Admission control to minimize rejections

Avrim Blum[1], Adam Kalai[2], and Jon Kleinberg[3]

[1] Carnegie Mellon University, Pittsburgh PA 15213,
avrim@cs.cmu.edu
[2] Massachusetts Institute of Technology, Cambridge MA 02139,
akalai@mit.edu
[3] Cornell University, Ithaca NY 14853,
kleinber@cs.cornell.edu

Abstract. Admission control (call control) is a well-studied online problem. We are given a fixed graph with edge capacities, and must process a sequence of calls that arrive over time, accepting some and rejecting others in order to stay within capacity limitations of the network. In the standard theoretical formulation, this problem is analyzed as a benefit problem: the goal is to devise an online algorithm that accepts at least a reasonable fraction of the maximum number of calls that could possibly have been accepted in hindsight. This formulation, however, has the property that even algorithms with optimal competitive ratios (typically $O(\log n)$ where n is the number of nodes) may end up rejecting the vast majority of calls even when it would have been possible in hindsight to reject only very few.

In this paper, we instead consider the goal of approximately *minimizing* the number of calls *rejected*. This is much more appropriate for settings in which rejections are intended to be rare events. In order to avoid trivial lower-bounds, we assume *preemption* is allowed and that calls are given to the algorithm as fixed paths. We show that in a number of cases, we can in fact achieve a competitive ratio of 2 for rejections (so if the optimal in hindsight rejects 0 then we reject 0; if the optimal rejects r then we reject at most $2r$). For other cases we get worse but nontrivial bounds. For the most general case of fixed paths in arbitrary graphs with arbitrary edge capacities, we achieve matching $\Theta(\sqrt{m})$ upper and lower bounds, where m is the number of edges. We also show a connection between these problems and online versions of the vertex-cover and set-cover problems (our factor-2 results give 2-approximations to slight generalizations of the vertex cover problem, much as [AAA99] show hardness results for the benefit version based on the hardness of approximability of independent set).

1 Introduction

In the well-studied *admission control* (or *call control*) problem, our job is to manage a network G (a graph with edge capacities) in the presence of online requests for communication (calls). Requests for communication may be accepted or rejected, and the goal of an online algorithm is to accept as many as possible while staying within the edge capacities of the network.

This problem has typically been studied as a *benefit problem*. That is, one compares the number of calls that could have been accepted in hindsight to the number actually accepted by the online algorithm, and tries to minimize this ratio. A number of papers have produced good bounds for this metric, such as the work of Awerbuch, Azar and Plotkin [AAP93] for the high-capacity setting, and Awerbuch et al. [AGLR94] for trees and other specific networks. A serious problem with viewing call-control as a benefit problem, however, is that even with, say, an $O(\log n)$ competitive ratio that would normally be considered quite good, it is possible that the algorithm may route only a $1/(\log n)$ fraction of the calls even if a solution routing nearly all of them is possible.[1] For many of the natural applications of admission control, even a modest constant fraction of rejections would be deemed unacceptable performance. Thus, for these types of applications, the benefit formulation appears fundamentally flawed.

In this paper we depart from the benefit metric and instead set our sights on the goal of *minimizing* the number of calls *rejected*. That is, if OPT (the optimal strategy in hindsight) rejects 0 then we should reject 0. If OPT rejects a small number, then we should reject only a small multiple of that. What we show is that for several natural cases, we can in fact achieve a competitive ratio of 2 for rejections. For other versions we can achieve worse but still non-trivial bounds. Of course, approximately minimizing rejections suffers from the reverse problem that the algorithm may accept no calls even if in hindsight it was possible to accept, say, half of them. However, in many applications, even *optimal* performance in such a case would be unacceptable: if one's network required one to reject a significant fraction of calls, then the correct response would be to upgrade the network. It is these types of settings that motivate our work.

