1-2008

Stochastic Analyses for Online Combinatorial Optimization Problems

Naveen Garg
Indian Institute of Technology - Delhi

Anupam Gupta
Carnegie Mellon University, anupamg@cs.cmu.edu

Stefano Leonardi
University of Roma "La Sapienza"

Piotr Sankowski
University of Roma "La Sapienza"

Published In
Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms , 942- 951.

Stochastic Analyses for Online Combinatorial Optimization Problems

Naveen Garg* Anupam Gupta† Stefano Leonardi‡ Piotr Sankowski‡

Abstract

In this paper, we study online algorithms when the input is not chosen adversarially, but consists of draws from some given probability distribution. While this model has been studied for online problems like paging and k-server, it is not known how to beat the $\Theta(\log n)$ bound for online Steiner tree if at each time instant, the demand vertex is a uniformly random vertex from the graph. For the online Steiner tree problem, we show that if each demand vertex is an independent draw from some probability distribution $\pi : V \to [0, 1]$, a variant of the natural greedy algorithm achieves $\mathbf{E}_\omega[\mathcal{A}(\omega)]/\mathbf{E}_\omega[\mathsf{OPT}(\omega)] = O(1)$; moreover, this result can be extended to some other subadditive problems. Both assumptions that the input sequence consists of *independent* draws from π, and that π is known to the algorithm are both essential; we show (almost) logarithmic lower bounds if either assumption is violated. Moreover, we give preliminary results on extending the Steiner tree results above to the related "expected ratio" measure $\mathbf{E}_\omega[\mathcal{A}(\omega)/\mathsf{OPT}(\omega)]$. Finally, we use these ideas to give an average-case analysis of the Universal TSP problem.

1 Introduction

The notion of competitive analysis is perhaps the central notion of online algorithms [41, 26, 30]: given an online algorithm $\mathcal{A}$ (perhaps a randomized one), the *competitive ratio* is defined as

$$\max_\omega \frac{\mathbf{E}_r[\mathcal{A}(\omega, r)]}{\mathsf{OPT}(\omega)}, \tag{1.1}$$

where r is the set of random coins flipped by the algorithm, the maximum is taken over all inputs ω, and $\mathsf{OPT}(\omega)$ is the optimal solution value on the input ω. This idea of comparing the performance of an online algorithm (which knows nothing about the future) to the optimal solution built in hindsight has led to crisp and clean problems, and strong upper and lower bounds on the competitive ratio are known for most problems of interest. However, while the strict definition of competitive ratio has led to much progress, it has had its drawbacks. The results in the online model are often unduly pessimistic, and the strict definition of competitive ratio does not allow us to make fine-grained distinctions between competing algorithms.

One of the reasons for the introduction of competitive analysis was that classical online problems like list-update, paging and k-server were easy when the inputs were drawn from a known probability distribution, and the hope was to extend our understanding when we did not know the input distribution precisely. (See [6, 13, 22] or Section 1.2 for many references to weaken the competitive analysis framework and strike a happy median between full-information and no-information.) However, for some combinatorial optimization problems, we do not understand even the case when the inputs are independently drawn from a probability distribution π given as input. An excellent example is the *online Steiner tree* problem, which has been studied both in the offline (see [37] and the references therein) and the online [38, 20, 3] case. In this work, we seek answers to the following question: *If the online Steiner tree requests are vertices of a graph drawn uniformly at random, can we do better than the* $\Theta(\log n)$ *online greedy algorithm?* (E.g., one may be streaming a video over a network to customers arriving from a known distribution; when a customer arrives she has to be served immediately.) In general, the goal is to formally study the interplay between stochastic information and online algorithms for combinatorial optimization problems in general. (Individual problems along

*Indian Institute of Technology, New Delhi, India. Part of this work was done when visiting the Dipartimento di Informatica e Sistemistica, University of Rome "La Sapienza", and the Max-Planck-Institut für Informatik, Saarbrücken, Germany.

†Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213. Research was partly supported by NSF awards CCF-0448095 and CCF-0729022, and an Alfred P. Sloan Fellowship. Part of this work was done when visiting the Dipartimento di Informatica e Sistemistica, University of Rome "La Sapienza".

‡Dipartimento di Informatica e Sistemistica, University of Rome "La Sapienza", Via Salaria 113, 00198 Rome, Italy. This work was partially supported by the EU within the 6th Framework Programme under contract no. 001907 "Dynamically Evolving, Large Scale Information Systems" (DELIS)

these lines have indeed been studied before [25, 31, 32]: see Section 1.2 for similarities and differences.)

Our Model. In order to state the questions and answers, let us formalize the model in which we work. We assume that requests are drawn from the universe U, and there is a probability distribution $\pi : U \to [0,1]$ over the space of requests.[1] The "adversary" generates the input sequence $\omega = \omega_1, \omega_2, \ldots$, by deciding on the length k of the input sequence, and then drawing k requests independently and identically from the probability distribution π. The adversary is allowed to choose any distribution π over the space of requests, but the online algorithm are allowed to sample from this distribution. Given $\ell \in \mathbb{Z}$, we use π^ℓ to denote the set U^ℓ of length-ℓ sequences from U endowed with the natural product measure, and hence $\omega \in \pi^\ell$ will imply that ω is chosen by taking ℓ i.i.d. samples from π.

Given an online algorithm ALG for problem Π, the *ratio-of-expectations* (RoE) is

$$\mathsf{RoE}(\mathsf{ALG}) = \max_{\pi} \max_{k} \frac{\mathbf{E}_{\omega \in \pi^k, r}[\mathsf{ALG}(\omega, r)]}{\mathbf{E}_{\omega \in \pi^k}[\mathsf{OPT}(\omega)]}. \tag{1.2}$$

Here r are the random coins of the algorithm. Hence, given any sequence length k (unknown to the algorithm), this quantity measures the expected cost of the algorithm over length k sequences, compared to the expected optimal cost for these sequences. This is a special case of the *diffuse adversary* of [29], and has been recently studied in the context of several scheduling problems (see, e.g., [39, 33]). Also, note the syntactic/philosophical similarity to the Max/Max objective of [5].)

1.1 Our Results

We show this result for the stochastic online Steiner tree problem:

THEOREM 1.1. (STEINER TREE RoE) *There is an online algorithm $\mathcal{A}$ that takes as input a graph $G = (V, E)$ and probabilities $\{\pi_v\}_{v \in V}$ so that if at each time step the input vertex is v with probability π_v (i.e., the input consists of i.i.d. draws from π), then $\mathsf{RoE}(\mathcal{A}) = O(1)$.*

These algorithms are related to work done on the maybecast problem, and work in offline stochastic problems. As for those cases, we extend the Steiner tree result to other problems as well.

