Off-line Admission Control for Advance Reservations in Star Networks

Udo Adamy[1], Thomas Erlebach[2,⋆], Dieter Mitsche[1], Ingo Schurr[1,⋆⋆], Bettina Speckmann[3], and Emo Welzl[1]

[1] Institute for Theoretical Computer Science, ETH Zürich, 8092 Zürich, Switzerland
{adamy, dmitsche, schurr, welzl}@inf.ethz.ch

[2] Department of Computer Science, University of Leicester, Leicester LE1 7RH, U.K
t.erlebach@mcs.le.ac.uk

[3] Department of Mathematics and Computer Science, TU Eindhoven, 5600 MB Eindhoven, The Netherlands
speckman@win.tue.nl

Abstract. Given a network together with a set of connection requests, call admission control is the problem of deciding which calls to accept and which ones to reject in order to maximize the total profit of the accepted requests. We consider call admission control problems with advance reservations in star networks. For the most general variant we present a constant-factor approximation algorithm resolving an open problem due to Erlebach. Our method is randomized and achieves an approximation ratio of 1/18. It can be generalized to accommodate call alternatives, in which case the approximation ratio is 1/24. We show how our method can be derandomized. In addition we prove that call admission control in star networks is $\mathcal{APX}$-hard even for very restricted variants of the problem.

1 Introduction

Call admission control (CAC) is a fundamental problem in the operation of communication networks. In its general form each connection request (call) has a certain bandwidth requirement and some time specification given by its starting time and its duration. If the network establishes a call, it first decides on a path from the sender to the receiver through which the call is being routed. Then it allocates the requested amount of bandwidth on all links along that path during the time period in which the call is active. In addition, each call is associated

⋆ Work mainly done while the author was with ETH Zürich, supported by the Swiss National Science Foundation under Contract No. 21-63563.00 (Project AAPCN) and the EU Thematic Network APPOL II (IST-2001-32007), with funding provided by the Swiss Federal Office for Education and Science.

⋆⋆ Supported by the joint Berlin/Zürich graduate program Combinatorics, Geometry, and Computation (CGC), financed by the German Science Foundation (DFG) and ETH Zürich.

with some profit, which is gained by the network provider only if the desired connection is established. CAC is the problem of deciding which calls to accept and which to reject with the goal of maximizing the total profit accrued by the accepted requests.

In this paper we consider CAC problems in star networks with advance reservations. In star networks there is always a unique path for each sender–receiver pair, and the routing issue mentioned above is trivial. Advance reservation means that resources are requested in advance of when they are really needed. In this setting the decision of acceptance or rejection of calls does not have to be made on-line, i.e., immediately when a new call arrives, but can be made off-line. The network can collect incoming requests over some period of time and then decide for the set of collected calls which of them to accept and which to reject. Advance reservations are therefore helpful for the network in optimizing its CAC decisions.

We also deal with call alternatives. In this scenario the user can specify several alternative requests for a connection to be established. The network can either accept exactly one of the call alternatives, or reject a call completely by rejecting all its alternatives. By specifying several alternatives the user can increase the chances that the network accepts the call.

Problem Definition. A star network consists of a set of outer nodes, each of which is connected exclusively to a unique center node. It is modeled by a simple, undirected graph $G = (V, E)$, whose vertex set $V = \{0, 1, \ldots, n\}$ represents the nodes of the network with vertex 0 being the center node. The edge set E consists of the edges $e_i = \{i, 0\}$ for $i = 1, \ldots, n$ corresponding to the links in the star network that connect an outer vertex i to the center node 0. The capacities of the links are given as a capacity function $c : E \mapsto \mathbf{R}^+$ that maps each edge $e \in E$ to a positive capacity $c(e)$.

A connection request or *call* i is specified by a tuple $(u_i, v_i, t_i, d_i, b_i, p_i)$ consisting of a source node $u_i \in V$, a destination node $v_i \in V$, a starting time $t_i \in \mathbf{N}$, a duration $d_i \in \mathbf{N}$, a positive bandwidth requirement $b_i \in \mathbf{R}^+$, and a profit $p_i \in \mathbf{R}$. For a set $\mathcal{R}$ of connection requests, a solution is a subset $Q \subseteq \mathcal{R}$ of accepted calls. It is *feasible* if the sum of bandwidth requirements of simultaneously active calls using the same edge does not exceed the capacity of that edge:

$$\forall e \in E : \forall t \in \mathbf{N} : \sum_{i \in Q(e,t)} b_i \leq c(e),$$

where $Q(e, t)$ is the set of accepted calls that use the edge e at time t. Our goal is to compute a feasible solution that maximizes the sum $\sum_{i \in Q} p_i$ of profits of the accepted calls. We refer to this problem as the general call admission control problem in star networks, or GCA in stars for short.

GCA in stars is an $\mathcal{NP}$-hard problem. If we drop the time specifications of the calls and restrict the network to consist of a single edge only, it is equivalent to the KNAPSACK problem, which is known to be $\mathcal{NP}$-hard [12]. Therefore, we are interested in finding good approximate solutions. A feasible solution for GCA is called a ρ-approximation, $\rho \leq 1$, if its total profit is at least a ρ-fraction of the profit of an optimal solution. An algorithm is called a ρ-approximation algorithm if it runs in polynomial time and always outputs a ρ-approximation. The parameter ρ is called the approximation ratio of such an algorithm.