We assume in our results that the online algorithm is allowed *preemption*: at any time we may preempt (reject) requests that had previously been accepted, and simply count it as if the request had been rejected from the start. This is one of the standard models and is necessary to achieve any nontrivial bound on rejections. A second assumption we make is that each request is for a unit demand on a fixed path. That is, the requests can be thought of as a sequence of paths $p_1, p_2, \ldots$, and the decision made by the online algorithm is just whether to accept or reject each path, and does not involve routing. Again, if routing is part of the algorithm's job, then even in very simple settings, no nontrivial bound is possible for our performance metric.[2] Our results are then as follows:

---

[1] In fact, prior to Leonardi et al. [LMSPR98] the situation was even worse. Depending on the types of requests made, many of the randomized algorithms would, with probability $1 - 1/(\log n)$, accept no calls at all. That is, the variance of possible benefits was high compared to the expectation.

[2] Consider a 4-cycle ABCD with capacity c on each edge. Imagine that we are given c calls connecting the diagonally opposite nodes A and C, and then we are given either c calls connecting A and B, or else c calls connecting A and D, with equal probability.

Admission control on a line: When the underlying graph is a line, we can achieve a competitive ratio of 2 for any set of edge capacities. That is, the algorithm will reject at most twice as many as the minimum possible in hindsight.

Admission control on a general graph: For general graphs, we can achieve a competitive ratio of 2 if all edge capacities are 1 (the disjoint paths case). This extends to a ratio of $c+1$ if all edge capacities are $\leq c$. For arbitrary capacities, we give a different algorithm that achieves a competitive ratio of $O(\sqrt{m})$, where m is the number of edges, which we match with an $\Omega(\sqrt{m})$ lower bound.

An interesting aspect of the rejection measure is that the easiest cases are when capacities are *low*. This is the opposite of the situation for the benefit measure, where low capacities are difficult and higher capacities make the problems easier. A preliminary version of this work appears in [BKK01].

1.1 Related work

As discussed above, existing work on admission control has primarily focused on the problem of maximizing the number of accepted calls, rather than minimizing the number of rejected calls. (See the surveys by Plotkin [Plo95] and Leonardi [Leo98].) The one exception we are aware of is the work of Kamath, Palmon, and Plotkin [KPP96], who provide performance guarantees in terms of a competitive ratio on rejections. Their setting is quite different from ours, however. Most significantly, they assume the input to be probabilistically generated, not worst-case. They consider calls that are generated according to a Poisson process, with exponentially distributed holding times, and also assume the maximum bandwidth of a call to be very small relative to the available edge capacity (i.e. large capacities). Moreover, their model does not allow preemption.

Routing on fixed paths, as we consider here, was studied by Alon, Arad, and Azar [AAA99] and Awerbuch, Azar, and Plotkin [AAP93] under the traditional measure of maximizing benefit. Of course, in linear networks, and more generally in tree networks, one is necessarily routing on fixed paths; for trees, work of Awerbuch et al. [ABFR94] provides $O(\log n)$-competitive algorithms for maximizing benefit. (See also the earlier work of [GG92,GGK$^+$93] for linear networks, and the improved probabilistic guarantees obtained by Leonardi et al. [LMSPR98].)

A number of previous papers have considered the performance gains obtainable by allowing preemption, in the context of maximizing benefit. Adler and Azar [AA99] show that allowing preemption leads to an $O(1)$-competitive algorithm for benefit maximization in linear networks.

Every on-line algorithm rejects $c/2$ calls in expectation, while it was possible to reject none off-line. Similarly, with n separate 4-cycles, the on-line algorithm rejects $nc/2$ calls in expectation, while OPT rejects none.

1.2 Notation and definitions

We are given a graph G, which may be directed or undirected, with m edges and n nodes. Each edge e has an integer capacity $c_e > 0$. We are also given a sequence of requests, $p_1, p_2, \ldots$, each of which is a simple path in the graph. Each path may either be accepted or rejected. The requirement is that for every edge e, the number of unrejected requests that have edge e should be no larger than c_e. We will call a set of rejection decisions *valid* if it satisfies this requirement.

In the off-line problem, we must simply find a small valid set of rejections. In the on-line problem, we are given requests one at a time, and we must choose to accept or reject the requests on-line so that the set of accepted requests never exceeds the capacity of any edge. We also allow our online algorithm to preempt an earlier request, i.e. we may reject a request after already accepting it. However, we may not accept a request after rejecting it.

