Competitive Non-Preemptive Call Control

Baruch Awerbuch* Yair Bartal† Amos Fiat† Adi Rosén‡

Abstract

We deal with randomized competitive algorithms for non-preemptive call control on tree-like switching networks. We give an $O(\log n)$ competitive algorithm for non-preemptive call scheduling on trees. We then introduce the complexities of variable call rates, call durations, and arbitrary call benefits, resulting in a polylog competitive algorithm for the combined problem. We also show that many algorithms for similar problems that can deal with fixed parameters such as rates and benefits can be randomly transformed to deal with variable values of the parameters.

Using randomization, this extends the work of [GGKMY] on call control for the line network to tree networks, without the preemption requirement, and while allowing arbitrary benefits, arbitrary rates, and arbitrary capacities on the links. Alternately, this can be viewed as a generalization of [AAP] for throughput competitive routing, limited to trees, but without the limitation of requiring that communication request rates be a small fraction of the link bandwidth.

In common to all our results is an algorithmic paradigm we call "Classify and Randomly Select". This tool seems useful for a variety of online problems.

*Lab. for Computer Science, MIT. E-mail: baruch@theory.lcs.mit.edu

†Department of Computer Science, School of Mathematics, Tel-Aviv University, Tel-Aviv 69978, Israel. Supported by a grant from the Israeli Academy of Sciences. E-mail: yairb@math.tau.ac.il, fiat@math.tau.ac.il,

‡Dept. of Computer Science, Univesity of Toronto, Toronto, Canada. adiro@cs.toronto.edu. Work done while with the Department of Computer Science, Tel-Aviv University, Tel-Aviv, Israel.

1 Introduction

In this paper we consider *on-line benefit problems* that arise in the context of the admission control of calls in high-speed communication networks. Networking technology focuses today on communication networks in which any communication is performed by the reservation of a certain "virtual circuit" between the communicating parties, and the transmission of their communication is done along these fixed lines [Bu, GKR]. This is in contrast to "packet switching" technology in which different portions of the communication between the parties may follow different routes.

Thus, given a communication network, a problem that arises naturally is that of devising a strategy as to the acceptance or rejection of communication requests, that would lead to the reservation of the required bandwidth along some route, and as to the choice of these routes. Obviously, in practice this strategy should be on-line since the decision must be taken without knowing the future requests. The aim of such algorithms would be to maximize some benefit function which is the sum of the benefits associated with the accepted requests. This could be either a uniform benefit, which would lead to the maximization of the number of accepted requests, or a value associated with each call (e.g., duration, an arbitrary price).

The specific problem that we consider is the problem of the admission of calls in a high-speed communication network: Given a network of processors, connected by communication links each having its own capacity, a sequence of communication requests is presented to an algorithm in an on-line manner. Each request is given by the identity of the communicating parties, the bandwidth required by the call, and additional parameters (for a formal definition of the problem see Section 3). We give a randomized $O(\log n)$-competitive algorithm for networks in the form of trees, where n is the size of the network.

The problem we consider here fall into the category of *on-line benefit problems* in which an on-line algorithm has to either accept or reject "entities" presented to it, and has to comply with some constraints. The aim is to maximize the number of entities accepted, or the sum of some values associated with each entity. We develop an on-line paradigm that we use to obtain the results in this chapter. This paradigm does not only apply to the problems presented here, but also to other on-line benefit problems. We call this paradigm "Classify and Randomly Select". It allows building randomized competitive on-line algorithms for benefit problems when (roughly speaking) one can classify the possible events into a relatively small number of disjoint classes. In fact, subsequnet to our work, this paradigm has been used in additional on-line algorithms [].

1.1 Related Work

The call admission control problem in its simplest form is the the following problem: Given a graph, and a set of pairs of vertices that are to communicate, one wishes to allow communication for as many of the pairs as possible. To enable such communication, a communication path connecting the two nodes has to be established, under the constraint that no two paths share an edge. This problem, known as the *maximum disjoint paths problem* in its off-line form, is NP-hard [Ka], and the recent interest in its on-line version has also motivated recent new results on approximation algorithms for it (for certain kinds of graphs) [GVY, KT1, KT2].

