In: Proceedings of the 3rd Workshop on Approximation and Randomization Algorithms in Communication Networks (ARACNE 2002), Rome, Italy, September 2002, pp. 51–64.

Call Admission Control for Advance Reservation Requests with Alternatives*

T. ERLEBACH
ETH Zürich, Switzerland

Abstract

Call admission control is the problem of deciding for a given set of connection requests which of them to accept and which to reject, with the goal of maximizing the profit obtained from the accepted requests. The problem is considered in a scenario with advance reservations where users can specify several alternatives for when and how they want their connections to be established. Two variants are studied: In BCA (batch call admission), all calls request to be established at the same time. In GCA (general call admission), every request specifies a starting time and duration. Both variants generalize the well-studied unsplittable flow problem. For star networks, approximation algorithms with constant ratio for BCA with arbitrary edge capacities and for GCA with unit edge capacities are presented. For GCA with arbitrary edge capacities, ratio $O(\log R)$ is achieved, where R is the ratio of the maximum to the minimum edge capacity. For trees and trees of rings with unit edge capacities, constant-factor approximation algorithms for BCA are given.

Keywords

Approximation Algorithm, Linear Programming, Unsplittable Flow.

1 Introduction

We consider call admission control (CAC) in networks with bandwidth reservation. Bandwidth reservation means that a connection request (call) specifies a certain bandwidth requirement and, if the call is established, the network allocates the requested amount of bandwidth on all links along some path from the sender to the receiver of the call. This is important for applications such as video conferencing that benefit from guaranteed quality of service.

We assume that each connection request is associated with a certain profit. The network provider obtains the profit only if the call is established. CAC is the

*Research partially supported by the Swiss National Science Foundation under Contract No. 2100-63563.00 (AAPCN) and Contract No. 5003-057753/3 (ANAISOFT/PCAC) and by EU Thematic Network APPOL II, IST-2001-32007, with funding provided by the Swiss Federal Office for Education and Science (BBW).

problem of deciding which requests to accept and which to reject, with the goal of maximizing the total profit obtained from the accepted calls. We consider CAC in a scenario with advance reservations and call alternatives. Advance reservation means that requests for a connection are submitted to the network earlier than the time when the connection should be established. A request is determined by giving the sender, the receiver, the bandwidth requirement, the starting-time, the duration, and the profit value. Allowing call alternatives means that a user can specify several alternative requests for how to establish a connection. The network can either accept one of the alternatives, or reject the call completely. We allow that all request parameters can be different for different alternatives of a call.

In the scenario that we consider, the network need not handle each request immediately, but is allowed to collect requests over some period of time and then decide about acceptance or rejection of the collected requests. For example, one could imagine that requests for a connection must be submitted to the network until 5.00pm on the day before the connection is to be established. After 5.00pm, the network could consider all advance reservations for the next day together and select those requests that maximize the total profit obtained. In this way, the problem becomes an off-line problem, and the network can use better algorithms for optimizing the CAC decisions.

Preliminaries and Problem Definitions. The network is represented by a simple, undirected graph $G = (V,E)$ and a function $B : E \mapsto \mathbb{R}$ that maps each edge $e \in E$ to a positive capacity $B(e)$. Let $B_{\min} = \min_{e \in E} B(e)$ and $B_{\max} = \max_{e \in E} B(e)$ denote the minimum and maximum edge capacity in the network, respectively, and let $R = B_{\max}/B_{\min}$. A call c is a set of call instances (alternatives), and each call instance $i = (u_i, v_i, b_i, t_i, d_i, p_i)$ is a tuple consisting of a source node $u_i \in V$, a destination node $v_i \in V$, a positive bandwidth requirement $b_i \in \mathbb{R}$, a starting time $t_i \in \mathbb{N}$, a duration $d_i \in \mathbb{N}$, and a profit $p_i \in \mathbb{R}$. For a given set C of calls, we denote by $I_C = \bigcup_{c \in C} c$ the set of all instances of all calls in C. If C is clear from the context, we also write I for I_C. The call to which a call instance i belongs is denoted by $c(i)$.

