinterval of $[t-\phi, t+1)$ of length ϕ, the optimal algorithm can serve at most $2|C(t)|$ of the requests which arrive during $[t-\phi, t)$. This means that at most $2|C(t)|\lceil(1+\phi)/\phi\rceil$ requests which arrive during the interval $[t-\phi, t)$ can be scheduled by the optimal algorithm at any time. Using ϕi for t, we get that the optimal can satisfy at most

$$2|C(\phi i)| \left\lceil \frac{1+\phi}{\phi} \right\rceil$$

requests which arrive in period i. This means that the adversary can satisfy at most

$$2 \left\lceil \frac{1+\phi}{\phi} \right\rceil \sum_i |C(\phi i)|$$

requests total which establishes Fact 2. □

6 General Lower Bounds

The following theorem shows that α (the notification deadline) must be at least β in order for a competitive algorithm to exist. This proof requires arbitrarily small service times.

THEOREM 6.1. *If $\alpha \leq \beta$, then there is a metric space on which no deterministic online algorithm can achieve a bounded competitive ratio.*

Proof. Let c be an arbitrary constant. The metric space will consist of many points, all β away from each other. At time $i \cdot \alpha$, c^i requests will be requested at a new unused point in the metric space. This starts with $i = 0$ and ends either when the algorithm decides not to visit a requested point or when $i > (1-\beta)/(\beta-\alpha)$. Let's say that the last group arrives at time $k\alpha$.

Consider two cases:

- $k \leq (1-\beta)/(\beta-\alpha)$: In this case, the adversary will serve all the requests. Note that this is possible since the server will arrive at the i^{th} point at time $\beta(i+1)$ and the requests arrived at time αi. Since $k \leq (1-\beta)/(\beta-\alpha)$, we know that $\beta(k+1) \leq \alpha k + 1$ and the requests will not have expired. Thus, the optimal will serve $1 + c + c^2 + \cdots c^k = (c^{k+1}-1)/(c-1)$ requests and the online algorithm will only serve $1 + c + c^2 + \cdots c^{k-1} = (c^k - 1)/(c-1)$ requests. This gives a ratio of $(c^{k+1}-1)/(c^k-1) \geq c - 1$.

- $k > (1-\beta)/(\beta-\alpha)$: In this case, the online algorithm can not make it to the last set. Note that the algorithm must decide whether or not to satisfy each set of requests before the next set arrives. By the time the last set has arrived at time αk, it has committed to travel a total of βk. If it commits to the last set, it will have to travel $\beta(k+1)$, which it can not do by time $1 + \alpha k$ since $k > (1-\beta)/(\beta-\alpha)$. The adversary will skip the first one and get the rest, $c + c^2 + \cdots + c^k$. The algorithm get's $1 + c^2 + \cdots + c^{k-1}$. This results in a competitive ratio of c.

Since c was chosen to be an arbitrary constant, the competitive ratio is unbounded.

□

The remaining three lower bounds hold even when $\alpha = 1$ which means that the online algorithm does not need to notify a request as to whether it will be serviced.

THEOREM 6.2. *There is a metric space on which no deterministic algorithm can achieve a better competitive ratio than $1/(1-\beta)$.*

Proof. Consider a metric space with one point A separated by a distance β from a cluster of points B. The points in B are all a distance of δ from each other. There are $1/\delta$ points in the cluster. δ will always be chosen so that $1/\delta$ is an integer and can be chosen to be arbitrarily large.

We will prove the lower bound by imagining an adversary which plays against an arbitrary online algorithm A. The adversary will concoct a sequence of arrivals such that the number of requests which A satisfies will be at most $1/(1-\beta)$ times the number of jobs which the adversary can satisfy on the same sequence.

Both algorithms (A and the adversary) start with their server on an arbitrary point in B. The arrival

sequence is as follows: a single request arrives at A every interval of length $1+\epsilon$. There are two cases:

The algorithm decides to service the k^{th} request, for some k: Let's say the k^{th} request arrived at time t. As soon as the algorithm arrives at A, a request arrives on each point in B. The algorithm must spent β time reaching B again and has only time to service $(1/\delta)(1-\beta)$ of the requests. (Recall, that they are each δ apart from each other). The adversary, on the other hand, remains at B and can service all $1/\delta$ requests.

The algorithm never goes to A: The adversary goes to A at the beginning to the sequence. The adversary services all the requests and the algorithm does not service any requests.

□

The following lower bound shows that the bound proven for Double-Gain is within a constant factor of the optimal competitive ratio.

THEOREM 6.3. *When $\beta \geq 1/2$, there is a metric space on which no deterministic algorithm can achieve a better competitive ratio than*

$$\frac{\beta}{1-\beta}\left\lceil\frac{1}{1-\beta}\right\rceil.$$

Proof. The metric space will have $n = mk$ points which will consist of m groups, each with k points. Any two points from the same group are a distance of $1-\beta$ apart. Any two points from different groups are β apart.

Every interval of length $1-\beta$, there will be an *arrival event* in which a request will arrive on each point in the metric space, with the following exceptions. If the online server is located at a point in a particular group or moving between two points within the same group at the time of an arrival event, then no request will arrive at any point within the group at that time. If the online server is moving between two points from different groups at the time of an arrival event, then no request will arrive on any of the points in the group that the server is moving away from or the group that the server is moving towards. In these cases, we say that a group has been *skipped*.

Once the online server reaches a group, it will only be able to serve one request. This is because no requests will have arrived on points in that group during the interval of β in which the server is moving towards the group. There is exactly one arrival event in the $1-\beta$ time preceding the time the server starts moving towards the group. All previous requests will have expired by the time the server reaches the group. The server will serve one request upon its arrival. By the time it travels the $1-\beta$ to reach another point in the group, all outstanding requests in the group will have expired. Thus, the online server serves at most one request every interval of length β.