For the on-line version of the problem, when the pairs of nodes are given in an on-line manner, Awerbuch, Azar and Plotkin [AAP] give a deterministic competitive algorithm for general networks, but under the major simplification that no single communications request can require more than $1/\log n$ of the bandwidth of a single link; this can also be modeled by the assumption that each link consists of $\log n$ parallel links. They obtain an $O(\log(nT))$-competitive algorithm, where T is the ratio between the longest and shortest durations of the calls (if all calls are of infinite duration then the competitive ratio is $O(\log n)$). If the above assumption is not valid, then the only known algorithms are for specific networks, or networks having given properties [BFKR, AGLR, KT2]. These works give competitive algorithms for the case where the link capacities are not arbitrary (either 1, uniform, or specific capacities due to a special structure of the network). [BFKR] give a deterministic $O(n)$-competitive algorithm for the $n \times n$ mesh. [AGLR] give a randomized algorithm for trees with uniform capacities with competitive ratio of $O(\log d)$, where d is the diameter of the network. They also give a randomized $O(\log \log n)$-competitive algorithm for the $n \times n$ tree of meshes. This algorithm can be extended to give an $O(\log n \log \log n)$-competitive algorithm for the $n \times n$ mesh. [KT2] give a randomized $O(\log n)$-competitive algorithm for a class of networks, that includes the mesh, and thus improve upon the result of [AGLR].

LOWER BOUNDS

2 On-line Benefit problems

We consider on-line benefit problems of the following form: Entities are presented to an on-line algorithm and it has to either accept or reject each of them, under some consistency constraints defined by the problem. We consider problems in which if it is possible to accept entity E together with some set of other entities $\mathcal{E}$, then it is also possible to accept entity E alone. That is, accepting an entity may only prohibit the acceptance of some other entity, but is never a necessary condition for accepting the other entity. Problems of this kind arise when there is some limited resource to be used, or when there are capacity constraints, as in our call admission control problem (e.g., [KVV, AAP, LT]). The algorithm accrues some benefit associated with each entity if it is accepted, and does not accrue any if it is rejected. The benefit need not be equal for all entities. The aim of the algorithm is to maximize the benefit gained which is the sum of the benefits gained for each accepted entity.

2.1 Classify and Randomly Select

In this section we describe a randomized on-line paradigm that we use in the algorithms presented below. This paradigm does not apply only to call admission and related problems, but may be of use to other on-line benefit problems as well. It allows one to build on-line competitive randomized algorithms for benefit problems, when one can divide (in an on-line manner) the entities that are presented to the algorithm into a set of classes, and when one has a set of competitive algorithms that are each able to handle one of those classes. More precisely, if the entities of the benefit problem can be divided (in an on-line manner) into Q disjoint classes, and one has for each such class a c-completive algorithm, then our method results in a $O(cQ)$-competitive randomized algorithm for the problem. Similar techniques have been implicitly used also by Lipton and Tomkins [LT].

Let the set of classes be $\mathcal{P}$, and assume that for every $P \in \mathcal{P}$ we have an on-line algorithm A_P that is c-competitive if presented with entities of class P. We define a randomized algorithm A' as follows: At each time the algorithm holds a value $0 \leq k \leq |\mathcal{P}|$, and a set of classes $\mathcal{G}_i \in \mathcal{P}$, $1 \leq i \leq k$; at the beginning $k = 0$. When a new entity E is presented to A', the entity is first classified into one of the classes of $\mathcal{P}$. If this class is not among the currently held k classes, then the new class is added, and k is incremented by 1. When a new class is added, then if A' did not yet *select* any of the classes $\mathcal{G}_i$ for $1 \leq i \leq k$, A' then chooses to select the new class so that the absolute probability that it is selected is $\frac{1}{|\mathcal{P}|}$. Once a class $\mathcal{G}_i$ is selected then A' uses $A_{\mathcal{G}_i}$, and presents to it all entities that fall into $\mathcal{G}_i$, accepting or rejecting the entity according to the decision of $A_{\mathcal{G}_i}$. All other entities are ignored (and rejected).

Theorem 1 *If for all $P \in \mathcal{P}$ A_P is a c-competitive algorithm, then A' is $O(c|\mathcal{P}|)$-competitive.*

Proof. For an algorithm A, and a sequence of entities G, let $A(G)$ be the set of entities accepted by A when G is presented to it. If A is randomized, then $A(G)$ is a random variable. Denote by $B(\cdot)$ the benefit associated with an entity or a set of entities (which is the sum of the benefits associated with the entities in the set).

Let G be the sequence of entities presented to A' and G_i the subsequence of G consisting of those entities which are classified into $\mathcal{G}_i$. Then $OPT(G) \subseteq \cup_{i=1}^{|\mathcal{P}|} OPT(G_i)$, and $B(OPT(G)) \leq \sum_{i=1}^{|\mathcal{P}|} B(OPT(G_i))$.

