# Online Learning for Global Cost Functions

Eyal Even-Dar
Google Research.
evendar@google.com

Robert Kleinberg
Cornell University
rdk@cs.cornell.edu

Shie Mannor
Technion
shie@ee.technion.ac.il

Yishay Mansour
Tel Aviv University.
mansour@tau.ac.il

**Abstract**

We consider an online learning setting where at each time step the decision maker chooses a distribution over $k$ alternatives, and then observes the loss of each alternative. Motivated by load balancing and job scheduling, we consider a global cost function (a multivariate function of the losses incurred by each alternative), rather than a summation of the instantaneous losses as done traditionally in online learning. Such global cost functions include the makespan (the maximum over the alternatives) and the $L_d$ norm (over the alternatives). Based on approachability theory, we design an algorithm that guarantees vanishing regret for this setting, where the regret is measured with respect to the best static decision that selects the same distribution over alternatives at every time step. For the special case of makespan cost we devise a simple and efficient algorithm. In contrast, we show that for concave global cost functions, such as $L_d$ norms for $d < 1$, the worst-case average regret does not vanish, and that in general for $L_d$ norms with $d > 1$ our bound can not be improved.

## 1 Introduction

We introduce a formulation of online learning with a global cost function. In this class of problems, a decision maker faces an environment with a fixed set of alternatives and tries to minimize a cost function that depends on the cumulative losses incurred by *each* alternative, and not merely on the sum of all losses. We therefore use the term global cost function.

To motivate the discussion, consider a job scheduling problem with jobs that can be distributed to multiple machines and suppose each job incurs a different load per machine. In this instance of the problem, the set of machines corresponds to the set of alternatives in the general formulation of the problem, and the loads correspond to losses. At each stage the decision maker first selects a distribution over machines and only then observes the loads of the machines. The load on each machine accumulates over time, weighted in each stage by the distribution the decision-maker selected. The objective of the decision maker is to minimize the makespan, i.e., the load of the most heavily loaded machine, or equivalently the infinity-norm of the vector of loads. The comparison class the decision maker considers is that of all static allocations (i.e., distributions over machines) and the decision maker wants to minimize the difference between the cost of the load vector and the cost of the best static allocation.

The online learning (or regret minimization) framework has had many successful applications in machine learning, game theory, and beyond; see [9]. In spite of the advantages of this approach it suffers from some inherent limitations. First, the size of the comparison class affects the magnitude of the regret (although only in a logarithmic way). Second, the approach usually assumes that there is essentially no *state* of the system and that the decision problem is the same at every stage (see however [4, 5, 6, 19, 10] for deviations from this standard approach). Finally, there is a tacit assumption that the objective function is additive in the losses over the time steps. This last assumption does not hold in many natural problems such as job scheduling and load balancing settings, for example.

While there are similarities in motivation between online competitive algorithms [8] and the regret minimization setup, the models have some significant differences. First, the comparison class is substantially different. While competitive analysis allows the optimal scheduler to be "arbitrary," regret minimization limits the size of the comparison class. On the other hand, the competitive analysis approach bounds only the *ratio* of the performances, while the regret minimization bounds the *difference* between them. Finally, there is also a very important difference in the information model. In regret minimization the decision maker first selects an alternative (or a distribution over alternatives) and only then observes the losses, while in the standard online scheduling setup the algorithm first observes the loads (losses) and only then selects a machine (alternative).

Our model assumes a finite number of alternatives. We use the standard information model for regret minimization problems: the decision maker first selects a distribution over alternatives and only then observes the losses. We have a static comparison class: the set of all algorithms that use the same distribution over the alternatives in every time step. The class of objective functions we consider is much in the spirit of the job scheduling, and includes objective functions such as makespan or $L_2$-norm. We bound the difference between the performance of our online algorithm and the performance of the best static distribution with respect to a *global* cost function, e.g., the $L_\infty$ norm of the loss vector, which corresponds to makespan. Note that in contrast to many regret minimization works, we consider not only the best alternative, but rather the best distribution over alternatives, so the comparison class in our setup is infinite.[1]

Our contributions include the following:

- We present the online learning model for a global cost function. This model is natural in the online learning setup and it has not been studied to the best of our knowledge.
- We present an algorithm for convex global cost functions (such as makespan, or any $L_d$-norm, for $d > 1$) that ensures the difference between the cost of our online algorithm and that of the best static distribution is bounded by $O(\sqrt{k/T})$, where $T$ is the number of time steps and $k$ is the number of alternatives. Our algorithm only requires that the objective is a convex and Lipschitz function. Our algorithm guaranteeing a vanishing average regret is based on approachability theory, and for both the $L_d$ norm and makespan runs in polynomial time.
- For the important special case of the makespan cost function we provide a specialized simple deterministic online algorithm, which is both computationally efficient and has a regret bound of $O(\log(k)/\sqrt{T})$, where $k$ is the number of different alternatives. Our algorithm is based on a recursive construction where we first solve the case of two alternatives and then provide a recursive construction for any number of alternatives. Our recursive construction maintains the computational efficiency of the two alternative algorithm, and its regret depends only logarithmically on the number of alternatives.
- We complement our algorithms with an impossibility result. We show that one cannot hope to have a similar result for *any* global cost function. Specifically, we show for a wide class of concave functions (including any $L_d$-norm, for $0 < d < 1$) that any online algorithm has $\Omega(1)$ regret. More specifically, we show that the ratio of the online cost to the best distribution cost is bounded away from one. In addition, we show that for any time horizon $T$, there is an $L_d$ norm such that the regret is at least $\Omega(1/\sqrt{T})$.

**Related Work:** Unfortunately, we can not give a comprehensive review of the large body of research on regret minimization algorithm, and we refer the interested reader to an excellent exposition in [9]. Most of the work in regret minimization considers additive (as opposed to global) cost, namely the losses in different time steps are added to define the cumulative loss. We now review some relevant related research. Regret minimization when there is a notion of a state was studied in [11, 20] in the context of Markov decision

[1] It is instructive to discuss the difference between minimizing an additive loss (as in the "best expert" problem) and minimizing the makespan (the $L_\infty$ norm of the loss vector). Consider two alternatives and a sequence of $T$ identical losses of $(2, 1)$. Minimizing the additive loss would always select alternative 2, and have cost $T$. Minimizing the makespan would select a distribution $(1/3, 2/3)$ and would have cost $(2/3)T$.

process. Our problem can also be modeled as a Markov decision process where the state is the loss vector and the costs are additive. The state space in such a model is, however, continuous[2] and is neither unichain nor irreducible. There has been an ongoing interest in extending the basic comparison class for the regret minimization algorithms, for example by introducing *shifting experts* [12] *time selection functions* [7] and *wide range regret* [18]. Still, all those works assume that the loss is additive between time steps.

A different research direction has been to improve the computational complexity of regret minimization algorithms, especially in the case that the comparison class is exponential in size. General computationally efficient transformations were given by [15], where the cost function is linear and the optimization oracle can be computed in polynomial time, and extended by [14], to the case where we are given only an approximate-optimization oracle.

There has been a sequence of works establishing the connection between online competitive algorithms [8] and online learning algorithm [9]. One issue is that online learning algorithms are stateless, while many of the problems addressed in the competitive analysis literature have a state (see [4]). For many problems one can use the online learning algorithms and guarantee a near-optimal static solution, however a straightforward application requires both exponential time and space. Computationally efficient solutions have been given to specific problems, including paging [5], data structures [6], and routing [1, 2].

In contrast to our work, all the above works concentrate on the case where the global cost function is additive between time steps.

## 2 Model and Preliminaries

We consider an online learning setup where a decision maker has a finite set $K = \{1, \ldots, k\}$ of $k$ alternatives to choose from. At each time step $t$, the algorithm $A$ selects a distribution $\alpha_t^A \in \Delta(K)$ over the set of alternatives $K$, where $\Delta(K)$ is the set of distributions over $K$. Following that the adversary selects a vector of losses $\ell_t \in [0,1]^k$. The average loss of alternative $i$ is denoted by $L_T(i) = \frac{1}{T}\sum_{t=1}^{T} \ell_t(i)$. The average loss of the online algorithm $A$ on alternative $i$ is $L_T^A(i) = \frac{1}{T}\sum_{t=1}^{T} \alpha_t^A(i)\ell_t(i)$ and its average loss vector is $L_T^A = (L_T^A(1), \ldots, L_T^A(k))$. Let the loss vector of a static allocation $\alpha \in \Delta(K)$ be $L_T^\alpha = \alpha \odot L_T$ where $x \odot y = (x(1)y(1), \ldots, x(k)y(k))$.

The objective of the online algorithm $A$ is to minimize a *global cost function* $C(L_T^A)$. In traditional online learning the global cost $C$ is assumed to be an additive function, i.e., $C(L_T^A) = \sum_{i=1}^{k} L_T^A(i)$. The main focus of this work is to consider alternative cost functions that arise in many natural problems such as job scheduling. More specifically we consider the following norms: the *makespan*, $C_\infty(L_T^A) = \max_{i=1}^{k} L_T^A(i)$, and the $d$-norm cost, $C_d(L_T^A) = (\sum_{i=1}^{k}(L_T^A(i))^d)^{1/d}$ for $d > 0$. Note that for $d > 1$ the $d$-norm cost is convex while for $d < 1$ it is concave.

Our comparison class is the class of static allocations for $\alpha \in \Delta(K)$. We define the *optimal cost function* $C^*(L_T)$ as the minimum over $\alpha \in \Delta(K)$ of $C(L_T^\alpha)$. We denote by $\alpha_C^*(L_T)$ a minimizing $\alpha \in \Delta(K)$ called the *optimal static allocation*,[3] i.e.,

$$C^*(L_T) = \min_{\alpha \in \Delta(K)} C(L_T^\alpha) = \min_{\alpha \in \Delta(K)} C(\alpha \odot L_T) = C(\alpha_C^*(L_T) \odot L_T).$$

We denote by $C_d^*$ and $C_\infty^*$ the global optimal cost function $C^*$ of the $d$-norm cost and the makespan, respectively.

The regret of algorithm $A$ at time $T$ given the vectors $L_1, L_2, \ldots, L_T$, is defined as:

$$R_T(A) = C(L_T^A) - C^*(L_T).$$

Given a subset of the time steps $\mathcal{B}$ we denote by $L_\mathcal{B} = \frac{1}{T}\sum_{t \in \mathcal{B}} \ell_t$ the contribution of the time steps in $\mathcal{B}$ to the final average loss, and for an algorithm $A$ we denote by $L_\mathcal{B}^A(i) = \frac{1}{T}\sum_{t \in \mathcal{B}} \alpha_t^A(i)\ell_t(i)$ the contribution

[2] Even if one discretizes the Markov decision process, the size will grow with time.

