

Follow the leader for online optimization

Adam Kalai
MIT

Santosh Vempala
MIT

Prepared for M.I.T. Operations Research Seminar Series
April 26, 2002

Abstract

Linear optimization is a central algorithmic problem with many applications. In this paper, we consider a natural online version: the optimization problem has to be solved repeatedly over a sequence of periods, where the objective direction for the upcoming period is unknown. This models online versions of well-known optimization problems, such as max-cut, variants of clustering, and also the classic online binary search tree problem. We present algorithms for these problem which are $(1 + o(1))$ -competitive with the optimal static solution chosen in hindsight. Our algorithms and proofs are motivated by geometric considerations.

1 Introduction

The standard linear optimization problem can be stated as follows:

- Feasible set $S \subset \mathbb{R}^n$
- Objective vector $c \in \mathbb{R}^n$
- Goal: choose $x \in S$ to minimize $c \cdot x$

This problem and many of its special cases have been widely studied, for example linear programming, semidefinite programming, and also discrete problems such as max-cut, traveling salesman, optimal binary search trees, etc.. Each of these problems has an online version, and some are more natural than others. We consider the following online linear optimization problem:

- Feasible set $S \subset \mathbb{R}^n$
- Bounded objective vectors $c_1, c_2, \dots \in \mathbb{R}^n$
- Choose $x_j \in S$ knowing only c_1, c_2, \dots, c_{j-1} .
- Goal: minimize $\sum_j c_j \cdot x_j$

Let's consider some examples.

(1) Max-cut: we have a multigraph and we must choose a cut. The score of a cut is the number of edges crossing the cut (we refer to score instead of cost for maximization problems). In the online version of this linear maximization problem¹, one edge is added at a time. Without knowledge of the next edge, we must choose a cut, and receive a score of 1 if the edge crosses the cut and 0 otherwise.

(2) Binary search trees: we have n nodes and a sequence of accesses to these nodes. We must choose a binary search tree that minimizes the sum of the depths of the accessed nodes. This can be viewed as a linear minimization problem where the sequence of accesses is represented by an objective vector c whose i th coordinate is the number of accesses to node i , and each tree is represented by a vector whose

¹To view max-cut as a linear optimization problem, consider a coordinate for each pair of vertices (u, v) . The objective vector c at each coordinate is the number of edges between u and v , and a cut is represented by a $0 - 1$ vector with 1s in the coordinates where u and v are on different sides. The feasible set in this case is not convex.

i th coordinate is the depth of node i . In the online version of this problem [16], we have to repeatedly choose a tree in anticipation of the next access.

(3) Linear programming: Consider the specific maximization example illustrated in Figure 1a. The offline version is trivial, but the online version, where we have a sequence of cost vectors c_j , is not. This example illustrates that the naive “follow-the-leader” algorithm performs poorly. Using what worked best for the empirical distribution of the past $c_1 + \dots + c_{j-1}$ will choose the vector $x_j = (1, 0)$ when $c_j = (0, 1)$ and $x_j = (0, 1)$ when $c_j = (1, 0)$. It’s not clear what follow-the-leader does on the first period, but the total score is at most $1/2 + 0 + 0 + \dots = 1/2$.

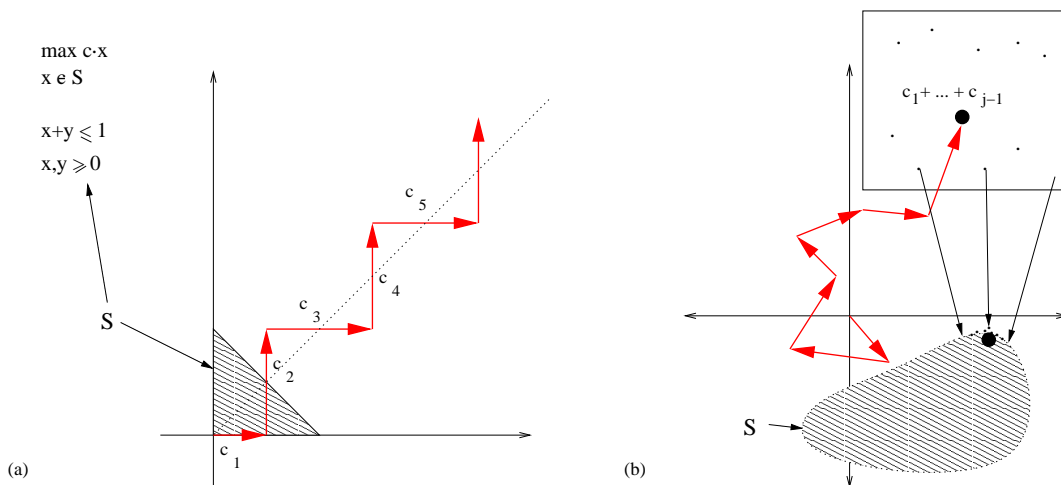


Figure 1: (a) An example online maximization problem in which doing what worked best in the past is bad. (b) An illustration of the $FEL(\mu, s)$ algorithm for μ uniform on a cube. Samples directions are drawn from the cube around $c_1 + \dots + c_{j-1}$ and then the maxima in those directions are averaged.

A further requirement that one might have is that the solution is not changed frequently from period to period. Our Follow the Lazy Leader (FLL) algorithm described in Section 1.1 addresses this issue and change an expected $O(\sqrt{t})$ times on a sequence of length t .

To analyze the performance of an online algorithm, we compare to the performance of an optimal offline algorithm [15]. If one had to use a single solution $x \in S$ over all t periods offline (knowing all c_j ’s in advance), one would use a solution which minimizes the total (equivalently average) cost. This would give a cost of,

$$\min\text{-cost}_j = \min_{x \in S} (c_1 + c_2 + \dots + c_j) \cdot x$$

We compare our cost to this minimum (called static offline adversary) in terms of both additive and multiplicative bounds². While we have the flexibility to change from period to period, on period j , we only know c_1, c_2, \dots, c_{j-1} .