Let OPT be a minimum valid set of rejections. We say that an algorithm is k-competitive (k may be a function of m, n, and c) if the number of requests *rejected* by this algorithm is at most $k|\mathrm{OPT}|$.

One final note: Our algorithms will sometimes decide to reject some requests even when not strictly necessary. Because we have preemption, these can always be implemented in a lazy manner. That is, such requests can be marked but not actually rejected until a new request arrives that causes a conflict with it.

2 Preliminaries: set-cover and vertex-cover

Imagine a situation where accepting all requests would put every edge one unit over capacity. The off-line problem is then exactly set cover, where each path is represented by the set of edges it contains, and we must choose a set of paths (to reject) that covers each edge at least once. In this case, the on-line admission control problem can be thought of as a version of online set-cover. While this is not a direct reduction, we hope the following discussion will serve as motivation and intuition for our algorithms later.

A well-known result for the set-cover problem is that if every point is in at most k sets, then there is a simple k-approximation algorithm: pick an arbitrary uncovered point, take all sets that cover it, and repeat. The case $k = 2$ corresponds to vertex cover.

A slight generalization of the $k = 2$ case is a setting in which a point may potentially be covered by many sets $s_1, s_2, \ldots$, but where we are guaranteed that some two of those sets s_i, s_j cover their union. Then one can achieve a 2-approximation as follows: pick an arbitrary uncovered point p, find two sets that cover the union of all sets covering p, take those two sets and repeat. This is a 2-approximation because each time two sets are chosen, they can be charged to whatever set s_p in the optimal solution is used to cover p. Because the two sets chosen by the algorithm contain s_p, we are guaranteed that each selection of two sets is charged to a unique set in the optimal cover.

Some of the results below can be viewed as an online version of this algorithm and guarantee.

3 Admission control on a line

We begin with the special case of a line graph. Each edge e has some arbitrary capacity c_e. A request corresponds to an interval on this line and the capacities limit the number of intervals covering any given edge that may be accepted. We show a 2-competitive algorithm, based on the set-cover idea above. The idea is that whenever a new request cannot be accepted due to capacity constraints, we arbitrarily select one of the edges that would go over capacity, and throw out the two requests p_l and p_r covering that edge that extend farthest to the left and farthest to the right, respectively. (One of these may or may not be the current request.) We then accept the current request if we did not throw it out. To be more precise:

1. If a request can be accepted, accept it.
2. If a request cannot be accepted, then choose an arbitrary edge e that would be put over capacity.
   (a) Among the unrejected requests that contain e (including the current request), let p_l be one that extends furthest to the left.
   (b) Among the unrejected requests that contain e (including the current request), let p_r be one that extends furthest to the right.
3. Reject p_l and p_r, and accept the current request if it is not one of $\{p_l, p_r\}$.

Theorem 1. *The above algorithm is 2-competitive.*

Proof. Consider some optimal valid rejection set OPT. Each time the algorithm rejects a pair of requests $\{p_l, p_r\}$, we will modify OPT (call the new set OPT') by adding at most 1 request to it, in order to maintain an invariant that OPT' is a superset of the requests rejected by the online algorithm. We do this as follows. Each time the online algorithm reaches case 2, we know that OPT must have rejected at least one request p_{opt} of those being considered by the online algorithm (i.e., at least one of those covering edge e that have not yet been rejected by the online algorithm). Therefore, when the online algorithm rejects p_l and p_r, we know that (viewing paths as sets of edges) $p_l \cup p_r \supseteq p_{opt}$. Therefore, if we put p_l and p_r into OPT', and then remove p_{opt} if neither p_l nor p_r had been in OPT' already, this only adds 1 to the size of OPT', maintains its status as a valid rejection set, and maintains our invariant. So, if t is the number of requests rejected by the online algorithm, then $t \leq |\mathrm{OPT}'| \leq |\mathrm{OPT}| + t/2$, and therefore $t \leq 2|\mathrm{OPT}|$. □

Another way of viewing this argument is that each time the algorithm rejects two requests p_l and p_r, we give OPT a "two-fer", allowing it to reject those two requests for the price of 1. Since OPT must reject some request contained in their union, it might as well take the offer. Inductively, at the end of the game, OPT has rejected the exact same set as the online algorithm, but at half the cost.

