Noname manuscript No.
(will be inserted by the editor)

Serve or Skip: The Power of Rejection in Online Bottleneck Matching

Barbara M. Anthony · Christine Chung

August 13, 2015

Abstract We consider the online matching problem, where n server-vertices lie in a metric space and n request-vertices that arrive over time each must immediately be permanently assigned to a server-vertex. We focus on the egalitarian bottleneck objective, where the goal is to minimize the maximum distance between any request and its server. It has been demonstrated that while there are effective algorithms for the utilitarian objective (minimizing total cost) in the resource augmentation setting where the offline adversary has half the resources, these are not effective for the egalitarian objective. Thus, we propose a new Serve-or-Skip bicriteria analysis model, where the online algorithm may reject or skip up to a specified number of requests, and propose two greedy algorithms: GRINN(t) and GRIN$^*(t)$. We show that the Serve-or-Skip model of resource augmentation analysis can essentially simulate the doubled-server-capacity model, and then examine the performance of GRINN(t) and GRIN$^*(t)$.

1 Introduction

We consider the well-studied problem of minimum-cost bipartite matching in a metric space. We are given n established points in the metric space, $s_1, s_2, \ldots, s_n$, referred to as *servers*, which form one side of the bipartition. Over time, the other n points, $r_1, r_2, \ldots, r_n$, appear in the metric space, and we refer to them as *requests*. We must permanently match or assign each request r_i, for $i = 1 \ldots n$, to a server upon its arrival, without any knowledge of future request locations $r_{i+1}, \ldots, r_n$.

The standard objective for this problem has been to minimize the *total cost* of the final matching, frequently referred to as the *min-weight* matching problem. Formally,

Barbara M. Anthony
Mathematics and Computer Science Department, Southwestern University, Georgetown, TX
E-mail: anthonyb@southwestern.edu

Christine Chung
Department of Computer Science, Connecticut College, New London, CT
E-mail: cchung@conncoll.edu

if $d(r,s)$ gives the *cost* or *distance* in the metric space from point r to point s, and we use $\mu(r_i)$ to denote the server matched to request r_i in the matching μ, the goal of the *min-weight matching problem* is to find a matching

$$\mu^* = \operatorname*{argmin}_{\mu} \sum_{i=1}^{n} d(r_i, \mu(r_i)).$$

In this work, however, we consider instead the objective of minimizing the maximum distance from any request to its assigned server. That is, we seek a matching

$$\mu^* = \operatorname*{argmin}_{\mu} \max_{i=1\ldots n} d(r_i, \mu(r_i)).$$

This objective is also known as the bottleneck objective, and we refer to this problem as the *online minimum-bottleneck matching problem*.

From an economic perspective, the total-cost objective is a *utilitarian* objective that may be natural in many situations (for example, when trying to stay within an overall, centralized budget), while the bottleneck objective is *egalitarian* and seeks to ensure fairness. Issues such as fairness are a growing concern in algorithm design, as part of the wide-spread emergence of game theoretic settings in computer science, due to the proliferation of internet-based, distributed and ad-hoc computing. However, in spite of the egalitarian objective growing in importance and relevance, to the best of our knowledge, the bottleneck objective has remained largely unexplored.

There is growing evidence that the utilitarian total-cost objective for our online matching problem is easier than the egalitarian bottleneck objective. Indeed, the total-cost objective has been quite well-understood with respect to deterministic algorithms for over two decades. Kalyanasundaram and Pruhs (1993), and independently, Khuller et al. (1994), first showed that the basic greedy algorithm that matches each arriving request to its nearest available server has an exponential competitive ratio, and that the best competitive ratio any deterministic algorithm can achieve is $2n-1$. They also give an algorithm called PERMUTATION that achieves this competitive ratio. (Interestingly, when the underlying metric is restricted to the line, the worst-known lowerbound is 9.001, given in Fuchs et al. (2005), yet no constant-competitive algorithm has been found.)

The fact that no sub-linear-competitive deterministic algorithm can be found for the utilitarian objective gave rise to a natural question: if the offline optimal solution is too lofty a benchmark, what natural adversary *can* an online algorithm hope to compete with? An answer was given in Kalyanasundaram and Pruhs (2000b), in the context of the Online Transportation Problem, a generalization of the min-cost matching problem that additionally specifies the number of servers at each server location. They showed that the greedy algorithm BALANCE is constant-*halfOPT-competitive*. In other words, the online algorithm is constant-competitive when given twice the server capacity at each server location.

Unfortunately, such a "weakened adversary" or "resource augmentation" approach does not help when it comes to the bottleneck objective. It was demonstrated in Anthony and Chung (2014) that PERMUTATION and BALANCE both fail to be better than $O(n)$-halfOPT-competitive for the bottleneck objective. Additionally, algorithms

(such as BALANCE) that are designed based on the assumption that each server location has the capacity to serve at least two requests, do not apply to the classic matching setting of one server per request. It was also shown early on in an unpublished manuscript by Idury and Schaffer (1992) that any deterministic algorithm must be at least $\approx 1.5n$-competitive (against the standard, unweakened, optimal, offline adversary).

These negative results motivate us to consider an alternate form of resource augmentation that gives a benchmark as simple and appealing as "half the server capacity," but perhaps powerful enough to compete with the offline optimal solution for the more difficult bottleneck objective. In the present work, we ask: what happens when the online algorithm is allowed to reject requests? In other words, what if the online algorithm is given a number of "free passes"?

Specifically, we propose the Serve-or-Skip (SoS) bicriteria analysis model: assume the online algorithm has an allowance of p *passes* or *skips*, which means the online algorithm may reject up to p of the requests without incurring any cost. This means that, in the case of the bottleneck (egalitarian) objective, a rejected request will not be a candidate for being the bottleneck match. We refer to an algorithm as *c-SoS(p)-competitive* if, when rejecting no more than p of the requests, it is c-competitive with the offline optimal solution.

One might expect the resource augmentation model of doubling the capacity of each server to be just as powerful as having permission to ignore half the requests. However, we show that SoS can readily simulate the result of any algorithm under the doubled-capacity model of resource augmentation, suggesting that it may in fact be a more "powerful" resource augmentation model (i.e., the "weaker" adversary of the two). We also propose two threshold-based algorithms and analyze their competitiveness against the offline optimal matching that is required to serve all requests.

We believe that SoS has, in some circumstances, more practical appeal as a benchmark than the previously proposed benchmark of doubling the resources, since in many real-world application areas, the question is one of "quality of service" rather than "service at all costs." We speculate that service providers in various real-world situations may not only be interested in quantifying the gains from rejecting a fraction of their incoming service requests, but also be more willing to entertain the notion of reducing their quality of service before they consider the idea of dramatically augmenting their resources.

1.1 Other related work

Incidentally, there has been an active line of recent work in online algorithms studying the power of allowing the online algorithm to have some recourse for its past actions. Megow et al. (2012) study the online minimum spanning tree problem, where points arrive online and the online algorithm must connect the point to the existing tree as they arrive. They show that by allowing for a small number of re-arrangements on previously-placed edges, a nearly optimal tree can be maintained. They also apply their technique to online TSP. Gu et al. (2013) then showed that with only a single retroactive edge-swap per step, a constant-competitive Steiner tree can be maintained

online. Finally, Gupta et al. (2014) show that in online b-matching, where a bipartite matching must be maintained online such that the degree of each node is at most b, allowing a constant number of total re-assignments of past matches suffices for a constant-competitiveness. They also apply their technique to online scheduling with unrelated machines in the restricted assignment model, as well as to an online single-sink flow problem.

Resource augmentation analysis, or using a weakened adversary, is a technique that has been well-established and effectively and successfully used in many domains, including matching, scheduling, and algorithmic game theory. In the literature on matching, the resource augmentation has been in the form of doubling (Kalyanasundaram and Pruhs (2000b); Anthony and Chung (2014)) or incrementing (Chung et al. (2008)) server site capacities. In the online machine scheduling literature, it has taken the form of augmenting the processing speed of machines (Kalyanasundaram and Pruhs (2000a); Phillips et al. (2002)). In auction theory, in an extensive line of work starting with Goldberg et al. (2001), including Hartline and Roughgarden (2009), a weakened adversary is used as a trade-off for the lack of information the auction mechanism has regarding the bidders true valuations for the items being sold. And in a seminal work of algorithmic game theory by Roughgarden and Tardos (2002), the optimal solution in a routing game is required to route twice as much traffic in exchange for centralized coordination among the players.

