MSc Thesis

Online Facility Location with Switching Costs

Lydia Zakynthinou

$\mu\Pi\lambda\forall$
Graduate Program in Logic, Algorithms and Computation
National and Kapodistrian University of Athens

Supervised by:
Dimitris Fotakis

Advisory Committee:

Dimitris Fotakis **Aris Pagourtzis** **Stathis Zachos**

November 2017

Acknowledgements

First of all, I deeply thank my advisor D. Fotakis for his continuing guidance and support and making me love theory way before this thesis had started taking place. I would also like to thank Prof. Zachos, Prof. Pagourtzis, and all the CoReLab members for the unique nurturing environment they have created. I also thank my parents and friends for their unconditional support. Last but not least, I need to express how grateful I am for all my professors during my studies in MPLA, who inspired me in each and every class I have taken. I consider myself very lucky to have been a part of this program.

Abstract

Online decision making is a large research area whose literature includes many different aspects and approaches. The problems it studies are based on the following setting. There is a decision-maker who has to make a decision iteratively with no knowledge of the future and receive the cost of their decision in each round. The goal is to perform well over time. Depending on the definition of what consists of a good performance, that is the benchmark to which we compare our algorithm's total cost, and on the assumptions made, different kinds of problems occur. A particularly interesting benchmark which captures many real life problems where the environment changes over time, is a solution which balances the trade-off between the optimal costs in each round and its stability. Online learning and competitive analysis are two frameworks which study problems in this setting. In this thesis we will discuss the differences between these two frameworks, the efforts to unify them and finally we will demonstrate how such a unifying approach can give a good approximation algorithm for the online facility location problem with switching costs, which falls into this general setting.

Keywords: online decision-making; online learning; online linear optimization; online convex optimization; competitive analysis; metrical task systems; facility location; switching costs; dynamic settings;

Contents

Chapter 1

Introduction

In the online learning framework, we consider the problem of a decision-maker who has to pick an action from an action set in every round. The decision-maker chooses her action, and a cost associated with that action is incurred. The costs can be assumed to arrive in an adversarial way and usually the decision-maker learns more information about the costs of the round than just what cost corresponds to her action. Since obviously we can not aim to minimize the total cost of our actions, the goal is to minimize the difference between our incurred total cost and the total cost of the best fixed action. This notion is called *regret* and is the most common measure of performance for online learning. It comes in several variations such as adaptive regret, which we will discuss in later chapters.

It is obvious that this benchmark is weak for many problems, since it may be the case that no single action performs well overall. Also, the cost incurred in each round depends only on this round's chosen action, although many similar online problems do not fit in this particular model. For example, a simple scenario would be to consider a problem where the cost we suffer in each round depends on the action we choose as well as the state we are (which depends on our previous actions). Another common feature is to add a *switching* cost (sometimes called a *movement* cost) in the online learning setting. For example, consider having to schedule processes which arrive online on a set of servers with different costs. In addition to this cost, an additional switching cost is incurred whenever you decide to assign the current procedure on a different server than the one currently running (imagine this is the cost of turning off and on servers). This also is a setting which is not modeled well by the online learning framework as we have described it so far.

This is where competitive analysis fits in. In competitive analysis, we

assume a more general framework, where the cost function is not as constricted and where the goal is to compete against an optimal offline solution. Regret is not the common measure of performance in this case. The measure of performance is the more well known *competitive* or *approximation* ratio between the total cost of an algorithm and the total cost of the optimal offline solution, which is the minimum total cost. Obviously, this is a harder performance criterion but there are several algorithms which yield a good competitive ratio. Finally, an assumption that differentiates the two areas is the assumption of *0-lookahead* and *1-lookahead*. In competitive analysis, the assumption is that we know the cost function in the current round before choosing our action, while in online learning we know the costs only after we have made our choice.

In later chapters, we will examine some settings that may differ slightly from one another but the main structure of the problem is the following. We assume that in each round we have to return an action (one could think of it as a solution to a problem), knowing the current cost function of the round, that is assuming 1-lookahead. The cost incurred in each round includes, aside from the cost of the action, a switching cost. The algorithms we will study will analyse their performance either with respect to the competitive ratio or the regret or both.

Chapter 2

Online Optimization with Switching Costs

In this chapter, we will make an introduction to the large area of Online Optimization. We will consider some of the first and most important approaches to unifying the two areas of Online Learning and Competitive Analysis and explain their results. We will then describe some of the more recent work in the area where very interesting techniques are employed. Some of these techniques will be applied to the Online Facility Location problem with Switching Costs in Chapter 4.

2.1 Online Learning and Metrical Task Systems

We start exploring the area between the two frameworks with a combination of the well known and studied problem of Experts in Online Learning ([24]) and Metrical Task Systems (MTS) ([12]) in Competitive Analysis. This is a simple setting and it provides a good ground for more complex problems.

The problem of Experts and MTS with a uniform metric has been studied extensively. In [10], which was one of the first attempts towards this direction, the authors prove that by tuning the parameters of known algorithms one can achieve fairly low regret and high competitive ratio. The notion of α-unfairness is used in this case, which essentially makes the solution we compete against to pay α times more for the switching cost than our algorithm. This notion captures well the trade-off between the two different frameworks, since for $\alpha = 1$ the competing solution is the optimal as it is considered in competitive analysis and for $\alpha \to \infty$ the benchmark solution consists of the best fixed policy in hindsight, just like in the online learning framework. Interpolating between different values of α one can interpolate between the algorithms which yield good results with respect to the desired benchmark.

However, this interpolation means that we can not use the same algorithm with the same parameters to achieve low regret and high competitive ratio simultaneously. Towards the search for an overall good algorithm, there have been several papers, such as [8], which provide good problem-specific algorithms but it wasn't until [13] that a simple unifying approach which can also be extended to more complex problems has been proposed. The authors of the latter paper, using primal-dual techniques from competitive analysis described in this survey [11], provide a unifying approach with good performance with respect to the regret and the competitive ratio which allows them to extend their results to more interesting settings.

We will now describe more formally the setting. 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 online 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 (or better the experts), which implies what action she should choose next. Since in online 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 over all rounds can be formulated as:

$$S_0 = \sum_{t=1}^{T} \mathbf{w}_{t-1}\mathbf{c}_t$$

Regret, which our goal is to minimize, is then formulated as follows:

$$\sum_{t=1}^{T} \mathbf{w}_{t-1}\mathbf{c}_t - \sum_{t=1}^{T} \mathbf{w}^*\mathbf{c}_t$$

where $\mathbf{w}^* = \arg\min_{w \geq 0, \|w\|_!=1} \sum_{t=1}^{T} \mathbf{w}\mathbf{c}_t$ is the fixed distribution over the experts with the minimum total cost.

In this in-between area, a notion that is very useful is that of *shifting* and *drifting* experts. In the case of shifting experts, we want to minimize the difference $\sum_{t=1}^{T} \mathbf{w}_{t-1}\mathbf{c}_t - \sum_{t=1}^{T} \mathbf{w}_t^*\mathbf{c}_t$ where $\mathbf{w}_t^*$ for $t \in \{0, \ldots, T-1\}$ is the optimal sequence of actions which changes at most k times. Drifting experts capture an even more general notion which generalizes shifting experts as well. The goal in this case is to minimize the optimal sequence under the constraint that $\sum_{t=1}^{T} \frac{1}{2}\|\mathbf{w}_t^* - \mathbf{w}_{t-1}^*\|_1 \leq k$, which restricts in some sense the distance of the movement of the distribution.

In the context of competitive analysis however, we assume 1-lookahead so the total cost for the distribution we choose in each round is:

$$S_1 = \sum_{t=1}^{T} \mathbf{w}_t\mathbf{c}_t$$

In this setting, we also suffer a switching cost

$$M = \sum_{t=1}^{T} \frac{1}{2}\|\mathbf{w}_t^* - \mathbf{w}_{t-1}^*\|_1$$

Finally, in contrast to online learning, we compare our results to an optimal sequence $\mathbf{w}_1^*, \ldots, \mathbf{w}_T^*$ which minimizes the total cost:

$$\sum_{t=1}^{T} \mathbf{w}_t\mathbf{c}_t + \sum_{t=1}^{T} \frac{1}{2}\|\mathbf{w}_t^* - \mathbf{w}_{t-1}^*\|_1$$

and has S_1^* total service cost and M^* total switching cost.

If we denote the optimal total cost as $OPT = S_1^* + M^*$ then we define the competitive ratio to be the minimum c such that

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

where d is a constant that does not depend on T.

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 $OPT(\alpha)$.

As in the paper we discuss, we make some preliminary observations about the relationships between these cost definitions.

- If OPT_k is the optimal k-drifting sequence, then $OPT(\alpha) \leq OPT_k + \alpha k$, since the optimal α-unfair sequence is the minimum over all α-unfair sequences, including the optimal k-drifting sequence which has a bounded switching cost by k.
- Because the service costs are bounded, it holds $S_0 \leq S_1 + M$. We can imagine that we switch every distribution with the previous one and suffer the 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.

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

This gives us a good idea about how the costs of the different settings are related. The algorithm that the paper proposes for the Experts/MTS problem is the following:

EXPERTS/MTS (ONLINE LEARNING FORMULATION)
Parameters: $\alpha \geq 1$, $\eta > 0$
Initialization: Set $w_{i,0} = \frac{1}{n}$ for all i.
At each time $t = [T]$**:**

1. Let $\mathbf{c}_t \in [0,1]^n$ be the service cost vector.
2. Using binary search, find the smallest a_t such that $\sum_{i=1}^{n} w_{i,t}$ is a distribution and

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

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

$$S_1 \leq OPT(\alpha) + \frac{\ln n}{\eta}$$

$$M \leq \left(1 + \frac{n}{e^{\eta\alpha} - 1}\right)(\eta OPT(\alpha) + \ln n)$$

For $\alpha \to \infty$ where we pointed out earlier that the setting approaches the classic online learning setting with experts, it holds that $S_0 \leq S_1 + M \leq (1+\eta)OPT(\infty) + \frac{\ln n}{\eta} + \ln n$ 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 also formulated equivalently in a primal-dual context. Aside from the fact that it helps with the analysis, this equivalence also indicates how these two areas we discuss are indeed connected.

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

EXPERTS/MTS (PRIMAL-DUAL FORMULATION)
Parameters: $\alpha \geq 1$, $\eta > 0$
Initialization: Set $w_{i,0} = \frac{1}{n}$, $b_{i,1} = \alpha - \frac{\ln(\frac{e^{\eta\alpha}+n-1}{n})}{\eta}$ for all $i \in \{1, \ldots, n\}$.
During execution maintain the relation: $w_{i,t} = \max\left\{0, \frac{e^{\eta(\alpha - b_{i,t+1})}}{e^{\eta\alpha}-1} - \frac{1}{e^{\eta\alpha}-1}\right\}$.
At each time $t = [T]$**:**

1. Let $\mathbf{c}_t \in [0,1]^n$ be the service cost vector.
2. Set $b_{i,t+1} = b_{i,t} + c_{i,t}$.
3. Using binary search, find the smallest a_t and set $b_{i,t+1} = b_{i,t+1} - a_t$ such that $\sum_{i=1}^n w_{i,t} = 1$.

The primal LP implied by the algorithm is the following:

$$\begin{array}{lll}
\min & \sum_{t=1}^T \sum_{i=1}^n c_{i,t} w_{i,t} + \sum_{t=1}^T \sum_{i=1}^n \alpha z_{i,t} & \\
\text{s.t.} & \sum_{i=1}^n w_{i,t} = 1 & \forall t \geq 0 \\
& z_{i,t} \geq w_{i,t} - w_{i,t-1} & \forall t \geq 1, \forall i \in [n] \\
& w_{i,t} \geq 0 & \forall t \geq 0, \forall i \in [n] \\
& z_{i,t} \geq 0 & \forall t \geq 1, \forall i \in [n]
\end{array}$$

And its dual whose variables the algorithm increases is:

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

The algorithm increases the dual variables of this LP and sets the primal variables accordingly, creating a feasible dual solution which bounds the primal solution's cost, inspired by techniques in [15].

2.2 Matroids

This primal-dual approach to online learning with switching costs is general enough to be applied to more complex problems, meaning that in each round we are given a service vector but the cost of the round is not just a linear combination on the actions. Instead, we can imagine that we are given the costs for an optimization problem (for example k-server ([8], [7]) or set cover ([2])), and we return a feasible solution in each round.

To get from the simple experts/MTS setting to one such as the one we just described, we are going to consider how using the same technique on matroids works in a similar way.

Matroids have a structure which can be used to describe many optimization problems. Let E be a finite set (instead of the action set), and I a collection of subsets of E, which are called *independent* sets. The structure $\mathcal{M} = (E, I)$ is called a matroid if it satisfies the following properties:

- If $S_1 \subseteq S_2$ and $S_2 \in I$ then $S_1 \in I$.
- If $S_1, S_2 \in I$ and $|S_1| > |S_2|$ then $\exists e \in S_1 \setminus S_2$ such that $S_2 \bigcup \{e\} \in I$.

The *base* B of $S \subseteq E$ is a maximal independent subset of S and all bases of S have the same size $r(S)$. By $B(\mathcal{M})$ we denote the convex hull of the incidence vectors of the bases of $\mathcal{M}$, namely the *base polytope* of $\mathcal{M}$. Finally, the *density* of a matroid is $\gamma(\mathcal{M}) = \max_{S \subseteq E, S \neq \emptyset} \{|S|/r(S)\}$.

To make the connection to the previous setting more clear, we think of the following equivalence between the elements of the problem. The algorithm maintains a distribution $\mathbf{w}_t \in B(\mathcal{M})$ over the elements of E, which correspond to the actions in the previous setting. As before, where we wanted to adjust the weights with the guarantee that $\mathbf{w}_t$ would remain a distribution (adding up to 1), now we want to adjust the weights with the guarantee that $\mathbf{w}_t$ would remain a fractional base (adding up to its rank). The algorithm in the online learning formulation is the following:

MATROIDS (ONLINE LEARNING FORMULATION)
Parameters: $\alpha \geq 1$, $\eta > 0$
Initialization: Find a fractional base $\mathbf{w}_0 \in B(\mathcal{M})$ such that $w_{e,0} \leq \frac{1}{\gamma(\mathcal{M})}$ for each $e \in E$ (equivalent to $\frac{1}{n}$ for the experts).
At each time $t = [T]$**:**

1. Let $\mathbf{c}_t \in [0,1]^n$ be the service cost vector.
2. For each $e \in E$, set $w_{e,t} = (w_{e,t-1} + \frac{1}{e^{\eta\alpha}-1})e^{-\eta c_{e,t}} - \frac{1}{e^{\eta\alpha}-1}$.
3. As long as $\sum_{e \in E} w_{e,t} < r(E)$: Let S be the set of elements that do not belong to a tight set. For each $e \in S$, update $w_{e,t} = (w_{e,t} + \frac{1}{e^{\eta\alpha}-1})e^{-\eta a_{S,t}} - \frac{1}{e^{\eta\alpha}-1}$ where $a_{S,t}$ is the smallest value such that there exists $e \in S$ to join a tight set.

We can see how the matroids generalize the previous problem, in the sense that the elements are the actions/experts, a base could be any single element and a fractional combination of the bases lives in the convex hull Δ_n. The rank of the element set is 1 and the density is n. With these observations, it is easy to see how the following performance guarantee of the algorithm for the matroid setting can be applied to the experts/MTS.

$$S_1 \leq OPT(\alpha) + \frac{r(E)}{\eta}\ln(\gamma(\mathcal{M})) + \frac{n\alpha}{e^{\eta\alpha}-1}$$

$$M \leq \Big(1 + \frac{n - r(E) + 1}{e^{\eta\alpha}-1}\Big)(\eta OPT(\alpha) + \ln(\gamma\mathcal{M}))$$

The algorithm we presented from [13] returns a distribution over the matroid bases, i.e. a fractional base, in each round and measures its expected cost. However, optimization problems usually require an integral solution in each round. For example, the minimum spanning tree problem, which can be formulated in the matroid setting, would require that in every round the algorithm returns a spanning tree and not a distribution over all possible spanning trees. Accordingly, the switching cost would be different. Therefore, algorithms for these settings usually require a rounding technique to be applied along with the online algorithm which returns the distribution.

