Return of the Boss Problem: Competing Online Against a Non-Adaptive Adversary

Magnús M. Halldórsson[1] and Hadas Shachnai[2]

[1] School of Computer Science, Reykjavik University, 101 Reykjavik, Iceland.
mmh@ru.is.
[2] Department of Computer Science, The Technion, Haifa 32000, Israel.
hadas@cs.technion.ac.il.

Abstract. We follow the travails of Enzo the baker, Orsino the oven man, and Beppe the planner. Their situation have a common theme: They know the input, in the form of a sequence of items, and they are not computationally constrained. Their issue is that they don't know in advance the time of reckoning, i.e. when their boss might show up, when they will be measured in terms of their progress on the prefix of the input sequence seen so far. Their goal is therefore to find a particular solution whose size on any prefix of the known input sequence is within best possible performance guarantees.

1 Doing OK When the Boss Shows up : Prefix Optimization

1.1 Enzo's Order Signups: Prefix Interval Selection

Enzo groaned with his arms curled over his head: "I'm in a fix – big-time. He's going to catch me at the worst possible moment."

"Who is?", I inquire.

"My boss could show up at any moment, and if I haven't signed up for my share, I can kiss my confetteria dream goodbye."

"So, why don't you? You can do it. You've got the brawn to handle any set of orders, and you've got the brains to figure out what is the maximum set that can be handled by a single person. What's holding you back?"

He slumps still lower in the seat. "It's not a matter of processing or computational power. Yes, I can figure it all out. I even know all the orders in advance; we always get the same set of orders on Fridays. So, of course, I could just find an optimal solution. But that's of no use."

"Now you really got me. You know everything and you can do anything, what could possibly be the problem?"

"Yeah, it's kind of funny. Look, let me explain the whole setup. Orders to the bakery arrive in a sequence. Each order has a given pickup time, and since people expect it to straight from the oven when they pick it up, it really means that the time for making it is fixed. "

Instance spec: Given is a sequence $\mathcal{I}_n = \langle I_1, I_2, \ldots, I_n \rangle$, where each I_i is an interval in $\Re$.

"As soon as an order arrives, I have to either sign up for it or assign it to somebody else. I can't work on two orders at the same time; they need my total concentration. We've got plenty of other guys that can handle those that I don't do. "

Solution spec: Produce a subsequence $\mathcal{I}'$ of intervals from $\mathcal{I}_n$ such that no pair of intervals I_a, I_b in $\mathcal{I}'$ overlap, i.e. $I_a \cap I_b = \emptyset$.
More generally, we seek an *independent set* in a given graph G; in the above situation G is an interval graph.

"Actually, I can decide the whole thing in advance, because I do know all the orders that arrive — we always get the same set of orders on that day of the week. But, that stupid boss doesn't know it, and he won't believe it."

"He insists on checking on me at any time to 'size me up for the big job', as he calls it. And, the frustrating thing is that when he does, all that matters to him is how many orders I've then signed up for."

This sort of makes sense to me. "So, you want to maintain always a good efficiency index. I mean, that what you've gotten, when the boss shows up, should be a large fraction of what could possibly be gotten if you'd knew when he shows up. Right?"

"Yeah, that's it. Or, maybe it's more of a sloth index."

"There's a catch here, though: you can't ever hope to get a good ratio! You see, suppose the first order was so long that it took all of your available time. Well, if you take it, you can't take anything else, and you're doomed. But if you don't take it, then if the boss shows up then, you've got an infinitely poor sloth index!"

"I knew it, I'm doomed" he whined.

"No, relax. You just need to modify your expectations a bit. You do try to minimize the ratio, but you count an empty solution as of size one. Which is fair, what diff does a single order make?"

Performance evaluation: For each prefix $\mathcal{I}_p = \langle I_1, I_2, \ldots, I_p \rangle$, the *performance* on the prefix is the quantity

$$\rho_p = \rho_p(\mathcal{I}') = \frac{\alpha(\mathcal{I}_p)}{|\mathcal{I}' \cap \mathcal{I}_p| + 1},$$

where $\alpha(\mathcal{I}_p)$ is the maximum set of disjoint intervals in $\mathcal{I}_p$. The objective is to minimize $\rho = \max_{p=1}^{n} \rho_p$, the worst performance on any prefix.