The expected benefit accrued by A' is

$$E[B(A'(G))] = \sum_{i=1}^{|\mathcal{P}|} \frac{1}{|\mathcal{P}|} \cdot B(A_{\mathcal{G}_i}(G_i)) \; .$$

Since $A_{\mathcal{G}_i}$ is c-competitive on the sequence of entities G_i, for any i, we have

$$E[B(A'(G))] \geq \sum_{i=1}^{|\mathcal{P}|} \frac{1}{|\mathcal{P}|} \cdot \frac{1}{c} \cdot B(OPT(G_i)) \geq \frac{1}{c \cdot |\mathcal{P}|} B(OPT(G)) \; .$$

□

The interpretation of $\mathcal{P}$ could be the set of possible classes. However, this may lead to high (or even infinite) competitive ratios. A more suitable interpretation is that $\mathcal{P}$ is the set of classes represented in the sequence of entities presented to the algorithm. The above paradigm requires that the algorithm know in advance this number of classes in $\mathcal{P}$ (Note that it is not required to know the actual classes, only their number). Using techniques of Lipton and Tomkins [LT] the above paradigm can be modified to suit the case where the number of classes is not known in advance, incurring a slightly larger increase in the competitive ratio. Define $f(\epsilon) = \sum_{i=1}^{\infty} \frac{1}{i^{1+\epsilon}}$. Note that $f(\epsilon) < \infty$ for all $\epsilon > 0$. When a new class $\mathcal{G}_k$ is first created and if A' did not yet select a class, then the algorithm now chooses to select $\mathcal{G}_k$ with probability such that the absolute probability it is selected is $p_k = \frac{1}{f(\epsilon)}\frac{1}{k^{1+\epsilon}}$ (instead of $\frac{1}{|\mathcal{P}|}$). Since $\sum_{i=1}^{\infty} p_i = 1$ these are valid probabilities. Obviously, although the number $|\mathcal{P}|$ is not known to the algorithm, at most $|\mathcal{P}|$ classes can be created.

Theorem 2 *If for all $P \in \mathcal{P}$ A_P is a c-competitive algorithm, then A' is an $O(c|\mathcal{P}|^{1+\epsilon})$-competitive.*

Proof. We follow and modify the proof of the previous theorem.

The expected benefit accrued by A' in the present case is

$$E[B(A'(G))] = \sum_{i=1}^{|\mathcal{P}|} p_i \cdot B(A(G_i)) \geq$$

$$\sum_{i=1}^{|\mathcal{P}|} p_{|\mathcal{P}|} \cdot B(A(G_i)) \ .$$

Since A is c-competitive on the sequence of entities G_i, for any i, we have

$$E[B(A'(G))] \geq \sum_{i=1}^{|\mathcal{P}|} p_{|\mathcal{P}|} \cdot \frac{1}{c} \cdot B(OPT(G_i)) \geq$$

$$\frac{1}{f(\epsilon)} \frac{1}{c} \frac{1}{|\mathcal{P}|^{1+\epsilon}} B(OPT(G)) \ .$$

□

A typical scenario in which our method is applicable is in benefit problems where the entities are parameterized by some parameter p, and we have a competitive algorithm that handles entities of the problem if the parameter differs by at most a factor of 2 (or some other constant). More precisely, given a c-competitive algorithm that is suitable for the case that parameter p assumes values that differ by at most a factor of 2, we build upon it and devise an $O(c \log M)$-competitive randomized algorithm that tolerates values that differ by a factor of at most M, i.e., M is the ratio between the largest and smallest value p gets in the sequence of entities presented. If the value M is known in advance then we get a $O(c \log M)$-competitive algorithm, while in the case that it is unknown we get a $O(c \log^{1+\epsilon} M)$-competitive algorithm. This is done by applying the above paradigm as follows: Let A be an algorithm tolerating values of p that differ by a factor of at most 2. We define a randomized algorithm A'. The algorithm will create in an on-line manner at most $L = \lceil \log M \rceil + 1$ classes $\mathcal{G}_i$, $1 \leq i \leq L$, and will associate with each class a number g_i. At each time the algorithm holds a value $0 \leq k \leq L$, and a set of classes $\mathcal{G}_i$, $1 \leq i \leq k$; at the beginning $k = 0$. Upon the arrival of a new entity E with parameter p such that $2^{j-1} < p \leq 2^j$, the algorithm checks if there is a class $\mathcal{G}_i$ such that $g_i = j$. If there is such a class, then the algorithm associates entity E with this class. Otherwise A' increments k by 1, and creates a new class $\mathcal{G}_k$ with $g_k = j$ and associates E with it. When a new class number k is created, then if A' did not yet select any of the classes $\mathcal{G}_i$ for $0 \leq i \leq k$, A' then chooses to select the new class so that the absolute probability it is selected is $\frac{1}{L}$, if L is known in advance, or p_k, if L is not known in advance. Thus, the randomized algorithm A' selects one class among the classes $\mathcal{G}_i$. It uses algorithm A presenting to it those entities that fall into the selected class, while ignoring (and rejecting) all other entities.

