Geometric algorithms for online optimization

Adam Kalai* Santosh Vempala*

Abstract

In this paper, we consider a natural online version of linear optimization: the 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 optimization problems, such as max-cut, variants of clustering, and also the classic online binary search tree problem. We present algorithms for this problem that are $(1+\epsilon)$-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:

Given a feasible convex set $S \subset \mathbb{R}^n$ and an objective vector $c \in \mathbb{R}^n$, find x in S that minimizes $c \cdot x$.

In an online repeated setting, the objective vectors may be unknown. We consider the following online problem:

Given a feasible convex set $S \subset \mathbb{R}^n$, and a sequence of objective vectors $c_1, c_2, \ldots c_t \in \mathbb{R}^n$, choose a sequence of points x_j in S that minimizes $\sum_j c_j \cdot x_j$. When choosing x_j, only $c_1, c_2, \ldots c_{j-1}$ are known.

We do not assume that the each cost vector is in any way near to the previous ones. We will present algorithms for this problem and analyze them by comparing their performance to the performance of an optimal static offline algorithm [15]. Indeed, if one had to use a single solution $x \in S$ over all t periods offline (knowing the c_j's in advance), the minimum cost one could achieve is,

$$\text{min-cost}_t = \min_{x \in S} (c_1 + c_2 + \cdots + c_t) \cdot x$$

We compare our cost to this minimum both in terms of additive regret and multiplicative competitive ratio[1]. Here are some examples of online optimization problems.
(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). As described in Section 3, this can be viewed as a linear maximization problem with a parameter for the number of times each edge occurs in the graph. In the online version, 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. As described in Section 4, this can be viewed as a linear minimization problem where the sequence of accesses is represented by an objective vector c whose ith coordinate is the number of accesses to node i, and each tree is represented by a vector whose ith coordinate is the depth of node i. In the online version of this problem [16], we

*Department of Mathematics and Laboratory for Computer Science, MIT. Email: {akalai, vempala}@math.mit.edu

[1] This may seem like an unfair comparison, because we allow online algorithms the flexibility to change each period. However, offline algorithms are given the c_j's in advance, and in general, no online algorithm can compare to $\sum \min_{x \in S} c_j \cdot x$.

have to repeatedly choose a tree in anticipation of the next access. For this problem our algorithm (*lazy trees*, Section 4) is $(1+\epsilon)$-competitive. In contrast, Splay trees 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.

The feasible sets in both these examples are discrete. However, we can consider S to be the convex hull of the feasible points viewed as vectors. This maintains the maximum point and puts these examples in our framework.

1.1 Follow the expected leader

Intuitively, on period j, one could use the best solution on the observed data $c_1, \ldots, c_{j-1}$. The example illustrated by Figure 1.1a demonstrates the problem with this "follow-the-leader" approach. Here $c_1 = (0.5, 0)$ while the remaining c_j's alternate between $(1,0)$ and $(0,1)$. After the first period, the leader is the vector $x_j = (1,0)$ when $c_j = (0,1)$ and $x_j = (0,1)$ when $c_j = (1,0)$, achieving a score of 0 always. In contrast the optimal static solution to this *maximization* problem gives max-score$_j = j/2$. The set S being the simplex corresponds to the classical "predicting from expert advice" problem [13] with a dimension for each expert. There are many elegant solutions to that problem, but, for more complex sets, we will need another idea.