"Yeah, ok. But it's still a Catch-22 situation: no matter what I do, I'm doomed. If I start grabbing orders as soon as they arrive, I won't be able to take on so many of the later orders, and he'll take me to task at the end of the day. But if I try to wait until the best set of orders starts showing up, he'll think I'm a lazy SOB that can't get started in the morning. There's no point even

trying to explain to him how thinking ahead could help. It's so Kafkaesque it's not even funny," he grumbles and shakes his head.

"Right, well, let's think constructively, and at least try to do the best we can. There is, by the way, a possibility that you can do better if you flip coins. "

"I'm not interested in some kind of average case."

"This is different. It means that no matter what how tricky your boss may be and even if he knows your solution strategy, you will always achieve some performance guarantee at the time he stops by. However, the guarantee is in expectation over the random coin flips, and not worst case..."

> **Randomized performance:** In the randomized case, a solution is a probability distribution π over the independent sets of the input graph G. The expected solution size $E_\pi[\mathcal{I}_p]$ on prefix $\mathcal{I}_p$ is the weighted sum $E_\pi[\mathcal{I}_p] = \sum_{I'} \Pr_\pi[I'] \cdot |I' \cap \mathcal{I}_p|$, where I' ranges over all independent sets in G. The performance ratio is then $\rho = \max_{p=1}^{n} \frac{\alpha(\mathcal{I}_p)}{\mathbb{E}_\pi[\mathcal{I}_p]}$.

1.2 Other prefix problems

"Right. BTW, some of the other guys are in similar situations, although they all have different types of tasks. For instance, Orsino the oven guy has to lay out the goods to be baked onto the baking plates. Not everything can go onto the same plate; it's not just the temperature, but, for instance, the slushy items can't go with the dry ones, square items will mess up the round ones, and the fragrance of certain items will affect other items, and so on. So, he needs to lay out all the goods onto the plates, and to do so as soon as they've been prepared."

"The problem is that if he uses more plates than necessary for the items ready at that time, the boss will get angry."

> **Prefix Coloring:** Given an ordered graph G, i.e. with an ordered vertex set $V = \langle v_1, v_2, \ldots, v_n \rangle$, find a coloring C of G such that $\rho = \max_{p=1}^{n} \frac{C(G_p)}{\chi(G_p)}$ is minimized, where G_p is the graph induced by $\langle v_1, v_2, \ldots, v_p \rangle$, $C(G_p)$ is the number of colors that C uses on G_p and $\chi(G_p)$ is the chromatic number of G_p.

Something about this rang a bell with me. "Actually, this really reminds me of this chap Beppe doing urban planning for the city. They had these new servers being set up all the time, each having links to some of the earlier ones. They had to have guards on one side of each link to protect against unauthorized entry. As soon as a server was set up, they had to decide there and then whether to make it a guard, because the cost of converting an older server into a guard was prohibitive. Of course, they could have made every server a guard, but that not only was time consuming but looked spectacularly stupid."

"Every now and then, some wise-crack newspaper reporter would look at the guard installations in the city and try to score point by discovering 'waste in the system', that much fewer were needed for the situation at that time. Of course, there never was any point in try to counteract by showing that this would be

needed in the future; by the time such corrections came to light, nobody was interested in the story any more."

Prefix Vertex Cover: Given an ordered graph G, find a vertex cover C of G such that $\rho = \max_{p=1}^{n} \frac{C(G_p)}{VC(G_p)}$ is minimized, $C(G_p) = C \cap V_p$ is the number of nodes from C in the prefix set V_p, and $VC(G_p)$ is the vertex cover number of G_p.

1.3 Related work

Enzo now stands up and looks me straight in the eye: "What should I do? You're the math guy, can you solve this?"

I instinctively curl my shoulders, "I'm more of a CS guy, actually. In any case, your problem doesn't really fall into any of our usual categories. I mean, it's not really online, since you know the whole input in advance. It's also not offline, since we don't know the length of the prefix where we will be measured. It's also not really a computational issue, since you're not computationally bounded. It is a question about robustness: performing well on all of a set of sub-instances; here the sub-instances are the prefixes."

Hassin and Rubinstein [12] gave a weighted matching algorithm that gave a $\sqrt{2}$-performance guarantee for the total weight of the p-heaviest edges, for all p.