¿From Theorems 1 and 2 we have,

Corollary 3 *Let M be the ratio between the highest and lowest values of p in the entities presented to A'. If A is a c-competitive algorithm that tolerates values of p that differ by at most a factor of 2, then, in case that M is known in advance, A' is $O(c \log M)$-competitive, and in case M is not known in advance, A' is $O(c \log^{1+\epsilon} M)$-competitive.*

3 The Model

A graph $G = (V, E)$ consisting of $|V| = n$ nodes represents a communication network where the nodes are communication switches and the edges are communication links. Every communication link $e \in E$ has a *bandwidth* (capacity) $c(e)$.

A call is represented by a tuple:

$$C = (v_s, v_t, r, t, d, b) ,$$

where v_s is the origin of the call, v_t is its destination, r is the bandwidth required by the call (its rate), t is the starting time (or arrival time) of the call, d its duration, and b the benefit accrued by the algorithm if it accepts the call.

Calls are presented to an algorithm in an on-line manner and the algorithm has, upon receipt of each call, to either *accept* or *reject* it. A call is accepted by assigning it to some path from v_s to v_t such that for each edge e in the path the sum of rates of calls using that edge at time t is at most $c(e) - r$. Once a call is accepted it continues to use the path for d time units until completion (i.e., preemption is disallowed). For any accepted call the algorithm accrues the benefit b associated with the call. The objective is to maximize the total benefit gained.

In Section 4 below we give results for the case that the graph G is a tree. In this case the algorithm can only decide whether to accept or reject the call (and not its routing).

3.1 Analysis

We analyze the performance of the algorithms using the competitive ratio measure, comparing the benefit obtained by the on-line algorithm to the optimal benefit obtainable on the same sequence. An on-line algorithm A for a benefit problem is c-competitive on a set $\mathcal{I}$ of instances if

$$\forall I \in \mathcal{I} \ \ \mathrm{Benefit_A}(I) \geq \frac{1}{c} \cdot \mathrm{Benefit_{OPT}}(I).$$

(Note that we do not allow here an additive constant.) In our setting an instance is composed of a network, and a sequence of requests for connections. Let $\mathcal{I}(n, R, T, B)$ be the set of instances of networks with size at most n, the ratio between the highest and lowest rates requested at most R, the ratio between the longest and shortest durations requested at most

T, and the ratio between the highest and lowest benefits at most B. The competitive ratios that we obtain are a function of these parameters. We thus obtain algorithms that are $c(n,R,T,B)$-competitive, that is

$$\forall I \in \mathcal{I}(n,R,T,B) \quad \text{Benefit}_{\text{A}}(I) \geq \frac{1}{c(n,R,T,B)} \cdot \text{Benefit}_{\text{OPT}}(I).$$

The algorithms we discuss here are sometimes randomized. In this case we are interested in the expectation of the benefit, over the random choices of the algorithm. That is, we prove competitive results against an oblivious adversary [BBK+] of the form,

$$\forall I \in \mathcal{I}(n,R,T,B) \quad E[\text{Benefit}_{\text{A}}(I)] \geq \frac{1}{c(n,R,T,B)} \cdot \text{Benefit}_{\text{OPT}}(I).$$

4 The Base Algorithm

We first consider the base case where all call durations are infinite, all benefits associated with the calls are equal (without loss of generality assume they are 1), and all rates are 1. In this case a call is defined by a pair $C = (v_s, v_t)$, since all other values are implicit. This problem has a more combinatorial flavor in that the problem consists of maximizing the number of disjoint paths. In this section we give a randomized algorithm for this base case on trees. We later build upon this base algorithm and extend it to deal with the case where all parameters may vary.

For this algorithm we use the "Classify and Randomly Select" paradigm. We show that we can classify the presented calls into $O(\log n)$ classes, such that for each class the greedy (deterministic) algorithm is 2-competitive. Thus we will classify each incoming call into one of the classes, and once a class has been selected we consider the calls that fall into this class and accept them greedily (i.e., we will not accept a call only if it is impossible to accommodate it in the tree under the capacity constraints). We will thus obtain an $O(\log n)$ competitive algorithm, which we call the base algorithm.