Of course, the best cost one could achieve on period j is $\min_{x \in S} c_j \cdot x$. However, it is impossible to be competitive against such a dynamically optimal solution (dynamic offline adversary).

An intuitive solution, which we’ve called follow-the-leader, does the following. On period j , it uses the best solution on the observed data c_1, \dots, c_{j-1} . In Example (3) above, this solution achieves a score of at most $1/2$, while the optimal static solution achieves a score of $t/2$ over t periods. A similar situation occurs in the max-cut problem. Consider a 3-node graph with edges u, v , and w , and request sequence u, v, w, u, v, w, \dots . In the worst case, follow-the-leader achieves a score of 0 while the best static solution gets $\frac{2}{3}t$. Note that randomly breaking ties for the leader does not solve this problem (in Example (3), there are no ties).

²Additive error is often called *regret* and multiplicative error is termed *competitive ratio*.

Our idea is to use randomness in the space of cost vectors rather than directly in the space of feasible solutions, which may have a complicated structure. We do not attempt to solve the optimization problem for arbitrary feasible sets S . Instead we assume we have some minimization oracle M , that, given a cost vector (direction) $c \in \mathbb{R}^n$ returns some minimum of $c \cdot x$ over $x \in S$. In other words,

$$M(c) = \arg \min_{x \in S} c \cdot x$$

Follow The Expected Leader **FEL**(μ, s) // Requires convex S

- On each period j :
 1. Choose r_1, r_2, \dots, r_s independently from distribution μ .
 2. Use $x_j := \frac{1}{s} \sum_{i=1}^s M(c_1 + c_2 + \dots + c_{j-1} + r_i)$

For FEL, as opposed to FLL, the set S must be convex because we are choosing x_j to be the average of points in S . If, in addition the set S has a separation oracle, then M and thus FEL can be implemented in polynomial time [7].

For the following theorems, we assume that $|c_j|_1 \leq 1$ for all j . Also, let D is an upper bounds on the L_1 diameter of the feasible set S . Admittedly, it is artificial to use an L_1 norm for this problem, since there is nothing inherently axis-aligned about the problem. In the appendix, we state corresponding bounds for the L_2 norm, which are in general incomparable, but have worse dependence on the dimension n and thus are worse for our applications. The difference in analysis is depicted later in Figure 2, where we can see that the non-overlap region of two cubes shifted by $|c|_1 = 1$ is smaller than the non-overlap of two balls shifted by $|c|_2 = 1$. In general one could consider a bounded set of possible objective vectors, which may not contain the origin, and simple modifications of our algorithms will give bounds that depend on the L_1 diameter of this set in a manner similar to D .

Theorem 1. *Let μ be uniform on the cube $\frac{1}{\epsilon}[-1, 1]^n$. Then for FEL(μ, s) (any $s \geq 1$) and FLL(μ),*

$$E[\text{FLL's cost}_t] = E[\text{FEL's cost}_t] \leq \text{min-cost}_t + D \left(\frac{\epsilon t}{2} + \frac{1}{\epsilon} \right)$$

If we set $\epsilon = 1/\sqrt{2^i}$ for 2^i periods, and then restart, (for $i = 1, 2, \dots$), then after t periods,

$$\begin{aligned} E[\text{FLL's cost}_t] &\leq \text{min-cost}_t + 6D\sqrt{t}, \\ \text{for } s = \log 1/\delta \text{ with prob. } &\geq 1 - \delta, \quad \text{FEL's cost}_t \leq \text{min-cost}_t + 7D\sqrt{t} \end{aligned}$$

Further, the expected number of times FLL changes solution or even queries M is only $O(\sqrt{t})$.

With no further assumptions, it is impossible to be constant competitive in a multiplicative sense, e.g. $\text{min-cost}_t = 0$. However, if we assume that all costs are positive, i.e. $c_j \cdot x \geq 0$ for all j and $x \in S$, then we obtain the following multiplicative guarantees. Similar guarantees are obtained if we assume $c_j \cdot x \leq 0$ for all j and $x \in S$.

Theorem 2. *Let $d\mu(x) \propto e^{-\epsilon|x|_1/2}$ for $0 \leq \epsilon \leq 1$. Then for FEL(μ, s) (any $s \geq 1$) and FLL(μ),*

$$E[\text{FLL's cost}_t] = E[\text{FEL's cost}_t] \leq (1 + \epsilon)\text{min-cost}_t + \frac{4D(\log n + 1)}{\epsilon}$$

Further, FLL changes solution or queries M with probability $\leq \epsilon$ each period. For an appropriate schedule of reducing ϵ and restarting,

$$\begin{aligned} \sqrt{E[\text{FLL's cost}_t]} &\leq \sqrt{\text{min-cost}_t} + \sqrt{O(\log n)D}, \\ \text{for } s = \log 1/\delta \text{ with prob. } &\geq 1 - \delta, \quad \sqrt{\text{FEL's cost}_t} \leq \sqrt{\text{min-cost}_t} + \sqrt{O(\log n)D}. \end{aligned}$$

These theorems are proven in Section 2.2. As easy applications, we present online linear programming (section 3) and statically optimal binary search trees³ (section 5). Online linear optimization also generalizes the well-known problem of predicting from expert advice (section 6), the case where S is a simplex.

We consider the situation when we have only an approximate minimization oracle A rather than an exact oracle M . In section 4, we show that our algorithm can be used with a large class of approximation algorithms, which are what we call *pointwise approximate*. Some examples include the max-cut algorithm of [8], the classification algorithm of [11], etc..