There are various works where robustness is with respect to a class of objective function. One prominent examples are those of scheduling under any L_p-norm [3, 2]. Goel and Meyerson [11] gave a general scheme for minimizing convex cost functions, such as in load balancing problems. Robust colorings of intervals were considered in [9], where throughput in any prefix of the coloring classes went into the objective function.

Prefix optimization can be viewed as online algorithm with complete knowledge of the future, or competing against a *non-adaptive* adversary. Several lower bound constructions for online computation are actually lower bounds on prefix computation; most prominent cases are for the classical machine scheduling problems [8, 4, 1]. Prefix optimization corresponds to the extreme case of lookahead, and thus can perhaps shed some light on properties of online computation.

As of yet, only few papers have explicitly addressed prefix optimization. Faigle, Kern and Turán [8] considered online algorithms and lower bounds for various online problems, giving a number of lower bounds in the prefix style. They posed the question of a constant factor approximation for the Prefix Coloring problem. Dani and Hayes [6] considered online algorithms for a geometric optimization problem and explicitly compared the competitive ratios possible against adaptive and non-adaptive adversaries.

"So, it relates to this online algorithms racket, but basically nobody has done exactly this. I sure hope you can at least come up with some ideas, man."

1.4 Our Results

We give nearly tight results for the best possible performance ratios for prefix versions of several fundamental optimization problems (in particular, those of Enzo, Beppe and Orsino).

We first consider the PREFIX IS problem and give an algorithm whose performance ratio on interval graphs is $O(\log \alpha)$, where $\alpha = \alpha(G)$ is the independence number of the graph. We derive a matching lower bound for any (randomized or deterministic) algorithm for the problem on interval graphs. We further give a randomized algorithm that achieves this ratio on general graphs. The algorithm is shown to achieve a logarithmic performance ratio for a wide class of maximum subset selection problems in the prefix model, including maximum clique, 0/1-knapsack, maximum coverage, and maximum k-colorable subgraph. For all of these problems the performance ratio is $O(\log(\alpha(I))$, where $\alpha(I)$ is the value of an optimal solution for the complete input sequence I.

For the PREFIX VERTEX COVER problem we show that any algorithm has a performance ratio of $\Omega(\sqrt{n})$, where n is the size of G. We give a deterministic algorithm that achieves this bound.

Finally, for PREFIX COLORING we give an algorithm whose performance ratio is 4. We further show that no algorithm achieves a ratio better than 2.

2 Enzo's Effectiveness Issue: Prefix Independent Set

"So, my friend, here's what I can tell you about your problem."
Let $\alpha = \alpha(G)$ be the independence number of the given graph G. We first present an approximation algorithm for PREFIX IS in interval graphs and then give a simple algorithm for general graphs.

Suppose that G is an interval graph. Consider the following algorithm, A_{int}, which uses as additional input parameter some $t \geq 1$. For the ordered sequence of intervals in G, let G_i be the shortest prefix satisfying $\alpha(G_i) \geq \alpha^{i/t}$, for $1 \leq i \leq t$ Algorithm A_{int} proceeds in the following steps.

1. Find in G_1 an IS of size $\alpha(G_1)$.
2. For $2 \leq i \leq t$ in sequence, find for G_i an IS, I_i, of size at least $\alpha(G_i)/t$, such that I_i conflicts with at most a fraction of $1/t$ of the vertices in each I_r, $1 \leq r \leq i-1$, and delete from I_r the vertices conflicting with I_i.
3. Output the solution $I = \cup_{i=1}^{t} I_i$.

"Was there a message to this? I thought you'd come up with something simpler. Why this sequence of solutions that you keep chipping off?" Enzo quizzed.
"When you put it formally it starts to look more complicated than it really is. The point is that we have to come up with some solution early on, even if it would be better for the long haul to just wait. This early piece of the input is G_1. We try to find a small part of the solution that doesn't mess up the rest of the input sequence by too much."

"We then need to repeat this on several prefixes, picking some portion of the available solution; if it requires smashing some of the family treasures, so be it, but make sure it's only a small fraction, 'minor collateral damage' ".

In analyzing A_{int}, we first need to show that a 'good' independent set can be found in each iteration, as specified in Step 2.