A more complete approach for the matroid setting is given in [26], where the authors provide an algorithm for the Multistage Matroid Maintenance problem, which is the maintenance of a matroid base in each round with switching costs, and their results are optimal unless $P = NP$. Also, their approach works for non-uniform switching costs as well, in contrast to the

approach we just presented. More specifically, by reducing the online multistage matroid maintenance problem to the online multistage spanning set maintenance problem, and solving the latter, they provide a $O(\log |E| \log rT)$ where E is the matroid element set and r is its rank. When the switching costs are uniform, the factor T is removed from the competitive ratio. They also prove that this is the optimal approximation ratio. It is worth mentioning however that no measure of regret is analysed in this case.

2.3 Online Convex Optimization

We are now going to generalize the problem a bit more. We consider the following formulation which describes a problem in Online Convex Optimization. In each round, a learner must choose an action $\mathbf{x}_t$ drawn from a convex decision space which doesn't necessarily have to be the convex space of probability distributions over a finite number of discrete elements as before. After that, the learner receives a convex cost function c_t and suffers cost $c_t(\mathbf{x}_t)$. The difference is that in this case, the learner receives an unknown convex function instead of a cost vector on the decisions set, so the setting again generalizes the Expert problem of Online learning that we described in the previous sections.

Notice that we abused the notation and used the same round index t for the decision and the cost function of the round even though the setting is 0-lookahead. But we are going to merge again this problem with the MTS problem, so we are going to have 1-lookahead eventually. To include the MTS problem in the setting, aside from 1-lookahead, we include a switching cost metric in the cost incurred by the learner in each round. In the MTS problem, the service cost functions c_t are not assumed to be convex and the decision space is usually discrete, but we will not incorporate these assumptions into the setting.

The problem we have now described, which generalizes the problem of the previous section and combines Online Convex Optimization and MTS is considered to be in Smoothed Online Convex Optimization, or SOCO. The goal is again to minimize the total cost of the algorithm. More specifically, in each round the cost incurred by the learner is

$$c_t(\mathbf{x}_t) + \|\mathbf{x}_t - \mathbf{x}_{t-1}\|$$

where $\|\cdot\|$ is any norm. Two very common assumptions we are going to include are that the decision space is bounded and closed and that the Euclidian norm of the gradient of the service cost function is bounded in every round.

There are several more specific problems which fit this setting and have received great attention, such as k-armed bandits or the k-server problem. Aside from many applications, there are also many algorithms for this setting, each making less or more assumptions on the cost functions and the switching cost metric.

Before moving on to describe some of those, we have to mention the Online Gradient Descent algorithm, which is a well-known algorithm for the Online Convex Optimization setting and also performs well in the SOCO setting with respect to Regret.

ONLINE GRADIENT DESCENT
Parameters: $\eta_t > 0$ changing in every round
Initialization: Choose an arbitrary $\mathbf{x}_1 \in F$, where F is the convex decision space.
At each time $t = [T]$**:**

1. Let c_t be the service cost function.
2. Suffer cost $c_t(\mathbf{x}_t)$.
3. Update the decision as follows:

$$\mathbf{x}_{t+1} = P(\mathbf{x}_t - \eta_t \nabla c_t(\mathbf{x}_t))$$

where $P(y) = \arg\min_{\mathbf{x} \in F} \|x - y\|_2$ 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. The well-known result is that for $\eta_t = \Theta(1/\sqrt{t})$ the regret after T rounds with respect to the best action in hindsight is $O(\sqrt{T})$. If we make some additional assumptions and choose $\eta_t = \Theta(1/t)$ instead, the regret bound falls to $O(\log T)$.

It turns out that the same algorithm provides a good regret bound even for switching costs. As noted in [33], if $\sum_{t=1}^{T} \eta_t = O(\gamma_1(T))$ and the algorithm's regret is $O(\gamma_2(T))$ for Online Convex Optimization problems, then the same algorithm guarantees an $O(\gamma_1(T) + \gamma_2(T))$ for SOCO problems. This means the two algorithms we mentioned before yield the same regret bounds for the SOCO setting as well.

From the competitive analysis point of view, there are also algorithms which achieve a good approximation ratio for the problem. In fact, the approximation ratio for the problem can be even constant for the case that the convex decision space is a line. More specifically, in [9], the authors give three algorithms for the SOCO problem if the convex decision space $F = \mathbb{R}$. 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 for any memoryless deterministic algorithm (and close to 1.86 which is the lower bound for probabilistic non-memoryless algorithms as well).

The randomized algorithm maintains a probability distribution over $\mathbb{R}$ and it is given a cost function $c_t(x)$ in each round. Then it computes the minimizer of the cost function but of course it does not return this point because it has to take into account the distance from the previous decision point as well. To do this, it computes two points, one to the right and one to the left of the minimizer, let us denote them by x_r and x_l respectively. The constraint is that the cost function's derivative at x_r must be twice the total probability mass to the right of this point and the corresponding (opposite) equality must hold for the left point as well. Then the probability is increased by $\frac{1}{2}c''(x)$ for each point in $[x_r, x_l]$ and is zero outside this segment. This way, we take into account more points so that we do not receive a large moving cost by choosing the minimizer, but the probability distribution is more concentrated around the minimizer, since the second derivative of the convex cost function is higher closer to the extreme point. The algorithm is analysed using a potential function as usual which takes into account two terms, the expected distance between the optimal point and the algorithm's "fractional" point and the expected distance between two randomly drawn points from the algorithm's probability distribution. The deterministic algorithm picks in each round the expected point and because of the convexity of the cost function and the absolute function, this does not affect the approximation ratio. Finally, the memoryless algorithm proposed by the authors is very simple. In each round, the algorithm returns the point x_t which is $c_t(x_t)/2$ away from the previous point x_{t-1} or the minimizer if it is within this bound.

For this result, the analysis of the algorithms is based in the potential function, which is a common analysis technique for Online Learning. Next, we will introduce some results based on another very useful technique for Online Convex Optimization problems, among which lies the framework we are going to use for the Facility Location problem with Switching Costs in the last chapter.

2.3.1 The Regularization technique

The general idea of *regularization* is to alter the objective function of an optimization problem in order to enforce some property on the optimal solution. The most common approach is to add a *regularizer* to the objective function which is a smooth convex function, such as the negative entropy or the Bregman divergence. The use of the regularizer will make the overall solution more stable and it has been used for the MTS problem ([1]) as well as the more general problem of SOCO, such as in [16].

In this paper, the authors propose an algorithm for any problem that can be formulated as a covering LP (which could potentially have precedence constraints as well). In their framework, they provide an $O(\log m)$ approximation online algorithm which returns a fractional solution satisfying a set of covering and fixed precedence constraints that change over time, where m is the maximal sparsity of the covering constraints. The example problem they use is the Online Set Cover problem with Switching Costs and they also provide an $O(\log m \log n)$ rounding algorithm for this problem where n is the number of elements. This setting implies that the convex space of the decision points is a polyhedron that can be described by covering and fixed precedence constraints. This is not a very restrictive implication, since many problems' solution space can be described by such constraints. For example, the fractional shortest path problem with time-varying costs , which is a well studied problem, falls into this category. In this problem, we are given a graph with edge costs and a source and a destination node that need to be connected by the shortest path. An online algorithm for this problem needs to maintain a capacity for each edge (hence the fractional) given new edge costs in each round. At the end of the round, it pays the service cost which is the total cost of the edges as well as the movement cost which is incurred for increasing and decreasing the capacity of the edges. The feasible set of solutions in each round is defined by the covering constraints corresponding to cuts which separate the source from the destination.

Let us first define the example problem the authors are using to demonstrate the framework. Consider the classic Set Cover problem, where we need to choose the smallest number of sets whose union includes all the elements. In the SOCO setting, in each round a subset of the elements need to be covered (the decision space changes over time as well). This model allows for sets to be both added and deleted in each round. Each chosen set however pays an opening cost (switching cost), as well as a service cost for every round it is chosen. Therefore an algorithm for this problem can add and delete

sets from the solution throughout its execution and pay the corresponding switching or movement cost.

More formally, the algorithm starts with $\mathbb{R}^n$ as the feasible solution space and it is given a new polyhedron P_t in each round, defined by covering and fixed precedence constraints, along with a cost vector $\mathbf{c}_t \in \mathbb{R}^n_+$. The objective function we seek to minimize is:

$$\sum_{t=1}^{T} \mathbf{c}_t \mathbf{y}_t + \sum_{t=1}^{T} \sum_{i=1}^{n} w_i \cdot |y_{i,t} - y_{i,t-1}|$$

where $\mathbf{y}_t \in P_t$ and the weights $\mathbf{w}$ model the case when the movement cost for the sets is not uniform.

Notice that as sets and elements are interchangeable in the Set Cover problem, this is an equivalent formulation of the problem. Let $y_{i,t}$ be the indicator variable, 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. Finally, let $S_{j,t}$ be the set of sets which include element j in round t. The LP which describes the problem is the following.

$$\begin{array}{lll} \min & \sum_{t=1}^{T} \mathbf{c}_t \mathbf{y}_t + \sum_{t=1}^{T} \sum_{i=1}^{n} w_i z_{i,t} & \\ \text{s.t.} & \sum_{i \in S_{j,t}} y_i, t \geq 1 & \forall t \geq 1, 1 \leq j \leq m_t \\ & z_{i,t} \geq y_{i,t} - y_{i,t-1} & \forall t \geq 1, 1 \leq i \leq n \\ & z_{i,t}, y_{i,t} \geq 0 & \forall t, 1 \leq i \leq n \end{array}$$

The 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.

To use regularization in this case would mean to add a smooth function to the objective function $\mathbf{c}_t \mathbf{y}_t$ in each round to ensure that the distribution $\mathbf{y}_t$ with 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

$$\sum_{i=1}^{n} (p_i \ln \frac{p_i}{q_i} + p_i - q_i)$$

The online regularization algorithm which is one of the two main results of the paper is the following.

REGULARIZATION ALGORITHM
Parameters: $\epsilon > 0$, $\eta = \ln(1 + n/\epsilon)$
Initialization: $y_{i,0} = 0 \ \forall i \in [n]$
At each time $t = [T]$**:**

1. Let $\mathbf{c}_t$ be the service cost vector and let P_t be the feasible set of solutions.
2. Solve the following convex program to obtain $\mathbf{y}_t$:

$$\mathbf{y}_t = \arg\min_{\mathbf{x} \in P_t} \left\{ \mathbf{c}_t \mathbf{x} + \frac{1}{\eta} \sum_{i=1}^{n} w_i \Big((x_i + \frac{\epsilon}{n}) \ln(\frac{x_i + \epsilon/n}{y_{i,t-1} + \epsilon/n}) - x_i \Big) \right\}$$

We examine the objective function of each round. Aside from the service costs, it seeks to minimize a regularization function, with the learning parameter η serving as a tuning parameter between the two terms. For a large η, the service cost is more important that the movement cost and the opposite holds for a small η. The regularization function is almost the relative entropy of the current point and the previous point returned by the algorithm, adjusted for the weighted case. One more difference is that the points are noisy, since we have added a small amount of noise ϵ uniformly distributed on the coordinates.

The analysis of the algorithm is presented in the last chapter, where we use the same framework for the Online Facility Location problem with Switching Costs.

2.3.2 Measuring performance with respect to Regret

So far, most algorithms we have examined achieve good competitive ratios. However, no regret guarantees were provided for these algorithms. In this section, we are going to present an algorithm which guarantees low regret, by tracking the best trajectory of the solutions under some assumptions. The setting is different from the general one we have seen so far, falling into the category of Online Convex Optimization with external regret, but the two are closely related.

The authors of [36] consider the following setting. In each round a learner chooses an action $\mathbf{x}_t$ and an adversary chooses the loss function c_t which satisfies some nice properties such as bounded and Lipschitz continuous gradients and strong convexity. The learner suffers a cost $c_t(\mathbf{x}_t)$ and also learns $\nabla c_t(\mathbf{x}_t)$. The benchmark of the algorithm is the *dynamic* regret. In this case, instead of comparing the total cost of the algorithm to that of the best fixed solution in hindsight, we compare it to the total cost of the best sequence of solutions. This is similar to the benchmark we have considered so far because it is with respect to the best sequence of solutions, but there are two differences. There is no switching cost, and the regret guarantee is an additive guarantee in contrast to the competitive ratio which is a multiplicative guarantee.

More formally, the dynamic regret is

$$\mathbf{R}(\mathbf{x}_1^*, \ldots, \mathbf{x}_T^*) = \sum_{t=1}^{T} c_t(\mathbf{x}_t) - \sum_{t=1}^{T} c_t(\mathbf{x}_t^*)$$

We also define the movement cost of a solution sequence as usual. Remember however that there is no such cost incurred in each round.

$$C_T(\mathbf{u}_1, \ldots, \mathbf{u}_T) = \sum_{t=2}^{T} \|\mathbf{u}_t - \mathbf{u}_{t-1}\|$$

The Online Gradient Descent algorithm for this problem is the following:

ONLINE GRADIENT DESCENT WITH CONSTANT STEP
Parameters: h, γ
Initialization: Arbitrary vector $\mathbf{x}_1$
At each time $t = [T]$**:**

1. Play action $\mathbf{x}_t$.
2. Observe $\nabla c_t(\mathbf{x}_t)$.
3. Compute $\hat{\mathbf{x}}_t = P(\mathbf{x}_t - \frac{1}{\gamma}\nabla c_t(\mathbf{x}_t))$.
4. Update $\mathbf{x}_{t+1} = (1-h)\mathbf{x}_t + h\hat{\mathbf{x}}_t$

We notice that the algorithm has 0-lookahead as in the pure online learning setting. Also γ is the tuning parameter which defines how much the solution will move towards the minimization direction and h is the tuning parameter which defines how much it will move from the previous point. In fact, the projected point $\hat{\mathbf{x}}_t$ is the minimizer of the first order approximation of the cost function and a regularization term.

$$\hat{\mathbf{x}}_t = \arg\min_{\mathbf{x}} \{\nabla c_t(\mathbf{x}_t)^T(\mathbf{x} - \mathbf{x}_t) + \frac{\gamma}{2}\|\mathbf{x} - \mathbf{x}_t\|^2\}$$

This algorithm guarantees an $O(1 + C_T(\mathbf{x}_1^*, \ldots, \mathbf{x}_T^*))$ regret. Intuitively, the more the optimal sequence of solutions changes, the harder it is to follow.

The algorithm achieves this performance with respect to dynamic regret, because in every round it manages to return a point which is closer to the best solution of the previous round compared to the previous round's solution. Hence, if the optimal sequence of solutions does not change too much over time, the algorithm's sequence of solutions essentially converges to it. More specifically, it holds that:

$$\|\mathbf{x}_{t+1} - \mathbf{x}_t^*\| \leq \rho\|\mathbf{x}_t - \mathbf{x}_t^*\|$$

where $\rho = (1 - h\mu/\gamma)^{1/2} < 1$ is a constant depending on the tuning parameters h and γ (which is chosen to be necessarily larger than the Lipschitz continuity constant) as well as μ which is the strong convexity parameter.

It is not hard to prove that if this inequality holds, then the total distance between the algorithm's solution and the optimal is also bounded by the movement of the optimal solution.

$$\sum_{t=1}^{T} \|\mathbf{x}_t - \mathbf{x}_t^*\| \leq K_1 \sum_{t=2}^{T} \mathbf{x}_t^* - \mathbf{x}_{t-1}^* + K_2$$

for K_1 and K_2 depending on the parameters and the initial values of the algorithm.

The regret bound follows if we use the boundedness of the gradient of the cost function c_t.

2.3.3 Regret and Competitiveness Incompatibility

There is an inherent reason why we have not yet encountered algorithms which have good performance with respect to the competitive ratio and the regret at the same time. As the authors of [4] prove, there is a fundamental incompatibility between these two metrics.

Theorem 1. *There is no algorithm (randomized or deterministic) which can achieve sublinear regret and constant competitive ratio in the SOCO setting, even when the cost functions are linear.*

This theorem is not based on a pathological example, rather in the case of fairly simple and linear cost functions (only two different functions, one for the even and one for the odd rounds), and for a linear decision space. It also includes the concept of α-unfairness, which interpolates between low regret and high competitive ratio but does not achieve both simultaneously.