1.1 Follow the lazy leader

FLL is designed to serve two purposes. First, in the cases where S is not convex, which include many of our examples, we cannot get high probability bounds but only expectation bounds when picking $x_j \in S$. Secondly, FLL is lazy in the sense that it rarely changes, i.e. $x_j = x_{j-1}$ most of the time. Essentially, the goal is to have the same exact distribution as FEL but being as lazy as possible.

For ease of notation, let

$$c_{1\dots j} = c_1 + c_2 + \dots + c_j$$

The distribution over $v = r + c_{1\dots j-1}$ when r is chosen according to μ is $\mu(v - c_{1\dots j-1})$. In order to update from $\mu(v - c_{1\dots j-1})$ to $\mu(v - c_{1\dots j})$, we stay with as large a probability as we can, and move with the remaining probability. In order to move in such a way achieve the new distribution, we assume that μ is symmetric about the origin, i.e. $\mu(x) = \mu(-x)$ and then we can do a simple reflection about the point $(c_{1\dots j} + c_{1\dots j-1})/2$. This gives:

Follow the lazy leader **FLL**(μ) // μ should be symmetric about origin

1. Choose v from distribution μ .
2. On each period j ,
 - (a) If v has changed, then $x_j := M(v)$, else $x_j := x_{j-1}$.
 - (b) After receiving c_j , with prob. $\max\left(0, 1 - \frac{d\mu(v - c_{1\dots j})}{d\mu(v - c_{1\dots j-1})}\right)$, set $v := c_{1\dots j} + c_{1\dots j-1} - v$.

2 Analysis

FEL and FLL may seem like very different algorithms, but in fact they have the same expectation every period. The advantage of FEL achieves its bounds with high probability, but it requires S to be convex and uses s calls to the oracle per period. FLL achieves the same expected bounds as FEL, does not require S to be convex, and uses few calls to the oracle. For the purpose of analysis, we introduce a third algorithm, Follow the Random Leader:

FRL(μ) : Initially choose r from μ . On period j use $x_j := M(r + c_{1\dots j-1})$.

Lemma 1. *FRL*(μ), *FLL*(μ), and *FEL*(μ, s) (for any s) have the same expected costs:

1. For all s , *FEL*(μ, s) and *FRL*(μ) have equal expected costs each period.
2. If $\forall x \in \mathbb{R}^n, d\mu(x) = d\mu(-x)$, *FLL*(μ) and *FRL*(μ) have equal expected costs each period.

Proof. The first statement follows trivially from linearity of expectation. Let ν_j be the distribution of v on period j . Then, for the second statement, it suffices to show that the distribution over v in period j of FLL is identical to the distribution over $r + c_{1\dots j-1}$ in FRL, i.e. $d\nu_j(x) = d\mu(x - c_{1\dots j-1})$, because FLL uses $M(v)$ while FRL uses $M(r + c_{1\dots j-1})$. This can be seen by induction. The base case is trivial.

³Our algorithm is $(1 + o(1))$ -competitive. In contrast, Splay trees [16] are $3 \log_2 3$ competitive [16]. We are referring to *static optimality*. We do not address the longstanding conjecture of dynamic optimality of splay trees, and our algorithms are not dynamically optimal.

If $d\nu_j(x) = d\mu(x - c_{1\dots j-1})$, then we can write $d\nu_{j+1}(x)$ in terms of the probabilities of getting to x from x and from $(c_{1\dots j} + c_{1\dots j-1} - x)$:

$$\begin{aligned} d\nu_{j+1}(x) &= d\nu_j(x) \left(1 - \max \left(0, 1 - \frac{d\mu(x - c_{1\dots j})}{d\mu(x - c_{1\dots j-1})} \right) \right) + \\ &\quad d\nu_j(c_{1\dots j} + c_{1\dots j-1} - x) \max \left(0, 1 - \frac{d\mu(c_{1\dots j-1} - x)}{d\mu(c_{1\dots j} - x)} \right) \\ &= \min(d\mu(x - c_{1\dots j-1}), d\mu(x - c_{1\dots j})) + \max(0, d\mu(c_{1\dots j} - x) - d\mu(c_{1\dots j-1} - x)) \end{aligned}$$

This last equality follows by induction hypothesis and a little algebra. Finally, since $d\mu(x) = d\mu(-x)$, the above is $d\mu(x - c_{1\dots j})$, which is what we wanted. \square

The probability of FLL querying the oracle during period j is

$$E_{\nu_j} \left[\max \left(0, 1 - \frac{d\mu(v - c_{1\dots j})}{d\mu(v - c_{1\dots j-1})} \right) \right] = \int_{v \in \mathbb{R}^n} \max(0, d\mu(v - c_{1\dots j-1}) - d\mu(v - c_{1\dots j})) dv \quad (1)$$

Definition 1. Let $\text{diff}_\mu(c)$ be the variation distance between μ and μ shifted by c , i.e.

$$\text{diff}_\mu(c) = \int_{x \in \mathbb{R}^n} \max(0, d\mu(x) - d\mu(x - c))$$

Lemma 2. The expected number of oracle queries of FLL(μ) is $\sum_j \text{diff}_\mu(c_j)$.

Proof. This follows from (1) and the change of variable $x = v - c_{1\dots j-1}$. \square

Since we are comparing ourselves to this min-cost $_t$, let us see how it changes each period.

$$\begin{aligned} \text{min-cost}_j - \text{min-cost}_{j-1} &= M(c_{1\dots j}) \cdot c_{1\dots j} - M(c_{1\dots j-1}) \cdot c_{1\dots j-1} \\ &\geq M(c_{1\dots j}) \cdot c_j \end{aligned}$$

The above follows by definition of M , because $M(c_{1\dots j}) \cdot c_{1\dots j-1} \geq M(c_{1\dots j-1}) \cdot c_{1\dots j-1}$.

