Generating Adversaries for Request-Answer Games

Todd Gormley*, Nicholas Reingold†, Eric Torng‡, and Jeffery Westbrook§

1 Introduction and Results

The k-server conjecture postulates the existence of an *algorithm* which is k-competitive for all values of k on all metric spaces. We give a procedure which is guaranteed to find a *complementary structure* - an adversary strategy and a metric space such that no algorithm is k-competitive against the adversary strategy and metric space - assuming such a structure exists. That is, we prove the complement of the k-server conjecture is recursively enumerable. In fact, we give a general procedure that can perform the following search for a fairly general subset of request-answer games [3]. For a given c, find a finite adversary strategy against which no algorithm can be better than c-competitive assuming such an adversary strategy exists. This essentially implies that for these request-answer games, "Is $c^{\Pi} \geq c$?" is recursively enumerable, where c^{Π} is the optimal competitive ratio for request-answer game Π. We have used a modified version of our procedure to find a lower bound of 1.85358 for the online load balancing problem on $m \geq 80$ identical machines, improving slightly upon the lower bound of 1.852 by Albers [1]. We can extend our procedure to produce oblivious adversaries for randomized online algorithms if we explicitly bound the number of random bits used by the randomized algorithm.

Request-answer games are a class of online problems defined by Ben-David *et al.* [3] in which each request must be serviced immediately by the online algorithm. Furthermore, while there may be an infinite number of legal requests, there are only a finite number of well-defined answers or responses for serving a request. Our procedure only applies to a subset of request-answer games which satisfy an additional "linearizability" constraint, but we defer a description of this linearizability constraint until after the following definition of an adversary. Fortunately, many problems satisfy this linearizability constraint including many variations of the k-server problem, the CNN problem, online load balancing, online virtual circuit routing, and online page migration.

We represent adversary strategies for request-answer games as game trees in which there are two kinds of nodes, adversary request nodes and online move nodes. No adversary request node is a leaf. The children of an adversary request node are online move nodes, one for each of the finite number of ways a reasonable online algorithm can service the current request node. Each online move node is either a leaf or has a single request node as a child. The root r of a tree T is an adversary request node.

For any node v, let $P(v)$ denote the path from the root to node v and $s(v)$ denote the corresponding request sequence. Each adversary request node v is annotated with the optimal move sequence $opt(v)$ for servicing the request sequence $s(v)$. We also use $opt(v)$ to denote the associated cost. For any online move node v, let $ms(v)$ denote the corresponding online move sequence that services $s(v)$ as well as the associated cost, and let $c(v) = \frac{ms(v)}{opt(p(v))}$ where $p(v)$ is the parent node of node v. This quantity $c(v)$ denotes the ratio of the online cost for servicing $s(v)$ to the optimal cost of servicing $s(v)$. For each root-leaf path P, define $c(P) = \max_{v \in P} c(P)$. Finally, for tree T, define $c(T) = \min_{P \in T} c(P)$. A tree T with value $c(T)$ corresponds to an adversary strategy which guarantees no online algorithm is better than $c(T)$-competitive for the corresponding finite request-answer problem Π.

Our goal is to find a tree T such that $c(T)$ is maximal. A simpler goal is, given any value c, find a tree T such that $c(T) \geq c$. We are able to satisfy this goal with the following added constraint. For any node v, let $h(v)$ denote the number of requests in $s(v)$. For any tree T, let $h(T) = \max_{v \in T} h(v)$.

THEOREM 1.1. *Given any value c and any integer n, there is a procedure which either finds a tree T such that $c(T) \geq c$ and $h(T) \leq n$ if such a tree exists or returns the fact that no such tree exists for "linearizable" request-answer games.*

By increasing n to ∞, this procedure turns into a semidecision procedure which, when given any value c, finds a finite tree T such that $c(T) \geq c$ if such a tree exists. We can extend this result to generate good oblivious adversaries for randomized algorithms by adding an additional constraint B on the number of

*Department of Computer Science and Engineering, Michigan State University. Supported in part by an MSU Professorial Assistantship.

†AT&T Research, Florham Park A015, 180 Park Avenue, P. O. Box 971, Florham Park, NJ 07932-0000, reingold@research.att.com.