The good news is that it is possible to nearly achieve the goal of simultaneous good performance in the linear decision space case. The authors of the same paper provide an algorithm, named *Randomly Biased Greedy* (RBG) which achieves a competitive ratio of $(1 + \gamma)$ and $O(\max\{T/\gamma, \gamma\}$ regret. Obviously, for $\gamma = \sqrt{T}$, the performance of the algorithm with respect to the two metrics is balanced, both the competitive ratio and the regret are $O(\sqrt{T})$. For $\gamma = 1$ the competitive ratio is 2 but as expected the regret is linear. The parameter γ defines the order of the norm that the algorithm uses to estimate the distance of the new and the previous decision point, so it manages how strict the algorithm is going to be about the movement cost.

Randomly Biased Greedy
Parameters: Norm N.
Initialization: $w_0(x) = N(x)$
At each time $t = [T]$**:**

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

2.4 Coin Betting

Another line of work which gives a new perspective to Online Convex Optimization algorithms is that of Orabona et al. Aside from providing a connection between Online Optimization and Coin Betting, which allows us to use algorithms from the latter to solve problems in the former, this line of work also addresses a disadvantage of the vast majority of algorithms in the field. All the previous algorithms which provide a low regret guarantee do so for certain values of the learning parameter (or the several parameters) of the algorithm, defined in the beginning of the algorithm. These values however assume prior knowledge on the number of rounds or on characteristics of the optimal solution sequence or the optimal solution sequence itself. The design of parameter-free algorithms for this area is important for applications since such knowledge usually can not be assumed. The fundamental connection between Coin Betting and certain Online Optimization settings yields optimal results for the latter achieved by parameter-free algorithms.

Again, we are going to consider the relatively simple setting of Online Linear Optimization first. In each round the learner chooses a point $\mathbf{x}_t$ from a convex decision set (let us denote this set by K) and receives a *reward* vector $\mathbf{g}_t$, aiming to minimize the regret with respect to any fixed point $\mathbf{u}$. Replacing the cost vector with a reward vector brings us closer to Coin Betting, which we will describe next, and does not change the setting of Online Linear Optimization since the rewards can also be negative. In [38], the authors focus on two common decision spaces we have already seen, the N-dimensional probability simplex Δ_N and a Hilbert space $\mathcal{H}$, which is a generalization of the Euclidian space in N dimensions. The problem with Δ_N as the decision space is the known problem of Learning with Expert Advise (LEA) we mentioned in earlier sections. For both decision spaces, we make the assumption that the rewards are bounded, more specifically $\mathbf{g}_t \in [0,1]^N$ for LEA and $\|\mathbf{g}\| \leq 1$ for the Hilbert space.

We will introduce now the very similar setting of Coin Betting. In each round, a gambler (the learner) makes a bet on the outcomes of an adversarial coin flip. The gambler starts with an initial amount of money $\epsilon > 0$ and she is not allowed to bet more than the money she has. We denote by g_t the outcome of the coin flip in round t, where $g_t = \{-1, 1\}$ and $+1$ denotes heads while -1 denotes tails. Let us denote by x_t the betted amount of the learner in round t, the sign of which corresponds to whether she betted on heads or tails. If the learner looses then she looses $|x|$ and if she wins she gets the betted amount back as well as the same amount as a reward. Let $Wealth_t$

be the learner's total wealth in round t. It holds that:

$$Wealth_t = \epsilon + \sum_{i=1}^{t} x_i g_i$$

since in each round the wealth increases or decreases by the betted amount $|x_t|$ depending on whether the bet and the outcome have the same sign. Also, the reward of the learner at the end of round t which is the difference between her wealth and her initial amount is:

$$Reward_t = Wealth_t - \epsilon$$

It is also convenient to formulate $x_t = \beta_t Wealth_{t-1}$, where $\beta_t \in [-1, 1]$ as the learner can not bet more money than she owns. So β_t is the fraction of the current wealth to bet.

If we relax the rewards to be $g_t \in [-1, 1]$ then it is easy to see how any algorithm for the coin betting problem could be used to solve the one dimensional Online Linear Optimization problem, where $-g_t$ would be the cost of the action in round t. The total reward of the algorithm in this case would be

$$Reward_T = \sum_{t=1}^{T} x_t g_t = Wealth_T - \epsilon$$

where the $Wealth_T$ is defined the same way it is defined for Coin Betting.

The goal of an Online Linear Optimization algorithm though is to minimize regret. Assume we have an algorithm A for Coin Betting. Let $F : V \to \mathbb{R}$ be a function such that $Wealth_t \geq F(\sum_{t=1}^{T} g_t)$ is the guarantee of the algorithm on the total wealth. Let $F^* : V^* \to \mathbb{R}$ be its Fenchel conjugate. V^* is the dual vector space of V, which is the space of all the linear functionals of V and a Fenchel conjugate is defined as $sup_{x \in V} \langle \theta, x \rangle - F(x)$. Also F is assumed to be a proper lower semi-continuous function so $F^{**} = F$ and since V is the Hilbert space, it also holds that $V^* = V = \mathcal{H}$.

The first result for this simple case is the following, linking the Reward and the Regret bounds:

For any sequence of x_t and g_t, and $\epsilon \in \mathbb{R}$

$$Reward_T = \sum_{t=1}^{T} \langle g_t, x_t \rangle \geq F(\sum_{t=1}^{T} g_t) - \epsilon \Leftrightarrow Regret_T(u) = \sum_{t=1}^{T} \langle g_t, u - x_t \rangle \leq F^*(u) + \epsilon, \forall u \in V^*$$

There is an optimal strategy for Coin Betting proposed by Kelly, if we assume that the coin flips are i.i.d. with probability of heads p. In this case

the fraction of the bet should be $\beta_t = 2p - 1$, constant in all rounds. Even without knowledge of p but if we assume knowledge of the total number of rounds T we can still get optimal results, choosing $\beta_t^* = \sum_{t=1}^{T} g_t/T$ which is the expected outcome, in all rounds. This gives a wealth guarantee

$$Wealth_T(\beta^*) = \epsilon e^{(T \cdot D(\frac{1}{2} + \frac{\sum_{t=1}^{T} g_t}{2T} \| \frac{1}{2}))}$$

where $D(\cdot\|\cdot)$ is the relative entropy of the two distributions. Finally, we rid ourselves from the assumption of knowledge of T and choose

$$\beta_t = 2(\frac{\frac{1}{2} + \sum_{i=1}^{t-1} \mathbf{1}[g_i = +1]}{t}) - 1$$

which approximates the probability of heads by the history of the outcomes. This estimator of the probability p is called the Krichevsky-Trofimov estimator and the wealth guarantee of this strategy is:

$$Wealth_T \geq \frac{Wealth_T(\beta^*)}{2\sqrt{T}}$$

For this strategy, which is called *adaptive Kelly betting*, one can prove that

$$Wealth_t \geq F_t(\sum_{i=1}^{t} g_i) \tag{2.1}$$

for a certain sequence of functions that are called *potentials*. Such functions have to satisfy some properties such as monotonicity, logarithmic convexity and twice-differentiability as well as the following inequality:

$$(1 + g\beta_t)F_{t-1}(x) \geq F_t(x + g)$$

for

$$\beta_t = \frac{F_t(x+1) - F_t(x-1)}{F_t(x+1) + F_t(x-1)}$$

which is essential for preserving equation 2.1 for all rounds.

Therefore, using $\beta_t = \frac{F_t(\|\sum_{i=1}^{t} g_i\|+1) - F_t(\|\sum_{i=1}^{t} g_i\|-1)}{F_t(\|\sum_{i=1}^{t} g_i\|+1) + F_t(\|\sum_{i=1}^{t} g_i\|-1)}$ for the general problem of OLO in a Hilber space (not necessarily one dimensional) achieves a regret bound

$$Regret_T(u) \leq F_T^*(\|u\|) + \epsilon, \forall T, \forall u$$

For the case of Learning from Expert Advice, we assume a prior distribution $\pi = (\pi_1, \ldots, \pi_N)$ over the Δ_N simplex and the idea is to use an algorithm A for the one dimensional OLO for each coordinate of the probability distribution, i.e. for each expert. An algorithm for LEA would compute in each round $\hat{p_i} = \pi_i \cdot [x_{i,t}]_+$ where $[x_{i,t}]_+$ is the positive part of the decision for expert i of the algorithm A. Then it would normalize this into a probability vector and return this as the decision of the current round. Finally, given the reward vector, it would define the reward for each i as:

$$\tilde{g}_{i,t} = \begin{cases} g_{i,t} - \langle g_t, p_t \rangle & x_{i,t} > 0 \\ [g_{i,t} - \langle g_t, p_t \rangle]_+ & x_{i,t} \leq 0 \end{cases}$$

which can be interpreted as the amount of reward i earns more than the average.

Given a guarantee of sequence of potentials F_t for the algorithm A, the algorithm for LEA that we just described is proven by the authors to give a guarantee of

$$Regret_T(u) \leq f_T^{-1}(D(u\|\pi)), \forall T, \forall u \in \Delta_N$$

where $f_t(x) = \ln(F_t(x))$.

Since we have reduced the two problems to the coin betting problem (equivalent to the OLO in one dimension Hilbert space), all we need is an actual good sequence of potentials to have algorithms with low regret guarantees. It turns out that the δ-shifted KT potentials are a good such sequence of potentials, which are defined as:

$$F_t(x) = \frac{2^t \Gamma(\delta+1)\Gamma(\frac{t+\delta+1}{2} + \frac{x}{2})\Gamma(\frac{t+\delta+1}{2} - \frac{x}{2})}{\Gamma(\frac{\delta+1}{2})^2 \Gamma(t+\delta+1)}$$

where $\Gamma()$ is the Euler gamma function.

These achieve $O(\|u\|\sqrt{T \ln(1 + \|u\|T)})$ regret for OLO in the Hilbert space and $O(\sqrt{T(1 + D(u\|\pi))})$ regret for LEA.

However, the setting we are interested in is more complicated than that. The environment is changing so static regret is not a good measure of performance since it assumes that a single decision can be a good solution overall. In [30] the authors extend their setting, by using *Strongly Adaptive Regret* and m-shift regret (corresponding to the case of the shifting experts we have mentioned). These types of regret compare the cost of the algorithm to a more challenging benchmark. More specifically, Strongly Adaptive Re-

gret is obtained by evaluating the static regret in each continuous interval $I = [I_1 \dots I_2] \subseteq [T]$ with length τ and it is given by the equation:

$$SA - Regret_T(\tau) = \max_{I \subseteq [T]:|I|=\tau} \Big(\sum_{t \in I} c_t(\mathbf{x}_t) - \min_{\mathbf{w}} \sum_{t \in I} c_t(\mathbf{w})) \Big)$$

The $m - Shift - Regret_T$ is defined as the difference between the total cost of the algorithm and the total cost of the optimal m-shifting sequence of decisions, which is restricted to change at most m times over the T rounds.

It is common in the design of a meta-algorithm for changing environments to assume a black-box $\mathcal{B}$ of a simple online learning algorithm with Lipsitchz convex cost functions, and run a new instance of that algorithm in every time-step. Applying this technique means that we instantiate a new run of a black-box $\mathcal{B}_J$ for each interval J. The decision of the algorithm at round t would then be a weighted average of the decisions of all the alive black-box algorithms at time t. The reason why this is a popular technique for changing environments is that the instances which begin at time t do not have any dependence on the previous rounds so if the environment changes in this round, these black-boxes will follow the change more quickly. However good this technique, it requires a new run in each round, which induces high computational complexity to the meta-algorithm. For this reason, we consider a more carefully designed set of intervals, the *geometric covering* intervals, which are intervals of doubling length, with intervals of size 2^k partitioning exactly the set $N \setminus \{1, \dots 2^k - 1\}$. Any set can be partitioned into geometric covering intervals and this way the number of alive black-boxes at each round t is reduced to $O(\log_2(t))$.

Strongly Adaptive regret depends on the meta-algorithm $\mathcal{M}$ and the black-box algorithm $\mathcal{B}$. There is an additive decomposition of the regret which separates these two sources of regret. The $SA - Regret$ for interval I which can be decomposed into $\bigcup_{i=-a}^{b} J^{(i)}$ is:

$$SA{-}Regret_I(\mathbf{w}) = \sum_{i=-a}^{b} \sum_{t \in J^{(i)}} (c_t(\mathbf{x}_t^{\mathcal{M}(\mathcal{B})}) - c_t(\mathbf{x}_t^{\mathcal{B}_{J^{(i)}}})) + \sum_{i=-a}^{b} \sum_{t \in J^{(i)}} (c_t(\mathbf{x}_t^{\mathcal{B}_{J^{(i)}}}) - c_t(\mathbf{w}))$$

The inner summations of the two terms are the meta regret and the black-box regret on $J^{(i)}$ respectively. It is proven that if the regret in each interval scales like $\tilde{O}(\sqrt{|J^{(i)}|})$ then the outer summation is $\tilde{O}(\sqrt{|I|})$. Since we know black-box algorithms which achieve this bound for regret, it remains to also find a meta algorithm with such a guarantee. The Coin Betting algorithm

for LEA which we described with δ-KT potentials is such an algorithm where each black-box is treated as an expert. This is not completely true because we have to account for the fact that not all experts/black-boxes are running at all rounds, but the algorithm can be adjusted to this case, borrowing some ideas from the Sleeping Bandits setting.

This algorithm manages to have $\tilde{O}(\sqrt{(I_2 - I_1)\log(I_2)})$ SA-Regret and $\tilde{O}(\sqrt{mT(\log N + \log T)})$ m-shifting regret, using Coin Betting as the black-box algorithm.

Chapter 3

The Facility Location Problem

In this chapter we will introduce the Facility Location problem. First, we are going to define the problem in its simplest offline form and then we are going to discuss several of the many variations that this problem has and some main results. The online version of the problem we also be discussed and the similar problem of Facility Leasing. Finally, we will state the dynamic version of the problem, that is the offline version with switching costs, and state the results and the techniques that have been used to solve it.

3.1 Definitions and Variants

The Facility Location problem, is a well know problem that has been studied for decades and in various forms. The main setting is that we are given a set of demand points, i.e. clients, C and a set of facilities F. Each facility $i \in F$ has an opening cost f_i and the cost of connecting client j with facility i is denoted as d_{ij} and is usually thought to describe the distance between the client and the facility. An algorithm for this problem should choose which facilities to open and connect each client to a facility. The goal is to minimize the total cost of the solution, which is the sum of the opening costs and the connection costs. One common way to think of this problem is that the facilities are possible positions for hospital buildings, and we need to decide where we are going to build hospitals so that the total opening cost of the hospitals and the total distance between the citizens (clients) and the hospitals is the lowest.

There is a large number of variants of the Facility Location problem. One reason for this is that the problem is hard but can accept good approximation algorithms or even optimal algorithms depending on the assumptions. Another reason is that it is related to many real world problems and different models and assumptions may be necessary in each case.

The previous description of the problem implies that every client needs to be connected to at least one facility in a feasible solution. However, there is a variant of the problem where each client has a different demand r_j, meaning that they need to be connected to at least r_j facilities. These kind of problems are called fault tolerant, because if a facility is suddenly down, the client can use another facility. It is better understood in the area of networks, where a facility could be a server or a router. Because fault tolerant Facility Location is a popular variant, the original problem is also called *unit* demand. Another variant of the problem, is to allow for clients to not be connected to a facility and pay a penalty instead. Even in this specific area there are different assumptions to what these penalties could be, where linear and submodular functions are the two main choices.

Another very common variant is the Capacitated Facility Location problem. This variant limits the number of clients that each facility i can be connected to by u_i. It would not make sense for example for a hospital to be able to treat an unlimited number of patients at the same time. A variant which is generalized by the previous one is the case where all capacities are the same. Again, the original problem has a new name because of the po-

pularity of this variant, and it is called the Uncapacitated Facility Location problem.

Last but not least, the most common assumption is that the points are in a metric space. This means that the distances between the clients and the facilities all satisfy the triangle inequality. This is a very important assumption, since the non-metric Facility Location problem can not be solved in polynomial time. This follows from an easy reduction of the NP-hard Set Cover problem to the Facility Location problem, where the sets represent the facilities, the elements represent the clients and we can consider all opening costs to be 1 and all connection costs to be 1 or ∞ depending on whether the element is in a set or not. Despite its hardness, the problem does admit an approximation algorithm, similar to the greedy Set Cover algorithm, which achieves $\log n$ approximation ratio.

