The Online Transportation Problem

Bala Kalyanasundaram[*] *Kirk R. Pruhs*[†]

Abstract

We study the online transportation problem under the assumption that the adversary has only half as many servers at each site as the online algorithm. We show that the greedy algorithm is $\Theta(\min(m, \log C))$-competitive under this assumption, where m is the number of server sites, and C is the total number of servers. We then present an algorithm BALANCE, which is a simple modification of the greedy algorithm, that is $O(1)$-competitive under this assumption.

1 Introduction

We consider the natural online version of the well-known transportation problem [2, 5]. The initial setting consists of a collection $S = \{s_1, \ldots, s_m\}$ of server sites in a metric space $\mathcal{M}$. Each server site s_j has a positive integral capacity c_j. The online algorithm $\mathcal{A}$ sees over time a sequence $R = \{r_1, \ldots r_n\}$ of requests for service, with each request being a point in $\mathcal{M}$. In response to the request r_i, $\mathcal{A}$ must select a site $s_{\sigma(i)}$ to service r_i. The cost for this assignment is the distance $d(s_{\sigma(i)}, r_i)$ in the metric space between $s_{\sigma(i)}$ and r_i. Each site s_j can service at most c_j requests. The dilemma faced by the online algorithm $\mathcal{A}$ is that, at the time of the request r_i, $\mathcal{A}$ is not aware of the location of the future requests. The goal for the online algorithm is to minimize $\frac{1}{n}\sum_{i=1}^{n} d(s_{\sigma(i)}, r_i)$, the *average cost* to service the requests. Note that this is equivalent to minimizing the *total cost* $\sum_{i=1}^{n} d(s_{\sigma(i)}, r_i)$.

For concreteness, consider the following two examples of online transportation problems. In the *fire station problem*, the site s_j is a fire station that contains c_j fire crews. Each request is a fire that must be handled by a fire crew. The problem is to assign the crews to the fire so as to minimize the average distance traveled to get to a fire. In the *school assignment problem*, the site s_j is a school that can has a capacity of c_j students. Each request is a new student who moves into the school district. The problem is to assign the children to a

[*] *kalyan@cs.pitt.edu*, Computer Science Dept., University of Pittsburgh, Pittsburgh, PA 15260, Supported in part by NSF under grant CCR-9202158.

[†] *kirk@cs.pitt.edu*, Computer Science Dept., University of Pittsburgh, Pittsburgh, PA 15260, Supported in part by NSF under grant CCR-9209283.

school so as to minimize the average distance traveled by the children to reach their schools.

The standard measure of "goodness" of an online algorithm is the competitive ratio. For the online transportation problem, the competitive ratio for an online algorithm $\mathcal{A}$ is the supremum over all possible instances I, of $\mathcal{A}(I)/OPT(I)$, where $\mathcal{A}(I)$ is the total cost of the assignment made by $\mathcal{A}$, and $OPT(I)$ is the total cost of the minimum cost assignment for instance I. The standard way to interpret the competitive ratio is as a payoff of a game played by the online algorithm $\mathcal{A}$ against an all powerful adversary that specifies the requests, and services them in the optimal way. Note that the instance I specifies the metric space as well as the values of each s_j, c_j, and r_i.

In [1, 3] the online assignment problem, a special case of online transportation in which each capacity $c_i = 1$, was studied. In [1], it was shown that the competitive ratio of the intuitively appealing greedy algorithm, which assigns the nearest available server site to the new request, has a competitive ratio of $2^m - 1$. In [1, 3], it was shown that the optimal deterministic competitive ratio is $2m - 1$. The algorithm that achieves this competitive ratio requires a shortest augmenting path computation for each request. These results illustrate some shortfalls of using competitive analysis, namely:

- The achievable competitive ratios often grow quickly with input size, and would seem to overly pessimistic for "normal" inputs.
- The algorithm that achieves the optimal competitive ratio is often unnecessarily complicated for "normal" inputs.
- The poor competitive ratio of an intuitive greedy algorithm may not reflect the fact that it may perform reasonably well on "normal" inputs.

In situations where competitive analysis suffers such shortcomings, it is important to find alternate ways to to identify online algorithms that would work well in practice. In this paper, we adopt a modified version of competitive analysis that we call the *weak adversary model.* Generally speaking, in this model the adversary is given slightly less resources than the online algorithm. The intuition is that for "normal" inputs, one might expect that the performance of an *offline* algorithm would not degrade significantly if its resources were slightly reduced. Hence, if we can prove that an online algorithm is competitive against an adversary with slightly less resources, then one might argue that the online algorithm will be competitive against an equivalently equipped offline algorithm on "normal" inputs. One can also view this weakening of adversary as measuring the additional resources required by the online algorithm to offset the decrease in performance due to the online nature of the problem.

In the case of the transportation problem we compare the online algorithm with c_i servers at s_i to an offline line algorithm with $a_i = c_i/2$ servers at s_i (we assume that c_i is even). Given an instance I of the online transportation problem with n requests, $n \leq \sum_{i=1}^{m} a_i$, we let I' be the same instance with each capacity c_i replaced by a_i. We then say the *halfopt-competitive ratio* of an online algorithm $\mathcal{A}$ is the supremum over all instances I, with $n \leq \sum_{i=1}^{m} a_i$ requests, of the ratio $\mathcal{A}(I)/OPT(I')$. We assume that the online algorithm has twice as many servers as the adversary because this is the least advantage that we can give to the online algorithm without annulling our analysis techniques.