Motivation. A lot of work on CAC considered the scenario that a connection request is presented to the network only at the time when the connection should be established. This model ignores the possibility that resources might be requested ahead of time when they are needed. A natural concept to incorporate this possibility is advance reservation [13, 18], which is attractive for both the network provider and the users. The network provider has the advantage of a higher flexibility and can use better algorithms to maximize the profit gained from the accepted calls. The user benefits from a guaranteed quality of service (QoS) once his/her call is accepted.

If we omit the time specification of the calls, GCA contains the maximum edge-disjoint paths (MEDP) problem as a special case (see [16] for more on this problem). Since already the MEDP problem is hard to approximate in general directed graphs [14], it is natural to restrict the network topology to simpler classes. In this paper we restrict ourselves to star networks. Although star networks are a very restricted class of networks, we note that with the results we present in this paper, star networks are now the most general class of networks for which a constant-factor approximation algorithm for GCA is known; even for line networks, i.e., networks consisting of a single path, the existence of a constant-factor approximation algorithm is an open problem (GCA in lines includes the maximum independent set problem in rectangle intersection graphs as a special case; see [5] for the best known approximation results for the latter problem).

Interestingly, CAC algorithms for star networks 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 [9], one requests a logical network connecting a set of terminal nodes and specifies for each terminal node the maximum rates at which traffic will ever be transmitted or received by that node. The provider has to reserve sufficient capacity for the logical network to ensure that any traffic matrix consistent with the specifications can be accommodated. Therefore, with respect to the available bandwidth, the logical network behaves like a star network with the terminal nodes as outer nodes. Thus, if a provider of a video-on-demand system, say, has rented capacity for the distribution network according to the hose model, the handling of advance reservations for video transmissions leads naturally to the problem of GCA in star networks.

It is an important task to investigate CAC problems with arbitrary capacities on the links of the network. For line networks, for example, there is a constant-factor approximation algorithm in the case of uniform capacities on the edges and no time specifications for the calls [3]. However, if we allow arbitrary edge capacities and assume no restrictions on the bandwidth requirements, no constant-factor approximation algorithm is known so far. Constant-factor approximations for this problem are known only under the "no-bottleneck" assumption, which requires that the maximum bandwidth requirement is not larger than the minimum edge capacity [7, 8].

Similarly, for star networks there is a constant-factor approximation algorithm for GCA with unit edge capacities, and only a logarithmic approximation for the case of arbitrary edge capacities. These algorithms were presented in a paper of Erlebach [11], who raised the open question whether there is a constant-factor approximation algorithm for GCA in stars.

Our Results. We resolve this open question by presenting the first constant-factor approximation algorithm for GCA in star networks. Our method is randomized and achieves an approximation ratio of 1/18. It is fast, because it does not involve linear programming. Instead, our algorithm is based on the local ratio technique of [3] and can easily be derandomized. We can also modify our algorithm to handle call alternatives and obtain a 1/24-approximation algorithm for this case. On the other hand, we prove that GCA in stars is $\mathcal{APX}$-hard even for very restricted versions of the problem.

Related Work. Many authors have investigated CAC problems for various network topologies in the on-line and off-line setting. In the on-line scenario each request must be accepted or rejected immediately without knowing the future events. We refer to [17, 6] for surveys about on-line CAC algorithms.

GCA for various network topologies is studied in a paper by Erlebach [11]. For trees and trees of rings with unit edge capacities he presents constant-factor approximation algorithms for CAC without time specifications for the calls. For star networks he obtains a 1/10-approximation algorithm both for GCA without time specifications and for GCA with unit edge capacities. For GCA in stars with arbitrary edge capacities he achieves a $1/O(\log R)$-approximation algorithm, where R is the ratio of the maximum edge capacity to the minimum edge capacity. All these algorithms are obtained by first solving a linear programming relaxation and then decomposing the optimal fractional solution into a convex combination of integral solutions. The output of the algorithm is the best of the integral solutions obtained this way.

Bar-Noy et al. [3] extend the so-called local ratio technique [4] to resource allocation and scheduling problems. They obtain a $\frac{1}{5}$-approximation algorithm for GCA on a single link. Their approach is also used by Lewin-Eytan et al. [18] to obtain results for lines, rings, and trees. For trees with unit edge capacities, they present a $\frac{1}{5}$-approximation for GCA without time specifications and a $1/O(\log m)$-approximation algorithm for GCA, where m is the number of requests.