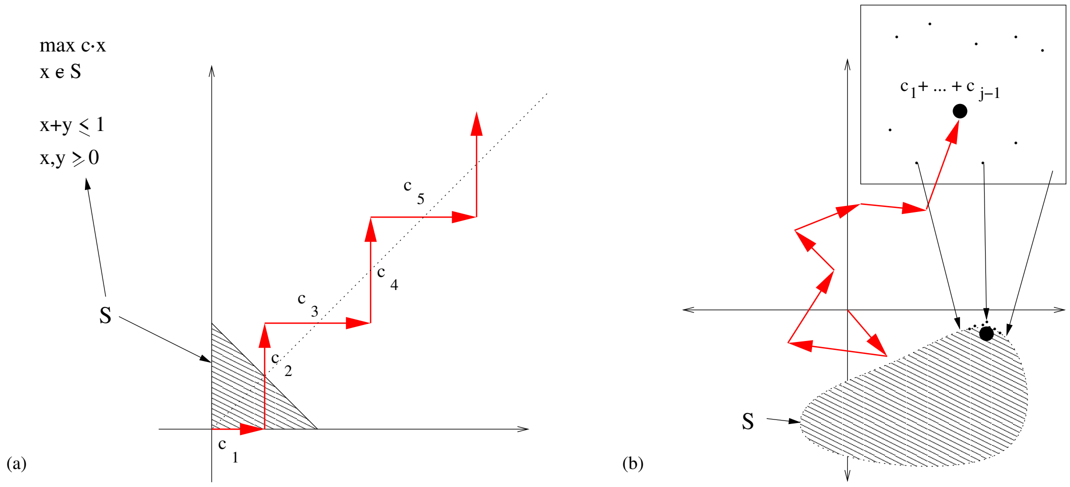

Figure 1: (a) An example online maximization problem in which following the leader is bad. (b) An illustration of the FEL(μ, s) algorithm for μ uniform on a cube. Samples directions are drawn from the cube around $c_1 + \cdots + c_{j-1}$ and then the maxima in those directions are averaged.

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 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,

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

Our basic approach, described below and depicted in Figure 1.1b, is similar to an algorithm used for list update problem [3].

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

We also assume that $|c_j|_1 \le 1$ for all j, and that D is an upper bound on the L_1 diameter of S, i.e. $|x-y|_1 \le D$ for $x, y \in S$.

Theorem 1. *Let μ be uniform on the cube $\frac{1}{\epsilon}[-1,1]^n$. Then for any sequence of costs $c_1, c_2, \ldots$, and any t, with probability at least $1 - e^{-st\epsilon^2/2}$,*

$$\textit{FEL}(\mu,s)\textit{'s cost}_t \le \textit{min-cost}_t + D\left(\epsilon t + 1/\epsilon\right).$$

The setting of ϵ that minimizes the RHS of the above is $\epsilon = 1/\sqrt{t}$, giving

$$\text{FEL}(\mu,\text{s})\text{'s cost}_t \le \text{min-cost}_t + 2D\sqrt{t}.$$

For such an ϵ, we need only $s = 2\ln 1/\delta$ samples to have the guarantee with probability $1-\delta$. In general, a standard ϵ-halving technique can be used over time to give this guarantee simultaneously for all t.

A larger issue is that Theorem 1 gives an additive regret bound. For a general S, it is impossible to get multiplicative bounds comparing our cost to min-cost$_t$, e.g. if min-cost$_t = 0$. However, if we assume that all costs are nonnegative then we obtain the following:

Theorem 2. *Let $d\mu(x) \propto e^{-\epsilon|x|_1/2}$ for $0 \le \epsilon \le 1$. Then for any sequence of costs $c_1, c_2, \ldots$ such that $c_j \cdot x \ge 0$ for all j and $x \in S$, and any t, with probability at least $1 - e^{-s\epsilon^2\Omega(\frac{\log n}{\epsilon} + \frac{\textit{min-cost}_t}{D})}$,*

$$\textit{FEL}(\mu,s)\textit{'s cost}_t \le (1+\epsilon)\textit{min-cost}_t + O(\log n)D/\epsilon.$$

Now the choice of ϵ that minimizes the RHS is $\epsilon = \min(1, \sqrt{O(\log n)D/\text{min-cost}_t})$, giving something that can be written as

$$\sqrt{\text{FEL}(\mu,\text{s})\text{'s cost}_t} \le \sqrt{\text{min-cost}_t} + \sqrt{O(\log n)D}.$$

Again, a standard ϵ-halving technique can be used over time to give these guarantees for all t.

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 are not as good for our applications.

1.2 Follow the lazy leader

Follow the lazy leader (FLL) is designed to serve two purposes. First, it allows for non-convex feasible sets S, so long as we still have a maximization oracle M. Many of our examples fall into this category. 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 FLL, we cannot get high probability bounds but only expectation bounds.