A very active and popular problem of study in the domain of online bipartite matching is known as the AdWords problem (see, e.g., Kalyanasundaram and Pruhs (2000c); Mehta et al. (2007); Goel and Mehta (2008); Devanur and Hayes (2009); Fernandes and Schouery (2014)). There, however, no assumption of an underlying metric space is made (i.e., the edge weights need not satisfy the triangle inequality), each server may be matched to multiple requests as long as its given "budget" is not exceeded, and the objective is to maximize the weight of the final matching.

1.2 Our results

We begin by showing that our proposed SoS model of resource augmentation analysis can simulate the doubled-server-capacity model, and hence any algorithm analyzed under that model immediately implies a corresponding algorithm with an SoS-competitive ratio at least as good as the original algorithm's halfOPT-competitive ratio.

We then propose a natural algorithm that assigns requests greedily, but skips any request that has no available servers within t times the distance to its nearest neighbor. We call the algorithm GrINN(t) for Greedy Inside Nearest Neighbor Threshold, and show that this algorithm has an SoS(p)-competitive ratio of $\Theta(2^{n-p})$ and that this ratio is tight. I.e., when allowed p free passes, the bottleneck cost of the GrINN(t) solution is guaranteed to be no more than $\Theta(2^{n-p})$ times that of the optimal offline assignment that matches all the requests. Hence each free pass improves the ratio by a factor of 2. Of course, for instances in which the algorithm does not run out of free passes, the SoS(p)-competitive ratio is at most t.

Table 1 Summary of the results in this work. Note that, trivially, $c = t$ for instances in which the algorithm does not run out of free passes. Also, the bottom-most lowerbound here on GRIN$^*(t)$ is stronger (higher) than the one above it, but we include the weaker bound because we find its structure to be intuitive and instructive, and perhaps useful as a general technique for lower-bounding in the SoS model of weakened adversary analysis.

Algorithm	Applicable values of t	Applicable values of p	SoS(p)-competitive ratio c
GRINN(t)	$1 < t < 2^{n-p} - 1$	$0 \le p \le n-1$	$c = \Theta(2^{n-p})$
GRIN$^*(t)$	$t = 1$	$0 \le p \le n-1$	$c = 2^{n-p} - 1$
	$t = 2$	$0 \le p \le (n-1)/2$	$c = \Omega(p2^{n-2p})$
	$1 < t \le 2^{(n-1)/p-1}$	$0 \le p \le \frac{n+1}{\lceil \log t \rceil + 1}$	$c \ge 2^{n-(\lceil \log t \rceil + 1)p-1}$
	$t > 2$	$0 \le p \le (n-1)/2$	$c \ge \frac{t^p - 2^p}{(t-2)t^p} + 2^{n-2p} - 1$

We then propose another greedy algorithm, which we refer to as GRIN$^*(t)$, which assigns requests greedily as long as there are available servers within t times the optimal offline bottleneck distance of the requests that have arrived so far. If there are no available servers within this threshold, the request is skipped. We then show that the SoS(p)-competitive ratio of GRIN$^*(1)$ is exponential in $n-p$, and that this is tight. So each free pass improves the competitive ratio by a factor of 2, as with GRINN(t). We also show that the SoS(p)-competitive ratio of GRIN$^*(2)$ is exponential in $n-2p$, suggesting, interestingly, that each free pass improves the competitive ratio by a factor of 4. We conjecture that this ratio is tight for GRIN$^*(2)$. Again, for instances in which the algorithm does not run out of free passes, the SoS(p)-competitive ratio is of course at most t. Finally, we also provide lowerbounds for the SoS-competitiveness of GRIN$^*(t)$ for any $t > 1$. Our results are summarized in Table 1.

Along the way, we also provide an upper bound on the competitive-ratio of the basic GREEDY algorithm for online bottleneck matching (with no resource augmentation). We show that GREEDY is $(2^n - 1)$-competitive, which essentially matches the existing lower bound given in Anthony and Chung (2014).

2 Preliminaries

We propose the Serve-or-Skip (SoS) model of resource augmentation analysis, where the online algorithm is endowed with an allotment of p *free passes* to be used as follows. Upon the arrival of each request, the online algorithm may choose to assign it irrevocably as usual, or, instead, use one of the p passes and reject the request, leaving it unassigned. After the online algorithm has used all p passes, it is required to assign all remaining incoming requests.

This is in contrast with the previously established resource augmentation model where the offline algorithm is assumed to have greater server capacity at each server location. Specifically, Kalyanasundaram and Pruhs (2000b) defined a *c-halfOPT-competitive* algorithm to be one that is c-competitive with an optimal offline solution that uses half the capacity at each server location (so this characterization only applies to settings where there are at least two servers per server location). We call an

algorithm *c-SoS(p)-competitive* if, when rejecting no more than p of the requests, it is c-competitive with the offline optimal solution.

2.1 SoS$\left(\frac{n}{2}\right)$-competitive vs halfOPT-competitive

We show that any online algorithm that is c-halfOPT-competitive can immediately be transformed into an algorithm that is c-SoS$\left(\frac{n}{2}\right)$-competitive, and hence that SoS$\left(\frac{n}{2}\right)$ is at least as easy a benchmark as halfOPT for online algorithms to compete with.

Theorem 1 *If an online algorithm X is c-halfOPT-competitive for minimum bottleneck (resp, weight) matching, there is an online algorithm Y that is c-SoS$\left(\frac{n}{2}\right)$-competitive for minimum bottleneck (resp, weight) matching.*

Proof Given algorithm X that is c-halfOPT-competitive, transform it to a c-SoS$\left(\frac{n}{2}\right)$-competitive algorithm Y as follows.

1. Pretend there is a "secondary server" at the location of each server in the input.
2. Run X, and
   whenever X wants to assign a request to an imaginary "secondary server," reject that request,
   otherwise, choose the same server that X does.

We know that Y will not use more than $\frac{n}{2}$ free passes, because if it does, then X must have assigned more than n requests, but there are only n requests total.

Since all of the assignments made by Y are also made by X, the bottleneck (resp, total) cost of the assignment made by Y will be no more than the bottleneck (resp, total) cost of X. □

This result immediately yields the following fact regarding the min-weight objective and the SoS-competitiveness of the algorithm that would result from transforming the previously proposed BALANCE algorithm (Kalyanasundaram and Pruhs (2000b)). Recall that BALANCE was shown to be O(1)-halfOPT-competitive for the online transportation problem, assuming there are at least two servers per server site.

Corollary 1 *Let* BALANCESOS *be the algorithm that results after applying the transformation described in Theorem 1 to the algorithm* BALANCE *(Kalyanasundaram and Pruhs (2000b)).* BALANCESOS *is O(1)-SoS(n/2)-competitive for the min-weight matching problem.*

2.2 Greedy Threshold Algorithms

Greedy algorithms are a natural first choice to consider in the SoS(p) model. An algorithm that merely assigns each request to the closest available server, without skipping any requests, is the naive greedy algorithm in the classical model. Thus, we must specify how the algorithm determines which requests, if any, to skip. One logical choice is to use a *threshold*, where if the desired server is distance x away, the algorithm greedily picks the closest available server within distance $t \cdot x$ for some

threshold $t \geq 1$; if no such server exists, it rejects the request. (Clearly it does not make sense to consider a threshold $t < 1$.)

It remains to define the 'desired' server. We first consider an algorithm we call GRINN(t), where the desired server is the *nearest neighbor* to the current request. Thus, when request r_i arrives, the algorithm with a threshold parameter $t \geq 1$ finds the minimum distance d_i to any server (available or not) from r_i. Formally, $d_i = \min_j d(r_i, s_j)$. It then greedily picks the closest *available* server within distance at most $t \cdot d_i$, assigning r_i to that server if it exists, and rejecting request r_i if there is no such server. If p rejections have already been made, the algorithm greedily assigns r_i to the nearest available server. While the simplicity of this algorithm is appealing, we show that its performance is exponential in the number of assigned requests for any threshold t.