2 Preliminaries: The Local Ratio Technique

Our approximation algorithm is based on the local ratio technique, which was developed by Bar-Yehuda and Even [4] and first used in our context by Bar-Noy et al. [3]. The general framework can be described as follows. Assume that we are given a profit vector $\boldsymbol{p} \in \mathbf{R}^n$ and a set $\mathcal{F}$ of feasibility constraints on vectors $\boldsymbol{x} \in \mathbf{R}^n$. A vector $\boldsymbol{x} \in \mathbf{R}^n$ is a *feasible* solution for the problem $(\mathcal{F}, \boldsymbol{p})$ if it satisfies all the constraints in $\mathcal{F}$. The *value* of a feasible solution $\boldsymbol{x}$ is the inner product $\boldsymbol{p} \cdot \boldsymbol{x}$. For a maximization problem, a feasible solution $\boldsymbol{x}$ is an r-approximation if $\boldsymbol{p} \cdot \boldsymbol{x} \geq r \cdot \boldsymbol{p} \cdot \boldsymbol{x}^*$, where $\boldsymbol{x}^*$ is an *optimal* solution, i.e., a solution whose value is maximal among all feasible solutions. The power of the technique is based on the following theorem.

Theorem 1. (Local Ratio [4, 3]) *Let $\mathcal{F}$ be a set of constraints and let $\boldsymbol{p}$, $\boldsymbol{p}'$ and $\boldsymbol{p}''$ be profit vectors such that $\boldsymbol{p} = \boldsymbol{p}' + \boldsymbol{p}''$. Then, if $\boldsymbol{x}$ is an r-approximation with respect to $(\mathcal{F}, \boldsymbol{p}')$ and with respect to $(\mathcal{F}, \boldsymbol{p}'')$, then $\boldsymbol{x}$ is an r-approximation with respect to $(\mathcal{F}, \boldsymbol{p})$.*

For computing approximate solutions we will use the so-called *unified algorithm* proposed in [3]. We generalize our problem by allowing negative profits as well.

The Unified Algorithm:

1. Delete all calls with non-positive profit.
2. If no calls remain, return $Q = \emptyset$.
3. Otherwise, select a call i and decompose $\boldsymbol{p} = \boldsymbol{p}' + \boldsymbol{p}''$. The choice of i and the decomposition depends on the problem at hand. For the GCA problem, we will always select i as a call with minimum end-time $t_i + d_i$, breaking ties arbitrarily.
4. Solve the problem recursively using $\boldsymbol{p}''$ as the profit vector. Let Q'' be the set returned.
5. If $Q'' \cup \{i\}$ is a feasible solution, return $Q = Q'' \cup \{i\}$. Otherwise, return $Q = Q''$.

The decomposition of $\boldsymbol{p}$ into $\boldsymbol{p}'$ and $\boldsymbol{p}''$ in step 3 will be specified separately for each problem to which we apply the algorithm.

We call a feasible solution *i-maximal* if it either contains the call i, or it does not contain the call i, but adding i to the solution will render it infeasible. The following lemma analyzes the quality of the solution produced by the algorithm.

Lemma 1. ([3]) *Let r be a constant. Suppose that in the algorithm above the choice of i and the decomposition $\boldsymbol{p} = \boldsymbol{p}' + \boldsymbol{p}''$ is always such that: (1) $p''_i = 0$, and (2) every i-maximal solution is an r-approximation with respect to $\boldsymbol{p}'$. Then, the algorithm terminates and the solution Q produced is an r-approximation with respect to $\boldsymbol{p}$.*

Proof. First note that since $p''_i = 0$, the call i will be deleted in the recursive call, and the algorithm will eventually terminate. Because the deletion of calls with non-positive profit in step 1 does not change the optimum value, it is sufficient to show that Q is an r-approximation with respect to the remaining calls. Since Q is i-maximal, the condition (2) implies that Q is an r-approximation with respect to $\boldsymbol{p}'$. It remains to show that Q is also an r-approximation with respect to $\boldsymbol{p}''$ (and then apply the local ratio theorem). We proceed by induction on the number of recursive calls of the algorithm. At the basis of the recursion, the algorithm returns the empty set, since no calls remain. This is clearly an optimal solution and, hence, also an r-approximation. For the induction step, assume that Q'' is an r-approximation with respect to $\boldsymbol{p}''$. Since $p''_i = 0$ and Q is either Q'' or $Q'' \cup \{i\}$, it follows that Q is an r-approximation with respect to $\boldsymbol{p}''$ as well. By the local ratio theorem (Theorem 1), Q is also an r-approximation with respect to $\boldsymbol{p}$. □

By Lemma 1 we only need to find a constant $r > 0$ such that every i-maximal solution is an r-approximation with respect to $\boldsymbol{p}'$. To do so, we will derive an upper bound p_{opt} on the optimum $\boldsymbol{p}'$-profit and a lower bound p_{max} on the $\boldsymbol{p}'$-profit of every i-maximal solution. The ratio $r = p_{max}/p_{opt}$ is then a lower bound on the approximation ratio of the algorithm.

3 Constant-Factor Approximation for GCA in Stars

In this section we present a 1/18-approximation algorithm for GCA in star networks. Without loss of generality, we can assume that every call in the set $\mathcal{R}$ uses exactly two edges. We partition the calls in $\mathcal{R}$ into three classes according to their bandwidth requirements. Consider a call $i \in \mathcal{R}$ and denote by e and f the edges that it uses. We classify the call i to be

– a *small* call, if it uses at most half of the capacity on both of its edges, i.e., $b_i \le c(e)/2$ and $b_i \le c(f)/2$,
– a *big* call, if it uses more than half of the capacity on both of its edges, i.e., $b_i > c(e)/2$ and $b_i > c(f)/2$,
– a *mixed* call, otherwise.