For ease of notation, let

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

When r is chosen according to μ, the distribution over $v = r + c_{1\ldots j-1}$ is $\mu(v - c_{1\ldots j-1})$. In order to update from $\mu(v - c_{1\ldots j-1})$ to $\mu(v - c_{1\ldots j})$, we stay with as large a probability as we can, and move with the remaining probability. In order to move in such a way as to 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 $(c_{1\ldots j} + c_{1\ldots j-1})/2$. This gives:

Follow the lazy leader **FLL**(μ) // μ symmetric about origin, S not necessarily convex

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. $\max\left(0, 1 - \frac{d\mu(v - c_{1\ldots j})}{d\mu(v - c_{1\ldots j-1})}\right)$, set $v := c_{1\ldots j} + c_{1\ldots j-1} - v$.

In Section 2, we show that the solution changes very rarely. In fact, we make very few calls to the maximization oracle.

Theorem 3. *For any distribution μ symmetric about the origin and any s and t,*

$$E[\textit{FLL}(\mu)\textit{'s cost}_t] = E[\textit{FEL}(\mu,s)\textit{'s cost}_t]$$

For either μ uniform on the cube or $\frac{1}{\epsilon}[-1,1]^n$ or $d\mu(x) \propto e^{-\epsilon|x|_1/2}$, $Pr[x_j \neq x_{j-1}] \leq \epsilon$.

Since oracle calls may be expensive, it is good that we are only making a call with probability ϵ. Thus, for the optimal setting of $\epsilon = 1/\sqrt{t}$, the number of solution changes and oracle calls made by FLL is $O(\sqrt{t})$.

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 $x_j := M(r + c_{1\ldots j-1})$.

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

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 $r + c_{1\ldots j-1}$ in FRL, i.e. $d\nu_j(x) = d\mu(x - c_{1\ldots j-1})$, because FLL uses $M(v)$ while FRL uses $M(r + c_{1\ldots j-1})$. This can be seen by induction. The base case is trivial. If $d\nu_j(x) = d\mu(x - c_{1\ldots j-1})$, then we can write $d\nu_{j+1}(x)$ in terms of the probabilities of getting to x from x and from $(c_{1\ldots j} + c_{1\ldots j-1} - x)$:

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

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

The probability of FLL querying the oracle during period j is

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

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

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

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

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

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

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

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

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 $M(c_{1\ldots j})$ on period j, we would have no regret. To be precise, summing from 1 to t, we get,

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

Now, follow the leader uses $M(c_{1\ldots j-1})$ 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, $M(c_{1\ldots j})$ is similar to $M(c_{1\ldots j-1})$.

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

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

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

Lemma 3.

$$\sum_{1}^{t} M(r + c_{1\ldots j}) \cdot c_j \leq \textit{min-cost}_t + (M(c_{1\ldots t}) - M(r)) \cdot r$$

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

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

By definition of M, we also have

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

Combining the above gives the lemma. □

2.0.1 Additive lemma

Based on the above lemma, in order to show that FRL does well, it would suffice to show that $M(r + c_{1\ldots j-1}) \cdot c_j$ is close to $M(r + c_{1\ldots j}) \cdot c_j$. 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,

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

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 $r_1 + c_{1\ldots j-1} = r_2 + c_{1\ldots j}$, 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,

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

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

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

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

Proof. This follows from the two lemmas and the definitions of min-cost$_t$ and FRL(μ)'s cost$_t$. $\square$

2.0.2 Multiplicative lemma

Like diff, let,

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

Then,

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

Combining this with Lemma 3, we get

Lemma 5.