An instance of the problem GENERALCALLADMISSION (GCA) is given by a graph G with edge capacities B and a set C of calls. A solution is a subset $A \subseteq I_C$ (the set of *accepted* call instances) and an assignment of a path $path(i)$ from u_i to v_i in G for every accepted call instance $i \in A$. A solution is feasible if (1) at most one call instance is accepted from each call, and (2) the sum of the bandwidth requirements of simultaneously active call instances using the same edge does not exceed the capacity of the edge:

$$\forall e \in E : \ \forall t \in \mathbb{N} : \sum_{i \in A(e,t)} b_i \leq B(e),$$

where $A(e,t) = \{i \in A : t_i \leq t < t_i + d \text{ and } e \in path(i)\}$ is the set of accepted call instances that use edge e at time t. The goal is to compute a feasible solution that maximizes the sum $\sum_{i \in A} p_i$ of the profits of the accepted call instances. We also consider the special case of GCA where all call instances overlap in time, called BATCHCALLADMISSION (BCA). In particular, BCA includes the case where all call instances have the same starting-time. If alternatives are not allowed, BCA is equivalent to the unsplittable flow problem [2].

We distinguish the cases of unit edge capacities (all edges have the same capacity) and arbitrary edge capacities. We refer to BCA and GCA in networks with unit edge capacities as unit-BCA and unit-GCA, respectively.

BCA is *NP*-hard even if the network consists of a single edge and every call consists of a single instance. This restricted version of BCA is equivalent to the KNAPSACK problem (the capacity of the edge corresponds to the size of the knapsack and the calls correspond to objects to be packed in the knapsack), which is *NP*-hard. Therefore, we are interested in approximation algorithms. We say that an algorithm for GCA or BCA is a ρ-approximation algorithm if it runs in polynomial time and always outputs a feasible solution whose total profit is at least a $1/\rho$-fraction of the optimal solution.

We consider BCA and GCA in stars, trees, and trees of rings. A star is a graph that consists of a center x and n nodes y_1, y_2, $\ldots$, y_n that are adjacent to x but not to each other. A tree is a connected graph without cycles. A tree of rings is a connected graph in which all biconnected components are rings; roughly speaking, it consists of a number of rings that are interconnected in a tree structure such that two rings share at most one node. Note that the path $path(i)$ for call instance i is uniquely determined by u_i and v_i in stars and trees.

GCA on a single link and unit-BCA on a line network (a chain of nodes) are closely related: the time slots in GCA correspond to the links in BCA, i.e., a call with starting time t_i and duration d_i in the GCA problem can be viewed as a call from node t_i to node $t_i + d_i$ in the BCA problem for a line network with nodes $0, 1, \ldots, \max_{i \in I} t_i + d_i$.

Motivation. In many studies on CAC, it is assumed that users submit connection requests only at the time when the connection should be established. For each request, the network must decide immediately whether to accept or reject the call. This on-line scenario makes it difficult for the provider to optimize the profit, and for the users there is always the uncertainty of whether it will be possible to establish the desired connection at the desired time. Advance reservations [15] and call alternatives are natural concepts that are attractive for the network provider as well as for users. For the network provider, they increase the flexibility and the potential of optimizing the profit obtained from the accepted requests. For the users, advance reservations that are confirmed ahead of time (although not immediately) provide a guarantee that the desired video conference etc. can take place at the desired time. By specifying several alternatives, the user can increase the probability that the network accepts the call.

Since already the unsplittable flow problem (a special case of BCA) is hard to approximate in general directed graphs [11], it is natural to consider special classes of network topologies. We consider stars, trees, and trees of rings. CAC

Table 1: Known results and results of this paper (in bold).

	unit-BCA	BCA	unit-GCA	GCA
single link	$1+\varepsilon$	$1+\varepsilon$	5	5
star	**5**	**10**	**10**	$\boldsymbol{O(\log R)}$
line	5	–	–	–
tree	**10**	–	–	–
tree of rings	**18**	–	–	–

algorithms for star networks actually apply to general networks if the provider has rented the capacity of his/her network from another provider according to the hose model of bandwidth reservations. In the hose model, there are ingress and egress limits for each border router of the network, and the network guarantees that it has enough bandwidth for any traffic pattern that does not violate the ingress and egress limits (see [7]). Thus, the network is equivalent to a star network for the purposes of admission control (see also [6]).

