WORST CASE INSTANCES ARE FRAGILE

Average Case and Smoothed Competitive Analysis of Algorithms

Guido Schäfer

Dissertation
zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

Saarbrücken
2004

Tag des Kolloquiums:	30. April 2004
Dekan:	Prof. Dr. Jörg Eschmeier
Prüfungsausschuss:	Prof. Dr. Reinhard Wilhelm (Vorsitzender)
	Prof. Dr. Dr.-Ing. E. h. Kurt Mehlhorn (Berichterstatter)
	Prof. Dr. Stefano Leonardi (Berichterstatter)
	Dr. Ernst Althaus

ABSTRACT

We describe three results in this thesis. We first present a heuristic improvement for a shortest path problem, which we termed *single-source many-targets shortest path problem*. In this problem, we need to compute a shortest path from a source node to a node that belongs to a designated target set. Dijkstra's algorithm can be used to solve this problem. We are interested in the single-source many-targets shortest path problem since matching algorithms repeatedly solve this problem so as to compute a maximum weighted matching in a bipartite graph. The heuristic is easy to implement and, as our experiments show, considerably reduces the running time of the matching algorithm. We provide an average case analysis which shows that a substantial fraction of queue operations is saved by Dijkstra's algorithm if the heuristic is used.

The second and third result are about the extension of smoothed complexity to the area of online algorithms. Smoothed complexity has been introduced by Spielman and Teng to explain the behavior of algorithms performing well in practice while having a poor worst case complexity. The idea is to add some noise to the initial input instances by perturbing the input values slightly at random and to analyze the performance of the algorithm on these perturbed instances. In this work, we apply this notion to two well-known online algorithms.

The first one is the multi-level feedback algorithm (MLF), minimizing the average flow time on a sequence of jobs released over time, when the processing times of these jobs are not known. MLF is known to work very well in practice, though it has a poor competitive ratio. As it turns out, the smoothed competitive ratio of MLF improves exponentially with the amount of random noise that is added; on average, MLF even admits a constant competitive ratio. We also prove that our bound is asymptotically tight.

The second algorithm that we consider is the work function algorithm (WFA) for metrical task systems, a general framework to model online problems. It is known that WFA has a poor competitive ratio. We believe that due to its generality it is interesting to analyze the smoothed competitive ratio of WFA. Our analysis reveals that the smoothed competitive ratio of WFA is much better than its worst case competitive ratio and that it depends on certain topological parameters of the underlying metric. We present asymptotic upper and matching lower bounds on the smoothed competitive ratio of WFA.

KURZZUSAMMENFASSUNG

In der vorliegenden Arbeit werden drei Resultate vorgestellt. Als erstes beschreiben wir eine Heuristik für eine Variante des kürzesten Wege Problems, welches wir das *Single-Source Many-Targets Shortest Path Problem* nennen. Gegeben sind ein ungerichteter Graph mit nichtnegativen Kantenkosten, ein Quellknoten s und eine Menge T von Zielknoten. Die Aufgabe ist es, einen kürzesten Weg vom Quellknoten s zu einem der Zielknoten in T zu berechnen. Dieses Problem wird wiederholt von Matching Algorithmen gelöst, um ein maximal gewichtetes Matching in bipartiten Graphen zu berechnen. Der Algorithmus von Dijkstra kann verwendet werden, um das *Single-Source Many-Targets Shortest Path Problem* zu lösen. Unsere Heuristik lässt sich leicht implementieren und erzielt, wie unsere Experimente zeigen, eine signifikante Laufzeitverbesserung des Matching Algorithmus. In den Experimenten auf Zufallsgraphen konnten wir eine Laufzeitverbesserung von bis zu einem Faktor 12 beobachten. Wir präsentieren eine *Average Case* Analyse, in der wir zeigen, dass die Heuristik auf Zufallsinstanzen eine nicht unerhebliche Anzahl von Operationen in der Ausführung von Dijkstra's Algorithmus einspart.

Im zweiten Teil der Arbeit erweitern wir die kürzlich von Spielman und Teng eingeführte *Smoothed Complexity* auf den Bereich der online Algorithmen. Die *Smoothed Complexity* ist ein neues Komplexitätsmaß, mit dem man versucht, die Effizienz eines Algorithmus in der Praxis in adäquater Weise zu repräsentieren. Die grundlegende Idee ist, die Eingabeinstanzen mehr oder weniger stark zufällig zu perturbieren, d. h. zu stören, und die Effizienz eines Algorithmus anhand seiner erwarteten Laufzeit auf diesen perturbierten Instanzen festzumachen. Im allgemeinen ist die *Smoothed Complexity* eines Algorithmus sehr viel geringer als seine *Worst Case Complexity*, wenn die *Worst Case* Instanzen künstlichen oder konstruierten Instanzen entsprechen, die in der Praxis so gut wie nie auftreten. Spielman und Teng führten die *Smoothed Complexity* im Zusammenhang mit der Laufzeit als Effizienzkriterium ein. Die zugrunde liegende Idee lässt sich jedoch auch auf andere Kriterien erweitern.