4.1 The Classification Algorithm

The classification algorithm that we present here builds upon an algorithm for the related problem of deterministic call admission control *with preemption* on trees [BFKR]. To define the classification we assign a number to each of the nodes in the following recursive way: Given a tree T of k nodes there always exists a vertex v such that its removal leaves a forest of trees, each of which contains at most $(2/3)k$ nodes. Such a node always exists as trees have 2/3-separators of size 1 (cf. [Le]). We associate levels with the vertices recursively, starting with level $l = 1$. We choose a vertex v as described above, assign level l to v and apply the procedure recursively to each of the subtrees, starting at level $l+1$. Thus each of the nodes of the tree is assigned a level in the range $1 \leq l \leq \lceil \log_{3/2} n \rceil$. We now assign a level to each of the edges, by giving it the lower level among the levels assigned to its two endpoints.

The classification algorithm can now be stated in a very simple way:

Let l be the lowest-numbered level among the levels of the edges that are included in a call (note that since we are considering a tree, each call has a single possible route). Then, classify the call into class number l.

The above rule classifies the calls into $O(\log n)$ disjoint classes.

We now consider the greedy on-line algorithm that accepts every call if possible. That is, it does not accept a call only if its acceptance will violate the capacity constraint of at least one edge. We argue that if we consider only the calls of a single class of the above defined classes, then the above greedy on-line algorithm is 2-competitive.

Claim 4 *Let p be a call of class l. Then p includes exactly one vertex of level l.*

Proof. Since p is of level l it includes an edge of level l. An edge of level l has one vertex adjacent to it which is of level l, and the call p includes this vertex. Assume towards a contradiction that p includes two distinct vertices of level l. Each of these vertices is in a distinct tree that was created when a vertex of level $l-1$ was extracted in the process of assigning levels to nodes. Therefore the call p must pass through this vertex of level $l-1$, and therefore through an edge of level at most $l-1$. Thus the call p is of a level strictly smaller than l. A contradiction. □

Lemma 5 *The greedy on-line algorithm applied to the calls of a single class is 2-competitive.*

Proof. By Claim 4 every call of level l passes in exactly one vertex of level l. For a node v of level l let P_v be the set of level l calls that pass in v. We thus partition the calls of level l into disjoint sets, and will show that within each such set the greedy algorithm is 2-competitive.

Consider a set P_v. We further partition this set. To do that we define an edge to be *saturated* if $\lfloor c(e) \rfloor$ accepted calls pass in it (i.e., it is impossible to add another call). At the time after the last request has been handled, consider the set of saturated edges around v, which are the closest to v. That is, for every v of level l, we define a set $S(v)$ that includes all saturated edges, such that for any $e \in S(v)$ there is no saturated edge between e and v. We now assign each call in P_v to the edges in $S(v)$ in which it passes (note that any call may pass in at most two such edges): We create the set $P(e)$ for every $e \in S(v)$, that includes all calls in P_v that pass trough e, and an additional set F of those calls in P_v that do not pass through any edge of $S(v)$.

First observe that all calls in F must have been accepted by greedy, since when they arrive they can be added.

Considering each set $P(e)$ as defined above, let $P_+^+(e)$ be the set of calls in $P(e)$ accepted by both greedy and the off-line algorithms, let $P_-^+(e)$ be the calls in $P(e)$ accepted by greedy but not by the off-line algorithm, let $P_+^-(e)$ be the calls in $P(e)$ accepted by the off-line but not by greedy, and let $P_-^-(e)$ be the calls in $P(e)$ rejected by both.

Let $C(e) = \lfloor c(e) \rfloor$. Since e has been saturated by greedy, greedy has accepted at least $C(e)$ calls that pass in e. The off-line algorithm has accepted at most $C(e)$ such calls. We get

$$|P_+^+(e)| + |P_-^+(e)| \geq C(e) \geq |P_+^+(e)| + |P_+^-(e)|.$$

Thus,

$$|P_-^+(e)| \geq |P_+^-(e)|.$$

Analogously to $P(e)$, we partition now P_v. Let $P_+^+(v)$ be the set of calls in P_v accepted by both greedy and the off-line algorithms, let $P_-^+(v)$ be the calls in P_v accepted by greedy but not by the off-line algorithm, let $P_+^-(v)$ be the calls in P_v accepted by the off-line algorithm but not by greedy, and let $P_-^-(v)$ be the calls in P_v rejected by both.

