μΠλ∀

National Kapodistrian University of Athens

Graduate Program of Logic, Algorithms and Computations

Department of Mathematics

Online Shortest Path with Switching Cost

Master Thesis

of

Tziotis Isidoros

Thesis' Advisor: Fotakis Dimitris
Assistant Professor NTUA.

Computation and Reasoning Laboratory

Athens, November 2017

Η παρούσα Διπλωματική Εργασία

εκπονήθηκε στα πλαίσια των σπουδών

για την απόκτηση του

Μεταπτυχιακού Διπλώματος Ειδίκευσης

στη

Λογική και Θεωρία Αλγορίθμων και Υπολογισμού

που απονέμει το

Τμήμα Μαθηματικών

του

Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών

Εγκρίθηκε την 22/11/2017 από Εξεταστική Επιτροπή

αποτελούμενη από τους:

	Ονοματεπώνυμο	**Βαθμίδα**	**Υπογραφή**
1.	Δημ. Φωτάκης	Επ. Καθηγητής	Φωτάκης
2.	Αρ. Παγουρτζής	Αν. Καθηγητής	[illegible]
3.	Ε. Ζάχος	Καθηγητής	Ε. Ζάχος

μΠλ∀

National Kapodistrian University of Athens

Graduate Program of Logic, Algorithms and Computations

Department of Mathematics

Online Shortest Path with Switching Cost

Master Thesis

of

Tziotis Isidoros

Thesis' Advisor: Fotakis Dimitris
Associate Professor NTUA.

Accepted by the committee 22th November 2017:

....	....	....
Fotakis Dimitris	Pagourtzis Aris	Zachos Stathis
Assistant Professor NTUA	Associate Professor NTUA	Professor NTUA

Computation and Reasoning Laboratory

Athens, November 2017

....
Tziotis Isidoros
Graduate Student of Department of Informatics & Telecommunications in National Kapodistrian University of Athens

Περίληψη

Ένα χαρακτηριστικό on-line πρόβλημα διεξάγεται σε γύρους, όπου σε κάθε γύρο ένας on-line αλγόριθμος δέχεται ένα αίτημα και οφείλει να το ικανοποιήσει. Θα επικεντρωθούμε σε μία συγκεκριμένη οικογένεια on-line προβλημάτων γνωστά ως Smooth On-line Convex Optimization (SOCO) προβήματα. Δύο γνωστά επιστημονικά πεδία που μελετούν τέτοια προβλήματα είναι το πεδίο competitive analysis και το πεδίο on-line learning. Θα εμβαθύνουμε στη σχέση των δύο πεδίων και θα εξηγήσουμε πως μπορούμε να επωφεληθούμε εισάγωντας την τεχνική regularization από το πεδίο του on-line learning στο competitive analysis. Στη συνέχεια θα εστιάσουμε σε μία τεχνική rounding η οποία εμφανίστηκε στη βιβλιογραφία τα τελευταία χρόνια και ονομάζεται exponential clocks. Τέλος, θα ορίσουμε ένα νέο πρόβλημα της οικογένειας SOCO, το On-line Shortest Paths with Switching Cost. Χρησιμοποιώντας εργαλεία από τη βιβλιογραφία θα πάρουμε μία on-line fractional λύση του προβλήματος θυσιάζοντας ένα λογαριθμικό παράγοντα. Θα ολοκληρώσουμε παρουσιάζοντας ένα νέο rounding αλγόριθμο χρησιμοποιώντας exponential clocks, ο οποίος θα επιτύχει μια $O(\log m \log n)-$ προσεγγιστική λύση για το πρόβλημα On-line Shortest Path with Switching Cost.

Λέξεις Κλειδιά

Online Convex Optimization, Smooth Optimization, On-line Learning, Συντομότερα Μονοπάτια, Exponential Clocks

Abstract

A typical on-line problem proceeds in rounds, where in each round an on-line algorithm is given a request and needs to serve it. We will focus on a specific class of on-line problems known as Smooth On-line Convex Optimization (SOCO) problems. Two mature research fields that study such problems are competitive analysis and on-line learning. We will dive into their interrelationship and we will explain how we can benefit by introducing regularization, a standard technique from on-line learning in the framework of competitive analysis. Subsequently, we will turn our attention towards a rounding technique introduced over the last couple of years, called exponential clocks. Finally, we will define a new problem in the class SOCO, namely On-line Shortest Path with Switching Cost. Using the toolbox provided by the literature we will obtain an on-line fractional solution sacrificing a logarithmic factor. We will wrap up presenting a new on-line rounding algorithm using exponential clocks which will derive a $O(\log m \log n)$-approximation for the On-line Shortest Path with Switching Cost problem.

Keywords

Online Convex Optimization, Smooth Optimization, Online Learning, Shortest Paths, Exponential Clocks

Acknowledgements

I would like to express my deep appreciation to my supervisor, professor Dimitris Fotakis, for his constant help, encouragement and guidance which led me to this point. Moreover, I would like to express my gratitude to my colleagues Myrto Galenianou, Lydia Zakinthinou and Thanasis Lianeas whose unwavering support played a major role in the successful fulfillment of this dissertation. Furthermore, I would like to thank especially my colleague Manolis Vlatakis whose help and encouragement had a crucial impact over the last stages of my thesis. Finally, I would like to thank my family. Nothing would be possible if it wasn't for their guidance and love.

Isidoros Tziotis

Contents

Chapter 1

Introduction

On-line algorithms represent a theoretical framework for studying problems in interactive computing. They model, in particular, that the input in an interactive system does not arrive as a batch but as a sequence of input portions and that the system must react in response to each incoming portion. Moreover, it is assumed that at the future input is unknown and as the name suggests, on-line algorithms consider the algorithmic aspects of interactive systems. That is we wish to design strategies that always compute good output through consecutive rounds. No assumptions are made about the input stream. The input can even be generated by an adversary that creates new input portions based on the system's reactions to previous ones (adaptive adversary). We seek algorithms that have a provably good performance. A very interesting class of problems that we are going to study here is the class of on-line convex optimization problems.

In the context of **on-line convex optimization (OCO)** we consider the problem of a learner/decision-maker who interacts with an environment in a sequence of rounds and needs to make decisions in the face of uncertainty. During each round t :

- The learner must choose an action x^t from a convex decision space F
- The environment reveals a cost convex function c^t
- The learner experiences cost $c^t(x^t)$.

Depending on the setting (i) can happen before (ii) or vice versa. The objective here is to design learning algorithms that minimize the cost suffered over a long period of time i.e. T rounds. OCO has a wide array on applications such as portfolio management [CO96; KV02] and network routing [Ban+03].

Making more specific assumptions, two main performance metrics arise in order to deal with such problems. The first is competitive ratio where the benchmark is the optimal off-line solution and the second is regret where the goal is to minimize the difference between our incurred total cost and the total cost of the best fixed action. There is a large amount of related work in the literature, exploring the connection between competitive analysis and on-line learning [BB97; Buc+16; BBK99; KV02]. Despite their core deferences, there are also techniques that can be successfully transfered from one field to the other, providing innovative results. One such technique of great interest is Regularization. Regularization appears in the literature as early as [KW97] and [Gor99]. More recent results appear in [BCN14] and [CL06; RS13].

In our work the main focus will be on deriving good competitive ration in a generalization of on-line convex optimization. This more general framework is called smoothed on-line convex optimization (SOCO) whose study was motivated by the development in dynamic control algorithms in data centers [Lin+13; Lin+12a] and other streaming applications [JV11]. The only change in smoothed on-line convex optimization compared to on-line convex optimization is that the cost experienced by the learner is now $c_t(x_t)+ \parallel x_t - x_{t-1} \parallel$, where $\parallel \cdot \parallel$ is an arbitrary norm. In words the learner experiences an additional smoothing cost or switching cost associated with changing his action over consecutive time steps. Lin et al. have studied in [Lin+12b], the interrelationship of regret and competitive ratio in the framework of SOCO.

A robust on-line technique which applies to a large family of covering SOCO problems was presented in the work of Buchbinder et al. [BCN14]. In each round their algorithm solves a linear relaxation of the problem, via primal dual techniques [Lin+12a] and produces on-line a fractional solution within a logarithmic factor from the optimal off-line solution. A key element in this technique is the notion of regularization. The second main result of this paper considers an on-line version of Set Cover with Service Cost. The authors propose a rounding algorithm based on exponential clocks. Exponential clocks have been used extensively over the last couple of years in a variety of combinatorial problems such as Multi-way Cut Problem [Lin+13], Dynamic Facility Location [KV02; Fot06], Set Cover [Lin+13] and others. Coupling their rounding technique with the first result of their paper the authors derive an on-line algorithm solving the aforementioned problem with a total approximation factor $O(\log m \log |S_{max}|)$, where m is the number of sets and $|S_{max}|$ is the maximum capacity among the sets.

The setting proposed in [BCN14] is strongly motivated from a variety of real life applications. Often, one needs to solve instances of the same combinatorial optimization problem that changes over time. While this is naturally handled by re-solving the optimization problem in each time step separately, it is reasonable to assume that changing the solution in between consecutive time steps often incurs a transition cost. Related work on the same framework focused on On-online Multistage Matroid Maintenance problem [GTW14] and On-line Dynamic Facility Location.

In this dissertation we set on the same direction and construct a randomized on-line algorithm for the On-line Shortest Path problem with Switching Cost. The first step of our algorithm utilizes the results provided in [BCN14] in order to obtain a fractional on-line solution. Then we proceed rounding the solution at every time step coupling exponential clocks with a variation of Dijkstra which derives similar approximation bounds to the ones achieved in [BCN14; GTW14]. Thus, our algorithm is $O(\log m \log n)-$approximate where n is the number of vertices and m is the number of edges.

Chapter 2

Smooth On-line Convex Optimization

2.1 On-line algorithms and competitive analysis

In this dissertation we will work in the framework of on-line problems. A typical on-line problem proceeds in rounds, where in each round the on-line algorithm is given a request and needs to serve it.

To provide a better understanding we introduce the reader to the representative context of on-line convex optimization (OCO). In on-line convex optimization (OCO) we consider the problem of a learner/decision-maker who interacts with an environment in a sequence of rounds and needs to make decisions in the face of uncertainty.

During each round t :

1. The learner must choose an action x^t from a convex decision space F.
2. The environment reveals a cost convex function c^t.
3. The learner experiences cost $c^t(x^t)$.

Depending on the setting (1) can happen before (2) or vice versa. The objective here is to design learning algorithms that minimize the cost suffered over a long period of time i.e. T rounds.

Competitive analysis and on-line learning are two important research fields studying variations of such on-line problems. In competitive analysis in each time step t the learner is picking his action x^t after the cost function c^t is revealed i.e. $1-$lookahead, thus suffering cost $c^t x^t$. Then his total cost is compared to the total cost of the optimal off-line algorithm. Let S_1 and OPT be the cost our algorithm and the optimal off-line solution pay respectively. For now we can think of $S_1 = \sum_{t=1}^{T} x^t c^t$. Let also d be some constant. Then the competitive ratio c is the smallest number such that :

$$S_1 \leq c \cdot OPT + d$$

In the field of on-line learning however, the learner is far weaker picking his action before the cost function c^t is revealed i.e. $0-$lookahead, thus paying $c^t x^{t-1}$. To compensate for the lack of information about the future the learner in this framework has a weaker benchmark to compare to, namely regret. Minimizing regret is minimizing the difference between our incurred total cost and the total cost of the best fixed action x^*:

$$\boxed{Regret = \sum_{t=1}^{T} x^{t-1} c^t - \sum_{t=1}^{T} x^* c^t}$$

where $x^* = \arg\min_{x} \sum_{t=1}^{T} x c^t$.