Summary of Results. We propose new approximation algorithms for BCA and GCA in stars and for unit-BCA in trees and trees of rings. The previously known approximation ratios for lines and single links as well as the ratios achieved by our new algorithms (in bold) are shown in Table 1. Our algorithms do not assume any restrictions on the bandwidth requirements. In fact, if one requires all bandwidth requirements to be small (at most a logarithmic fraction of the smallest edge capacity), a constant-factor approximation algorithm for BCA in arbitrary graphs can easily be obtained using randomized rounding. Regarding the known results shown in Table 1, ratio 5 for unit-BCA in lines and GCA on single links is achieved by an algorithm from [3] and ratio $1+\varepsilon$ for BCA on single links can be obtained by adapting an FPTAS for KNAPSACK (e.g. [12]). Table entries "–" indicate that no good approximation algorithm is known for these problem variants yet.

Related Work. We are not aware of previous work addressing the GCA problem in our formulation (with advance reservations and call alternatives), but call admission control in networks with bandwidth reservation has been studied by many authors. Much of the previous work has dealt with the on-line scenario in which each request must be accepted or rejected immediately, see [14] and [5, Chapter 13] for surveys. Here, we mention only the on-line algorithm from [1] that achieves a ratio of $O(\log nTF)$ for calls with durations in $[1,T]$ and profit values in $[b_i d_i, F b_i d_i]$ provided that the maximum bandwidth requirement is at most a logarithmic fraction of the minimum edge capacity. This algorithm can deal with advance reservations.

In the off-line case, the maximum edge-disjoint paths problem (MEDP), a special case of BCA, has received considerable attention; see [13] for an introduction. For general directed graphs, the problem is not approximable within $m^{0.5-\varepsilon}$ unless $P = NP$, where m is the number of edges [11]. Approximation within ratio $O(\sqrt{m})$ is possible for the more general unsplittable flow problem (BCA without alternatives) if the maximum bandwidth requirement is at most the minimum edge capacity (see [2] and the references given there).

Concerning the approach we use in our approximation algorithms, a paper by Bar-Noy et al. [4] is highly relevant. That paper considers a scheduling problem in which the total weight of jobs completing by their deadlines is maximized. A linear programming relaxation is solved and the fractional solution is decomposed into a convex combination of integral solutions using a coloring algorithm. The best of the integral solutions obtained in this way is output by the algorithm. This technique was generalized to weighted independent set problems for certain classes of graphs that can be colored with a constant times the clique number many colors in [10]. In [16], the technique of [4] was applied to other scheduling problems with rejection. Among other results, they give a 4-approximation algorithm for a resource-constrained scheduling problem that includes GCA on a single link and unit-BCA in a line network (without alternatives) as special cases.

A unified approach to resource allocation and scheduling using the so-called *local ratio technique* is presented in [3]. One of the applications of their approach is GCA on a single link (and thus also unit-BCA in line networks), for which they obtain a 5-approximation algorithm. This approach is extended to lines, rings, and trees in [15]. For trees with n nodes, they obtain approximation ratio 5 for unit-BCA and $O(\log n)$ for unit-GCA without alternatives.

2 Call Admission in Star Networks

We assume throughout this paper that every call instance is such that it could be established if there was no other traffic in the network, i.e., we assume that $b_i \leq B(e)$ for all $e \in path(i)$ and for all $i \in I$.

2.1 Linear Programming Relaxation

Let $T = \max_{i\in I} t_i + d_i$. Consider the following linear programming relaxation of GCA, which uses a variable x_i for every call instance i that indicates whether i is accepted ($x_i = 1$) or rejected ($x_i = 0$):