In this paper we present the following results. In section 3, we show that the halfopt-competitive ratio of the greedy algorithm is $\Theta(\min(m, \log C))$, where $C = \sum_{i=1}^{m} c_i$ is the sum of the capacities. If the server capacity of each site is constant, then the halfopt-competitive ratio is logarithmic in m, a significant improvement over the exponential bound on the traditional competitive ratio. In section 4, we describe the algorithm Balance, which is a simple modification of the greedy algorithm, and has a halfopt-competitive ratio that is $O(1)$. Recall that the traditional competitive ratio of every deterministic online algorithm is $\Omega(m)$.

We now summarize related results. The weakened adversary model was introduced in [6] in the context of studying paging. This model has also been used to study variants of the k-server problem, a generalization of the paging problem (see for example [8]). References to other other suggested variants of competitive analysis can be found in [4]. Further ancillary results on online assignment, which are not directly related to the results in this paper, can be found in [1]. In [7], the average competitive ratio for the greedy algorithm in the online assignment problem is studied under the assumption that the metric space is the Euclidean plane and the points are uniformly distributed in a unit square. The offline transportation problem can be solved in polynomial-time[2, 5].

2 Preliminaries

In this section we introduce some definitions, facts, and concepts that are common to the remaining sections. We generally begin by assuming the simplifying condition that the online capacity c_i of each server site is two. We will think of s_i as containing two *online servers* s_i^1 and s_i^2 that move to service requests. We also think of s_i as containing one *adversary server* s_i^a. We assume, without loss of generality, that the adversary services request r_i with s_i^a. We use $s_{\sigma(i)}$ to denote the site that the online algorithm uses to service request r_i. We define a weighted bipartite graph $\mathcal{G} = (S \cup R, E)$, which we call the *response graph*, by including an *online edge* $(r_i, s_{\sigma(i)})$, and an *adversary edge* (r_i, s_i) for each request r_i. The weight of each edge (r_i, s_j) in $\mathcal{G}$ is the distance $d(r_i, s_j)$ between the r_i and s_j in the underlying metric space. In figure 1, the online edges are the solid edges, the adversary edges are the dashed edges, the server sites are the filled circles, and the requests are the question marks. The notation shown in figure 1 will be used throughout this paper.

Lemma 1 Assume r_i is request vertex that is in a cycle in $\mathcal{G}$. Let $\mathcal{T}$ be the connected component of $\mathcal{G} - (s_{\sigma(i)}, r_i)$ that contains r_i. Then $\mathcal{T}$ is a tree.

Proof: Consider a breadth first search of $\mathcal{T}$ starting from r_i. Such a search divides $\mathcal{T}$ into levels $L_1, \ldots, L_b$, where the vertices in level L_k are k hops from

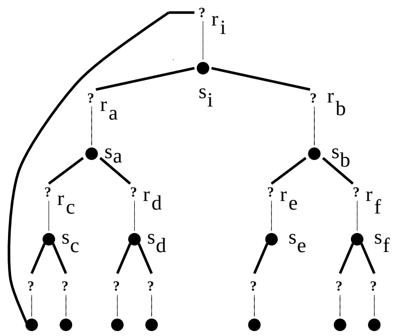

Figure 1: An Example Connected Component of the Response Graph

r_i. The vertices in the odd levels are server vertices, and the vertices in even levels are request vertices. Edges that go from an even level to an odd level are adversary edges, and edges that go from an odd level to an even level are online edges. Assume to reach a contradiction that C is a cycle in $\mathcal{T}$. Let y be the vertex in C that is in the highest level L_c (i.e., largest c), and let x and z be the two vertices adjacent to y in C. Note that it may be the case that $x = z$. Now it must be the case that x and z are in L_{c-1} or we would get a contradiction to one of the bipartitieness of $\mathcal{G}$, the definition of x, or the definitions of the levels. If c is odd, then we get a contradiction to the fact that the adversary has only one server per site. If c is even, we get a contradiction to the fact that the online algorithm only uses one server to service each request. ■

Let $\mathcal{T}$ be a tree as described in lemma 1. If we root the tree $\mathcal{T}$ at r_i then $\mathcal{T}$ has the following structure. For each request $r_j \in \mathcal{T}$, the one child of r_j is s_j. If r_j is not the root, then the server site $s_{\sigma(j)}$ is the parent of r_j in $\mathcal{T}$. The leaves of $\mathcal{T}$ are server sites with no incident online edges. We denote the total cost of the adversary edges in a tree $\mathcal{T}$ by $OPT(\mathcal{T})$, that is, $OPT(\mathcal{T}) = \sum_{(r_i, s_i) \in \mathcal{T}} d(r_i, s_i)$. Analogously, we define $ON(\mathcal{T}) = \sum_{r_i \in \mathcal{T}} d(r_i, s_{\sigma(i)})$. Note that $ON(\mathcal{T})$ includes the cost of the online edge incident to the root of $\mathcal{T}$, even though this edge is not in $\mathcal{T}$. For a vertex $x \in \mathcal{T}$, we define the leaf distance $ld(x)$ to be the minimum over all leaves s_j in $\mathcal{T}$ of the distance between x and s_j. If a server at site s_j serviced the root r_i of $\mathcal{T}$, then $ld(s_j) = d(s_j, r_i) + ld(r_i)$. If x is a node in $\mathcal{T}$, we define $\mathcal{T}(x)$ to the the subtree of $\mathcal{T}$ rooted at x.

In this paper, log means the logarithm base 2.