Both areas have been extensively studied and important results are known. In the last couple of years a systematic attempt has been made to unify and compare the results from these two related fields. Although later we will come back to further explore this direction our primary focus remains on competitive analysis. To give a round view on competitive analysis and eventually move towards a more general definition than the one we gave for on-line convex optimization, we will introduce two typical competitive analysis problems, On-line Set Cover [Alo+03] and Metrical Task Systems[BLS92].

The Set Cover problem is defined as follows. Let $X = \{1, 2, ..., n\}$ be a ground set of n elements and let $\mathcal{S}$ be a family of subsets of X, $|\mathcal{S}| = m$. A cover is a collection of sets such that their union is X. Each $S \in \mathcal{S}$ has a nonnegative cost c_S associated with it. The goal is to find a cover of minimum cost. The set cover problem is a classic $\mathbf{NP}-$hard problem that was studied extensively in the literature, and the best approximation factor achievable for it in polynomial time (assuming $\mathbf{P} \neq \mathbf{NP}$) is $\Theta(\log n)$ [Lov75; AT96]. Now consider the following **On-line version of the Set Cover problem**, described as a game between an algorithm and an adversary. An adversary gives elements to the algorithm from X one-by-one. Once a new element is given, the algorithm has to cover it by some set of S containing it. Denote by $X' \subset X$ the set of elements given by the adversary. Our assumption is that the set cover instance, i.e. the elements of X and the members of $\mathcal{S}$, is known in advance to the algorithm. The objective is to minimize the total cost of the sets chosen by the algorithm. However, the algorithm does not know in advance the set of elements X' given by the adversary, i.e., X' may be a strict subset of X in general. Let C denote the family of sets in $\mathcal{S}$ that the algorithm chooses. At the end of the game the adversary also produces (off-line) a family of sets C_{OPT} that covers all the elements belonging to X'. The performance of the algorithm is defined to be the ratio between the cost of C and the cost of C_{OPT} . The maximum ratio, taken over all input sequences, is defined to be the competitive ratio of the algorithm.

For the problem above there is a $O(\log m \log n)$ approximation algorithm. More importantly we note the following:

- Initially the feasible region in $\mathcal{R}^m$ since no element needs to be covered.
- Upon arrival of element t the convex feasible region P_t is defined to be the intersection of P_{t-1} and the new covering constrain corresponding to element t.
- In the on-line set cover there is no service cost and thus there is no reason to drop any set chosen in some previous round.

This is a typical on-line problem which fits well in the context of OCO. Let us now move on to an other fundamental problem in competitive analysis:

A **Task System** is a pair (S, d) where $S = \{s_1, s_2, \ldots, s_n\}$ is a set of states and $d : S \times S \to \mathbb{R}$ is a distance function. If d is a metric, (S, d) is a Metrical Task System (MTS). An input to the task system is a sequence $\sigma = T_1, T_2, \ldots, T_l$ such that for each i, T_i is a vector of n non-negative entries that determine the processing costs for the n states when processing the ith task. An algorithm for the task system produces a schedule π that determines the sequence of states. For instance, $\pi(i) = s_j$ means that the ith task T_i is run in state s_j. The processing cost of a schedule is :

$$cost(\pi, \sigma) = \sum_{i=1}^{l} d(\pi(i-1), \pi(i)) + T_i(\pi(i))$$

The objective of the algorithm is to find a schedule such that the cost is minimized.

This on-line problem has a slightly different flavor than what we have seen so far. Although the feasible region is the same in every round we can see that the notion of states is introduced. This element is of vital importance in many fundamental problems in competitive analysis such as the k-Server problem [BBN10], the k -Armed Bandit problem [AT96; GM09] and the Stock Portfolio Management. Despite its importance the context of on-line convex optimization fails to capture this concept adequately. Thus we will proceed to generalize the context we are going to work in, to a more suitable one.

2.2 From OCO to SOCO

In the previous section we saw that the On-line Set Cover problem was sufficiently accommodated in the context of OCO whereas for the Metrical Task Systems there was a need to encapsulate the notion of states deriving from the previous rounds. An elegant, widely spread way to capture this, is the addition of switching or movement cost. In this case the learner experiences an additional smoothing cost associated with changing his action over consecutive time steps. Having this in mind we will extend the framework we provided for on-line convex optimization.

In **smooth on-line convex optimization (SOCO)** we consider the problem of a learner/decision-maker who interacts with an environment in a sequence of rounds and needs to make decisions in the face of uncertainty.

During each round t :

- The learner must choose an action x^t from a convex decision space F.
- The environment reveals a cost convex function c^t.
- The learner experiences cost $c^t(x^t) + \|x_t - x_{y-1}\|$.

Although we will see some results from the literature where different assumptions are made, for our convenience we can think the movement cost to be measured in ℓ_1. Furthermore, we assume that F is closed and bounded and that the Euclidean norm of the gradient of the service cost function is bounded. A famous algorithm designed for the problem we just described is the On-line Gradient Descent Algorithm. It turns out that not only does it give nice results in terms of competitive analysis but it also achieves good guarantees with respect to regret.

Online Gradient Descent:

Parameters: $\eta_t > 0$ changing in every round.

Initialization: Choose an arbitrary $x_1 \in F$, where F is the convex decision space.

At each time $t = [T]$:

(a) Let $c_t \in [0,1]^n$ be the service cost vector.

(b) Suffer cost $c_t(x_t)$.

(c) Update the decision as follows:

$$x_{t+1} = P(x_t - \eta_t \nabla c_t(x_t))$$

where $P(y) = \arg\min_{x\in F} \|x-y\|_2$ is the projection of point y on the convex space F under the Euclidian norm.

This algorithm achieves sublinear regret, depending on the choice of the learning rates η_t. For $\eta_t = \Theta(1/\sqrt{t})$ the regret after T rounds with respect to the best action in hindsight is $O(\sqrt{T})$. Making some additional assumptions and choosing $\eta_t = \Theta(\frac{1}{t})$ instead, the regret bound falls to $O(\log T)$.

As the authors of [Lin+12b] point out, the same algorithm provides a good regret bound even when switching cost is added. That is if $\sum_{t=1}^{n} \eta_t = O(\gamma_1(T))$ and the algorithm's regret is $O(\gamma_2(T))$ for Online Convex Optimization problems, then the same algorithm guarantees an $O(\gamma_1(T) + \gamma_2(T))$ regret for SOCO problems which implies that the two algorithms we mentioned before achieve the same regret bounds for the SOCO setting.

Coming back to the framework of competitive analysis we can also find a variety of results achieving good approximation ratio under different assumptions. Interestingly if we restrict our decision space to be a line then the approximation ratio for the problem can be even constant. In [BBN10], the authors present three algorithms for the SOCO problem given that the convex decision space $F = R$. A randomized algorithm that yields a fractional solution and a corresponding deterministic algorithm with competitive ratio 2 as well as a "memoryless" algorithm which only remembers the previous decision point x_{t-1} and has a 3-competitive ratio, which is also optimal (compared to any other memoryless deterministic algorithm). An important feature in the analysis of these algorithms is the usage of a suitable potential function. The method of defining appropriately and taking advantage of some potential function in the analysis of on-line learning techniques, is a recurring theme with great interest.

Before we move on we will present a natural problem from SOCO called On-line Set Cover with Service Cost. This problem encompasses features from both On-line Set Cover and MTS. Specifically, the feasible region P_t is going to change over the rounds and the notion of states is going to be captured by the introduction of switching cost on the sets chosen by our algorithm.

In order to define concretely the **On-line Set Cover with Service Cost** first let us recall the classic On-line Set Cover problem. In each round we are given elements which need to be covered by some set whose opening cost we pay only once. Transitioning in this new version of the problem, we are forced in each round t, to pay a service cost for every set we choose to use. Moreover, we are paying an opening cost(switching cost) whenever we opt to add in our solution a set which was not in the solution of the previous round. It is clear that an algorithm should both add and delete sets from the solution throughout its execution in order to achieve optimality in this setup.

More formally, the algorithm starts with $P_0 = \mathcal{R}^n$ (here we opt to denote with n the number of sets and m the number of elements in the ground set) as the feasible solution space and it is given a new polyhedron P_t in each round, defined by covering, along with a cost vector $c_t \in \mathcal{R}^n_+$. The objective function we seek to minimize is:

$$\sum_{t=1}^{T} c_t y_t + \sum_{t=1}^{T}\sum_{i=1}^{n} w_i \cdot \mid y_{i,t} - y_{i,t-1} \mid$$

where $y_t \in P_t$ and the movement cost is not uniform.

Later we are going to further analyze this problem thus it is an opportune moment to familiarize the reader with the Linear Program describing On-line Set Cover with Service Cost:

$$\begin{array}{lll}
\min & \displaystyle\sum_{t=1}^{T} c_t y_t + \sum_{t=1}^{T}\sum_{i=1}^{n} w_i z_{i,t} & \\
\text{subject to} & \displaystyle\sum_{i \in S_{j,t}} y_{i,t} \geq 1 & \forall t \geq 1, 0 \leq j \leq m_t \\
& z_{i,t} \geq y_{i,t} - y_{i,t-1} & \forall t \geq 1, 1 \leq i \leq n \\
& z_{i,t}, y_{i,t} \geq 0 & \forall t, 1 \leq i \leq n
\end{array}$$

The variables $y_{i,t}$ are indicators denoting if we include set i in the solution for this round or not. In each round, the subset of elements to be covered changes, so we denote by m_t the new number of constraints for this round. Also, $S_{j,t}$ is the set of sets which include element j in round t. The first constraint ensures that the j-th element to be covered is indeed fractionally covered and $\sum_{i=1}^{n} w_i z_{i,t} = \sum_{i=1}^{n} w_i \max\{0, y_{i,t} - y_{i,t-1}\}$ denotes the movement cost in round t.

2.3 Unifying competitive analysis and on-line learning

At this point we are going to revisit on-line learning. Despite the differences we already pointed out between competitive analysis and on-line learning there has been significant work in the direction of exploring to what extent common techniques can be applied to these rich and mature fields. Ideally, we would like to come up with a single algorithm which could guarantee both small regret bounds and good competitive ratio.

The first concrete attempt appears in [BB00] where the authors explore the area between the two frameworks with a combination of the well known and studied problem of Experts in on-line learning [FS97] and the Metrical Task Systems [BLS92] in competitive analysis. Specifically, this paper showed how certain algorithms, based on tuning some parameters, are able to interpolate between a reasonable regret bound and a reasonable competitive ratio. The interpolation was performed using the notion of α-unfair competitive ratio, which forces the policy we compete with to pay α times more for the movement cost.