We thus consider an improved version of the threshold algorithm which makes its decision about serving or skipping a request based on what the optimal solution would do with the set of requests that have arrived so far. We denote the algorithm GRIN$^*(t)$ with an associated parameter $t \geq 1$. Let OPT$_i$ refer to the optimal offline matching on the set of requests $\{r_1, r_2, \ldots r_i\}$, i.e., if M_i refers to the set of all partial matchings between the first i requests and any i of the n servers $\{s_1, \ldots, s_n\}$,

$$\text{OPT}_i = \operatorname*{argmin}_{\mu_i \in M_i} \max_{j=1\ldots i} d(r_j, \mu_i(r_j)).$$

(We abuse notation to let OPT$_i$ represent either the set of assignments made, or the bottleneck cost of said set of assignments.) With the arrival of request r_i, if the nearest available server is within distance $t \cdot \text{OPT}_i$ of r_i, then assign it to r_i, otherwise reject/skip r_i. Note that OPT$_i$ can be computed efficiently (e.g., as described in Gabow and Tarjan (1988); Garfinkel (1971)) from OPT$_{i-1}$ and thus the assignments made by GRIN$^*(t)$ are polynomial-time computable.

2.3 Observations about GRIN$^*(t)$ and GRINN(t)

We now make some basic observations about GRIN$^*(t)$ and GRINN(t) that will be useful in analyzing their performance in the SoS(p) model. Let OPT = OPT$_n$.

Lemma 1 GRIN$^*(t)$ *(resp.* GRINN(t)*) always assigns the first request, with a cost of at most* OPT$_1 \leq$ OPT.

Proof The first request can always be assigned to the server it was assigned to by OPT$_1$, which is its nearest neighbor. GRIN$^*(t)$ and GRINN(t) will make exactly this assignment. Since the OPT$_i$ values are non-decreasing, OPT$_1 \leq$ OPT. □

We now provide an instance that requires $n-1$ skips for GRIN$^*(1)$ to stay optimal, and then show that GRIN$^*(1)$ is in fact 1-SoS$(n-1)$-competitive (i.e., optimal when it can reject $n-1$ requests). This instance is a *subdivided star* where all but one edge in the original star is subdivided into two edges, as shown in Figure 1.

Lemma 2 *If* GRIN$^*(1)$ *(resp.* GRINN(1)*) is 1-SoS(p)-competitive, then $p \geq n-1$.*

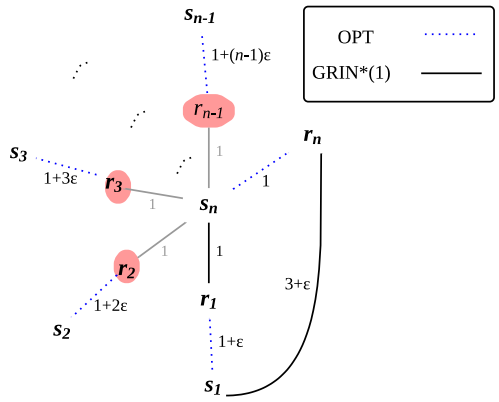

Fig. 1 The subdivided star instance of Lemma 2. Requests skipped by GRIN$^*(1)$ are highlighted.

Proof Let there be a centrally-located server, called s_n, at the root of a subdivided star, with n requests each a distance of 1 from the center server. As shown in Figure 1, servers $1 \dots n-1$ are located at the leaves of the star, with server s_i emanating a distance of $1+i\varepsilon$ from r_i, for $i = 1 \dots n-1$. Thus, the subdivided edges of the star each have a request at the division and a server at the leaf s_i, while the one undivided edge has a request at the leaf r_n. All distances $d(r,s)$ not given explicitly are equal to the cost of the path from r to s in the subdivided star.

Note that $\text{OPT}_i = 1+(i-1)\varepsilon$ for $i = 1 \dots n$, as it always matches r_i to s_n and r_j to s_j for $j = 1 \dots i-1$. On the other hand, GRIN$^*(1)$ (or GRINN(1)) skips every request after the first request, which it assigns to the root. If GRIN$^*(1)$ (or GRINN(1)) is only allowed to skip $p < n-1$ requests, it will skip requests $r_2 \dots r_{p+1}$, forcing it to assign all remaining requests, up to and including r_n. The bottleneck edge is the one associated with r_n, with a distance of $3+\varepsilon$ while $\text{OPT} = 1+(n-1)\varepsilon$. □

Corollary 2 GRIN$^*(1)$ *(resp.* GRINN(1)*) is 1-SoS$(n-1)$-competitive for online min-bottleneck matching.*

Proof Since Lemma 1 ensures that the first request is always assigned with a cost of at most OPT, GRIN$^*(1)$ (resp. GRINN(1)) can skip all of the remaining requests. With a threshold value of 1, GRIN$^*(1)$ will only assign later requests if the assignment cost is at most $\text{OPT}_i \leq \text{OPT}$, guaranteeing that the bottleneck cost is at most OPT. Likewise, GRINN(1) will only assign to the nearest neighbor; that distance is again at most OPT. □

3 Upper Bounds for GRIN$^*(t)$ and GRINN(t)

In this section we prove upper bounds on the performance of GRIN$^*(t)$ and GRINN(t). We note that, trivially, the SoS(p)-competitive ratio of both algorithms is t for instances in which the algorithm does not run out of free passes.

For the remainder of this work, we define $k = n - p$ to be the minimum number of assignments that must be made by the online algorithm. We first consider $t = 1$ and then generalize to an arbitrary threshold $t \geq 1$. We show that the $\mathrm{SoS}(p)$-competitive ratio of both $\textsc{Grin}^*(t)$ and $\textsc{GriNN}(t)$ is $O(2^{n-p})$. In the next section we show that this ratio is tight for $\textsc{Grin}^*(1)$ as well as $\textsc{GriNN}(t)$, for $t > 1$, so each free pass reduces the ratio by a factor of 2.

3.1 Upper Bound for $t = 1$

Theorem 2 *For $1 \leq p \leq n-1$, $\textsc{Grin}^*(1)$ is $(2^{n-p}-1)$-SoS(p)-competitive for online minimum bottleneck matching.*

Proof By definition of $\textsc{Grin}^*(1)$, and since the OPT_i are nondecreasing in i, if more than k requests are assigned, they must each have assignment cost at most $\mathrm{OPT} = \mathrm{OPT}_n$. Thus, we may assume that exactly k requests are assigned, and only consider these requests for the remainder of the proof.

We relabel these assigned requests to be $r_1, \ldots, r_k$, where the subscripts represent the relative order of arrival (and thus assignment) within these k requests. We will show inductively that the assignment cost of r_i is at most $(2^i - 1) \cdot \mathrm{OPT}$ for $i = 1 \ldots k$.

Base case: $i = 1$. By Lemma 1, the first request r_1 is always assigned with a cost of at most OPT, which satisfies the claim.

Inductive case: Assume that the assignment cost of r_j is at most $(2^j - 1) \cdot \mathrm{OPT}$ for all $1 \leq j \leq i$. Consider the assignment cost of r_{i+1}. Let s_{i+1} be the server that OPT_{i+1} assigns to r_{i+1}. If s_{i+1} is available, the assignment cost is at most $\mathrm{OPT}_{i+1} \leq \mathrm{OPT}$.

Thus, we may assume that s_{i+1} is not available, and it is used by some r_j with $j \leq i$. We thus consider the graph consisting of edges in OPT_{i+1} and the i edges assigned thus far by $\textsc{Grin}^*(1)$. Since r_{i+1} is not yet matched by $\textsc{Grin}^*(1)$, there must be a path in this graph from r_{i+1} to some s_a that is used by OPT_{i+1} but not currently matched by $\textsc{Grin}^*(1)$. Observe that said path must begin with an edge in OPT_{i+1} and alternate between edges in OPT_{i+1} and edges in $\textsc{Grin}^*(1)$, terminating with an edge in OPT_{i+1}.

We can thus use triangle inequality to compute the distance from this available s_a to r_{i+1}, giving an upper bound on the assignment cost of r_{i+1}. Since all distances are nonnegative, additional edges either cause the total cost to increase or stay the same. Thus, in the worst case, the path includes all of the i assignments already made, as well as the $i+1$ edges in OPT_i.

By strong induction, the cost of the i assignments already made is at most

$$\sum_{h=1}^{i} (2^h - 1) \cdot \mathrm{OPT} = \left(2^{i+1} - i - 2\right) \cdot \mathrm{OPT}.$$

Noting that the OPT_i are non-decreasing, the total cost of the $i+1$ edges from OPT_{i+1} is upper-bounded by $(i+1) \cdot \mathrm{OPT}$. Adding this to the expression above bounds the total distance (and thus assignment cost) of r_{i+1} to some available server by $(2^{i+1} - 1) \cdot \mathrm{OPT}$, completing the inductive proof. □

Corollary 3 *For* $1 \leq p \leq n-1$, GRINN(1) *is* $(2^{n-p}-1)$*-SoS*(p)*-competitive for online minimum bottleneck matching.*

Proof The proof is identical to that of Theorem 2, with the observation that the distance from a request to its nearest neighbor is naturally at most OPT. □

Incidentally, a similar proof also yields the essentially-tight upper-bound of 2^n-1 on the competitive-ratio of the basic GREEDY algorithm for the bottleneck matching problem (without the aid of resource augmentation). The matching lower bound was given in Anthony and Chung (2014). Due to its similarity to the proof of Theorem 2, we defer the proof of this fact to Appendix B.