3.1.1 Static Facility Location

The static Facility Location problem is just the original offline problem, where "static" is used to distinguish this problem from the dynamic one which we will focus on towards the last part of the thesis. We consider only the metric case. Even for the metric case, [25] proved as soon as 1999 that the problem does not have an approximation algorithm with ratio better than 1.463, given that some common complexity assumptions hold. The facility location problem has two types of cost. The facility opening cost and the connection cost. An even more detailed bound presented in [28] states that there is no (γ_f, γ_c) approximation factor for the opening and connection costs respectively with $\gamma_c < 1 + 2e^{-\gamma_f}$. The gap between the approximation factor and the lower bound is not yet closed ([17]).

The first constant factor approximation for the problem was 3.16 ([39]) and later the ratio was improved to $1 + 2/\epsilon$ ([19]). The first result was based in [32] which provides a technique for rounding LP solutions, and the second result used LPs and randomized rounding as well. The 3-approximation algorithm in [29] uses a primal-dual scheme. We are going to present the LP formulation and relaxation of the problem, as it is apparently used in many algorithms and it provides the reader with a good intuition behind the variables of the problem. Aside from that, the algorithms for the dynamic version of facility location will also use the corresponding LP formulation of that problem.

Let y_i and x_{ij} be the variables that denote if facility i is open and if client j is connected to facility i respectively.

$$\begin{array}{lll} \min & \sum_{i \in F, j \in C} d_{ij} x_{ij} + \sum_{i \in F} f_i y_i & \\ \text{s.t.} & \sum_{i \in F} x_{ij} \geq 1 & \forall j \in C \\ & y_i - x_{ij} \geq 0 & \forall i \in F, \forall j \in C \\ & x_{ij} \in \{0, 1\} & \forall i \in F, \forall j \in C \\ & y_i \in \{0, 1\} & \forall i \in F \end{array}$$

The first constraint demands that every client should be connected to at least one facility and the second constraint demands that if a client is connected to a facility then that facility should be open. The LP relaxation of the problem is the following:

$$
\begin{array}{lll}
\min & \sum_{i\in F, j\in C} d_{ij}x_{ij} + \sum_{i\in F} f_i y_i & \\
\text{s.t.} & \sum_{i\in F} x_{ij} \geq 1 & \forall j \in C \\
& y_i - x_{ij} \geq 0 & \forall i \in F, \forall j \in C \\
& x_{ij} \geq 0 & \forall i \in F, \forall j \in C \\
& y_i \geq 0 & \forall i \in F
\end{array}
$$

In the LP relaxation, the variables can take any non-negative real value and this allows for the constraints to be satisfied in a fractional way. This means that a client can be connected by 0.3 to one facility and by 0.7 to another facility, this way being covered, while the respective facilities should be at least 0.3 and 0.7 open.

The dual LP program is the following:

$$
\begin{array}{lll}
\max & \sum_{j\in C} a_j & \\
\text{s.t.} & \sum_{j\in C} e_{ij} \leq f_i & \forall i \in F \\
& a_j - e_{ij} \leq d_{ij} & \forall i \in F, \forall j \in C \\
& a_j \geq 0 & \forall j \in C \\
& e_{ij} \geq 0 & \forall i \in F, \forall j \in C
\end{array}
$$

The dual program provides an intuition to the role of each variable in the LP. Let us assume the integral case for simplicity. The dual variable a_j denotes how much a client must pay in total to be served and e_{ij} is the amount of that which goes towards the opening cost of the facility they are connected to. More specifically, if we look at the complementary slackness conditions, we can see that for every $y_i = 1$, that is every open facility i, the opening cost must be covered, i.e. $\sum_{j\in C} e_{ij} = f_i$. In addition, for every $x_{ij} = 1$, that is if client i is connected to facility j, then $a_j - e_{ij} = d_{ij}$ must hold. This equality denotes that client j must pay e_{ij} just for the opening of the facility and d_{ij} for the connection, while this is the total amount a_j they must pay since they are not connected to any other facility and they do not contribute any amount $e_{i'j}$ for another facility i'.

The algorithm by Jain and Vazirani, increases the dual variables (making sure no constraint is violated at each time step) until all clients are connected and then chooses a subset of these facilities to open and connects the remaining clients to the closest facility "indirectly", achieving an approximation factor of 3, which is only due to the opening cost. The analysis is tight and there is an example in which the algorithm can construct a dual solution whose value is $3 - \epsilon$ away from the optimal ([18]).

There have been many other approaches for this problem. In [6] the authors prove a local search heuristic with $5 + \epsilon$ approximation ratio. In [27] the authors use the dual fitting method. This method is based on the following idea: in the greedy algorithm (similar to the algorithm for set cover) the dual program is infeasible. However, if the dual program is shrunk (devided by a certain factor γ) then it becomes feasible. Since the dual solution serves as a lower bound to the dual solution, this means that we have a γ approximation factor and the problem is then to find the minimum γ. Finally, the best known approximation ratio so far in 1.488 ([31]).

3.1.2 Online Facility Location

In the online version of the Facility Location problem, which was introduced in its most known form in [34], the clients arrive one at a time and we must maintain a set of open facilities at each time so that every client is connected. The goal is to minimize the total opening and connection cost over time. Also, a common assumption we are going to make from now on is that the opening costs are uniform, that is every facility has the same opening cost f. This variant of the problem is very natural and this can be demonstrated with the network example we mentioned for the offline case. In this example, we are building a network and we need to connect clients to servers with the smallest total cost, which includes the connection cost as well as an "opening" cost whenever we buy a new server. New clients may need to join the network after the original structure has been built and this is modeled by this online variant of the problem. The benchmark is the minimum total cost achieved by the optimal offline solution.

Any algorithm for this problem essentially needs to maintain a balance between the opening and the connection cost. In this first paper which introduced the problem, Meyerson gives a constant approximation algorithm for the case the clients arrive in a random order. The intuition behind his algorithm is that there is a *potential* of $d(F_{i-1}, u_i)$ to each demand point u_i, which is an upper bound to its connection cost and how much it can contribute to the opening of facilities close to it ([23]). This algorithm opens a new facility on u_i's location with probability $d(F_{i-1}, u_i)/f$ and connects it to the nearest facility otherwise. Unfortunately, if the clients arrive in an adversarial order, then no algorithm can give a constant approximation ratio. There is however an $O(\log n)$ competitive algorithm, where n denotes the number of demands. Both these algorithms are randomized.

In this setting, any facility opens irrevocably. This may capture in some sense the network example we gave but it is quite restrictive for other real life problems. This is why another variant of the problem, namely the Incremental Facility Location problem was defined. In the Incremental Facility Location problem, the clients arrive one by one and must be assigned a facility upon arrival. But the algorithm can merge two facilities by closing one of them and re-assigning its clients to the second one. The goal is the same.

There is a lower bound of $\Omega(\frac{\log n}{\log \log n})$ on the competitive ratio of any algorithm for the Online Facility Location problem ([22]). This lower bound does not depend on a rare instance of the problem and it holds even for sim-

ple metric spaces such as the real line. It holds for randomized algorithms against an oblivious adversary. To prove the lower bound, the construction uses a metric space which is described by a Hierarchically Well-Separated Tree and proves the bound for deterministic algorithms, using Yao's principle afterwards to extend the result to randomized algorithms. However, in the same paper Fotakis presents a deterministic algorithm with an asymptotically matching competitive ratio of $O(\frac{\log n}{\log \log n})$. The analysis is based on the fact that any metric space has a hierarchical cover with the property that any component is either large or is relatively well-separated and on a potential function argument which distinguishes between the two kinds of components. Then the well-separated components can be treated as independent instances with a single facility as their optimal solutions, and for the large diameter components the algorithm incurs a bounded additional cost.

As for the incremental version, the problem admits a constant competitive ratio. The algorithm of [21] maintains its facility configuration, its merge configuration consisting of a merge ball for each facility, and the set of unsatisfied demands. Every demand is unsatisfied upon arrival and holds a potential equal to its distance to the closest facility. If the unsatisfied neighbourhood of a new demand has accumulated enough potential (a fraction of the opening cost of a facility), then we decide it is worth it to open a facility and we open it in the location of the new demand. Then all the demands in the neighbourhood are satisfied. The merge balls initially have a small radius and include only the facility to which they belong and any new facility in the ball of an old one is merged with the old one. To ensure that there will not be too much merging to increase the connection costs dramatically, the ball decreases over the course of the algorithm.

A more in depth presentation of the various results on the Online and Incremental Facility Location problem, as well as some more related models (streaming) and problems (k-clustering) can be found in [23].

3.1.3 Facility Leasing

In [5], the authors introduced another variant of the Facility Location problem. They introduced the notion of time. The problem can be described as follows: We are given a set of facilities F and a set of clients C. There are distinct time periods from time 1 to T and at each time period t, a subset C_t of clients must be served by a facility that is open at that time. There are k different lease types available $\ell_1, \ldots, \ell_k$ and each facility can be leased at any period t for ℓ_j days at a cost f_i^j. The lease cost can be dependent on both the facility and the lease period. The goal is to minimize the total connection and leasing costs while ensuring that for each time period t, clients in C_t can be served by at least one open facility in that period. This variant of the problem can model other real life problems, where the facilities and the demands have a limited lifespan.

By formulating the problem in its LP relaxation, the authors of [37] managed to treat the problem as a generalized Facility Location problem and achieved a constant approximation ratio of 3 for the offline version of the problem using the classic algorithm by Jain and Vazirani. More specifically, they extend the definitions of the facilities and the clients to capture a specific time. The facilities are defined as triples including the facility, the lease type, and the time t and the clients are defined as tuples including the client and the time t.

For the online version of Facility Leasing, Nagarajan and Williamson presented in the same paper a primal-dual $O(k \log n)$-competitive algorithm. For the online version it is important to clarify that the potential facility locations and their lease types are given to the algorithm in advance and only the client sets C_t arrive online. Online Facility Leasing is a generalization of Online Facility Location and the Online Parking Permit. The latter is defined in [35] and it can be seen as a variant of a well known problem in online algorithms, the ski rental problem. The example for this problem is the following: A commuter has the choice to go to work either on foot or by driving and we don't know in advance what she will choose. On any driving day, she can apply for a parking permit, picking from a set of parking durations and permits following the subadditive property. There are k different types of permits available and each one has a duration and a cost. The driving days are revealed one at a time and the goal is to minimize the cost paid by the algorithm. The benchmark is the optimal cost of an offline algorithm. In the same paper, Meyerson provides an $O(k)$-competitive deterministic algorithm for the problem.

The best known lower bounds for the Online Facility Leasing problem are $\Omega(k + \frac{\log n}{\log \log n})$ and $\Omega(\log k + \frac{\log n}{\log \log n})$ which follow the lower bounds on the Online Facility Location and the Parking Permit problems respectively.

3.2 Dynamic Facility Location

The problem we are interested in, which is closer to the area of research we described in the previous chapter is the Online Facility Location problem with Switching Costs. If the algorithm is not required to be online, the problem changes into what is called in the literature the Dynamic Facility Location problem.

As introduced in [20], the Dynamic Facility Location problem can be described as follows: Let $t \in \{1, \dots, T\}$ be the time variable. At each time t, we are given the set of connection costs $\{d_{ij}^t | j \in [n], i \in [m]\}$, where each d_{ij}^t denotes the connection cost between client j and facility i at time t. We also know in advance the uniform opening cost f of the facilities which does not change over time. At each time t, there are different connection costs, and in the end we should return a feasible solution. A solution is described by the vectors $y^t \in \{0,1\}^n$ and $x^t \in \{0,1\}^{m \times n}$, where y_i^t is an indicator variable for the opening of facility i at time t and x_{ij}^t is an indicator variable for the connection of client j to facility i at time t. In order for the solution to be feasible, each client should be served, i.e. they should be connected to one facility. Therefore, it should hold that

$$\sum_{i=1}^{m} x_{ij}^t = 1 \ \forall j \in [n]$$

In addition, no client should be connected to a closed facility, therefore

$$x_{ij}^t \leq y_i^t \ \forall j \in [n] \forall i \in [m]$$

The goal is to minimize over time the total opening cost, connection cost and switching cost. The latter is what differentiates this problem from the Facility Location problem. The switching cost at time t is given by $g \sum_{j=1}^{m} \sum_{i=1}^{n} z_{ij}^t$ where $z_{ij} = \mathbb{1}\{x_{ij}^t \neq x_{ij}^{t-1}\}$. From the definition, an amount g of switching cost is incurred whenever the connection status between a client and a facility changes. More formally, the benchmark for our algorithm is:

$$\min f \sum_{t=1}^{T} \sum_{i=1}^{m} y_i^t + \sum_{t=1}^{T} \sum_{j=1}^{n} \sum_{i=1}^{m} x_{ij}^t d_{ij}^t + g \sum_{t=1}^{T} \sum_{j=1}^{n} \sum_{i=1}^{m} z_{ij}^t$$

The linear relaxation of our problem is the following:

$$
\begin{array}{lll}
\min & f\sum_{t=1}^{T}\sum_{i=1}^{m} y_i^t + \sum_{t=1}^{T}\sum_{j=1}^{n}\sum_{i=1}^{m} x_{ij}^t d_{ij}^t + g\sum_{t=1}^{T}\sum_{j=1}^{n}\sum_{i=1}^{m} z_{ij}^t & \\
\text{s.t.} & x_{ij}^t \leq y_i^t & \forall t \in [T], \forall i \in [m], \forall j \in [n] \\
& \sum_{i=1}^{m} x_{ij}^t \geq 1 & \forall t \in [T], \forall j \in [n] \\
& z_{ij}^t \geq x_{ij}^t - x_{ij}^{t-1} & \forall t \in [T], \forall i \in [m], \forall j \in [n] \\
& y_i^t \geq 0 & \forall t \in [T], \forall j \in [n] \\
& x_{ij}^t \geq 0 & \forall t \in [T], \forall i \in [m], \forall j \in [n] \\
& z_{ij}^t \geq 0 & \forall t \in [T], \forall i \in [m], \forall j \in [n]
\end{array}
$$

Since y_i^t depends also on the time t, the problem includes the case that facilities can be opened and closed throughout the course of the algorithm. One way to think about it is that the facilities have an *hourly* cost and we pay for them only the times we use them. However, the change in the facility's state does not incur any cost.

The algorithms for this problem assume the optimal fractional solution given by the LP and use a rounding technique on this solution.

3.2.1 Rounding Techniques

Logarithmic approximxation

In [20], where the problem was introduced, the authors give a randomized rounding algorithm. Let us assume for now that $y_i^t = y_i$, that it the facilities we choose to open are open at all times. Let (x, y, z) denote the optimal fractional solution obtained by the LP relaxation. The rounding algorithm is the following:

RANDOMIZED ROUNDING FOR DYNAMIC FL

1. Draw a facility at random $\Gamma = 2\log(2nT)\sum_{i\in F} y_i$ times independently with a distribution proportional to y. These are the open facilities of the algorithm. Denote this set by A.
2. For each client j determine when they should change facility using the z-variables.
   - Partition time greedily into ℓ_j intervals $[t_k^j, t_{k+1}^j)$ such that $t_1^j = 1, t_{\ell_j+1}^j = T+1$ and every t_{k+1}^j is inductively defined as the largest $t \in (t_k^j, T+1]$ such that
   $$\sum_{i\in F}(\min_{t_k^j \le u < t} x_{ij}^u) \ge 1/2$$
   - For each time interval $[t_k^j, t_{k+1}^j)$ connect j to the facility in A which is cheapest for j at that time interval.

The expected opening cost of the algorithm is obviously at most $\Gamma f = 2f\log(2nT)\sum_{i\in F} y_i$ since it is higher when we choose different facilities in each draw and there are $\Gamma = 2\log(2nT)\sum_{i\in F} y_i$ draws.

The intuition for this algorithm is that it tries to maintain a stable solution, but when the z variable is high enough it switches. The way the intervals are chosen, the solution is guaranteed to not pay more than twice as much as the optimal solution for the switching cost. The inequality that defines the intervals seems complicated but it essentially does the following: In each round, we update the minimum connection cost we have seen so far in this interval for each facility that j could be connected to. Obviously the sum of these numbers does not increase and could only decrease, because we

maintain the minimum for each facility. Intuitively, if this number stayed the same, it would mean that the environment does not change so the interval should be long and there is no need to switch. Let us explain how the inequality is connected to the z-variables.