$$(LP_1) \quad \max \sum_{i\in I} p_i x_i \tag{1}$$

$$\text{s.t.} \sum_{i\in c} x_i \leq 1, \quad c \in C \tag{2}$$

$$\sum_{i\in I(e,t)} x_i b_i \leq B(e), \quad e \in E, 0 \leq t < T \tag{3}$$

$$0 \leq x_i \leq 1, \quad i \in I \tag{4}$$

where $I(e,t) = \{i \in I : e \in path(i), t_i \leq t < t_i + d_i\}$ is the set of all call instances that use edge e at time t. Constraint (2) says that at most one instance of a call can be accepted. Constraint (3) says that, for any edge e and any time t, the sum of the bandwidth requirements of accepted call instances that use edge e at time t is at most $B(e)$, ensuring that the capacity of e is not exceeded. (Note that it suffices to include constraints (3) only for those values of t that are the starting time of at least one call instance; hence, the size of LP_1 is polynomial in the size of the input.) The objective function (1) simply states that the goal is to maximize the total profit of accepted call instances. If we required that $x_i \in \{0,1\}$ for all $i \in I$, we would have an integer linear programming formulation of GCA (but nothing would be gained because solving an ILP is an *NP*-hard problem). Instead, we relax the integrality constraint and allow the x_i to take arbitrary values between zero and one (constraint (4)). The resulting linear program can be solved optimally in polynomial time, yielding values $\hat{x}_i$ for all $i \in I$. The corresponding objective value $\hat{z} = \sum_{i \in I} p_i \hat{x}_i$ is an upper bound on the optimal integral solution. As the values $\hat{x}_i$ can be fractional, we have to show how an integral solution can be derived.

2.2 Discretizing the LP Solution

As the next step, we round the values $\hat{x}_i$ down to values $\bar{x}_i$ that are integral multiples of $\frac{1}{N}$ for some integer N. This can be done without losing much in the objective function, i.e., for any given $\delta > 0$ we can achieve $\bar{z} = \sum_{i \in I} \bar{x}_i p_i \geq (\sum_{i \in I} \hat{x}_i p_i)/(1+\delta)$ by choosing N sufficiently large (depending on δ). For a fixed $\delta > 0$, the value of N can be chosen polynomial in the size of the input (using arguments explained in [10]).

Now consider the set I' of call instances that contains $N\bar{x}_i$ copies of every instance $i \in I$. Note that $\sum_{i \in I'} p_i = \sum_{i \in I} N\bar{x}_i p_i = N\bar{z}$. Our goal is to partition I' into $k \leq \alpha N$ feasible solutions A_1, A_2, ..., A_k, where we try to have α as small as possible. If we have achieved that, a pigeonhole argument shows that at least one of the solutions must have total profit at least $\frac{N\bar{z}}{k} \geq \frac{1}{\alpha}\bar{z}$. Let A_j be the feasible solution that maximizes the total profit among $A_1, \ldots, A_k$. Our algorithm outputs A_j and, consequently, achieves approximation ratio $\alpha(1+\delta)$.

In fact, it is possible to skip the rounding to multiples of $\frac{1}{N}$ and put the fractional call instances into I' directly. I' is then partitioned into fractions of feasible solutions, but otherwise the algorithms remain essentially the same. The advantage is that we can avoid losing a factor of $1+\delta$ due to the rounding to multiples of $\frac{1}{N}$ in this way. Nevertheless, the approach described above (with the rounding) is conceptually simpler and so we use it in this paper. For the formulation of our theorems, we assume that the fractional approach (without rounding) is used so that we do not lose the factor of $1+\delta$ in the approximation ratio.

1. Maintain feasible solutions $A_1, A_2, \ldots$; initially, all A_j are empty.
2. Consider the instances in group 1 in arbitrary order, and then the instances in group 2 in arbitrary order. Add each instance i to the solution A_j such that A_j remains feasible and j is minimized among all such solutions A_j.

Figure 1: Algorithm for groups 1 and 2.

2.3 Solving the Partitioning Problem

We want to partition I' into feasible (integral) solutions to the original instance of BCA or GCA. In the following we will refer to these feasible solutions as *bins*. From the way in which I' was obtained from a feasible (fractional) solution to LP_1, we know that I' contains at most N call instances belonging to the same call (from constraint (2)) and that $\sum_{i \in I'(e,t)} b_i \leq N \cdot B(e)$ for all $e \in E$ and all $t = 0, 1, \ldots, T$ (from constraint (3)). We refer to the value $\sum_{i \in I'(e,t)} b_i / B(e)$ as the load of edge e at time t. Thus constraint (3) tells us that the maximum load of an edge due to call instances in I' is at most N.

Partitioning for BCA In the case of BCA, the partitioning problem is related to call scheduling. Call scheduling in stars and trees was studied in [9]. However, there are some differences: While the makespan of the schedule produced by approximation algorithms for call scheduling is compared with an optimal schedule, we compare the number of bins used by a partitioning algorithm with the maximum load of an edge. Furthermore, different instances of the same call must not be put in the same bin, a constraint that does not exist for call scheduling (where there is only one instance of each call).