We denote the set of small, big, and mixed calls by $\mathcal{R}_{small}$, $\mathcal{R}_{big}$, and $\mathcal{R}_{mixed}$, respectively. We give a constant-factor approximation for each of the three classes. The algorithm for the original problem is then as follows. Partition the calls in $\mathcal{R}$ into the three classes and solve the problem for each of the classes separately. Output the solution with the largest profit among the three solutions for the classes.

Small Calls. We approximate this set using the unified algorithm from above. Let i be the call with minimum end-time $t_i + d_i$ selected by the algorithm in step 3 and denote by e and f the edges that call i uses, such that $c(e) \le c(f)$. We decompose the profit function $\boldsymbol{p} = \boldsymbol{p}' + \boldsymbol{p}''$ by defining $\boldsymbol{p}'$ according to

$$p'_j = p_i \cdot \begin{cases} 1 & \text{if } j = i, \\ \alpha \cdot (c(f) - b_i) \cdot b_j & \text{if } i \text{ and } j \text{ intersect on edge } e \text{ at time } t_i + d_i, \\ \alpha \cdot (c(e) - b_i) \cdot b_j & \text{if } i \text{ and } j \text{ intersect only on edge } f \text{ at time } t_i + d_i, \\ 0 & \text{otherwise,} \end{cases}$$

where α is a parameter that will be determined later. Note that $p''_i = 0$. The values of the profit function $\boldsymbol{p}'' = \boldsymbol{p} - \boldsymbol{p}'$ may be non-positive.

Lemma 2. *The set of small calls admits an approximation ratio of* 1/4.

Proof. Due to Lemma 1 it suffices to show that every i-maximal solution is a 1/4-approximation with respect to $\boldsymbol{p}'$. Let Q^* be a $\boldsymbol{p}'$-optimal solution. In case i is not in Q^*, the contribution of other calls in Q^* using edge e is at most $p_i \cdot \alpha \cdot (c(f) - b_i) \cdot c(e)$, since the total bandwidth of all these calls intersecting at time $t_i + d_i$ is at most $c(e)$. Likewise the contribution on edge f is at most $p_i \cdot \alpha \cdot (c(e) - b_i) \cdot c(f)$. In case i belongs to Q^*, the call i adds p_i to the $\boldsymbol{p}'$-profit of Q^*, and uses b_i of the capacity of the edges e and f. The contribution of other calls in Q^* to the $\boldsymbol{p}'$-profit is then at most $p_i \cdot \alpha \cdot (c(f) - b_i) \cdot (c(e) - b_i)$ on edge e, and $p_i \cdot \alpha \cdot (c(e) - b_i) \cdot (c(f) - b_i)$ on edge f.

If we set $X = (c(f) - b_i) \cdot (c(e) - b_i)$, then the $\boldsymbol{p}'$-profit of Q^* is at most

$$p_i \cdot \max\{\alpha \cdot (c(f) - b_i) \cdot c(e) + \alpha \cdot (c(e) - b_i) \cdot c(f), 1 + 2 \cdot \alpha \cdot X\}.$$

To derive a lower bound on the $\boldsymbol{p}'$-value of an i-maximal solution Q, we distinguish two cases. If the call i is in Q, it contributes p_i itself. Thus the $\boldsymbol{p}'$-profit of Q is at least p_i. Otherwise, the call i is blocked by other calls in Q. This means that the total bandwidth of the calls preventing call i from being accepted must be greater than $c(e) - b_i$ on edge e or greater than $c(f) - b_i$ on edge f, respectively. Hence, the $\boldsymbol{p}'$-profit of the blocking calls is at least $p_i \cdot \alpha \cdot X$ (no matter on which of the edges e or f the call i is blocked). The minimum $p_i \cdot \min\{1, \alpha \cdot X\}$ of both expressions is a lower bound on the $\boldsymbol{p}'$-value of every i-maximal solution. Altogether, the approximation ratio r is given by

$$r = \frac{\min\{1, \alpha \cdot X\}}{\max\{\alpha \cdot (c(f) - b_i) \cdot c(e) + \alpha \cdot (c(e) - b_i) \cdot c(f), 1 + 2 \cdot \alpha \cdot X\}},$$

which for $\alpha = \frac{1}{X} = \frac{1}{(c(f)-b_i)\cdot(c(e)-b_i)}$ gives $r = 1/\max\left\{\frac{c(e)}{c(e)-b_i} + \frac{c(f)}{c(f)-b_i}, 3\right\}$. Finally, the fact that i is a small call, i.e., $b_i \leq c(e)/2$ and $b_i \leq c(f)/2$, implies that r is at least 1/4. □

Big Calls. In the case of big calls, no two calls using the same edge simultaneously can be in a feasible solution. Therefore we may assume that $b_j = 1$ for all big calls j and that $c(e) = 1$ for all edges $e \in E$. To define the decomposition $\boldsymbol{p} = \boldsymbol{p}' + \boldsymbol{p}''$, we set $\boldsymbol{p}'$ to be

$$p'_j = p_i \cdot \begin{cases} 1 & \text{if } j = i \text{ or } i \text{ and } j \text{ intersect at time } t_i + d_i, \\ 0 & \text{otherwise.} \end{cases}$$

The proof of the following lemma is similar to (but easier than) the proof of Lemma 2.