We notice that in the limit, as α goes to infinity the competing policy becomes essentially static, and the setting becomes reminiscent of on-line learning. On the other hand by setting α equal to 1 the competing solution becomes the optimal as it is considered in competitive analysis. Thus, this scheme seems appropriate to track a sweat spot in between the two different benchmarks.

It turns out however, that the interpolation suggested above is not capable of achieving the best of the two worlds at the same time. Subsequently, many algorithms were proposed such as in [BBN10], trying to reach the same Holy Grail but it was Buhbinder et al. who proposed a rigorous unified approach in [Buc+16]. The authors via primal-dual techniques, described in this survey [BE98], provide good performance guarantees with respect to regret and competitive ratio which allows them to extend their results to more interesting settings.

Let us proceed to describe formally the setting as it is introduced in the latter paper. We denote by T the total number of rounds. We also have a set E of actions, with $|E| = n$ to chose from. In the experts setting in on-line learning, the decision maker maintains a distribution of weights over the set of actions, which can be considered to be a probability distribution over the actions. Based on that distribution the action of the next round is chosen. Since in on-line learning we have 0-lookahead, we denote by w_i^{t-1} the weight of the i-th expert that the algorithm has chosen at the end of the previous round, and by c_i^t the cost incurred in the current round for expert i which we assume is bounded in $[0, 1]$ by scaling. The total cost our algorithm is going to suffer over all rounds is given as:

$$S_0 = \sum_{t=1}^{T} w_{t-1} c_t$$

Whereas we can express the regret as:

$$\sum_{t=1}^{T} w_{t-1}c_t - \sum_{t=1}^{T} w^* c_t$$

where $w^* = \underset{w \geq 0, \|w\|_1 = 1}{\arg\min} \sum_{t=1}^{T} w c_t$ is the fixed distribution over the experts with the minimum total cost. At this point it is useful to mention the notion of *shifting* and drifting experts, which seems to bridge on-line learning and competitive analysis. In the case of shifting experts, we want to minimize the difference $\sum_{t=1}^{T} w_{t-1}c_t - \sum_{t=1}^{T} w_t^* c_t$ where w^* for $t \in \{0, ..., T-1\}$ is the optimal sequence of actions which changes at most k times. Drifting experts is an even more general notion which captures shifting experts as well. The goal in this case is to minimize the optimal sequence under the constraint that $\sum_{t=1}^{T} \frac{1}{2}\|w_t^* - w_{t-1}^*\|_1 \leq k$ which restricts in some sense the distance of the movement of the distribution. In the paper we are considering, drifting experts are used as a mean to measure the algorithm's performance over the "sweat spot" area, i.e. the area where we have good bounds with respect to both regret and competitive ratio.

In the context of competitive analysis again we assume 1-lookahead and thus the total cost for the distribution we choose in each round is:

$$S_1 = \sum_{t=1}^{T} w_t c_t$$

In this setting, we also have an additional switching cost :

$$M = \sum_{t=1}^{T} \frac{1}{2}\|w_t^* - w_{t-1}^*\|_1$$

In contrast to on-line learning, we compare our results to an optimal sequence $w_1^*, ..., w_T^*$ which minimizes the total cost:

$$\sum_{t=1}^{T} w_t c_t + \sum_{t=1}^{T} \frac{1}{2}\|w_t^* - w_{t-1}^*\|_1$$

and which has S_1^* total service cost and M^* total switching cost. Let $OPT = S_1^* + M^*$ be the optimal total cost. Then, in accordance with the definition we gave earlier the competitive ratio is the minimum c such that

$$S_1 + M \leq c \cdot OPT + d$$

where d is a constant. Finally, we formalize the α-unfair setting, where the optimal sequence is the minimizing sequence of $S_1 + \alpha M$ and we denote its total cost by $OPT(\alpha)$.

Having all the definition nailed down we can now proceed to highlight the take away points as stated by the authors.

1. Let OPT_k denote the optimal k-drifting sequence, then $OPT(\alpha) \leq OPT_k + \alpha k$.
   To see why this holds true, it is enough to consider the optimal k-drifting sequence with the additional term ak as an α-unfair sequence.
2. Since c_t is bounded for every t we have $S_0 \leq S_1 + M$.
   To derive this, one only needs to replace every distribution with the previous one and pay the appropriate switching cost if they are different.

Based on the above, the following inequalities hold, given that we have a c competitive algorithm for the α-unfair setting.

$$S_0 \leq S_1 + M \leq cOPT(\alpha) + d \leq cOPT_k + c\alpha k + d$$

The core algorithm the authors propose is the following:

Experts/MTS Algorithm (learning-style formulation)

Parameters:$\alpha \geq 1, \eta > 0$

Initialization: Set $w_{i,0} = \frac{1}{n}$ for all i.

At each time $t =\mid T \mid$

(a) Let $c_t \in [0,1]^n$ be the service cost vector.

(b) Using binary search, find the smallest a_t such that $\sum_{i=1}^{n} w_{i,t}$t is a distribution and

$$w_{i,t} = \max\{0, (w_{i,t-1} + \frac{1}{e^{\eta\alpha} - 1})e^{-\eta(c_{i,t} - a_t)} - \frac{1}{e^{\eta\alpha} - 1}\}$$

This algorithm achieves the following competitive ratio and regret as a function of the parameters α and η:

$$S_1 \le OPT(\alpha) + \frac{\ln n}{\eta}$$
$$\le (1 + \frac{n}{e^{\eta\alpha}-1})(\eta\ OPT(\alpha) + \ln n)$$

Setting $\alpha = 1$ and $\eta = \ln n + \ln\ln n$ we get the best known bound for MTS on uniform metrics. For $\alpha \to \infty$ where we pointed out earlier that our the setting approaches the classic on-line learning setting with experts, it holds that $S_0 \le S_1 + M \le (1+\eta)OPT(\infty) + \frac{\ln n}{\eta} + \ln n$ which matches the known bound from the multiplicative weights update framework.

The proof of the bound is based on the fact that this algorithm can be formulated equivalently in a primal-dual context which provides valuable insight regarding the inherent connection of the aforementioned problems.

The primal-dual formulation of the algorithm is the following:

Experts/MTS Algorithm(fractional primal-dual formulation):

Parameters: $\alpha \ge 1, \eta > 0$

Initialization: Set $w_{i,0} = \frac{1}{n}, b_{i,1} = \alpha - \frac{\ln(\frac{e^{\eta\alpha}+n-1}{n})}{\eta}$ for all $i \in \{1, ..., n\}$.

During execution maintain the relation:

$$w_{i,t} = \max\{0, \frac{e^{\eta(\alpha - b_{i,t+1})}}{e^{\eta\alpha}-1} - \frac{1}{e^{\eta\alpha}-1}\}$$

At each time $t = [T]$:

(a) Let $c_t \in [0,1]^n$ be the service cost vector.

(b) Set $b_{i,t+1} = b_{i,t} + c_{i,t}$

(c) Using binary search find the smallest a_t and set $b_{i,t+1} = b_{i,t} - a_t$ such that $\sum_{i=1}^{n} w_{i,t} = 1$

The algorithm increases the dual variables of the following LP and sets the primal variables accordingly. The authors then show that the dual solution constructed by the algorithm is feasible, and that the cost of the primal solution is bounded by the dual using the toolbox provided in [BN09].

The primal LP is the following:

$$\min \quad \sum_{t=1}^{T}\sum_{i=1}^{n} c_{i,t}w_{i,t} + \sum_{t=1}^{T}\sum_{i=1}^{n} \alpha z_{i,t}$$

$$\begin{aligned}
s.t. \quad \sum_{i=1}^{n} w_{i,t} &= 1 && \forall t \geq 0 \\
z_{i,t} &\geq w_{i,t} - w_{i,t-1} && \forall t > 1, \forall i \in [n] \\
w_{i,t} &\geq 0 && \forall t > 0, \forall i \in [n] \\
z_{i,t} &\geq 0 && \forall t > 1, \forall i \in [n]
\end{aligned}$$

And its dual whose variables the algorithm increases is:

$$\max \quad \sum_{t=0}^{T} a_t$$

$$\begin{aligned}
\text{s.t.} a_0 + b_{i,1} &\leq 0 && \forall i \in [n], t = 0 \\
b_{i,t+1} &\leq b_{i,t} + c_{i,t} - a_t && \forall i \in [n], t \geq 1 \\
0 \leq b_{i,t} &\leq \alpha && \forall i \in [n], t \geq 0
\end{aligned}$$

So far we have seen that despite their differences competitive analysis and on-line learning are characterized by a tight interrelationship. For most of the algorithms studied so far this boils down to appropriate interpolation of some parameters. However, we have not yet seen a candidate succeeding in deriving the result we hoped for. That is no algorithm discussed so far can achieve both sublinear regret and constant competitive ratio at the same time. The chase for such an optimistic result came to an end when the authors of [And+15] proved the following theorem:

Theorem 2.3.1. *There is no algorithm (randomized of deterministic) which can achieve sub-linear regret and constant competitive ratio in the SOCO setting, even when the cost functions are linear.*

After giving the hardness result, the authors provide a new algorithm with close to optimal bounds with respect to both metrics. The *Randomly Biased Greedy* algorithm achieves $(1+\gamma)$ competitive ratio and $O(\max\{T/\gamma, \gamma\})$ regret. Setting $\gamma = \sqrt{T}$ the algorithm guarantees $O(\sqrt{T})$ bounds regarding both the regret and the competitive ration. Opting for constant ratio and linear regret it suffices to set $\gamma = 1$. Intuitively, γ decides the distance be-tween the new and the previous point in the decision space which in turn defines the movement cost.

```
Randomly Biased Greedy

    Parameters: Norm N.

    Initialization: Set $w_0 = N(x)$.

    At each time $t = [T]$ :

     (a) $w_t(x) = \min_y \{w_{t-1}(y) + c_t(y) + N(x-y)\}$.
     (b) Generate a random number $r \in (-1, 1)$ uniformly.
     (c) Return $x_t = \arg\min_x \{w_{t-1}(x) + rN(x)\}$.
```

2.4 Common ground - The Regularization technique

Regularization is a common technique introduced in the field of on-line learning. As we mentioned earlier the learner in this setup has no knowledge about the future (0-lookahead) and thus a notion of stability is crucial in order to derive satisfying bounds. The general idea of regularization is to alter the objective function of an optimization problem in order to enforce stability on the optimal solution. The most common approach is to add some smooth and convex regularizing term to the objective function, such as the relative entropy or the Bregman divergence. Thus, sacrificing some approximation factor we end up with an overall more stable objective which is easier for the learner to track.

Moving beyond the boundary of on-line learning and coming back to competitive analysis we notice that any SOCO algorithm must try to mimic the configurations of the optimal off-line solutions on one hand while minimizing the movement cost on the other. Thus, it is clear that in some sense our algorithm has to maintain stability in a similar way to on-line learning. This high level connection allows us to successfully transfer the regularization technique to competitive analysis problems such as MTS problem [Abe+10] and On-line Set Cover with Service Cost [BCN14].

In the work of Buchbinder et al.[BCN14] the authors propose a general on-line algorithm for any SOCO problem that can be formulated as a covering LP i.e. the convex space of the decision points is a polyhedron that can be described by covering constrains in every round. Losing a $O(\log n)$ approximation factor with respect to the optimal off-line solution the algorithm returns on-line a fractional solution to the problem. Here n denotes the maximal sparsity of the covering constraints which in the worst case is as large as the number of variables. This work is based on regularization via relative entropy and primal dual techniques developed in this survey [BN09].