We proceed as follows. First, we classify the call instances in I' into the following groups. Group 1 contains all instances i such that $b_i > B(e)/2$ for all $e \in path(i)$. Group 2 contains all instances i such that $b_i \leq B(e)/2$ for all $e \in path(i)$. Group 3 contains all instances i using two edges e_1 and e_2 with $B(e_1) > B(e_2)$ and satisfying $b_i \leq B(e_1)/2$ and $b_i > B(e_2)/2$.

The instances in groups 1 and 2 are treated separately from the instances in group 3. For instances in groups 1 and 2, the partitioning algorithm shown in Fig. 1 is used.

Let k_1 be the number of non-empty solutions A_j after the execution of this algorithm. The proof of the following lemma is omitted due to lack of space.

Lemma 1 $k_1 \leq 5N$.

Now consider the call instances in group 3. Use the algorithm shown in Fig. 2 to partition them into feasible solutions. We remark that the order in which this algorithm processes the call instances in group 3 is essential: if they were processed in arbitrary order, no constant approximation ratio could be achieved.

1. Maintain feasible solutions $B_1, B_2, \ldots$; initially, all B_j are empty.
2. Let $e_1, e_2, \ldots, e_n$ be the edges of the star in order of non-increasing capacities, i.e., $B(e_1) \geq B(e_2) \geq \cdots \geq B(e_n)$.
3. For $q = 1, 2, \ldots, n$, do the following: let $I(e_q)$ be the set of all call instances i that use e_q, but do not use an edge with smaller index than e_q; consider the call instances $i \in I(e_q)$ in arbitrary order, and add each such instance to the solution B_j such that B_j remains feasible and j is minimized among all such solutions B_j.

Figure 2: Algorithm for group 3.

Let k_2 be the number of non-empty solutions B_j after the execution of this algorithm.

Lemma 2 $k_2 \leq 5N$.

Proof. Consider an instance i from group 3 and assume that i is put in solution B_j by the algorithm. Let e_q and e_r be the edges used by i, with $q < r$. By the definition of group 3, $b_i \leq B(e_q)/2$ and $b_i > B(e_r)/2$. At most $N-1$ solutions B_h are blocked by other instances of the same call. At most $2N$ solutions are blocked because less than b_i bandwidth is available on e_q (as the bandwidth occupied on e_q is at least $B(e_q) - b_i \geq B(e_q)/2$ in all these solutions). At most $2N$ solutions are blocked because less than b_i bandwidth is available on e_r: this is because every previously processed call instance using e_r occupies more than half the capacity of e_r; so whenever a solution is blocked on e_r, more than $B(e_r)/2$ of the capacity of e_r is occupied in that solution. Summing up, we obtain that $j \leq 5N$. No instance is added to a solution with index greater than $5N$, so we have $k_2 \leq 5N$. □

So we have partitioned I' into at most $10N$ feasible solutions. At least one of these feasible solutions must have total profit at least $\bar{z}/10$. (Otherwise, the total profit of all solutions would be smaller than $N\bar{z}$, a contradiction.) The feasible solution produced by our algorithm has a total profit that is at least a $1/10$-fraction of the optimal profit.

In the case of unit-BCA, there are no calls in group 3, and so we can partition I' into at most $5N$ feasible solutions.

Theorem 1 *For star networks, there is a* 10*-approximation algorithm for BCA and a* 5*-approximation algorithm for unit-BCA.*

Partitioning for GCA Now we consider the general case where the call instances in I' can have arbitrary starting times and durations. We say that a call instance i using edges e_1 and e_2 with $B(e_1) \geq B(e_2)$ is *big* if $b_i > B(e_1)/3$ (and thus $b_i > B(e_2)/3$), is *small* if $b_i \leq B(e_1)/3$ and $b_i \leq B(e_2)/2$, and is *mixed* if $b_i \leq B(e_1)/3$ and $b_i > B(e_2)/2$,

1. Maintain feasible solutions $A_1, A_2, \ldots$; initially, all A_j are empty.
2. Consider the instances in I' in the order of non-decreasing starting times. Add each instance i to the solution A_j such that A_j remains feasible and j is minimized among all such solutions A_j.

Figure 3: Partitioning algorithm for GCA.

We propose the greedy algorithm shown in Fig. 3 for partitioning I' into feasible solutions. Let k be the number of non-empty solutions at the end of the algorithm. The proof of the following lemma is omitted due to lack of space.