3 Analysis of the Greedy Algorithm

We begin with the upper bound on the competitive ratio for the algorithm Greedy, which uses the nearest available server to service each request. We first assume that the online capacity of each server site is two, and then show how to extend the proof to the general case.

Theorem 2 The halfopt-competitive ratio of GREEDY for online transportation, with $m \geq 2$ sites and two online servers per site, is at most $2 \log m$.

In order to prove this theorem, we will divide the response graph $\mathcal{G}$ into edge disjoint rooted trees, $\mathcal{T}^1, \ldots, \mathcal{T}^l$. For each such tree, we will establish the competitive bound independently.

Our construction yields trees ($\mathcal{T}^j$'s) that satisfy the following *tree invariants*:

1. Each nonleaf server site s_i in $\mathcal{T}^j$ has two incident online edges in $\mathcal{T}^j$.
2. Each leaf of $\mathcal{T}^j$ is a server site that had an unused server at the time of each request in $\mathcal{T}^j$.

Using the following iterative procedure to construct the trees.

Tree Construction Procedure: Assume that trees $\mathcal{T}^1, \ldots, \mathcal{T}^{j-1}$ have been constructed. We explain how to construct $\mathcal{T}^j$ in our next iteration. During this construction we will modify $\mathcal{G}$. The root of $\mathcal{T}^j$ is the most latest request $r_{\alpha(j)}$ not included in a previous tree $\mathcal{T}^1, \ldots, \mathcal{T}^{j-1}$. The online edge incident to $r_{\alpha(j)}$ is removed from $\mathcal{G}$. Let L be the collection of server vertices s_i such that s_i is reachable from $r_{\alpha(j)}$, and s_i currently has at most one incident online edge in $\mathcal{G}$. Note that an $s_i \in L$ might have originally had two incident online edges if one or both of these edge lead to the root of one of the trees $\mathcal{T}^1, \ldots, \mathcal{T}^j$. Now let $\mathcal{T}^j$ be the edges on paths in $\mathcal{G}$ from $r_{\alpha(j)}$ to the server vertices in L. Note that by lemma 1 there is a unique path from $r_{\alpha(j)}$ to each vertex in L. It is not hard to see that $\mathcal{T}^j$ satisfies the tree invariants. The edges and request vertices in $\mathcal{T}^j$ are then removed from $\mathcal{G}$, and we proceed to our next iteration to construct $\mathcal{T}^{j+1}$ if $\mathcal{G}$ still contains edges.

We now fix a particular tree, say $\mathcal{T} = \mathcal{T}^j$, and for simplicity drop the j superscript.

Lemma 3 For each request $r_i \in \mathcal{T}$, $ld(r_i) \leq OPT(\mathcal{T}(r_i))$, and $d(s_{\sigma(i)}, r_i) \leq OPT(\mathcal{T}(r_i))$.

Proof: The proof is by induction on the number k of request nodes in the induced tree $\mathcal{T}(r_i)$. If $k = 1$ then the child s_i of r_i is a leaf, and $\mathcal{T}(r_i)$ consists of one adversary edge (r_i, s_i) with $d(r_i, s_i) = ld(r_i) = OPT(\mathcal{T}(r_i))$. That $d(s_{\sigma(i)}, r_i) \leq OPT(\mathcal{T}(r_i))$ follows by the definition of GREEDY.

Now suppose $k > 1$. If s_i is not a leaf then it has two children in $\mathcal{T}(r_i)$, say r_a and r_b (see figure 1). By induction, it must be the case that $ld(r_a) \leq OPT(\mathcal{T}(r_a))$, $d(s_i, r_a) \leq OPT(\mathcal{T}(r_a))$, $ld(r_b) \leq OPT(\mathcal{T}(r_b))$, and $d(s_i, r_b) \leq OPT(\mathcal{T}(r_b))$. Therefore by induction and the triangle inequality,

$$
\begin{aligned}
ld(r_i) &\le d(r_i, s_i) + \min[d(s_i, r_a) + ld(r_a), d(s_i, r_b) + ld(r_b)] \\
&\le d(r_i, s_i) + \min[d(s_i, r_a) + OPT(\mathcal{T}(r_a)), d(s_i, r_b) + OPT(\mathcal{T}(r_b))] \\
&\le d(r_i, s_i) + \min[2 \cdot OPT(\mathcal{T}(r_a)), 2 \cdot OPT(\mathcal{T}(r_b))] \qquad (1) \\
&\le d(r_i, s_i) + OPT(\mathcal{T}(r_a)) + OPT(\mathcal{T}(r_b)) \\
&= OPT(\mathcal{T}(r_i))
\end{aligned}
$$

The fact that $d(s_{\sigma(i)}, r_i) \le OPT(\mathcal{T}(r_i))$ then follows since GREEDY would only assign $s_{\sigma(i)}$ to r_i if $d(s_{\sigma(i)}, r_i) \le ld(r_i)$. ■

Lemma 4 Let r_i be a request in $\mathcal{T}$, and let k be the number of request vertices in $\mathcal{T}(r_i)$. Then $ON(\mathcal{T}(r_i)) \le 2 \cdot \max(1, \log k) \cdot OPT(\mathcal{T}(r_i))$.

Proof: The proof is by induction on k. Observe that according to our tree construction, each server node has either two children or none. Therefore, k must be odd.

For $k = 1$, $ON(\mathcal{T}(r_i)) = d(s_{\sigma(i)}, r_i)$, which is less than $OPT(\mathcal{T}(r_i)) = d(r_i, s_i)$ by the definition of GREEDY.