The example problem used in this paper is On-line Set Cover problem whose definition and LP formulation we have already stated. In order to understand how the aforementioned algorithm derives the logarithmic approximation we are going to need the primal and the dual relaxations of the problem:

$$
\begin{array}{lll}
(P)\ \ \min & \displaystyle\sum_{t=1}^{T} c_t y_t + \sum_{t=1}^{T}\sum_{i=1}^{n} w_i z_{i,t} & \\
\text{subject to} & \displaystyle\sum_{i\in S_{j,t}} y_{i,t} \geq 1 & \forall t \geq 1, 0 \leq j \leq m_t \\
& z_{i,t} \geq y_{i,t} - y_{i,t-1} & \forall t \geq 1, 1 \leq i \leq n \\
& z_{i,t}, y_{i,t} \geq 0 & \forall t, 1 \leq i \leq n
\end{array}
$$

$$
\begin{array}{llll}
(D) & \max & \sum_{t=1}^{T}\sum_{j=1}^{m_t} a_{j,t} & \\
 & \text{subject to} & b_{i,t} \le w_i & \forall i \forall t \ge 1 \\
 & b_{i,t+1} - b_{i,t} \le c_{i,t} - & \sum_{j | i \in S_{j,t}} a_{j,t} & \forall i \forall t \ge 1 \\
 & & a_{j,t}, b_{i,t} \ge 0 & \forall i, j \forall t \ge 1
\end{array}
$$

To use regularization in this case would mean to add a smooth function to the objective function $c_t y_t$ in each round to ensure that the distribution y_t will not change too much from the previous round. A natural function which is an indication of the resemblance of two distributions is the relative entropy function

$$\Delta(w \parallel u) = \sum_{i=1}^{n} w_i \ln \frac{w_i}{u_i} + u_i - w_i$$

The on-line regularization algorithm which is one of the two main results of the paper is the following:

Regularization Algorithm

Parameters: $\epsilon > 0,\ \eta = ln(1 + \frac{n}{\epsilon})$

Initialization:$y_{i,0} = 0 \ \ \forall i \in [n]$

At each time $t = [T]$:

(a) Let $c_t \in \mathcal{R}_{+}^{n}$ be the service cost vector and let P_t be the feasible set of solutions.

(b) Solve the following convex program to obtain y_t

$$y_t = \arg\min_{x \in P_t} \{c_t x + \tfrac{1}{\eta} \sum_{i=1}^{n} w_i ((x_i + \frac{\epsilon}{n}) \ln(\frac{x_i + \frac{\epsilon}{n}}{y_{i,t-1} + \frac{\epsilon}{n}}) - x_i)\}$$

Using the regularization introduced above we can isolate in each round t the instance of the problem and produce a new regularized LP P'.

$$
\begin{array}{llll}
(P^{REG}) & \min & c_t y_t + \frac{1}{\eta}\sum_{i=1}^{n} w_i((y_{i,t}+\frac{\epsilon}{n})\ln(\frac{y_{i,t}+\frac{\epsilon}{n}}{y_{i,t-1}+\frac{\epsilon}{n}}) - y_{i,t}) & \\
 & \text{subject to} & \sum_{i\in S_{j,t}} y_{i,t} \geq 1 & \forall t\geq 1, 0\leq j\leq m_t \\
 & & y_{i,t}\geq 0 & \forall t, 1\leq i\leq n
\end{array}
$$

In this new LP we note that the learning parameter η serves as a tuning parameter between service and movement cost. For large values of η the service cost is favored and the opposite holds true when η is small. The regularization function is almost the relative entropy of the current point and the previous point adjusted for the weighted case. One more difference is that the points are noisy, since we have added a small amount of noise uniformly distributed on the coordinates.

Before we present the related result given in [BCN14] we will provide a high level description. In each round t a new P^{REG} is constructed on-line using information only from rounds t and $t-1$. Since the new LP has convex objective function over a convex feasible region it is solvable in polynomial time using standard convex optimization techniques. Using K.K.T.−(Karash-Kuhn-Tucker)− optimality conditions for each of these, t in number, P^{REG}s the authors of this paper prove that the sequence of solutions produced on-line, constitute a feasible solution for (D) (defined above). Surprisingly, this solution is also proven to be within a $\log n$-multiplicative factor from the optimal solution of (D), delivering immediately, the desired result through weak duality. In order to present the proof we first have to derive the appropriate relations from the K.K.T optimality conditions. Thus, for each P^{REG} we define a Lagrangian variable $a_{j,t}$ for every covering constrain in the current round.

For all $1\leq j\leq m_t$,

$$\sum_{i\in S_{j,t}} y_{i,t} - 1 \geq 0 \tag{2.1}$$

$$a_{j,t}\left(\sum_{i\in S_{j,t}} y_{i,t} - 1\right) = 0 \tag{2.2}$$

Also for all $1\leq i\leq n$,

$$c_{i,t} + \frac{w_i}{\eta}\ln\left(\frac{y_{i,t}+\frac{\epsilon}{n}}{y_{i,t-1}+\frac{\epsilon}{n}}\right) - \sum_{j:i\in S_{j,t}} a_{j,t} \geq 0 \tag{2.3}$$

$$y_{i,t}\left(c_{i,t} + \frac{w_i}{\eta}\ln\left(\frac{y_{i,t}+\frac{\epsilon}{n}}{y_{i,t-1}+\frac{\epsilon}{n}}\right) - \sum_{j:i\in S_{j,t}} a_{j,t}\right) = 0 \tag{2.4}$$

Plugging in (D) the $a_{j,t}$ from the consecutive P^{REG} and setting $b_{i,t+1} = \frac{w_i}{\eta}\ln(\frac{1+\epsilon/n}{y_{i,t}+\epsilon/n})$ we create a feasible solution. To prove feasibility it is enough to see that

$$b_{i,t+1} - b_{i,t} = -\frac{w_i}{\eta}\ln(\frac{y_{i,t}+\epsilon/n}{y_{i,t-1}+\epsilon/n}) \leq c_{i,t} - \sum_{j:i\in S_{j,t}} a_{j,t}$$

where the inequality follows from 2.3. And also

$$0 \leq b_{i,t+1} = \frac{w_i}{\ln(1+\epsilon/n)}\ln(\frac{1+\epsilon/n}{y_{i,t}+\epsilon/n}) \leq w_i$$

since $0 \leq y_{i,t} \leq 1$.
As we stated above we will now proceed to bound the movement (M) and the service (S) cost of the constructed solution with respect to the optimal solution of (D).

$$M_t = \eta \sum_{y_{i,t}>y_{i,t-1}} \frac{w_i}{\eta}(y_{i,t}-y_{i,t-1}) \leq \eta \sum_{y_{i,t}>y_{i,t-1}} \left(y_{i,t}+\frac{\epsilon}{n}\right)\cdot\left(\frac{w_i}{\eta}\ln\left(\frac{y_{i,t}+\frac{\epsilon}{n}}{y_{i,t-1}+\frac{\epsilon}{n}}\right)\right) \tag{2.5}$$

$$= \eta \sum_{y_{i,t}>y_{i,t-1}} \left(y_{i,t}+\frac{\epsilon}{n}\right)\cdot\left(\sum_{j|i\in S_{j,t}} a_{j,t} - c_{i,t}\right) \tag{2.6}$$

$$\leq \eta\sum_{i=1}^{n}\left(y_{i,t}+\frac{\epsilon}{n}\right)\sum_{j|i\in S_{j,t}} a_{j,t} = \eta\sum_{j=1}^{m_t} a_{j,t}\left(\sum_{i\in S_{j,t}} y_{i,t} + \frac{\epsilon \mid S_{j,t}\mid}{n}\right) \tag{2.7}$$

$$\leq \eta\left(1+\frac{\epsilon k}{n}\right)\sum_{j=1}^{m_t} a_{j,t}. \tag{2.8}$$

$$S = \sum_{t=1}^{T}\sum_{i=1}^{n} c_{i,t}y_{i,t} = \sum_{t=1}^{T}\sum_{j=1}^{m_t} a_{j,t}\sum_{i\in S_{j,t}} y_{i,t} - \frac{1}{\eta}\sum_{t=1}^{T}\sum_{i=1}^{n} w_i\cdot y_{i,t}\ln\left(\frac{y_{i,t}+\frac{\epsilon}{n}}{y_{i,t-1}+\frac{\epsilon}{n}}\right) \tag{2.9}$$

$$= \sum_{t=1}^{T}\sum_{j=1}^{m_t} a_{j,t} - \frac{1}{\eta}\sum_{i=1}^{n} w_i\left\{\sum_{t=1}^{T}\left(y_{i,t+\frac{\epsilon}{n}}\right)\ln\left(\frac{y_{i,t}+\frac{\epsilon}{n}}{y_{i,t-1}+\frac{\epsilon}{n}}\right) - \frac{\epsilon}{n}\sum_{t=1}^{T}\ln\left(\frac{y_{i,t}+\frac{\epsilon}{n}}{y_{i,t-1}+\frac{\epsilon}{n}}\right)\right\} \tag{2.10}$$

$$\leq \sum_{t=1}^{T}\sum_{j=1}^{m_t} a_{j,t} - \frac{1}{\eta}\sum_{i=1}^{n} w_i\left\{\left(\sum_{t=1}^{T}\left(y_{i,t+\frac{\epsilon}{n}}\right)\right)\ln\left(\frac{\sum_{t=1}^{T}\left(y_{i,t}+\frac{\epsilon}{n}\right)}{\sum_{t=1}^{T}\left(y_{i,t-1}+\frac{\epsilon}{n}\right)}\right) - \frac{\epsilon}{n}\ln\left(\frac{y_{i,T}+\frac{\epsilon}{n}}{y_{i,0}+\frac{\epsilon}{n}}\right)\right\} \tag{2.11}$$

$$\leq \sum_{t=1}^{T}\sum_{j=1}^{m_t} a_{j,t} = \textit{ value of } (D). \tag{2.12}$$

Regarding the movement cost, inequality 2.5 follows as $a - b \leq a\ln(a/b)$, equality 2.6 and inequality 2.8 follow from the KKT optimality conditions. Respectively for the service cost equalities 2.9 and 2.10 derive from KKT conditions and inequality (2.7) follows by telescopic sum and by the fact that $\sum_i a_i \log(a_i/b_i) \leq (\sum_i a_i) \log(\frac{\sum_i a_i}{\sum_i b_i})$.

Summing the movement and the service costs over all rounds t and substituting η provides the logarithmic approximation.

It turns out that this technique can be generalized to a broader class of problems whose linear program apart from covering, contains constrains of the form $\sum_{i \in S_{i,t}} y_{i,t} \geq r_{j,t}$ or even precedence constrains $x \leq y$. Problems falling in this framework are facility location formulations, allocation problems and many others such as the On-line Shortest Path with Switching Cost which we will study further in Chapter 4.