In dieser Arbeit übertragen wir das Konzept der *Smoothed Complexity* auf online Algorithmen. Generell wird die Effizienz eines online Algorithmus anhand seines *Competitive Ratio* gemessen. Dieser gibt jedoch oftmals die tatsächliche Effizienz des Algorithmus in der Praxis nicht akkurat wieder. Es liegt daher nahe, sich der Idee der *Smoothed Complexity* zu bedienen und die Effizienz eines online Algorithmus anhand seines *Smoothed Competitive Ratio* zu messen. Wir verwenden diese neue Idee, um die Effizienz von zwei wohlbekannten online Algorithmen zu analysieren.

Der erste ist bekannt als der *Multi-Level Feedback Algorithm* (MLF) und wird zum *Scheduling* von Prozessen verwendet, deren Ausführungszeiten nicht bekannt sind. Hierbei ist das Ziel, die durchschnittliche Flusszeit (*average flow time*) zu minimieren, d. h. die durchschnittliche Zeit, die die Prozesse im System verbringen. Sind die Ausführungszeiten aus dem Bereich $[1, 2^K]$, so hat MLF einen *Competitive Ratio* von $\Theta(2^K)$. Dennoch erweist sich dieser Algorithmus in der Praxis als äußerst effizient; er wird u. a. in Windows NT und Unix verwendet. Wir analysieren MLF unter der Verwendung des *Partial Bit Randomization Models*, d. h. wir nehmen an, dass die Ausführungszeiten ganze Zahlen aus dem Bereich $[1, 2^K]$ sind, die perturbiert werden, indem man die letzten k Bits durch Zufallsbits ersetzt. Für dieses Modell zeigen wir, dass MLF im wesentlichen einen *Smoothed Competitive Ratio* von $O(2^{K-k})$ hat. Insbesondere impliziert dies, dass der erwartete *Competitive Ratio* von MLF konstant ist, wenn die Ausführungszeiten zufällig aus dem Bereich $[1, 2^K]$ gewählt werden. Desweiteren beweisen wir untere Schranken, die zeigen, das unsere Analyse bis auf einen konstanten Faktor scharf ist. Für eine Vielzahl anderer *Smoothing Models* zeigen wir, dass MLF einen *Smoothed Competitive Ratio* von $\Omega(2^K)$ hat.

Der zweite Algorithmus, den wir betrachten, ist der *Work Function Algorithm* (WFA) für *Metrical Task Systems*. Gegeben ist ein ungerichteter Graph G mit nicht-negativen Kantenkosten. Der online Algorithmus befindet sich zu Beginn in einem Startknoten s und muss eine Folge von Aufträgen (*tasks*) bearbeiten. Hierbei spezifiziert ein Auftrag für jeden Knoten des Graphen die Ausführungskosten (*request cost*), die entstehen, wenn der Algorithmus den Auftrag in diesem Knoten bearbeitet. Der Algorithmus kann sich im Graphen G bewegen, wodurch Reisekosten (*travel cost*) der zurückgelegten Distanz entstehen. Das Ziel ist es, die Folge von Aufträgen zu bearbeiten und dabei die gesamten Ausführungskosten plus Reisekosten zu minimieren. Eine Vielzahl von online Problemen lassen sich als *Metrical Task Systems* formulieren. Die Analyse des *Smoothed Competitive Ratio* von WFA ist daher besonders interessant. Es ist bekannt, dass WFA einen *Competitive Ratio* von $\Theta(n)$ hat, wobei n die Anzahl der Knoten in G ist. In der Analyse verwenden wir ein *Symmetric Additive Smoothing Model*, um die Ausführungskosten zu perturbieren. In diesem Modell werden zu den Ausführungskosten Zufallszahlen addiert, die bezüglich einer symmetrischen Distribution mit Erwartungswert Null gewählt werden. Unsere Analyse zeigt, dass der *Smoothed Competitive Ratio* von WFA von bestimmten topologischen Parametern des Graphen G abhängt, wie der minimalen Kantenlänge $U_{\min}$, dem maximalen Grad D, dem Kantendurchmesser $diam$, etc. Ist zum Beispiel das Verhältnis zwischen maximaler und minimaler Kantenlänge in G durch eine Konstante beschränkt, erhalten wir einen *Smoothed Competitive Ratio* von $O(diam(U_{\min}/\sigma + \log(D)))$ und von $O(\sqrt{n(U_{\min}/\sigma + \log(D))})$, wobei σ die Standardabweichung der zugrundeliegenden Distribution bezeichnet. Insbesondere erhalten wir für Perturbationen der Größenordnung $\sigma = \Theta(U_{\min})$ einen *Smoothed Competitive Ratio* von $O(\log(n))$ auf vollständigen Graphen und von $O(\sqrt{n})$ auf Liniengraphen. Wir zeigen auch, dass unsere Analyse bis auf einen konstanten Faktor scharf ist.