Now consider the case $k = 3$. Let the two children of s_i in $\mathcal{T}(r_i)$ be r_a and r_b. Note that

$$ON(\mathcal{T}(r_i)) = d(s_{\sigma(i)}, r_i) + d(s_i, r_a) + d(s_i, r_b)$$

and that

$$OPT(\mathcal{T}(r_i)) = d(r_i, s_i) + d(s_a, r_a) + d(s_b, r_b)$$

By the definition of GREEDY, $d(s_i, r_a) \le d(s_a, r_a)$ and $d(s_i, r_b) \le d(s_b, r_b)$. By lemma 3, $d(s_{\sigma(i)}, r_i) \le OPT(\mathcal{T}(r_i))$. Hence, by substitution,

$$ON(\mathcal{T}(r_i)) \le OPT(\mathcal{T}(r_i)) + d(s_a, r_a) + d(s_b, r_b) \le 2 \cdot OPT(\mathcal{T}(r_i))$$

Now consider the case $k > 3$. Once again let the two children of s_i in $\mathcal{T}(r_i)$ be r_a and r_b. Notice that $ON(\mathcal{T}(r_i)) = ON(\mathcal{T}(r_a)) + ON(\mathcal{T}(r_b)) + d(s_{\sigma(i)}, r_i)$. By equation 1,

$$d(s_{\sigma(i)}, r_i) \le d(r_i, s_i) + 2 \cdot \min[OPT(\mathcal{T}(r_a)), OPT(\mathcal{T}(r_b))]$$

Hence,

$$
\begin{aligned}
ON(\mathcal{T}(r_i)) \le\ & ON(\mathcal{T}(r_a)) + ON(\mathcal{T}(r_b)) + \\
& d(r_i, s_i) + 2 \cdot \min[OPT(\mathcal{T}(r_a)), OPT(\mathcal{T}(r_b))] \qquad (2)
\end{aligned}
$$

Let $w = OPT(\mathcal{T}(r_a)) + OPT(\mathcal{T}(r_b))$, and $x = OPT(\mathcal{T}(r_a))$. Assume, without loss of generality, that $OPT(\mathcal{T}(r_a)) \le OPT(\mathcal{T}(r_b))$. Hence, $x \le w/2$. Also let y be the number of request nodes in $\mathcal{T}(r_a)$. We now break the proof into cases.

In the first case, we assume that both $\mathcal{T}(r_a)$ and $\mathcal{T}(r_b)$ consist of more than one request vertex. Hence, $3 \le y \le k - 4$. By induction $ON(\mathcal{T}(r_a)) \le 2x \log y$

and $ON(\mathcal{T}(r_b)) \leq 2(w-x)\log(k-1-y)$. Hence, by substituting into equation 2 we get

$$ON(\mathcal{T}(r_i)) \leq 2x\log y + 2(w-x)\log(k-1-y) + 2x + d(r_i, s_i)$$

Since $OPT(\mathcal{T}(r_i)) = w + d(r_i, s_i)$, in order to show $ON(\mathcal{T}(r_i)) \leq 2\log k \cdot OPT(\mathcal{T}(r_i))$ it is sufficient to show

$$2x\log y + 2(w-x)\log(k-1-y) + 2x + d(r_i, s_i) \leq 2(w + d(r_i, s_i))\log k$$

In turn it is sufficient to show that

$$x\log y + (w-x)\log(k-1-y) + x \leq w\log k \tag{3}$$

Let $f(x, y) = x\log y + (w-x)\log(k-1-y) + x$ be the left hand side of this inequality. Notice that $f(x, y)$ is linear in x. Hence the maximum of $f(x, y)$ must occur at the boundary, that is, at the point $x = 0$ or the point $x = w/2$. Inequality 3 follows immediately for $x = 0$. For $x = w/2$ we have to find the value of y that maximizes $f(w/2, y) = (w/2)(\log y + \log(k-1-y) + 1)$. The derivative of $f(w/2, y)$ with respect to y is $\frac{w}{2\ln 2}(\frac{1}{y} - \frac{1}{k-1-y}))$. Hence, one can see that the maximum $f(w/2, y)$ occurs at $y = (k-1)/2$. We now must show that

$$f(w/2, \frac{k-1}{2}) = w(\log\frac{k-1}{2} + 1) \leq w\log k$$

which by algebraic simplification is equivalent to $\log(k-1) \leq \log k$.

We now consider the case that $\mathcal{T}(r_a)$ contains only one request. So $y = 1$. By induction $ON(\mathcal{T}(r_a)) \leq 2x$ and $ON(\mathcal{T}(r_b)) \leq 2(w-x)\log(k-2)$. Hence, by substituting into equation 2 we get

$$ON(\mathcal{T}(r_i)) \leq 2x + 2(w-x)\log(k-2) + 2x + d(r_i, s_i)$$

In order to show that $ON(\mathcal{T}(r_i)) \leq 2\log k \cdot OPT(\mathcal{T}(r_i))$ it is sufficient to show

$$2x + 2(w-x)\log(k-2) + 2x \leq 2w\log k$$

Since the left hand side is linear in x, we need only consider $x = 0$ and $x = w/2$. The case $x = 0$ follows immediately. If $x = w/2$ we are left with showing $(w/2)\log(k-2) + w \leq w\log k$. This is equivalent to $\log(2\sqrt{k-2}) \leq \log k$, or $(2\sqrt{k-2}) \leq k$, which one can verify holds for $k \geq 3$.