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

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 $c\in\mathbb{R}^n$,*

$$\mathit{diff}_\mu(c) = \le \frac{|c|_1}{R}.$$

Proof. WLOG we can assume that all the components of c, $c^1, c^2, \ldots, c^n$, 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,

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

$\square$

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

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

Proof. We can think of $\mathrm{diff}_\mu(c)$ 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

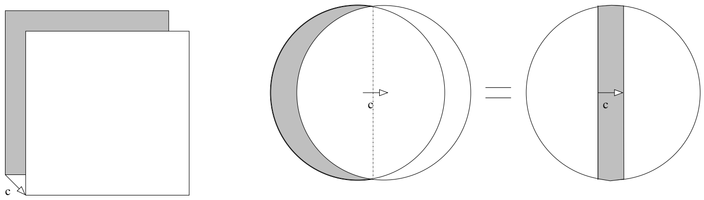

Figure 2: The variation distance between shifted cubes and balls. In the balls, one can see that the two shaded regions have the same area if the nonshaded sections have the same area.

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,

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

Here we have used the fact that

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

($\Gamma(x)$ 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 $c \in \mathbb{R}^n$,*

$$rat_\mu(c) \leq e^{\epsilon|c|}.$$

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 $|c|_1 \leq 1$. 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 and FRL have the same expectation.

To show the high probability bounds, note that our total cost is the sum of t periods, each period being the average of s random variables. Each variable takes a value in an interval of size at most D. By Chernoff bounds, the probability that this sum is larger than its expectation by $\epsilon tD/2$ is at most $e^{-st\epsilon^2/2}$. □

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 $e^{\epsilon/2} \leq 1 + 0.7\epsilon$ for $0 \leq \epsilon \leq 1$, the statement in the theorem holds in expectation. To get the high probability guarantee, we apply the multiplicative Chernoff bound. Our total cost is a sum of s random variables in each period. Each random variable varies in a range of size at most D, and we can take the expected value of their sum to be the right hand side above. This guarantees that with s samples, the cost is at most an additional $1 + 0.1\epsilon$ factor with probability at least $1 - e^{-s\epsilon^2 \Omega(\frac{\log n}{\epsilon} + \frac{mc_t}{D})}$. □

Proof. (of Theorem 3) In general, an easy argument can be used to show that $\text{diff}_\mu(c) \leq \text{rat}_\mu(c) - 1$. This combined with the Lemma 8 shows that for $|c| \leq 1$ and $d\mu(x) \propto e^{-\epsilon|x|/2}$, $\text{diff}_\mu(c) \leq e^{\epsilon/2} - 1 \leq \epsilon$. Also, from Lemma 6, for μ uniform over the cube $\frac{1}{\epsilon}[-1,1]^n$, $\text{diff}_\mu(c) \leq \epsilon/2$. Combining these with Lemmas 1 and Lemma 2, gives what we need. □

3 Approximation algorithms

We have shown 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, the online version can be solved to within $1 + \epsilon$. 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 $x^1, \ldots, x^n$, 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 4. *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:*

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

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

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

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 $D \leq n^2$. □

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

4 Binary search trees

For our purposes, a binary search tree is a binary tree where the nodes are labeled $1, 2, \ldots, n$ 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 $\{1, 2, \ldots, n\}$, 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 $\mathbf{LT}(N)$

1. For $1 \leq i \leq n$, let $a_i := 0$ and choose v_i randomly from $\{1, 2, \ldots, N\}$
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,

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

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 $O(n)$.

5 Predicting from expert advice

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

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

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 cost[2].

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.

6 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.

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

[2] Imagine 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.

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.

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.

7 Appendix: L_2 guarantees

Here we assume $|c_j|_2 \le 1$ and D_2 is an upper bound on the L_2 diameter of S.

Theorem 5. *Let μ be uniform on the ball of radius $\frac{1}{\epsilon}$. Then for FEL(μ, s) (any $s \geq 1$) and FLL(μ),*

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

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

$$E[\textit{FLL's cost}_t] \leq \textit{min-cost}_t + O(n^{1/4})D_2\sqrt{t}$$

With no further assumptions, it is impossible to be constant competitive in a multiplicative sense, e.g. 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 6. *Let $d\mu$ be proportional to $e^{-\epsilon|x|_2/2}$ for $0 \leq \epsilon \leq 1$. Then for FEL(μ, s) (any $s \geq 1$) and FLL(μ),*

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

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