THEOREM 1.2. (EXTENDING TO OTHER PROBLEMS) *There are online algorithms for Steiner Forest, Facility Location and Vertex Cover with $\mathsf{RoE} = O(1)$.*

[1] In the following, we use "requests," "demands," and "clients" interchangeably.

We show that some access to the distribution, and also the independence is necessary: if the input consists of i.i.d. draws from a fixed but unknown distribution, or if the draws are not independent but drawn from some given Markov chain, then there is an $\Omega(\frac{\log n}{\log \log n})$ lower bound for Steiner tree.

We also give initial results in the model where we are given the input length k and distribution π (given only as a black-box), and we want to minimize the *expected ratio* $\mathsf{EoR}(\mathcal{A}) = \mathbf{E}_{\omega \in \pi^k}[\mathcal{A}(\omega)/\mathsf{OPT}(\omega)]$.

THEOREM 1.3. *There is an $\mathsf{EoR} = O(\log \log n)$-algorithm for the Steiner tree problem in the above model.*

This result, which appears in Section 5, uses fairly different techniques from the results above: it is an interesting question to extend this result to match those in the RoE setting. Finally, we show how to get results for the so-called Universal TSP problem, where we have to give a single permutation τ such that if vertices arrive independently of each other (each with probability p_v), then the expected length of the tour induced by τ on this random set is $O(1)$ times the expected length of the optimal tour.

1.2 Related Work

For books and surveys on *online algorithms*, see, e.g., [6, 13, 1, 2], and the many references therein. There have been many prior attempts to relax the notion of competitive analysis: e.g., using *access graphs and working sets* [7, 11, 12, 23, 4] or *Markov distributions* [27] to capture locality of reference in paging-type problems, *comparative analysis* to compare against the best algorithm in some given class [29], *resource augmentation* to level the playing field by giving the online algorithm more power than the offline one [24, 34, 43], or the idea of *statistical adversaries* [35, 9] (where the online sequences have certain statistical properties) or *diffuse adversaries* [29, 44] (where the sequences are generated from a small set of distributions known to the algorithm). In most of these cases, it was easy to handle the case when each request is an i.i.d. sample from a given distribution, and the challenge was to obtain a happy median between this simple case and the unbounded adversary. In this paper, we consider problems which are interesting even with i.i.d. draws from a fixed distribution. A different approach was the Max/Max objective of Ben-David and Borodin [5], which has similarities to our RoE objective.

While *stochastic optimization* problems have a long history in the operations research community, they have been studied in the algorithms community only for a few years; see, e.g., [21, 36, 19, 40]. The setting sounds similar to the online case: the input is gradually

revealed over k stages, and partial decisions are made as time goes on—however, the goal is to have a feasible solution at the end and only the rising costs force us to make decisions early in the game, whereas in the online case we need a feasible solution after every prefix but do not have to worry about the costs increasing. Also, while the offline work sought to enlarge the scope of inputs—instead of one input, we wanted to handle a class of inputs "on the average"—whereas this work on stochastic online algorithms seeks to to restrict the power of the adversary: instead of having the adversary hand-pick the demands in the input stream, we now posit that these are drawn at random.

Some of these lines of work are especially close to ours: Karger and Minkoff [25] (see also [16]) give an $O(1)$ approximation for the *maybecast* problem: given a graph and probabilities p_v for each vertex, one needs to fix a path P_v for each vertex v of the graph to the root, such that if each vertex is added to a set S with probability p_v (independently of all other vertices), the cost $\mathbf{E}_S[c(\cup_{v\in S} P_v)]$ is minimized. A minor difference between our problem and theirs is that vertices arrive independently in their case and it is a one-shot problem, whereas we sample a single random vertex each time, and the process goes on for some *unknown* number of rounds. Another difference is that they seek an *approximation algorithm* and hence compare to the optimal expected value achievable by a union of paths fixed before the random set S is chosen, whereas we need to compare our solution to the best Steiner tree on $S \cup \{\text{root}\}$ itself. (These two quantities turn out to be the close, which is part of what we have to prove.)

The papers of Meyerson [31] (for facility location) and Meyerson, Munagala and Plotkin [32] (for the access network design problem) consider the *"random-permutation model"*. Here the adversary picks a set S of demand points (kept secret from the algorithm); the points in S are then presented to the online algorithm in random order. In these papers, the competitive ratio is $O(\log n)$ in the purely adversarial case, but with the random permutation the expected cost of the algorithm becomes $O(1) \times \mathsf{OPT}(S)$. These results are quite strong, since in a loose sense, the adversary is choosing the distribution π_S which is uniformly spread over its support S, and their algorithms gets their performance guarantee without access to this distribution. However, this generality of the definition is also its shortcoming: in the full version of the paper, we give an $\Omega(\frac{\log n}{\log\log n})$ lower bound for Steiner tree in this model.

Finally, some recent work in online scheduling and paging tries to extend work in the RoE model to the "expected-ratio" model EoR (from Section 5)—see, e.g., [39, 42, 33].

2 Online Steiner Tree

In this section we develop algorithms that are competitive under the ratio-of-expectations (RoE) measure (1.2) for the the online Steiner tree problem. We show that while the standard greedy online algorithm performs badly in the stochastic setting, we can devise algorithms that obtain $\mathsf{RoE} = O(1)$ for the problem. In the next section, we extend these ideas to general subadditive problems that admit certain kinds of "good" algorithms.

As a warm-up to illustrate the concepts we will use, we consider the Steiner tree problem; our results would be applicable to a much wider variety of problems. In the online Steiner tree problem, we are given a graph $G = (V, E)$ with and a root vertex r and edge costs/lengths $c : E \to \mathbb{R}_{\geq 0}$; w.l.o.g., we assume that the edge costs satisfy the triangle inequality. The input clients U are the vertices V of the graph, and hence the adversary provides a sequence of vertices (possibly with repetitions) $v_1, v_2, \ldots$ from V. At each point in time t, the online algorithm must maintain a subgraph S_t which connects the root r and all vertices $\{v_1, \ldots, v_t\}$ seen so far in the input sequence; note that the decisions of the online algorithm are irrevocable, and hence $S_t \subseteq S_{t+1}$.

2.1 Online Steiner: Known Sequence Length

We first suppose we know the length k of the input sequence: .

A1. Choose a set of vertices D by drawing from the distribution π independently k times.

A2. Construct a ρ-approximate Steiner tree T_M over the set $D \cup \{r\}$, and set $S_0 = T_M$.