It holds that:

$$\sum_{t_k^j \le t < t_{k+1}^j} \sum_{i \in F} z_{ij}^t > 1/2, \forall j, \forall [t_k^j, t_{k+1}^j), k < \ell_j$$

Proof. For all t, and in particular for $t = t_k^j$ it holds that

$$\sum_{i \in F} x_{ij}^{t_k^j} = 1 \tag{3.1}$$

since x is the optimal and feasible solution of the LP. Since $k < \ell_j$, meaning we have not reached the end of the interval, it also holds that $\sum_{i \in F} (\min_{t_k^j \le t \le t_{k+1}^j} x_{ij}^t) < 1/2$

Let t_i be the time in the whole interval when the minimum connection cost for our client and facility i occurs, that is $x_{ij}^{t_i} = \min_{t_k^j \le t \le t_{k+1}^j} x_{ij}^t$. The previous inequality is then formulated as

$$\sum_{i \in F} x_{ij}^{t_i} < 1/2 \tag{3.2}$$

The z-variables are non-negative, so $z_{ij}^t \ge 0$. It follows that $\sum_{t_k^j \le t < t_{k+1}^j} z_{ij}^t \ge \sum_{t_k^j \le t < t_i} z_{ij}^t \ge \sum_{t_k^j \le t < t_i} (x_{ij}^t - x_{ij}^{t+1}) = x_{ij}^{t_k^j} - x_{ij}^{t_i}$ where the last inequality holds because of the LP constraint on z_{ij}^t.

The sum of this quantity for all facilities is:

$$\sum_{i \in F} \sum_{t_k^j \le t < t_{k+1}^j} z_{ij}^t \ge \sum_{i \in F} (x_{ij}^{t_k^j} - x_{ij}^{t_i}) \overset{3.1,3.2}{>} 1 - 1/2 = 1/2$$

□

This fact yields an easy bound for the switching cost. The total switching cost of client j is g times the number of intervals minus 1. But for every interval, except for the last one, the optimal solution's z_{ij}^t's add up to at least $1/2$ over the interval so the optimal solution pays at least $g/2$ for any interval. Therefore, the total switching cost of the rounded solution is at most twice the switching cost of the optimal LP fractional solution, so at most twice the cost of the optimal integral one.

It remains to bound the connection cost for each client j for each interval $I = [t_k^j, t_{k+1}^j)$. Let x_{ij}^I be the minimum connection cost for client j and facility i within this interval, that is $x_{ij}^I = \min_{t \in I} x_{ij}^t$. The distribution which is implied by these x_{ij}^I's is $\hat{x}_{ij}^I = \frac{x_{ij}^I}{\sum_{i' \in A} x_{i'j}^I}$. It seems a little bold to decide on the open facilities the way the rounding algorithm does, because it is early on the algorithm, it is random (following the distribution implied by the optimal solution of course) and it seems to not take into account the connection costs. However, the authors prove that in fact the facility selection process can be simulated by selecting a facility according to $\hat{x}_{ij}^I$, losing only a factor of 2 in the approximation ratio.

More specifically, consider the probability $p_j^I = \frac{\sum_{i \in F} x_{ij}^I}{\sum_{i \in F} y_i}$. Since $x_{ij}^I \leq x_{ij}^t \leq y_i$, where the first inequality holds because of the definition of x_{ij}^I and the second holds because of the LP's constraint, this is indeed a probability. If with probability p_j^I we sample proportionally to x_{ij}^I and with the remaining probability we sample proportionally to $(y_i - x_{ij}^I)$, then we sample in fact proportionally to y_i:

$$\begin{aligned} \Pr\{i \text{ is selected}\} &= \frac{\sum_{i \in F} x_{ij}^I}{\sum_{i \in F} y_i} \cdot \frac{x_{ij}^I}{\sum_{i \in F} x_{ij}^I} + \left(1 - \frac{\sum_{i \in F} x_{ij}^I}{\sum_{i \in F} y_i}\right) \cdot \frac{y_i - x_{ij}^I}{\sum_{i \in F} (y_i - x_{ij}^I)} \\ &= \frac{x_{ij}^I}{\sum_{i \in F} y_i} + \frac{\sum_{i \in F} y_i - \sum_{i \in F} x_{ij}^I}{\sum_{i \in F} y_i} \cdot \frac{y_i - x_{ij}^I}{\sum_{i \in F} y_i - \sum_{i \in F} x_{ij}^I} \\ &= \frac{y_i}{\sum_{i \in F} y_i} \end{aligned} \tag{3.3}$$

More formally, we consider the following selection process, as a thought experiment. If U is a uniform real number in the interval $[0, \sum_{i \in F} y_i)$, we say that facility i is selected if U happens to fall into the sub-interval $[\sum_{k<i} y_k, \sum_{k \leq i} y_k)$. Also, event B_j^I occurs when $U \in [\sum_{k<i} y_k, \sum_{k<i} y_k + x_{ij}^I)$, which is an even smaller interval. To sum up, according to this selection process: i is distributed according to y_i. Also,

$$\Pr[B_j^I] = p_j^I = \frac{\sum_{i \in F} x_{ij}^I}{\sum_{i \in F} y_i} \geq \frac{1}{2 \sum_{i \in F} y_i} \tag{3.4}$$

where the last inequality holds because of the algorithm's criterion. Finally, conditioned to B_j^I, i is selected proportionally to x_{ij}^I.

The selection process is repeated $\Gamma = 2 \log(2nT) \sum_{i \in F} y_i$ times independently. For a pair (j, I), the probability that event B_j^I never occurs is at most $(1 - p_j^I)^\Gamma \overset{(1-x)^a \leq e^{-ax}, 3.4}{\leq} e^{-\frac{\Gamma}{2 \sum_{i \in F} y_i}} = \frac{1}{2nT}$. Since there are at most nT pairs of

(j, I), the union bound ensures that the events B_j^I occur at least once with probability at least $1/2$.

When B_j^I occurs, the selection is selected proportionally to x_{ij}^I, so for each time $t \in I$ the expected connection cost is:

$$\sum_{i \in F} \frac{x_{ij}^I}{\sum_{i \in F} x_{ij}^I} d_{ij}^t \leq \frac{1}{1/2} \sum_{i \in F} x_{ij}^t d_{ij}^t$$

where the last inequality holds because $x_{ij}^t \geq x_{ij}^I$ by the definition of x_{ij}^I and the algorithm's criterion. This means that with probability $1/2$, the expected connection cost is at most twice the optimal connection cost of the fractional solution. Therefore, in total it holds for the expected cost that:

$$\mathbb{E}[C] \leq 2 \log(2nT) \cdot OPT_{LP}$$

with probability at least $1/2$. Using Markov's inequality, we conclude that

$$\begin{aligned} &\Pr\{C \leq 4 \log(2nT) \cdot OPT_{LP}\} \\ &= \Pr\{C \leq 2(2 \log(2nT)) \cdot OPT_{LP} | \text{ all } B_j^I \text{ occur at least once } \} \cdot \Pr\{ \text{ all } B_j^I \text{ occur at least once}\} \\ &= \tfrac{1}{4} \end{aligned} \tag{3.5}$$

If we repeat this algorithm $O(\log \frac{1}{\epsilon})$ times, then the best solution increases the success probability to $1 - \epsilon$.

Let us go back to the case where y_i^t is not fixed. The following is a rounding algorithm for this case, which is similar to the previous algorithm we analysed.

Randomized Rounding for DFL with hourly costs

1. For each facility i, pick a random threshold ρ_i, such that $\Pr\{\rho_i > a\} = e^{-2a\log(2nT)}$ for all $a \geq 0$. Open facility i at all times t such that $y_i^t > \rho_i$. Let A_t be the set of open facilities at time t.

2. For each client j determine when they should change facility using the z-variables.

   - Partition time greedily into ℓ_j intervals $[t_k^j, t_{k+1}^j)$ such that $t_1^j = 1, t_{\ell_j+1}^j = T+1$ and every t_{k+1}^j is inductively defined as the largest $t \in (t_k^j, T+1]$ such that

   $$\sum_{i \in F} (\min_{t_k^j \leq u < t} x_{ij}^u) \geq 1/2$$

   - For each time interval $I = [t_k^j, t_{k+1}^j)$ and facility i, connect j to the facility in A_t that minimizes the ratio ρ_i / x_{ij}^I, where $x_{ij}^I = \min_{t \in I} x_{ij}^t$.

Notice that an alternative way to define ρ_i is that it is a sample from an exponential distribution with rate $1/(2\log(2nT))$. Exponentially distributed random variables have many nice properties, which explain their use in designing rounding algorithms. *Exponential Clocks* is an example of that. For an introduction to Exponential Clocks, jump to 4.2.1. It is easier to understand the analysis of this algorithm, having exponential clocks in mind.

The probability that a facility i is opened is $\Pr\{\rho_i \leq y_i^t\} = 1 - e^{-2y_i^t \log(2nT)} \leq y_i^t \log(2nT)$. Therefore the expected opening cost is again at most $f \sum_{i \in F} \sum_{t=1}^{T} y_i^t \cdot 2\log(2nT)$. The switching cost also does not change since we use the same criterion.

For the connection cost we will try to follow the same reasoning as before. We want to imagine again that a facility is sampled according to x_{ij}^I, but we also need this facility to be alive at time t. We will consider the case that we only open facility i if $x_{ij}^I > \rho_i$. In the last step, we assign facility i to client j if ρ_i / x_{ij}^I is minimum. So we get a facility if it also holds that $\rho_i / x_{ij}^I < 1$.

The probability that client j is not covered by an open facility this way is

$$\Pr\{\min_{i \in F}(\rho_i / x_{ij}^I) \geq 1\} \overset{\rho_i \text{ i.i.d.}}{=} \Pi_{i \in F} \Pr\{\rho_i \geq x_{ij}^I\} = e^{-2(\sum_{i \in F} x_{ij}^I)\log(2nT)}$$

But this probability is less than $\frac{1}{2nT}$ since $\sum_{i \in F} x_{ij}^I \leq 1/2$. Then the union bound of the probabilities of the opposite event, that is that all clients are covered in all intervals is at least $1/2$. Conditioned to this event, the expected connection cost between client j and every facility at time t is:

$$\sum_{i \in F} \frac{\Pr\{\rho_i \leq x_{ij}^I\}}{1/2} d_{ij}^t \leq \sum_{i \in F} 2(1 - e^{-2x_{ij}^I \log(2nT)}) d_{ij}^t \leq \sum_{i \in F} 4x_{ij}^I \log(2nT) d_{ij}^t$$

So in the same way as before, with probability $1/4$ the cost of the algorithm is at most $4\log(2nT)$ times the cost of the optimal solution.

Constant Approximation

Borrowing some parts of the previous algorithm, the authors of [3] give a constant approximation rounding algorithm for Dynamic Facility Location. And in fact that constant is 14 which is relatively small. The algorithm assumes the optimal fractional solution of the LP program, and applies a preprocessing on this solution before doing the rounding.

The preprocessing consists of two parts. The first and most important ensures that the solution will not switch too often, based on the same idea from [20]. More specifically, if in each of the intervals that the previous algorithm defines, we set $x_{ij}^t = \frac{\min_{u \in I} x_{ij}^u}{\sum_{i' \in F} \min_{u \in I} x_{i'ij}^u}$ to be the value of the variable for all $t \in I$, then this is at most twice the optimal value. If we double the y_i^t variable as well, the solution remains feasible. Finally, for the z variables, we set $z_{ij}^t = x_{ij}^t - x_{ij}^{t+1}$ and we increase them until the total switching cost is twice the optimal. After this preprocessing, the new solution (x, y, z) has lost a 2 approximation factor but satisfies the following property

$$\sum_{t=1}^{T-1} |Z^t| \leq \sum_{t=1}^{T-1} \sum_{i \in F} \sum_{j \in C} z_{ij}^t$$

where $|Z^t|$ is the number of clients who changed their connection variables in that step. The second part of the preprocessing uses the standard technique of duplicating facilities with care and ensures that all $x_{ij}^t \in \{0, y_i^t\}$ and $y_i^t \in \{0, o_i\}$ for some $o_i \in [0, 1]$.

The algorithm is the following:

CONSTANT APPROX. RANDOMIZED ROUNDING FOR DFL
Initiate: Sample independently an exponential clock Q_i with expectation o_i for each facility i and R_j with expectation 1 for each client j.
At each time $t \in [T]$:

1. Consider the clients in a non-decreasing order of their exponential clocks.
2. When client j is considered:
   - Find the facility with the smallest clock among the facilities such that $x_{ij}^t > 0$, i.e. $i = \arg\min_{i:x_{ij}^t>0} Q_i$.
   - Find the client with the smallest clock in the neighbourhood of i, i.e. $j' = \arg\min_{j':x_{ij'}^t>0} R_{j'}$.
   - If j is the smallest clock in the previous step, that is $j = j'$, then connect j to i, otherwise connect j to j''s facility.

The procedure is well defined, since if j' is different than j and it is the smallest clock, then it is smaller than j so it would have been examined before j meaning that it is already connected to a facility.

There is an alternative description of the previous algorithm which uses graphs and is more intuitive for the proofs which will follow. Let $\mathcal{SG}(x^t)$ be the support graph of x^t, that is a bipartite undirected graph with vertices $F \cup C$ where an edge $\{i, j\}$ exists if $x_{ij}^t > 0$. Another graph $\mathcal{CG}(x^t)$ is called the connection graph where every vertex has exactly one outgoing edge towards the vertex with the smallest clock in its neighbourhood. The algorithm finds all lenght-2 cycles in the connection graph and opens any facility that appears in such a cycle. Then a client j follows the path $P_j(x^t)$ defined by the unique outgoing edges of each vertex and stops before it is about to visit an already visited vertex. The client j is connected to the facility visited last in this path. It is easy to see that there exists no cycle with length more than 2 in the connection graph. This implies that every path stops at a length-2 cycle and this is why the algorithm is guaranteed to assign a facility to every client. The equivalence of the two descriptions of the algorithm follows from this observation, since essentially the only time the algorithm opens a facility is when it is the facility with the smallest clock in the neighbourhood of the client with the smallest clock in its own neighbourhood, which is exactly the length-2 cycle.

To analyse the algorithm we define the random indicator variables X_{ij}^t

and Y_i^t. The expected costs of the algorithm are bounded as follows:

1. Opening cost: $\mathbb{E}[\sum_{i \in F} f_i Y_i^t] \leq \sum_{i \in F} f_i y_i^t$
2. Connection cost: $\mathbb{E}[\sum_{i \in F} \sum_{j \in C} d_{ij}^t X_{ij}^t] \leq 6 \sum_{i \in F} \sum_{j \in C} d_{ij}^t x_{ij}^t$
3. Switching cost: $g\mathbb{E}[\sum_{j \in C} \mathbf{1}\{X_{ij}^t \neq X_{ij}^{t+1} \text{ for some } i \in F\}] \leq 7g|Z^t|$

If these bounds hold, then the overall approximation ratio of the algorithm is indeed 14 because of the switching cost approximation factor and the preprocessing of the fractional solution which incurred an additional 2 approximation factor.

The opening cost is relatively easy to bound. If the facility is not in any client's support in the graph, then $\mathbb{E}[Y_i^t] = 0$. Otherwise, let j denote the client with the smallest clock in the neighbourhood of i in the connection graph and let $F(j)$ denote the facilities in j's neighbourhood in the support graph. Facility i will be opened only if it has the smallest clock in $F(j)$, which by the properties of the exponential clocks would mean that $\mathbb{E}[Y_i^t] = \frac{y_i^t}{\sum_{i' \in F(j)} y_{i'}^t} = y_i^t$, where the last equality holds because $\sum_{i' \in Fj} y_{i'}^t = \sum_{i' \in F(j)} x_{i'j}^t = 1$ by the second preprocessing and the optimality conditions.

The switching cost is not that obvious but the main idea is that if only one client k changes their connection variables, then she only affects her own connection path and the clients which included k in their paths either before or after the change of k. But the paths that contain k are at most 3 in expectation (which is proven in the connection cost bound) so in expectation there are at most 7 clients whose connection paths are different. Therefore the switching cost at time t is at most $7|Z^t|$.

Because the proof for the connection cost bound is rather complicated, we will drop the index t and use the simplified notation x_{ij}, y_i, $\mathcal{SG}$, $\mathcal{CG}$ and P_j. By the triangle inequality, the connection cost of a client is at most the sum of the connection costs of the edges of her connection path, therefore we will bound instead $\mathbb{E}[\sum_{j \in C} d_t(P_j)] = \sum_{\{i,j\} \in \mathcal{SG}} d_{ij}^t \mathbb{E}[|j' \in C | (i,j') or (j',i) \in P_{j'}|]$. If the expected number of paths that use an edge $\{i,j\}$ is $6x_{ij}^t$ then the overall bound follows. To do that, a bound on the probability that a connection path starts with a given prefix is required.