3.2 Upper Bound for $t > 1$

In generalizing the upper bound for GRIN$^*(t)$ to $t > 1$, any of the greedy assignments made by GRIN$^*(t)$ may in fact cost up to a factor of t more than the optimal solution on the set of requests thus far. The following proof parallels that of Theorem 2, but has some notable distinctions because requests that would have been skipped when $t = 1$ may now be assigned.

Theorem 3 *For* $1 \leq p \leq n-1$, GRIN$^*(t)$ *is* $\max\{2^{n-p}-1, t\}$*-SoS*(p)*-competitive for online minimum bottleneck matching.*

Proof Since GRIN$^*(t)$ only assigns a request if it has no skips left or the cost is below the threshold, if more than k requests are assigned, these additional requests (above k) must all have assignment costs of at most $t \cdot \text{OPT}$. Thus we may consider only the first k requests that are assigned for the remainder of the proof.

Relabeling the requests and using strong induction as in Theorem 2 then shows that the assignment cost of r_i is at most $\max\{2^i-1, t\} \cdot \text{OPT}$. Specifically, the base case is unchanged, and the inductive case now relies on the observation that either the alternating path has the same bound as in Theorem 2 or that request r_{i+1} was assigned to some available server within distance $t \cdot \text{OPT}_{i+1}$. Thus, taking the larger of these two gives the upper bound for general t. □

The result in Theorem 3 again extends naturally to the GRINN(t) algorithm.

Corollary 4 *For* $1 \leq p \leq n-1$, GRINN(t) *is* $\max\{2^{n-p}-1, t\}$*-SoS(p)-competitive for online minimum bottleneck matching.*

In the next section we provide a matching lower bound for GRINN(t) (and for GRIN$^*(1)$) that, combined with the upper bound in this section, shows that each skip improves the SoS(p)-competitiveness of GRINN(t) (resp. GRIN$^*(1)$) by a factor of 2.

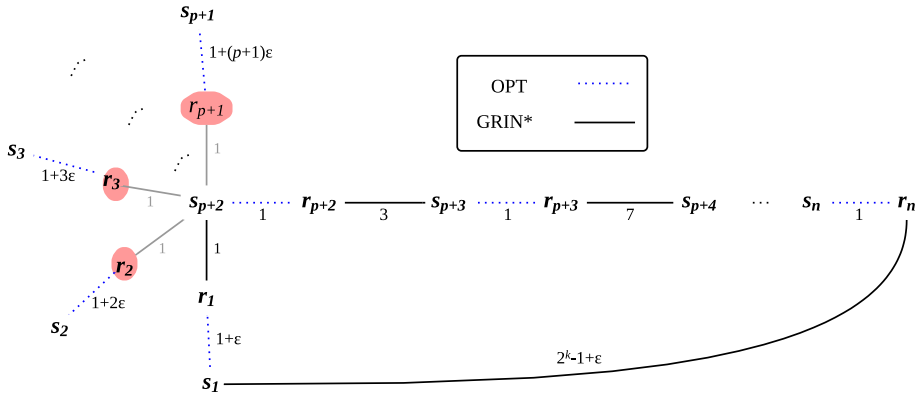

Fig. 2 The broom-with-subdivided-star instance. Requests skipped by GRIN$^*(1)$ are highlighted.

4 Lower Bounds

4.1 Lower Bound for GRINN(t) and GRIN$^*(1)$

We provide a lowerbound on GRIN$^*(1)$, and use the same instance to provide a lower bound for the cost of GRINN(t) in the SoS(p) model for arbitrary $p = n-k$. We use an example whose structure is comprised of a broom with a subdivided star. While a standard broom graph consists of a path and a star, this specialized broom graph consists of a path, one of whose endpoints is the center of a subdivided star.

Theorem 4 *For $0 \le p \le n-1$, if GRIN$^*(1)$ is c-SoS(p)-competitive for online minimum bottleneck matching, then $c \ge (2^{n-p} - 1 - \varepsilon)$.*

Proof The requests are numbered in order of arrival (see Figure 2). The broom with a subdivided star consists of a star portion, centered at s_{p+2} with $p+1$ leaves, and a longer handle. In particular, r_1 through r_{p+1} are each a distance 1 away from the center, s_{p+2}. For $i = 1 \ldots p+1$, there is a server s_i that is at a distance of $1+i\varepsilon$ away from r_i, making these servers all a distance greater than 2 from the center. The remaining requests and servers lie along a line emanating from the center, terminating with request r_n. Specifically, until r_n is reached, r_{p+2} is a distance 1 from the center. Server s_{p+3} is 3 units further along the line, with request r_{p+3} 1 unit further. Server s_{p+4} is 7 units further, followed by request r_{p+4} 1 unit further. Server s_{p+5} is 15 units further, followed by request r_{p+5} 1 unit further. In general, s_{p+j} is $2^{j-1}-1$ units past request r_{p+j-1}, and 1 unit before r_{p+j}.

The optimal solution assigns each request r_i in the star portion (i.e., $1 \le i \le p+1$) to its corresponding leaf server s_i for a cost of $1+i\varepsilon$, and each request on the handle portion also to its corresponding server (r_i to s_i for $p+2 \le i \le n$), for a cost of 1. Hence, OPT is $1+(p+1)\varepsilon$.

We now consider the behavior of GRIN$^*(1)$. Again, the first request is always assigned, so r_1 is greedily assigned to the center of the star portion, s_{p+2}. With the arrival of r_2, OPT$_2$ would have assigned r_1 to s_1 and r_2 to s_{p+2}, so OPT$_2$ is $1+\varepsilon$, causing GRIN$^*(1)$ to skip r_2 (since the closest available server, s_2, is $1+2\varepsilon$ away).

Similarly, for the remaining requests on the star portion, that is, for $i = 2 \ldots p+1$, when r_i arrives, OPT_i is $1+(i-1)\varepsilon$ and the nearest available server, s_i is at a distance $1+i\varepsilon$ so the algorithm skips request r_i. Hence, it uses up all of the allowed p skips. Thus, all of the remaining requests must be assigned, and will be done so greedily. In particular, r_{p+2} greedily chooses the server s_{p+3} that is 3 to the right instead of paying $3+\varepsilon$ (or more) to use a server in the star portion. Similarly, each of the remaining requests will choose to go further along the handle when possible rather than paying slightly more to go back to the star portion. Thus, the final request r_n, the leaf of the handle, must traverse the entire length of the handle, a distance of

$$1 + \sum_{i=p+2}^{n-1} (2^{i-p} - 1 + 1),$$

and then go $\varepsilon + 2$ to the closest leaf of the star (s_1) for a total cost of

$$\varepsilon + 1 + 2 + \sum_{i=2}^{n-p-1} 2^i = \varepsilon + \sum_{i=0}^{n-p-1} 2^i = 2^{n-p} - 1 + \varepsilon,$$

growing exponentially in the number of requests assigned by the algorithm. □

In contrast with GRIN$^*(1)$, GRIN$^*(t)$, $t \geq 2$, for example, does well on the broom with subdivided star. GRINN(t), on the other hand, is not as effective. We can easily modify the distances in the graph to be more challenging for GRINN(t).

Corollary 5 *For $0 \leq p \leq n-1$, if* GRINN(t) *is c-SoS(p)-competitive for online minimum bottleneck matching, then $c \geq \Omega(2^{n-p})$.*

Proof The instance here is similar to that of Theorem 4 (Figure 2), however the distances $d(r_i, s_i)$, for $i = 1 \ldots p+1$ are instead $t+\varepsilon$. The edge from r_{p+2} to s_{p+3} now has cost $t+2$, the edge from r_{p+3} to s_{p+4} has cost $2t+5$, the edge from r_{p+4} to s_{p+5} has cost $4t+11$, and in general the edge from r_{p+i} to s_{p+i+1} has cost $2^{i-2}t + 3 \cdot 2^{i-2} - 1$ for all $i = 2 \ldots k$.