ACKNOWLEDGEMENTS

I am indebted to many people that assisted in completing the work on this thesis. First of all, I would like to thank my *Doktorvater*, Kurt Mehlhorn, for his support, faith, and generosity throughout the preparation of this thesis. It has been a great pleasure to be a member of the Algorithms and Complexity group at the Max-Planck-Institut für Informatik at Saarbrücken. I enjoyed the international atmosphere and the excellent research environment. I am especially grateful to Holger Bast, Susan Hert, Sven Thiel, Hisao Tamaki, Gerhard Weikum, Roxane Wetzel, and Anja Becker.

This work would have been impossible without the financial support through a Graduiertenkolleg fellowship, granted by the Deutsche Forschungsgemeinschaft, through a Ph. D. position at the Max-Planck-Institut für Informatik, and through the AMORE project, granted by the Directorate-General for Research of the European Commission.

I am grateful to a very special group of people at the Dipartimento di Informatica e Sistemistica at the University of Rome, "La Spienza". Through them I experienced that there can be much more to research than I had previously imagined. I would like to thank Stefano Leonardi for being an advisor and a friend, Alberto Marchetti-Spaccamela for making my stays in Rome possible, Giorgio Ausiello, Luca Becchetti & la nonna di fiori, Tanya Buhnik ("So boring!"), Andrea Vitaletti, Debora Donato, and Luigi Laura. Furthermore, I am indebted to Tjark Vredeveld, also for proofreading this thesis.

I would also like to express my gratitude to friends that contributed to the completion of this work, though not directly, yet more than they might think: Lorenzo Lancellotti, Marta, Elena, and Beppe, Tobias Polzin, Naveen Sivadasan, René Beier, Mark Hubertus, Michael Klepper & Duffy, and Kerstin Wöffler.

I owe a debt of gratitude to my family for their support and belief during the last three and a half years: Sabine, Udo and Nicole, my parents Gisela and Peter-Josef, moreover, Meike Krott and Wolfram, and Roswitha and Günter Krott. I am deeply indebted to Sabine Krott for her constant support, strength, and unshakeable faith.

Finally, I would like to use this opportunity to thank my maths teacher, Gernot Zünskes, and my computer science teacher, Friedrich Esser. Somehow this venture is rooted there.

CONTENTS

1. INTRODUCTION

One of the major objectives in the area of algorithmics is to design algorithms that have a good performance, the performance of an algorithm representing the time needed to solve a problem, the quality of a computed solution, etc. Often the overall performance of an algorithm is subsumed under its *worst case complexity*, a complexity measure which reflects the performance of the algorithm on a most unfortunate input instance. As an example, consider an algorithm ALG that sorts a given sequence $\check{I}$ of n numbers and let $T(\check{I})$ denote the time needed by ALG to sort $\check{I}$. The worst case complexity $\varphi(n)$ of ALG is then defined as the maximum of $T(\check{I})$ over all input sequences $\check{I}$ of n numbers. An obvious reason for characterizing the performance of an algorithm by its worst case complexity is that it provides a very strong notion of performance guarantee. In our example we are guaranteed that ALG sorts any given sequence of n numbers within $\varphi(n)$ time. In many applications this guarantee is essential. Another reason for the popularity of worst case complexity is that it can usually be estimated easily.

On the other hand the worst case complexity of an algorithm might be an over-pessimistic estimation of its actual performance in practice. In many applications we are interested in the typical behavior of an algorithm rather than in its worst case behavior. For example, it might very well be that instances that actually force the algorithm into its worst case behavior are highly artificial or constructed and therefore almost certainly never occur in practice. As a consequence, the worst case complexity of the algorithm does not accurately reflect its actual performance and it therefore fails to explain the good behavior of an algorithm in practice.

An alternative complexity measure is the *average case complexity*. Here we assume that input instances are chosen randomly according to a specific probability distribution. The average case complexity then measures the expected performance of an algorithm on these instances. Although we might obtain deeper insights into the performance of an algorithm by analyzing its average case complexity, for most applications it is not natural to assume that the input is random.

We therefore seek a complexity measure that truly reflects the performance of an algorithm. Recently, Spielman and Teng proposed *smoothed complexity*, a complexity measure that attempts to explain the success of algorithms that are known to work well in practice while having a poor worst case performance. The basic idea is simple: Given an input instance we perturb the input values slightly at random and analyze the expected performance of the algorithm on these perturbed instances. Intuitively, one could regard smoothed complexity

as a measure of fragility of worst case instances. If the worst case instances of an algorithm are highly artificial, the smoothed complexity of the algorithm is usually much smaller than its worst case complexity. Spielman and Teng proved that the simplex method, an algorithm that solves linear programming problems, with a certain pivot rule has polynomial smoothed complexity, while its worst case complexity was known to be exponential. Another argument as to why it makes sense to analyze the performance of an algorithm on perturbed instances is that often the input values themselves are estimates and therefore inherently noisy.