Chapter 3

Exponential Clocks

3.1 Introduction to exponential clocks

In this chapter we are going to dive into a new rounding technique which has been applied with success in the framework of SOCO. First, we will define exponential clocks and we will point out some useful properties they have. Subsequently, we will go through some applications of exponential clocks and we will try to understand the underlying advantages and disadvantages of this technique in order to utilize it in an attempt to tackle the On-line Shortest Path with Switching Cost problem.

Definition 3.1.1. *By exponential clocks we mean competing independent exponential random variables. An exponential clock wins a competition if it has the smallest value among all participating exponential clocks. A random variable X is distributed according to the exponential distribution with rate λ if it has density $f_X(x) = \lambda e^{-\lambda x} \quad \forall x \geq 0$ and $f_X(x) = 0$ otherwise. We denote this by $X \sim exp(\lambda)$.*

Below we will give a high level description of the steps that an exponential clock rounding algorithm usually follows:

- We associate each variable y_i of our problem with a random value sampled in the beginning of our algorithm from the exponential distribution $X_i \sim \mathcal{E}xp(\lambda)$
- We then define the clock of each variable to be $\frac{X_i}{y_i}$ where y_i is the fractional value of the variable given by the relaxed linear program.
- The clocks are competing with each other or versus a threshold in order to be rounded. Smaller clocks are more likely to be rounded during our algorithm.

We are going to state some basic properties of exponential clocks:

- $F(x) = \begin{cases} 1 - e^{-\lambda x}, & x \geq 0 \\ 0, & \text{otherwise} \end{cases}$
- Let $X \sim \mathcal{E}xp(\lambda)$ then $\mathbb{E}[X] = \frac{1}{\lambda}$
- Let $X \sim \mathcal{E}xp(\lambda)$ and $c > 0$, then $\frac{X}{c} \sim \mathcal{E}xp(\lambda c)$
- Let $X_1, \ldots, X_k$ be independent random variables for which $X_i \sim \mathcal{E}xp(\lambda_i)$:
  (a) $\min\{X_1, ..., X_k\} \sim \mathcal{E}xp(\sum_{i=1}^{k} \lambda_i)$
  (b) $\Pr[X_i \leq min_{j \neq i}\{X_j\}] = \frac{\lambda_i}{\lambda_1 + \ldots + \lambda_k}$
- For any independent exponential random variables X, Y
  $Pr[X \leq Y | X \geq t] = \frac{\lambda_X}{\lambda_X + \lambda_Y} e^{-\lambda_Y t}$
- The probability to have two exponential clocks with the same value is zero.

Before we move on studying application we are going to give some underlying characteristic of exponential clocks which make them so useful in many combinatorial problems:

1. $\Pr[\min\{X_1/y_1, ..., X_k/y_k\} > a] < e^{-a\lambda \sum_{i=1}^{k} y_i}$. That means that if our relaxed LP picks a set of fractional values such that the sum of them is greater than 1 then our rounding algorithm (setting a as threshold) will have exponentially small chance to not round at least one of these variables.
2. The second property is related with on-line problems with switching cost. Since we are assigning a value Z_i to every random variable at the beginning of the algorithm, we are in fact defining a quantitative ordering over the variables. As a corollary our algorithm is desensitized to small fluctuations in the fractional solution.

The first application we are going to see is the classic Set Cover problem. The usage of exponential clocks in this simple example is an ideal way to introduce the reader to this technique. Recall that the *Set Cover* problem is defined as follows. Let $X = \{1, 2, ..., n\}$ be a ground set of n elements and let $\mathcal{S}$ be a family of subsets of X, $|\mathcal{S}| = m$. A cover is a collection of sets such that their union is X. Each $S \in \mathcal{S}$ has a nonnegative cost c_S associated with it. The goal is to find a cover of minimum cost.

And the respective linear program relaxation is :

$$\begin{array}{lll} \min & \sum_{S\in\mathcal{S}} c_t y_t & \\ \text{subject to} & \sum_{S:x\in X} y_S \geq 1 & \forall x \in X \\ & y_S \geq 0 & \forall S \in \mathcal{S} \end{array}$$

Set Cover Rounding Algorithm

- Initialization: $\forall S \in \mathcal{S}$ choose i.i.d. random variables $Z_S \sim \mathcal{E}xp(1)$.
- Output $\bigcup_{x\in X} \arg\min_{S:x\in\mathcal{S}}\{\frac{Z_S}{y_S}\}$

Notice that the smaller the value of Z_S and the bigger the fractional value y_S, the more likely it is for set S to be chosen. In a sense on one hand the exponential random variables are defining a quantitative ordering over the sets and on the other the more a set is chosen by the LP relaxation the better it is and thus we should include it with higher probability in our solution. We are going to show that given a fractional solution for the Set Cover, this algorithm rounds the solution suffering only $\ln(S_{max}) + 1$ approximation:

Let $A_{x,S}$ be the event that :

$$\frac{Z_S}{y_S} \leq min_{S'\in S}\{\frac{Z_{S'}}{y_{S'}}|x \in S', S' \neq S\}.$$

Then

$$
\begin{aligned}
Pr[S \in \mathcal{S} \textit{ is chosen }] &= \Pr[\exists x : A_{x,S}] \\
&= \begin{cases} \Pr[\exists x : A_{x,S} \mid Z_S/y_S < \alpha] \cdot \Pr[Z_S/y_S < \alpha] \\ + \\ \Pr[\exists x : A_{x,S} \mid Z_S/y_S > \alpha] \cdot \Pr[Z_S/y_S > \alpha] \end{cases} \\
&\leq \Pr[Z_S/y_S < \alpha] + Pr[\exists x : A_{x,S} \mid Z_S/y_S > \alpha] \cdot \Pr[Z_S/y_S > \alpha] \\
&= \begin{cases} \int_0^\alpha y_S e^{-y_S t} dt \\ + \\ Pr[\exists x : A_{x,S} \mid Z_S/y_S > \alpha] \int_a^\infty y_S e^{-y_S t} dt \end{cases} \\
&= (1 - e^{-y_S \alpha}) + Pr[\exists x : A_{x,S} | Z_S/y_S > \alpha] e^{-y_S \alpha} \\
&\leq 1 - e^{-y_S \alpha}) + e^{-y_S \alpha} \sum_{x \in S} \Pr[A_{x,S} | Z_S/y_S > \alpha] - (\textit{ by union bound }) \\
&\leq y_S \alpha + e^{-y_S \alpha} \sum_{x \in S} \frac{y_S}{\sum\limits_{S' : x \in S'} y_{S'}} e^{-\alpha(\sum\limits_{S' : x \in S'} (y_{S'}) - y_S)} - \left(\begin{Bmatrix} \textit{by exponential clock properties} \\ \textit{and} \\ e^x \geq 1 + x \end{Bmatrix} \right) \\
&\leq y_S(\alpha + |S| e^{-\alpha}) = y_S(\ln |S_{max}| + 1) - \textit{ (setting } \alpha = \ln |S_{max}| \textit{)} \\
&\leq y_S(\ln S_{max}) + 1)
\end{aligned}
$$

This proof is given in the appendix of [BNS13]. The main result of the paper is an exponential clocks based algorithm for the Multiway-Cut problem. The authors map the instance of the Multiway-Cut problem to a simplex of appropriate dimensions. Then, they randomly partition the simplex using exponential clocks, deriving a good approximate solution for the initial problem. Instead of focusing further on this problem, we will see three other more instructive applications.

3.2 Facility Location with stable intervals

Eisenstat et al. in [EMS14] introduced a Dynamic version of Facility Location with Switching Cost. Formally, we are given a set F of m *facilities* and a set C of n *clients* together with a finite sequence of distances $(d_t)_{1\leq t\leq T}$ over $F \times C$, a non-negative *facility opening cost* f and a non-negative *client switching cost* g. The goal is to output a subset $A \subseteq F$ of facilities and, for each time step $t \in [T]$, an assignment $\phi_t : C \to A$ of facilities to clients, so as to minimize:

$$f \cdot \sum_{1\leq t\leq T} \#A_t + \sum_{1\leq t\leq T, j\in C} d_t(\phi_t(j), j) + g \cdot \sum_{1\leq t<T} \sum_{j\in C} \mathbf{1}\{\phi_t(j) \neq \phi_{t+1}(j)\},$$

that is to say the sum of the service cost (f for each open facility in each time step), of the total *distance cost* to connect each client to its assigned facility at every time step, and of the *switching cost* for each client (g per change of facility per client).

A linear relaxation. For an integer programming formulation, we define indicator 0-1 variables y_i, x_{ij}^t, and z_{ij}^t for $i \in F$, $j \in C$, and $t \in [T]$: $y_i = 1$ iff facility i is open; $x_{ij}^t = 1$ iff client j is connected to facility i at time t; and $z_{ij}^t = 1$ iff client j is connected to facility i at time t but no more at time $t+1$. The dynamic facility location problem is then equivalent to finding an integer solution to the following linear programming relaxation.

$$\left[\begin{array}{lc} \min & f \cdot \sum_{1\leq t\leq T} \sum_{i\in A_t} y_i^t + \sum_{j\in C} \sum_{1\leq t\leq T} \sum_{i\in F} x_{ij}^t \cdot d_t(i,j) + g \cdot \sum_{j\in C} \sum_{1\leq t<T} \sum_{i\in F} z_{ij}^t \\ \text{such that :} & (\forall ijt)\ x_{ij}^t \leq y_i^t \\ & (\forall jt)\ \sum_{i\in F} x_{ij}^t = 1 \\ & (\forall ij,\ \forall t<T)\ z_{ij}^t \geq x_{ij}^t - x_{ij}^{t+1} \\ & y_i^t, x_{ij}^t, z_{ij}^t \geq 0 \end{array}\right]$$

The authors provide a randomized rounding algorithm proving the following theorem:

Theorem 3.2.1 (Hourly opening cost)**.** *There is a polynomial time randomized algorithm which outputs a solution to the dynamic facility location problem with hourly opening cost whose cost verifies:*

$$\Pr[cost \leq 4\log(2nT) \cdot \mathbf{OPT}] \geq \Pr[cost \leq 4\log(2nT) \cdot \mathbf{LP}] \geq 1/4.$$

Algorithm 1 Fixed opening cost

```
• Solve the linear program LP (3.2). Let $(x, y, z)$ be the solution obtained.
• Draw a facility at random $\Gamma = 2\log(2nT)\sum_{i\in F} y_i$ times independently, with distribution
  proportional to $y$; let $A$ be the resulting multiset of facilities.
for For each client $j$ do
    • Determine when it should change from one facility to another using the $z$-variables, and
    assign it to the cheapest selected facility between each change:
        (a) Partition time greedily into $\ell_j$ intervals $[t^j_k, t^j_{k+1})$ such that $t^j_1 = 1$ and where $t^j_{k+1}$
    is inductively defined as the largest $t \in (t^j_k, T+1]$ such that $\sum_{i\in F}\left(\min_{t^j_k \le u < t} x^u_{ij}\right) \ge 1/2$, and
    $t^j_{\ell_j+1} = T+1$;
        (b) For each time interval $[t^j_k, t^j_{k+1})$, connect $j$ to the facility in $A$ that is cheapest for $j$
    for that time interval.
end for
```

The algorithm's success is a product of the exponential clocks' properties and the clever definition of stable intervals for each client j. These intervals make sure that our algorithm will not reassign client j unless the fractional solution is guaranteed to pay at least $g/2$ switching cost. This novelty combined with a more accurate analysis actually derives better approximation on the follow up work in [ANS17].