Lemma 1. *Let t and the interval graph G_i be defined as above, and let k be a number between 1 and t. Let I be an independent set of G_{k-1}, partitioned into sets $I_1, I_2, \ldots, I_{k-1}$ where $I_r \subseteq V(G_r)$, for $1 \leq r \leq k-1$. Then, there exists an IS I_k in G_k of size at least $(\alpha(G_k) - 2\alpha(G_{k-1}))/t$ which conflicts with at most $1/t$-fraction of the intervals in each I_r, $1 \leq r < k$.*

Proof. We say that an interval a flanks an interval b if a overlaps one of the endpoints of b, i.e., a and b overlap but a is not contained in b. Let J be a maximum IS of G_k, and let J' be the set of intervals in J that do not flank any interval in I. Observe that $|J'| \geq |J| - 2|I| \geq \alpha(G_k) - 2\alpha(G_{k-1})$. Note that each interval in J' intersects at most one interval in I.

For sets A and B of intervals, let $N_B[A] = \{b \in B | \exists a \in A, a \cap b \neq \emptyset\}$ be the set of intervals in A that overlap some interval in B. For each interval a in I let the weight of a be the number of intervals in J' intersecting a, or $wt(a) = |\{b \in J' : a \cap b \neq \emptyset\}| = |N_{J'}[\{a\}]|$. For each $j = 1, \ldots, k-1$ in parallel, find a maximum weight subset Q_j in I_j of size $|I_j|/t$, and let $Q = \cup_{j=1}^{k-1} Q_j$. Finally, let $I_k = N_{J'}[Q]$ be the claimed set of intervals from G_k.

By construction, I_k is an IS and is of size at least $\sum_{a \in I} wt(a)/t = |J'|/t \geq (\alpha(G_k) - 2\alpha(G_{k-1}))/t$. Also, by definition, I_k conflicts with a set of size $|I_r|/t$ from each set I_r, $r = 1, \ldots, k-1$. Hence, the claim. ∎

Theorem 1. *A_{int} yields a performance ratio of $O(\log \alpha)$ for* PREFIX IS *on interval graphs.*

Proof. Let $t = \log_3 \alpha$. Then, $\alpha(G_k) = 3\alpha(G_{k-1})$, for each $1 < k \leq t$. We distinguish between two cases.

(i) Suppose that the prefix $\hat{G}$ presented to A_{int} is shorter than G_1, then A_{int} outputs an empty set, while OPT has an IS of size at most $\alpha^{1/t}$. We get that

$$\frac{OPT(\hat{G})}{A_{int}(\hat{G}) + 1} \leq \alpha^{1/t} \leq 3 \ .$$

(ii) Otherwise, let k be the maximum value such that $G_k \subseteq \hat{G}$. By the construction in Step 2, each IS reduces the size of any preceding IS at most by factor $1/t$. Hence, by Lemma 1 we get that

$$|I_k| \geq \frac{\alpha(G_k) - 2\alpha(G_{k-1})}{t} \left(1 - \frac{1}{t}\right)^{t-k} \geq \frac{\alpha^{k/t}}{3t} \left(1 - \frac{1}{t}\right)^{t-k} \geq \frac{\alpha^{k/t}}{3t} \cdot e^{-1} \ .$$

Since $OPT(\hat{G}) \leq \alpha(G_{k+1}) = \alpha^{(k+1)/t}$ and $A_{int}(\hat{G}) \geq |I_k|$, we get a ratio of $O(t\alpha^{1/t}) = O(\log \alpha)$. ∎

"This may not be exactly what you were looking for, Enzo, but it is quite interesting to some of us. Particularly the fact that the ratio is in terms of the optimal solution size, α, rather than some general property of the input, like the number of items, n."

2.1 Matching Lower Bound

"You know, what you've come up with is alright. But, maybe if we think a bit harder, we could get solutions that are always just a few percent off the best possible. Can't you get one of those really smart guys, like Luca or Pino, to help you out? Or better yet, one of those hot-shot women," Enzo grinned.
"Easy, easy. There're limits to everything. In fact, what we outlined above is essentially the best possible.

Theorem 2. *Any algorithm (even randomized) for* PREFIX IS *in interval graphs has performance ratio of* $\Omega(\log \alpha)$.