We now consider the case that $\mathcal{T}(r_b)$ contains only one request. So $y = k-2$. By induction $ON(\mathcal{T}(r_a)) \leq 2x\log(k-2)$ and $ON(\mathcal{T}(r_b)) \leq 2(w-x)$. Hence, by substituting into equation 2 we get

$$ON(\mathcal{T}(r_i)) \leq 2x\log(k-2) + 2(w-x) + 2x + d(r_i, s_i)$$

In order to show that $ON(\mathcal{T}(r_i)) \leq 2\log k \cdot OPT(\mathcal{T}(r_i))$ it is sufficient to show that

$$x\log(k-2) + (w-x) + x \leq w\log k$$

Since the left hand side is linear in x, one need only verify that the inequality holds at the boundaries $x = 0$ and $x = w/2$. ■

Proof: (of Theorem 2.) Applying lemma 4 to each tree $\mathcal{T}^i$, we get the desired result. ∎

We now extend the result to the case that the online capacities are larger than two. Recall that $C = \sum_{i=1}^{m} c_i$ is the total online capacity.

Theorem 5 The halfopt-competitive ratio of GREEDY for online transportation is $O(\min(m, \log C))$.

Proof: The upper bound of $O(\log C)$ is immediate by theorem 2 if we conceptually split a server site with c_i online servers into $c_i/2$ sites with 2 arbitrary online servers and 1 arbitrary adversary server.

To see the $O(m)$ bound we need to be more careful about how we split the server sites up. Assume that the tree construction procedure just constructed a tree $\mathcal{T}^k$. We perform some pruning of sT^k, if necessary, before we proceed to construct $\mathcal{T}^{k+1}$. If no root-to-leaf path in $\mathcal{T}^k$ passes through two server sites that are at the same location, then the number of vertices in $\mathcal{T}^k$ is $O(2^m)$. Hence, the $O(m)$ bound follows from lemma 4.

If $\mathcal{T}^k$ contains root-to-leaf paths that pass through two server sites that are at the same location, we show how to modify $\mathcal{T}^k$ to remove such paths. Assume that $\mathcal{T}^k$ contains a root-to-leaf path that first passes through s_i and then passes through s_j, where s_i and s_j are at the same location. We modify $\mathcal{T}^k$ by making server vertex s_j the child of r_i in $\mathcal{T}^k$. See figure 2. Note that may remove edges and vertices originally below s_i from $\mathcal{T}^k$. We repeat this process until $\mathcal{T}^k$ has no root to leaf path passing through two server sites at the same location. Notice that the resulting tree $\mathcal{T}^k$ still satisfies the tree invariants. Now we start the construction of $\mathcal{T}^{k+1}$.

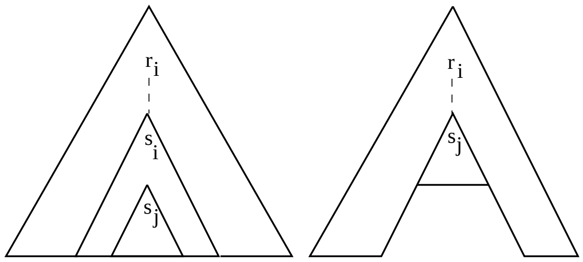

Figure 2: The original $\mathcal{T}^k$ on the left, and the new $\mathcal{T}^k$ on the right

∎

We now prove an asymptotically matching lower bound for the halfopt-competitive ratio for GREEDY.

Theorem 6 The halfopt-competitive ratio of GREEDY for the transportation problem is $\Omega(\min(m, \log C))$.

Proof: Assume without loss of generality that $m = \Theta(\log C)$. We embed m server sites on the real line. The server site s_1 is located at the point -1. The server site s_i $(2 \leq i \leq m)$ is located at $2^{i-1} - 1$. The online algorithm has $c_i = 2^{m-i+1}$ servers at s_i, while the adversary has $a_i = 2^{m-i}$ servers at s_i. Thus the online algorithm has a total of $C = 2^{m+1} - 2$ servers, while the adversary has a total of $A = 2^m - 1$ servers. The requests occur in m batches. The first batch consists of 2^{m-1} requests at the point 0. The ith batch $(2 \leq i \leq m)$ contains 2^{m-i} requests that occur at $2^{i-1} - 1$, the location of s_i. GREEDY responds to batch i $(i < m)$ by answering each request in batch i with a server at site s_{i+1}, thus depleting site s_{i+1}. GREEDY responds to the mth batch by moving one server from s_1. Thus the total online cost is $m2^{m-1}$. By using the servers in s_i to handle batch i $(1 \leq i \leq m)$ it is possible to obtain a total cost of 2^{m-1}. ■

4 The Algorithm Balance

In this section we present an algorithm, BALANCE, with a halfopt-competitive ratio of $O(1)$.

Algorithm BALANCE: At each site s_h we classify half of the servers as *primary* and half of the servers as *secondary*. Let $c > 5 + 4\sqrt{2}$ be some constant. Define the *pseudo-distance* from a request r_i to a primary server at site s_j to be $d(s_j, r_i)$, and the *pseudo-distance* from r_i to a secondary server at site s_j to be $c \cdot d(s_j, r_i)$. BALANCE services each request r_i with an arbitrary server with minimal pseudo-distance from r_i.