Sabanne et al.[EMS14] utilize again exponential clocks in order to achieve constant approximation for the same problem. To do so they first execute two preprocessings. The first is heavily related to the notion of stable intervals as we defined them previously.

Lemma 3.2.1. *Given an LP solution, we can, by increasing its cost by at most a factor of 2, obtain in polynomial time a feasible solution (x, y, z) satisfying:*

- *If we let $Z^t = \{j \in C \mid x^t_{ij} \neq x^{t+1}_{ij} \text{ for some } i \in F\}$ denote the set of clients that changed its fractional connection between time step t and $t+1$, then $\sum_{t=1}^{T-1}|Z^t| \le \sum_{t=1}^{T-1}\sum_{i\in F, j\in C} z^t_{ij}$.*

The second preprocessing is obtained by using the standard trick of duplicating facilities, while being careful that if the connection variables of a client remain the same between two consecutive time steps, they remain so even after the preprocessing.

Observation 3.2.2. *Without loss of generality, we may assume that (x, y, z) satisfies the following:*

1. *For any facility $i \in F$, client $j \in C$, and time step $t \in [T]$, $x^t_{ij} \in \{0, y^t_i\}$.*
2. *For each facility $i \in F$, there exists $o_i \in [0, 1]$ such that $y^t_i \in \{0, o_i\}$ for each time step $t \in [T]$.*

Given a preprocessed solution (x, y, z) to the linear programming relaxation that satisfies the properties of Lemma 3.2.1 and Observation 3.2.2, our algorithm proceeds by first making a random choice and then opening facilities and connecting clients in each time step.

Rounding method of Svenson, Norouzi-Fard and An

Random choice: Sample independently an exponential clock $Q_i \sim \mathcal{E}xp(o_i)$ for each facility $i \in F$ and an exponential clock $R_j \sim \mathcal{E}xp(1)$ for each client $j \in C$.

Opening and connecting: At each time step $t \in [T]$, open facilities and connect clients as follows. Consider the clients in the non-decreasing order of their sampled clocks (R_j's). When client $j \in C$ is considered, find the facility $i = \arg\min_{i:x^t_{ij}>0} Q_i$ of the smallest clock among the facilities that j is connected to in the support of x^t. Similarly, let $j' = \arg\min_{j':x^t_{ij'}>0} R_{j'}$ be the client with the smallest clock in the neighborhood of i. The connection of j at time t is now determined as follows: if $j = j'$ then open i and connect j to i; otherwise, connect j to the same facility as j'.

Here the improvement lies on the choice of facility i where we assign client j in each stable interval. The exponential clocks again define a quantitative ordering between facilities and clients. This creates neighborhoods where a single facility dominates, meaning that all the clients in this region connect to it. This dominance keeps the solution produced by the algorithm stable deriving a constant bound with respect to the switching cost. However, it also allows distant connections which increases the service cost. Using the properties of exponential clocks and some combinatoric arguments the authors show that the probability that client j connects to facility i diminishes rapidly as the distance grows longer. Thus the main result of this paper is the following theorem.

Theorem 3.2.3. *There is a randomized* $14-$*approximation algorithm for the dynamic facility location problem.*

Summing up the takeaway points regarding the applications we have seen so far we can note the following:

- We have seen cases where the exponential clocks compete with each other in order to be selected and cases where specific thresholds determine the selection of the clocks.
- The introduction of stable intervals seems to be effective when the variables suffering switching cost add up to some a priory known value. However, the construction of these intervals requires lookahead which implies that this method can not be transferred immediately to some on-line setting like the one we are about to see next.

3.3 On-line Set Cover with Service Cost revisited

Let us shift to the on-line framework and recall the On-line Set Cover with Service Cost once more. Using regularization Buchbinder et al. presented an on-line $O(\log m)-$approximation algorithm which computes on the fly a fractional solution to this problem. Following the work of [BCN14] we will now see the on-line rounding algorithm proposed by the authors eventually deriving an $O(\log m \log n)-$approximation.

Rounding Algorithm for On-line Set Cover with Service Cost

```
1. parameter : $\alpha \geq 0$
2. $\forall S \in \mathcal{S}$ choose i.i.d. random variables $Z_S \sim \mathcal{E}xp(1)$.
3. $\forall e \in \mathcal{E}$, choose i.i.d. random variable $Z_e \sim \mathcal{E}xp(1)$
4. at any time step $t$, let $y_{S,t}$ denote the current fractional value of $S$.
5. **for** $t = 1, 2, ...T$ **do**
    (a) let $A_t = \{S \in \mathcal{S} \mid \frac{Z_S}{y_{S,t}} < \alpha\}$.
    (b) $B_t = \bigcup_{e \in \mathcal{E}} \Big\{S | S = \arg\min_{S':e \in S'} \{\frac{Z_{S'}}{y_{S',t}}\} \text{ and } \frac{Z_S}{y_S} < \frac{Z_e}{\max\{0, 1 - \sum_{S:e\in S} y_{S,t}\}}\Big\}$.
    (c) output $A_t \cup B_t$.
```

Here we notice that there exists a threshold α such that any clock greater than that, is included in A_t. For the rest of the clocks there is a competition related with each element e which decides which sets S will end up in B_t. The high level intuition related with sets A_t and B_t is that all the "nice" sets S are included immediately in the solution through A_t whereas B_t ensures that the final solution we will return, is indeed feasible.

The analysis bounds the service cost using standard techniques similar to the previous applications but for the switching cost a new approach is presented. The authors separate every time step from t to $t + 1$ into consecutive sub-steps where the fractional value of only one variable changes. Considering cases it is proven that every such change can not affect the expected cost too much, eventually deriving the desired $(\log n)-$ approximation. We are going to dive into more detail in chapter 4 where we are going to prove a similar bound for the On-line Shortest Path with Switching Cost.

Chapter 4

On-line Shortest Path with Switching Cost

4.1 Problem Formulation

Let $G = \{V, E\}$ be a directed weighted graph with no negative length cycles. Let also $|V| = n$ and $|E| = m$. We are giving firstly the integral LP formulation for the Dynamic Shortest $s-t$ Path with Switching Cost and afterwards we will present its fractional relaxation:

$$\min \sum_{t=1}^{T} \sum_{(i,j)\in E} c_{ij}^t x_{ij}^t + g \sum_{t=1}^{T} \sum_{(i,j)\in E} z_{ij}^t$$

$$s.t. \quad \sum_{(i,j)\in\delta^+(i)} x_{ij}^t - \sum_{(j,i)\in\delta^-(i)} x_{ji}^t = \begin{cases} 1, & \text{if } i = s \\ -1, & \text{if } i = t \\ 0, & \forall i \in V \setminus \{s,t\} \end{cases}$$

$$\begin{aligned} \sum_{(i,j)\in\delta^+(i)} x_{ij}^t &\leq 1 && \forall i \in V \\ z_{ij}^t &\geq x_{ij}^t - x_{ij}^{t-1} && \forall t, \forall (i,j) \in E \\ x_{ij}^t &\in \{0,1\} && \forall t, \forall (i,j) \in E \\ z_{ij}^t &\in \{0,1\} && \forall t, \forall (i,j) \in E \end{aligned}$$

Integral Version

$$
\begin{aligned}
&\min \sum_{t=1}^{T} \sum_{(i,j)\in E} c_{ij}^t x_{ij}^t + g \sum_{t=1}^{T} \sum_{(i,j)\in E} z_{ij}^t \\
&s.t. \sum_{(i,j)\in\delta^+(i)} x_{ij}^t - \sum_{(j,i)\in\delta^-(i)} x_{ji}^t = \begin{cases} 1, & \text{if } i = s \\ -1, & \text{if } i = t \\ 0, & \forall i \in V \setminus \{s,t\} \end{cases} \\
&\sum_{(i,j)\in\delta^+(i)} x_{ij}^t \leq 1 && \forall i \in V \\
&z_{ij}^t \geq x_{ij}^t - x_{ij}^{t-1} && \forall t, \forall (i,j) \in E \\
&x_{ij}^t \geq 0 && \forall t, \forall (i,j) \in E \\
&z_{ij}^t \geq 0 && \forall t, \forall (i,j) \in E
\end{aligned}
$$

Fractional Version

Interpreting the integral formulation we get that variables x_{ij}^t are indicators taking value either 0 or 1 depending on whether the corresponding edge (i, j) is included in the shortest path at time t. The c_{ij}^ts represent the weights or costs associated with each edge and z_{ij}^t capture the switching cost as usual. The constrains we use are the standard shortest paths constrains for the case where no negative length cycle exists, extended with the corresponding constrains for z_{ij}^t.

Regarding the linear programming relaxation we make the following observations :

- From the above constraints, it is implied that:

$$\sum_{(i,j)\in\delta^+(S)} x_{ij}^t \geq 1 \;\; \forall t, \;\; \forall S \text{ s.t. } : S \subset V \text{ and } s \in S \text{ and } t \notin S$$

- Furthermore, it is easy to verify that this formulation belongs in the framework provided by Buchbinder et al. in [**15**] and thus a fractional solution can be obtained on-line, via regularization, sacrificing an $O(\log m)$ approximation factor.

Given a fractional solution to the above LP we will proceed to round on-line the solution in each time step t losing an other logarithmic factor on the approximation. To this end we will present a new algorithm which taking advantage of the exponential clocks properties will return a feasible solution in a $O(\log n)-$approximation.

4.2 Rounding Algorithm for On-line Shortest Paths with switching Costs

Before we state the rounding algorithm we should try to deeply understand the problem we are trying to solve and its inherent structure. A couple of things differentiate this problem from the one we saw at the end of the previous chapter. First of all, obtaining feasibility for the rounded solution is not trivial in this problem. Let's just think for a minute the case where we opt in each round t to pick the edges whose exponential clock is higher than some threshold α. This would include all the "good" edges similarly to On-line Set Cover with Service Cost. However, this set of edges is far from providing a feasible solution. An attempt to restore feasibility, by making sure that every $s-t$ cut is bridged, is hindered by the exponentially big number of cuts. Even worse, there is a strong dependence between the edges that constitute an $s-t$ shortest path. Changing even one edge could trigger a chain of changes creating a new, totally different shortest path. The above argumentation poses a serious difficulty both in extending our set of edges to a feasible set but more importantly it makes our effort to bound the switching cost bleak.

The high level idea in the algorithm we propose is the following. In each round t we indeed opt to pay for the "good" edges of the graph i.e. the edges whose clock is smaller than some threshold α. Having these edges in the solution we set out to obtain feasibility as cheaply as possible. To this end we elect to run Dijkstra's algorithm on a carefully modified copy of our graph. Note that Dijkstra not only guarantees feasibility but also restricts, nicely, the number of cuts we have to consider. We are now left with figuring out the appropriate modification for our graph.

- All the vertices and the edges remain the same in the graph. Thus, there is no way to add or remove paths from s to t.

- Given that we have already paid for the "good" edges in the graph it is natural to let our Dijkstra to freely cross each one of them. With this reasoning we assign zero weight on each such edge.