Consider a client j_0 and its connection path P_{j_0}. The set $prefix(P_{j_0})$ is the set of all prefixes of this path, that is all the subpaths of P_{j_0} that start at

j_0. The set $C(i)$ denotes the set of clients that are in the neighbourhood of a facility i in the support graph $\mathcal{SG}$. Similarly, $F(j)$ is the set of facilities in the neighbourhood of j in $\mathcal{SG}$. The sets $C(i_1, \ldots, i_\ell)$ and $F(j_1, \ldots, j_\ell)$ denote the union of the respective neighbourhoods. Finally, for a set of facilities F', $y(F') = \sum_{i \in F'} y_i$.

$$\Pr[\langle j_0, i_1, j_1, \ldots, i_k, j_k, i_{k+1} \rangle \in prefix(P_{j_0})] \leq \Pi_{\ell=1}^{k} \frac{1}{|C(i_1, \ldots, i_\ell)|} \Pi_{\ell=0}^{k} \frac{y_{i_{\ell+1}}}{y(F(j_0, j_1, \ldots, j_\ell))}$$

$$\Pr[\langle j_0, i_1, j_1, \ldots, i_k, j_k \rangle \in prefix(P_{j_0})] \leq \Pi_{\ell=1}^{k} \frac{1}{|C(i_1, \ldots, i_\ell)|} \Pi_{\ell=0}^{k-1} \frac{y_{i_{\ell+1}}}{y(F(j_0, j_1, \ldots, j_\ell))}$$

For the first probability, the event is equivalent to the event that all the arcs of the path exist in $\mathcal{CG}$. First we bound the probability that the arcs $(i_1, j_1), \ldots, (i_k, j_k)$ exist in $\mathcal{CG}$. For such an edge (i_ℓ, j_ℓ) to exist, j_ℓ has to be the smallest clock of all the clients in $C(i_\ell)$. Also, since $j_{\ell-1} \in C(i_\ell)$, it must hold that $R_{j_\ell} < R_{j_{\ell-1}}$ and by repeating this argument we can see that the exponential clocks must increase as ℓ increases in order for all the arcs to exist. Therefore, the arcs exist only if

$$R_{j_\ell} = \min\{R_j | j \in C(i_1, \ldots, i_\ell)\}, \forall \ell \in [k]$$

Each of these exponential clocks has a rate of 1, so the rate of that minimum value, given by the exponential clocks' properties is $\frac{1}{|C(i_1, \ldots, i_k)|}$. By the memorylessness property, this probability is the same if we replace k with each ℓ. So in the end we get the first term of the bound, which states that the probability that all these arcs exist in the connection graph is $\Pi_{\ell=1}^{k} \frac{1}{|C(i_1, \ldots, i_\ell)|}$. We now need to bound the probability that all the arcs $(j_0, i_1), \ldots, (j_k, i_{k+1})$ exist in the connection graph. Similarly, each of these arcs exists if the exponential clock of the facility at the endpoint of the arc is the smallest for all previous clients. The clock of a facility is distributed exponentially with rate $o_i = y_i$, so each individual probability of $Q_{i_{\ell+1}}$ being the minimum clock is $\frac{y_{i_{\ell+1}}}{y(F(j_0, j_1, \ldots, j_\ell))}$. Again by the memorylessness property, the probability that all these arcs exist is $\Pi_{\ell=0}^{k} \frac{y_{i_{\ell+1}}}{y(F(j_0, j_1, \ldots, j_\ell))}$. The overall bound follows by the fact that the client clocks and the facility clocks are independent. The second probability bound is also proven the same way.

With the prefix probability bound, we can go on to bound the expected number of connection paths that use an edge in the support graph. More

specifically, the expected number of connection paths that visits k clients before going through arc (i,j) ((j,i) respectively) is at most $\frac{x_{ij}}{2^{\max\{0,k-2\}}}$ ($\frac{x_{ij}}{2^{\max\{0,k-1\}}}$ respectively). The proof is based in first estimating the expected number of connection paths that have prefixes that visit k clients before going through the arc. This helps complete the proof of the bound, since the expected number of paths that use an arc (i,j) are then $\sum_{k=1}^{\infty} \frac{x_{ij}}{2^{\max\{0,k-2\}}} = 3x_{ij}$ (and the same holds for (j,i)). So the total number of paths that use an edge $\{i,j\}$ of the support graph are at most $6x_{ij}$.

Chapter 4

An Algorithm for the Online Facility Location Problem with Switching Costs

In this chapter, we consider the online facility location problem with switching costs. In this setting, we seek to minimize the total cost of the solution over time, which includes the opening costs and the connection costs of each round, as well as the switching costs incurred by the connection changes. We present an online Regularization Algorithm to find a $O(\log m)$-approximate fractional solution and an online Rounding Algorithm which achieves a $O(\log n)$-approximation with the use of competing *exponential clocks*. The analysis follows the same steps as in [16] where the authors present similar regularization and rounding algorithms for the online set cover problem with switching costs, as well as any problem that can be formulated as a convex linear program.

4.1 The Regularization Algorithm

To make it possible for this chapter to be read independently, we will describe again the online facility location problem with switching costs as introduced in [20]. Let $t \in \{1, \dots, T\}$ be the time variable. At each time t, we are given the set of connection costs $\{d_{ij}^t | j \in [n], i \in [m]\}$, where each d_{ij}^t denotes the connection cost between client j and facility i at time t. We also know in advance the opening cost f of the facilities which does not change over time.

Since we study the online version of the problem, at each time t, after we are given the new connection costs, we should return a feasible solution. A solution is described by the vectors $y \in \{0,1\}^n$ and $x \in \{0,1\}^{m \times n}$, where y_i^t is an indicator variable for the opening of facility i at time t and x_{ij}^t is an indicator variable for the connection of client j to facility i at time t. In order for the solution to be feasible, each client should be served, i.e. they should be connected to one facility. Therefore, it should hold that

$$\sum_{i=1}^{m} x_{ij} = 1 \ \forall j \in [n]$$

In addition, no client should be connected to a closed facility, therefore

$$x_{ij} \leq y_i \ \forall j \in [n] \forall i \in [m]$$

The goal is to minimize over time the total opening cost, connection cost and switching cost. The latter is what differentiates this problem from the online facility location problem. The switching cost at time t is given by $\sum_{j=1}^{m} \sum_{i=1}^{n} z_{ij}^t$ where $z_{ij} = \mathbb{1}\{x_{ij}^t \neq x_{ij}^{t-1}\}$. From the definition, a unit of switching cost is incurred whenever the connection status between a client and a facility changes. More formally, the benchmark for our algorithm is:

$$\min f \sum_{t=1}^{T} \sum_{i=1}^{m} y_i^t + \sum_{t=1}^{T} \sum_{j=1}^{n} \sum_{i=1}^{m} x_{ij}^t d_{ij}^t + \sum_{t=1}^{T} \sum_{j=1}^{n} \sum_{i=1}^{m} z_{ij}^t$$

Notice that there is not a corresponding movement cost for the opening and closing of facilities in our setting.

We shall now introduce the linear relaxation of our problem. The LP, denoted as (P), is the following:

$$
\begin{array}{lll}
\min & f\sum_{t=1}^{T}\sum_{i=1}^{m} y_i^t + \sum_{t=1}^{T}\sum_{j=1}^{n}\sum_{i=1}^{m} x_{ij}^t d_{ij}^t + \sum_{t=1}^{T}\sum_{j=1}^{n}\sum_{i=1}^{m} z_{ij}^t & \\
\text{s.t.} & x_{ij}^t \leq y_i^t & \forall t \in [T], \forall i \in [m], \forall j \in [n] \\
& \sum_{i=1}^{m} x_{ij}^t \geq 1 & \forall t \in [T], \forall j \in [n] \\
& z_{ij}^t \geq x_{ij}^t - x_{ij}^{t-1} & \forall t \in [T], \forall i \in [m], \forall j \in [n] \\
& y_i^t \geq 0 & \forall t \in [T], \forall j \in [n] \\
& x_{ij}^t \geq 0 & \forall t \in [T], \forall i \in [m], \forall j \in [n] \\
& z_{ij}^t \geq 0 & \forall t \in [T], \forall i \in [m], \forall j \in [n]
\end{array}
$$

Notice that the covering constraint is an inequality instead of an equality but because of the structure of the objective function and the fact that this is a minimization LP the two are equivalent.

Theorem 2. *The Regularization Algorithm is an online $O(logm)$-approximation algorithm for the relaxed online facility location problem with switching costs described by (P).*

Let $S = \{(y, x) \in \mathbb{R}_+^n \cdot \mathbb{R}_+^{m \times n} \mid x_{ij} \leq y_i \ \forall t \in [T], i \in [m], j \in [n] \text{ and } \sum_{i=1}^{m} x_{ij} \geq 1 \ \forall t \in [T], j \in [n]\}$ be the feasible solution set which is the same for all $t \in [T]$. The aforementioned Regularization Algorithm is the following:

REGULARIZATION ALGORITHM
Parameters: $\epsilon > 0$, $\eta = \ln(1 + n/\epsilon)$
Initialization: Set $y_i^0 = 0 \ \forall i \in [m]$ and $x_{ij}^0 = 0 \ \forall i \in [m], j \in [n]$.
At each time $t = [T]$:

1. Let $d^t \in \mathbb{R}_+^{m \times n}$ be the connection cost vector.
2. Solve the following linear program (P') to obtain the fractional solution (y^t, x^t):

$$(y^t, x^t) = \arg\min_{(y,x) \in S} \left\{ f \sum_{i=1}^{m} y_i + \sum_{j=1}^{n}\sum_{i=1}^{m} x_{ij} \cdot d_{ij}^t + \frac{1}{\eta} \sum_{j=1}^{n}\sum_{i=1}^{m} \left[\left((x_{ij} + \frac{\epsilon}{n}) \ln \frac{x_{ij} + \frac{\epsilon}{n}}{x_{ij}^{t-1} + \frac{\epsilon}{n}} \right) - x_{ij} \right] \right\}$$

For the proof of Theorem 2 we need to present more explicitly the L.P. (P') solved by the Regularization Algorithm at time t, as well as the dual L.P. of (P), denoted by (D).

It is easy to see that (P') is formulated as follows:

$$
\begin{array}{lll}
\min & f\sum_{i=1}^{m} y_i + \sum_{j=1}^{n}\sum_{i=1}^{m} x_{ij}\cdot d_{ij}^t + \frac{1}{\eta}\sum_{j=1}^{n}\sum_{i=1}^{m}\left[\left((x_{ij}+\frac{\epsilon}{n})\ln\frac{x_{ij}+\frac{\epsilon}{n}}{x_{ij}^{t-1}+\frac{\epsilon}{n}}\right) - x_{ij}\right] & \\
\text{s.t.} & x_{ij} \le y_i & \forall i\in[m], \forall j\in[n] \\
& \sum_{i=1}^{m} x_{ij} \ge 1 & \forall j\in[n] \\
& 0 \le y_i \le 1 & \forall j\in[n] \\
& 0 \le x_{ij} \le 1 & \forall i\in[m], \forall j\in[n]
\end{array}
$$

In order to formulate the dual (D), we define the dual variables corresponding to the constraints of the primal L.P as shown next.

- $x_{ij}^t \le y_i^t \rightarrow e_{ij}^t$ for all $t\in[T]$, $i\in[m]$, $j\in[n]$
- $\sum_{i=1}^{m} x_{ij}^t \ge 1 \rightarrow a_j^t$ for all $t\in[T]$, $j\in[n]$
- $z_{ij}^t \ge x_{ij}^t - x_{ij}^{t-1} \rightarrow b_{ij}^t$ for all $t\in[T]$, $i\in[m]$, $j\in[n]$

Then it is easy to see that the dual L.P. (D) is formulated as:

$$
\begin{array}{lll}
\max & \sum_{t=1}^{T}\sum_{j=1}^{n} a_j^t & \\
\text{s.t.} & \sum_{j=1}^{n} e_{ij}^t \le f & \forall t\in[T], \forall i\in[m] \\
& b_{ij}^t \le 1 & \forall t\in[T], \forall i\in[m], \forall j\in[n] \\
& b_{ij}^{t+1} - b_{ij}^t \le d_{ij}^t + e_{ij}^t - a_j^t & \forall t\in[T], \forall i\in[m], \forall j\in[n] \\
& b_{ij}^t \ge 0 & \forall t\in[T], \forall i\in[m], \forall j\in[n] \\
& e_{ij}^t \ge 0 & \forall t\in[T], \forall i\in[m], \forall j\in[n] \\
& a_j^t \ge 0 & \forall t\in[T], \forall j\in[n]
\end{array}
$$

To prove the performance of the Regularization Algorithm as stated in Theorem 2, we will show that the set of dual variables of the solutions that (P') returns is a feasible solution for the (D) within a factor of

$(1 + (1 + \epsilon') \ln(1 + \frac{m}{\epsilon'}))$ of the optimal. More specifically, we will prove the relationship of the optimal primal solutions the Algorithm provides with its dual optimal solutions using the KKT conditions for (P') and its dual. Consequently, we will prove that these dual optimal solutions are feasible solutions for the dual of (P). Hence we will prove the relationship of the solution of our Algorithm to the optimal solution of (D), whose cost equals the cost of the optimal solution of (P).

We define e^*_{ij} to be the optimal dual variables corresponding to the precedence constraints of (P'), i.e. the constraints of the form $x_{ij} \leq y_i$, and a^*_j to be the optimal dual variables corresponding to the covering constraints, i.e. the ones of the form $\sum_{i=1}^{m} x_{ij} \geq 1$.
The KKT conditions for the optimal solutions of (P'), denoted by (y^*, x^*), and its dual L.P. are the following:

$$x^*_{ij} - y^*_i \leq 0 \ \forall j \in [n] \forall i \in [m] \tag{4.1}$$

$$1 - \sum_{i=1}^{m} x^*_{ij} \leq 0 \ \forall j \in [n] \tag{4.2}$$

$$e^*_{ij}(x^*_{ij} - y^*_i) = 0 \ \forall j \in [n] \forall i \in [m] \tag{4.3}$$

$$a^*_j(1 - \sum_{i=1}^{m} x^*_{ij}) = 0 \ \forall j \in [n] \tag{4.4}$$

$$f - \sum_{j=1}^{n} e^*_{ij} \geq 0 \ \forall i \in [m] \tag{4.5}$$

$$y^*_i(f - \sum_{j=1}^{n} e^*_{ij}) = 0 \ \forall i \in [m] \tag{4.6}$$

$$d^t_{ij} + \frac{1}{\eta} \ln \frac{x^*_{ij} + \frac{\epsilon}{n}}{x^{t-1}_{ij} + \frac{\epsilon}{n}} + e^*_{ij} - a^*_j \geq 0 \ \forall i \in [m] \forall j \in [n] \tag{4.7}$$

$$x^*_{ij}(d^t_{ij} + \frac{1}{\eta} \ln \frac{x^*_{ij} + \frac{\epsilon}{n}}{x^{t-1}_{ij} + \frac{\epsilon}{n}} + e^*_{ij} - a^*_j) = 0 \ \forall i \in [m] \forall j \in [m] \tag{4.8}$$

$$a^*_j \geq 0 \ \forall j \in [n] \tag{4.9}$$

$$e^*_{ij} \geq 0 \ \forall i \in [m] \forall j \in [n] \tag{4.10}$$

Claim 3. *The set of optimal solutions for each round of the dual L.P. of* (P')*,* $(a^{*,t}, e^{*,t})$*, which satisfy the KKT conditions* (4.1) - (4.10)*, along with an appropriate* b_{ij}^t*, consist of a feasible solution for* (D)*.*

Proof of Claim 3. Set the variables of (D) at time t to be:

$$a_j^t = a_j^{*,t}$$

and

$$e_{ij}^t = e_{ij}^{*,t}$$

Also, set

$$b_{ij}^{t+1} = \frac{1}{\eta} \ln \frac{1 + \frac{\epsilon}{n}}{x_{ij}^{*,t} + \frac{\epsilon}{n}}$$