Since each call can pass in at most two edges $e \in S(v)$, we get

$$2|P_-^+(v)| \geq |F| + \sum_{e \in S(v)} |P_-^+(e)| \geq |F| + \sum_{e \in S(v)} |P_+^-(e)| \geq |P_+^-(v)| \ .$$

Thus,

$$(|P_+^+(v)| + |P_+^-(v)|)/(|P_+^+(v)| + |P_-^+(v)|) \leq (|P_+^+(v)| + 2|P_-^+(v)|)/(|P_+^+(v)| + |P_-^+(v)|) \leq 2.$$

□

4.2 The Randomized Base Algorithm

Let $N \leq \lceil \log_{3/2} n \rceil$ be the maximum level number assigned to the edges of the tree. The randomized call admission algorithm is now built by using the "Classify and Randomly Select" paradigm, applied to the N classes defined by the above classification algorithm, and using the greedy algorithm for the class selected. We thus have the following theorem.

Theorem 6 *The Base Algorithm for networks in the form of trees is $O(\log n)$-competitive, where n is the number of nodes in the tree.*

5 The General Case

In this section we consider the general case when all the parameters of the call, its rate, duration, and benefit may vary. To obtain an algorithm for this case we use the "Classify and Randomly Select" paradigm and apply an algorithm that tolerates the case that these values vary by at most a factor of 2. For the case of varying rates we show that our base algorithm is still $O(\log n)$-competitive if the rates vary by at most a factor of 2. For the case of the duration and benefit we show that any call admission algorithm for the base case can be modified to suit the case that the duration, or benefit, vary by at most a factor of 2, and therefore one can apply our paradigm to any such algorithm.

5.1 Varying Rates

As mentioned above we show that our algorithm for the base case is $O(\log n)$-competitive even when the rates vary by a factor of at most 2. More precisely we show that,

Lemma 7 *If all calls have rate r such that $\alpha < r \leq 2\alpha$, for some α, then the greedy on-line algorithm, applied to the calls of a single class l defined in the classification algorithm, is 8-competitive.*

Proof. We modify the proof of Lemma 5. Every call of level l passes in one vertex of level l. We partition the calls of level l according to the node of level l in which they pass. For a node v of level l let P_v be the set of level l calls that pass in v. We thus partition the calls of level l into disjoint sets, and for each such set we prove that greedy is 8-competitive.

We first eliminate from the sequence of requests those calls that cannot possibly be accepted. These are calls that have rate r and pass through an edge e such that $c(e) < r$. Such calls cannot be accepted by any algorithm, and we ignore them. In what follows we consider the subsequence obtained from the original sequence by eliminating such calls.

We define a saturated edge. An edge e is saturated if it accommodates $k > \max\{0, c(e)/2\alpha - 1\}$ calls. Note that greedy accepts any call in the (modified) sequence that, when presented, does not pass through any saturated edge.

Consider a set P_v. We further partition this set. At the time after the last request has been handled, consider the set of saturated edges around v. That is, for every v of level l, we define a set $S(v)$ that includes all saturated edges, such that for any $e \in S(v)$ there is no saturated edge between e and v. We now assign each call in P_v to the edges in $S(v)$ in which it passes (note that any call may pass in at most two such edges): We create the set $P(e)$ for every $e \in S(v)$, which includes all calls in P_v that pass through e, and an additional set F of those calls in P_v that do not pass through any edge of $S(v)$.

First observe that all calls in F must have been accepted by greedy, since when they arrive they can be added.

Considering each set $P(e)$ as defined above, let $P_+^+(e)$ be the set of calls in $P(e)$ accepted by both greedy and the off-line algorithms, let $P_-^+(e)$ be the calls in $P(e)$ accepted by greedy but not by the off-line algorithm, let $P_+^-(e)$ be the calls in $P(e)$ accepted by the off-line but not by greedy, and let $P_-^-(e)$ be the calls in $P(e)$ rejected by both.

Since e has been saturated by greedy, greedy has accepted more than $\max\{0, c(e)/2\alpha - 1\}$ calls that pass in e. The off-line algorithm has accepted at most $c(e)/\alpha$ such calls. We get

$$\begin{aligned} 2(|P_+^+(e)| + |P_-^+(e)|) &> 2(c(e)/2\alpha - 1) \\ &= c(e)/\alpha - 2 \\ &\geq |P_+^+(e)| + |P_+^-(e)| - 2. \end{aligned}$$

Since $e \in S(v)$, greedy has accepted at least one call that pass in e, i.e., $|P_+^+(e)| + |P_-^+(e)| \geq 1$. We have

$$\begin{aligned} 4(|P_+^+(e)| + |P_-^+(e)|) &\geq 2(|P_+^+(e)| + |P_-^+(e)|) + 2 \\ &> |P_+^+(e)| + |P_+^-(e)| \end{aligned}$$

We get

$$3|P_+^+(e)| + 4|P_-^+(e)| \geq |P_+^-(e)|.$$

Analogously to $P(e)$, we partition now P_v. Let $P_+^+(v)$ be the set of calls in P_v accepted by both greedy and the off-line algorithms, let $P_-^+(v)$ be the calls in P_v accepted by greedy but not by the off-line algorithm, let $P_+^-(v)$ be the calls in P_v accepted by the off-line algorithm but not by greedy, and let $P_-^-(v)$ be the calls in P_v rejected by both.