While Spielman and Teng introduced smoothed complexity having the running time of an algorithm as performance criteria in mind, it also makes sense to apply smoothed complexity to other criteria.

In this thesis, we present three results. The first one is a heuristic improvement for a variant of a shortest path problem, which we termed the *single-source many-targets shortest path problem*. Given a directed graph with non-negative edge costs, a source node s, and a designated subset T of target nodes, the task is to compute a shortest path from s to a node in T. Dijkstra's algorithm can easily be adapted to solve this problem. Our interest in this problem originates from the fact that matching algorithms repeatedly solve this problem so as to compute a maximum weighted matching in a bipartite graph. The heuristic is easy to implement and significantly reduces the running time of the matching algorithm. In our experiments on random instances, we observed an improvement in running time by a factor of up to 12. We support this observation by providing a partial average case analysis on the number of queue operations that are saved during the execution of Dijkstra's algorithm if the heuristic is used. More specifically, for random graphs with average degree c and uniform random edge costs in $[0, 1]$, we show that on expectation at least a fraction of $1 - (2 + \ln(c))/c$ of the queue operations is saved.

The second and third result of this thesis are concerned with the extension of smoothed complexity to the area of online algorithms. Online problems have a notion of time associated with the input. That is, the input is revealed over time and an online algorithm has to take decisions without knowing the whole input sequence. In contrast, an offline algorithm knows the entire input sequence in advance. So far, the performance of an online algorithm was commonly measured by means of its *(worst case) competitive ratio*, which is defined as the maximum over all input instances of the ratio between the cost of the online algorithm and the cost of an optimal offline algorithm. Several online algorithms that are known to work well in practice have a poor competitive ratio. We therefore propose to characterize the performance of an online algorithm by its *smoothed competitive ratio* rather than by its worst case competitive ratio. We apply this new notion to two well-known online algorithms, the *multi-level feedback algorithm* for the non-clairvoyant scheduling problem and the *work function algorithm* for problems that can be formulated as metrical task systems. For both algorithms it turns out that the smoothed competitive ratio is much better than the worst case competitive ratio. Moreover, from the analyses we obtain new insights into the behavior of the algorithms.

In a non-clairvoyant scheduling problem, jobs are released over time and have to be scheduled on a machine while the actual processing times of the jobs are not known. In this setting, we also allow that jobs are preempted. The objective is to minimize the *average flow time*, i.e., the average time spent by jobs in the system. This problem has several natural applications in practice. A very successful algorithm for the non-clairvoyant scheduling problem is the multi-level feedback algorithm (MLF), which is also used in Windows NT and Unix. Although MLF performs very well in practice, it has a poor worst case performance guarantee. More precisely, if the processing times are in $[1, 2^K]$ for some $K \geq 0$, MLF has a competitive ratio of $\Omega(2^K)$. In this work, we attempt to explain the success of MLF in practice using the novel notion of smoothed competitiveness. We smoothen the initial integral processing times in $[1, 2^K]$ by changing the k least significant bits at random. Under this smoothing model, we prove that MLF has a smoothed competitive ratio of essentially $O(2^{K-k})$. That is, the smoothed competitive ratio of MLF improves exponentially with the amount of random noise that is added; on random processing times, MLF even admits a constant competitive ratio. We also prove that this bound is asymptotically tight. Moreover, we establish a lower bound of $\Omega(2^K)$ for various other smoothing models, including symmetric smoothing models suggested by Spielman and Teng.

Metrical task systems can be described as follows. An online algorithm resides in a graph G and may move in this graph at a cost equal to the distance. The algorithm has to service a sequence of tasks, arriving one at a time. Each task specifies for each node a request cost that is incurred if the algorithm services the task in this particular node. The objective is to minimize the total request cost plus the total travel cost. Several important online problems can be modeled as metrical task systems. A powerful algorithm for this whole class of online problems is the work function algorithm (WFA). Here, too, it is known that the algorithm has a poor competitive ratio of $\Theta(n)$, where n denotes the number of nodes in the underlying graph G. However, very little is known about the performance of WFA in practice. We believe that due to its generality it is interesting to analyze the smoothed competitive ratio of WFA. We smoothen the request costs of the tasks by means of a symmetric additive smoothing model; that is, to each request cost we add a random number that is chosen from a symmetric distribution with mean zero and standard deviation σ. Our analysis reveals that the smoothed competitive ratio of WFA depends on certain topological parameters of the underlying graph G, such as the minimum edge length $U_{\min}$, maximum degree D, edge diameter $diam$, etc. For example, if the ratio between the maximum and the minimum edge length of G is bounded by a constant, we obtain a smoothed competitive ratio of $O(diam(U_{\min}/\sigma + \log(D)))$ and of $O(\sqrt{n(U_{\min}/\sigma + \log(D))})$. In particular, for perturbations with $\sigma = \Theta(U_{\min})$, WFA has smoothed competitive ratio $O(\log(n))$ on a clique and $O(\sqrt{n})$ on a line. We also prove that all our bounds are asymptotically tight.