To prove that the solution above is feasible for (D), we prove that it satisfies its constraints one by one:

- By (4.5): $\sum_{j=1}^{n} e_{ij}^t \leq f$
- $b_{ij}^t = \frac{1}{\eta} \ln \frac{1+\frac{\epsilon}{n}}{x_{ij}^{*,t-1}+\frac{\epsilon}{n}} = \frac{1}{\ln(1+\frac{n}{\epsilon})} \ln \frac{1+\frac{\epsilon}{n}}{x_{ij}^{*,t-1}+\frac{\epsilon}{n}} \overset{x^{*,t} \geq 0}{\leq} \frac{1}{\ln(1+\frac{n}{\epsilon})} \ln \frac{1+\frac{\epsilon}{n}}{\frac{\epsilon}{n}} = \frac{\ln(\frac{n}{\epsilon}+1)}{\ln(\frac{n}{\epsilon}+1)} = 1$
- $b_{ij}^{t+1} - b_{ij}^t = -\frac{1}{\eta} \ln \frac{x_{ij}^{*,t}+\frac{\epsilon}{n}}{x_{ij}^{*,t-1}+\frac{\epsilon}{n}} \leq d_{ij}^t + e_{ij}^t - a_j^t$, where the last inequality holds due to (4.7).
- $b_{ij}^t = \frac{1}{\eta} \ln \frac{1+\frac{\epsilon}{n}}{x_{ij}^{*,t-1}+\frac{\epsilon}{n}} \overset{x^{*,t} \leq 1}{\geq} \frac{1}{\eta} \ln \frac{1+\frac{\epsilon}{n}}{1+\frac{\epsilon}{n}} = 0$
- By (4.10): $e_{ij}^t \geq 0$
- By (4.9): $a_j^t \geq 0$

□

To bound the total switching cost and the total service cost (opening and connection costs) of our Algorithm, we are going to need the following inequalities.

$$h - k \leq h \ln(h/k) \text{ for any } h, k > 0 \tag{4.11}$$

$$\sum_i h_i \ln(h_i/k_i) \leq \Big(\sum_i h_i\Big) \log \frac{\sum_i h_i}{\sum_i k_i} \tag{4.12}$$

Claim 4. *The switching cost of our Algorithm, M, is at most $\eta(1+\frac{\epsilon m}{n})$ times the cost of the dual feasible solution of Claim 3.*

Proof of Claim 4. Let M_t be the switching cost of our Algorithm at time t.

$$\begin{aligned}
M_t &= \sum_{t: x_{ij}^{*,t} > x_{ij}^{*,t-1}} (x_{ij}^{*,t} - x_{ij}^{*,t-1}) \\
&= \eta \tfrac{1}{\eta} \sum_{t: x_{ij}^{*,t} > x_{ij}^{*,t-1}} (x_{ij}^{*,t} - x_{ij}^{*,t-1}) \\
&= \eta \tfrac{1}{\eta} \sum_{t: x_{ij}^{*,t} > x_{ij}^{*,t-1}} (x_{ij}^{*,t} + \tfrac{\epsilon}{n} - (x_{ij}^{*,t-1} + \tfrac{\epsilon}{n})) \\
&\overset{(4.11)}{\leq} \eta \sum_{t: x_{ij}^{*,t} > x_{ij}^{*,t-1}} (x_{ij}^{*,t} + \tfrac{\epsilon}{n}) \tfrac{1}{\eta} \ln \tfrac{x_{ij}^{*,t} + \frac{\epsilon}{n}}{x_{ij}^{*,t-1} + \frac{\epsilon}{n}} \\
&\overset{x_{ij}^{*,t} > x_{ij}^{*,t-1} \geq 0, (4.8)}{=} \eta \sum_{i=1}^{m} \sum_{j=1}^{n} (x_{ij}^{*,t} + \tfrac{\epsilon}{n}) a_j^{*,t} \\
&= \eta \sum_{j=1}^{n} a_j^{*,t} \Big(\sum_{i=1}^{m} (x_{ij}^{*,t} + \tfrac{\epsilon}{n}) \Big) \\
&= \eta \sum_{j=1}^{n} a_j^{*,t} \Big(\sum_{i=1}^{m} x_{ij}^{*,t} + \tfrac{\epsilon m}{n} \Big) \\
&\overset{(4.4)}{=} \eta \sum_{j=1}^{n} a_j^{*,t} (1 + \tfrac{\epsilon m}{n}) \\
&= \eta (1 + \tfrac{\epsilon m}{n}) \sum_{j=1}^{n} a_j^{*,t}
\end{aligned}$$

Hence,

$$M = \sum_{t=1}^{T} M_t \leq \eta (1 + \frac{\epsilon m}{n}) \sum_{t=1}^{T} \sum_{j=1}^{n} a_j^{*,t} \tag{4.13}$$

□

Claim 5. *The total service cost of the Algorithm, S, is less than the cost of the dual feasible solution of Claim 3.*

Proof of Claim 5.

$$
\begin{aligned}
S &= \sum_{t=1}^{T}\Big[f\sum_{i=1}^{m}y_i^{*,t}+\sum_{j=1}^{n}\sum_{i=1}^{m}x_{ij}^{*,t}d_{ij}^{t}\Big]\\
&\overset{(4.6),(4.8)}{=}\sum_{t=1}^{T}\Big[\sum_{i=1}^{m}y_i^{*,t}\Big(\sum_{j=1}^{n}e_{ij}^{*,t}\Big)+\sum_{j=1}^{n}\sum_{i=1}^{m}x_{ij}^{*,t}(a_j^{*,t}-e_{ij}^{*,t}-\tfrac{1}{\eta}\ln\tfrac{x_{ij}^{*,t}+\frac{\epsilon}{n}}{x_{ij}^{*,t-1}+\frac{\epsilon}{n}})\Big]\\
&=\sum_{t=1}^{T}\Big[\sum_{i=1}^{m}\sum_{j=1}^{n}(y_i^{*,t}-x_{ij}^{*,t})e_{ij}^{*,t}+\sum_{j=1}^{n}\sum_{i=1}^{m}x_{ij}^{*,t}a_j^{*,t}-\tfrac{1}{\eta}\sum_{j=1}^{n}\sum_{i=1}^{m}x_{ij}^{*,t}\ln\tfrac{x_{ij}^{*,t}+\frac{\epsilon}{n}}{x_{ij}^{*,t-1}+\frac{\epsilon}{n}}\Big]\\
&\overset{(4.3)}{=}\sum_{t=1}^{T}\Big[\sum_{j=1}^{n}a_j^{*,t}\Big(\sum_{i=1}^{m}x_{ij}^{*,t}\Big)-\tfrac{1}{\eta}\sum_{j=1}^{n}\sum_{i=1}^{m}x_{ij}^{*,t}\ln\tfrac{x_{ij}^{*,t}+\frac{\epsilon}{n}}{x_{ij}^{*,t-1}+\frac{\epsilon}{n}}\Big]\\
&\overset{(4.4)}{=}\sum_{t=1}^{T}\sum_{j=1}^{n}a_j^{*,t}-\tfrac{1}{\eta}\sum_{j=1}^{n}\sum_{i=1}^{m}\Big[\sum_{t=1}^{T}(x_{ij}^{*,t}+\tfrac{\epsilon}{n})\ln\tfrac{x_{ij}^{*,t}+\frac{\epsilon}{n}}{x_{ij}^{*,t-1}+\frac{\epsilon}{n}}-\tfrac{\epsilon}{n}\sum_{t=1}^{T}\ln\tfrac{x_{ij}^{*,t}+\frac{\epsilon}{n}}{x_{ij}^{*,t-1}+\frac{\epsilon}{n}}\Big]\\
&\overset{(4.12)}{\leq}\sum_{t=1}^{T}\sum_{j=1}^{n}a_j^{*,t}-\tfrac{1}{\eta}\sum_{j=1}^{n}\sum_{i=1}^{m}\Big[\Big(\sum_{t=1}^{T}(x_{ij}^{*,t}+\tfrac{\epsilon}{n})\Big)\ln\tfrac{\sum_{t=1}^{T}(x_{ij}^{*,t}+\frac{\epsilon}{n})}{\sum_{t=1}^{T}(x_{ij}^{*,t-1}+\frac{\epsilon}{n})}-\tfrac{\epsilon}{n}\ln\tfrac{x_{ij}^{*,T}+\frac{\epsilon}{n}}{x_{ij}^{*,0}+\frac{\epsilon}{n}}\Big]
\end{aligned}
$$

Notice that if the two terms in the bracket of the right hand of the inequality above cancel each other out, the inequality holds:

$$
-\frac{\epsilon}{n}\ln\frac{x_{ij}^{*,T}+\frac{\epsilon}{n}}{x_{ij}^{*,0}+\frac{\epsilon}{n}}\overset{x_{ij}^{*,0}=0}{=}(x_{ij}^{*,0}+\frac{\epsilon}{n})\ln\frac{x_{ij}^{*,0}+\frac{\epsilon}{n}}{x_{ij}^{*,T}+\frac{\epsilon}{n}}\overset{(4.11)}{\geq}x_{ij}^{*,0}-x_{ij}^{*,T}
$$

$$
\Big(\sum_{t=1}^{T}(x_{ij}^{*,t}+\frac{\epsilon}{n})\Big)\ln\frac{\sum_{t=1}^{T}(x_{ij}^{*,t}+\frac{\epsilon}{n})}{\sum_{t=1}^{T}(x_{ij}^{*,t-1}+\frac{\epsilon}{n})}\overset{(4.11)}{\geq}\sum_{t=1}^{T}(x_{ij}^{*,t}+\frac{\epsilon}{n})-\sum_{t=1}^{T}(x_{ij}^{*,t-1}+\frac{\epsilon}{n})=x_{ij}^{*,0}-x_{ij}^{*,T}
$$

Therefore, it holds that

$$
S\leq\sum_{t=1}^{T}\sum_{j=1}^{n}a_j^{*,t}
$$

□

We can now easily prove the performance of our Algorithm.

Proof of Theorem 2. By Claim 4 and Claim 5, the total cost of our Algorithm is:

$$\begin{aligned} S + M &\leq [1 + \eta(1 + \tfrac{\epsilon m}{n})] \sum_{t=1}^{T} \sum_{j=1}^{n} a_j^{*,t} \\ &= [1 + \ln(1 + \tfrac{n}{\epsilon})(1 + \tfrac{\epsilon m}{n})] \sum_{t=1}^{T} \sum_{j=1}^{n} a_j^{*,t} \\ &\overset{Claim\ 3}{\leq} [1 + \ln(1 + \tfrac{n}{\epsilon})(1 + \tfrac{\epsilon m}{n})]\mathrm{OPT}(D) \\ &\overset{\epsilon' = \frac{\epsilon m}{n}}{=} [1 + (1 + \epsilon')\ln(1 + \tfrac{m}{\epsilon'})]\mathrm{OPT}(D) \\ &= [1 + (1 + \epsilon')\ln(1 + \tfrac{m}{\epsilon'})]\mathrm{OPT}(P) \end{aligned}$$

□

4.2 The Rounding Algorithm

4.2.1 Exponential Clocks

The rounding technique we apply to round the fractional solution of the previous section is based on exponential clocks, as are all the previous rounding algorithms for this problem we will discuss.

Definition 6. *Exponential clocks are competing independent exponential random variables. A random variable X is distributed according to the exponential distribution with rate λ, denoted as $X \sim \exp(\lambda)$, if its probability density function is*

$$f_X(x) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

The independent random variables are said to be *competing* because the random variable with the smallest value indicates which facility or connection is chosen in the rounding.

Some of the useful properties of exponential clocks are the following:

1. If $X \sim \exp(\lambda)$ and $c > 0$, then $\frac{X}{c} \sim \exp(\lambda c)$.
2. Let $X_1, \ldots, X_k$ be independent random variables for which $X_i \sim \exp(\lambda_i)$ $\forall i$. Then it holds that:
   - $\min\{X_1, \ldots, X_k\} \sim \exp(\lambda_1 + \ldots + \lambda_k)$
   - $\Pr[X_i \leq \min_{j \neq i}\{X_j\}] = \frac{\lambda_i}{\lambda_1 + \ldots + \lambda_k}$

Also, note that for any two independent exponential random variables X, Y it holds that

$$\Pr[X \leq Y \mid X \geq t] = \frac{\lambda_X}{\lambda_X + \lambda_Y} e^{-\lambda_Y t} \tag{4.14}$$

Exponential clocks can be used to round fractional solutions of covering problems in general. In [14], the authors present an algorithm for the Multiway Cut problem which uses exponential clocks along with other techniques and they also provide a randomized rounding algorithm for the Set Cover problem based on exponential clocks. We present the latter as an easy exercise for the reader who is not familiar with exponential clocks as well as for later reference to some parts of the proof that are similar to the proof for the rounding algorithm of the Online Facility Location problem with Switching Costs.

The instance of the Set Cover problem consists of a set of elements $E = \{e_1, \ldots, e_n\}$ and m subsets of E, $\mathcal{S} = \{S_1, \ldots, S_m\}$, where each $S \in \mathcal{S}$ is associated with a cost c_S. The standard relaxation for the Set Cover problem is:

$$\begin{aligned} \min \quad & \sum_{S \in \mathcal{S}} c_S y_S \\ & \sum_{S : e \in S} y_S \geq 1 \ \forall e \in E \end{aligned}$$

SET COVER ROUNDING ALGORITHM
Initialization: Choose i.i.d. random variables $Z_S \sim \exp(1)$ $\forall S \in \mathcal{S}$.
Output

$$\bigcup_{e \in E} \arg\min_{S : e \in S} \left\{ \frac{Z_S}{y_S} \right\}$$

where $y \in \mathbb{R}_+^{|\mathcal{S}|}$ is the fractional solution of the LP.

Theorem 7. *The Set Cover Rounding Algorithm yields an integral solution of cost at most* $(\ln S_{\max} + 1)$ *times the cost of the optimal solution, where* $S_{\max} = \max_{S \in \mathcal{S}} |S|$.

Proof. It suffices to prove that $\Pr[S \in \mathcal{S} \text{ is chosen}] \leq (\ln(|S|) + 1) y_S$.

Let $A_{e,S}$ denote the event on which S has the smallest clock value among all sets that could cover e. More formally,

$$A_{e,S} \leftrightarrow \frac{Z_S}{y_S} \leq \min_{S' \in \mathcal{S}} \left\{ \frac{Z_{S'}}{y_{S'}} \mid e \in S', S' \neq S \right\}$$

Therefore, it holds that:

$$
\begin{aligned}
\Pr[S \in \mathcal{S} \text{ is chosen}] &= \Pr[\exists e : A_{e,S}] \\
&= \Pr[\exists e : A_{e,S} \mid Z_S/y_S < a] \cdot \Pr[Z_S/y_S < a] \\
&\qquad + \Pr[\exists e : A_{e,S} \mid Z_S/y_S \geq a] \cdot \Pr[Z_S/y_S \geq a] \\
&\leq \Pr[Z_S/y_S < a] + \Pr[\exists e : A_{e,S} \mid Z_S/y_S \geq a] \cdot \Pr[Z_S/y_S \geq a] \\
&= \int_0^a y_S e^{-y_S x} dx + \Pr[\exists e : A_{e,S} \mid Z_S/y_S \geq a] \int_a^\infty y_S e^{-y_S x} dx \\
&= (1 - e^{-y_S a}) + \Pr[\exists e : A_{e,S} \mid Z_S/y_S \geq a] e^{-y_S a} \\
&\overset{\text{U.B.}}{\leq} (1 - e^{-y_S a}) + e^{-y_S a} \sum_{e \in S} \Pr[A_{e,S} \mid Z_S/y_S \geq a] \\
&\overset{1-e^x \leq x}{\leq} y_S a + e^{-y_S a} \sum_{e \in S} \Pr[A_{e,S} \mid Z_S/y_S \geq a] \\
&\overset{(4.14)}{=} y_S a + e^{-y_S a} \sum_{e \in S} \left[\frac{y_S}{\sum_{S' : e \in S'} y_{S'}} e^{-a\left(\sum_{S' : e \in S'} y_{S'} - y_S\right)} \right] \\
&\overset{\sum_{S' : e \in S'} y_{S'} \geq 1}{\leq} y_S(a + |S| e^{-a}) \\
&\overset{a \triangleq \ln |S|}{=} y_S(\ln |S| + 1) \\
&\leq y_S(\ln S_{\max} + 1)
\end{aligned}
\tag{4.15}
$$

□

4.2.2 The Rounding Algorithm and its analysis

The Rounding Algorithm for the Online Facility Location Problem with Switching Costs is the following:

ROUNDING ALGORITHM
Parameter: $a \geq 0$
Initialization: Choose i.i.d. random variables $Z_{ij} \sim \exp(1)\ \forall i \in [m], j \in [n]$.
At each time $t = [T]$**:**

1. Let x_{ij}^t be the value of the connection variable in the fractional solution of the Regularization Algorithm.
2. If $\frac{Z_{ij}}{x_{ij}^t} < a$ then open facility i and connect j to i.
3. For each client $j \in [n]$ who has not yet been connected, open facility $i = \arg\min_{i' \in [m]} \frac{Z_{i'j}}{x_{i'j}^t}$ and connect j to i.