Of course, with the knowledge of c_j , the best thing to do on period j would be $M(c_j)$. However, the above also tells us that if we were to use $M(c_{1\dots j})$ on period j , we would have no regret. To be precise, summing from 1 to t , we get,

$$\sum_{j=1}^t M(c_{1\dots j}) \cdot c_j \leq \text{min-cost}_t \quad (2)$$

Now, follow the leader uses $M(c_{1\dots j-1})$ on period j , which appears similar to the above. However, as in Example (3), the leader may change very often. Our plan is to add randomness so that, in expectation, $M(c_{1\dots j})$ is similar to $M(c_{1\dots j-1})$.

That is why we introduced FRL. Over t periods, it has an expected cost of

$$E[\text{FRL}(\mu)\text{'s cost}_t] = E\left[\sum_1^t M(r + c_{1\dots j-1}) \cdot c_j\right]$$

While we cannot yet bound this cost, we can bound the cost of using $M(r + c_{1\dots j})$ on period j .

Lemma 3.

$$\sum_1^t M(r + c_{1\dots j}) \cdot c_j \leq \text{min-cost}_t + (M(c_{1\dots t}) - M(r)) \cdot r$$

Proof. If we pretended that r was the objective on an initial day 0, then (2) gives,

$$M(r) \cdot r + \sum_1^t M(c_{1\dots j}) \cdot c_j \leq M(r + c_{1\dots t}) \cdot (r + c_{1\dots t})$$

By definition of M , we also have

$$M(r + c_{1\dots t}) \cdot (r + c_{1\dots t}) \leq M(c_{1\dots t}) \cdot (r + c_{1\dots t}) = \text{min-cost}_t + M(c_{1\dots t}) \cdot r$$

Combining the above gives the lemma. \square

2.0.1 Additive lemma

Based on the above lemma, in order to show that FRL does well, it would suffice to show that $M(r + c_{1\dots j-1}) \cdot c_j$ is close to $M(r + c_{1\dots j}) \cdot c_j$. However, for a single choice of r , these two quantities may be very different. Fortunately, with the right kind of randomness, we can bound their expected difference,

$$E[(M(r + c_{1\dots j-1}) - M(r + c_{1\dots j})) \cdot c_j] = E[(M(r_1 + c_{1\dots j-1}) - M(r_2 + c_{1\dots j})) \cdot c_j] \quad (3)$$

Without changing the expectation, we can correlate r_1 and r_2 as we like, as long as they are each individually distributed according to μ . In particular, we choose to make $r_1 = r_2 + c_j$ as often as possible. How often we can make them equal depends on how much the distribution μ changes when we shift it by c_j , i.e. the variation distance between $\mu(x)$ and $\mu(x - c_j)$, $\text{diff}_\mu(c_j)$. With probability $1 - \text{diff}_\mu(c_j)$, we make $r_1 = r_2 + c_j$ so $r_1 + c_{1\dots j-1} = r_2 + c_{1\dots j}$, and the difference in (3) is zero. Thus with probability at most $\text{diff}_\mu(c_j)$ the quantity is nonzero and is at most $\max_{a,b \in S}(a - b) \cdot c_j$. Thus we have argued that,

$$E[(M(r + c_{1\dots j-1}) - M(r + c_{1\dots j})) \cdot c_j] \leq \text{diff}_\mu(c_j) \max_{a,b \in S}(a - b) \cdot c_j$$

Putting this together with Lemma 3, we can bound our additive regret,

Lemma 4. *For r chosen from μ , over t periods,*

$$E[\text{FRL}(\mu) \text{'s cost}_t] \leq \text{min-cost}_t + \sum_1^t \text{diff}_\mu(c_j) \max_{a,b \in S}(a - b) \cdot c_j + E[\max_{a,b \in S}(a - b) \cdot r]$$

Proof. This follows from the two lemmas and the definitions of min-cost_t and $\text{FRL}(\mu)$'s cost_t . \square

2.0.2 Multiplicative lemma

Like diff , let,

$$\text{rat}_\mu(c_j) = \max_{x \in \mathbb{R}^n} \frac{d\mu(x)}{d\mu(x - c_j)} \quad (4)$$

Then,

$$\begin{aligned} E[M(r + c_{1\dots j-1}) \cdot c_j] &= \int_{r \in \mathbb{R}^n} M(r + c_{1\dots j-1}) \cdot c_j d\mu(r) \\ &= \int_{r \in \mathbb{R}^n} M(r + c_{1\dots j}) d\mu(r + c_j) \\ &\leq \int_{r \in \mathbb{R}^n} M(r + c_{1\dots j}) \text{rat}_\mu(c_j) d\mu(r) \\ &= \text{rat}_\mu(c_j) E[M(r + c_{1\dots j}) \cdot c_j] \end{aligned}$$

Combining this with Lemma 3, we get

Lemma 5.

$$E[\text{FRL}(\mu) \text{'s cost}_t] \leq (\max_j \text{rat}_\mu(c_j)) (\text{min-cost}_t + E[\max_{a,b \in S}(a - b) \cdot r])$$

2.1 Geometric parameters

For any vector $c \in \mathbb{R}^n$, let μ_c be the distribution defined as $\mu(x) = \mu_c(x - c)$.