Thus, the bottleneck edge, the edge out of r_n, has a cost of $2^{n-p-2}t + 3 \cdot 2^{n-p-2} - 1 + \varepsilon$. OPT assigns each r_i to the corresponding s_i, for a bottleneck cost of $t+\varepsilon$. □

We note that taken together with Corollary 4, we have now given a tight analysis of the SoS-competitiveness of GRINN(t).

4.2 Lower Bounds for GRIN$^*(2)$

We now return to GRIN$^*(2)$, having noted that it does well on some instances on which GRIN$^*(1)$ does poorly. We present two lowerbound instances in this section, the first weaker, but straightforward and instructive, and perhaps useful as a basic lower-bounding technique in the SoS analysis model.

It is well known (and we can see, i.e., from the "handle" portion of our broom instance) that a standard greedy algorithm that is not able to reject requests can do quite poorly, even when all the servers and requests lie on a single line. Intuitively, the

SoS(p) model allows GRIN$^*(t)$ to recover from a poor decision by skipping a request. Yet, what if the instance consists of numerous line segments, that are mutually far apart? In such an input instance, each segment (of 2 requests/servers) forces a skip, and on the last segment there are no skips left, so the final request will have to travel the length of the segment: a distance of 2^{n-2p-1}.

Theorem 5 *For $p \leq \frac{n-1}{2}$, GRIN$^*(2)$ is no better than 2^{n-2p-1}-SoS(p)-competitive. (Hence each pass reduces this lower bound on the competitive ratio by a factor of 4.)*

Proof The instance consists of $p+1$ line segments as follows, with request numbers indicating their arrival order, as usual. The first line segment consists of servers s_1 located at $-1-\varepsilon$ and s_2 at value 1 and requests r_1 located at 0 and r_2 at $1+2\varepsilon$. In general, segment i, for $i = 1,\ldots,p$, consists of server locations s_{2i-1} located at $-1-\varepsilon$ on line segment i, s_{2i} at value 1 on line segment i, and requests r_{2i-1} located at 0 on line segment i and r_{2i} at $1+2\varepsilon$ on line segment i. Segment i, for $i = 1\ldots p+1$, is far enough (at least 2^n) away from all points on previous segments $j < i$. The final segment has the standard server locations of the known worst instance against the basic greedy algorithm that matches each arriving request to its nearest available server: server s_j for $j = 2p+1,\ldots,n$ are at locations $-1-\varepsilon, 1, 3, 7, \ldots, 2^{n-2p-1}$. Requests r_j for $j = 2p+1,\ldots,n$ arrive at the following corresponding locations on the final segment: $0, 1, 3, 7, \ldots, 2^{n-2p-1}$.

OPT will assign each r_i to the corresponding s_i, for a bottleneck cost of $1+\varepsilon$ (with many assignments having cost 0). GRIN$^*(2)$ assigns r_1 to s_2 (the cheapest assignment) and then the cost of assigning r_2 to s_1 is $2+3\varepsilon$, which exceeds the allowable cost of $2 \cdot \text{OPT} = 2+2\varepsilon$, and all other servers are at least 2^n away, so r_2 is skipped. On the next segment, GRIN$^*(2)$ similarly assigns r_3 to s_4, and skips r_5, and so on for the first p segments. After p segments, the algorithm has used all of its skips, so its behavior on the remaining segment is to assign all requests to the servers to their immediate right, assigning the final request to the leftmost server of segment p for a cost of 2^{n-2p-1}. □

Note that when half the requests may be rejected ($p \approx n/2$), GRIN$^*(2)$ is constant-SoS-competitive on this instance. Unfortunately, our next theorem shows even when $p \approx n/2$, GRIN$^*(2)$ has an SoS-competitive ratio no better than linear.

While the previous instance required the algorithm to assign at least two assigned requests per skipped request, this next instance is able to force a skipped request for each assigned request. We conjecture that this is the worst-possible for GRIN$^*(2)$ and that there is a matching upperbound on the SoS-competitive ratio.

Theorem 6 *For $p \leq (n-1)/2$, GRIN$^*(2)$ has a SoS(p)-competitive ratio no better than $\Omega(p2^{n-2p})$.*

Proof Consider the following input instance, pictured in Figure 3. The structure of the instance is that requests arrive from left to right, with requests r_i, for $i = 1,3,\ldots,n$ (n an odd integer) assigned by GRIN$^*(2)$ arriving on a line, and skipped requests r_i, $i = 2,4,\ldots,n-1$, arriving "above" the line after each assigned request. (Note that for now we assume $p = k-1 = (n-1)/2$, and we describe the general case

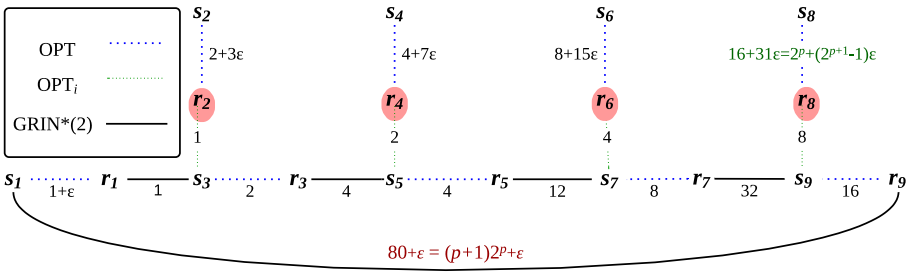

Fig. 3 Requests are numbered in order of arrival and requests skipped by the online algorithm GRIN*(2) are highlighted. The special case of the instance where $p = k-1 = (n-1)/2$ is pictured here. Note that in this case GRIN*$(2) = (n-p)2^p + \varepsilon = (p+1)2^p + \varepsilon$. In the more general version of this input instance, when $p \leq k-1$, GRIN*$(2) = p2^{k-1} + 2^k - 2^p + \varepsilon$

of $p \leq k-1$ below.) Define $d(r_1, s_1) = 1 + \varepsilon$. For each server s_i, for $i = 3, 5, \ldots, n$, numbered according to the index of the request that is matched to it by OPT, we have $d(r_{i-1}, s_i) = 2^{(i-1)/2-1}$ and $d(r_i, s_i) = 2^{(i-1)/2}$. For each $i = 2, 4, \ldots, 2p$ we have $d(r_i, s_i) = 2^{i/2} + (2^{i/2+1} - 1)\varepsilon$. Finally, each request r_i, for $i = 3, 5, \ldots, n-2$, has

$$d(r_i, s_{i+2}) = 2d(r_{i-2}, s_i) + d(r_i, s_i), \tag{1}$$

with $d(r_1, s_3) = 1$. This is effectively the total distance from r_i back to s_1, and it can be verified via substitution that in closed form, for $i = 3, 5, \ldots n-2$,

$$d(r_i, s_{i+2}) = (i+1)2^{(i-3)/2}.$$

It can be easily verified that $d(r_n, s_1) = (n-p)2^{n-p-1}$ by letting $i = n$ and $p = (n-1)/2$.

Each of the requests r_i, for $i = 1, 3, \ldots, n-2$ is assigned by GRIN*(2) to the server to its right s_{i+2}, and by OPT$_i$ for $i = 3, 5, \ldots, n$ to the server to its left s_i. For each request r_i, for $i = 2, 4, \ldots, n-1$, OPT$_i$ initially assigns it to server s_{i+1} on the line, at a cost of $2^{i/2-1}$, and therefore r_i is skipped by GRIN*(2). But upon arrival of request r_{i+1}, OPT$_{i+1}$ reassigns r_i to s_i, which doubles the bottleneck cost of OPT. Finally, when r_n arrives, GRIN*(2) has no choice but to match it to s_1 for a final bottleneck cost of $(n-p)2^{n-p-1} + \varepsilon = k2^{k-1} + \varepsilon = (p+1)2^p + \varepsilon$, since $p = k-1$, and the added ε term comes from the definition of $d(r_1, s_1)$.