[3] When the minimum is not unique we assume it can be selected according to some predefined order.

of the time steps in $\mathcal{B}$ to its final average loss on alternative $i$. The vector $(L_{\mathcal{B}}^A(1), \ldots, L_{\mathcal{B}}^A(k))$ is denoted by $L_{\mathcal{B}}^A$.

We will assume that the cost function, $C$, is a convex function while $C^*$ is a concave function, and that both of them are Lipschitz continuous (the only exception is Section5.2). The following lemma is immediate from the definition of the norms.

**Lemma 2.1.** *The d-norm global cost function $C_d$, for $d > 1$, and the makespan global cost function $C_\infty$ are both convex and Lipschitz functions.*

The challenging part is to show that the optimal cost function $C^*$ is a concave function. In Appendix A we prove this for $d$-norm (Lemma A.2) and the makespan (Lemma A.4).

**Lemma 2.2.** *The d-norm global optimal cost function $C_d^*$, for $d > 1$, and the makespan global optimal cost function $C_\infty^*$ are both concave and Lipschitz functions.*

See Lemmas A.1 and A.3 in the appendix for a derivation of the optimal cost function and optimal static allocation for both the $d$-norm and the makespan.

**Remark:** Our model is essentially fractional, where the decision maker selects a distribution, and each alternative receives exactly its fraction according to the distribution. In settings where fractional decisions are disallowed or are not meaningful, one can modify the algorithm so that it randomly samples a single action from its distribution in each period. The following lemma attests to the fact that the extra loss incurred by this random sampling is small.

**Lemma 2.3.** *Consider an algorithm $A$ whose regret in the fractional model is $R_T$. If we modify $A$ by randomly sampling a single action from its distribution in each period, the expected regret of the modified algorithm is bounded by $R_T + 19\Lambda/\sqrt{T}$, where $\Lambda$ denotes the Lipschitz constant of the cost function $C$.*

*Proof.* Denote the modified algorithm by $MA$. Let $\alpha_t$ denote the distribution selected by algorithm $A$ at time $t$, and let $\widehat{\alpha}_t$ denote the distribution that assigns all of its probability mass to the action sampled by $MA$ at time $t$. The random vector $\widehat{\alpha}_t - \alpha_t$ has expectation zero, conditional on the algorithm's state at the start of round $t$. Consequently the random vector $(\widehat{\alpha}_t - \alpha_t) \odot \ell_t$ also has conditional expectation zero, and the sequence,

$$X_t = \sum_{s=1}^{t} (\widehat{\alpha}_t - \alpha_t) \odot \ell_t \ ,$$

is a vector-valued martingale. This martingale has bounded differences, since the 2-norm of the vector $X_t - X_{t-1} = (\widehat{\alpha}_t - \alpha_t) \odot \ell_t$ is bounded above by 1. Using the Kallenberg-Sztencel concentration inequality [16], as strengthened by Hayes [13], we may conclude that for all $a > 0$,

$$\Pr\left(\|X_T\|_2 \geq a\right) < 2e^2 e^{-a^2/2T},$$

and consequently,

$$\mathbb{E}\left[\|X_T\|_2\right] < 2e^2 \int_0^\infty e^{-a^2/2T}\, da = e^2\sqrt{2\pi T} < 19\sqrt{T}. \tag{1}$$

Note that $L_T^{MA} = L_T^A + \frac{1}{T} X_T$. Thus, the amount by which the regret of $MA$ exceeds that of $A$ can be expressed as $C(L_T^{MA}) - C(L_T^A)$ and is bounded above by $\frac{\Lambda}{T}\|X_T\|_2$. The lemma now follows by applying (1). □

# 3 Low regret based on approachability theory

In this section we use Blackwell's approachability theory to construct a strategy that has zero asymptotic regret. Approachability theory concerns two-player games with vector-valued rather than scalar-valued payoffs. It provides conditions under which a player of such a game can force the average payoff vector to converge to a specified set, in repeated play. In this section we review the basics of approachability, deferring proofs to Appendix B, and then we apply the theory to the problem of regret minimization with global cost functions.

## 3.1 Review of approachability theory

Our treatment of approachability theory parallels Blackwell's original paper [3] and the treatment given in textbooks, e.g., [9]. The version of Blackwell's Theorem that we present here is designed to emphasize explicit convergence rates, and our definition of a game with vector payoffs is slightly more general than the standard one because we allow the players' mixed strategy sets to be arbitrary compact, convex sets rather than requiring them to be simplices. This generalization, which does not materially change the proof of Blackwell's Theorem, is required because the adversary's strategy set in our setting is a hypercube rather than a simplex.

To state the theorem, suppose that $A \subseteq \mathbb{R}^a$, $B \subseteq \mathbb{R}^b$ are compact, convex sets and that $m : \mathbb{R}^a \times \mathbb{R}^b \to \mathbb{R}^c$ is a vector-valued bilinear function; one may interpret $m$ as the payoff function of a game with strategy sets $A$ and $B$. Let $V$ be any closed, convex subset of $\mathbb{R}^c$; $V$ is interpreted as a "target set," and one player's goal is to guide the average payoff vector into the target set when repeatedly playing the game against an adversary. We will use $\|x\|$ to denote the 2-norm of a vector $x$, and we will use $\text{dist}(x, V)$ to denote the Euclidean distance, from $x$ to $V$, i.e., $\text{dist}(x, V) = \inf_{y \in V} \|x - y\|$.

**Definition 3.1.** *We say that $V$ is* approachable with convergence rate $r(t)$ *if there exists an algorithm for choosing a sequence of vectors $a_1, a_2, \ldots \in A$ such that for every sequence $b_1, b_2, \ldots \in B$ and all integers $t \geq 1$:*

1. *the value of $a_t$ depends only on the values of $m(a_s, b_s)$ for $s < t$;*
2. *the average payoff $\widehat{m}_t = \frac{1}{t} \sum_{s=1}^{t} m(a_s, b_s)$ satisfies $\text{dist}(\widehat{m}_t, V) \leq r(t)$.*

**Definition 3.2.** *We say that $V$ satisfies the* Blackwell criterion *if $V$ is closed and convex and for every $b \in B$ there exists $a \in A$ such that $m(a, b) \in V$.*

**Theorem 3.3** (Blackwell,1956)**.** *Given compact, convex sets $A \subseteq \mathbb{R}^a$, $B \subseteq \mathbb{R}^b$ and a bilinear function $m : \mathbb{R}^a \times \mathbb{R}^b \to \mathbb{R}^c$, if $V$ satisfies the Blackwell criterion then $V$ is approachable with convergence rate $r(t) = 2D/\sqrt{t}$, where $D = \sup_{x,y \in m(A \times B)} \|x - y\|$ is the diameter of the set $m(A \times B)$.*

We provide a proof of Theorem 3.3 in Appendix B.

## 3.2 Application to regret minimization with global costs

To apply Blackwell's Approachability Theorem to online learning with global cost functions, we begin by specifying a game with vector payoffs and a target set. We then prove that the target set satisfies the Blackwell criterion, and we invoke Theorem 3.3 to deduce that it is approachable and to obtain a bound on the convergence rate. We conclude the section by explicitly describing a regret minimizing strategy based on approachability theory.

Consider a repeated game against Nature where the decision maker chooses at every stage an action $\alpha \in \Delta(K)$ and Nature chooses an action $\ell \in [0, 1]^k$. The vector-valued reward obtained by the decision maker is a $2k$-dimensional vector $m(\alpha, \ell)$ whose first $k$ coordinates are $\alpha \odot \ell$ and the last $k$ coordinates are $\ell$. Let us denote this immediate vector-valued reward by $m_t = m(\alpha_t, \ell_t) = (\alpha_t \odot \ell_t, \ell_t)$. It easily follows that the average reward $\widehat{m}_T = (\sum_{t=1}^{T} m_t)/T$ is equal to $(L_T^A, L_T)$. We are now ready to define the target set in this game,

$$S = \{(x, y) \in \mathbb{R}^k \times \mathbb{R}^k : x, y \geq 0; C(x) \leq C^*(y)\}. \tag{2}$$

Note that $S$ is a set in $\mathbb{R}^{2k}$. We will show that $S$ satisfies the Blackwell criterion. We first claim that $S$ is convex.

**Lemma 3.4.** *Suppose that $C$ is convex and $C^*$ is concave. Then $S$ is convex.*

*Proof.* Suppose that $(x^i, y^i) \in S$ for $i = 1, 2$ and fix $\beta \in [0, 1]$ (we let $x^i$ and $y^i$ denote vectors in $\mathbb{R}^k$). Since $C$ is convex we have that,

$$C(\beta x^1 + (1 - \beta)x^2) \leq \beta C(x^1) + (1 - \beta)C(x^2).$$

Since $(x^i, y^i) \in S$ for $i = 1, 2$ we have that $C(x^i) \leq C^*(y^i)$ for $i = 1, 2$, and hence

$$\beta C(x^1) + (1-\beta)C(x^2) \leq \beta C^*(y^1) + (1-\beta)C^*(y^2).$$

Using the fact that $C^*$ is concave we obtain that,

$$\beta C^*(y^1) + (1-\beta)C^*(y^2) \leq C^*(\beta y^1 + (1-\beta)y^2).$$

Combining the above inequalities we conclude that,

$$C(\beta x^1 + (1-\beta)x^2) \leq C^*(\beta y^1 + (1-\beta)y^2),$$

completing the proof. □

**Proposition 3.5.** *The set $S$ is approachable, with convergence rate $\sqrt{8k/t}$.*

*Proof.* It is obvious from the definition of $S$ that it is a closed set, and Lemma 3.4 established that it is convex. To finish verifying that $S$ satisfies Blackwell's criterion, we must prove that for every $\ell \in [0,1]^k$ there exists a $\alpha \in \Delta(K)$ such that $m(\alpha, \ell) \in S$. This is easy: let $\alpha = \alpha^*_C(\ell)$. Then $m(\alpha, \ell) = (\alpha^*_C(\ell) \odot \ell, \ell)$ belongs to $S$, by the definition of the function $\alpha^*_C$.