The above algorithm and analysis also applies if the underlying graph is a cycle (with fixed paths).

4 General graphs

4.1 The low capacity case

Theorem 2. *On a general graph G, if every edge e has capacity $c_e \leq c$, then there is a simple $(c+1)$-competitive algorithm.*

Proof. The algorithm is just an online version of the k-approximation to set cover:

1. If a request can be accepted, accept it.
2. If a request cannot be accepted, then choose some edge e that would be over capacity. Reject the current request along with the c_e (unrejected) other requests that contain e.

This algorithm rejects sets of requests of size $\leq c+1$ that all share an edge. These sets are disjoint. Any valid rejection set must include at least one request from each of these sets. Therefore, the algorithm achieves a competitive ratio of $c+1$. □

Thus, if all edges have capacity 1 (the disjoint paths case) we have a 2-competitive algorithm.

4.2 General capacities

The above algorithm gets worse as the capacities in the graph become large. Can we achieve a bound independent of the capacities for general graphs? The connection to set-cover suggests that perhaps we could achieve an $O(\log m)$ bound. However, it turns out that the online nature of the problem makes that impossible. What we show instead are a set of matching $\Theta(\sqrt{m})$ upper and lower bounds. We begin with the lower bound.

Theorem 3. *There is a $\Omega(\sqrt{m})$ lower bound on the competitive ratio of any online algorithm for general graphs with arbitrary capacities. This holds for randomized algorithms as well.*

Proof. For clarity, we will use a multigraph for our lower bound, and then at the end, convert the multigraph into a standard (non-multi) graph. The multigraph consists of $k+1$ vertices $\{0, 1, \ldots, k\}$ arranged in a line, with k edges connecting each vertex to the next. So the total number of edges is k^2. Each edge has capacity k^{k-1}.

We begin by seeing k^k paths of length k, one for each possible route between vertex 0 and vertex k. By design, these will fill all edges exactly to capacity. We then see k single-edge paths: the first path is a random edge between vertex 0 and vertex 1; the second is a random edge between vertex 1 and vertex 2, and so on.

The offline algorithm needs only to reject one path, namely the path among the first k^k that happens to match the sequence of k single-edge paths seen at the end. However, any online algorithm must reject at least $k/2$ in expectation. That is because if $j < k$ paths have been rejected so far, then the next single-edge path seen has at least a $(k-j)/k$ chance of causing its edge to go over capacity. Hence, as long as $j < k/2$, the next edge causes a rejection with probability at least $1/2$. Therefore, the competitive ratio of any online algorithm is at least $k/2$ which is $\Omega(\sqrt{m})$.

We can convert the multigraph in the above argument into a standard (non-multi) graph by just putting a node in the middle of each edge, which only doubles the total number of edges. $\square$

Remark: Our lower bound requires an exponential number of requests and exponential capacity sizes. Perhaps one might be able to give guarantees logarithmic in m if we also allow logarithmic dependence on the maximum capacity.

We now give a matching $O(\sqrt{m})$ upper bound. Specifically, we present a $4\sqrt{m}$-competitive algorithm for an arbitrary multigraph with m edges. The algorithm is as follows, starting with zero "chips" on every edge and $R = 0$.

1. If a request covers at least $\sqrt{m}$ edges that have chips, then
   (a) Reject the request
   (b) Remove one chip from each of the request's edges (that have chips)
2. If a request cannot be accepted (some edge would be over capacity), then
   (a) Reject the request
   (b) $R = R + 1$
   (c) If R is a multiple of $\sqrt{m}$, then
      i. Add a chip to each edge of the graph.
      ii. Reapply Step 1 to every accepted request so far.
3. Else, accept the request.