Our goal is now to show that the halfopt-competitive ratio of BALANCE for online transportation, with two online servers per site, is $O(1)$. We first break the response graph $\mathcal{G}$ into disjoint trees. Let $C_1, \ldots, C_l$ be the connected components of the response graph $\mathcal{G}$. By lemma 1 each connected component of the response graph contains a unique cycle. Let $r_{\alpha(i)}$ be the most recent request in the cycle in C_i. Let $\mathcal{T}^j$ be the tree that is C_j minus the online edge incident to $r_{\alpha(j)}$, and we set the root of $\mathcal{T}^j$ to be $r_{\alpha(i)}$. Each such tree $\mathcal{T}^j$ satisfies the following two tree invariants:

1. Each nonleaf server site $s_i \in \mathcal{T}^j$, with one incident online edge in $\mathcal{T}$, had the secondary server available just before the time of each request $r_h \in \mathcal{T}^j$.

2. Each leaf of $\mathcal{T}^j$ is a server site s_i that had both of its servers available just before the time of each request in $\mathcal{T}^j$.

We now fix a particular tree, say $\mathcal{T} = \mathcal{T}^j$, and for simplicity drop the superscript j. In order to show that the halfopt-competitive ratio of BALANCE is $O(1)$ it is sufficient to show that $ON(\mathcal{T}) = O(OPT(\mathcal{T}))$.

Definition 7 Let s_i be a generic server site in $\mathcal{T}$. We say that the *primary server child* s_a of s_i is the server site that the adversary used to service the request serviced by s_i^1, and the *secondary server child* s_b of s_i is the server site that the adversary used to service the request serviced by s_i^2. The *server parent* s_p of s_i is the server site used by BALANCE to service r_i. The site s_i is a *double* if it has two server children, and otherwise s_i is a *single*.

Lemma 8 If BALANCE uses a server at site s_p to handle a request $r_i \in \mathcal{T}$, then

$$d(s_p, r_i) \leq d(r_i, s_i) + ld(s_i)$$

and

$$ld(s_p) \leq 2(d(r_i, s_i) + ld(s_i)) \tag{4}$$

Proof: Observe that there is a path in $\mathcal{T}$ from r_i to some leaf s_k with total length at most $d(r_i, s_i) + ld(s_i)$. If $s_p \neq s_k$ then $d(s_p, r_i) \leq d(r_i, s_i) + ld(s_i)$ since BALANCE didn't use s_k to service r_i. If $s_p = s_k$ then $d(s_p, r_i) \leq d(r_i, s_i) + ld(s_i)$ by the triangle inequality. Now

$$ld(s_p) \leq d(s_p, r_i) + d(s_i, r_i) + ld(s_i) \leq 2(d(r_i, s_i) + ld(s_i))$$

∎

Lemma 9 Assume that BALANCE uses a secondary server s_p^2 to handle a request $r_i \in \mathcal{T}$ that is not the root of $\mathcal{T}$. Then

$$d(s_p, r_i) \leq (1/c)(d(r_i, s_i) + ld(s_i))$$

and

$$ld(s_p) \leq (1 + 1/c)(d(r_i, s_i) + ld(s_i))$$

Proof: Observe that there is a path in $\mathcal{T}$ from r_i to some leaf s_k with total length at most $d(r_i, s_i) + ld(s_i)$. Then $d(s_p, r_i) \leq (1/c)(d(r_i, s_i) + ld(s_i))$ since BALANCE didn't use the primary server at s_k to service r_i. Now

$$ld(s_p) \leq d(s_p, r_i) + d(s_i, r_i) + ld(s_i) \leq (1 + 1/c)(d(r_i, s_i) + ld(s_i))$$

∎

Definition 10 We let $\gamma = \frac{2c+2}{3c+1}$.

Fact 11 For all nonnegative reals x and y, and for all $c > 1$, $\min(2x, (1 + 1/c)y) \leq \gamma(x + y)$.

Proof: Suppose $2x \geq (1 + 1/c)y$. It suffices to show that $\min(2x, (1 + 1/c)y) = (1 + 1/c)y \leq \gamma(\frac{1}{2}(1 + 1/c)y + y)$. Simple algebra shows that $(1 + 1/c)y = \gamma(\frac{1}{2}(1 + 1/c)y + y)$

On the other hand, suppose $2x < (1 + 1/c)y$. It suffices to show that $\min(2x, (1 + 1/c)y) = 2x \leq \gamma(x + \frac{2c}{c+1}x)$. Simple algebra shows that $2x = \gamma(x + \frac{2c}{c+1}x)$. ∎

Lemma 12 Assume that s_i is a double server site with server children s_a and s_b. Then

$$ld(s_i) \leq \gamma(d(r_a, s_a) + d(r_b, s_b) + ld(s_a) + ld(s_b)) \tag{5}$$

Proof: Note that $ld(s_i) \leq 2(d(r_a, s_a) + ld(s_a))$ by lemma 8, and that $ld(s_i) \leq (1 + 1/c)(d(r_b, s_b) + ld(s_b))$ by lemma 9. The lemma then follows by fact 11. ■

Definition 13 A server site s_i is *local* if either s_i is a single, or one of s_i's server children is a single. Otherwise, s_i is *global*. For convenience, we call a request r_i *local* if s_i is local. Otherwise r_i is *global*. For a server site s_i we use $z(s_p)$ to denote $d(s_i, r_i) + d(s_a, r_a) + d(s_b, r_b)$.

We now break the accounting of the online edges in $\mathcal{T}$ into cases. We first show that for every local server s_i, the cost for BALANCE to serve r_i is $O(z(s_p))$.