The bound on the convergence rate follows from the convergence rate $2D/\sqrt{t}$ given in Theorem 3.3; we merely need to calculate an upper bound on $D$, the diameter of the set $m(\Delta(K) \times [0,1]^k)$. An easy way to do this is to observe that the set is contained in the hypercube $[0,1]^{2k}$, whose diameter is $\sqrt{2k}$. □

Since $S$ is approachable, let us now explain how to approach it in a constructive way, using the algorithm specified in the proof of Theorem 3.3 in Appendix B. Given the current vector valued reward $\widehat{m}_t = (L^A_t, L_t)$ we first compute the point $s_t \in S$ that is closest to $\widehat{m}_t$. For that, we need to solve the following convex optimization problem:

$$\begin{aligned} \min_{x,y \in \mathbb{R}^k} & \|(L^A_t, L_t) - (x, y)\|_2 \\ \text{s.t. } & C(x) \leq C^*(y) \\ & x \geq 0. \end{aligned} \tag{3}$$

According to Blackwell's approaching strategy (see Appendix B), if the solution to (3) is 0, i.e., $\widehat{m}_t \in S$, the decision maker can act arbitrarily at time $t$. If not, then let $u = \widehat{m}_t - (x, y)$ where $(x, y)$ are the minimizers of (3), and let us write $u = (u_\alpha, u_\ell)$ to denote the two $k$-dimensional components of $u$. We are looking for an action $\alpha \in \Delta(K)$ that guarantees that the expected reward at time $t+1$ is on the other side of the supporting hyperplane at $(x, y)$; in other words $\alpha$ must satisfy

$$\sup_{\ell \in [0,1]^k} m(\alpha, \ell) \cdot u \leq \sup_{s \in S} s \cdot u. \tag{4}$$

We know that such a mixed action exists since $S$ satisfies the Blackwell criterion. (See the proof of Theorem 3.3.) The optimization problem we try to solve is therefore:

$$\min_{\alpha \in \Delta(K)} \max_{\ell \in [0,1]^k} u_\alpha^\top (\alpha \odot \ell) + u_\ell^\top \ell.$$

The objective function is bilinear in $(\alpha, \ell)$, so the inner maximization over $\ell$ is always accomplished by choosing one of the extreme points, i.e., a vector $\ell \in \{0,1\}^k$. In fact, it is evident that the vector $\ell \in \{0,1\}^k$ that maximizes the objective function is defined by $\ell(i) = 1$ if $u_\alpha(i)\alpha(i) + u_\ell(i) > 0$ and otherwise $\ell(i) = 0$. Hence

$$\max_{\ell \in [0,1]^k} u_\alpha^\top (\alpha \odot \ell) + u_\ell^\top \ell = \sum_{i=1}^k \max\{0, u_\alpha(i)\alpha(i) + u_\ell(i)\}.$$

The problem of choosing $\alpha$ has thus been reduced to the optimization problem

$$\min_{\alpha \in \Delta(K)} \sum_{i=1}^{k} \max\{0, u_\alpha(i)\alpha(i) + u_\ell(i)\}$$

which is equivalent to

$$\begin{aligned} \min \ & \sum_{i=1}^{k} \beta(i) \\ \text{s.t.} \quad & \beta(i) \geq u_\alpha(i)\alpha(i) + u_\ell(i) \quad \forall i \\ & \sum_{i=1}^{k} \alpha(i) = 1 \\ & \beta(i), \alpha(i) \geq 0 \quad \forall i, \end{aligned} \tag{5}$$

a linear program of size $O(k)$. We summarize the above discussion in the following Theorem.

**Theorem 3.6.** *The following strategy guarantees that regret converges to 0.*

1. *At every stage $t$ solve (3).*
2. *If the solution is 0, play an arbitrary action.*
3. *If the solution is more than 0, compute $(u_\alpha, u_\ell) = \widehat{m}_t - (x, y)$ where $(x, y)$ are the minimizers of (3) and solve (5). Play a maximizing $\alpha$.*

*If $\Lambda$ is an upper bound on the Lipschitz constant of both $C$ and $C^*$, then the algorithm's regret is bounded above by $4\Lambda\sqrt{k/T}$. Moreover, the running time is polynomial, assuming that the convex program* (3) *can be solved in polynomial time.*

*Proof.* The foregoing discussion establishes that the algorithm is an approaching strategy to $S$, and that in fact its average payoff vector $\widehat{m}_T = (L_T^A, L_T)$ satisfies $\text{dist}(\widehat{m}_T, S) \leq \sqrt{8k/T}$. The only missing part is to argue that approaching $S$ implies minimizing the regret.

Let $(x, y) \in S$ be such that $\|\widehat{m}_T - (x, y)\| \leq \sqrt{8k/T}$. The algorithm's regret is $C(L_T^A) - C^*(L_T)$, and using the fact that $C(x) - C^*(y) \leq 0$ (the definition of $S$) we obtain the following regret bound:

$$\begin{aligned} C(L_T^A) - C^*(L_T) &= [C(x) - C^*(y)] \ + \ [C(L_T^A) - C(x)] \ + \ [C^*(y) - C^*(L_T)] \\ &\leq \Lambda(\|L_T^A - x\| + \|L_T - y\|) \\ &\leq \Lambda\sqrt{2\|L_T^A - x\|^2 + 2\|L_T - y\|^2} \\ &= \Lambda\sqrt{2\|\widehat{m}_T - (x, y)\|^2} \leq 4\Lambda\sqrt{k/T}. \end{aligned}$$

The bound on the algorithm's running time is a direct consequence of the fact that (5) is a linear program of polynomial size. □

We note that while in principle there may be a dependence between $k$ and the Lipschitz constant of $C$ and $C^*$, for the case of $d$-norms where $d > 1$ (including $d = \infty$) it is not hard to show that the Lipschitz constant of both $C$ and $C^*$ is bounded by 1. This implies that the convergence rate is $O(\sqrt{k/T})$ for $d$-norms.

# 4 Algorithms for Makespan

In this section we present algorithms for the makespan cost function. We propose algorithms that are simple to implement, computationally efficient, and have an improved regret bound. For $k$ alternatives the average regret vanishes at the rate of $O((\log k)T^{-1/2})$. Throughout the section, we will use the terms "loss" and "load" interchangeably.

The construction of the algorithms is done in two parts. The first part is to show an algorithm for the basic case of two alternatives (Section 4.1). The second part is a recursive construction, that builds an algorithm for $2^r$ alternatives from two algorithms for $2^{r-1}$ alternatives, and one algorithm for two alternatives. The recursive construction essentially builds a complete binary tree, and the main issue in the construction is to define the losses that each algorithm observes (Section 4.2).

For the analysis of both parts of the construction, we will make use of a simple but powerful lemma asserting that given any partition of the losses to subsets, if we bound the regret in each subset, then the sum of the subsets' regrets will bound the regret of the online algorithm.

**Lemma 4.1.** *Suppose that $C$ is convex scale-invariant[4] and $C^*$ is concave scale-invariant. Let $\mathcal{T}_i$ be a partition of the $T$ time steps to $m$ sets, and let $ON$ be an online algorithm such that for each $\mathcal{T}_i$ we have $C(L^{ON}_{\mathcal{T}_i}) - C^*(L_{\mathcal{T}_i}) \leq R_i$. Then,*

$$C(L^{ON}_T) - C^*(L_T) \leq \sum_{i=1}^{m} R_i.$$

*Proof.* Our assumption that $C$ is scale-invariant implies that for every scalar $\lambda \geq 0$, $C(\lambda x) = \lambda C(x)$ and similarly, since $C^*$ is scale-invariant, we have $C^*(\lambda x) = \lambda C^*(x)$. The inequality $C^*(x+y) \geq C^*(x) + C^*(y)$ now follows by an easy calculation:

$$C^*(x+y) = 2C^*\left(\frac{x+y}{2}\right) \geq 2\left[\frac{1}{2}C^*(x) + \frac{1}{2}C^*(y)\right] = C^*(x) + C^*(y). \tag{6}$$

Similarly, the inequality $C(x+y) \leq C(x) + C(y)$ follows from,

$$C(x+y) = 2C\left(\frac{x+y}{2}\right) \leq 2\left[\frac{1}{2}C(x) + \frac{1}{2}C(y)\right] = C(x) + C(y). \tag{7}$$

From the assumption in the theorem we have that $\sum_{i=1}^m C(L^{ON}_{\mathcal{T}_i}) - \sum_{i=1}^m C^*(L_{\mathcal{T}_i}) \leq \sum_{i=1}^m R_i$. Since $\mathcal{T}_i$ is a partition, $L_T = \sum_{i=1}^m L_{\mathcal{T}_i}$ and $L^{ON}_T = \sum_{i=1}^m L^{ON}_{\mathcal{T}_i}$. From (6) we have, $C^*(L_T) \geq \sum_{i=1}^m C^*(L_{\mathcal{T}_i})$, and from (7) we have, $C(L^{ON}_T) \leq \sum_{i=1}^m C(L^{ON}_{\mathcal{T}_i})$. Combining the three inequalities proves the theorem. □

The importance of Lemma 4.1 is the following. If we are able to partition the time steps, in retrospect, and bound the regret in each part of the partition, then the above theorem states that the overall regret is bounded by the sum of the local regrets on each piece of the partition.

## 4.1 Two Alternatives

We derive a simple algorithm for the case of two alternatives (machines) and the makespan metric. Before introducing the algorithm formally we would like to motivate its design. The static optimum balances the loads on both machine (see Lemma A.3), so it makes sense that the online algorithm would try to balance the loads on the machines as well. The online algorithm can achieve this by shifting weight to the less loaded machine. More concretely, at time $t$ let $\Lambda_t$ be the difference between the increase of the load on machine 2 and machine 1. (Note we are considering the load derived by the online algorithm, so this measure is an algorithm dependent one.) Then, at time $t+1$ the weights of the machines are updated proportional to $\Lambda_t$, with a constant learning rate. This will imply that the weight of a machine is proportional to the difference in the loads (given the current online algorithm assignment). This will be the essence of our online algorithm. We can now formally introduce the algorithm, which we call DIFF.

[4] A function $f$ is scale invariant, if for any $\lambda \geq 0$ we have $f(\lambda x) = \lambda f(x)$.

**The DIFF Algorithm:**
At time $t = 1$ we have $p_1(1) = p_1(2) = 1/2$.
At time $t \geq 2$ we have:
$p_{t+1}(1) = p_t(1) + \frac{p_t(2)\ell_t(2) - p_t(1)\ell_t(1)}{\sqrt{T}}$, and $p_t(2) = 1 - p_t(1)$.

Figure 1: Algorithm DIFF for two alternatives.