A3. When the actual input sequence of k vertices arrives, run the *greedy algorithm* on this sequence—namely, connect the t^{th} input vertex v_t to the closest vertex in the subtree S_{t-1}.

The algorithm is very similar to those used for *offline* two-stage stochastic problems. Although the proofs from the offline stochastic case cannot be used here directly, since here we have to build the tree in an online fashion as the vertices arrive, we can give the following guarantee:

THEOREM 2.1. (STEINER TREE FOR FIXED k) *The ratio-of-expectations of the above algorithm is* $(2 + \rho)$.

Proof. Let D be the sequence of k "dummy" requests in the anticipatory solution, and let R be the set of k "real" requests arriving online: recall that both are identically distributed. Note also that in the Steiner tree problem, repetitions are irrelevant, and hence we can associate

both D and R with the set of vertices contained in these sequences. It is clear that the cost of the optimal Steiner tree on D has expected cost $\mathbf{E}[\mathsf{OPT}(R)]$, and hence the cost of the anticipatory solution is at most $\rho\mathbf{E}[OPT(R)]$.

Now, to consider the cost of connections added in Step **A3**: since we run the greedy algorithm, if the i^{th} vertex of R is some node v, then the cost incurred is upper bounded by $d(v, D \cup \{r\})$, the distance from v to the nearest vertex in $D \cup \{r\}$. Since each node is an i.i.d. sample from π, by linearity of expectations, the expected cost of the real solution is at most $k \times \mathbf{E}_{v\in\pi, D\in\pi^k}[d(v, D \cup \{r\})]$.

In turn, this expected cost does not decrease if we were to take a smaller set $D' \in \pi^{k-1}$ of $k-1$ samples. Finally, the expression $\mathbf{E}_{v\in\pi, D'\in\pi^{k-1}}[d(v, D' \cup \{r\})]$ is upper bounded by the expression $\mathbf{E}_{D''\in\pi^k}[\frac{1}{k}\cdot MST(D'' \cup \{r\})]$. To see this, we can imagine picking D' and v by picking a set D'' of k samples, and randomly choosing one of them to be the vertex v; moreover, for each vertex in the MST, we "assign" it its parent edge, and note that the distance to its closest vertex is bounded by the length of this edge. Hence the cost of the greedy connections is at most

$$k \times \mathbf{E}_{v\in\pi, D\in\pi^k}[d(v, D)] \le k \times \mathbf{E}_{v\in\pi, D'\in\pi^{k-1}}[d(v, D')]$$
$$\le k \times \mathbf{E}_{D''\in\pi^k}[\tfrac{1}{k} \cdot MST(D'')] = \mathbf{E}_{D\in\pi^k}[MST(D)].$$

And hence the cost of the greedy connections is at most the cost of the MST on the anticipatory set. Since the MST is a 2-approximation to the Steiner tree, this is at most twice the optimum cost, and we get the $(2+\rho)$-approximation, as claimed.

As in earlier proofs of this form [18, 19], we can optimize the number of samples taken in Step **A1** depending on the approximation guarantee ρ; however, we skip these minor optimizations in the interests of clarity. Moreover, since we are in the online setting, we could conceivably compute the optimal Steiner tree to get $\rho = 1$.

Lazy Version: We can change the algorithm so that the edges of T_M are bought only when they are needed for the actual demands: in that case the algorithm is very similar to that of [25]. (Section 1.2 gives a detailed comparison of that work and ours: the main difference is that while we compare the cost of the tree built to $\mathbf{E}_R[\mathsf{OPT}(R)]$, the proof in [25] compares it to the best performance achievable by a set of fixed paths.) Also, if we use a light approximate-shortest-path tree (LAST) [28] in this lazy version, we can ensure that no single demand vertex has to use a path that is much longer than his shortest path to the currently-built network. Details of these extensions will be in the final version.

2.2 Online Steiner: Unknown Sequence Length

In this section, we show how to remove the assumption that we know the number of points in the input sequence. The first idea one might try is to "guess-and-verify": guess that the number of points in the input sequence, and if that is incorrect, double the guess and try again. This turns out to be a bad idea (with our standard example of the line being a counterexample). So instead of "scaling" on the input length, we scale on the cost of the anticipatory solution we build—each time we build an anticipatory solution costing about twice as much as before, and wait until we see as many vertices as in that solution. If we allow ourselves to perform non-polynomial computations, doing this is quite easy: but in the following we will show how to achieve this with polynomial amounts of computaiton.

For the following, let us define $Z_\ell = \mathbf{E}_{\omega\in\pi^\ell}[\mathsf{OPT}(\omega)]$ to be the cost of the optimal Steiner tree on a random sequence ω of ℓ i.i.d. draws from the distribution π. Sub-additivity immediately implies that for any $c \ge 1$, $Z_{c\ell} \le cZ_\ell$. For clarity of argument, we distinguish between sequences ω, the multiset of vertices in it (denoted by $\mathsf{Mset}[\omega]$), and the set of distinct vertices contained in it (denoted by $\mathsf{Set}[\omega]$).

Using arguments used in Theorem 2.1, we know that it is enough to buy a tree on some "dummy" set D at the beginning (such that $r \in D$), and then connect each of the real clients in $\mathsf{Set}[\omega]$ to its closest point in this tree. In fact, the theorem implies that

$$Z'_\ell = \min_{\substack{D_\ell\in\binom{V}{\ell} \\ r\in D_\ell}} (c(MST(D_\ell)) + \mathbf{E}_{\omega\in\pi^\ell}[\sum_{x\in\mathsf{Set}[\omega]} d(x, D_\ell)]) \tag{2.3}$$

is at least Z_ℓ (since it is the cost of a tree that connects ℓ random points), and at most $4Z_\ell$ (by Theorem 2.1). Moreover, if we have a sequence length ℓ in mind, while one can imagine complicated online algorithms that buy edges at various points of time—the analysis shows that it is enough to choose from one of 2^{n-1} distinct policies, each defined by the set $D_\ell \subseteq V$ such that $r \in D_\ell$.