- On the other hand considering the rest of the edges we prefer edges with higher fractional values and smaller exponential random variables i.e. the smaller the clock the more likely it should be for this edge to be included in the solution. We set the weight of each such edge to be the value of its corresponding clock and let Dijkstra to the rest.

```
ROUNDING ALGORITHM FOR ON-LINE SHORTEST PATHS WITH SWITCHING COST

1. Set Parameter: $a \geq 0$
2. $\forall (i,j) \in E$ choose i.i.d. random variable $Z_{ij} \sim \mathcal{E}xp(1)$
3. At any time step $t$ let $x_{ij}^t$ be the current fractional value of edge $(i,j)$.
4. for $t = 1, 2, .., T$ do:
    (a) Let $A_t = \{(i,j) \in E | \ \frac{Z_{ij}}{x_{ij}^t} < a\}$
    (b) Create the graph $G^t = \{V, E\}$ with the same vertices and edges as
        $G$ but different weights on the edges:
          • For any edge $(i,j) \in A^t$,  $w(i,j) = 0$.
          • For any other edge $(i',j') \notin A^t, w(i',j') = \frac{Z_{i'j'}}{x_{i'j'}^t}$.
    (c) Run Dijkstra in $G^t$ from $s$. Then let $B_t$ be the edges on the shortest
        path from $s$ to $t$ that are not included in $A_t$.
        (In order to simplify the analysis here we opt to define $B_t$ as the
        set of all edges −not in $A_t$− in the shortest path tree that Dijkstra
        will create reaching for $t$)
    (d) output $A_t \cup B_t$.
```

Theorem 4.2.1. ROUNDING ALGORITHM *achieves* $r = O(\log m \log n)$ *approximation ratio.*

Proof. Our goal is to bound the switching and service cost of our algorithm with respect to the quantities that the fractional solution will pay. Thus, we will achieve a multiplicative approximation which will extend to the total cost that the integral optimal solution will pay. Before we begin bounding the cost we will compute **the expected cardinality of set** B_t in any time step t

Lemma. $\mathbb{E}[|B_t|] \leq ne^{-a}$

Proof. We know that in order for Dijkstra to compute the shortest path from s to t it will face at most $n-1$, $s-t$ cuts and it will make a choice regarding which edge to pick to cross each cut. First, we will focus on one such cut S and will compute the probability that no edge from A_t crosses S.

$$\Pr\Big[\min_{(i,j)\in\delta^+(S)} \big(\frac{Z_{ij}}{x_{ij}^t}\big) > a\Big] = exp(-a \sum_{(i,j)\in\delta^+(S)} x_{ij}^t) \leq exp(-a) = e^{-a}$$

where the last inequality follows from the fact that $\sum_{(i,j)\in\delta^+(S)} x_{ij}^t \geq 1$ due to the feasibility constrain.

Let us now define the random variable $X_i^t = 1$ when no edge from A_t crosses the cut S_i and $X_i^t = 0$ otherwise. Let k be the number of cuts that Dijkstra will face while computing the shortest $s - t$ path. Thus we have:

$$E[\sum_{i=1}^{k} X_i^t] = \sum_{i=1}^{k} E[X_i^t] \ \leq ne^{-a}$$

□

Corollary. *It follows immediately that the expected switching cost with respect to B_t is at most ne^{-a} which is 1 for $a = \log n$.*

However, we want to replace this additive quantity with a multiplicative approximation. Thus, we will analyze more thoroughly how the changes in each x_{ij}^t will affect the expected switching cost of our algorithm. To this end between time steps $t-1$ and t we will consider m in between sub-steps so that in each of them only the fractional value of one of the edges can change. First, we will consider all the edges whose fractional value will increase and then those whose fractional value will decrease. Clearly, adding the switching and service cost over all mT sub-steps can only be greater than the total cost that our algorithm will pay since we may end up double counting some additions in the sets A and B. Also notice that while we are considering the fractional values that increase, A_{t-1} is a subset of the current set A and similarly while we are considering the fractional values that decrease, A_t is a subset of the current A. Thus the expected size of B conditioned on the additions or removals of specific edges in set A over the sub-steps can only be smaller than ne^{-a}.

Also let $Y_{(i,j),S}^t$ be the minimal clock in cut S other than the clock of edge (i,j) i.e.

$$Y_{(i,j),S}^t = \min_{(i',j') \in \delta^+(S) | (i',j') \neq (i,j)} \frac{Z_{i'j'}^t}{x_{i'j'}^t}$$

By the properties of the exponential clocks $Y_{(i,j),S}^t$ is an exponential random variable with :

$$\lambda = \sum_{(i,j) \neq (i',j')} x_{i',j'}^t \geq 1 - x_{i,j}^t$$

$\boxed{\text{Step \#A :}\textbf{Switching cost deriving from increasing fractional variables}}$

Consider edge $(i,j) \in E$ whose fractional value increases from x_{ij}^{t-1} to $x_{ij}^{t} = x_{ij}^{t-1} + \Delta$. The probability that this increase will result in one more addition in our set A_t is:

$$\begin{aligned}
\Pr\Big[(i,j) \in A_t \textit{ and } (i,j) \notin (A_{t-1} \cup B_{t-1})\Big] &\leq \Pr\Big[(i,j) \in A_t \textit{ and } (i,j) \notin A_{t-1}\Big] \\
&= \Pr\Big[\frac{Z_{ij}}{x_{ij}^{t-1} + \Delta} < a < \frac{Z_{ij}}{x_{ij}^{t-1}}\Big] \\
&= e^{-ax_{ij}^{t-1}} - e^{-ax_{ij}^{t-1}+\Delta} \\
&= e^{-ax_{ij}^{t-1}}(1 - e^{-a\Delta}) \leq a\Delta
\end{aligned}$$

This addition to set A can in turn trigger additions on set B. However the expected size of B conditioned on the event that $(i,j) \in A_t$ is smaller than ne^{-a} deriving expected switching cost $a\Delta ne^{-a}$.

$$\text{Let us now consider the event : } \{(i,j) \notin A_t, B_{t-1} \text{ and } (i,j) \in B_t\}$$

To bound the probability of the event above let us fist consider a specific $s-t$ cut S. The probability that no edge from A crosses this cut and the clock of (i,j) is the minimum only after the increase of its fractional value is below. Note that $Y_{ij,S}$ is referring to the current sub-step:

$$\begin{aligned}
\Pr[\frac{Z_{ij}}{x_{ij}^{t-1} + \Delta} < Y_{ij,S} < \frac{Z_{ij}}{x_{ij}^{t-1}} \textit{ and } \frac{Z_{ij}}{x_{ij}^{t-1} + \Delta} > a] &\leq \Pr[\frac{Z_{ij}}{x_{ij}^{t-1} + \Delta} < Y_{ij,S} < \frac{Z_{ij}}{x_{ij}^{t-1}} \textit{ and } Y_{ij,S} > a] \\
&= \int_a^\infty f_Y(k) \Pr[\frac{Z_{ij}}{x_{ij}^{t-1} + \Delta} < k < \frac{Z_{ij}}{x_{ij}^{t-1}}]dk \\
&= \int_a^\infty \lambda e^{-\lambda k}(e^{-x_{ij}^{t-1}k} - e^{-(x_{ij}^{t-1}+\Delta)k})dk \\
&= \frac{\lambda}{x_{ij}^{t-1} + \lambda} e^{-a(x_{ij}^{t-1}+\lambda)} - \frac{\lambda}{x_{ij}^{t-1} + \Delta + \lambda} e^{-a(x_{ij}^{t-1}+\Delta+\lambda)} \\
&\leq e^{-a} - \frac{1}{\Delta+1} e^{-a(\Delta+1)} \\
&= e^{-a}(1 - \frac{1}{\Delta+1} e^{-a\Delta}) \\
&\leq e^{-a}(1 - e^{-(a+1)\Delta}) \\
&\leq e^{-a}(a+1)\Delta
\end{aligned}$$

Since Dijkstra will face at most n such cuts from union bound we have that :

$$\Pr[(i,j) \notin A_t, B_{t-1} \cap (i,j) \in B_t] \leq ne^{-a}(a+1)\Delta$$

Conditioning on the addition of edge (i, j) to set B we can now bound from above the expected size of B with $1 + ne^{-a}$. An easy way to derive this is by assuming that edge (i, j) belongs to A. The current set A is a superset of A_t which implies that the expected size of B at the current sub-step is smaller than ne^{-a}. It follows that the conditioned expected size of B is at most $1 + ne^{-a}$.
Adding everything up we get that the expected switching cost is bounded above by

$$\begin{aligned} ne^{-a}(a+1)\Delta(1+ne^{-a}) &= ne^{-a}(a+1)\Delta + n^2e^{-2a}(a+1)\Delta \\ &= \Delta ne^{-a}(a+1+ne^{-a}(a+1)) \end{aligned}$$

Thus the total expected switching cost that the increase from x_{ij}^{t-1} to $x_{ij}^t = x_{ij}^{t-1} + \Delta$ is bounded above by

$$a\Delta + \Delta ne^{-a}(a+a+1+ne^{-a}(a+1)) = a\Delta + \Delta ne^{-a}(2a+1+ne^{-a}(a+1))$$

Step #B :**Switching cost deriving from decreasing fractional variables**

Consider edge $(i, j) \in E$ whose fractional value decreases from x_{ij}^{t-1} to $x_{ij}^t = x_{ij}^{t-1} - \Delta$. The probability that edge (i, j) belonged to set A_{t-1} and was removed is bounded above by :

$$\begin{aligned} \Pr[(i,j) \in A_{t-1} \ and \ (i,j) \notin A_t] &= \Pr[\frac{Z_{ij}}{x_{ij}^{t-1}} < a < \frac{Z_{ij}}{x_{ij}^{t-1} - \Delta}] \\ &= e^{-ax_{ij}^{t-1} - \Delta} - e^{-ax_{ij}^{t-1}} \\ &= e^{-ax_{ij}^{t-1}}(e^{a\Delta} - 1) \\ &\leq a\Delta \end{aligned}$$

This deletion from set A can in turn trigger additions in set B. However again the conditioned expected size of B is smaller than ne^{-a} deriving expected switching cost at most $a\Delta ne^{-a}$.

Let us now consider the event : $\{ \ (i,j) \notin A_t, B_t \ \text{ and } \ (i,j) \in B_{t-1} \ \}$.

To bound the probability of the event above let us first consider a specific $s - t$ cut S. The probability that no edge from A crosses this cut and the clock of (i, j) is the minimum only before the decrease of its fractional value is

$$\Pr[\frac{Z_{ij}}{x_{ij}^{t-1}} < Y_{ij,S} < \frac{Z_{ij}}{x_{ij}^{t-1} - \Delta} \ and \ \frac{Z_{ij}}{x_{ij}^{t-1}} > a] \leq \Pr[\frac{Z_{ij}}{x_{ij}^{t-1}} < Y_{ij,S} < \frac{Z_{ij}}{x_{ij}^{t-1} - \Delta} \ and \ Y_{ij,S} > a]$$

Following the same analysis as before we can upper bound this probability from $e^{-a}(a + 1)\Delta$ Using union bound on the $s - t$ cuts we get that $\Pr[(i, j) \notin A_t, B_t \ and \ (i, j) \in B_{t-1}] \leq ne^{-a}(a + 1)\Delta$

At this point we differentiate from the increasing case noticing that the conditional expected size of B is bounded above from ne^{-a} from which we derive that the expected switching cost is $n^2e^{-2a}(a+1)\Delta$

Thus the total expected switching cost that the decrease from x_{ij}^{t-1} to $x_{ij}^{t} = x_{ij}^{t-1} - \Delta$ is bounded above by $\Delta ne^{-a}(a + ne^{-a}(a+1))$

Summing up we get that the total expected switching cost of our algorithm suffers due to Δ movement in the fractional solution can be at most :

$$a\Delta + \Delta ne^{-a}(3a + 1 + 2ne^{-a}(a+1))$$

Having derived an appropriate multiplicative bound for the switching cost we now turn our attention towards the service cost of our algorithm in comparison with the service cost paid by fractional solution.