Lemma 6. *Let μ be the uniform distribution on a cube of side R in \mathbb{R}^n . Then for any vector $c \in \mathbb{R}^n$,*

$$\text{diff}_\mu(c) \leq \frac{|c|_1}{R}.$$

Proof. WLOG we can assume that all the components of c, c^1, c^2, \dots, c^n , are nonnegative; also, suppose μ is uniform over the cube of points where each coordinate is between 0 and R . Then μ_c is uniform over the cube of points x such that x^i is between c^i and $R + c^i$. Hence, the points that lie in the first cube but not the second are exactly those for which $x^i < c^i$, for some i . The measure of these points, by the union bound, is at most,

$$\text{diff}_\mu(c) \leq \sum \frac{c^i}{R} = \frac{|c|_1}{R}$$

□

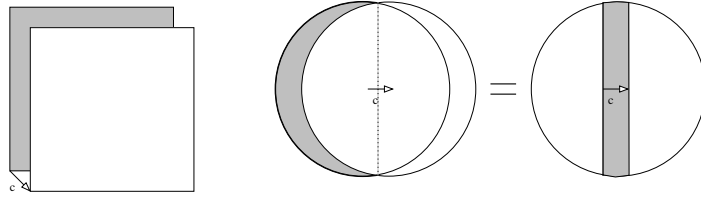


Figure 2: The variation distance between shifted cubes and balls. In the balls, one can see that the two shaded regions have the same area if the nonshaded sections have the same area.

Lemma 7. *Let μ be the uniform distribution on a ball of radius R in \mathbb{R}^n . Then for any vector $c \in \mathbb{R}^n$,*

$$\text{diff}_\mu(c) \leq \frac{|c|_2 \sqrt{n}}{R}.$$

Proof. We can think of $\text{diff}_\mu(c)$ as proportional to the volume of the shaded region in Figure 2, i.e. “non-overlap” of two balls, of radius R , whose centers are at distance $|c|_2$ from each other. As seen in the figure, this is exactly the fraction of a ball of radius R that lies in between two parallel planes each at distance $|c|_2/2$ from the center of the ball. This can be upper bounded as the volume of a cylinder whose base is an $(n - 1)$ -dimensional ball of radius R and its height is $|c|_2$. Thus,

$$\begin{aligned} \text{diff}_\mu(c) &\leq \frac{|c|_2 \text{vol}(B_{n-1}(R))}{\text{vol}(B_n(R))} \\ &\leq \frac{|c|_2 \sqrt{n}}{R}. \end{aligned}$$

Here we have used the fact that

$$\text{vol}(B_n(r)) = \frac{2R^n \pi^{n/2}}{n \Gamma(n/2)},$$

($\Gamma(x)$ is the Euler Gamma function; for a positive integer x , $\Gamma(x) = (x - 1)!$). □

Lemma 8. *Let μ be the distribution with $d\mu(x)$ proportional to $e^{-\epsilon|x|}$. Then for any vector $c \in \mathbb{R}^n$,*

$$\text{rat}_\mu(c) \leq e^{\epsilon|c|}.$$

Proof. For any x , by the triangle inequality,

$$\begin{aligned} \text{rat}_\mu(c) &= \frac{e^{-\epsilon|x|_1}}{e^{-\epsilon|x+c|_1}} \\ &= e^{\epsilon(|x+c|_1 - |x|_1)} \\ &\leq e^{\epsilon|c|_1}. \end{aligned}$$

□

2.2 Performance guarantees

Again, D is an upper bounds on the L_1 diameter of the feasible set S . We assume $|c_j|_1 \leq 1$ for all j . Now we prove the theorems in the introduction.

Proof. (of Theorem 1) We apply Lemma 4 to the chosen μ . For a cube of side $2/\epsilon$, by Lemma 6, we have $\text{diff}_\mu(c_j) \leq \frac{\epsilon}{2}|c_j|_1 \leq \frac{\epsilon}{2}$. Furthermore, $\max_{a,b \in S}(a-b) \cdot c_j \leq D$ since $|c_j|_1 \leq 1$. Finally, $\max(a-b) \cdot r \leq D/\epsilon$ for r in this cube. By Lemma 4, we have,

$$E[\text{FRL's cost}]_t \leq \text{min-cost}_t + D \left(\frac{\epsilon t}{2} + \frac{1}{\epsilon} \right)$$

By Lemma 1, FEL, FRL, and FLL have the same expectation (since μ is center symmetric). By Lemma 2, FLL makes an expected number of queries equal to $\text{diff}_\mu(c_j) \leq \epsilon/2$ queries per period.

Next we consider restarting the process every 2^i periods, for $i = 1, 2, \dots$. First, suppose we've completed phases $i = 1, 2, \dots, k$ so $t = 2^{k+1} - 2$. Then by the above,

$$E[\text{FRL's cost}]_t \leq \text{min-cost}_t + D \sum_{i=1}^k \left(\frac{2^{-i/2} 2^i}{2} + \frac{1}{2^{-i/2}} \right)$$

Simple algebra shows that the above sum is at most $\frac{3}{2}(1 + \sqrt{2})\sqrt{t}$. We would have to multiply t by at most 2 to make it of the form $2^{k+1} - 1$, so our regret is at most $D\frac{3}{2}(1 + \sqrt{2})\sqrt{2t} \leq 6D\sqrt{t}$.

To show the high probability bounds, note that our total cost is the sum of t periods, each period being the average of $\ln(1/\delta)$ random variables. Each variable is nonnegative and at most D . By Chernoff bounds, the probability that this sum is larger than its expectation by $D\sqrt{t}$ is at most δ . \square

Proof. (of Theorem 2) Using Lemmas 5 and 8, we get

$$E[\text{FRL}(\mu)\text{'s cost}_t] \leq e^{\epsilon/2}(\text{min-cost}_t + E[\max_{a,b \in S}(a-b) \cdot r])$$

Now, $E[|r|_\infty] \leq \frac{2+2\ln n}{\epsilon}$ and since $u \cdot v \leq |u|_1|v|_\infty$, we get

$$E[\text{FRL's cost}] \leq e^{\epsilon/2}(\text{min-cost}_t + \frac{2(\log n + 1)D}{\epsilon})$$

Using Lemma 1 and the fact that $e^{\epsilon/2} \leq 1 + \epsilon \leq 2$, the first equality in the theorem follows.