Lemma 14

$$\sum_{\text{local } r_i} d(s_p, r_i) \leq (c^2 + c + 1)OPT(\mathcal{T})$$

Further for each parent server s_p of a local server s_i

$$ld(s_p) \leq (c^2 + 1)z(s_p) + ld(s_i) \tag{6}$$

Proof: We break the proof into two cases. In the first case assume that s_i is a single. It must be the case that $d(s_p, r_i) \leq c \cdot d(r_i, s_i)$ since BALANCE didn't use a server at s_i to handle r_i. Hence,

$$\sum_{\text{single local } s_i} d(s_p, r_i) \leq c \sum_{\text{single local } s_i} d(s_i, r_i) \leq cOPT(\mathcal{T})$$

Further

$$ld(s_p) \leq (c+1)d(s_i, r_i) + ld(s_i)$$

In the second case assume that s_i is double, and one of s_i's server children is a single. For simplicity assume that s_a is a single; one can verify that the following argument also holds if s_b is a single. Since s_a is a single we can apply the analysis from the previous case to get that $d(s_i, r_a) \leq c \cdot d(r_a, s_a)$. Since BALANCE serviced r_i with a server at s_p instead of the unused server at s_a we get that

$$d(s_p, r_i) \leq c \cdot d(s_a, r_i) \leq c\,[d(s_i, r_i) + d(s_i, r_a) + d(s_a, r_a)]$$

Hence, by substitution,

$$d(s_p, r_i) \leq c \cdot d(r_i, s_i) + (c^2 + 1)d(r_a, s_a) \tag{7}$$

Thus

$$\begin{aligned}
\sum_{\text{double local } s_i} d(s_p, r_i) &\leq c \sum_{\text{double local } s_i} d(r_i, s_i) + \\
&\quad (c^2+1) \sum_{\text{double local } s_i} d(r_a, s_a) \\
&\leq (c^2 + c + 1)OPT(\mathcal{T})
\end{aligned}$$

Furthermore,

$$\begin{aligned} ld(s_p) &\le d(s_p, r_i) + d(r_i, s_i) + ld(s_i) \\ &\le (c+1)d(r_i, s_i) + (c^2+1)d(s_a, r_a) + ld(s_i) \end{aligned}$$

The result then follows. ∎

Now we consider the online cost of servicing the global r_i's. Observe that by lemma 8 it is the case that

$$\sum_{\text{global } r_i \in \mathcal{T}} d(s_p, r_i) \le \sum_{\text{global } r_i \in \mathcal{T}} [d(s_i, r_i) + ld(s_i)] \le OPT(\mathcal{T}) + \sum_{\text{global } s_i \in \mathcal{T}} ld(s_i)$$

We now will show that

$$\sum_{\text{global } s_i \in \mathcal{T}} ld(s_i) = O(OPT(\mathcal{T}))$$

Definition 15 Let s_i be a global server site in $\mathcal{T}$. We define $S(s_i)$ to be the set of global server sites in the subtree of $\mathcal{T}$ rooted at s_i with the property that, for any server site $s_j \in S(s_i)$, there are no global server sites in the unique path from s_i to s_j in $\mathcal{T}$. We define $LC(s_i)$ to be the sum of the cost of the offline edges (s_j, r_j) in the subtree rooted at s_i with the property that the unique path from s_i to r_j in $\mathcal{T}$ does not pass through a global server site.

To give an alternative explanation of $LC(s_i)$ consider pruning $\mathcal{T}$ at the global server vertices, which results in a collection of trees rooted at global server vertices. Then $LC(s_i)$ is the total cost of the offline edges in the tree rooted at s_i.

Lemma 16 For any global server site s_i,

$$ld(s_i) \le 2(c^2+1)LC(s_i) + 2\gamma^2 \sum_{s_j \in S(s_i)} ld(s_j)$$

Proof: If the server children s_a and s_b of s_i are both global then

$$ld(s_i) \le \gamma(d(r_a, s_a) + d(r_b, s_b)) + \gamma(ld(s_a) + ld(s_b))$$

Hence, the result follows in this case since $\gamma \le 2\gamma^2 < 1$.

Now assume that at least one server child, say s_a, of s_i is local. Note that s_a and s_b must be a double, since s_i is global. Further assume for the moment that s_b of s_i is global. Let s_c and s_d be the two server children of s_a. Then using equation 5 we get that

$$\begin{aligned} ld(s_i) \le\ & \gamma(d(r_a, s_a) + d(r_b, s_b)) + \gamma ld(s_b) + \\ & \gamma^2(d(r_c, s_c) + d(r_d, s_d)) + \gamma^2(ld(s_c) + ld(s_d)) \end{aligned}$$

In this case we continue to expand $ld(s_c)$ and $ld(s_d)$ using the general expansion method described below. Now consider the case that both s_a and s_b are local. Let s_e and s_f be the server children of s_b. Then using equation 5 we get that

$$\begin{aligned} ld(s_i) \quad \leq \quad & \gamma(d(r_a, s_a) + d(r_b, s_b)) + \\ & \gamma^2(d(r_c, s_c) + d(r_d, s_d) + d(r_e, s_e) + d(r_f, s_f)) + \\ & \gamma^2(ld(s_c) + ld(s_d) + ld(s_e) + ld(s_f)) \end{aligned}$$

In this case we continue to expand $ld(s_c)$, $ld(s_d)$, $ld(s_e)$ and $ld(s_f)$ using the general expansion method described below.