LEMMA 2.1. *Given a length ℓ, the graph $G = (V, E)$, and black-box access to the distribution π, there is an algorithm that runs in time $\ell \times \mathrm{poly}(n, \epsilon^{-1}, \log\delta^{-1})$, and with probability $1-\delta$ outputs a set $\widehat{D}_\ell$ which contains the root r, and which is an $2.92+\epsilon$-approximate minimizer of (2.3).*

Proof. To find a minimizer of (2.3), instead of choosing the input string from the distribution π^ℓ, sample N sequences $\omega_1, \omega_2, \ldots, \omega_N$ from π^ℓ (for a value of N to be fixed later), and consider the set D that c-minimizes

the "sample average"

(2.4)

$$\min_{D \in \binom{V}{\ell}} \left(c(MST(D)) + \frac{1}{N} \sum_{i=1}^{N} \Big[\sum_{x \in \mathsf{Set}[\omega_i]} d(x, D) \Big] \right).$$

First, we claim we can find a solution D that approximately minimizes (2.4): note that we merely have an instance of the so-called *single-source rent-or-buy problem*. For this problem, we have a deterministic 2.92-approximation [10].

Secondly, we want to claim that this minimizer for (2.4) is also a minimizer of (2.3) with high probability. This is not true in general, but if we repeat the above sampling process $\tau = \epsilon^{-1} \log \delta^{-1}$ times and take the minimizer that results in the lowest value of (2.4) among these τ runs, we can use a "sample average" theorem of Charikar et al. [8, Theorem 2] to argue that for a suitable value of N, this minimizer will also be an $2.92 + \epsilon$-minimizer of the expression (2.3) with probability $1-\delta$. Let us call this set $\widehat{D}_\ell$. (Note that we *do not* claim that the values of the expressions (2.3) and (2.4) are close for this set, just that this approximate minimizer for the latter is also one for the former.)

It remains to figure out a suitable value of N: in our case, there are 2^{n-1} possible choices for D. Also, for each choice D, we want the difference in the objective function value (2.3) between picking D and picking the empty set $\emptyset$ to be at most $\lambda \times c(MST(D))$; in this case it is clear that this value $\lambda \leq \min\{\ell, n\}$. Plugging these values into [8, Theorem 2], it suffices to choose N to be $\Theta(\epsilon^{-4} \lambda^2 \tau \log(2^{n-1}) \log \delta^{-1}$, which is $\mathrm{poly}(n, \epsilon^{-1}, \log \delta^{-1})$. Hence, if we can sample each sequence in time $T(\ell)$, we can find a $2.92 + \epsilon$-minimizer for (2.3) in time $T(\ell) \times \mathrm{poly}(n, \epsilon^{-1}, \log \delta^{-1})$ with probability at least $1 - \delta$. Since $T(\ell) \leq \ell$, the claim follows: in the full version of the paper, we show how to reduce the time to get one sample from ℓ-length strings of a product distribution.

Lemma 2.2. *Given a length ℓ, the graph $G = (V, E)$ and probabilities π_v of each vertex arriving, there is a $\mathrm{poly}(n, \ell, \log \delta^{-1})$-time algorithm that outputs an estimate $\widehat{Z}_\ell \in [1, 11.68 + 4\epsilon] \times Z_\ell$ with probability $1 - \delta$.*

Proof. To obtain a $11.68 + 4\epsilon$-approximation to Z_ℓ, it enough to find a $2.92 + \epsilon$-approximate minimizer $\widehat{D}_\ell$ for the expression Z'_ℓ in (2.3), and this is done in Lemma 2.1. Now, it remains to calculate the value of (2.3) for this set $\widehat{D}_\ell$: this can be easily done given that we know the probabilities π_v and can calculate the probability $p_v = 1 - (1 - \pi_v)^\ell$ that v appears in a sequence of length ℓ, and then use linearity of expectation. (Note that this is the first time we used the actual values π_v.) This is our estimate $\widehat{Z}_\ell$.

B1. Set $i = 0$ and $S_0 = \emptyset$. We process the input clients $\{v_1, v_2, \ldots\}$ one by one.

B2. Let v_k be the k-th client and let $S_k := S_{k-1}$.

B3. If $k > t_i$ then goto Step **B5**

B4. Connect the k-th input vertex v_k *greedily* to the closest vertex in the tree S_k.

B5. Set $i \leftarrow i + 1$.

B6. Construct a ρ-approximate Steiner tree T_{t_i} over the set $\widehat{D}_{t_i}$. Set $S_k := S_k \cup T_{t_i}$. Goto Step **B4**.

Figure 1: Algorithm for Unknown Sequence Lengths

In order to compute the estimate $\widehat{Z}_\ell$, when it is needed, we use the above lemma with $\delta = \frac{1}{(\ell+1)\ell \log n}$. Note, that $\sum_{\ell=1}^{\infty} \frac{1}{\ell(\ell+1)\log n} = \frac{1}{\log n}$. Hence, we get:

Fact 2.1. *For the setting $\delta^{-1} = (\ell + 1)\ell \log n$, all estimates $\widehat{Z}_\ell$ are correct with probability $1 - 1/\log n$.*

Now we define $t_0 = 0$, $t_1 = 1$, and t_i to be the smallest value ℓ such that $\widehat{Z}_\ell \geq 2^i \cdot \widehat{Z}_1$. The following fact also uses subadditivity:

Fact 2.2. $c(MST(\widehat{D}_{t_i})) \leq \widehat{Z}_{t_i} \leq \widehat{Z}_1 + 2^i \widehat{Z}_1 \leq \widehat{Z}_1 + 2\widehat{Z}_{t_{i-1}} \leq 3\widehat{Z}_{t_{i-1}}$.

We run the algorithm from Figure 1 in parallel with the standard greedy algorithm [2]: The following theorem is almost immediate from the discussion above.

Theorem 2.2. (Steiner Tree for Unknown k) *Suppose the final length of the input sequence is k: the expected total cost incurred by the above algorithm is $O(1) \times Z_\ell$.*

Proof. When all estimates $\widehat{Z}_\ell$ fall into the ϵ bound we can bound the cost of the algorithm in the following way. Let the final length $k \in (t_{i^*-1}, t_{i^*}]$. For any $j \leq i^*$, the expected cost for the input elements at positions $(t_{j-1}, t_j]$ is at most $\widehat{Z}_{t_j}$—note that here we account for both the anticipatory network built in Step **B6**,

[2] By "run both algorithms in parallel", we mean that we run the two algorithms as independent instances feeding the input stream to both instances, but pausing the more expensive run when the cheaper one has finished serving all the requests seen thus far. This standard trick ensures that we never pay more than $2\times$ the cost of the greedy algorithm in the *worst case*: as we see, we do much better on average.

as well as the connection costs incurred in Step **B4**. Summing this over all the relevant values of j, the total cost is at most $2\,\widehat{Z}_{t_{i^*}} \le 2 \cdot 3 \cdot \widehat{Z}_{t_{i^*-1}} \le 6\widehat{Z}_\ell \le (70.08 + \epsilon) \times Z_\ell$. However, with probability at most $1/\log n$ (see Fact 2.1) we fail to compute the estimates $\widehat{Z}_\ell$, we pay $O(\log n)$ times the optimum cost (since we are running in parallel with the greedy algorithm). Since for each input sequence we pay $O(\log n)$ times more with probability at most $\frac{1}{\log n}$, this contributes only $O(\mathsf{OPT})$ more to the cost, and completes the proof of the theorem.