Sketch of second part: Again we run in phases i for $i = 1, 2, \dots$. During phase i we use $\epsilon = 1/\sqrt{2^i}$ until the minimum static cost during that phase is at least $2^i D$, then restart for the next phase. It is tedious but not difficult to show that this gives an expected cost,

$$E[\text{FLL's cost}] = E[\text{FEL's cost}] \leq \text{min-cost}_t + \sqrt{\text{min-cost}_t O(\log n)D} + O(\log n)D.$$

Taking square roots of the above implies the theorem. \square

A similar theorem holds for maximization problems with $c_j \cdot x \geq 0$.

3 online linear programming

In this section we consider an online version of linear programming. In the standard linear programming problem, we have n variables x^1, x^2, \dots, x^n and m constraints of the form $Ax \geq b$. The problem is to maximize a linear function $c \cdot x$. It is well-known that this classic problem can be solved in polynomial time [10, 7, 9].

In the online version, we have a repeated optimization problem where on period j , there is a new objective vector c_j . The problem is to pick a solution x_j on period j knowing only c_1, c_2, \dots, c_{j-1} . The

cost on period j is then $c_j \cdot x_j$. The goal is to minimize the total cost compared to the best single solution x picked in hindsight, which is the solution to the following linear program:

$$\begin{aligned} \text{min-cost}_t &= \min(c_1 + c_2 + \dots + c_t) \cdot x \\ &Ax \geq b \end{aligned}$$

We can apply our Follow the Expected Leader algorithm to this problem with L_2 norms described in the appendix. Let D_2 be the L_2 diameter of $\{x | Ax \geq b\}$.

Corollary 1. *Over t periods, $FEL(\mu, 2 \ln 1/\delta)$ with an ϵ -schedule and μ being uniform on a ball of radius \sqrt{n}/ϵ has the following guarantee: With prob. at least $1 - \delta$,*

$$FEL\text{'s cost}_t \leq \text{min-cost}_t + O(n^{1/4} D_2) \sqrt{t}$$

Corollary 2. *Suppose over t periods $c_j \cdot x \geq 0$ for all $1 \leq j \leq t$ and x s.t. $Ax \leq b$. Then $FEL(\mu, 2 \ln 1/\delta)$ with an ϵ -schedule and $d\mu \propto e^{-\epsilon|x|^2}$ has the following guarantee: With prob. at least $1 - \delta$,*

$$\sqrt{FEL\text{'s cost}_t} \leq \sqrt{\text{min-cost}_t} + O(\sqrt{nD}).$$

4 Approximation algorithms

In the previous section, we saw that the online version of linear optimization can be solved using an optimal offline algorithm. In particular, when the offline optimization problem can be solved exactly in polynomial-time, so can the online version. In this section, we consider the situation when the algorithm for the offline optimization problem is only guaranteed to find an *approximate* optimum.

We could apply our online algorithms here, with the change that instead of calling an exact optimization oracle M , we have access to an approximation algorithm A . We say that A achieves an α -approximation if, on any input, the cost of the solution it finds is at most α times the minimum solution for a minimization problem.

Here we consider optimization problems that have approximation algorithms with a special property which we call a *point-wise approximation*. For such problems, the cost incurred by our online algorithms using the approximation algorithm A instead of the exact oracle M goes up by at most α , the approximation factor of A .

Definition 2. *An approximation algorithm A for a linear minimization problem on variables x^1, \dots, x^n , is said to achieve an α point-wise approximation, if on any input instance x , the solution it finds, $A(x)$, has the property that $E[A(x)^i] < \alpha y^i$ for all i , for some optimal solution y .*

The definition for maximization problems is analogous. Several algorithms have point-wise guarantees, e.g. the max-cut algorithm of [8], the metric labeling algorithm of [11] etc.

Let S be the feasible set of a linear optimization problem with diameter D (in L_1 norm). Let A be an approximation algorithm that achieves an α point-wise approximation for the problem. In the online version of the problem the cost vector on period j is c_j . Below n is the dimension of the cost vectors.

Theorem 3. *Suppose that for an online minimization problem, the cost vectors are non-decreasing and vary by at most 1 (in L_1 norm) in a single period. Then FLL has the following guarantee:*

$$\sqrt{E[\text{online-cost}_t]} \leq \sqrt{\alpha(\sqrt{\text{min-cost}_t} + O(\sqrt{D \log n}))}.$$

A similar statement can be made for maximization problems, replacing $+$ by $-$.

As an example, consider the max-cut problem. In each new period, an edge is requested. The online algorithm maintains a cut in anticipation. The score on period j is 1 if the edge is cut and 0 otherwise. The goal is to have as large a reward as possible compared to the reward of the best single cut chosen in hindsight.

Corollary 3. *For the online max-cut problem on a multigraph on n nodes, FLL applied with the approximation algorithm of [8] has the following guarantee:*

$$\sqrt{E[\text{online-score}_t]} \geq 0.937 \sqrt{\text{max-score}_t} - O(n \sqrt{\log n}).$$

Proof. It is easy to see that the randomized algorithm of [8] for the offline problem has a pointwise guarantee of 0.87856. It remains to bound C and D . $C \leq 1$. $D \leq n^2$. \square

Other examples of approximation algorithms with pointwise guarantees include the randomized vertex ordering algorithms of [12, 6, 14, 5].