Lemma 3 $k = O(N \log R)$.

One of the solutions $A_1, \ldots, A_k$ must have total profit at least $\bar{z}/O(\log R)$. Thus we obtain approximation ratio $O(\log R)$ for the case of arbitrary edge capacities.

In the case of unit edge capacities ($B(e) = 1$ for all $e \in E$), there are no mixed calls. A call instance i with $b_i \leq \frac{1}{2}$ is blocked in at most $N-1$ solutions by other instances of the same call and in at most $4N$ solutions because an edge in $path(i)$ does not have enough bandwidth available at time t_i. Hence, such a call instance can be put in one of $5N$ solutions. A call instance i with $b_i > \frac{1}{2}$ can be blocked in at most $4N$ solutions by other call instances with bandwidth requirement greater than $\frac{1}{2}$ and in at most $N-1$ solutions by other instances of the same call. Hence it can be put in one of $10N$ solutions. Thus we can partition I' into at most $10N$ feasible solutions in the case of unit edge capacities.

Theorem 2 *For star networks, there is a* 10*-approximation algorithm for unit-GCA and an* $O(\log R)$*-approximation algorithm for GCA, where* R *is the ratio of the maximum edge capacity to the minimum edge capacity.*

3 Trees and Trees of Rings

In this section we present algorithms that achieve constant approximation ratio for unit-BCA in trees and trees of rings. We use the same basic approach as for star networks. We assume that $B(e) = 1$ for all edges e.

Let us consider tree networks first. An optimal fractional solution to LP_1 is computed and then the fractional values $\hat{x}_i$ are rounded to values $\bar{x}_i$ that are multiples of $\frac{1}{N}$ for some sufficiently large integer N. (Alternatively, the fractional approach without rounding that was mentioned in Sect. 2 can be used.) A set I' of call instances is obtained by taking $N\bar{x}_i$ copies of every instance $i \in I$.

Assume that the tree is rooted at an arbitrary node. For a call instance i, let $lca(i)$ be the least common ancestor of $path(i)$, i.e., the node that is closest to the root among all nodes on $path(i)$.

1. Maintain feasible solutions $A_1, A_2, \dots$; initially, all A_j are empty.
2. Consider the instances $i \in I'$ in the order of non-decreasing distance between $lca(i)$ and the root of the tree. Add each instance i to the solution A_j such that A_j remains feasible and j is minimized among all such solutions A_j.

Figure 4: Partitioning for BCA in tree networks.

The set I' is then partitioned into feasible solutions using the algorithm shown in Fig. 4. Let k be the number of non-empty solutions A_j produced by the algorithm.

Lemma 4 $k \leq 10N$.

Proof. Consider a call instance $i \in I'$. Assume that i is put into A_j by the algorithm. At most $N-1$ solutions A_h are blocked because they contain another instance of the same call.

Let $v = lca(i)$. Let e_1 and e_2 be the two edges incident to v that are on $path(i)$. (If there is only one such edge, the argument is simpler.) Assume that i is blocked from some solution A_h because not enough bandwidth is available. Because of the order in which the algorithm processes the path instances, either e_1 or e_2 does not have enough bandwidth available. If $b_i \leq \frac{1}{2}$, there can be at most $4N$ such solutions A_h, because in each of these solutions either e_1 or e_2 is used by calls with total bandwidth greater than $\frac{1}{2}$ and because the maximum load is at most N. As at most $N-1$ solutions are blocked by other instances of the same call as i, we have $j \leq 5N$ in this case. If $b_i > \frac{1}{2}$, let q be the maximum index of a set A_q, $q < j$, such that a call i' with bandwidth $b_{i'} \leq \frac{1}{2}$ uses e_1 or e_2 in A_q. We have $j - q \leq 5N$, because in each solution A_r, $q < r < j$, e_1 or e_2 is used by a call with bandwidth greater than $\frac{1}{2}$ or A_r contains another instance of the same call. As $b_{i'} \leq \frac{1}{2}$, the above argument shows that $q \leq 5N$, and we can conclude that $j \leq 10N$ and, therefore, $k \leq 10N$. □

Theorem 3 *There is a* 10*-approximation algorithm for unit-BCA in trees.*

Now we consider trees of rings. Unlike in stars, lines, or trees, there are several different paths between any pair of nodes in a tree of rings. It is no longer sufficient to decide which requests to accept and which to reject; we must also determine the paths along which the accepted connections are established.