Lemma 3. *The set of big calls admits an approximation ratio of* 1/2.

Mixed Calls. Mixed calls use at most half of the capacity of one of their two edges, and need more than half of the capacity of the other edge. They cause the logarithmic factor in the approximation ratio shown by Erlebach in [11]. We briefly sketch the reason for this. Erlebach defines mixed calls as calls using at most 1/3 of the capacity of one of their edges and more than 1/2 of the capacity of the other. For an edge e, some mixed calls using e may use at most 1/3 of the capacity of e, while others may use more than 1/2. For a call using at most 1/3 of the capacity of edge e, Erlebach [11] has to argue about the second edge f used by this call; on that edge f, the call uses more than half of the capacity. If there is another call using at most 1/3 of the capacity of that edge f, he has to argue about the second edge of that call, and so on. The capacity of the edges considered in this chain of arguments decreases by a factor of 2/3 in each step; thus the number of steps is $O(\log R)$, where R is the ratio of the maximum to the minimum edge capacity, and this factor enters into the approximation ratio.

Here, we present a randomized procedure to approximate the mixed calls within a constant factor of the optimum. Our method can easily be derandomized, as we will discuss later on. We perform the following random experiment on the star network. Every outer node $v \in \{1, \dots, n\}$ is put independently with probability 1/2 in a set A and with probability 1/2 in a set B. Then we consider only those mixed calls that have one endpoint in the set A and the other endpoint in the set B. We denote this set of calls by $\mathcal{R}_{A,B}$. The probability that a mixed call belongs to $\mathcal{R}_{A,B}$ is exactly 1/2. This implies that the expected profit of an optimal solution for the set $\mathcal{R}_{A,B}$ is at least half the profit of an optimal solution for all mixed calls. Hence, we lose only a factor of 2 in expectation.

After this random experiment, all calls in $\mathcal{R}_{A,B}$ connect a vertex in A with a vertex in B. For a call $i \in \mathcal{R}_{A,B}$, let $e(i)$ denote the edge of the path of i connecting its endpoint in A with the center node 0, and let $f(i)$ denote the edge of the path between the center node and its endpoint in B. We further partition the calls in $\mathcal{R}_{A,B}$ according to their bandwidths. If a mixed call $i \in \mathcal{R}_{A,B}$ uses more than half of the capacity of the edge $e(i)$ (and at most half of the capacity of the edge $f(i)$), we put the call i in the set $\mathcal{R}_A$. Otherwise, we put the call i in the set $\mathcal{R}_B$. By considering $\mathcal{R}_A$ and $\mathcal{R}_B$ separately, we avoid the problem encountered in the analysis in [11], i.e., we do no longer have to deal with mixed calls occupying a large fraction and those occupying a small fraction of the capacity of the same edge at the same time.

Lemma 4. *Each of the sets $\mathcal{R}_A$ and $\mathcal{R}_B$ admits an approximation ratio of* 1/3.

Proof. To approximate the sets $\mathcal{R}_A$ and $\mathcal{R}_B$ we again use the unified algorithm. For the set $\mathcal{R}_A$ we decompose $\boldsymbol{p} = \boldsymbol{p}' + \boldsymbol{p}''$ by specifying $\boldsymbol{p}'$ to be

$$p'_j = p_i \cdot \begin{cases} 1 & \text{if } j = i, \\ \alpha/2 & \text{if } i \text{ and } j \text{ intersect on edge } e(i) \text{ at time } t_i + d_i, \\ \alpha \cdot \frac{b_j}{c(f(i))} & \text{if } i \text{ and } j \text{ intersect only on edge } f(i) \text{ at time } t_i + d_i, \\ 0 & \text{otherwise.} \end{cases}$$

A $\boldsymbol{p}'$-optimal solution accepts on edge $e(i)$ either the call i with $\boldsymbol{p}'$-profit p_i or one other call that intersects call i on the edge $e(i)$ and gives profit $p_i \cdot \alpha/2$. On the edge $f(i)$ the total bandwidth of accepted calls is bounded by $c(f(i))$, yielding a bound of $p_i \cdot \alpha$ on the optimal $\boldsymbol{p}'$-profit on that edge. An i-maximal solution contains either the call i with $\boldsymbol{p}'$-profit p_i or is blocked by calls with profit at least $p_i \cdot \alpha/2$. Hence, the approximation ratio r is given by

$$r = \frac{\min\{1, \alpha/2\}}{\max\{1, \alpha/2\} + \alpha},$$

which, for $\alpha = 2$, gives $r = 1/3$.

By symmetry, the set $\mathcal{R}_B$ can be approximated with the same ratio. □

After approximating both sets $\mathcal{R}_A$ and $\mathcal{R}_B$ by the unified algorithm, we output the solution that has a larger profit.

Corollary 1. *The set of mixed calls admits an approximation ratio of* $1/12$.