5 Binary search trees

For our purposes, a binary search tree is a binary tree where the nodes are labeled $1, 2, \dots, n$ and have the property that everything in a node's left subtree has smaller labels than the node, while everything in a node's right subtree has larger labels. Given a sequence of *accesses* from $\{1, 2, \dots, n\}$, the cost on that sequence for a given tree is the sum of the depths of the nodes in the sequence, because that is the time required to find the node.

Sleator and Tarjan viewed this as an online problem in which an algorithm is allowed to change the tree dynamically (but at a cost). They showed that splay trees perform, to within a factor of $3 \log_2 3$ as well as the best static tree, where the best is chosen in hindsight [16].

Suppose that there are a_i accesses to element i , for $1 \leq i \leq n$. Then the cost of a tree T is

$$\sum_{i=1}^n a_i \text{depth}_T(i) = a \cdot \text{depth}_T,$$

where depth_T is the vector of depths of nodes in tree T . So, in fact, finding the best static tree for a sequence of accesses is a linear optimization problem, where the objective is the vector a and the feasible set is the set of depth vectors of trees. Dynamic programming can be used to find the best tree in hindsight in time $O(n^2)$. Since we must pick a single tree and we would like to change trees few times, we use a variant of FLL.

Lazy trees **LT**(N)

1. For $1 \leq i \leq n$, let $a_i := 0$ and choose v_i randomly from $\{1, 2, \dots, N\}$
2. After i is accessed,
 - (a) $a_i := a_i + 1$
 - (b) If $a_i \geq v_i$ then
 - i. $v_i := v_i + N$
 - ii. Change trees to the best tree as if there were v_i accesses to node i .

While we could have directly applied FLL, we chose to use the above algorithm because it maintains the essential properties of FLL but does not involve pretending there were negative or fractional accesses and updates v one coordinate at a time. The essential property it maintains is that at any period v_i is randomly distributed between $a_i + 1$ and $a_i + N$, for each i . The L_1 diameter of the set of tree depth vectors is $O(n^2)$, so the above algorithm gets, after t accesses,

$$E[\text{LT}(N)\text{-cost}_t] \leq \text{min-cost}_t + O(n^2) \left(\frac{t}{N} + N \right)$$

Using a schedule on N , e.g. running with $N = 2^i$ for 4^i periods,

$$\begin{aligned} E[\text{LT-cost}_t] &\leq \text{min-cost}_t + O(n^2 \sqrt{t}) \\ &\leq \text{min-cost}_t + O(n^2 \sqrt{\text{min-cost}_t}) \end{aligned}$$

The last step follows because any tree must pay at least 1 for each access. Note that this is better than $1 + \epsilon$ competitive for any ϵ as the length of the access sequence grows. We have ignored the cost of changing trees. However, this algorithm changes trees an expected $O(\sqrt{t})$ times over t accesses, each time the computation of the tree takes $O(n^2)$ and the number of rotations required is $O(n)$.

6 Predicting from expert advice

The problem of predicting from expert advice [13] is the special case where

$$S = \{x \in \mathbb{R}^n \mid \sum x^i = 1, x^i \geq 0 \text{ for } 1 \leq i \leq n\}$$

Each coordinate is thought of as an expert which incurs a cost each period. Say the vector of costs on period j is c_j . We can choose to make any prediction which is a convex combination of the experts' predictions, $x_j \in S$, and our cost is $x_j \cdot c_j$. The problem is that we require $|c_j|_1 \leq 1$ and typically only each component is bounded between 0 and 1. For the multiplicative bounds, we can fix this problem by observing that the worst case for our algorithm (and in fact most algorithms) is when each period only one expert incurs cost⁴.

This type of bound gives the standard $(1 + \epsilon)\text{min-cost}_t + O(\log n/\epsilon)$ bounds of weighted majority. Furthermore, one can view weighted majority itself as a FEL style algorithm for a particular distribution μ , but where the computation of the expected leader is done exactly rather than by sampling. It is trivial to see that the leader does not change much from period to period because we multiply weights by only $(1 - \epsilon)$ and this gives a nice analysis of the algorithm.

Conversely, one may be tempted to solve our problem using the tools of predicting from expert advice. For, most of our applications could also be solved by combining a set of experts. In terms of the feasible set, one could have an expert for each vertex. Then doing as well as the best expert would imply doing as well as the best point in the feasible set. However, this has several problems. First there may be infinitely many vertices. This may be solved by discretization, but straightforward discretization would require a number of experts exponential in n , which would give bounds with linear dependence on n rather than logarithmic dependence. Second, combining these experts would require averaging over an exponential set. While this is possible for some particular problems, either implicitly or with complicated random walks, each particular problem requires a new solution. Finally, it appears that for many linear optimization problems, the relevant parameter is the dimension of the space of cost vectors rather than the number of feasible solutions.

7 Conclusions and Related Work

We hope that the online linear optimization algorithms presented here will be useful for solving problems in online algorithms and mistake-bound learning. We present algorithms that take an optimization oracle and add randomness to the set of inputs (cost vectors) to achieve good online performance with polynomial run times. While it is common to add randomness or weights to compute a feasible solution to such problems, our algorithms give a generalization of how to do this in terms of optimizing and adding randomness to the space of costs.

Of course, following the leader is not a new idea. In fact, FRL was inspired by a similar solution for one particular problem, the list update problem [3]. However, their algorithm does not generalize to other problems with arbitrary feasible sets.

The results of this paper lead to several questions. Although, as remarked in the introduction, dynamic optimality is impossible without additional costs, it is possible to be competitive with the best dynamic solution that is allowed to change k times [13]. Using a simple trick presented for the experts framework in [1], any $(1 + \epsilon)$ -competitive algorithm with an additive a/ϵ term can be used to get a $(1 - \epsilon)$ -competitive algorithm against a solution that can change k times, with an additive ka/ϵ^2 term.