First, we sketch how LP_1 can be modified for trees of rings. We view the network as a flow network and each call instance i as a commodity that is to be routed from u_i to v_i. The variable x_i represents the amount of commodity i that is routed from u_i to v_i. We introduce additional variables x_{ie} that represent the amount of commodity i that is routed through edge e. Furthermore, we add flow conservation constraints (for nodes v different from u_i and v_i, the amount of commodity i that reaches v must be equal to the amount of commodity i that leaves v) to the LP formulation. In this way, we can still obtain a fractional optimal solution to the linear programming relaxation. Using path stripping, we obtain a set of fractional paths for each call instance. Each fractional path for a call instance constitutes a *routed call instance*. Again, we round all fractions to multiples of $\frac{1}{N}$ and multiply by N to obtain a new set I' of routed call instances.

The set I' is partitioned into feasible solutions using the same basic approach as for trees (Fig. 4), but the order in which the algorithm processes the routed call instances is as follows: Consider a depth-first search traversal of the tree of rings and let $pre(v)$ denote the pre-order number of v (i.e., the nodes are numbered in the order in which they are discovered by the DFS). For each routed call instance i, let $pre(i)$ be the minimum pre-order number among all nodes on the corresponding path. Then the routed call instances are processed in order of non-decreasing $pre(i)$ (similar to the path coloring algorithm for trees of rings in [8]).

We claim that I' is partitioned into at most $18N$ solutions. Consider a routed call instance i that is put into A_j. Let v be the node with $pre(v) = pre(i)$. See Fig. 5, where the shaded nodes are the nodes with smaller pre-order number than v. There are two basic types for the path of i: it can be a path that does not use an edge in the ring of v that contains shaded nodes (for example, a path from a to b), or it can be a path that uses at least one edge in that ring (for example, a path from b to c). In either case, there are four edges such that any previously assigned path that intersects the path of i must use at least one of these edges: if the path of i is of the first type, these are the edges numbered 1 to 4; if the path is of the second type, these are the edges numbered 3 to 6. Consider the case that $b_i \leq \frac{1}{2}$. If i is blocked in a solution A_h because of bandwidth, one of the edges on the path of i must have less than $\frac{1}{2}$ bandwidth available in A_h. But all paths in A_h that intersect i must use one of four particular edges, so A_h contributes at least $\frac{1}{2}$ to the load on these four edges. So there can be at most $8N$ such solutions A_h. As there are at most $N-1$ other instances of the same call, we have $j \leq 9N$.

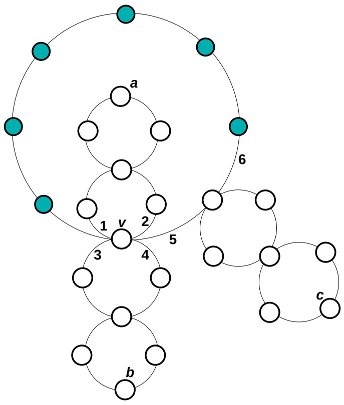

Figure 5: Tree of rings.

Now assume that $b_i > \frac{1}{2}$. There can be at most $8N$ solutions in which i is blocked because the path of another call with bandwidth larger than $\frac{1}{2}$ intersects the path of i, and at most $N-1$ solutions in which i is blocked because of another instance of the same call. So we get $j \leq 9N + 9N = 18N$. Hence, I' is partitioned into at most $18N$ feasible solutions, and we obtain the following theorem.

Theorem 4 *There is an* 18*-approximation algorithm for unit-BCA in trees of rings.*

4 Conclusion

We have presented approximation algorithms for optimization problems modeling call admission control with advance reservations and call alternatives. The algorithms achieve an approximation ratio that is either constant or of the order $O(\log R)$, where R is the ratio of the maximum edge capacity to the minimum edge capacity of the network. Our results can easily be adapted to the case of bidirected stars, trees, and trees of rings.