Smoothed competitive analysis is a natural alternative to worst case competitive analysis, and we strongly believe that this new notion will help to characterize the performance of online algorithms accurately in the future.

The thesis is structured as follows. In Chapter 2 we define different complexity measures and introduce smoothed complexity. Moreover, we review some basic concepts, results, and techniques from probability theory. The subject of Chapter 3 is the heuristic improvement for the single-source many-targets shortest path problem. In Chapter 4 we introduce smoothed competitive analysis and investigate the smoothed competitiveness of the multi-level feedback algorithm. Then, in Chapter 5, we present a smoothed competitive analysis of the work function algorithm for metrical task systems. Finally, we offer a short conclusion. A list of notations and their definitions that are used throughout this work can be found in Appendix A.

2. PRELIMINARIES

We discuss different complexity measures, define *smoothed complexity* and four different smoothing models. After that, we review some basic concepts and techniques from probability theory.

2.1 Worst Case and Average Case Complexity

The worst case complexity measures the performance of an algorithm under the assumption that the algorithm is run on most unfortunate input instances. More precisely, for an algorithm ALG and an input instance $\check{I}$, let $T(\check{I})$ be a performance measure of ALG on input $\check{I}$, e.g., its running time. Let $\mathcal{I}$ denote the set of all input instances to ALG, and let $\mathcal{I}(n)$ refer to all input instances of size n. Subsequently, we assume that an input instance of size n consists of n real values. The *worst case complexity* $\varphi(n)$ of ALG on input instances of size n is defined as the maximum of $T(\check{I})$ over all instances $\check{I}$ of size n, i.e.,

$$\varphi(n) := \max_{\check{I} \in \mathcal{I}(n)} T(\check{I}).$$

The worst case complexity $\varphi(n)$ of an algorithm provides a very strong notion of performance guarantee: It states that on every input of size n the algorithm is guaranteed to have a performance of at most $\varphi(n)$.

On the other hand, worst case complexity does often not reflect the typical behavior of an algorithm in practice. Worst case instances might be pathological instances that rarely occur in practice and therefore the performance on these instances is not representative for the overall performance of the algorithm. Put differently, the worst case complexity of an algorithm might be an over-pessimistic estimation of its true performance. Hence, this complexity measure may fail to characterize the actual performance of an algorithm in practice.

Another complexity measure is the average case complexity. Here we assume that each input instance $\check{I}$ is chosen from a probability distribution f over $\mathcal{I}(n)$. The *average case complexity* $\varphi(n)$ measures the expected performance of ALG over all instances $\check{I}$ of size n; more formally,

$$\varphi(n) := \mathbf{E}_{\check{I} \xleftarrow{f} \mathcal{I}(n)}[T(\check{I})].$$

In average case analyses we assume that the input is entirely chosen at random. Very often this gives new insights into the algorithm. Real-world instances, however, are most

likely not random. Consequently, also average case complexity does not adequately reflect the performance of an algorithm in practice.

2.2 Smoothed Complexity

Recently, Spielman and Teng [ST01] proposed *smoothed complexity*, a complexity measure that attempts to explain the success of algorithms that are known to work well in practice while having a poor worst case performance. Smoothed complexity can be seen as a hybrid between worst case and average case complexity. The basic idea is to randomly perturb the input instances of an algorithm and to analyze its performance on the perturbed instances. The *smoothed complexity* of an algorithm is expressed in terms of the input size and the magnitude of perturbation. More formally, the smoothed complexity of an algorithm ALG is defined as follows. Let $\check{I} = (\check{x}_1, \ldots, \check{x}_n)$, $\check{x}_j \in \mathbb{R}$ for each $j \in [n]$, be an input instance from $\mathcal{I}(n)$; we call $\check{I}$ the *adversarial*, *original*, or *initial* input instance. We perturb, or *smoothen*, $\check{I}$ by adding some random noise to each input value. For each $\check{x}_j$, $j \in [n]$, we choose some random number ε_j from a probability distribution f and define $x_j := \check{x}_j + \varepsilon_j$. The magnitude of perturbation depends on a *smoothing parameter* σ. We use I to refer to a *perturbed*, or *smoothed*, instance with entries $(x_1, \ldots, x_n)$. The set of all smoothed instances that are obtainable from $\check{I}$ define a neighborhood $N(\check{I}, \sigma)$ of $\check{I}$, whose size depends on the smoothing parameter σ. The smoothed complexity $\varphi(n, \sigma)$ of ALG is defined as the maximum over all adversarial instances $\check{I}$ of size n of the expected performance of ALG over all smoothed instances I in $N(\check{I}, \sigma)$:

$$\varphi(n, \sigma) := \max_{\check{I} \in \mathcal{I}(n)} \mathbf{E}_{I \xleftarrow{f} N(\check{I}, \sigma)}[T(I)]. \tag{2.1}$$

Intuitively, the smoothed complexity of an algorithm is much smaller than its worst case complexity if worst case instances are isolated peaks in the (instance $\times$ performance) space; see Figure 2.1. If we slightly perturb these instances, the performance on the perturbed instances improves drastically. In some sense, in smoothed analysis we attempt to answer the question of how fragile worst case instances are.