3 Extending to Sub-additive Problems

In this section, we show how to extend the results in the Steiner tree case to a general subadditive problem Π. Let U denote a universe of clients (these make up the input sequence) and E a set of elements used to define a solution. For a subset of clients $U' \subseteq U$, define $\mathsf{Sols}(U') \subseteq 2^E$ as the set of all possible solutions of U'. The cost of a solution $F \subseteq E$ is given by $c(F) = \sum_{e \in E} c(e)$ where $c(e)$ is the cost of element $e \in E$. Let $\mathsf{OPT}(U') \in \mathsf{Sols}(U')$ be the solution of minimum cost.

We assume the presence of two algorithms associated with the problem Π:

- The first algorithm, $\mathcal{A}$, takes a set of clients $U' \subseteq U$ and outputs a solution $\mathcal{A}(U') \in \mathsf{Sols}(U')$, which is an α-approximation to $\mathsf{OPT}(U')$.
- The second is the *augmentation algorithm* $\mathcal{B}$, which takes a solution $E' \in \mathsf{Sols}(E')$, and an input client $x \in U$, and outputs $\mathcal{B}(E', x) \subseteq E$ such that $E' \cup \mathcal{B}(E', x) \in \mathsf{Sols}(U' \cup \{x\})$.

  The augmentation procedure is *monotone* if for any "extra" elements $E'' \subseteq E$, $c(\mathcal{B}(E' \cup E'', x)) \le c(\mathcal{B}(E', x))$. I.e., adding extra elements in the initial solution does not increase the cost of augmentation.

We will also assume the presence of a cost-sharing scheme $\xi(U', x)$ which assigns a *cost share* to each $x \in U$ such that $\xi(U', x) = 0$ for $x \notin U'$, and the fairness property:

$$\sum_{x \in U'} \xi(U', x) \le c(\mathsf{OPT}(U')). \tag{3.5}$$

The main property of interest for the cost-sharing scheme will be the *strictness* with respect to $(\mathcal{A}, \mathcal{B})$:

- Given a set $U' \subseteq U$ and client $x \in U$,

  $$c(\mathcal{B}(\mathcal{A}(U'), x)) \le \beta \cdot \xi(U' \cup \{x\}, x), \tag{3.6}$$

  i.e., the cost $c(\mathcal{B}(\mathcal{A}(U'), x))$ of taking the solution on U' and augmenting it to serve yet another client x is not much more than the cost share of e in the client set $U' \cup \{e\}$.

3.1 Subadditive Problems: Known Sequence Lengths The general algorithm (given in Figure 2) stays essentially the same: let the clients arriving online be $x_1, x_2, \ldots$, let R_t denote the prefix $\{x_1, x_2, \ldots, x_t\}$, and $R = R_k$ be the final sequence.

C1. Choose a set of vertices D by drawing from the distribution π independently k times.

C2. Compute and buy $S_0 = \mathcal{A}(D)$.

C3. When the clients in the input sequence arrive online (with x_t being the t^{th} request)—set $S_t = S_{t-1} \cup \mathcal{B}(S_{t-1}, x_t) \in \mathsf{Sols}(D \cup R_t)$.

Figure 2: Algorithm for Subadditive Problems

THEOREM 3.1. *Given an α-approximation algorithm $\mathcal{A}$, a monotone augmentation procedure $\mathcal{B}$, if there exist β-strict cost-shares ξ (w.r.t. $(\mathcal{A}, \mathcal{B})$), then the above online algorithm is an $(\alpha + \beta)$-approximation algorithm for the subadditive problem Π.*

The proof of the above theorem is deferred to the full version of the paper, as is the proof of the following theorem which essentially follows from previous results on strict cost-shares in [17, 19, 14].

LEMMA 3.1. (APPLYING GENERAL FRAMEWORK) *For the following problems, there exists an α-offline approximation algorithm $\mathcal{A}$, a monotone augmentation procedure $\mathcal{B}$, anda cost-sharing mechanism ξ that is β-strict with respect to $(\mathcal{A}, \mathcal{B})$:*

- *For Uncap. Facility Location (UFL) problem, $\alpha = 3$ and $\beta = 3$;*
- *For Steiner Forest, $\alpha = 2$, and $\beta = 3$.*
- *For Vertex Cover, $\alpha = \beta = 3$. (For t-hypergraph vertex cover, $\alpha = \beta = t + 1$.)*

3.2 Subadditive Problems: Unknown Sequence Lengths Just as in Section 2.2, we can extend algorithms tailored for a fixed value of the sequence length to general cases where we do not know the sequence length in advance. The ideas are similar to those for Steiner tree: the only problem-dependent facts that we need to show are the following.

- The number of samples N we need depends on $\log M$, where M is the number of different anticipatory solutions possible. Note that for Facility Location and Vertex Cover, say, $M \le 2^n$, since

each anticipatory solution is merely a set of nodes; hence this gives us a polynomial dependence.

- The problem should not be too sensitive. In particular, for every choice of the anticipatory solution D, the change in cost between running the augmenting algorithm $\mathcal{B}$ after building D and running $\mathcal{B}$ starting with the empty anticipatory solution, should be at most $\lambda \times$ cost of building D. The number of samples N required depends on poly(λ). For Facility Location and Vertex Cover, we can bound λ by n.

- We should be able to define and solve a "rent-or-buy" versions of the problem Π. Again, for Facility Location and Vertex Cover, this can be done by modifying the current LP rounding solutions. The details appear in the full version.

Of course, we need all these facts only if we want the running time of the online algorithm to be polynomial: if we are allowed to perform unbounded computation (as is sometimes allowed in the design of online algorithms, when we are only interested in the price of the lack of foresight), moving to unknown sequence lengths is easy.

4 Lower Bounds

One may ask whether our assumptions are too restrictive: suppose we did not insist that the requests were independent of the past history, or if the draws were from some fixed *but unknown* distribution—could our results be proved in such models? We answer this in the negative: not only do our results make use of these features, they are *necessary* to get $O(\log^{1-\epsilon} n)$-type results. Here is a summary of results, the details for which will be included in the full version of the paper. All the following results also hold in the RoE model from the following section.