For general $p \leq k-1$, we simply omit any excess requests arriving above the line, stopping after p have arrived. In other words, we include one to-be-skipped request "above" each assigned request until we have a total of p skipped requests. The remaining requests all arrive on the line following the same pattern as the previous requests that have arrived on the line, continuing to increment their index value from left to right. Specifically, all distances are defined as above in equation (1) for $i = 3, 5, \ldots, 2p+1$. But now, for $i = 2p+2 \ldots n$, we define $d(r_i, s_i) = 2^p$, and

$$d(r_i, s_{i+1}) = 2d(r_{i-1}, s_i) + 2^p \tag{2}$$

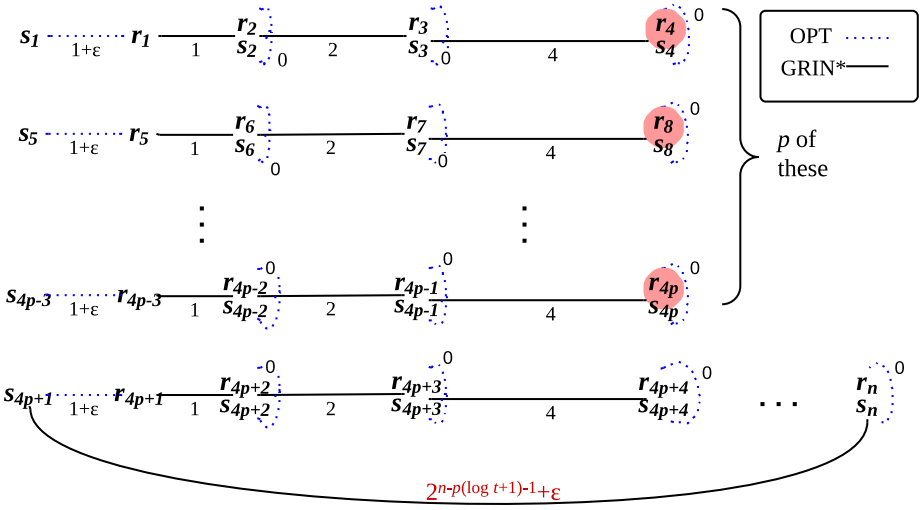

Fig. 4 An instance for $t = 6$ is illustrated here, yielding an SoS(p)-competitive ratio of $2^{n-p(\lceil \log t \rceil+1)-1} = 2^{n-4p-1}-1$. Requests are numbered in order of arrival and requests skipped by the online algorithm GRIN* are highlighted.

for $i = 2p+2 \ldots n-1$. The bottleneck cost is the total distance from r_n back to s_1, which, using the two sets of recurrences (1) and (2), comes to

$$\underbrace{2(2(2\ldots(2}_{k-1 \text{ of these}}(1)+\underbrace{2)+4)+8)+\ldots+2^p)}_{p \text{ of these}}+\underbrace{2^p)+\ldots+2^p)+2^p}_{k-p-1 \text{ of these}}$$

$$= \underbrace{2^0 \cdot 2^{k-1}+2^1 \cdot 2^{k-2}+\ldots+2^{p-1} \cdot 2^{k-p}}_{p \text{ of these}}+2^p \cdot 2^{k-p-1}+\ldots+2^p \cdot 2^0$$

$$= \underbrace{2^{k-1}+\ldots+2^{k-1}}_{p \text{ of these}}+2^{k-1}+2^{k-2}+\ldots+2^p$$

$$= p2^{k-1}+\sum_{j=1}^{k-p} 2^{k-j} = p2^{k-1}+\sum_{j=p}^{k-1} 2^j$$

$$= p2^{k-1}+2^k-1-(2^p-1) = p2^{n-p-1}+2^{n-p}-2^p.$$

Note that the above total is short by a single additive ε from the distance $d(r_1,s_1)$. OPT, on the other hand, had a final bottleneck cost of $2^p+(2^{p+1}-1)\varepsilon$. Hence, GRIN$^*(2)$/OPT $\approx (p2^{n-p-1}+2\cdot 2^{n-p-1}-2^p)/2^p = (p+2)2^{n-2p-1}-1$. □

4.3 Lower Bounds for GRIN$^*(t)$

Extending the instance of Theorem 5 to larger thresholds simply entails lengthening the segments, as illustrated in Figure 4. As the threshold increases we also get a lower ratio. Specifically, for a threshold of t, a segment of $\lceil \log t \rceil + 1$ requests is needed to force each free pass, yielding the following corollary to Theorem 5.

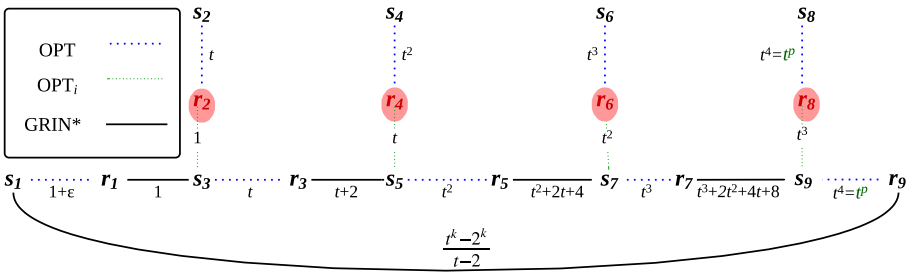

Fig. 5 Requests are numbered in order of arrival and requests skipped by GRIN* are highlighted. Note that for simplicity the epsilon terms in the vertical distances from Figure 3 have been omitted from this illustration. The special case of the instance where $p = k-1$ and hence GRIN$^* = (t^k - 2^k)/(t-2)$, is pictured here. But the general cost for any $p \le k-1$ is GRIN$^* = \sum_{j=0}^{p-1} t^j 2^{k-j-1} + \sum_{j=p}^{k-1} t^p 2^{k-j-1} \ge \frac{t^p-2^p}{t-2} + t^p(2^{k-p}-1)$.

Corollary 6 GRIN$^*(t)$ *has an* SOS(p)*-competitive ratio no better than* $2^{n-p(\lceil \log t \rceil+1)-1}$.

A somewhat tighter lowerbound on GRIN$^*(t)$, for all $t > 2$, is based on the instance of Theorem 6 in Figure 3. In particular, we extend the distance from r_2 to s_2 in that instance to $t+(t+1)\varepsilon$, and use t for the distance from r_3 to s_3. We follow these initial distances and extend the remaining distances as follows. Explicitly, $d(r_1,s_1) = 1+\varepsilon$, $d(r_i,s_i) = t^{(i-1)/2}$ for odd $3 \le i \le 2p+1$, $d(r_i,s_i) = t^{i/2} + \varepsilon \sum_{j=0}^{i/2} t^j$ for even $i \le 2p$, $d(r_i,s_{i+1}) = t^{i/2-1}$ for even $i \le 2p$, and $d(r_i,s_{i+2}) = 2d(r_{i-2},s_i) + d(r_i,s_i)$ for odd $3 \le i \le 2p+1$. This completes the portion of the instance with even-numbered requests arriving above the line, all of which GRIN$^*(t)$ will pass on. (See Figure 5, though note that epsilon terms on the distances other than $d(r_1,s_1)$ have been omitted, for simplicity.) The remaining requests all arrive on the line with distances defined analogously to those of Equation (2) in Theorem 6 as follows. For $i = 2p+2,\ldots,n$, we define $d(r_i,s_i) = t^p$ (since there will be no more passes these vertical distances need no longer increase) and $d(r_i,s_{i+1}) = 2d(r_{i-1},s_i) + t^p$ (equivalent to the total distance from r_i back to r_1, as usual). These adjustments yield the following corollary to Theorem 6.

Corollary 7 *Suppressing epsilon terms, for* $t > 2$ GRIN$^*(t)$ *has an* SOS(p)*-competitive ratio no better than*

$$\frac{t^p - 2^p}{(t-2)t^p} + 2^{n-2p} - 1.$$

Proof As in Figure 3, Figure 5 illustrates the special case of $p = k-1 = \frac{n-1}{2}$, where, following the same reasoning as in Theorem 6,

$$\text{GRIN}^*(t) = \sum_{j=0}^{k-1} t^j 2^{k-j-1} = \frac{t^k - 2^k}{t-2}.$$

But for general $p \le k-1$, the skipped requests beyond p that are pictured above the main line segment would be incorporated into the remainder of the line segment, as

described in the proof of Theorem 6, yielding a bottleneck cost of

$$\begin{aligned}\textsc{Grin}^*(t) &= \sum_{j=0}^{p-1} t^j 2^{k-j-1} + \sum_{j=p}^{k-1} t^p 2^{k-j-1} \\ &\geq \sum_{j=0}^{p-1} t^j 2^{p-j-1} + t^p \sum_{j=0}^{k-p-1} 2^{k-p-j-1} \\ &= \frac{t^p - 2^p}{t-2} + t^p(2^{k-p}-1).\end{aligned}$$

To derive the final ratio, divide by the cost of OPT, which is t^p (suppressing epsilons). □

5 Conclusion

This work investigates the relationship between two forms of bicriteria analysis for online minimum bottleneck matching: the traditional one where resource augmentation comes in the form of added resources, and the one we propose here, where we are effectively allowing for some "degradation of service" in exchange for being online. Since the service provider is able to decline a number of requests upon arrival, this model may be more relevant to some practitioners than having to service all requests, but needing to supplement their resources for doing so. In addition, algorithms that perform well against an adversary with half the servers per server location, may depend on the assumption that there are at least two servers per server location, and may not readily translate to the native online matching setting. It is also interesting to consider the implications of the extra power afforded under the SoS(p) model, namely the responsibility to wield it wisely. While the greater freedom allowed under the SoS(p) model is appealing, more limited freedom can have the benefit of restricting the number of poor decisions a greedy algorithm can make. This may make designing good algorithms for the SoS(p) model more challenging.