Proof. The larger solution for the sets $\mathcal{R}_A$ and $\mathcal{R}_B$ has profit at least $1/6$ times the profit of an optimal solution for the set $\mathcal{R}_{A,B}$, which, in expectation, is a factor $1/2$ away from the optimal solution for all mixed calls. □

We have approximation ratio $1/4$, $1/2$, and $1/12$ for the small calls, big calls, and mixed calls, respectively. As our algorithm outputs the solution with largest profit among the three individual solutions, we obtain the following theorem.

Theorem 2. *The above algorithm is a* $1/18$*-approximation algorithm for GCA in stars.*

3.1 Derandomization

The random process for filtering the mixed calls can be derandomized by reducing the size of the sample space. We refer to [1–Chapter 15] for an overview. In the analysis of our randomized algorithm, we used two properties of our random assignment. Firstly, the probability that a node v is assigned to the set A is $1/2$, and secondly the pairwise independence of the events, which guarantees that each call "survives" the experiment (i.e., its two endpoints are put into different sets) with probability $1/2$.

We employ a linear-size sample space that preserves these two properties. If we choose assignments uniformly at random from this sample space, our previous analysis remains valid. Thus, if we exhaustively search the sample space, we are guaranteed to find an assignment of nodes to the sets A and B such that the profit of an optimal solution for calls in $\mathcal{R}_{A,B}$ is at least half the profit of an optimal solution for all mixed calls. The construction of the linear-size sample space is omitted due to space limitations.

3.2 Call Alternatives

Our approximation algorithm can be generalized to accommodate alternatives for the calls with only a slight decrease of the approximation ratio. Call alternatives allow the specification of several alternatives for establishing a connection request. For example, a connection can be established either from 8:00 a.m. to

11:00 a.m. gaining profit p_1 or from 2:00 p.m. to 4:00 p.m. with profit p_2. We allow all parameters (including source node, destination node, and bandwidth requirement) of different alternatives of a call to be different. A solution obeying the capacity constraints is feasible if it contains at most one of the alternatives per call. The goal is to find the feasible solution with the largest profit. For this generalized problem, we use exactly the same algorithm as above and change only the decomposition of the profit function in the unified algorithm for each class of calls. In this case, the approximation ratios for the set of small calls, big calls, and mixed calls become 1/5, 1/3, and 1/16, respectively. This leads to the following theorem.

Theorem 3. *GCA in stars with call alternatives admits an approximation ratio of* 1/24.

4 $\mathcal{APX}$-Hardness

In this section we prove that already a very restricted variant of the general call admission control problem in stars is $\mathcal{APX}$-hard. The problem variant we consider is the following. We are given a star network with unit capacity on every edge, and a set $\mathcal{R}$ of calls. Each call i is associated with a starting time $t_i \in \{0, 1, 2\}$, has unit profit, and needs one unit of bandwidth on its edges. Every call i has duration $d_i = 2$. The goal is to compute a feasible subset $Q \subseteq \mathcal{R}$ of maximum cardinality. In the following we refer to this problem variant as Star-GCA-simple, because many parameters of the general version are simplified by fixing them to be small constants. A set $Q \subseteq \mathcal{R}$ is a feasible solution for Star-GCA-simple if at most one path per edge is active at any time.

We remark that the cases with one or two different starting times can be solved optimally in polynomial time (still assuming that all calls have the same duration). To see this, note that a maximum cardinality subset among a given set of calls that overlap in time can be obtained using a maximum matching computation [10, 19] in the graph with an edge $\{u, v\}$ for every request with endpoints u and v. This settles the case of one starting time. If there are two different starting times, either all calls overlap in time or the problem decomposes into two instances on disjoint time intervals.

If we allow three or more different starting times, the restricted variant of general call admission in star networks becomes difficult to solve, which is expressed in the next theorem.

Theorem 4. *The problem* Star-GCA-simple *is $\mathcal{APX}$-hard.*

Corollary 2. *GCA in stars is $\mathcal{APX}$-hard.*

In particular, Theorem 4 implies that there is no polynomial-time approximation scheme for the restricted and its more general variants unless $\mathcal{P} = \mathcal{NP}$. Thus, the constant-factor approximations we have presented are best possible (except possibly for the constants) in this sense.

We will prove the theorem by an approximation preserving reduction, which is defined as follows [2].

Definition 1 (*AP*-reduction). *Let P_1 and P_2 be two optimization problems in NPO. For a solution y to an instance x of P_i, let $\mathrm{ratio}_{P_i}(x, y)$ denote the ratio between the value of an optimal solution to x and the value of y (or the reciprocal of this ratio, whichever is larger than 1). The problem P_1 is called AP-reducible to P_2 if two functions f and g and a positive constant $\alpha > 1$ with the following properties exist:*

(i) For any instance x of P_1 and for any rational $r > 1$, $f(x, r)$ is an instance of P_2.
(ii) For any instance x of P_1 and for any rational $r > 1$, if there is a solution to x, then there is also a solution to $f(x, r)$.
(iii) For any instance x of P_1, for any rational $r > 1$, and for any solution y to $f(x, r)$, $g(x, y, r)$ is a solution to x.
(iv) f and g are computable in polynomial time for any fixed rational r.
(v) For any instance x of P_1, for any rational $r > 1$, and for any solution y to $f(x, r)$, $\mathrm{ratio}_{P_2}(f(x, r), y) \le r$ implies $\mathrm{ratio}_{P_1}(x, g(x, y, r)) \le 1 + \alpha(r - 1)$.