**The DIFF Algorithm:** We denote the distribution of DIFF at time $t$ by $(p_t(1), p_t(2))$. Algorithm DIFF starts with $p_1(1) = p_1(2) = 1/2$, and the update of the distribution of DIFF is done as follows,

$$p_{t+1}(1) = p_t(1) + \frac{p_t(2)\ell_t(2) - p_t(1)\ell_t(1)}{\sqrt{T}},$$

and by definition $p_t(2) = 1 - p_t(1)$.

Let $L_T^D$ be the loads generated by DIFF on the alternatives, i.e., $L_T^D(i) = \sum_{t=1}^T \ell_t(i)p_t(i)$, and let $\Lambda_t = L_t^D(2) - L_t^D(1)$ be the difference in the loads of the machine at time $t$ due to algorithm DIFF. The first step of the proof is to show that the algorithm DIFF derives legitimate weights, i.e., $p_t(i) \in [0, 1]$.

**Lemma 4.2.** *For any loss sequence $\ell$ we have $p_t(i) \in (0, 1)$ for $i \in \{1, 2\}$, and $|\Lambda_t| \leq \sqrt{T}/2$*

*Proof.* Consider the update in the weight of alternative 1 at time $t$, i.e., $p_t(1) = p_{t-1}(1) - \lambda_{t-1}/\sqrt{T}$, where $\lambda_{t-1} = p_{t-1}(2)\ell_{t-1}t(2) - p_{t-1}(1)\ell_{t-1}(1)$. Clearly, $\lambda_{t-1} \geq -p_{t-1}(1)$ and hence $p_t(1) \geq p_{t-1}(1)(1 - \frac{1}{\sqrt{T}})$. Therefore, $p_1(1) > 0$. Similarly, $p_t(2) > 0$. Since $p_t(1) + p_t(2) = 1$ then $p_t(1), p_t(2) \in (0, 1)$.

To bound $\Lambda_t$ note that

$$p_t(1) = (1/2) + \frac{\sum_{s=1}^t \lambda_s}{\sqrt{T}} = (1/2) + \frac{\Lambda_t}{\sqrt{T}}.$$

Since $p_t(1) \in (0, 1)$ then $|\Lambda_t| \leq \sqrt{T}/2$. □

The proof of the algorithm is rather technical, but the main idea of the proof is rather simple. Think of the weight of alternative 1 as a walk on the interval $[0, 1]$. The walk starts at $1/2$ and lets assume that it also ends there (this will imply that we end up with perfectly balanced loads). Now segment the interval $[0, 1]$ to $O(\sqrt{T})$ equal size sub-intervals. For simplicity, assume that when we make an update, from $p_t(1)$ to $p_{t+1}(1)$ both are in some sub-interval (maybe reaching the border point of that sub-interval). Now we can present the main idea of the proof. Consider all the updates that are done in a given sub-interval. First, all the updates have fairly similar weight, since they are in the same sub-interval. Second, The sum of the loads incurred on both machines in those time steps is *identical*. Now consider the static optimum only for those updates. We will show that it is also in that sub-interval. To complete the proof we partition the time steps using the sub-intervals. For each such subset of time steps, we have the the average regret bounded by $O(1/\sqrt{T})$, and hence by Lemma 4.1 the regret is bounded by $O(\sqrt{T})$.

The main idea of the proof would be to consider a run of the DIFF algorithm, and transform it to a run, we call "well-behaved." We will show that the cost of DIFF is identical in both runs and the cost of the static optimum cost increases only slightly. On well-behaved runs we prove the regret bound.

Let us first introduce the notion of a run and a well behaved run.

**Definition 4.3.** *A* run *$R$ of length $T$ is a sequence of pairs $(\ell_t, p_t)$, where $\ell_t \in [0, 1]^k$ and $p_t \in \Delta(k)$ for $1 \leq t \leq T$. Given a run $R$ we define $L_T^R(i) = \sum_{t=1}^T \ell_t(i)p_t(i)$, and the online cost of a run $R$ is $C(L_T^R)$. We also define the loads of a run $R$ as $L_T^{R,s}(i) = \sum_{t=1}^T \ell_t(i)$, and the optimum static cost of a run $R$ is $C^*(L_T^{R,s})$.*

Given a lost sequence $\ell$ the algorithm DIFF generates a run $R^D$ in a natural way, i.e., $(\ell_t, p_t)$. We would like to transform the run $R^D$ to a well behaved run $\widehat{R}^D$. We define a well behaved run as follows.

**Definition 4.4.** *A run $R$ of length $T$ is well behaved if:*

1. *At the end of the run $R$, the loads on both alternatives are identical, i.e., $L_T^R(1) = L_T^R(2)$.*
2. *Let $Z_i = 10i$ for $-0.1\sqrt{T} \leq i \leq 0.1\sqrt{T}$ and let $\Lambda_t^R = L_t^R(2) - L_t^R(1)$. Then we require that for each $t$ there exists an $i$ such that both $\Lambda_t$ and $\Lambda_{t+1}$ are in $[Z_i, Z_{i+1}]$. (Note that we can have $\Lambda_t \in (Z_i, Z_{i+1}]$, $\Lambda_{t+1} = Z_i$, and then have, $\Lambda_{t+2}$ in $[Z_{i-1}, Z_i)$.)*
3. *For each $t$ we have $|\frac{1}{2} + \frac{\Lambda_t}{\sqrt{T}} - p_t(1)| \leq \frac{1}{\sqrt{T}}$.*

The following lemma shows how to map a run to a well behaved run.

**Lemma 4.5.** *Every run $R^D$ of length $T$ can be mapped to a well behaved run $\widehat{R}^D$ of length $\widehat{T}$ such that,*

1. *The length of the run does not change significantly, i.e., $T \leq \widehat{T} \leq 2T + 2\sqrt{T}$.*
2. *The online cost of both runs is identical, i.e., $C_\infty(L_T^R) = C_\infty(L_{\widehat{T}}^{\widehat{R}})$.*
3. *The static optimum increases by at most $\sqrt{T}$, i.e., $C_\infty^*(L_{\widehat{T}}^{\widehat{R},s}) \leq C_\infty^*(L_T^{R,s}) + \sqrt{T}$.*

*Proof.* Given the run $R$ we will modify it incrementally to $\widehat{R}$. First we modify $R$ to $\widehat{R}_0$ by adding losses at the end so that the machines end with identical loads. W.L.O.G., assume that $p_T(1) > 1/2$. We add losses $\ell_\tau = (1, 0)$ and update the probabilities $p_\tau$ using DIFF. Since we are using DIFF to compute $p_\tau$ we have that $p_\tau(1) = \frac{1}{2} + \frac{\Lambda_\tau}{\sqrt{T}}$. Note that when the alternatives are balanced we have $p_\tau(1) = 1/2$. Since initially the difference is at most $\sqrt{T}/2$ (Lemma 4.2), and $p_\tau(1) > 1/2$ until we complete, we need to add at most $\sqrt{T}$ time steps. (In the last time step we might have $\ell_\tau = (\alpha, 0)$ for some $\alpha \in [0, 1]$.) Note that for $\widehat{R}_0$ we have $C_\infty(L_T^R) = C_\infty(L_{\widehat{T}}^{\widehat{R}_0})$ since the additional losses did not increase the online makespan. Also, since we added at most $\sqrt{T}$ time steps then $C_\infty^*(L_{\widehat{T}}^{\widehat{R}_0,s}) \leq C_\infty^*(L_T^{R,s}) + T^{1/2}$.

Next we will concentrate on the requirements on $\Lambda_t$. Consider $t$ to be the first time in $\widehat{R}_j$ in which the requirement is violated. We replace $t$ by two time steps in $\widehat{R}_{j+1}$. In the first time step we have $(\gamma\ell_t, p_t)$ and in the following time step we have $((1-\gamma)\ell_t, p_t)$, for some $\gamma \in (0, 1)$. Clearly, the new run $\widehat{R}_{j+1}$ will have $L_T^{\widehat{R}_{j+1}} = L_{\widehat{T}}^{\widehat{R}_j}$, $L_T^{\widehat{R}_{j+1},s} = L_{\widehat{T}}^{\widehat{R}_j,s}$ and $|\frac{1}{2} + \frac{\Lambda_t}{\sqrt{T}}| \leq \frac{1}{\sqrt{T}}$. Now we need to show that there is such a $\gamma$. Assume that $\Lambda_{t-1} < Z_i < \Lambda_t$. The $Z_i = \gamma\Lambda_t + (1-\gamma)\Lambda_{t-1}$, for some $\gamma \in [0, 1]$. Since the only difference between $\Lambda_t$ and $\Lambda_{t-1}$ is in time step $t$, scaling the losses of $\ell_t$ to $\gamma\ell_t$ will achieve the desired effect.

Denote by $\widehat{R}$ the final run and $\widehat{T}$ its length. We have that $T \leq \widehat{T} \leq 2T + 2\sqrt{T}$ since in the transformation to $\widehat{R}_0$ we added at most $\sqrt{T}$ time steps, and in the second part we at most double the number of time steps. In the second part we do not modify either $L_{\widehat{T}_0}^{\widehat{R}_0}$ or $L_{\widehat{T}_0}^{\widehat{R}_0,s}$, so the online cost and the static optimum cost are not modified. □

Next we need to show that for "well behaved" runs we have low regret. We first partition the time steps to subsets, depending on $\Lambda_t$. If $\Lambda_{t-1}, \Lambda_t \in [Z_i, Z_{i+1}]$ then we assign time step $t$ to the set $\mathcal{T}_i$. This is clearly a partition of the time steps. Given the losses at time steps $\mathcal{T}_i$, we denote static optimal allocation for $\mathcal{T}_i$ by $(q_i^*, 1 - q_i^*)$.