Concerning future work, it would be interesting to see whether the $O(\log R)$-approximation algorithm for GCA in stars can be improved to constant ratio and whether there are constant-factor approximation algorithms for BCA (with arbitrary capacities) or GCA in lines, trees, or trees of rings. Besides, it might be possible to extend the approach to other network topologies. An interesting generalization of GCA would be to consider the case where the capacity of an edge can change over time, i.e., the capacity of an edge e at time t is given by an arbitrary value $B(e,t)$. This problem is encountered in a network with advance reservation if some reservations have already been made (thus occupying part of the bandwidth for certain time periods) and a new group of reservation requests is to be processed.

Acknowledgments. The author is grateful to Chirdeep Chhabra, Stamatis Stefanakos, Burkhard Stiller, and Danica Vukadinović for interesting and helpful discussions in the context of the projects AAPCN and PCAC.

References

[1] AWERBUCH, B., AZAR, Y., AND PLOTKIN, S. Throughput-competitive on-line routing. In *Proc. 34th Ann. Symp. on Foundations of Computer Science FOCS* (1993), pp. 32–40.

[2] AZAR, Y., AND REGEV, O. Strongly polynomial algorithms for the unsplittable flow problem. In *Proc. 8th Integer Programming and Combinatorial Optimization Conference IPCO* (2001), LNCS 2081, pp. 15–29.

[3] BAR-NOY, A., BAR-YEHUDA, R., FREUND, A., NAOR, J., AND SCHIEBER, B. A unified approach to approximating resource allocation and scheduling. *J. ACM 48*, 5 (2001), 1069–1090.

[4] BAR-NOY, A., GUHA, S., NAOR, J., AND SCHIEBER, B. Approximating the throughput of multiple machines in real-time scheduling. *SIAM J. Comput. 31*, 2 (2001), 331–352.

[5] BORODIN, A., AND EL-YANIV, R. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[6] CHHABRA, C., ERLEBACH, T., STILLER, B., AND VUKADINOVIĆ, D. Price-Based Call Admission Control in a Single DiffServ Domain. TIK-Report 135, ETH Zürich, May 2002.

[7] DUFFIELD, N., GOYAL, P., AND GREENBERG, A. A flexible model for resource management in virtual private networks. In *ACM SIGCOMM* (1999).

[8] ERLEBACH, T. Approximation algorithms and complexity results for path problems in trees of rings. In *Proc. 26th Int. Symp. on Mathematical Foundations of Computer Science MFCS* (2001), LNCS 2136, pp. 351–362.

[9] ERLEBACH, T., AND JANSEN, K. Off-line and on-line call-scheduling in stars and trees. In *Proc. 23rd Int. Workshop on Graph-Theoretic Concepts in Computer Science WG* (1997), LNCS 1335, pp. 199–213.

[10] ERLEBACH, T., AND JANSEN, K. Conversion of coloring algorithms into maximum weight independent set algorithms. In *ICALP Workshops 2000* (2000), Proceedings in Informatics 8, Carleton Scientific, pp. 135–145. Workshop on Approximation and Randomization Algorithms in Communication Networks ARACNE 2000.

[11] GURUSWAMI, V., KHANNA, S., RAJARAMAN, R., SHEPHERD, B., AND YANNAKAKIS, M. Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. In *Proc. 31st Ann. ACM Symp. on Theory of Computing STOC* (1999), pp. 19–28.

[12] IBARRA, O. H., AND KIM, C. E. Fast approximation algorithms for the knapsack and sum of subset problems. *J. ACM 22*, 4 (1975), 463–468.

[13] KLEINBERG, J. *Approximation algorithms for disjoint paths problems*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1996.

[14] LEONARDI, S. On-line network routing. In *Online Algorithms: The State of the Art*, A. Fiat and G. J. Woeginger, Eds., LNCS 1442. Springer-Verlag, Berlin, 1998.

[15] LEWIN-EYTAN, L., NAOR, J., AND ORDA, A. Routing and admission control in networks with advance reservations. In *Proc. 5th Int. Workshop on Approximation Algorithms for Combinatorial Optimization APPROX* (2002), LNCS. To appear.

[16] PHILLIPS, C. A., UMA, R., AND WEIN, J. Off-line admission control for general scheduling problems. *Journal of Scheduling 3* (2000), 365–381.

Thomas Erlebach is with the Computer Engineering and Networks Laboratory (TIK), Department of Information Technology and Electrical Engineering, ETH Zürich, CH-8092 Zürich, Switzerland. E-mail: erlebach@tik.ee.ethz.ch