Proof. Let α be a number, and let $k = \log \alpha + 1$. Consider the graph $G = (V, E)$ which consists of k subsets of vertices, $V_1, \ldots, V_k$. The subset V_i consists of 2^{i-1} vertices numbered $\{v_{i,1}, \ldots, v_{i,2^{i-1}}\}$. The set of edges in G is given by $E = \{(v_{i,h}, v_{j,\ell}) | 1 \leq i < j \leq k, 1 \leq h \leq 2^{i-1}, 2^{j-i}(h-1) + 1 \leq \ell \leq 2^{j-i}h\}$. In the ordered sequence representing G, all the vertices in V_i precede those in V_{i+1}, for $i = 1, 2, \ldots, k-1$. The graph G can be represented as an interval graph where a vertex $v_{i,h}$ corresponds to the interval $[(h-1)2^{k-i}, h2^{k-i})$; see Fig. 1.

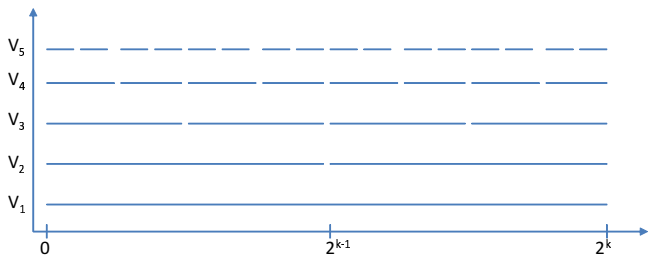

Fig. 1. Interval graph that gives a $\Omega(\log \alpha)$ lower bound.

In view of Yao's lemma we give a probability distribution over the prefixes of G and upper bound the expected performance of any deterministic algorithm over this distribution. Let $U_i = \cup_{j=1}^{i} V_i$. With probability $1/(2^i(1-2^{-k}))$ the prefix is $\hat{G} = G[U_i]$, for $i = 1, 2, \ldots, k$.

First observe that

$$\mathbb{E}[OPT(\hat{G})] = \sum_{i=1}^{k} \Pr[\hat{G} = G[U_i]] \cdot |U_i| = \sum_{i=1}^{k} \frac{1}{2^i} \cdot 2^{i-1} \geq \frac{\log \alpha}{2} \ .$$

We now show that $\mathbb{E}[A(\hat{G})] \leq 1$, which yields the theorem.

Consider any given deterministic algorithm A that produces an independent set I_A on G. For any $v \in V_i$, let $N_{IS}(v)$ be the set of neighbors of v in $V \setminus U_i$ that are selected by A into I_A, and let $\hat{N}_{IS}(v) = N_{IS}(v) \cup \{v\}$. We denote by $A(v) = |\hat{N}_{IS}(v)|$ the increase in $|I_A|$ due to the selection of vertices in $\hat{N}_{IS}(v)$. We say that vertex $v \in V_i$ is *possible* for A if there are no edges between v and any of the vertices selected by A in U_{i-1}.

Claim 1. Let $v \in V_i$ be a possible vertex for A, $1 \leq i \leq k$. Then, if v is not selected for I_A, we have that $\mathbb{E}[A(v)|v \textit{ not selected}] \leq \mathbb{E}[A(v)|v \textit{ is selected}] = 1$.

Proof. The proof is by backward induction. For the base case we take $i = k$. Clearly, if A does not select the vertex v then $A(v) = 0$, since V_k is the last subset of vertices, and the claim holds. Now, assume that the claim holds for all vertices v in V_i, then we show that it holds also for the vertices in V_{i-1}. It is easy to show (details omitted) that $\Pr[V_{i+1} \subseteq \hat{G} | V_i \subseteq \hat{G}] = 1/2$, for $i = 1, 2, \ldots, k-1$. Observe that G is constructed so that if a vertex $u \in V_i$ is adjacent to a vertex $v \in V_j$ and v is adjacent to vertex $w \in V_k$, for $i < j < k$, then u is also adjacent to v_k; namely, the presentation ordering of the graph is transitive. If a vertex $v_{i-1,h}$ is not selected, then A can select its neighbors in V_i: $v_{i,2h-1}, v_{i,2h}$, and we get that

$$A(v_{i-1,h}) \leq \frac{1}{2}(\mathbb{E}[A(v_i, 2h-1)] + \mathbb{E}[A(v_i, 2h)]) \ .$$

From the induction hypothesis we have $\mathbb{E}[A(v_i, 2h-1)] = \mathbb{E}[A(v_i, 2h)] \leq 1$. ■

To complete the proof of the theorem we note that, if A selects for the solution the single vertex in V_1 then $A(\hat{G}) = 1$; else, by Claim 1, we get that $A(\hat{G}) \leq 1$. ■

2.2 Prefix IS in General graphs

> "Allow me to entertain ourselves by generalizing the problem you posed to independent sets in general graphs.