Theorem 4. *The above algorithm is $O(\sqrt{m})$-competitive.*

Analysis. Observe that the number of rejections from Step 1 (line 1a, including rejections caused by 2cii) is no more than the number of rejections from Step 2 (line 2a). This is because, after R Step-2 rejections, we have placed no more than $R/\sqrt{m}$ chips on each edge, and every Step-1 rejection removes at least $\sqrt{m}$ chips. Thus, to show that the algorithm is $4\sqrt{m}$-competitive, we will show that $R \leq 2|OPT|\sqrt{m}$. From here on, when we refer to a rejection, we mean a Step-2 rejection.

Just before we perform each rejection, we "blame" it on an individual edge in one of OPT's rejections, as follows. Let e be the first of the edges that would have gone over capacity had we accepted the request. We blame the rejection on the first OPT rejection that we have not yet rejected, has e, and has not yet been blamed for e. Not only must there be some such OPT rejection to blame, but it must come no later in the request sequence than the current request. To see this, say we have had e as a blame edge t times before, and we have rejected r of OPT's rejections that have e. Then, including the current request, we must have seen $c_e + r + t + 1$ requests that contain e. OPT must also reject at least

$r + t + 1$ requests with e, and we have rejected r of these and blamed t of them for e, leaving at least 1 previous OPT rejection to blame.

Also notice that after $|OPT|$ chips have been removed from an edge, we will not blame any more rejections on that edge. This is because the total number of requests that have an edge does not exceed $c_e + |OPT|$. Finally, it suffices to show that no OPT rejection is blamed for more than $\sqrt{m}$ rejections after $R = |OPT|\sqrt{m}$. At this point, we have placed $|OPT|$ chips on each edge, where some chips may have been removed. Fix an OPT rejection. Look at the first rejection after $R = |OPT|\sqrt{m}$ blamed on it. Since we did not reject the OPT rejection in Step 1, it has less than $\sqrt{m}$ edges with chips. All of its other edges must have had at least $|OPT|$ chips removed, so they will never blamed again. Thus, we will blame at most $\sqrt{m}$ edges on each OPT rejection after $R = |OPT|\sqrt{m}$, which implies $R \leq 2|OPT|\sqrt{m}$ and the total number of rejections is at most $4|OPT|\sqrt{m}$.

5 The offline case

It is interesting to consider the *offline* version of our problem because of its connection to set-cover. For the off-line problem, we know the *excess* of each edge, i.e. the number of calls that include that edge minus its capacity. Let n_e be the excess of edge e. Our goal is to reject the fewest requests such that each edge e is contained in at least n_e rejections. This can be thought of as generalization of a set cover problem, where each point e has an associated number n_e, and instead of the usual goal of covering each point at least once, a legal cover must cover point e at least n_e times.

Let us define $N = \sum_e n_e$; that is, N is the total sum of the excesses. Then, the usual analysis of greedy set-cover gives us an $O(\log N)$ approximation to this generalized problem. In particular, if we imagine placing n_e chips on point e, then the greedy set-cover algorithm becomes: take the set that covers the most points of those with chips on them, remove one chip from each point covered, and repeat. If the optimal solution uses k sets, then at each step, the greedy algorithm must remove at least a $1/k$ fraction of the chips remaining, giving us the $O(\log N)$ ratio.

A natural question is whether this upper bound can be improved to $O(\log m)$ where m is the number of points (edges). This would be strictly better than what is achievable for the online problem. It is not clear if the greedy algorithm can be used to achieve this, but we *can* get $O(\log m)$ via randomized rounding as follows.

Formulate the problem as a linear program, where $0 \leq f_i \leq 1$ is the fraction of set s_i to take (the fraction of the ith request to reject). Our objective is to minimize $\sum f_i$ subject to the chip (capacity) constraints $\sum_{i:e \in s_i} f_i \geq n_e$, for every point e. The minimum value of the objective will be no larger than k, the value of the optimal integer solution. We solve the LP efficiently and then round the solution. To round, we choose each set independently, with probability of choosing set s_i equal to $\min(1, 5 f_i \log m)$.