The striking result of Spielman and Teng [ST01] was to show that the the simplex method with a certain pivot rule has *polynomial* smoothed complexity if the coefficients of the constraint matrix are perturbed by a normal distribution with mean zero and standard deviation σ. In a series of successive papers [BD02, DST02, SST02, ST03, ST02] smoothed complexity was successfully applied to characterize the performance of other algorithms.

Spielman and Teng introduced smoothed complexity having the running time of an algorithm as performance measure in mind. However, the idea underlying smoothed analysis is generic and, as will be seen in Chapter 4, naturally extends to other performance criteria.

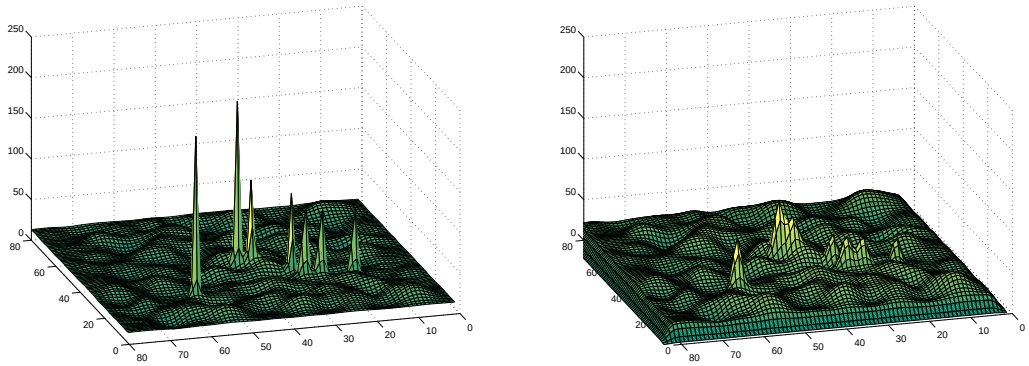

Figure 2.1: Left: Original (instance $\times$ performance) space. Worst case complexity measures the height of the highest peak. Right: Smoothed (instance $\times$ performance) space. Smoothed complexity measures the height of the highest peak in the smoothed space. Since worst case instances are isolated and sharp peaks (on the left), the smoothed complexity is much smaller than the worst case complexity. Very kindly, both figures were provided by Daniel Spielman; see also the Smoothed Analysis Homepage [sah].

2.3 Smoothing Models

The adversarial input instance may be smoothed according to different smoothing models. We discuss four different smoothing models below.

Assume that the adversarial input instance is given as $\check{I} = (\check{x}_1, \ldots, \check{x}_n)$. We refer to a smoothed instance by $I = (x_1, \ldots, x_n)$. Let f be a symmetric distribution with mean zero and standard deviation σ.

Additive Symmetric Smoothing Model. In the additive symmetric smoothing model each $\check{x}_j$, $j \in [n]$, is perturbed symmetrically around its initial value by adding some additive noise. For each $\check{x}_j$, $j \in [n]$, we choose an ε_j independently at random from f and define

$$x_j := \check{x}_j + \varepsilon_j, \quad \text{where } \varepsilon_j \leftarrow f.$$

Observe that in this model the magnitude of perturbation is independent of the initial value $\check{x}_j$.

Additive Relative Symmetric Smoothing Model. The additive relative symmetric smoothing model is similar to the previous one except that the magnitude of distortion depends on the initial instance $\check{I}$. Let $\vartheta : \mathcal{I}(n) \to \mathbb{R}$ be a function on the set of all adversarial instances of size n. For each $\check{x}_j$, $j \in [n]$, we choose an ε_j independently at random from f and define

$$x_j := \check{x}_j + \vartheta(\check{I}) \cdot \varepsilon_j, \quad \text{where } \varepsilon_j \leftarrow f.$$

Spielman and Teng [ST01] defined $\vartheta(\check{I}) := \max_{j \in [n]} \check{x}_j$. However, also other functions might be reasonable. We could even define a different function ϑ_j for each $j \in [n]$.

Relative Smoothing Model. In the relative smoothing model the input is smoothed symmetrically around its initial value by adding some relative noise. For each $\check{x}_j$, $j \in [n]$, we independently choose some ε_j from f and define

$$x_j := \check{x}_j(1 + \varepsilon_j), \quad \text{where } \varepsilon_j \leftarrow f.$$

Spielman and Teng [ST01] introduced the above three models with f being a normal distribution with mean zero and standard deviation σ. In their analysis, they use the additive symmetric smoothing model.

We define a fourth model, the *partial bit randomization model* (see also [BBM03, BCKV04]), which is particularly useful if the input instance consists of K-bit integers. In this model the input values are not smoothed symmetrically.