The properties (i)-(iv) ensure that there are polynomial time transformations f and g that map instances of P_1 to instances of P_2 and solutions for instances of P_2 back to solutions for the original instance of P_1, respectively. The heart of the AP-reduction is given by property (v), which intuitively says that an r-approximation algorithm for P_2 implies the existence of a $(1 + \alpha(r - 1))$-approximation algorithm for P_1.

In the sequel we present an AP-reduction from the maximum 3-dimensional matching problem, which is defined as follows: Given a set $D \subseteq X \times Y \times Z$, where X, Y and Z are disjoint sets, the goal is to find a matching $M \subseteq D$ for D of maximum cardinality, i.e., a largest set $M \subseteq D$ such that no two elements in M agree in any coordinate. The maximum 3-dimensional matching problem is known to be $\mathcal{APX}$-complete even if each of the elements in X, Y and Z occurs in at most three triples in D [15]. We refer to this problem as the bounded maximum 3-dimensional matching problem.

In this bounded version of the problem, each triple can intersect at most six other triples, which implies that the maximum matching contains at least $|D|/7$ triples. Moreover, the following lemma is easy to prove.

Lemma 5. *There is a greedy procedure that computes a 1/3-approximation for the bounded maximum 3-dimensional matching problem.*

Let $D \subseteq X \times Y \times Z$ be an instance of the maximum 3-dimensional matching problem. The function f of the AP-reduction is given by the following construction of an instance of Star-GCA-simple. It does not depend on the parameter r.

Let vertex 0 be the center vertex of the star. For every element $x_i \in X$, we add the vertex x_i to the star and connect it to vertex 0 by an edge $\{x_i, 0\}$. We do the same for every $y_i \in Y$ and for every $z_i \in Z$. For each triple $d_j = (x_j, y_j, z_j) \in D$,

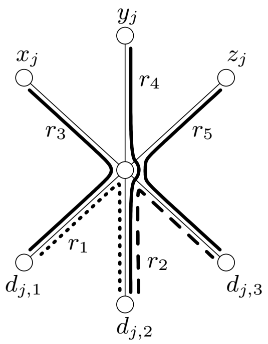

Fig. 1. The building block for a triple $d_j = (x_j, y_j, z_j)$

we add three more vertices $d_{j,1}$, $d_{j,2}$, and $d_{j,3}$ to the star and connect them to the center by the three edges $\{d_{j,1}, 0\}$, $\{d_{j,2}, 0\}$, and $\{d_{j,3}, 0\}$. In addition, we add the following 5 requests to $\mathcal{R}$ (see Figure 1):

- $r_1 = (d_{j,1}, d_{j,2})$ with interval $[0, 2)$
- $r_2 = (d_{j,2}, d_{j,3})$ with interval $[2, 4)$
- $r_3 = (d_{j,1}, x_j)$ with interval $[1, 3)$
- $r_4 = (d_{j,2}, y_j)$ with interval $[1, 3)$
- $r_5 = (d_{j,3}, z_j)$ with interval $[1, 3)$

Note that the dotted request r_1 and the dashed request r_2 have disjoint time intervals, whereas the solid requests r_3, r_4, and r_5 do not share any edge. The idea behind this is the following. For every triple $d_i \in D$, a solution either accepts the two request r_1 and r_2 without affecting any edge connecting a vertex from the sets X, Y, Z to the center, but blocking the requests r_3, r_4 and r_5, or it accepts the three requests r_3, r_4 and r_5 at the price of blocking all three edges connecting the center to the elements of the triple d_i. More than three requests per triple are not feasible.

Lemma 6. *Let $D \subseteq X \times Y \times Z$ be an instance of the maximum 3-dimensional matching problem, and let $(G, \mathcal{R})$ be the corresponding instance of* Star-GCA-simple *defined above. There is a feasible solution for $(G, \mathcal{R})$ that accepts $2|D|+k$ requests if and only if D has a matching of size k.*

Proof. Suppose there is a feasible solution Q for the instance $(G, \mathcal{R})$ of size $2|D| + k$. Since no more than three requests in Q belong to the same triple, there are at least k triples for which three requests are in Q. The only possibility for one triple $d_i = (x_i, y_i, z_i)$ to have three of its requests accepted is the choice that accepts the three requests containing the vertices x_i, y_i and z_i. But then these vertices are blocked for the requests of all other triples. The feasibility of Q implies that all k triples are disjoint. Hence, they form a matching of size k.