We showed in this work that the most natural greedy algorithm, GRINN(t), has a competitive ratio that is exponential in the number of requests assigned, and that this ratio is tight, implying that each free pass improves the performance ratio of GRINN(t) by a factor of 2. We proposed an improvement to this algorithm, GRIN*(t), and show that it does better under certain circumstances. Future work includes developing additional algorithms that may be constant-SoS(p)-competitive. This work provides further evidence for the fact that the bottleneck objective for online matching is quite different and far more elusive than the utilitarian total-cost objective. We believe that with the internet-induced trend toward decentralization, systems with multiple autonomous agents, and pervasive game theoretic concerns about fairness, a better understanding of optimizing for egalitarian objectives will become essential, and developing benchmarks of practical relevance is one path to gaining such an understanding.

Acknowledgements A preliminary version of this work was published in the proceedings of the 8th Annual International Conference on Combinatorial Optimization and Applications, COCOA 2014. We would like to thank the anonymous reviewers for their careful reading of our manuscript and their many insightful comments and suggestions.

References

B. M. Anthony and C. Chung. Online bottleneck matching. *J. Comb. Optim.*, 27(1):100–114, 2014. Preliminary version appeared in *COCOA*, pp. 257-268, 2012.

C. Chung, K. Pruhs, and P. Uthaisombut. The online transportation problem: On the exponential boost of one extra server. In *LATIN*, pages 228–239, 2008.

N. R. Devanur and T. P. Hayes. The adwords problem: Online keyword matching with budgeted bidders under random permutations. In *Proceedings of the 10th ACM Conference on Electronic Commerce*, EC '09, pages 71–78, 2009.

C. G. Fernandes and R. C. S. Schouery. Second-price ad auctions with binary bids and markets with good competition. *Theor. Comput. Sci.*, 540–541:103–114, 2014.

B. Fuchs, W. Hochstättler, and W. Kern. Online matching on a line. *Theor. Comput. Sci.*, 332(1–3): 251–264, 2005.

H. N. Gabow and R. E. Tarjan. Algorithms for two bottleneck optimization problems. *Journal of Algorithms*, 9(3):411–417, 1988.

R. S. Garfinkel. An improved algorithm for the bottleneck assignment problem. *Operations Research*, 19 (7):pp. 1747–1751, 1971.

G. Goel and A. Mehta. Online budgeted matching in random input models with applications to adwords. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 982–991, 2008.

A. V. Goldberg, J. D. Hartline, and A. Wright. Competitive auctions and digital goods. In *Proceedings of the 12th annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, pages 735–744, 2001.

A. Gu, A. Gupta, and A. Kumar. The power of deferral: maintaining a constant-competitive Steiner tree online. In *Proceedings of the 45th annual ACM Symposium on Theory of Computing*, pages 525–534. ACM, 2013.

A. Gupta, A. Kumar, and C. Stein. Maintaining assignments online: Matching, scheduling, and flows. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014*, pages 468–479, 2014.

J. D. Hartline and T. Roughgarden. Simple versus optimal mechanisms. In *ACM Conference on Electronic Commerce*, pages 225–234, 2009.

R. Idury and A. Schaffer. A better lower bound for on-line bottleneck matching, manuscript. http://www.ncbi.nlm.nih.gov/core/assets/cbb/files/Firehouse.pdf, 1992.

B. Kalyanasundaram and K. Pruhs. Online weighted matching. *J. Algorithms*, 14(3):478–488, 1993. Preliminary version appeared in *SODA*, pp. 231-240, 1991.

B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. *J. ACM*, 47:617–643, July 2000a. Preliminary version appeared in *FOCS*, pp. 214-221, 1995.

B. Kalyanasundaram and K. Pruhs. The online transportation problem. *SIAM J. Discrete Math.*, 13(3): 370–383, 2000b. Preliminary version appeared in *ESA*, pp. 484-493, 1995.

B. Kalyanasundaram and K. R. Pruhs. An optimal deterministic algorithm for online b-matching. *Theoretical Computer Science*, 233(1):319–325, 2000c.

S. Khuller, S. G. Mitchell, and V. V. Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theor. Comput. Sci.*, 127:255–267, May 1994.

N. Megow, M. Skutella, J. Verschae, and A. Wiese. The power of recourse for online MST and TSP. In *Proceedings of Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Part I*, pages 689–700, 2012.

A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. Adwords and generalized online matching. *J. ACM*, 54 (5), Oct. 2007.

C. A. Phillips, C. Stein, E. Torng, and J. Wein. Optimal time-critical scheduling via resource augmentation. *Algorithmica*, 32(2):163–200, 2002. Preliminary version appeared in *STOC*, pp. 140-149, 1997.

T. Roughgarden and É. Tardos. How bad is selfish routing? *J. ACM*, 49(2):236–259, 2002. Preliminary version appeared in *FOCS*, pp. 93-102, 2000.

A Limitations on Requests GRIN$^*(t)$ can Skip

We formalize some observations about which requests GRIN$^*(t)$ can and cannot skip. These observations may be useful in obtaining better bounds on the performance of GRIN$^*(t)$, or provide insight into algorithms which may be more promising in the SoS(p) model. As noted in Lemma 1, r_1 must be assigned by GRIN$^*(t)$. We now show that GRIN$^*(t)$ for $t \geq 2$ cannot pass on both requests r_2 and r_3.

Lemma 3 *There is no instance in which* GRIN$^*(t)$ *for* $t \geq 2$ *passes on both requests* r_2 *and* r_3.

Proof Note that if $p < 2$, then the model itself immediately precludes passing on two requests. Thus we may assume $p \geq 2$ for the remainder of the proof.

We may restrict our proof to the case of $n = 3$, since additional servers/requests only allow for more potential assignments. We know that r_1 is assigned (as guaranteed by Lemma 1). We then assume r_2 and r_3 are not assigned, and show that leads to a contradiction.

Our requests are thus labeled r_1, r_2, r_3 in order of arrival. Our servers are labeled s_a, s_b, s_c. Without loss of generality we let r_1 be assigned to s_a. Thus, $d(r_1, s_a) = \text{OPT}_1$.

Since r_2 is not assigned by GRIN$^*(t)$, the available servers must be outside of the threshold bound, and thus

$$d(r_2, s_b) > t\text{OPT}_2, \text{ and} \tag{3}$$
$$d(r_2, s_c) > t\text{OPT}_2. \tag{4}$$

Similarly, r_3 is not assigned by GRIN$^*(t)$, so

$$d(r_3, s_b) > t\text{OPT}_3, \text{ and} \tag{5}$$
$$d(r_3, s_c) > t\text{OPT}_3. \tag{6}$$