Since each call can pass in at most two edges $e \in S(v)$, we get,

$$6|P_+^+(v)| + 8|P_-^+(v)| \geq |F| + \sum_{e \in S(v)} (3|P_+^+(e)| + 4|P_-^+(e)|) \geq |F| + \sum_{e \in S(v)} |P_+^-(e)| \geq |P_+^-(v)| \ .$$

Thus,

$$(|P_+^+(v)| + |P_+^-(v)|)/(|P_+^+(v)| + |P_-^+(v)|) \leq (7|P_+^+| + 8|P_-^+|)/(|P_+^+(v)| + |P_-^+(v)|) \leq 8 \ .$$

□

Denote by R the ratio between the highest and lowest rates in the set of calls presented. Then, we can apply "Classify and randomly Select", by classifying each call into one of $O(\log R)$ classes, such that in each class the rates of the calls vary by at most a factor of 2, and applying the base algorithm to this class. We conclude

Theorem 8 *There is a randomized on-line call admission algorithm for trees with arbitrary capacities for the links, with competitive ratio $O(\log n \log R)$. If the ratio R is not known in advance then the competitive ratio is $O(\log n \log^{1+\epsilon} R)$.*

5.2 Varying Benefits and Durations

If the calls are not of the same benefit, but the ratio between the highest and lowest benefits is B, then we can transform any call admission algorithm for the base case of equal benefits (on any graph) into one that can handle this case at a cost of augmenting the competitive ratio by a factor of $O(\log B)$. This is done by applying the scaling method of the "Classify and Randomly Select" paradigm. We classify the calls into $O(\log B)$ classes according to their benefits, and apply the base algorithm whose competitive ratio will only increase by a factor of at most 2, on any of these classes.

Corollary 9 *Any c-competitive call admission algorithm for the case of equal benefits can be transformed into an $O(c \log B)$-competitive randomized algorithm for the call admission problem with calls with arbitrary benefits. If the ratio B is not known in advance then the competitive ratio is $O(c \log^{1+\epsilon} B)$.*

We can also handle the case where the calls have a starting time and a duration, which is known to the on-line algorithm at the time the call arrives.

Let T be the ratio between the longest and the shortest durations. By Corollary 3 we can use a competitive call admission algorithm that handles durations that vary by a factor of at most 2, to get a randomized algorithm that handles arbitrary durations with an $O(\log T)$ multiplicative increase in the competitive ratio.

We now show that any call admission algorithm that assumes infinite durations can be transformed into an algorithm to handle durations that vary by a factor of at most 2, augmenting the competitive ratio by only a constant factor. Let all durations d be such that $t^* < d \leq 2t^*$. Divide the time axis into slots of length t^*, and number the slots with increasing numbers. Choose with equal probability a number $q \in \{0, 1, 2\}$, and mark all slots numbered n such that n is congruent to q modulo 3 as *relevant* slots. The algorithm now considers only calls with starting times in a relevant slot, ignoring all other calls. If we consider only calls that arrive in one such slot, then no call terminates before another arrives; therefore, the case is the same as the case of infinite durations. Furthermore, every call that starts in a relevant time slot terminates before the beginning of the next relevant time slot; thus every such time slot is handled separately.

Corollary 10 *Any c-competitive call admission algorithm that handles infinite durations can be transformed into an $O(c \log T)$-competitive randomized algorithm for the call admission problem with calls of arbitrary durations. If the ratio T is not known in advance then the competitive ratio is $O(c \log^{1+\epsilon} T)$.*

We note that all three extensions can be combined to handle the general case.

6 Conclusions

References

[AAFPW] J.Aspenes, Y.Azar, A.Fiat, S.Plotkin, and O.Waarts. On-line machine scheduling with applications to load balancing and virtual circuit routing. Proc. 23rd Annual ACM Symposium on Theory of Computing, San Diego, May 1993.

[AAP] B.Awerbuch, Y.Azar, and S.Plotkin, Throughput-Competitive Online Routing. To appear in Proc. 34th Symposium on Foundations of Computer Science. November 1993.

[APW] Y.Azar., S.Plotkin, and O.Waarts, Routing of virtual circuits with limited duration. Manuscript.