For general graphs, we can argue tight bounds on performance guarantees. Let G_1 be the shortest prefix of the input graph G for which $\alpha(G_1) = \alpha_1$ where $\alpha_1 = \lceil \sqrt{\alpha} \rceil$. The algorithm A_{gen} finds an independent set of size α_1 in G_1 and simply outputs this set.

Theorem 3. *Algorithm A_{gen} yields a performance ratio of at most $\sqrt{\alpha}$ for* PREFIX IS.

Proof. We distinguish between two types of prefixes given to the algorithm. Suppose the prefix is strictly shorter than G_1. Then an optimal algorithm yields an IS of size at most $\alpha_1 - 1$, for a performance ratio of at most $\alpha_1 - 1 \leq \sqrt{\alpha}$. On the other hand, if G_1 is contained in the prefix $\hat{G}$ then the approximation ratio is at most $\alpha/\alpha_1 \leq \sqrt{\alpha}$. ■

We can easily argue a matching lower bound.

Theorem 4. *The performance ratio of any deterministic algorithm for* PREFIX IS *is at least* $\lfloor\sqrt{\alpha}\rfloor$. *It is also* $\Omega(\sqrt{n})$.

Proof. Let $N = \lfloor\sqrt{n}\rfloor$. Consider the following complete bipartite graph $G = (U, V, E)$. The vertices U are $1, 2, \ldots, N$, while V has $N+1, \ldots, n$. The ordering of the graph is by vertex number.

A deterministic algorithm A run on G can pick either vertices from U or from V. If it picks vertices from U, then on the prefix $B = G$ the optimal solution is of size $n - N$, while the algorithm solution is of size at most N, giving a ratio of at least $n/N - 1 \geq \sqrt{n} - 1$. The ratio is also at least $\alpha/N \geq \alpha/\lfloor\sqrt{\alpha}\rfloor \geq \sqrt{\alpha}$. On the other hand, if it picks vertices from V, then on the prefix $B' = G[U]$, the subgraph of G induced by U, $A(B') = 0$ while $\alpha(G') = N$, for a ratio of $N = \lfloor\sqrt{n}\rfloor \geq \lfloor\sqrt{\alpha}\rfloor$. Hence, the performance ratio of A on G is at least $N = \sqrt{n} \geq \sqrt{\alpha}$. ∎

> "You haven't said anything here about coin flips. Can they help?", Enzo inquired.
> "For your problem of intervals, they don't. But, for the general problem, on general graphs, a simple randomized approach does improve the situation dramatically. In fact, it's kind of neat to phrase it in terms of still more general problem framework."

In the following we consider a wide class of maximization subset selection problems in the prefix model.

Definition 1. *A problem* Π *is* hereditary *if for any input* I *of* Π, *if* $I' \subseteq I$ *is a feasible solution for* Π, *then any subset* $I'' \subseteq I'$ *is also a feasible solution.*

Note that many subset selection problems are hereditary. This includes maximum independent set, maximum clique, 0/1-knapsack, maximum coverage, and maximum k-colorable subgraph, among others.

Given an input I for a subset selection problem Π, let $\alpha = \alpha(I)$ be the size of an optimal solution for I.

Theorem 5. *Let* Π *be a hereditary maximum subset selection problem. Then, there is an algorithm for* Π *with a performance ratio of* $O(\log(\alpha))$ *for the* PREFIX$-\Pi$ *problem.*

Proof. Consider the following algorithm. Let P be an input for Π with optimal value α and let $k = \lceil \lg \alpha \rceil$. Define prefixes $P_1, \ldots, P_k$ of P where $P_k = P$ and for $i = 1, \ldots, k-1$, P_i is the shortest prefix with $\alpha(P_i) \geq 2^{i-1}$. The algorithm now selects one of the prefixes P_i uniformly at random, each with probability $1/k$.