Partial Bit Randomization Model. Assume that each $\check{x}_j$, $j \in [n]$, is a K-bit integer. We perturb each $\check{x}_j$ by replacing the k least significant bits, for $0 \leq k \leq K$, with some random number. More precisely, for each $\check{x}_j$, $j \in [n]$, we independently choose some random number ε_j from a probability distribution f over $[0, 2^k - 1]$ and define

$$x_j := 2^k \left\lfloor \frac{\check{x}_j}{2^k} \right\rfloor + \varepsilon_j, \quad \text{where } \varepsilon_j \xleftarrow{f} [0, 2^k - 1].$$

For $k = 0$ the smoothed values are equal to the initial values. For $k = K$ the smoothed values are randomly chosen from $[0, 2^K - 1]$.

2.4 Basic Concepts and Techniques from Probability Theory

2.4.1 Sample Space, Events, and Probability Distribution

A *probability space* is a mathematical description of a random experiment. It consists of a *sample space* Ω, which is a discrete set of elementary events, and a *probability distribution* $\mathbf{P}$, which assigns to each *event* $A \subseteq \Omega$ a number representing the "likelihood" that one of the elementary events in A occurs. We use ω to denote an elementary event. A probability distribution $\mathbf{P} : 2^\Omega \to [0, 1]$ is a function satisfying:

1. $\mathbf{P}[\emptyset] = 0$, and $\mathbf{P}[\Omega] = 1$.
2. If $A_1, \ldots, A_n$ is a pairwise disjoint collection of subsets of Ω, i.e., $A_i \cap A_j = \emptyset$ for all pairs $i, j, i \neq j$, then
$$\mathbf{P}\left[\bigcup_{i \in [n]} A_i\right] = \sum_{i \in [n]} \mathbf{P}[A_j]\,.$$

The following properties follow immediately from the definitions above.

Theorem 2.4.1. *Let Ω be a sample space with probability distribution $\mathbf{P}$, and let A and B be two events. Then the following holds.*

1. *For the complement $\bar{A} := \Omega \setminus A$ of an event A, we have $\mathbf{P}[\bar{A}] = 1 - \mathbf{P}[A]$.*
2. *If $A \subseteq B$ then $\mathbf{P}[B] = \mathbf{P}[A] + \mathbf{P}[B \setminus A] \geq \mathbf{P}[A]$.*
3. *For any two events $A, B \subseteq \Omega$,*

$$\mathbf{P}[A \cup B] = \mathbf{P}[A] + \mathbf{P}[B] - \mathbf{P}[A \cap B] \leq \mathbf{P}[A] + \mathbf{P}[B].$$

2.4.2 Conditional Probability and Independence

If some a-priori knowledge of the outcome of an experiment is available, we may want to calculate the probability of an event A given that an event B occurs. This is formalized in the notion of *conditional probabilities*.

Definition 2.4.1 (conditional probability). *Let A and B be two events of Ω. If $\mathbf{P}[B] > 0$ then the* conditional probability of A given that B occurs *is defined as*

$$\mathbf{P}[A \mid B] := \frac{\mathbf{P}[A \cap B]}{\mathbf{P}[B]}.$$

Having introduced the notion of conditional probabilities we can formulate the *total probability theorem*.

Theorem 2.4.2 (total probability theorem). *Let $B_1, \ldots, B_n$ be a partition of Ω, and let $\mathbf{P}[B_i] > 0$ for each $i \in [n]$. Then, for any event A,*

$$\mathbf{P}[A] = \sum_{i \in [n]} \mathbf{P}[A \cap B_i] = \sum_{i \in [n]} \mathbf{P}[A \mid B_i] \, \mathbf{P}[B_i].$$

In general, the occurrence of some event B changes the probability of an event A. That is, in general $\mathbf{P}[A \mid B] \neq \mathbf{P}[A]$. If however the occurrence of B does not influence the probability of A, i.e., $\mathbf{P}[A \mid B] = \mathbf{P}[A]$, we say that A and B are *independent*. Equivalently, A and B are independent if $\mathbf{P}[A \cap B] = \mathbf{P}[A] \, \mathbf{P}[B]$. We generalize this concept in the following definition.

Definition 2.4.2 (independence of events). *A collection $A_1, \ldots, A_n$ of events is* independent *if*

$$\mathbf{P}\left[\bigcap_{i \in S} A_i\right] = \prod_{i \in S} \mathbf{P}[A_i] \quad \textit{for every subset } S \subseteq [n].$$

Observe that *pairwise* independence of the events $A_1, \ldots, A_n$, i.e., for each $i, j, i \neq j$, A_i and A_j are independent, does not imply independence as defined above.

Moreover, we would like to point out that the independence of two events A and B does not imply their conditional independence. More precisely, let C be an event with $\mathbf{P}[C] > 0$. Then

$$\mathbf{P}[A \cap B] = \mathbf{P}[A]\,\mathbf{P}[B] \quad \not\Rightarrow \quad \mathbf{P}[A \cap B \,|\, C] = \mathbf{P}[A \,|\, C]\,\mathbf{P}[B \,|\, C]\,.$$

2.4.3 Random Variables, Expectation, and Variance

A *random variable* X is a real-valued function $X : \Omega \to \mathbb{R}$. For a value x of X, we define the event $(X \leq x) := \{\omega \in \Omega : X(\omega) \leq x\}$.