- Consider a model where the (random) demand point v_t at time t is dependent only on v_{t-1}, the demand point at time $t-1$. In other words, given a probability distribution $\pi^{(v)}$ for each vertex $v \in V$, the t^{th} demand point is drawn from the distribution $\pi^{(v_{t-1})}$.

  For such models, we show a $\Omega(\frac{\log n}{\log \log n})$ lower bound on the performance of any online algorithm for the Steiner Tree problem, and the Uncapacitated Facility Location problem. These results almost follow easily from known lower bounds for the two problems [15, 3].

- Another natural question to ask is whether the algorithm really needs (black-box) access to the distribution π: suppose there *exists* some distribution from which the clients are being drawn independently at each step, but we do not allow the algorithm to sample from this distribution. For the Steiner tree, there is an $\Omega(\log n)$ lower bound in this model.

- Finally, a stochastic model that has seen success is the "random permutation" model: here the adversary chooses a set S of clients which is initially not known to the algorithm. The clients in S are then presented to the algorithm *in a random order*. In this model, $O(1)$-competitive ratios were shown for facility location and the access-network design problem [31, 32]. Here we show an $\Omega(\log n)$ lower bound for Steiner tree.

5 Expectation of Ratios: Online Steiner Tree

An objective function that is often more challenging to work with is the *expected ratio* (EoR)

$$\mathsf{EoR}(\mathsf{ALG}) = \max_{\pi} \max_{k} \mathbf{E}_{\omega \in \pi^k} \left[\frac{\mathbf{E}_r[\mathcal{A}(\omega, r)]}{\mathsf{OPT}(\omega)} \right]. \tag{5.7}$$

Note that the outer expectation is over the random choice of inputs, and the inner one is over the random coins r of the online algorithm.

Loosely, it turns out ot be somewhat more difficult, since we have to compare the expected ratio of the cost of the built solution to the cost of the optimal one *on the same set of clients*. The analysis techniques from the RoE case do not seem to extend to this case; for RoE, we were counting on the fact that if we suffer a large cost on an instance, the optimum also suffers a large cost on *some* instance. Clearly such analyses are not enough to prove results for EoR. In this paper, we show that *when the sequence length k is known* to the algorithm, we can get substantial improvements over previous results; it remains an open question to extend this to cases where the sequence length is not known in advance. You should note that the bounds on EoR might be not comparable to the bounds on RoE .

The main difference in our algorithm is that instead of generating one "dummy" set D, we generate L such sets and choose the best of them (for L to be specified later). The algorithm given in Figure 3 is again run in parallel with the greedy solution, to ensure that the worst-case competitive ratio is still $O(\log n)$.

Let us denote by ω the sequence given to the algorithm as input, and $R = \mathsf{Mset}[\omega]$ the corresponding multiset. The following lemma is immediate from symmetry.

LEMMA 5.1. *With probability at least $1 - \frac{1}{L+1}$, the cost of least expensive tree T_{i^*} is no more than $\rho OPT(R)$.*

D1. Sample L different k element multisets $D_1, \ldots, D_L$ from the distribution π.

D2. For each i, find ρ-approximate Steiner tree T_i on the set $D_i \cup \{r\}$, but do not buy these edges.

D3. Choose i^* such that the cost of T_{i^*} is the least, and buy these edges: i.e., set $S_0 = T_{i^*}$.

D4. Connect the k input vertices greedily: connect the t^{th} input vertex v_t to the closest node in the tree S_{t-1} to get the tree S_t.

Figure 3: Algorithm for EoR Steiner Tree

Now we prove that the cost of connecting the real vertices R to the tree T_i is small as well. First, we need the following technical lemma that each vertex of the graph sees at most $O(\log nL)$ real vertices from R closer to it than the closest dummy vertex from any one of the L sets D_i. Formally, given a parameter $\ell \in \mathbb{Z}$, for a vertex $v \in V$ and each index $i \in \{1, \ldots, L\}$, let $N^\ell_{i,v}$ be the (random) multiset of ℓ nearest vertices to v (excluding v) in $Z_i = R \cup D_i$. (Proof will be included in the full version of the paper).

LEMMA 5.2. *Let $\ell = \lceil 3 \log nL \rceil$. Then for all $i \in \{1, \ldots, L\}$ and for all $v \in V$,*

$$\mathbf{P}[N^\ell_{i,v} \subseteq R] \leq \frac{Ln}{2^\ell} \leq \frac{1}{n^2},$$

where the probability is taken over the choices of R and D_i. In other words, with probability at least $1 - \frac{1}{n^2}$, all the sets $N^\ell_{i,v}$ contain at least one vertex from D_i.

Hence, with probability at least $1 - n^{-2}$, every vertex v has a vertex from the anticipatory set D_{i^*} that is no further than its ℓ^{th}-closest vertex in the actual demand set R. Having this fact at hand, we are ready to prove the following.

LEMMA 5.3. *The total cost of connecting the demand vertices in R incurred in Step **D4** is $O(\mathsf{OPT}(R) \cdot \log\log(nL))$ with probability at least $1 - \frac{1}{n^2}$.*

Proof. Let us take the multiset R, and let us denote by T_R the optimal Steiner tree on it. Take the tree T_R, build an Eulerian tour, and short-cut to get the tour $\mathbb{T}_R = \langle v_1, \ldots, v_k, v_1 \rangle$. Note that the cost of the tour is at most $2\mathsf{OPT}(R)$. Divide $\mathbb{T}_R$ into contiguous $\ell = \lceil 3 \log k \rceil$-element "segments" $\mathbb{T}_j = \langle v_{j\ell}, \ldots, v_{(j+1)\ell} \rangle$, for each index $j \in \{1, \ldots, \frac{k}{\ell}\}$: note that the length of these pieces adds up to the length of $\mathbb{T}_R$, which is at most $2\mathsf{OPT}(R)$.

Now consider the run of the above algorithm on the demand set R. For every fixed $j \in \{1, \ldots, \frac{k}{\ell}\}$ consider the *first* vertex w_j in the segment $\mathbb{T}_j$ that is given to the algorithm (and suppose this arrives at time t). Applying the trivial union bound to Lemma 5.2 gives us that with probability at least $1 - \frac{1}{n^2}$, the set $N^\ell_{i^*,w_j}$ around w_j contains a vertex from D_{i^*}. This in turn implies that the distance from this first node $w_j \in \mathbb{T}_j$ to the anticipatory set D_{i^*} is no more than the length of $\mathbb{T}_j$, since by the construction of the segments the node w_j sees at least ℓ nodes within distance equal to the length of P_j. (In fact, in the worst case this set N^ℓ_{i,w_j} of ℓ nearest nodes to w_j contains just the vertices of P_j.)