One natural question that arises is: what happens if we introduce a cost for moving, say proportional to $|x_j - x_{j-1}|$? Can we now be dynamically competitive? Another question is how follow-the-leader style algorithms can be used for dynamic problems. For some problems, modifying the cost function may allow one to show that the dynamic leader does not change much.

⁴Imagine comparing two scenarios, one with one period $c_1 = (a, b)$ and the second with two periods $c_1 = (a, 0)$ and $c_2 = (0, b)$. It is not difficult to see that our cost in the second scenario is larger, because we have more weight on the second expert after the first period. Nevertheless, the cost of the best expert in both scenarios is the same.

It would be nice to better take advantage of the geometry of the feasible sets and the set from which the costs are permitted. It would also be nice to be able to have algorithms without a parameter ϵ , that perhaps add randomness period by period.

Finally, it would be great to generalize FEL to nonlinear problems such as portfolio prediction [4].

References

- [1] Avrim Blum. On-line Learning and the Metrical Task System Problem. Presented at the *DIMACS workshop on Online Decision Making*, July 1999. See <http://www.cs.cmu.edu/~avrim/surveys.html>.
- [2] Avrim Blum and Carl Burch. On-line learning and the metrical task system problem. *Machine Learning*, 39(1):35–58, April 2000.
- [3] Avrim Blum, Shuchi Chawla, and Adam Kalai. Static Optimality and Dynamic Search Optimality in Lists and Trees. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '02)*, 2002.
- [4] Thomas Cover. Universal Portfolios. In *Math. Finance* 1, 1-29, 1991.
- [5] J. Dunagan and S. Vempala, “On Euclidean embeddings and bandwidth minimization,” *Proc. of the 5th Intl. Symp. on Randomization and Approximation techniques in Computer Science*, 2001.
- [6] U. Feige, “Approximating the bandwidth via volume respecting embeddings,” in Proc. 30th ACM Symposium on the Theory of Computing, 1998.
- [7] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer, 1988.
- [8] M. Goemans and D. Williamson, “Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming,” *J. ACM*, 42, 1115–1145, 1995.
- [9] N. K. Karmarkar, “A new polynomial-time algorithm for linear programming,” *Combinatorica*, 4:373–395, 1984.
- [10] L. G. Khachiyan, “A polynomial algorithm in linear programming,” (in Russian), *Doklady Akedamii Nauk SSSR*, 244, 1093–1096, 1979 (English translation: *Soviet Mathematics Doklady*, 20, 191–194, 1979).
- [11] J. Kleinberg, E. Tardos, “Approximation Algorithms for Classification Problems with Pairwise Relationships: Metric Labeling and Markov Random Fields,” *Proc. of 39th Foundations of Computer Science*, 1999.
- [12] N. Linial, E. London and Y. Rabinovich, “The geometry of graphs and some of its algorithmic applications,” *Proc. of 35th Foundations of Computer Science*, 1994.
- [13] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.
- [14] S. Rao, “Small distortion and volume preserving embeddings for planar and Euclidean metrics,” *Proc. of Symposium on Computational Geometry*, 1999.
- [15] Daniel Sleator and Robert Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28:202-208, 1985.
- [16] Daniel Sleator and Robert Tarjan. Self-Adjusting Binary Search Trees. *Journal of the ACM* 32:652-686, 1985.

8 Appendix: L_2 guarantees

Here we assume $|c_j|_2 \leq 1$ and D_2 is an upper bound on the diameter of S .

Theorem 4. Let μ be uniform on the ball of radius $\frac{1}{\epsilon}$. Then for $FEL(\mu, s)$ (any $s \geq 1$) and $FLL(\mu)$,

$$E[FLL's\ cost_t] = E[FEL's\ cost_t] \leq \min\text{-cost}_t + D_2 \left(\epsilon t \sqrt{n} + \frac{1}{\epsilon} \right)$$

For an appropriate schedule of reducing ϵ , after t periods,

$$\begin{aligned} E[FLL's\ cost_t] &\leq \min\text{-cost}_t + O(n^{1/4})D_2\sqrt{t}, \\ \text{for } s = \log 1/\delta \text{ with prob. } &\geq 1 - \delta, \quad FEL's\ cost_t \leq \min\text{-cost}_t + O(n^{1/4})D_2\sqrt{t}. \end{aligned}$$

Further, the expected number of times FLL changes solution or even queries M is only $O(n^{1/4}\sqrt{t})$.

With no further assumptions, it is impossible to be constant competitive in a multiplicative sense, e.g. $\min\text{-cost}_t = 0$. However, if we assume that all costs are positive, i.e. $c_j \cdot x \geq 0$ for all j and $x \in S$, then we obtain the following multiplicative guarantees. Similar guarantees are obtained if we assume $c_j \cdot x \leq 0$ for all j and $x \in S$.

Theorem 5. Let $d\mu$ be proportional to $e^{-\epsilon|x|_2/2}$ for $0 \leq \epsilon \leq 1$. Then for $FEL(\mu, s)$ (any $s \geq 1$) and $FLL(\mu)$,

$$E[FLL's\ cost_t] = E[FEL's\ cost_t] \leq (1 + \epsilon)\min\text{-cost}_t + \frac{O(nD_2)}{\epsilon}$$

Further FLL changes solution or queries with probability $\leq \epsilon$ each period.

For an appropriate schedule of reducing ϵ and restarting,

$$\begin{aligned} \sqrt{E[FLL's\ cost_t]} &\leq \sqrt{\min\text{-cost}_t} + \sqrt{O(nD_2)}, \\ \text{for } s = \log 1/\delta \text{ with prob. } &\geq 1 - \delta, \quad \sqrt{FEL's\ cost_t} \leq \sqrt{\min\text{-cost}_t} + \sqrt{O(nD_2)}. \end{aligned}$$