Let j be such that the prefix $\hat{P}$ presented satisfies $P_j \subseteq \hat{P} \subseteq P_{j+1}$. Since $\hat{P}$ is non-empty, it must contain the unit prefix P_1. Then, the value of the optimal solution is at most twice the value of the solution for P_j. With probability $1/k$ our algorithm obtains an optimal solution on P_j. Hence, the expected size of the solution found by the algorithm is at least $1/(2k)$ fraction of optimal. ∎

Corollary 1. *There is a randomized* $O(\log \alpha))$*-approximation algorithm for* PREFIX IS.

3 Beppe's Guarding Business: Prefix Vertex Cover

"Beppe situation must be easier. I've heard this vertex cover problem, as you call it, is much easier" Enzo remarked philosophically.
"It is in many respects easier. For instance, it's easy when the optimal solution is small (or Fixed Parameter Tractable, as we say), which the general independent set problem is not. And it's easily approximable in polynomial time within factor 2, while the IS problem is notoriously hard."
"Yeah, you should know. You're the one who keeps doing IS in one way or another, in spite of these pathetic approximations."
"Hey, no need to get personal here, buddy. We all do what we can. But, back to Beppe's problem, it turns out to be actually harder than your prefix IS problem! Randomization won't help him at all."

Theorem 6. *Any algorithm for* Prefix Vertex Cover *has performance ratio of at least* $\sqrt{n}$.

Proof. We use the complete bipartite graph $G = (U, V, E)$ from Theorem 4, where U has nodes $1, 2, \ldots N = \lfloor\sqrt{n}\rfloor$ and V nodes $N+1, N+2, \ldots, n$.

If any vertex in U is missing in a cover, then we need to select all the vertices in V. Thus, the only minimal vertex covers for B are U and V. Any randomized algorithm R_{vc} selects one of these solutions with probability at least $1/2$. If $C = U$ with probability at least $1/2$, then for a prefix $\hat{B}$ that consists of all the vertices in U we get that $\frac{R_{vc}(\hat{B})}{OPT(\hat{B})+1} = \Omega(\sqrt{n})$, since $E = \emptyset$. On the other hand, if V is selected for the solution with probability at least $1/2$, then for the prefix $\hat{B} = B$, we have that $\frac{R_{vc}(\hat{B})}{OPT(\hat{B})+1} \geq \frac{n-\sqrt{n}}{2(\sqrt{n}+1)} = \Omega(\sqrt{n})$, since an optimal solution is $C = U$. ∎

We now show that a matching upper bound is obtained by a deterministic algorithm. Let $G_1 = (V_1, E_1)$ be the prefix graph for which the minimum vertex cover is of size at least $\sqrt{n}$ for the first time. A_{vc} finds a minimum vertex cover $S \subseteq V_1$ for G_1 and outputs $S \cup (V \setminus V_1)$.

Theorem 7. *Algorithm A_{vc} yields a ratio of $\sqrt{n}$ to the optimal for* Prefix Vertex Cover.

Proof. We note that if $\hat{G} \subseteq G_1$ then A_{vc} outputs $S' \subseteq S$ where $|S'| \leq \sqrt{n}$, while an optimal algorithm may output an empty set. If $G_1 \subseteq \hat{G}$ then $|S \cup (\hat{V} \setminus V_1)| \leq n$, while $OPT(\hat{G}) \geq OPT(G_1) \geq \sqrt{n}$. The claim follows. ∎

4 Orsino's Oven Schedule: Prefix Graph Coloring

"What's your take then on Orso's situation?" Enzo asked me the following evening. Is he dug as deep as Beppe?"

"Actually, this looks much more promising" I replied. "We can use a standard trick of the trade called doubling to come out pretty well."
"Play double-or-nothing until we finally win" he suggested hopefully.
"If you like. We use the nice property of geometric sums, i.e. $1+2+4+8+...+2^k$ that they add up to not too much, or only twice the last term."
"Ah, so we first handle the first node, then the next two, then the next four, and so on?"
"You're catching on, Enzo, but we actually need to use it slightly differently. We first find the initial set of order that can be laid on a single plate. Then, the next prefix that can be laid onto two plates. Third, the sequence of the following orders for which four plates suffice. And so on, doubling the number of plates in each step."
"Ok! So we use new set of plates for each of these, uh, groups."
"Exactly. And why is that ok?"
"Because they add up to not too much! Maybe I should try this CS business; you think I might have a shot at a Gödel award?"