Now we will bound the cost that our algorithm suffers because of the set A_t, B_t respectively.

Step #C :**Service cost with respect to A_t**

For each time step t for each edge (i,j) with fractional value x_{ij}^t the probability that it will be chosen in A_t is:

$$\Pr[(i,j) \in A_t] = \Pr[\frac{Z_{ij}}{x_{ij}^t} < a] = 1 - e^{-x_{ij}^t a} \leq x_{ij}^t a$$

Thus the service cost due to the edges in A_t is at most a times the service cost that the fractional solution pays.

Step #D :**Service cost with respect to B_t.**

For each time step t each edge $(i,j) \in E$ is chosen in B_t due to a specific cut S that Dijkstra faces with probability:

$$\begin{aligned}\Pr\left[\frac{Z_{ij}}{x_{ij}^t} < \min_{(i',j')\in\delta^+(S)|(i',j')\neq(i,j)} \frac{Z_{i'j'}}{x_{i'j'}^t}\middle|\frac{Z_{ij}}{x_{ij}^t} \geq a\right] \Pr[\frac{Z_{ij}}{x_{ij}^t} \geq a] &= \frac{x_{i,j}^t}{\sum\limits_{(i,j)\in\delta^+(S)} x_{ij}^t} exp(-a \sum_{(i',j')\neq(i,j)} x_{i'j'}^t)e^{-ax_{ij}^t} \\ &\leq x_{ij}^t exp(-a \sum_{(i,j)\in\delta^+(S)} x_{ij}^t) \\ &\leq x_{ij}^t e^{-a}\end{aligned}$$

For the first equality we used that :

$$\Pr_{Z,Y \sim \mathcal{E}xp(\lambda_{Z,Y})} \Big[Z < Y | Z \geq c \Big] = \frac{\lambda_Y}{\lambda_Z + \lambda_Y} e^{-c\lambda_Y}$$

for Z, Y independent exponential random variables. The inequality is following again from the feasibility constrain $\sum_{(i,j) \in \delta^+(S)} x_{ij}^t \geq 1$.
Thus the expected service cost due to B_t is at most ne^{-a} times the service cost of the optimal solution, since Dijkstra will face at most n cuts.
Setting appropriately $a = \log n$ we get that the multiplicative factor our algorithm suffers with respect to the switching cost is :

$$a + ne^{-a}(3a + 1 + 2ne^{-a}(a + 1)) = \log n + 5 \log n + 3 = 6 \log n + 3$$

Clearly the respective factor related with the service cost is smaller:

$$a + ne^{-a} = \log n$$

Finally, let $Cost^{alg}$ denote the total cost our algorithm pays and $Cost^{frac}$ and $Cost^{int}$ the respective costs for the optimal fractional solution and the optimal integral solution.Then we can state the following:

$$\boxed{\mathbb{E}[Cost^{alg}] \leq (6 \log n + 3) Cost^{frac}}$$

$\square$

Chapter 5

Conclusion

5.1 Summing up

In this dissertation our main focus was set on on-line settings where a learner has to make decisions in the face of uncertainty. We defined the class of On-line Convex Optimization problems and its generalization Smooth On-line Convex Optimization. We introduced the reader to competitive analysis and on-line learning. We pointed out their underlining differences, we explored the boundaries between the two areas and stated a fundamental incompatibility result. We then explained how the regularization technique can be transferred from on-line learning to competitive analysis deriving innovating result.

We then approached a recent rounding method called exponential clocks. We explored different applications such as the classic and the On-line Set Cover with Service Cost and the Off-line Dynamic Facility Location. We concluded identifying some nice properties of exponential clocks which eventually we utilize in order to get a new result.

In the last part of our work we define a natural problem which belongs in the class of Smooth On-line Convex Optimization, namely On-line Shortest Path with Switching Cost. We take advantage of the existing framework in the literature in order to obtain, on-line, a fractional solution within an $O(\log m)$ multiplicative factor from the optimal fractional solution. As a final step we device a new rounding algorithm that works on-line and guarantees providing an integral solution to our problem with total approximation ratio $O(\log m \log n)$.

Bibliography

[Abe+10] Jacob D. Abernethy et al. "A Regularization Approach to Metrical Task Systems". In: *Algorithmic Learning Theory, 21st International Conference, ALT 2010, Canberra, Australia, October 6-8, 2010. Proceedings*. 2010, pp. 270–284.

[Alo+03] Noga Alon et al. "The online set cover problem". In: *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*. 2003, pp. 100–105.

[And+15] Lachlan L. H. Andrew et al. "A Tale of Two Metrics: Simultaneous Bounds on Competitiveness and Regret". In: *CoRR* abs/1508.03769 (2015). URL: http://arxiv.org/abs/1508.03769.

[ANS17] Hyung-Chan An, Ashkan Norouzi-Fard, and Ola Svensson. "Dynamic Facility Location via Exponential Clocks". In: *ACM Trans. Algorithms* 13.2 (2017), 21:1–21:20.

[AT96] M. Asawa and D. Teneketzis. "Multi-armed bandits with switching penalties". In: *IEEE Transactions on Automatic Control* 41.3 (Mar. 1996), pp. 328–348.

[Ban+03] Nikhil Bansal et al. "Online oblivious routing". In: *SPAA 2003: Proceedings of the Fifteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, June 7-9, 2003, San Diego, California, USA (part of FCRC 2003)*. 2003, pp. 44–49.

[BB00] Avrim Blum and Carl Burch. "On-line Learning and the Metrical Task System Problem". In: *Machine Learning* 39.1 (2000), pp. 35–58.

[BB97] Avrim Blum and Carl Burch. "On-line Learning and the Metrical Task System Problem". In: *Proceedings of the Tenth Annual Conference on Computational Learning Theory, COLT 1997, Nashville, Tennessee, USA, July 6-9, 1997*. 1997, pp. 45–53.

[BBK99] Avrim Blum, Carl Burch, and Adam Kalai. "Finely-Competitive Paging". In: *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*. 1999, pp. 450–458.

[BBN10] Nikhil Bansal, Niv Buchbinder, and Joseph Naor. "Towards the Randomized k-Server Conjecture: A Primal-Dual Approach". In: *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*. 2010, pp. 40–55.

[BCN14] Niv Buchbinder, Shahar Chen, and Joseph Naor. “Competitive Analysis via Regularization”. In: *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*. 2014, pp. 436–444.

[BE98] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.

[BLS92] Allan Borodin, Nathan Linial, and Michael E. Saks. “An Optimal On-Line Algorithm for Metrical Task System”. In: *J. ACM* 39.4 (1992), pp. 745–763.

[BN09] Niv Buchbinder and Joseph Naor. “The Design of Competitive Online Algorithms via a Primal-Dual Approach”. In: *Foundations and Trends in Theoretical Computer Science* 3.2-3 (2009), pp. 93–263. URL: https://doi.org/10.1561/0400000024.

[BNS13] Niv Buchbinder, Joseph Naor, and Roy Schwartz. “Simplex partitioning via exponential clocks and the multiway cut problem”. In: *Symposium on Theory of Computing Conference, STOC’13, Palo Alto, CA, USA, June 1-4, 2013*. 2013, pp. 535–544.

[Buc+16] Niv Buchbinder et al. “Unified Algorithms for Online Learning and Competitive Analysis”. In: *Math. Oper. Res.* 41.2 (2016), pp. 612–625.

[CL06] Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

[CO96] Thomas M. Cover and Erik Ordentlich. “Universal portfolios with side information”. In: *IEEE Trans. Information Theory* 42.2 (1996), pp. 348–363.

[EMS14] David Eisenstat, Claire Mathieu, and Nicolas Schabanel. “Facility Location in Evolving Metrics”. In: *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*. 2014, pp. 459–470.

[Fot06] Dimitris Fotakis. “Incremental algorithms for Facility Location and k-Median”. In: *Theor. Comput. Sci.* 361.2-3 (2006), pp. 275–313.

[FS97] Yoav Freund and Robert E. Schapire. “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting”. In: *J. Comput. Syst. Sci.* 55.1 (1997), pp. 119–139.

[GM09] Sudipto Guha and Kamesh Munagala. “Multi-armed Bandits with Metric Switching Costs”. In: *Automata, Languages and Programming, 36th Internatilonal Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part II*. 2009, pp. 496–507.

[Gor99] Geoffrey J. Gordon. “Regret Bounds for Prediction Problems”. In: *Proceedings of the Twelfth Annual Conference on Computational Learning Theory, COLT 1999, Santa Cruz, CA, USA, July 7-9, 1999*. 1999, pp. 29–40.

[GTW14] Anupam Gupta, Kunal Talwar, and Udi Wieder. “Changing Bases: Multistage Optimization for Matroids and Matchings”. In: *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*. 2014, pp. 563–575.

[JV11] Vinay Joseph and Gustavo de Veciana. “Variability Aware Network Utility Maximization”. In: *CoRR* abs/1111.3728 (2011). arXiv: `1111.3728`. URL: `http://arxiv.org/abs/1111.3728`.

[KV02] Adam Kalai and Santosh Vempala. “Efficient Algorithms for Universal Portfolios”. In: *Journal of Machine Learning Research* 3 (2002), pp. 423–440.

[KW97] Jyrki Kivinen and Manfred K. Warmuth. “Exponentiated Gradient Versus Gradient Descent for Linear Predictors”. In: *Inf. Comput.* 132.1 (1997), pp. 1–63.

[Lin+12a] Minghong Lin et al. “Online algorithms for geographical load balancing”. In: *2012 International Green Computing Conference, IGCC 2012, San Jose, CA, USA, June 4-8, 2012*. 2012, pp. 1–10.

[Lin+12b] Minghong Lin et al. “Online optimization with switching cost”. In: *SIGMETRICS Performance Evaluation Review* 40.3 (2012), pp. 98–100.

[Lin+13] Minghong Lin et al. “Dynamic Right-Sizing for Power-Proportional Data Centers”. In: *IEEE/ACM Trans. Netw.* 21.5 (2013), pp. 1378–1391.

[Lov75] L. Lovász. “On the ratio of optimal integral and fractional covers”. In: *Discrete Mathematics* 13.4 (1975), pp. 383–390.

[RS13] Alexander Rakhlin and Karthik Sridharan. “Online Learning with Predictable Sequences”. In: *COLT 2013 - The 26th Annual Conference on Learning Theory, June 12-14, 2013, Princeton University, NJ, USA*. 2013, pp. 993–1019.