**Lemma 4.6.** *Let $z_i = \frac{1}{2} + \frac{Z_i}{\sqrt{T}}$. Assuming that $\mathcal{T}_i \neq \emptyset$, then, $|q_i^* - z_i| \leq \frac{12}{\sqrt{T}}$.*

*Proof.* For the updates in $\mathcal{T}_i$ we have that $\sum_{t \in \mathcal{T}_i} \lambda_t = 0$, where $\lambda_t = p_t(2)\ell_t(2) - p_t(1)\ell_t(1)$. Consider the function,

$$\begin{aligned} f(x) &= \sum_{t \in \mathcal{T}_i} x\, p_t(1) - \sum_{t \in \mathcal{T}_i} (1-x) p_t(2) \\ &= x(L_{\mathcal{T}_i}(1) + L_{\mathcal{T}_i}(2)) - L_{\mathcal{T}_i}(2), \end{aligned}$$

**The MULTI Algorithm $A^K_{2^r}$:**
*Alternative set:* $K$ of size $2^r$ partitioned to $I$ and $J$, $|I| = |J| = 2^{r-1}$.
*Procedures:* $A_2, A^I_{2^{r-1}}, A^J_{2^{r-1}}$.
*Action selection:*
Algorithm $A^I_{2^{r-1}}$ gives $x_t \in \Delta(I)$.
Algorithm $A^J_{2^{r-1}}$ gives $x_t \in \Delta(J)$.
Algorithm $A_2$ gives $(z_t, 1 - z_t)$.
Output $\alpha_t \in \Delta(I \cup J)$ where,
for $i \in I$ set $\alpha_t(i) = x_t(i)z_t$, and
for $j \in J$ set $\alpha_t(j) = y_t(j)(1 - z_t)$.
*Loss distribution:*
Given the loss $\ell_t$.
Algorithm $A^I_{2^{r-1}}$ receives $\ell_t(I)$.
Algorithm $A^J_{2^{r-1}}$ receives $\ell_t(J)$.
Algorithm $A_2$ receives $(\omega_t(I), \omega_t(J))$,
where $\omega_t(I) = 2^{-r+1} \sum_{i \in I} \ell_t(i)x_t(i)$,
and $\omega_t(J) = 2^{-r+1} \sum_{j \in J} \ell_t(j)y_t(j)$.

Figure 2: Algorithm for multiple alternatives.

where $L_{\mathcal{T}} = \sum_{t \in \mathcal{T}} \ell_t$.

First, for $q^*_i$ we have that $f(q^*_i) = 0$. Second, since for any $t \in \mathcal{T}_i$ we have that $\Lambda_t \in [Z_i, Z_{i+1}]$. This implies that $p_t(1) \geq z_i - 1/\sqrt{T}$ and $p_t(1) \leq z_{i+1} + 1/\sqrt{T}$. Therefore $f(z_i - 1/\sqrt{T}) < 0$ and $f(z_{i+1} + 1/\sqrt{T}) > 0$. Hence $q^*_i \in [z_i - 1/\sqrt{T}, z_{i+1} + 1/\sqrt{T}]$. The lemma follows since $z_{i+1} - z_i = 10/\sqrt{T}$. □

**Theorem 4.7.** *For any loss sequence $\ell$, we have that*

$$\textit{Regret} = C_\infty(L^D) - C^*_\infty(L) \leq \frac{48}{\sqrt{T}}.$$

*Proof.* Given $R$ the run of DIFF on $\ell$, using Lemma 4.5 we map it to a well behaved run $\widehat{R}$, and this increases the regret by at most $\sqrt{T}$. Given $\widehat{R}$ we partition the time steps to $\mathcal{T}_i$. For each $\mathcal{T}_i$ the regret is at most $|\mathcal{T}_i|\frac{12}{\sqrt{T}}$. By Lemma 4.1 we can sum of the $\mathcal{T}_i$ and get that the regret is bounded by $\sum_i |\mathcal{T}_i|\frac{12}{\sqrt{T}} \leq (2T + 2\sqrt{T})\frac{12}{\sqrt{T}} \leq 48\sqrt{T}$. □

## 4.2 Multiple alternatives

In this section we show how to transform a low regret algorithm for two alternatives to one that can handle multiple alternatives. For the base case we assume that we are given an online algorithm $A_2$, such that for any loss sequence $\ell_1, \ldots, \ell_T$ over two alternatives, guarantees that its average regret is at most $\alpha/\sqrt{T}$. Using $A_2$ we will build a sequence of algorithms $A_{2^r}$, such that for any loss sequence $\ell_1, \ldots, \ell_T$ over $2^r$ alternatives, $A_{2^r}$ guarantees that its average regret is at most $O(\alpha\, r/\sqrt{T})$.

Our basic step in the construction is to build an algorithm $A_{2^r}$ using two instances of $A_{2^{r-1}}$ and one instance of $A_2$. Algorithm $A_{2^r}$ would work as follows. We partition the set of actions $K$ to two subsets of size $2^{r-1}$, denoted by $I$ and $J$; for each subset we create a copy of $A_{2^{r-1}}$ which we denote by $A^I_{2^{r-1}}$ and $A^J_{2^{r-1}}$. The third instance $A^M_2$ would receive as an input the average loss of $A^I_{2^{r-1}}$ and $A^J_{2^{r-1}}$. Let us be more precise about the construction of $A_{2^r}$.

At each time step $t$, $A_{2^r}$ receives a distribution $x_t \in \Delta(I)$ from $A^I_{2^{r-1}}$, a distribution $y_t \in \Delta(J)$ from $A^J_{2^{r-1}}$ and probability $z_t \in [0, 1]$ from $A^M_2$. The distribution of $A_{2^r}$ at time $t$, $\alpha_t \in \Delta(I \cup J)$, is defined as

follows. For actions $i \in I$ we set $\alpha_t(i) = x_t(i)z_t$, and for actions $j \in J$ we set $\alpha_t(j) = y_t(j)(1 - z_t)$. Given the action $\alpha_t$, $A_{2^r}$ observes a loss vector $\ell_t$. It then provides $A^I_{2^{r-1}}$ with the loss vector $(\ell_t(i))_{i\in I}$, $A^J_{2^{r-1}}$ with the loss vector $(\ell_t(j))_{j\in J}$, and $A^M_2$ with the loss vector $(\omega_t(I), \omega_t(J))$, where $\omega_t(I) = 2^{-r+1}\sum_{i\in I}\ell_t(i)x_t(i)$ and $\omega_t(J) = 2^{-r+1}\sum_{j\in J}\ell_t(j)y_t(j)$.

The tricky part in the analysis is to relate the sequences $x_t, y_t$ and $z_t$ to the actual aggregate loss of the individual alternatives. We will do this in two steps. First we will bound the difference between the average loss on the alternatives, and the optimal solution. Then we will show that the losses on the various alternatives are approximately balanced, bounding the difference between the maximum and the minimum loss.

The input to algorithm $A^M_2$ on its first alternative (generated by $A^I_{2^{r-1}}$) has an average loss

$$W(I) \quad = \quad \frac{1}{T}\sum_t \omega_t(I) = \frac{1}{T}\left[\sum_t 2^{-r+1}\sum_{i\in I} x_t(i)\ell_t(i)\right].$$

Similarly, on the second alternative it has an average loss

$$W(J) \quad = \quad \frac{1}{T}\sum_t \omega_t(J) = \frac{1}{T}\left[\sum_t 2^{-r+1}\sum_{j\in J} y_t(j)\ell_t(j)\right].$$

First, we derive the following technical lemma (its proof is in Appendix C).

**Lemma 4.8.** *Assume that $A^M_2$ is at height $r$ (part of $A^{I\cup J}_{2^r}$). Then*

$$W(I \cup J) \le \frac{1}{\frac{1}{W(I)} + \frac{1}{W(J)}} + \frac{\alpha}{\sqrt{T}} \le \frac{1}{\sum_{a\in I\cup J}\frac{1}{L_T(a)}} + \frac{r\alpha}{\sqrt{T}}.$$

The following claim, based on Lemma 4.2, bounds the difference between the average of all alternatives and the loss of a specific alternative.

**Claim 4.9.** *For any alternative $i \in K$ we have, $|W(K) - L_T^{A_{2^r}}(i)| \le \frac{r}{\sqrt{T}}$.*

*Proof.* We can view the algorithm $A_{2^r}$ as generating a complete binary tree. Consider a path from the root to a node that represents alternative $i \in K$. Let the sets of alternatives on the path be $I_0, \ldots, I_r$, where $I_0 = K$ and $I_r = \{i\}$. We are interested in bounding $|W(I_0) - W(I_r)|$. Clearly,

$$|W(I_0) - W(I_r)| \le \sum_{i=1}^{r} |W(I_{i-1}) - W(I_i)|.$$

Since $W(I_{i-1}) = \frac{1}{2}(W(I_i) + W(I_{i-1} - I_i))$, we have that $|W(I_{i-1}) - W(I_i)| = \frac{1}{2}|W(I_i) - W(I_{i-1} - I_i)|$. Since the alternative sets $I_i$ and $I_{i-1} - I_i$ are the alternatives of an $A^M_2$ algorithm (at depth $i$), by Corollary **??** we have that $|W(I_i) - W(I_{i-1} - I_i)| \le \frac{2}{\sqrt{T}}$. This implies that $|W(I_{i-1}) - W(I_i)| \le \frac{1}{\sqrt{T}}$. Therefore, $|W(I_0) - W(I_r)| \le \frac{r}{\sqrt{T}}$. □

Now we can combine the two claims to the following theorem.

**Theorem 4.10.** *Suppose the global cost function is makespan. For any set $K$ of $2^r$ alternatives, and for any loss sequence $\ell_1, \ldots, \ell_T$, algorithm $A_{2^r}$ will have regret at most $O(\frac{\log|K|}{\sqrt{T}})$, i.e., $C_\infty(L_T^{A_{2^r}}) - C^*_\infty(\ell) \le 49\frac{r}{\sqrt{T}}$.*

*Proof.* By Lemma 4.8 we have that $W(K) - C^*_\infty(\ell) \le \frac{r\alpha}{\sqrt{T}}$. By Claim 4.9 we have that for any alternative $i \in K$, $L_T^{A_{2^r}}(i) - W(K) \le \frac{r}{\sqrt{T}}$. Therefore, $C_\infty(L_T^{A_{2^r}}) - C^*_\infty(\ell) \le 49\frac{r}{\sqrt{T}}$, since $\alpha \le 48$ by Theorem 4.7. □

# 5 Lower Bounds

## 5.1 Lower Bound for $d$-norms

Our first lower bound shows that it is impossible to have a regret below $\sqrt{T}$ for any $d$-norm, and thus make our main result tight.