For $k = 1, 2, \ldots$, let t_k be the largest value such that $\alpha(G_{t_k}) \leq 2^k$, and let $t_0 = 0$ for convenience. The algorithm A simply colors each set $V_{t_k} \setminus V_{t_{k-1}}$ with 2^k fresh colors.

Theorem 8. *The algorithm A yields a performance guarantee of 4 for* PREFIX COLORING.

Proof. Let p be a number, $1 \leq p \leq n$, and let k be the smallest number such that $2^k \geq \alpha(G_p)$. So, $\alpha(G_p) \geq 2^{k-1} + 1$. Observe that

$$A(G_p) \leq \sum_{i=0}^{k} 2^i = 2^{k+1} - 1 < 4 \cdot 2^{k-1} < 4\alpha(G_p) \ .$$

Since this holds for all p simultaneously, the theorem follows. ∎

"Can we also do better here in the randomized case?"
"Actually, yes, a little."

Let β be a uniformly random value from $[0, 1]$, and let $a_0, a_1, a_2, \ldots$ be the sequence given by $a_i = \lceil \beta e^i \rceil$. Let t_k be the largest value such that $\alpha(G_{t_k}) \leq a_k$. Modify the algorithm A to use a_k colors on each set $V_{t_k} \setminus V_{t_{k-1}}$.

Theorem 9. *The modified algorithm A has randomized performance guarantee of e for Prefix Graph Coloring.*

The proof is similar to the arguments used for certain coloring problems with demands [7, 10].

"So, is that the best we can do."
"It's the best that I can come up with, but it's also provably close to the best possible."

Proposition 1. *There is no algorithm with performance guarantee less than 2 for Prefix Graph Coloring.*

"But, I'll leave that for you to figure out, hot shot :-)"

5 Epilogue

"This is all cute'n stuff, but it ain't mean nothin out there in the field, does it?"

"It's always hard to say where theoretical results kick in. It does though tell us something about how robust we can make computation. It doesn't have to involve your boss, of course; it could be any unpredictable event like the electricity going off. You could think of this as a sort of defensive problem-solving. In these days of global security threats, ain't that what we all have to concern ourselves with?"

"Yeah, or you might have your little island economy suddenly going off the cliff", Enzo chuckles. "Good luck in cashing in on these ideas..."

References

1. S. Albers. Better bounds for online scheduling. In *STOC*, pp.130–139, 1997.
2. Y. Azar and A. Epstein. Convex programming for scheduling unrelated parallel machines. In *STOC*, 2005.
3. Y. Azar, L. Epstein, Y. Richter, and G. J. Woeginger. All-norm approximation algorithms. *J. Algorithms*, 52:120–133, 2004.
4. Y. Bartal, H. J. Karloff, Y. Rabani. A better lower bound for on-line scheduling. *Inf. Process. Lett.* 50(3): 113-116, 1994.
5. A. Borodin and R. El-Yaniv. *Online computation and competitive analysis.* Cambridge University Press, 1998.
6. V. Dani and T. P. Hayes. Robbing the bandit: Less regret in online geometric optimization against an adaptive adversary. In *SODA*, 2006.
7. L. Epstein and A. Levin. On the Max Coloring Problem. In *WAOA*, 142–155, 2007.
8. U. Faigle, W. Kern, and G. Turán. On the performance of on-line algorithms for partition problems. *Acta Cybernetica*, 9:107–119, 1989.
9. T. Fukunaga, M. M. Halldórsson, and H. Nagamochi. Robust cost colorings. In SODA, 2008.
10. R. Gandhi, M. M. Halldórsson, G. Kortsarz and H. Shachnai. Approximating non-preemptive open-shop scheduling and related problems. *ACM Transactions on Algorithms*, 2(1):116–129, 2006.
11. A. Goel and A. Meyerson. Simultaneous optimization via approximate majorization for concave profits or convex costs. *Algorithmica*, 44:301–323, 2006.
12. R. Hassin and S. Rubinstein. Robust matchings. *SIAM J. Disc. Math*, 15(4):530–537, 2002.