[BK] B.Kalyanasundaram and K.Pruhs, Online load balancing of temporary tasks. Manuscript.

[BBK+] S. Ben-David, A. Borodin, R.M. Karp, G. Tardos, and A. Wigderson. On the power of randomization in online algorithms. In *Proc.of the 22nd Ann. ACM Symp. on Theory of Computing*, pages 379–386, may 1990.

[FKL+88] Fiat, Karp, Luby, McGeoch, Sleator, and Young. Competitive paging algorithms. Technical report, Carnegie Mellon, 1988.

[GG] J.A.Garay and I.S.Gopal, Call preemption in Communications Networks, Proc. INFOCOM '92, Florence, Italy, 1992.

[GGKMY] J.Garay, I.S.Gopel, S.Kutten, Y.Mansour, and M.Yung, Efficient on-line call control algorithms, In Proc. of 2nd Annual Israel Conference of Computing and Systems, Netania, Israel, 1993.

[LT] R.J.Lipton, and A.Tomkins, Online Interval Scheduling, Manuscript.

[ST] Sleator and Tarjan. Amortized efficiency of list update and paging rules. *Communication of the ACM*, 28(2):202–208, 1985.

[AAFPW] J. Aspenes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. On-Line Load Balancing with Application to Machine Scheduling and Virtual Circuit Routing. In *Proc. 25th Annual ACM Symp. on Theory of Computing*, pages 623–631, 1993.

[AAP] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput-Competitive On-Line Routing. In *Proc. 34th Symp. on Foundations of Computer Science*, pages 32–40, 1993.

[AGLR] B. Awerbuch, R. Gawlick, F.T. Leighton, and Y. Rabani. On-Line Admission Control and Circuit Routing for High Performance Computing and Communication. In *Proc. of the 35th Annual Symp. on Foundations of Computer Science*, pages 412–423, 1994.

[AKPPW] Y. Azar., B. Kalyanasundaram, S. Plotkin, K. Pruhs and O. Waarts. Online Load Balancing of Temporary Tasks. In *Proc. 3rd Workshop on Algorithms and Data Structures*. pages 119–130, 1993.

[Bu] R. Buck. The Oracle Medai Server for nCUBE Massively Parallel Systems. In *Proc. of the 8th International Parallel Processing Symp.* pages 670–673, 1994.

[BFKR] A. Blum, A. Fiat, H. Karloff and Y. Rabani. Personal communication. 1993.

[BFL] Y. Bartal, A. Fiat, and S. Leonardi. Personal communication. 1995.

[GGKMY] J. Garay, I.S. Gopel, S.Kutten, Y.Mansour, and M.Yung. Efficient on-line call control algorithms. In *Proc. 2nd Israel Symp. on Theory of Computing and Systems*, pages 285–293, 1993.

[GKR] R. Gawlick, C. Kamanek, and K.G. Ramakrishnan. On-Line Routing for Virtual Private Networks. Manuscript. 1994.

[GVY] N. Garg, V. Vazirani, and M. Yannakakis. Primal Dual Approximation Algorithms for Integral Flow and Multicat in Trees, with Applications to Matching and Set Cover. In *Proc. of ICALP*, pages 64–75, 1993.

[Ka] R.M. Karp. Reducibility among Combinatorial Problems. *Complexity of Computer Computations*, R.E. Miller, J.W. Thatcher, Eds. New York: Plenum Press, 1972, pages 85–103.

[KT1] J. Kleinberg, and E. Tardos. Approximations for the Disjoint Paths Problem in High-Diameter Planar networks. In *Proc. 27th Annual ACM Symp. on Theory of Computing*, pages 26–35, 1995.

[KT2] J. Kleinberg, and E. Tardos. Disjoint Paths in Densely Embedded Graphs. In *Proc. 36th Symp. on Foundations of Computer Science*, 1995, To appear.

[KVV] R. Karp, U. Vazirani, and V. vazirani. An Optimal Algorithm for On-Line Bipartite Matching. In *Proc. 22nd Annual ACM Symp. on Theory of Computing*, pages 352–358, 1990.

[LT] R.J. Lipton, and A. Tomkins. Online Interval Scheduling. In *Proc. 5th ACM-SIAM Symp. on Discrete Algorithms*, pages 302–320, 1994.

[ST] D.D. Sleator and R.E. Tarjan. Amortized Efficiency of List Update and Paging Rules. *Communication of the ACM*, 28(2) pages 202–208, 1985.

[Le] J. Van Leeuwen, (ed.). *Handbook of Theoretical Computer Science*, Vol. A, *Algorithms and Complexity.* The MIT Press, 1990.