**Theorem 5.1.** *For every $T$ there exists a convex-concave cost function $C_{f(T)}$ such that the regret of any algorithm $A$ is at least $\Omega(\sqrt{T})$*

*Proof.* For every $T$ we will consider the following global cost function, $C_{1+\epsilon_T}$, where $\epsilon_T$ is positive and will be determined later. By definition $C_{1+\epsilon_T}$ is a convex-concave function. Next we show that an algorithm with regret $R(T)$ to with respect to $C_{1+\epsilon_T}$ then it has an $R(T)+o(\sqrt{T})$ regret in standard settings, i.e., sum of losses. We first observe that for small enough $\epsilon$ the following holds

$$\|(\alpha L_1)^{1+\epsilon} + (\beta L_2)^{1+\epsilon} - (\alpha L_1 + \beta L_2)^{1+\epsilon}\|_{1+\epsilon} \leq c(L_1+L_2)\epsilon,$$

where $c$ is some constant. Since for every $\alpha$ and $\beta$ the above holds, we immediately get that $\|C^*_{1+\epsilon_T}(L_1,L_2) - \max(\alpha L_1, \beta L_2)\| \leq \epsilon \max(L_1,L_2)$. Using the fact that the losses are bounded and by setting $\epsilon_t$ to $o(1/\sqrt{T})$ we obtain that $C^*_{1+\epsilon_T}$ approximates the standard $\ell_1$ norm by $o(\sqrt{T})$ and that for every algorithm $|C_{1+\epsilon_T}(L^T_A) - \|L^T_A\|_1|$ is bounded by $o(\sqrt{T})$. Now, if an algorithm has a regret of $o(\sqrt{T})$ for $(1+\epsilon_t)$-norm, then it has a regret of $o(\sqrt{T})$ in the 1-norm as well, which is impossible to attain (see [9]). □

## 5.2 Lower Bound for Non-Convex functions

In this section we show a lower bound that holds for a large range of non-convex global cost functions, defined as follows.

**Definition 5.2.** *A function $f$ is $\gamma_f$-strictly concave if for every $x > 0$: (1) $\frac{f(x/3)+f(2x/3)}{f(x)} \geq \gamma_f > 1$, (2) $\lim_{x\to 0} xf'(x) = 0$. (3) $f$ is non-negative concave and increasing (4) $\lim_{x\to 0} f(x) = 0$*

Note that any function $f(x) = x^\beta$ for $\beta < 1$, is $\gamma_f$-strictly concave with $\gamma_f = \left(\frac{2}{3}\right)^\beta + \left(\frac{1}{3}\right)^\beta$.

**Theorem 5.3.** *Let $C(L^A_T) = \sum_{i=1}^k f(L^A_T(i))$, where $f$ is $(\gamma_f)$-strictly concave. For any online algorithm $A$ there is an input sequence such that $C(L^A_T)/C^*(L_T) > \gamma_f$.*

The proof can be found in Appendix D.

# 6 Open Problems

In this work we defined the setting of an online learning with a global cost function $C$. For the case that $C$ is convex and $C^*$ is concave, we showed that there are online algorithms with vanishing regret. On the other hand, we showed that for certain concave functions $C$ the regret is not vanishing. Giving a complete characterization when does a cost function $C$ enable a vanishing regret is very challenging open problem. In this section we outline some possible research directions to address this problem.

First, if $C$ is such that for some monotone function $g$, $C' = g(C)$ satisfy the same properties as $d$-norm: $C'$ is convex, $C'^*(L) = \min_\alpha C'(L \odot \alpha)$ is concave, and $\alpha^*(L)$ is Lipschitz, the algorithm of Section 3 for $C'$ would still lead to vanishing regret for $C$. For example, $C(L) = \sum_{i=1}^k (L_T(i))^d$ can be solved with that approach.

A more interesting case is when the properties are violated. Obviously, in light of the lower bound of Section 5.2 attaining $C^*$ in all cases is impossible. The question is if there is some more relaxed goal that can be attained. As in [19], one can take the convex hull of the target set in Eq. (2) and still consider an approaching strategy. It can be shown that by using the approachability strategy to the convex hull,

the decision maker minimizes the regret obtained with respect to the function $C^C$ defined below (i.e., $R_T^C(A) = C(L_T^A) - C^C(L_T)$). Specifically:

$$C^C(L) = \sup_{\substack{L^1,L^2,\ldots,L^m,\ \beta^1,\ldots\beta^m,\ \beta_j\geq 0:\\ \sum\beta_j=1,\ \sum_{j=1}^m \beta^j L^j = L}} C\left(\sum_{j=1}^m \beta^j \alpha_C^*(L^j)\odot L^j\right).$$

It follows that $C^*(L) \leq C^C(L)$ and $C^*(L) = C^C(L)$ if $C$ is convex and $C^*$ concave (since in that case the convex hull of $S$ equals $S$). In general, however, $C^C$ may be strictly larger than $C^*$. The question if $C^C$ is the lowest cost that can be attained against any loss sequence is left for future research.

A different open problem is a computational one. The approachability-based scheme requires projecting a point to a convex set. While this can be done in polynomial time, the resulting policy is still complicated comparing to regret minimization algorithms for the additive cost case or the algorithm for the makespan we developed. It would be interesting to devise regret minimization algorithms for $d$-norm cost functions that are simple in the style, perhaps, of follow the perturbed leader or exponential weights (see [9] for a review of these algorithms).

Finally, although we have shown that for any horizon $T$ there is an $L_d$ norm that has regret $\Omega(\sqrt{T})$, there is a significant room for improvement. Specifically, we are not able to show any $\omega(1)$ lower bound for either the $L_2$ norm or the makespan. Such lower bounds would significantly increase our understanding of the model and contrast the difference with the linear case.

## Acknowledgements

Robert Kleinberg was supported by NSF Awards CCF-0643934, IIS-0905467, and AF-0910940, a Microsoft Research New Faculty Fellowship, and an Alfred P. Sloan Foundation Fellowship. Shie Mannor was supported by a Horev Fellowship and by the Israel Science Foundation (contract 890015). Yishay Mansour was supported in part by a grant from the Ministry of Science, by a grant from the Israel Science Foundation, by grant No. 2008-321 from the United States-Israel Binational Science Foundation (BSF), The Israeli Centers of Research Excellence (I-CORE) program, (Center No. 4/11), and by the IST Programme of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886. This publication reflects the authors' views only.

# References

[1] Baruch Awerbuch and Robert Kleinberg. Online linear optimization and adaptive routing. *J. Comput. Syst. Sci.*, 74(1):97–114, 2008.

[2] Baruch Awerbuch and Yishay Mansour. Adapting to a reliable network path. In *PODC*, pages 360–367, 2003.

[3] D. Blackwell. An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6:1–8, 1956.

[4] Avrim Blum and Carl Burch. On-line learning and the metrical task system problem. In *COLT*, pages 45–53, 1997.

[5] Avrim Blum, Carl Burch, and Adam Kalai. Finely-competitive paging. In *FOCS*, pages 450–458, 1999.

[6] Avrim Blum, Shuchi Chawla, and Adam Kalai. Static optimality and dynamic search-optimality in lists and trees. *Algorithmica*, 36(3):249–260, 2003. (A preliminary version appeared in SODA 2002.).

[7] Avrim Blum and Yishay Mansour. From external to internal regret. *J. Mach. Learn. Res.*, 8:1307–1324, 2007.

[8] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[9] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, New York, 2006.

[10] D. P. de Farias and N. Megiddo. Combining expert advice in reactive environments. *Journal of the ACM*, 53(5):762–799, 2006.

[11] Eyal Even-Dar, Sham M. Kakade, and Yishay Mansour. Online Markov decision processes. *Math. Oper. Res.*, 34(3):726–736, 2009.

[12] Yoav Freund, Robert E. Schapire, Yoram Singer, and Manfred K. Warmuth. Using and combining predictors that specialize. In *STOC*, pages 334–343, 1997.

[13] Thomas P. Hayes. A large-deviation inequality for vector-valued martingales, 2005. Working paper, available at http://www.cs.unm.edu/∼hayes/papers/.

[14] Sham M. Kakade, Adam Tauman Kalai, and Katrina Ligett. Playing games with approximation algorithms. In *STOC*, pages 546–555, 2007.

[15] A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291 – 307, 2005. An earlier version appeared in COLT 2003.

[16] Olav Kallenberg and Rafal Sztencel. Some dimension-free features of vector-valued martingales. *Probab. Th. Rel. Fields*, 88:215–247, 1991.

[17] Hellmuth Kneser. Sur un théorème fondamental de la théorie des jeux. *Computes Rendues Acad. Sci. Paris*, 234:2418–2420, 1952.

[18] Ehud Lehrer. A wide range no-regret theorem. *Games and Economic Behavior*, 42(1):101–115, 2003.

[19] S. Mannor and N. Shimkin. The empirical Bayes envelope and regret minimization in competitive Markov decision processes. *MOR*, 28(2):327–345, 2003.

[20] Jia-Yuan Yu, Shie Mannor, and Nahum Shimkin. Markov decision processes with arbitrary reward processes. *Math. Oper. Res.*, 34(3):737–757, 2009.

# A Properties of Norms

We are now ready to extend the results to any norm. Let us start by computing what is the optimal assignment and what is its cost.

**Lemma A.1.** *Consider the d-norm global cost function and a sequence of losses $\ell_1, \ldots, \ell_T$. The optimal stationary distribution is given by* $\alpha^*_{C_d}(\ell) = \left( \frac{1/L_T(i)^{\frac{d}{d-1}}}{\sum_{j=1}^k 1/L_T(j)^{\frac{d}{d-1}}} \right)_{i \in K}$ *and the optimal global cost function is given by*

$$C^*_d(\ell) = \left( \frac{1}{\sum_{j=1}^k 1/L_T(j)^{\frac{d}{d-1}}} \right)^{\frac{d-1}{d}}.$$

*Proof.* Recall that $L_T(1), \ldots, L_T(k)$ are the average losses of the alternatives. To compute the optimal $d$-norm static distribution, we need to optimize the following cost function $C_d(\ell) = \left( \sum_{i=1}^k (\alpha(i) L_T(i))^d \right)^{1/d}$

subject to the constraint that $\sum_{i=1}^{k} \alpha(i) = 1$. Using Lagrange multipliers we obtain that the optimum is obtained for the following values

$$(\alpha^*_{C_d}(\ell))(i) = \alpha(i) = \frac{1/L_T(i)^{\frac{d}{d-1}}}{\sum_{j=1}^{k} 1/L_T(j)^{\frac{d}{d-1}}}.$$

This implies that the cost of the optimal distribution is given by

$$\begin{aligned}
C^*_d(L) &= \left(\sum_{i=1}^{k} (\alpha(i)L_T(i))^d\right)^{1/d} \\
&= \left(\sum_{i=1}^{k} \left(\frac{1/L_T(i)^{\frac{d}{d-1}}}{\sum_{j=1}^{k} L_T(j)^{\frac{d}{d-1}}} L_T(i)\right)^d\right)^{1/d} \\
&= \left(\sum_{i=1}^{k} \left(\frac{\frac{1}{L_T(i)^{\frac{1}{d-1}}}}{\sum_{j=1}^{k} 1/L_T(j)^{\frac{d}{d-1}}}\right)^d\right)^{1/d} \\
&= \frac{1}{\sum_{j=1}^{k} 1/L_T(j)^{\frac{d}{d-1}}} \left(\sum_{i=1}^{k} \frac{1}{L_T(i)^{\frac{d}{d-1}}}\right)^{1/d} \\
&= \left(\frac{1}{\sum_{j=1}^{k} 1/L_T(j)^{\frac{d}{d-1}}}\right)^{\frac{d-1}{d}}. \qquad \square
\end{aligned}$$

After computing $C^*_d$ we are ready to prove its concavity.