Next, Chernoff bounds imply that a given point e is covered at least n_e times with probability $\geq 1-1/m^2$. To see this, first note that we do not have to worry about those sets that have $5f_i \log m \geq 1$, because we will select them for certain. So ignore these sets. Let z_e be the sum of the f_i for the remaining sets s_i that have e, and we can assume $z_e \geq 1$ else we are already done. We expect to select $5z_e \log m$ sets covering e, and the only way in which we could fail is if the number we actually select is less than z_e. By the multiplicative Chernoff bounds,[3] this happens with probability less than

$$e^{-\left(1-\frac{1}{5\log m}\right)^2(5z_e \log m)/2} \leq \frac{1}{m^2}$$

for sufficiently large m. Thus with probability at least $1-m/m^2$ we will have *all* points covered the desired number of times. Furthermore, the expected number of sets chosen is $O(k \log m)$, as desired.

6 Conclusions

We have shown that in a number of natural cases, we can achieve good, or at least nontrivial bounds for minimizing the number of rejections in admission control. One open question left by these results is that our $\Omega(\sqrt{m})$ lower bound requires an exponential number of requests and exponential size capacities. Perhaps one might be able to achieve bounds that are logarithmic in m if we also allow logarithmic dependence on the maximum capacity c. Another interesting open problem is whether these results can be extended to the case where different requests may have different demands.

7 Acknowledgements

We would like to thank the referees for a number of helpful comments. This research was conducted while the second author was at Carnegie Mellon University, and was supported in part by NSF grants CCR-9732705 and CCR-0085982, NSF Faculty Early Career Development Award CCR-9701399, a David and Lucile Packard Foundation Fellowship, an ONR Young Investigator Award, and an IBM Graduate Fellowship.

References

[AA99] R. Adler and Y. Azar. Beating the logarithmic lower bound: randomized preemptive disjoint paths and call control algorithms. In *Proceedings of 10th SODA*, pages 1–10, 1999.

[3] If X is a sum of independent $\{0,1\}$-valued random variables, with $\mu = E[X]$, the multiplicative Chernoff bound states that $Pr[X < (1-\epsilon)\mu] \leq e^{-\epsilon^2\mu/2}$.

[AAA99] N. Alon, U. Arad, and Y. Azar. Independent sets in hypergraphs with applications to routing via fixed paths. In *Proceedings of 2nd APPROX*, pages 16–27, 1999.

[AAP93] Baruch Awerbuch, Yossi Azar, and Serge Plotkin. Throughput-competitive online routing. In *Proc. 34th Symp. Foundations of Computer Science*, pages 32–40, 1993.

[ABFR94] Baruch Awerbuch, Yair Bartal, Amos Fiat, and Adi Rosén. Competitive non-preemptive call control. In *Proc. 5th Symp. on Discrete Algorithms*, pages 312–320, 1994.

[AGLR94] Baruch Awerbuch, Rainer Gawlick, Tom Leighton, and Yuval Rabani. On-line admission control and circuit routing for high performance computing and communication. In *Proc. 35th Symp. Foundations of Computer Science*, pages 412–423, 1994.

[BKK01] Avrim Blum, Adam Kalai, and Jon Kleinberg. Admission control to minimize rejections. *Proceedings of WADS 2001*. LNCS 2125, pages 155–164, 2001.

[GG92] Juan A. Garay and I. S. Gopal. Call preemption in communications networks. In *Proc. INFOCOM '92*, pages 1043–1050, 1992.

[GGK+93] Juan A. Garay, I. S. Gopal, Shay Kutten, Yishay 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.

[KPP96] A. Kamath, O. Palmon, and Serge Plotkin. Routing and admission control in general topology networks with poisson arrivals. In *Proc. 7th Symp. on Discrete Algorithms*, pages 269–278, 1996.

[Leo98] S. Leonardi. On-line network routing. In A. Fiat and G. Woeginger, editors, *On-line Algorithms*. Springer-Verlag, 1998.

[LMSPR98] Stefano Leonardi, Alberto Marchetti-Spaccamela, A. Presciutti, and Adi Rosèn. On-line randomized call-control revisited. In *Proc. 9th Symp. on Discrete Algorithms*, pages 323–332, 1998.

[Plo95] Serge Plotkin. Competitive routing in atm networks. *IEEE J. Selected Areas in Communications*, pages 1128–1136, 1995.