Consider the assignments made by OPT_2. Since r_2 is not assigned by GRIN$^*(t)$, OPT_2 must assign r_2 to s_a. (If not, GRIN$^*(t)$ would have made the same assignment of r_2 as OPT_2.) Hence, r_1 must be assigned to either s_b or s_c by OPT_2, giving

$$\min\{d(r_1, s_b), d(r_1, s_c)\} \leq \text{OPT}_2.$$

Now consider the assignments made by OPT_3. Since r_3 is not assigned by GRIN$^*(t)$, OPT_3 must assign r_3 to s_a, giving $d(r_3, s_a) \leq OPT_3$. Thus, OPT_3 must assign r_2 to either s_b or s_c. Hence, at least one of $d(r_2, s_b)$ and $d(r_2, s_c)$ is at most OPT_3. Since both $d(r_2, s_b)$ and $d(r_2, s_c)$ are at least $t \cdot OPT_2$, this guarantees that

$$OPT_3 > t \cdot OPT_2.$$

By triangle inequality, $d(r_3, s_b) \leq d(r_3, s_a) + d(s_a, r_1) + d(r_1, s_b)$. Likewise, $d(r_3, s_c) \leq d(r_3, s_a) + d(s_a, r_1) + d(r_1, s_c)$. Combining the observations that $d(r_3, s_a)$ is at most OPT_3, $d(s_a, r_1)$ is at most OPT_1, and that at least one of $d(r_1, s_b), d(r_1, s_c) \leq OPT_2$ gives that at least one of $d(r_3, s_b)$ and $d(r_3, s_c)$ is at most $OPT_3 + OPT_1 + OPT_2$. Since the OPT_i are nondecreasing, $OPT_3 + OPT_1 + OPT_2 \leq OPT_3 + 2OPT_2$, and using the fact that $OPT_3 > t \cdot OPT_2$, we get that at least one of $d(r_3, s_b)$ and $d(r_3, s_c)$ is at most $2 \cdot OPT_3$, which (since $t \geq 2$) contradicts the previous observation that $d(r_3, s_b) > t \cdot OPT_3$ and $d(r_3, s_c) > t \cdot OPT_3$. □

It is natural to consider extensions of this lemma, such as the analogous statement about r_4 and r_5 and subsequent pairings, and to ask whether the parameter t may dictate what fraction of the requests may be skipped at any point in time. We show, however, that it is not in fact true that GRIN$^*(t)$ for $t \geq 2$ in the SoS(p) model cannot pass on both requests r_4 and r_5. (This also invalidates the more general possibility that by request k, GRIN$^*(t)$ can have skipped at most $\lfloor k/t \rfloor$ requests.)

We now show by example, illustrated in Figure 6, that GRIN$^*(2)$ for in the SoS(p) model can in fact pass on both requests r_4 and r_5. Observe that $\text{OPT}_1 = 1, \text{OPT}_2 = 1 + \varepsilon, \text{OPT}_3 = 2, \text{OPT}_4 = 2 + 3\varepsilon, \text{OPT}_5 = 2 + 3\varepsilon$. In the final OPT, that is, OPT_5, r_i is assigned to s_i for $i = 1 \ldots n$. GRIN$^*(2)$ assigns r_1 to s_5. Since OPT_2 is $1 + \varepsilon$, GRIN$^*(2)$ skips request r_2 since the nearest available server is at $2 + 3\varepsilon > 2 \cdot (1 + \varepsilon)$ away. Request r_3 is then greedily assigned to s_4, a distance of 1 away. Request r_4 is skipped since there are no available servers within $2 \cdot (2 + 3\varepsilon)$, as is r_5 for the same reason. Thus, not only are both r_4 and r_5 skipped, but also three requests were skipped out of five, indicating that there are instances where more than half of the total requests are skipped by GRIN$^*(2)$.

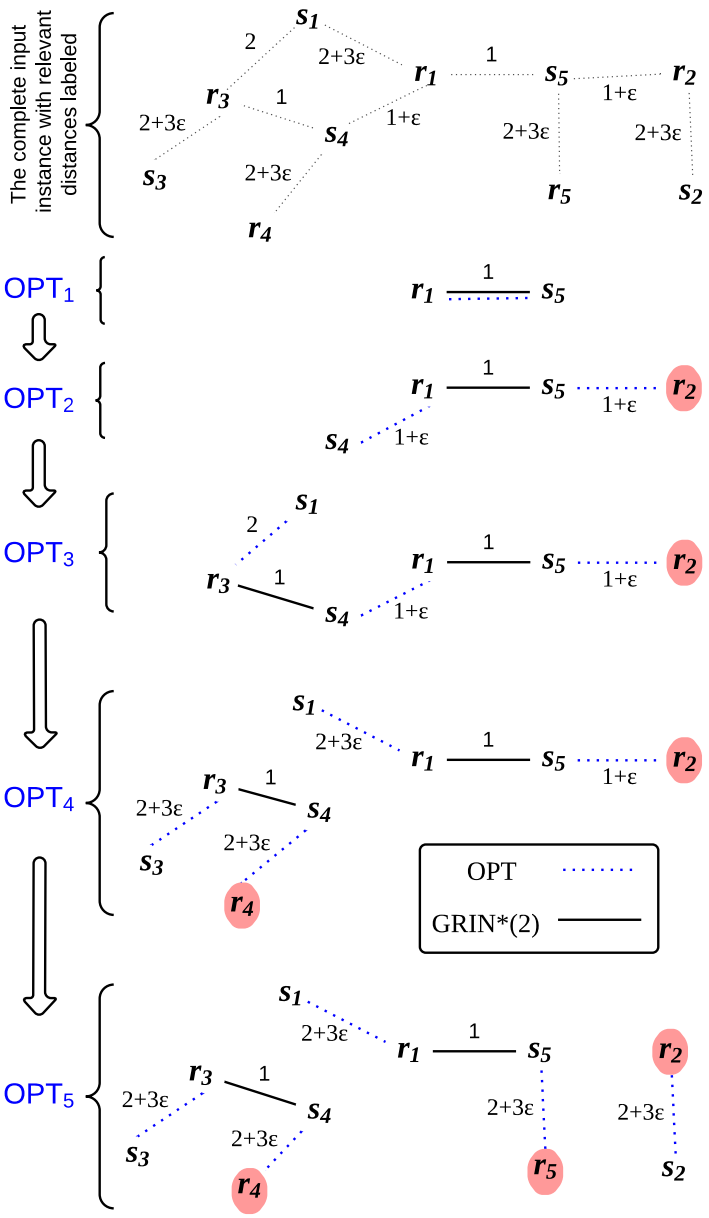

Fig. 6 An instance where $n = 5$ in which more than half the requests are skipped by GRIN$^*(2)$. Requests skipped by GRIN$^*(2)$ are highlighted.

B Upper bound for the basic Greedy Algorithm

The proof of the following theorem closely echos that of the proof of Theorem 2 in Section 3. This theorem essentially closes the gap that remained in Anthony and Chung (2014) between the lower bound on the competitiveness of GREEDY (of 2^{n-1}) and the upper bound.

Theorem 7 *The basic algorithm* GREEDY*, which simply assigns requests to the closest available server, is* $2^n - 1$*-competitive for online minimum bottleneck matching, and this is tight. (Note that this guarantee is without the aid of resource augmentation.)*

Proof We will show inductively that the assignment cost of r_i is at most $(2^i - 1) \cdot \text{OPT}$ for $i = 1 \ldots n$.

Base case: $i = 1$. By definition of GREEDY, the first request r_1 is always assigned with a cost of at most OPT, which satisfies the claim.

Inductive case: Assume the assignment cost of r_j is at most $(2^j - 1) \cdot \text{OPT}$ for all $1 \leq j \leq i$. Consider the assignment cost of r_{i+1}. Let s_{i+1} be the server that OPT_{i+1} assigns to r_{i+1}. If s_{i+1} is available, the assignment cost is at most $\text{OPT}_{i+1} \leq \text{OPT}$.

Thus, we may assume that s_{i+1} is not available, and is hence used by some r_j with $j \leq i$. We thus consider the graph consisting of $i+1$ edges of OPT_{i+1} and the i edges assigned thus far by GREEDY. Since r_{i+1} is not yet matched by GREEDY, there must be a path in this graph from r_{i+1} to some s_a that is used by OPT_{i+1} but not currently matched by GREEDY. Observe that said path must begin with an edge in OPT_{i+1} and alternate between edges in OPT_{i+1} and edges in GREEDY, terminating with an edge in OPT_{i+1}.

We can thus use triangle inequality to compute the distance from this available s_a to r_{i+1}, giving an upper bound on the assignment cost of r_{i+1}. Since all distances are nonnegative, additional edges either cause the total cost to increase or stay the same. Thus, in the worst case, the path includes all of the i assignments already made, as well as the $i+1$ edges in OPT_i.

By strong induction, the total cost of the i assignments already made is at most

$$\sum_{h=1}^{i} (2^h - 1) \cdot \text{OPT} = \left(2^{i+1} - i - 2\right) \cdot \text{OPT}.$$

Since the OPT_i are non-decreasing, the cost of the $i+1$ edges from OPT_{i+1} is no more than $(i+1) \cdot \text{OPT}$. Adding this to the cost above gives an upper bound on the total distance (and thus assignment cost) of r_{i+1} to s_a of $(2^{i+1} - 1) \cdot \text{OPT}$, completing the inductive proof.