**Lemma A.2.** *The function $C^*_d(\ell)$ is concave.*

*Proof.* In order to show that $C^*_d$ is concave we will prove that its Hessian is negative semidefinite. We first compute the derivative and the second partial derivatives. We define for simplicity $\beta = \frac{d}{d-1} > 1$ and $\gamma^*(x) = \frac{1}{\sum_{j=1}^{k} 1/x_j^{\beta}}$ thus $C^*_d(x) = \gamma^*(x)^{1/\beta}$.

$$\begin{aligned}
\frac{\partial C^*_d}{\partial x(i)} &= \frac{1}{\beta}(\gamma^*(x))^{1/\beta - 1} \frac{\beta x(i)^{-(\beta+1)}}{\left(\sum_{m=1}^{k} \frac{1}{x(m)^{\beta}}\right)^2} \\
&= x(i)^{-(\beta+1)} \gamma^*(x)^{\frac{1}{\beta}+1} \\
\frac{\partial^2 C^*_d}{\partial x(i)\partial x(j)} &= x(i)^{-(\beta+1)} (\frac{1}{\beta} + 1)(\gamma^*(x))^{1/\beta} \frac{\beta x(j)^{-(\beta+1)}}{\left(\sum_{m=1}^{k} \frac{1}{x(m)^{\beta}}\right)^2} \\
&= (\beta+1)\gamma^*(x)^{\frac{1}{\beta}+2} x(i)^{-(\beta+1)} x(j)^{-(\beta+1)}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^2 C_d^*}{\partial^2 x(i)} &= -(\beta+1)x(i)^{-(\beta+2)}\gamma^*(x)^{\frac{1}{\beta}+1} \\
&\quad +x(i)^{-(\beta+1)}(\frac{1}{\beta}+1)\gamma^*(x)^{\frac{1}{\beta}}\frac{\beta x(i)^{-(\beta+1)}}{\left(\sum_{m=1}^k \frac{1}{x(m)^\beta}\right)^2} \\
&= -(\beta+1))x(i)^{-(\beta+2)}\gamma^*(x)^{\frac{1}{\beta}+1} + (\beta+1)x(i)^{-2(\beta+1)}\gamma^*(x)^{\frac{1}{\beta}+2} \\
&= (\beta+1)\gamma^*(x)^{\frac{1}{\beta}+2}\left(-x(i)^{-(\beta+2)}\sum_{m=1}^k x(m)^{-\beta} + x(i)^{-2(\beta+1)}\right) \\
&= (\beta+1)\gamma^*(x)^{\frac{1}{\beta}+2}\left(-x(i)^{-(\beta+2)}\sum_{m=1,k\neq i}^k x(m)^{-\beta}\right).
\end{aligned}$$

Since all entries of the Hessian are factor of $(\beta+1)\gamma^*(x)^{\frac{1}{\beta}+2}$ which is positive we can eliminate it without effecting the semidefinte negative property. Let $H$ be the Hessian and $a$ any vector, then,

$$\begin{aligned}
aHa' &= \sum_{i=}^k \sum_{j=i+1}^k 2x(i)^{-(\beta+1)}x(j)^{-(\beta+1)}a_i a_j + \sum_{i=1}^k a_i^2\left(-x(i)^{-(\beta+2)}\sum_{m=1,k\neq i}^k x(m)^{-\beta}\right) \\
&= \sum_{i=1}^k \sum_{j=i+1}^k \Big(-a_i^2 x(i)^{-(\beta+2)}x(j)^{-\beta} - a_j^2 x(j)^{-(\beta+2)}x(i)^{-\beta} + 2x(i)^{-(\beta+1)}x(j)^{-(\beta+1)}a_i a_j\Big) \\
&= \sum_{i=1}^k \sum_{j=i+1}^k x(i)^{-\beta}x(j)^{-\beta}\left(-\frac{a_i^2}{x(i)^2} - \frac{a_j^2}{x(j)^2} + \frac{2a_i a_j}{x(i)x(j)}\right) \\
&= \sum_{i=1}^k \sum_{j=i+1}^k -x(i)^{-\beta}x(j)^{-\beta}(\frac{a_i}{x(i)} - \frac{a_j}{x(j)})^2 \leq 0. \quad \square
\end{aligned}$$

Now consider the makespan metric which is the standard metric for load balancing and job scheduling tasks. The optimal stationary distribution is the one which balances load evenly across all $k$ machines, which implies the following:

**Lemma A.3.** *Consider the makespan global cost function and a sequence of losses $\ell_1, \ldots, \ell_T$. The optimal stationary distribution is given by $\alpha^*_{C_\infty}(\ell) = \left(\frac{1/L_T(i)}{\sum_{j=1}^k 1/L_T(j)}\right)_{i\in K}$ and the optimal global cost function is given by*

$$C^*_\infty(\ell) = \frac{1}{T}\left(\frac{1}{\sum_{j=1}^k 1/L_T(j)}\right).$$

Having derived the formula for $C^*_\infty$, we now proceed to prove that it is concave.

**Lemma A.4.** *The function $C^*_\infty$ is concave.*

*Proof.* In order to do so we will show that the Hessian of $C^*$ is negative semidefinite. We start by computing the partial derivatives of $C^*$.

$$\begin{aligned}
\frac{\partial C^*}{\partial x(i)} &= \frac{1/x(i)^2}{(\sum_{j=1}^k 1/x(j))^2} \\
\frac{\partial^2 C^*}{\partial x(i)\partial x(j)} &= \frac{2/(x(i)^2x(j)^2)}{(\sum_{j=1}^k 1/x(j))^3}, \\
\frac{\partial^2 C^*}{\partial x(i)^2} &= \frac{-2\sum_{j\neq i}\frac{1}{x(i)^3x(j)}}{(\sum_{j=1}^k 1/x(j))^3}.
\end{aligned}$$

Next we show that $aHa' < 0$, where $H$ is the Hessian and $a$ is any vector. To simplify the computation we eliminate the common factor $2/(\sum_{j=1}^{k} 1/x(j))^3$ which is positive from all entries.

$$\begin{aligned} aHa' &= \sum_{i=1}^{k}\sum_{j=i+1}^{k} \frac{2a_i a_j}{x(i)^2 x(j)^2} - \sum_{i=1}^{k}\sum_{j\neq i} \frac{a_i^2}{x(i)^3 x(j)} \\ &= \sum_{i=1}^{k}\sum_{j=i+1}^{k} \frac{1}{x(i)x(j)} \left( -\frac{a_i^2}{x(i)^2} + \frac{2a_i a_j}{x(i)x(j)} - -\frac{a_j^2}{x(j)^2} \right) \\ &= \sum_{i=1}^{k}\sum_{j=i+1}^{k} \frac{-1}{x(i)x(j)} \left( \frac{a_i}{x(i)} - \frac{a_j}{x(j)} \right)^2 \leq 0. \end{aligned}$$

Therefore, we showed that $C^*$ is concave and we conclude the proof. □

# B Proof of Blackwell's Theorem

This section provides a proof of Theorem 3.3, Blackwell's Approachability Theorem. For convenience, we begin by restating the theorem, preceded by the relevant definitions.

**Definition 3.1.** *We say that $V$ is* approachable with convergence rate $r(t)$ *if there exists an algorithm for choosing a sequence of vectors $a_1, a_2, \ldots \in A$ such that for every sequence $b_1, b_2, \ldots \in B$ and all integers $t \geq 1$:*

1. *the value of $a_t$ depends only on the values of $m(a_s, b_s)$ for $s < t$;*
2. *the average payoff $\widehat{m}_t = \frac{1}{t}\sum_{s=1}^{t} m(a_s, b_s)$ satisfies $\mathrm{dist}(\widehat{m}_t, V) \leq r(t)$.*

**Definition 3.2.** *We say that $V$ satisfies the* Blackwell criterion *if $V$ is closed and convex and for every $b \in B$ there exists $a \in A$ such that $m(a, b) \in V$.*

**Theorem 3.3.** *Given compact, convex sets $A \subseteq \mathbb{R}^a$, $B \subseteq \mathbb{R}^b$ and a bilinear function $m : \mathbb{R}^a \times \mathbb{R}^b \to \mathbb{R}^c$, if $V$ satisfies the Blackwell criterion then $V$ is approachable with convergence rate $r(t) = 2D/\sqrt{t}$, where $D = \sup_{x,y\in m(A\times B)} \|x - y\|$ is the diameter of the set $m(A \times B)$ and $D \geq \sup_{x,y\in V} \|x - y\|$.*

*Proof.* For every vector $u \in \mathbb{R}^c$, the mapping $(a, b) \mapsto m(a, b) \cdot u$ defines a two-player game with scalar payoff. Kneser's Minimax Theorem [17] (a generalization of von Neumann's Minimax Theorem) implies that

$$\inf_{a\in A} \sup_{b\in B} \{m(a, b) \cdot u\} = \sup_{b\in B} \inf_{a\in A} \{m(a, b) \cdot u\}.$$

Our assumption that for every $b \in B$ there exists $a \in A$ such that $m(a, b) \in V$ implies that the right side is less than or equal to $\sup_{s\in V} s \cdot u$. Our assumption that $A$ is compact implies that the infimum on the left side is attained at some $a \in A$. Putting these observations together, we conclude that for every $u \in \mathbb{R}^c$, there exists some $a = a(u) \in A$ such that

$$\sup_{b\in B} m(a(u), b) \cdot u \leq \sup_{s\in V} s \cdot u, \tag{8}$$

a fact which will be crucial to designing and analyzing the algorithm whose existence is asserted by the theorem.