‡Department of Computer Science and Engineering, 3115 Engineering Building, Michigan State University, East Lansing, MI 48824, torng@cse.msu.edu. Supported in part by NSF Career Grant CCR-9701679.

§AT&T Research, Florham Park A001, 180 Park Avenue, P. O. Box 971, Florham Park, NJ 07932-0000, jeffw@research.att.com.

random bits used by the randomized online algorithm.

We now describe linearizability and the basic idea behind our algorithm for generating deterministic adversaries. We want to find an optimal adversary tree T with $h(T) \leq n$, so we create a template tree $T(V)$ with $h(T) \leq n$ where the request nodes are variables, and we now want to find an assignment A of the variables V such that $c(T(A)) \geq c$.

We use linear programming to perform this search. Specifically, we encode $c(T(V)) \geq c$ as a single program with constraints that are linear in the variables V except for the use of max's and min's. Note, this assumes that the costs $opt(v)$ and $ms(v)$ as well as all side constraints such as the metric space is a line must be representable by linear constraints on the variables V. Problems which satisfy this admittedly vague condition are linearizable.

We eliminate the max's and min's to create linear programs in one of the two following ways. If we have a constraint of the form $f(V) \geq \max(f_1(V), f_2(V), \ldots, f_i(V)$, we replace it with the constraints $f(V) \geq f_1(V), f(V) \geq f_2(V), \ldots, f(V) \geq f_i(V)$. If we have a constraint of the form $f(V) \leq \max(f_1(V), f_2(V), \ldots, f_i(V)$, we create i distinct programs where the constraint in program j for $1 \leq j \leq i$ is replaced by $f(V) \leq f_j(V)$. Repeated application of these two operations results in a set of linear programs such that if any one of these programs is feasible, the feasible solution corresponds to an assignment A of the variables V such that $c(T(A)) \geq c$. On the other hand, if none of the linear programs is feasible, then no tree T with $h(T) \leq n$ exists such that $c(T) \geq c$.

2 Related Work

Our work is somewhat similar to the work of Manasse *et al.* [5] where the authors present a decision procedure for computing the optimal competitive ratio for server problems on fixed, finite metric spaces. In particular, their procedure solves the following problem. Given any c and a finite point metric space M in which the distance function between all points is specified, answer whether or not there exists a c-competitive algorithm for M.

Their result is more general than ours in that it does not require an a priori bound on the number of requests generated by the adversary. On the other hand, ours is more general in several ways. First, our procedure applies to a wider class of problems. Furthermore, when restricted to the k-server problem, our procedure does not require the metric space to be finite or prespecified. In fact, our procedure *finds* a metric space which satisfies the desired lower bound constraint. Finally, because $h(T)$ ranges over the natural numbers which are countably infinite, our decision procedure implies that the complement of the k-server conjecture is recursively enumerable. On the other hand, because the number of metric spaces is uncountably infinite, it is not obvious that their decision procedure generalizes to any recursive enumerability result.

Our work was inspired by previous methods used to derive lower bounds for online load balancing on identical machines [1, 2, 4]. These authors chose specific sequences of job sizes, represented as variables, and constructed constraints on the job sizes which enforced specific packing requirements such as the online algorithm must place each new job on the currently least loaded machine. Our work removes the ingenuity required to first generate a good packing by searching over the space of all packings.

References

[1] S. Albers. Better bounds for online scheduling. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 130–139, 1997.

[2] Y. Bartal, H. Karloff, and Y. Rabani. A better lower bound for on-line scheduling. *Information Processing Letters*, 50:113–116, 1994.

[3] S. Ben-David, A. Borodin, R.M. Karp, G. Tardos, and A. Wigderson. On the power of randomization in online algorithms. In *Proc. 22nd ACM Symposium on Theory of Computing*, pages 379–388, 1990.

[4] B. Chen, A. van Vliet, and G. Woeginger. New lower and upper bounds for on-line scheduling. *Operations Research Letters*, 16:221–230, 1994.

[5] M.S. Manasse, McGeoch L.A., and D.D. Sleator. Competitive algorithms for on-line problems. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 322–333, 1988.