The optimal server will stay at a group until it is skipped. That is, it will stay through the last arrival event before the group is skipped and then go to another group. The server will plan its departure so that it arrives at the new group just after an arrival event. Thus, the server will be in transit between groups for $\lceil\beta/(1-\beta)\rceil$ arrival events.

Let's say that the server leaves a group at time t. It will select a new group to visit which has was not skipped in the arrival event preceding time t and will be the last group to be skipped after time t. Since at most two groups are skipped every arrival event, the server will find a group at which it will stay for $(m-2)/2$ arrival events after time t. Thus, the optimal server will be serving requests for $(m-2)/2 - \lceil\beta/(1-\beta)\rceil$ arrival events every $(m-2)/2$ arrival events.

The optimal moves from point to point within a group every $1-\beta$, arriving at each point just after an arrival event. When it arrives at a point, it can serve all the $\lceil 1/(1-\beta)\rceil$ requests which are waiting at that point and have not yet expired. Then it moves to the another point in the group and serves all the requests waiting at that point. As long as k, the number of points in a group is at least $\lceil 1/(1-\beta)\rceil$, it can serve $\lceil 1/(1-\beta)\rceil$ requests every arrival event that it is not in transit between groups. Thus, the rate at which it serves requests is

$$\frac{1}{1-\beta}\left\lceil\frac{1}{1-\beta}\right\rceil\left(1-\frac{2\beta}{(1-\beta)(m-2)}\right).$$

As m gets large, this number gets arbitrarily close to

$$\frac{1}{1-\beta}\left\lceil\frac{1}{1-\beta}\right\rceil.$$

Putting this together with the upper bound on the rate of service for the online algorithm, we have that no algorithm can have a competitive ratio better than

$$\frac{\beta}{1-\beta}\left\lceil\frac{1}{1-\beta}\right\rceil.$$

□

The following lower bound gives a stronger bound for the case where β is small.

THEOREM 6.4. *No deterministic online algorithm can obtain a competitive ratio less than* 2.

Proof. Let $N = 1 + \lceil 1/\beta \rceil$. Consider a uniform metric space with $6N$ points, each spaced β apart. The adversary will have an arrival event every interval of length β. The i^{th} arrival event takes place at time βi. At the i^{th} arrival event, a request will arrive at node $i \pmod{3N}$ and $3N + (i \pmod{3N})$. The time interval between two requests arriving at the same node is $3N\beta$ which is greater than 3.

Consider the i^{th} arrival event. Both of the requests for this arrival event expire at time $1+\beta i$. The adversary will plan on satisfying exactly one of these requests for this arrival event at time $1+\beta i-\epsilon$, where $\epsilon << \beta$. Note that the online algorithm can be at only one of these two nodes during the interval $[1+\beta(i-1/2), 1+\beta(i+1/2))$. The adversary will choose to satisfy the request at the node where the online algorithm does not have a server during this interval. Furthermore, the adversary will arrange to have a request arrive at that node at time $1 + \beta i - \epsilon$. This ensures that the adversary can satisfy two requests every interval of length β.

There is only ever one active request at each node at any time, except for the nodes where the adversary is currently serving a request. For these exceptions, there are two active requests for an interval of ϵ. We have determined that the online algorithm is not at these nodes for an interval of length β which covers these intervals of overlap. Furthermore, except for these overlapping requests, each request expires at least one time unit before the next one arrives. Since the travel time between nodes is β, the algorithm can only satisfy a request every interval of length β. □

7 Conclusions

Naturally, it is the goal of this research to find the optimal competitive ratio which can be obtained on any metric space as a function of α and β. We believe that a stronger upper bound can be obtained for the algorithm Double-Gain. It would also be interesting to extend the bound for Double-Gain to the case of non-uniform job windows. Another natural algorithm to consider is the BATCH algorithm with insertions, where the algorithm can service a newly arrived request or a request which is about to expire if it can be handled without having to drop any of the requests in the current BATCH path.

References

[1] B. Awerbuch, Y. Azar, A. Blum, S. Vempala. New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen. *SIAM Journal on Computing*, vol.28, (no.1), SIAM, 1998. p.254-62.

[2] G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie, M. Talamo. Algorithms for the On-line Traveling Salesman. *Algorithmica*, vol. 29 (no. 4), April 2001, pages 560-81.

[3] D. Bertsimas and G. Van Ryzin. Stochastic and Dynamic Vehicle Routing in the Euclidean Plane with Multiple Capacitated Vehicles. *Operations Research* 41(1), 60 – 76.

[4] D. Bertsiman and G. Van Ryzin. A Stochastic and Dynamic Vehicle Routing Problem in the Euclidean Plane. *Operations Research* 39(4), 601 – 615.

[5] X. Deng and C. Papadimitriou Exploring an unknown graph. *Proceedings. 31st Annual Symposium on Foundations of Computer Science*, 1990. p.355-61.

[6] P. Jaillet. A priori Solution of a Traveling Salesman Problem in Which a Random Subset of the Customers Are Visited. *Operations Research*, 36, 929-936 (1988).

[7] M. Manasse, L.A. McGeogh, and D. Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11, 208-230, 1990.

[8] W. Powell, P. Jaillet and A. Odoni. Stochastic and dynamic networks and routing. *Network Routing*, M. Ball, T. Maganti, C. Monma and G. Nemhauser (eds.). North-Holland, Amsterdam (1995).

[9] H. Psarafits. Dynamic Vehicle Routing Problems. *Vehicle Routing: Methods and Studies*, B.L. Golden and A.A. Assad (eds.). North-Holland, Amsterdam (1988).