To state the algorithm, let us first define the following notation. Let $\widehat{m}_0$ be an arbitrary element of $V$, and for $t > 0$ let $\widehat{m}_t = \frac{1}{t}\sum_{s=1}^{t} m(a_s, b_s)$ denote the average payoff vectors realized in the first $t$ rounds. Let $\delta_t = \mathrm{dist}(\widehat{m}_t, V)$ and let $s_t$ be the point in $V$ that is closest to $\widehat{m}_t$ in Euclidean distance. At time $t + 1 \geq 1$ the algorithm computes the vectors $\widehat{m}_t$ and $s_t$. If $\widehat{m}_t \in V$ then it chooses $a_{t+1}$ to be an arbitrary element of $A$. Otherwise, it chooses $a_{t+1} = a(\widehat{m}_t - s_t)$.

To analyze the algorithm we analyze the relationship between $\delta_{t+1}$ and $\delta_t$. Putting

$$\begin{aligned} u &= \widehat{m}_t - s_t \\ v &= m(a_{t+1}, b_{t+1}) - s_t \end{aligned}$$

and using the facts that

$$\begin{aligned} \delta_t &= \|u\| \\ \delta_{t+1} &\leq \|\widehat{m}_{t+1} - s_t\| \\ \widehat{m}_{t+1} - s_t &= \left(\frac{t}{t+1}\right) u + \left(\frac{1}{t+1}\right) v \end{aligned}$$

we obtain the bound

$$\delta_{t+1}^2 \leq \left(\frac{t}{t+1}\right)^2 \delta_t^2 + \frac{2t}{(t+1)^2}(v \cdot u) + \left(\frac{1}{t+1}\right)^2 \|v\|^2. \tag{9}$$

The dot product $v \cdot u$ occurring in the middle term is non-positive. This can be seen by a case analysis: if $\widehat{m}_t \in V$ then $u = 0$, whereas if $\widehat{m}_t \notin V$ then the vector $a_{t+1} = a(u)$ satisfies (8), and thus

$$v \cdot u = (m(a_{t+1}, b_{t+1}) - s_t) \cdot u \leq \sup_{s \in V}\{(s - s_t) \cdot u\}.$$

To prove that the right side is non-positive, we observe that $(s - s_t) \cdot u$ is the derivative of the function $-\frac{1}{2}\|\widehat{m}_t - (xs_t + (1-x)s)\|^2$ at $x = 1$. This function is maximized at $s_t$, by our assumptions that $V$ is convex (hence contains $xs_t + (1-x)s$ for $x \in [0, 1]$) and that $s_t$ is the point of $V$ that minimizes the distance to $\widehat{m}_t$.

Having proven that the middle term on the right side of (9) is non-positive, we turn to the task of bounding the third term. The triangle inequality implies that

$$\|v\| = \|m(a_{t+1}, b_{t+1}) - s_t\| \leq \|m(a_{t+1}, b_{t+1}) - \widehat{m}_t\| + \|\widehat{m}_t - s_t\|. \tag{10}$$

The convexity of the 2-norm implies that for any $m(a, b) \in m(A \times B)$, we have

$$\|m(a, b) - \widehat{m}_t\| \leq \frac{1}{t}\sum_{s=1}^{t} \|m(a, b) - m(a_s, b_s)\| \leq \operatorname{diam}(m(A \times B)) = D. \tag{11}$$

In particular, the first term on the right side of (10) is bounded above by $D$. The second term is also bounded above by $D$. To see this, note that there is some $m(a, b) \in V$ (recall that for every $b \in B$ there exists $a \in A$ such that $m(a, b) \in V$, which is much stronger). Hence,

$$\|\widehat{m}_t - s_t\| = \min_{s \in V} \|\widehat{m}_t - s\| \leq \|\widehat{m}_t - m(a, b)\| \leq D,$$

where the last inequality follows by another application of (11).

Applying these bounds to (9), we have thus derived

$$\begin{aligned} \delta_{t+1}^2 &\leq \left(\frac{t}{t+1}\right)^2 \delta_t^2 + \left(\frac{1}{t+1}\right)^2 \cdot (2D)^2 \\ (t+1)^2\delta_{t+1}^2 &\leq t^2\delta_t^2 + 4D^2. \end{aligned}$$

Together with the initial condition $\delta_0 = 0$, an obvious induction now establishes that $t^2\delta_t^2 \leq 4D^2t$ for all $t \geq 0$. The bound $\delta_t \leq 2D/\sqrt{t}$ in the theorem statement follows by an algebraic manipulation. □

# C Proof of Lemma 4.8

Let $W_1 = W(I)$ and $W_2 = W(J)$. The first inequality follows from the fact that our assumption of the regret of $A_2^M$ implies that $\max\{W_1, W_2\} \leq \frac{1}{1/W_1+1/W_2} + \frac{\alpha}{\sqrt{T}}$. This also proves the base of the induction for $r = 1$.

For the proof let $\pi_1 = \prod_{i\in I} L_T(i)$, $\pi_2 = \prod_{j\in J} L_T(j)$ and $\pi = \pi_1\pi_2$. Also, $s_1 = \sum_{i\in I} \prod_{k\in I\setminus\{i\}} L_T(k)$, $s_2 = \sum_{j\in J} \prod_{k\in J\setminus\{j\}} L_T(k)$, and $s = s_1\pi_2 + s_2\pi_1$. One can verify that the optimal cost $opt = (\sum_i 1/L_T(i))^{-1}$ equals to $\pi/s$.

Let $\beta = \frac{(r-1)\alpha}{\sqrt{T}}$. From the inductive assumption that we have $W_i \leq \pi_i/s_i + \beta$ for $i \in \{1, 2\}$. The proof follows from the following.

$$\begin{aligned}
\frac{1}{1/W_1 + 1/W_2} + \frac{\alpha}{\sqrt{T}} &\leq \frac{1}{\frac{1}{\pi_1/s_1+\beta} + \frac{1}{\pi_2/s_2+\beta}} + \frac{\alpha}{\sqrt{T}} \\
&= \frac{1}{\frac{s_1}{\pi_1+s_1\beta} + \frac{s_2}{\pi_2+s_2\beta}} + \frac{\alpha}{\sqrt{T}} \\
&= \frac{(\pi_1 + s_1\beta)(\pi_2 + s_2\beta)}{s_1(\pi_2 + s_2\beta) + s_2(\pi_1 + s_1\beta)} + \frac{\alpha}{\sqrt{T}} \\
&= \frac{\pi_1\pi_2 + \beta(s_1\pi_2 + s_2\pi_1) + \beta^2 s_1 s_2}{s_1\pi_2 + s_2\pi_1 + 2s_2 s_1\beta} + \frac{\alpha}{\sqrt{T}} \\
&= \frac{\pi + \beta s + \beta^2 s_1 s_2}{s + 2s_2 s_1\beta} + \frac{\alpha}{\sqrt{T}} \\
&\leq \frac{\pi}{s} + \beta\frac{s + \beta s_1 s_2}{s + 2s_2 s_1\beta} + \frac{\alpha}{\sqrt{T}} \\
&\leq opt + \beta + \frac{\alpha}{\sqrt{T}} = opt + r\frac{\alpha}{\sqrt{T}}. \qquad \square
\end{aligned}$$

# D Lower bound for a concave loss function

**Theorem 5.3.** Consider the following three sequences of losses: (1) sequence $\sigma_1$ has $t$ time steps each with losses $(\delta, \delta)$, (2) sequence $\sigma_2$ starts with sequence $\sigma_1$ followed by $t$ time steps each with losses $(1, \delta)$, (3) sequence $\sigma_3$ starts with sequence $\sigma_1$ followed by $t$ time steps each with losses $(\delta, 1)$ and $\delta$ will be determined later

The *opt* for all sequences $\sigma_1, \sigma_2$ and $\sigma_3$ has a loss of $f(\delta)$.

Given an online algorithm $A$, after $\sigma_1$ it has an average loss of $\beta\delta$ on alternative 1 and $(1-\beta)\delta$ on alternative 2. Therefore, its loss on $\sigma_1$ is $f(\beta\delta) + f((1-\beta)\delta)$. If $\beta \in [1/3, 2/3]$ then the loss of $A$ is at least $f((1/3)\delta) + f((2/3)\delta)$ compared to a loss of $f(\delta)$ of $OPT$. Hence $C(L_T^A)/C^*(\sigma_1) \geq \gamma_f$.

If $\beta > 2/3$ then consider the performance of $A$ on $\sigma_2$. On the first $t$ time steps it behaves the same as for $\sigma_1$ and at the remaining $t$ time steps it splits the weight such that its average loss on those steps is $\lambda$ for alternative 1 and $(1-\lambda)\delta$ for alternative 2. Therefore its cost function is $f((\beta\delta+\lambda)/2)+f((1-\beta+1-\lambda)\delta/2)$. First, we note that for $\lambda > 1/2$, we have that $C(L_T^A) \geq f(1/4)$. Since the cost of OPT is $f(\delta)$ and as $\delta$ goes to 0 $C^*(\sigma_2)$ goes to zero, then for sufficiently small $\delta$ we have that $C(L_T^A)/C^*(\sigma_2) \geq \gamma_f$. For $\lambda \leq 1/2$ define $h(\lambda) = f((\beta\delta + \lambda)/2) + f((1 - \beta + 1 - \lambda)\delta/2)\delta/2$. First, we would like to show that $h'(\lambda) > 0$ for every $\lambda \in [0, 1/2]$. Consider,

$$h'(\lambda) = f'((\beta\delta + \lambda)/2)/2 - \frac{\delta}{2}f'((1 - \beta + 1 - \lambda)\delta/2) > 0.$$

Since $f$ is concave $f'$ is decreasing and given our assumption that $\delta f'(\delta/2)$ goes to zero as $\delta$ goes to zero, then for small enough $\delta$ and $\lambda \leq 1/2$ we can bound as follows:

$$\begin{aligned} f'((\beta\delta+\lambda)/2) &\geq f'(1) \geq 4\delta f'(\delta/4) \\ &\geq \delta f'((1-\beta+1-\lambda)\delta/2). \end{aligned}$$

Thus $h(\lambda)$ is increasing and its minimum is obtained at $\lambda = 0$. Therefore the cost function of algorithm, $A$, is at least $f(2\delta/3) + f(\delta/3)$ while the optimum achieves $f(\delta)$. Hence $C(L_T^A)/C^*(\sigma_2) \geq \gamma_f$.

The case of $\beta < 1/3$ is similar to that of $\beta > 2/3$, just using $\sigma_3$ rather than $\sigma_2$. □