Since we initialized $S_0 = T_{i^*}$ to be the tree on this set D_{i^*}, the distance from w_j to the current set S_{t-1} is $d(w_j, S_{t-1}) \leq d(w_j, S_0) \leq d(w_j, D_{i^*})$, which by the preceding argument is no larger than the cost of $\mathbb{T}_j$. Now the cost of connecting the other vertices in R lying in the segment $\mathbb{T}_j$ is no higher than the cost of the greedy algorithm run solely on the nodes from the segment $\mathbb{T}_j$. Since the competitive ratio of the online Steiner tree is logarithmic in the number of *input nodes*, and we are looking at $\ell = O(\log nL)$ input nodes from the segment $\mathbb{T}_j$, the cost we incur is at most $O(\mathsf{Cost}(\mathbb{T}_j) \cdot \log \ell) = O(\mathsf{Cost}(\mathbb{T}_j) \cdot \log\log nL)$. Summing up over all the segments $j \in \{1, \ldots, \frac{k}{\ell}\}$ we get the total cost of connecting R to the anticipatory solution S_0 is at most $O(\mathsf{OPT}(R) \cdot \log\log nL)$ with probability at least $1 - \frac{1}{n^2}$.

Now we are ready to prove that the above algorithm has a good expected competitive ratio for the case when the sequence length k is known.

THEOREM 5.1. *Setting $L = O(\log n)$, the expected competitive ratio of the above algorithm is $O(\log\log n)$.*

Proof. Suppose either of Lemma 5.1 or Lemma 5.3 fails: this happens with probability at most $\frac{1}{L+1} + \frac{L}{n^2} \leq \frac{2}{\log n}$. In this failure case, we use the fact that we ran our algorithm in parallel with the greedy online Steiner tree algorithm, which is $O(\log k) = O(\log n)$ competitive. Hence, the contribution of these two bad cases to the EoR is only a constant.

If neither of the two lemmas fail, we see that the cost of the antipatory solution on D_i is at most $\rho_{ST}\mathsf{OPT}(R)$ by Lemma 5.1; the cost of connecting the actual demand set R having built the anticipatory solution is $O(\mathsf{OPT}(R) \cdot \log\log n)$ by Lemma 5.3. Hence the expected ratio is $O(\log\log n)$.

6 Stochastic Universal TSP

Using the techniques from the previous sections, we can obtain results for the average case of the Universal TSP

problem. To define this problem, for a set R and a permutation τ, define $\tau|_R$ to be the length of the *tour induced by τ on R*: i.e., start from any vertex in R, and keep visiting vertices in R in the order prescribed by τ until you hit the start vertex; we let $c(\tau|_R)$ be the length of the tour. In the classical Universal TSP problem, given a metric space (V, d), we are asked for a permutation τ such that $\max_{R \subseteq V} \frac{c(\tau|_R)}{c(TSP(D))}$ is as small as possible.

In the stochastic variant we study here, we are given a probability p_v of node $v \in V$ arriving (independent of the other nodes) and needing service from the salesperson.[3] The measure of goodness is:

$$\mathsf{RoE}_{TSP}(\tau) = \frac{\mathbf{E}_R[\mathsf{Cost}(\tau|_R)]}{\mathbf{E}_R[\mathsf{Cost}(\mathsf{T}SP(R))]},$$

We can obtain the following result using techniques from the previous sections:

THEOREM 6.1. (UNIVERSAL TSP RESULT) *Given a metric space with n points, and probabilities p_v for each vertex v independently demanding service. If the expected number of demands $\sum_v p_v = \Omega(1)$, there exists a permutation τ_1 such that $\mathsf{RoE}_{TSP}(\tau_1) = O(1)$.*

For lack of space, the proof of the theorem is deferred to the full version of the paper. The above theorem has been independently proved by Shmoys and Talwar (unpublished manuscript).

Acknowledgments The first- and second-named authors would like to thank the University of Rome for its generous hospitality. Many thanks to Avrim Blum, Moses Charikar, Chandra Chekuri, Rajmohan Rajaraman, Tim Roughgarden, David Shmoys, and Kunal Talwar for discussions and questions over the years on the topic of stochastic online algorithms.

References

[1] Susanne Albers. Online algorithms: a survey. *Math. Program.*, 97(1-2, Ser. B):3–26, 2003. ISMP, 2003 (Copenhagen).

[2] Susanne Albers and Stefano Leonardi. On-line algorithms. *ACM Comput. Surv.*, 31(3es):4, 1999.

[3] Noga Alon and Yossi Azar. On-line Steiner trees in the Euclidean plane. *Discrete Comput. Geom.*, 10(2):113–121, 1993.

[4] Luca Becchetti. Modelling locality: a probabilistic analysis of LRU and FWF. In *Algorithms—ESA 2004*, volume 3221 of *Lecture Notes in Comput. Sci.*, pages 98–109. Springer, Berlin, 2004.

[5] S. Ben-David and A. Borodin. A new measure for the study of on-line algorithms. *Algorithmica*, 11(1):73–91, 1994.

[6] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, New York, 1998.

[7] Allan Borodin, Sandy Irani, Prabhakar Raghavan, and Baruch Schieber. Competitive paging with locality of reference. *J. Comput. System Sci.*, 50(2):244–258, 1995. 23rd Symposium on the Theory of Computing (New Orleans, LA, 1991).

[8] Moses Charikar, Chandra Chekuri, and Martin Pál. Sampling bounds for stochastic optimization. In *Approximation, randomization and combinatorial optimization*, volume 3624 of *Lecture Notes in Comput. Sci.*, pages 257–269. Springer, Berlin, 2005.

[9] Andrew Chou, Jeremy Cooperstock, Ran El-Yaniv, Michael Klugerman, and Tom Leighton. The statistical adversary allows optimal money-making trading strategies. In *SODA '95: Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 467–476, Philadelphia, PA, USA, 1995. Society for Industrial and Applied Mathematics.

[10] F. Eisenbrand, F. Grandoni, T. Rothvoß, and G. Schäfer. A tighter analysis of random sampling for connected facility location. Submitted, 2007.

[11] Amos Fiat and Anna R. Karlin. Randomized and multipointer paging with locality of reference. In *STOC '95: Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 626–634, New York, NY, USA, 1995. ACM Press.

[12] Amos Fiat and Ziv Rosen. Experimental studies of access graph based heuristics: beating the LRU standard? In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (New Orleans, LA, 1997)*, pages 63–72, New York, 1997. ACM.