Theorem 8. *The Rounding Algorithm yields an integral solution with cost at most* $(\log n + 1)$ *times the cost of the fractional solution of the Regularization Algorithm for the Online Facility Location Problem with Switching Costs.*

Proof. The proof follows three steps to bound the total switching cost, opening cost and connection cost of the integral solution.

Switching Cost:

We break down the total movement from time $t-1$ to time t in the fractional solution into $m \times n$ intermediate steps, on each of which only the value of exactly one x_{ij} is changed. We take first all the x_{ij}'s whose value increases and then all the x_{ij}'s whose value decreases, thus managing to preserve a feasible solution in all the intermediate steps. This way, the total switching cost from time $t-1$ to time t of the fractional solution does not change while the integral switching cost could only increase due to possible changes in the intermediate steps.

First, we will prove the bound in the case the connection variable decreases by δ, i.e. $x_{ij}^t = x_{ij}^{t-1} - \delta$.

Let $Y_{ij} = \min_{i' \neq i} \frac{Z_{i'j}}{x^t_{i'j}}$. By the properties of the exponential clocks $Y_{ij} \sim \exp(\lambda)$ where $\lambda = \sum_{i' \neq i} x^t_{i'j} \geq 1 - x^t_{ij}$.

When the value of x_{ij} decreases, the value of $\frac{Z_{ij}}{x_{ij}}$ increases. Therefore, no connection that had not been chosen at time $t-1$ can be chosen at time t due to the first condition. The same is true for the second condition for the connection ij, but due to this same condition and the increase of $\frac{Z_{ij}}{x_{ij}}$, another connection could turn minimal that had not been chosen in the previous time step. This is the only case, when a switching cost is incurred. The probability of this event is bounded by:

$$\begin{aligned}
&\Pr\left[\tfrac{Z_{ij}}{x_{ij}^{t-1}} \leq Y_{ij} < \tfrac{Z_{ij}}{x_{ij}^{t-1}-\delta} \text{ and } Y_{ij} \geq a\right] \\
&= \int_a^\infty f_Y(k) \Pr\left[\tfrac{Z_{ij}}{x_{ij}^{t-1}} \leq k < \tfrac{Z_{ij}}{x_{ij}^{t-1}-\delta}\right] dk \\
&= \int_a^\infty \lambda e^{-\lambda k}(e^{-(x_{ij}^{t-1}-\delta)k} - e^{-x_{ij}^{t-1}k}) dk \\
&= \tfrac{\lambda}{x_{ij}^t - \delta + \lambda} e^{-a(x_{ij}^{t-1}-\delta+\lambda)} - \tfrac{\lambda}{x_{ij}^{t-1}} e^{-a(x_{ij}^{t-1}+\lambda)} \\
&\leq e^{-a} - \tfrac{1}{\delta+1} e^{-a(\delta+1)} \\
&= e^{-a}(1 - \tfrac{1}{\delta+1} e^{-a\delta}) \\
&\leq e^{-a}(1 - e^{-(a+1)\delta}) \\
&\leq e^{-a}(a+1)\delta
\end{aligned} \tag{4.16}$$

Hence, the expected switching cost in the integral solution caused by a δ decrease in x_{ij} is at most $e^{-a}(a+1)\delta$.

We will now bound the switching cost in the case x_{ij} increases from time $t-1$ to time t, i.e. $x^t_{ij} = x^{t-1}_{ij} + \delta$. Connection ij could be chosen due to the first or the second condition of the rounding algorithm, while no other connection can be chosen due to this change.

The probability that connection ij is chosen at time t and not at time $t-1$ due to the first condition is:

$$\Pr\left[\frac{Z_{ij}}{x_{ij}^{t-1}+\delta} < a \leq \frac{Z_{ij}}{x_{ij}^{t-1}}\right] = e^{-ax_{ij}^{t-1}}(1-e^{-a\delta}) \overset{x_{ij}^{t-1}\geq 0}{\leq} a\delta \tag{4.17}$$

The probability that connection ij is chosen at time t and not at time $t-1$ due to the second condition is:

$$\begin{aligned}
&\Pr\left[\tfrac{Z_{ij}}{x_{ij}^{t-1}+\delta} \leq Y_{ij} \leq \tfrac{Z_{ij}}{x_{ij}^{t-1}} \text{ and } \tfrac{Z_{ij}}{x_{ij}^{t-1}+\delta} \geq a\right] \\
&\leq \Pr\left[\tfrac{Z_{ij}}{x_{ij}^{t-1}+\delta} \leq Y_{ij} \leq \tfrac{Z_{ij}}{x_{ij}^{t-1}} \text{ and } Y_{ij} \geq a\right] \\
&= \int_a^\infty f_Y(k) \Pr\left[\tfrac{Z_{ij}}{x_{ij}^{t-1}+\delta} \leq Y_{ij} \leq \tfrac{Z_{ij}}{x_{ij}^{t-1}}\right] dk \\
&= \int_a^\infty \lambda e^{-\lambda k}(e^{-x_{ij}^{t-1}k} - e^{-(x_{ij}^{t-1}+\delta)k})dk \\
&= \tfrac{\lambda}{x_{ij}^t+\lambda} e^{-a(x_{ij}^{t-1}+\lambda)} - \tfrac{\lambda}{x_{ij}^{t-1}+\delta+\lambda} e^{-a(x_{ij}^{t-1}+\delta+\lambda)} \\
&\leq e^{-a} - \tfrac{1}{\delta+1} e^{-a(\delta+1)} \\
&= e^{-a}(1 - \tfrac{1}{\delta+1} e^{-a\delta}) \\
&\leq e^{-a}(1 - e^{-(a+1)\delta}) \\
&\leq e^{-a}(a+1)\delta
\end{aligned} \tag{4.18}$$

Hence, the expected switching cost in the integral solution caused by a δ increase in x_{ij} is at most $\max\{a\delta, e^{-a}(a+1)\delta\} = e^{-a}(a+1)\delta$ (by (4.17) and (4.18)). Taking into account (4.16), the total switching cost of the integral solution is expected to be at most $e^{-a}(a+1)\delta$ times the total switching cost of the fractional solution.

Opening Cost:

To bound the opening cost of the facilities, we recall the Set Cover Rounding Algorithm from the previous section. It is easy to see that we can follow almost the exact same analysis to prove that the probability that facility i is opened at time t is at most:

$$(a + ne^{-a})x_{ij}^t \leq (a + ne^{-a})y_i^t$$

Therefore, the total opening cost of the facilities in the integral solution is at most $(a + ne^{-a})$ times the total opening cost of the fractional solution.

Connection Cost:

Finally, each connection ij is chosen with probability at most:

$$\begin{aligned}
&\Pr\left[\tfrac{Z_{ij}}{x_{ij}} < a\right] + \Pr\left[\tfrac{Z_{ij}}{x_{ij}} = \min_{i\in[m]}\{\tfrac{Z_{ij}}{x_{ij}}\} \mid \tfrac{Z_{ij}}{x_{ij}} \geq a\right] \cdot \Pr\left[\tfrac{Z_{ij}}{x_{ij}} \geq a\right] \\
&= (1 - e^{-ax^t_{ij}}) + \Pr\left[\tfrac{Z_{ij}}{x_{ij}} = \min_{i\in[m]}\{\tfrac{Z_{ij}}{x_{ij}}\} \mid \tfrac{Z_{ij}}{x_{ij}} \geq a\right] e^{-ax^t_{ij}} \\
&= (1 - e^{-ax^t_{ij}}) + \tfrac{x_{ij}}{\sum_{i'} x^t_{i'j}} e^{-a\sum_{i'\neq i} x^t_{i'j}} e^{-ax^t_{ij}} \qquad (4.19) \\
&\leq (1 - e^{-ax^t_{ij}}) + x^t_{ij} e^{-a} \\
&\leq ax^t_{ij} + x^t_{ij} e^{-a} \\
&= (a + e^{-a}) x^t_{ij}
\end{aligned}$$

Therefore, the total connection cost in the integral solution is at most $(a + e^{-a})$ times the total connection cost of the fractional solution.

If we set $a \triangleq \ln n$, then the approximation ratios for the switching, opening and connection costs are $(\ln n+1)\frac{1}{n}$, $(\ln n+1)$ and $(\ln n+\frac{1}{n})$ respectively.

We can conclude that the total cost of the integral solution is at most $(\ln n + 1)$ times the total cost of the fractional solution. □

Since the fractional solution is $O(\log m)$ competitive with respect to the optimal solution, the complete integral algorithm achieves an $O(\log m \log n)$ competitive ratio.

Notice that the analysis holds even if the opening cost is not the same for all facilities and the switching cost is not 1 for all changes.

Bibliography

[1] Jacob Abernethy, Peter L. Bartlett, Niv Buchbinder, and Isabelle Stanton. A regularization approach to metrical task systems. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6331 LNAI, pages 270–284, 2010.

[2] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph (Seffi) Naor. The Online Set Cover Problem. *SIAM Journal on Computing*, 39(2):361–370, 2009.

[3] An, Hyung-chan; Norouzi-Fard, Ashkan; Svensson, Ola;. Dynamic Facility Location via Exponential Clocks. pages 708–721, 2015.

[4] Lachlan L H Andrew, Siddharth Barman, Katrina Ligett, Minghong Lin, Adam Meyerson, Alan Roytman, and Adam Wierman. A Tale of Two Metrics: Simultaneous Bounds on Competitiveness and Regret. *The 26th Annual Conference on Learning Theory*, 30:741–763, 2013.

[5] B Anthony and Anupam Gupta. Infrastructure leasing problems. *Integer Programming and Combinatorial Optimization*, pages 424–438, 2007.

[6] Vijay Arya, Vijay Arya, Naveen Garg, Naveen Garg, Rohit Khandekar, Rohit Khandekar, Adam Meyerson, Adam Meyerson, Kamesh Munagala, Kamesh Munagala, Vinayaka Pandit, and Vinayaka Pandit. Local search heuristic for k-median and facility location problems. *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, 33(3):21–29, 2001.

[7] Nikhil Bansal, Niv Buchbinder, Aleksander Ma??dry, and Joseph Naor. A polylogarithmic-competitive algorithm for the k-server problem. In *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 267–276, 2011.

[8] Nikhil Bansal, Niv Buchbinder, and Joseph (Seffi) Naor. Towards the randomized k-server conjecture: a primal-dual approach. In *SODA '10 Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete algorithms*, pages 40–55, 2010.

[9] Nikhil Bansal, Anupam Gupta, Ravishankar Krishnaswamy, Kirk Pruhs, Kevin Schewior, and Cliff Stein. A 2-competitive algorithm for online convex optimization with switching costs. *Leibniz International Proceedings in Informatics, LIPIcs*, 40:96–109, 2015.

[10] Avrim Blum and Carl Burch. On-line learning and the Metrical Task Systems problem. In *COLT '97 Proceedings of the tenth annual conference on Computational learning theory*, pages 45–53, 1997.

[11] Allan Borodin, Ran El-Yaniv, and Ran Borodin, Allan; El-Yaniv. Online Computation and Competitive Analysis, 1998.

[12] Allan Borodin, Nathan Linial, and Michael E. Saks. An optimal on-line algorithm for metrical task system. *Journal of the ACM*, 39(4):745–763, 1992.

[13] Niv Buchbinder. Unified Algorithms for Online Learning and Competitive Analysis. *Colt*, 23:1–18, 2012.

[14] Niv Buchbinder, Joseph (Seffi) Naor, and Roy Schwartz. Simplex partitioning via exponential clocks and the multiway cut problem. *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing - STOC '13*, page 535, 2013.

[15] Niv Buchbinder and Joseph (Seffi) Naor. The Design of Competitive Online Algorithms via a Primal—Dual Approach. *Foundations and Trends® in Theoretical Computer Science*, 3(2–3):93–263, 2009.

[16] Joseph (Seffi) Buchbinder, Niv; Chen, Shahar; Naor. Competitive analysis via regularization. *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 436–444, 2014.

[17] Jaroslaw Byrka and Karen Aardal. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. *SIAM Journal on Computing*, 39(6):2212, 2007.

[18] Moses Charikar and Sudipto Guha. Improved Combinatorial Algorithms for Faciltiy Location Problems. *SIAM Journal on Computing*, 34:803–824, 2005.

[19] Fabián a. Chudak and David B. Shmoys. Improved Approximation Algorithms for the Uncapacitated Facility Location Problem. *SIAM Journal on Computing*, 33(1):1–25, 2003.

[20] David Eisenstat, Claire Mathieu, and Nicolas Schabanel. Facility location in evolving metrics. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8573 LNCS(PART 2):459–470, 2014.

[21] Dimitris Fotakis. Incremental algorithms for Facility Location and k-Median. *Theoretical Computer Science*, 361(2-3):275–313, 2006.

[22] Dimitris Fotakis. On the Competitive Ratio for Online Facility Location. *Algorithmica*, 50(1):1–57, 2008.

[23] Dimitris Fotakis. Online and Incremental Algorithms for Facility Location. *ACM SIGACT*, 42(1):97–131, 2011.

[24] Yoav Freund and Robert E Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[25] Sudipto Guha and Samir Khuller. Greedy Strikes Back: Improved Facility Location Algorithms. *Journal of Algorithms*, 31(1):228–248, 1999.

[26] Anupam Gupta, Kunal Talwar, and Udi Wieder. Changing bases: Multistage optimization for matroids and matchings. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8572 LNCS, pages 563–575, 2014.

[27] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, Vijay V Vazirani, K Jain, E Markakis, A Saberi, and V V Vazirani. Greedy Facility Location Algorithms Analyzed Using Dual Fitting with Factor-Revealing LP. *Lecture Notes in Computer Science*, 2129:127–137, 2002.

[28] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. *STOC 02 Proceedings of the thiryfourth annual ACM symposium on Theory of computing*, pages 731–740, 2002.

[29] Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k-Median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM*, 48(2):274–296, 2001.

[30] Kwang-Sung Jun, Francesco Orabona, Stephen Wright, and Rebecca Willett. Improved Strongly Adaptive Online Learning using Coin Betting. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 54:943–951, 2017.

[31] Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. In *Information and Computation*, volume 222, pages 45–58, 2013.

[32] Jyh Han Lin and Jeffrey Scott Vitter. Approximation algorithms for geometric median problems. *Information Processing Letters*, 44(5):245–249, 1992.

[33] Minghong Lin, Adam Wierman, Alan Roytman, Adam Meyerson, and Lachlan L.H. Andrew. Online optimization with switching cost. *ACM SIGMETRICS Performance Evaluation Review*, 40(3):98, 2012.

[34] A. Meyerson. Online facility location. *Proceedings 2001 IEEE International Conference on Cluster Computing*, (January), 2001.

[35] Adam Meyerson. The parking permit problem. *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, 2005:274–282, 2005.

[36] Aryan Mokhtari, Shahin Shahrampour, Ali Jadbabaie, and Alejandro Ribeiro. Online optimization in dynamic environments: Improved regret rates for strongly convex problems. In *2016 IEEE 55th Conference on Decision and Control, CDC 2016*, pages 7195–7201, 2016.

[37] Chandrashekhar Nagarajan and David P. Williamson. Offline and online facility leasing. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5035 LNCS:303–315, 2008.

[38] Francesco Orabona. Parameter-Free Convex Learning through Coin Betting. *ICML, AutoML workshop*, pages 1–7, 2016.

[39] David B. Shmoys, Éva Tardos, and Karen Aardal. Approximation algorithms for facility location problems. *STOC '97: 29th Annual Symposium on Theory of Computing*, 20244(20244):265–274, 1997.