Conversely, if there is a matching $M \subseteq D$ of size k, we can construct a feasible solution Q for the instance $(G, \mathcal{R})$ as follows. For every triple $d_i \in M$, put the three requests r_3, r_4 and r_5 into Q. Since the triples in M are disjoint, Q is feasible so far. For each of the remaining triples in $D \setminus M$, we can safely add the two requests r_1 and r_2 to Q without creating any conflict. Thus, Q is feasible by construction, and consists of $2|D| + k$ requests. □

The function g of the AP-reduction takes as arguments the instance D of the bounded maximum 3-dimensional matching problem, a solution Q of the instance $(G, \mathcal{R})$ and the parameter r, which will not be used. It first computes a solution M_1 for the instance D using the greedy procedure of Lemma 5. Secondly, it composes a matching M_2 out of the accepted requests in the solution Q. Whenever all three requests r_3, r_4 and r_5 corresponding to some triple $d_i \in D$ are in Q, it adds the triple d_i to the matching M_2. The value of $g(D, Q, r)$ is given by the larger of the two matchings M_1 and M_2. Thus, $|g(D, Q, r)| = \max\{|M_1|, |M_2|\}$. In addition, we have $|M_1| \geq |M^*|/3$ by Lemma 5, where M^* is a maximum matching for D, and $|M_2| \geq |Q| - 2|D|$ by Lemma 6.

So far the properties (i) – (iv) of the AP-reduction have been shown to be satisfied, and we will now show that property (v) holds with $\alpha = 43$. Therefore, let M^* be a maximum matching for the instance D. Then by Lemma 6 an optimal solution Q for $(G, \mathcal{R})$ consists of $|M^*| + 2|D|$ requests. Assume that we have an r-approximation for $(G, \mathcal{R})$, that is a solution Q that contains at least $(|M^*| + 2|D|)/r$ requests.

If $r \geq 45/43$, the inequality $|g(D, Q, r)| \geq |M_1| \geq |M^*|/3$ shows that g computes a 1/3-approximation. Since $3 = 1 + 43(\frac{45}{43} - 1) \leq 1 + 43(r-1)$, property (v) with $\alpha = 43$ holds in this case.

Otherwise $r < 45/43$. From $|Q| \geq (|M^*| + 2|D|)/r$, we get

$$\begin{aligned}|Q| &\geq \frac{2r|D| + |M^*| - 2(r-1)|D|}{r} = 2|D| + \frac{|M^*| - 2(r-1)|D|}{r} \\ &\geq 2|D| + \frac{|M^*|(1 - 14(r-1))}{r},\end{aligned}$$

where we used $|D| \leq 7|M^*|$ (which holds in the bounded version) in the last inequality. As $|g(D, Q, r)| \geq |M_2| \geq |Q| - 2|D| \geq (1 - 14(r-1))|M^*|/r$, we get that $\text{ratio}_{P_1}(D, g(D, Q, r))$ is at most

$$\frac{|M^*|}{(1 - 14(r-1))|M^*|/r} = 1 + \frac{15}{15 - 14r}(r-1) \leq 1 + 43(r-1),$$

where the last inequality holds for $1 < r < 45/43$. Again, property (v) is fulfilled with $\alpha = 43$, which completes the proof of the theorem.

References

1. N. Alon and J. H. Spencer. *The Probabilistic Method.* Wiley, New York, Second edition, 1992.
2. G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation. Combinatorial Optimization Problems and their Approximability Properties.* Springer-Verlag, Berlin, 1999.
3. A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. S. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM (JACM)*, 48(5):1069–1090, 2001.

4. R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discrete Mathematics*, 25:27–46, 1985.
5. P. Berman, B. DasGupta, S. Muthukrishnan, and S. Ramaswami. Improved approximation algorithms for rectangle tiling and packing. In *Proceedings of the 12th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA)*, pages 427–436, 2001.
6. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.
7. A. Chakrabarti, C. Chekuri, A. Gupta, and A. Kumar. Approximation algorithms for the unsplittable flow problem. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, LNCS 2462, pages 51–66. Springer-Verlag, 2002.
8. C. Chekuri, M. Mydlarz, and F. B. Shepherd. Multicommodity demand flow in a tree. In *Proceedings of the 30th International Colloquium on Automata, Languages, and Programming (ICALP)*, LNCS 2719, pages 410–425. Springer-Verlag, 2003.
9. N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. van der Merwe. Resource management with hoses: point-to-cloud services for virtual private networks. *IEEE/ACM Transactions on Networking (TON)*, 10(5):679–692, 2002.
10. J. Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.
11. T. Erlebach. Call admission control for advance reservation requests with alternatives. In *Proceedings of the 3rd Workshop on Approximation and Randomization Algorithms in Communication Networks (ARACNE)*, Proceedings in Informatics, pages 51–64. Carleton Scientific, 2002.
12. M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York-San Francisco, 1979.
13. R. A. Guérin and A. Orda. Networks with advance reservations: The routing perspective. In *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 118–127, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.
14. V. Guruswami, S. Khanna, R. Rajaraman, B. Shepherd, and M. Yannakakis. Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC)*, pages 19–28, 1999.
15. V. Kann. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters*, 37:27–35, 1991.
16. J. Kleinberg. *Approximation algorithms for disjoint paths problems.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1996.
17. S. Leonardi. On-line network routing. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms: The State of the Art*, LNCS 1442. Springer-Verlag, Berlin, 1998.
18. L. Lewin-Eytan, J. S. Naor, and A. Orda. Routing and admission control in networks with advance reservations. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, LNCS 2462, pages 215–228. Springer-Verlag, 2002.
19. S. Micali and V. V. Vazirani. An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matchings in general graphs. In *Proceedings of the 21st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 17–27, 1980.