[13] Amos Fiat and Gerhard J. Woeginger, editors. *Online algorithms*, volume 1442 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998. The state of the art, Papers from the Workshop on the Competitive Analysis of On-line Algorithms held in Schloss Dagstuhl, June 1996.

[14] Lisa Fleischer, Jochen Könemann, Stefano Leonardi, and Guido Schäfer. Simple cost sharing schemes for multicommodity rent-or-buy and stochastic steiner tree. In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 663–670, New York, NY, USA, 2006. ACM Press.

[15] Dimitris Fotakis. On the competitive ratio for online facility location. In *Automata, languages and programming*, volume 2719 of *Lecture Notes in Comput. Sci.*, pages 637–652. Springer, Berlin, 2003.

[16] Sudipto Guha, Adam Meyerson, and Kamesh Munagala. Hierarchical placement and network design prob-

[3] Note the slightly different model than in the previous sections: we could have assumed that the set R is obtained by taking some ℓ samples from a distribution π over vertices such that $\sum_v \pi_v = 1$, instead of drawing one set R where each vertex independently arrives with probability p_v. We feel that the p_v model is better for the Universal TSP prolem, though many of the results continue to hold in π_v case.

lems. In *Proceedings of the 41th Symposium on the Foundations of Computer Science (FOCS)*, pages 603–612, 2000.

[17] Anupam Gupta, Amit Kumar, Martin Pál, and Tim Roughgarden. Approximations via cost-sharing. In *Proceedings of the 44th Symposium on the Foundations of Computer Science (FOCS)*, pages 606–615, 2003.

[18] Anupam Gupta, Amit Kumar, and Tim Roughgarden. Simpler and better approximation algorithms for network design. In *Proceedings of the 35th ACM Symposium on the Theory of Computing (STOC)*, pages 365–372, 2003.

[19] Anupam Gupta, Martin Pál, R. Ravi, and Amitabh Sinha. Boosted sampling: Approximation algorithms for stochastic optimization problems. In *Proceedings of the 36th ACM Symposium on the Theory of Computing (STOC)*, pages 417–426, 2004.

[20] Makoto Imase and Bernard M. Waxman. Dynamic Steiner tree problem. *SIAM J. Discrete Math.*, 4(3):369–384, 1991.

[21] Nicole Immorlica, David Karger, Maria Minkoff, and Vahab Mirrokni. On the costs and benefits of procrastination: Approximation algorithms for stochastic combinatorial optimization problems. In *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 684–693, 2004.

[22] Sandy Irani and Anna R. Karlin. On online computation. In Dorit Hochbaum, editor, *Approximation Algorithms for NP Hard Problems*. PWS publishing Co, 1996.

[23] Sandy Irani, Anna R. Karlin, and Steven Phillips. Strongly competitive algorithms for paging with locality of reference. *SIAM J. Comput.*, 25(3):477–497, 1996.

[24] Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. *J. ACM*, 47(4):617–643, 2000.

[25] David R. Karger and Maria Minkoff. Building Steiner trees with incomplete global knowledge. In *Proceedings of the 41th Symposium on the Foundations of Computer Science (FOCS)*, pages 613–623, 2000.

[26] Anna R. Karlin, Mark S. Manasse, Larry Rudolph, and Daniel D. Sleator. Competitive snoopy caching. *Algorithmica*, 3(1):79–119, 1988.

[27] Anna R. Karlin, Steven J. Phillips, and Prabhakar Raghavan. Markov paging. *SIAM J. Comput.*, 30(3):906–922, 2000.

[28] Samir Khuller, Balaji Raghavachari, and Neal E. Young. Balancing minimum spanning and shortest path trees. *Algorithmica*, 14(4):305–322, 1995.

[29] Elias Koutsoupias and Christos H. Papadimitriou. Beyond competitive analysis. *SIAM J. Comput.*, 30(1):300–317, 2000.

[30] Mark S. Manasse, Lyle A. McGeoch, and Daniel D. Sleator. Competitive algorithms for server problems. *J. Algorithms*, 11(2):208–230, 1990.

[31] Adam Meyerson. Online facility location. In *42nd IEEE Symposium on Foundations of Computer Science (Las Vegas, NV, 2001)*, pages 426–431. IEEE Computer Soc., Los Alamitos, CA, 2001.

[32] Adam Meyerson, Kamesh Munagala, and Serge Plotkin. Designing networks incrementally. In *Proceedings of the 42nd Symposium on the Foundations of Computer Science (FOCS)*, pages 406–415, 2001.

[33] Konstantinos Panagiotou and Alexander Souza. On adequate performance measures for paging. In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 487–496, New York, NY, USA, 2006. ACM Press.

[34] C. A. Phillips, C. Stein, E. Torng, and J. Wein. Optimal time-critical scheduling via resource augmentation. *Algorithmica*, 32(2):163–200, 2002.

[35] Prabhakar Raghavan. A statistical adversary for online algorithms. In *Online Algorithms*, volume 53 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 79–83. Amer. Math. Soc., Providence, RI, 1991.

[36] R. Ravi and Amitabh Sinha. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. In *Proceedings of the 10th Integer Programming and Combinatorial Optimization Conference (IPCO)*, pages 101–115, 2004.

[37] Gabriel Robins and Alexander Zelikovsky. Tighter bounds for graph Steiner tree approximation. *SIAM J. Discrete Math.*, 19(1):122–134, 2005.

[38] Daniel J. Rosenkrantz, Richard E. Stearns, and Philip M. Lewis, II. An analysis of several heuristics for the traveling salesman problem. *SIAM J. Comput.*, 6(3):563–581, 1977.

[39] Mark Scharbrodt, Thomas Schickinger, and Angelika Steger. A new average case analysis for completion time scheduling. *J. ACM*, 53(1):121–146, 2006.

[40] David B. Shmoys and Chaitanya Swamy. An approximation scheme for stochastic linear programming and its application to stochastic integer programs. *J. ACM*, 53(6):978–1012, 2006.

[41] Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Comm. ACM*, 28(2):202–208, 1985.

[42] Alexander Souza and Angelika Steger. The expected competitive ratio for weighted completion time scheduling. *Theory Comput. Syst.*, 39(1):121–136, 2006.

[43] Neal Young. On-line caching as cache size varies. In *Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco, CA, 1991)*, pages 241–250, New York, 1991. ACM.

[44] Neal E. Young. On-line paging against adversarially biased random inputs. *J. Algorithms*, 37(1):218–235, 2000. Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco, CA, 1998).