We now describe the general expansion method. We expand each $ld(s_j)$ using equations 4 and 6. A term of the form $ld(s_j)$ is expanded according to the following rules:

1. If s_j is a leaf in $\mathcal{T}$ then $ld(s_j)$ is set to 0.
2. If s_j has a local server child s_k, then $ld(s_j)$ is set to $(c^2+1)z(s_j) + ld(s_k)$, which is valid by equation 6. Observe that since s_k is local it will be subsequently expanded.
3. If none of s_j's server children are local then $ld(s_j)$ is set to $2(d(r_k, s_k) + ld(s_k))$, where s_k is an arbitrary server child of s_k. This is valid by equation 4. Note that in this case the term $ld(s_k)$ is not expanded again since $s_k \in S(s_i)$.

The $d(s_x, r_x)$ terms that appear in this general expansion process are all included in $LC(s_i)$. Hence, we get

$$ld(s_i) \leq \alpha LC(s_i) + \beta \sum_{s_j \in S(s_i)} ld(s_j)$$

for some α and β. Since each offline edge appears in this general expansion at most twice, and in each case the coefficient is at most (c^2+1), we can conclude that $\alpha \leq 2(c^2+1)$. Note that in rule 2, the coefficient in front of the $ld(s_j)$ term before the expansion is the same as the coefficient in front of the $ld(s_k)$ term after expansion. Since the coefficient in front of each ld term is γ^2 before the application of any of the general expansion rules, and the only way that it can change is by application of rule 3, which is a terminal expansion, each coefficient on a $ld(s_j)$ term when general expansion terminates is at most $2\gamma^2$. The result then follows. ■

Lemma 17

$$\sum_{\text{global } s_i \in \mathcal{T}} ld(s_i) \leq \frac{2(c^2+1)}{1-2\gamma^2} OPT(\mathcal{T})$$

Proof: First observe that if s_i and s_j are two different global server sites, then there is no common offline cost in the sums $LC(s_i)$ and $LC(s_j)$. As a consequence we get

$$\sum_{\text{global } s_i \in \mathcal{T}} LC(s_i) \leq OPT(\mathcal{T})$$

Therefore, it suffices to show that

$$\sum_{\text{global } s_i \in \mathcal{T}} ld(s_i) \leq \frac{2(c^2+1)}{1-2\gamma^2} \sum_{\text{global } s_i \in \mathcal{T}} LC(s_i)$$

Applying lemma 16 we have that

$$\sum_{\text{global } s_i \in \mathcal{T}} ld(s_i) \leq 2(c^2+1) \sum_{\text{global } s_i \in \mathcal{T}} LC(s_i) + \sum_{\text{global } s_i \in \mathcal{T}} 2\gamma^2 \sum_{s_j \in S(s_i)} ld(s_j)$$

Since $S(s_i) \cap S(s_j) = \emptyset$ for $i \neq j$, we get that

$$\sum_{\text{global } s_i \in \mathcal{T}} 2\gamma^2 \sum_{s_j \in S(s_i)} ld(s_j) \leq 2\gamma^2 \sum_{\text{global } s_i \in \mathcal{T}} ld(s_i)$$

Now substituting back, we get

$$\sum_{\text{global } s_i \in \mathcal{T}} ld(s_i) \leq \frac{2(c^2+1)}{1-2\gamma^2} \sum_{\text{global } s_i \in \mathcal{T}} LC(s_i)$$

The result then follows. ■

Theorem 18 The halfopt-competitive ratio of BALANCE, with two online servers per site, in the online transportation problem is

$$(c^2+c+2) + \frac{2(c^2+1)}{1-2\gamma^2}$$

Proof: The total online cost is $(c^2+c+1)OPT(\mathcal{T})$ from lemma 14, plus $OPT(\mathcal{T})$, plus $\frac{2(c^2+1)}{1-2\gamma^2}OPT(\mathcal{T})$ from lemma 17. ■

We now assume an arbitrary number of online servers per site.

Theorem 19 The halfopt-competitive ratio of GREEDY for the online transportation problem is

$$(c^2+c+2) + \frac{2(c^2+1)}{1-2\gamma^2}$$

Proof: This follows immediately by conceptually splitting each server site s_i into $c_i/2$ sites. ■

5 Conclusion

The most obvious avenue for further investigation is to determine the competitive ratio in the weakened adversary model when the adversary's capacity is more than half of the online capacity. It seems that some new techniques will be needed in this case since the response graph no longer has the treelike property from lemma 1 that was so critical in our proofs.

References

[1] B. Kalyanasundaram, and K. Pruhs, "Online weighted matching", *Journal of Algorithms*, **14**, 478–488, 1993.

[2] J. Kennington, and R. Helgason, *Algorithms for Network Programming*, John Wiley and Sons, 1980.

[3] S. Khuller, S. Mitchell, and V. Vazirani, "On-line algorithms for weighted matchings and stable marriages", *Theoretical Computer Science*, **127**(2), 255–267, 1994.

[4] E. Koutsoupias, and C. Papadimitriou, "Beyond competitive analysis", *Proceedings of the IEEE Conference on Foundations of Computer Science*, 394–400, 1994.

[5] E. Lawler, *Combinatorial Optimization: Networks and Matroids*, Sanders College Publishing, 1976.

[6] D. Sleator and R. Tarjan, "Amortized efficiency of list update and paging rules", *Communications of the ACM*, **28**, 202–208, 1985.

[7] Y. T. Tsai, C. Y. Tang and Y. Y. Chen, "Average performance of a greedy algorithm for the on-line minimum matching problem on Euclidean space", *Information Processing Letters*, 1994.

[8] N. Young, "The k-server dual and loose competitiveness for paging," *Algorithmica*, **11**, 525–541, 1994.