Definition 2.4.3 (distribution function). *The* distribution function $F_X : \mathbb{R} \to [0,1]$ *of a random variable X is defined as $F_X(x) := \mathbf{P}[X \leq x]$.*

In general, we distinguish between *discrete* and *continuous* random variables. A random variable X is discrete if it only takes values from a finite or countably infinite subset of $\mathbb{R}$, while X is continuous if it has a distribution function F_X whose derivative F'_X is a positive, integrable function. Subsequently, we only consider discrete random variables.

For a discrete random variable X and some $x \in \mathbb{R}$ we can define the event $(X = x) := \{\omega \in \Omega : X(\omega) = x\}$.

Definition 2.4.4 (density function). *The* density function $f_X : \mathbb{R} \to [0,1]$ *of a random variable X is defined as $f_X(x) := \mathbf{P}[X = x]$.*

We call a random variable X *binary* if it only takes values 0 and 1. For an event A, we define the *indicator variable* $X_A : \Omega \to \{0,1\}$ of A as $X_A(\omega) := 1$ if $\omega \in A$, and $X_A(\omega) := 0$ otherwise. Observe that X_A is a binary random variable taking value 1 or 0 with probability $\mathbf{P}[A]$ or $1 - \mathbf{P}[A]$, respectively.

Two discrete random variables X and Y are independent if the events $(X = x)$ and $(Y = y)$ are independent for all x and y.

Definition 2.4.5 (independence of random variables). *A collection $X_1, \ldots, X_n$ of (discrete) random variables is* independent *if the events $(X_i = x_i)$, $i \in [n]$, are independent for all possible choices of x_i, $i \in [n]$, of values of X_i.*

We next introduce the *expectation* of a random variable.

Definition 2.4.6 (expectation of random variable). *The* expectation, *or* mean, $\mathbf{E}[X]$ *of a discrete random variable X is defined as*

$$\mathbf{E}[X] := \sum_{\omega \in \Omega} X(\omega)\mathbf{P}[\omega] \quad \textit{or, equivalently,} \quad \mathbf{E}[X] := \sum_{x \in \mathbb{R}} x\mathbf{P}[X = x]\,,$$

whenever the sum converges absolutely.

(We slightly abuse notation in the second definition, where we actually sum over all $x \in \mathbb{R}$ with $\mathbf{P}[X = x] > 0$.)

If X is a non-negative integer valued random variable then the above definitions are equivalent to $\mathbf{E}[X] = \sum_{x=0}^{\infty} \mathbf{P}[X > x]$.

The expectation $\mathbf{E}[\cdot]$ can be treated as a linear operator. In particular, it has the following properties.

Theorem 2.4.3. *Let X and Y be two random variables.*

1. *If $a, b \in \mathbb{R}$ then $\mathbf{E}[aX + b] = a\mathbf{E}[X] + b$.*
2. *If $f : \mathbb{R} \to \mathbb{R}$ is a function then $\mathbf{E}[f(X)] = \sum_{x \in \mathbb{R}} f(x)\mathbf{P}[X = x]$. Moreover, if f is a linear function then $\mathbf{E}[f(X)] = f(\mathbf{E}[X])$.*
3. *If X and Y are independent then $\mathbf{E}[XY] = \mathbf{E}[X]\mathbf{E}[Y]$.*

The second part of 2 in the above theorem does not generalize to non-linear functions. However, if f is convex, or concave, we can use Jensen's inequality.

A function $f : \mathbb{R} \to \mathbb{R}$ is *convex* if for any $x_1, x_2 \in \mathbb{R}$ and $0 \leq \lambda \leq 1$ the following inequality holds:

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2).$$

f is *concave* if the opposite inequality holds.

Theorem 2.4.4 (Jensen's inequality). *Let X be a random variable and let $f : \mathbb{R} \to \mathbb{R}$ be a convex function. Then $\mathbf{E}[f(X)] \geq f(\mathbf{E}[X])$. If f is concave then $\mathbf{E}[f(X)] \leq f(\mathbf{E}[X])$.*

We say that a random variable X *stochastically dominates* a random variable Y if $\mathbf{P}[X > z] \geq \mathbf{P}[Y > z]$ for each $z \in \mathbb{R}$.

Theorem 2.4.5. *Let X and Y be random variables with finite expectations and assume X stochastically dominates Y. Then $\mathbf{E}[X] \geq \mathbf{E}[Y]$. Equality holds if and only if X and Y are identically distributed.*

We extend the notion of conditional probabilities to the expectation.

Definition 2.4.7. *The* conditional expectation *of X given an event A with $\mathbf{P}[A] > 0$ is defined as*

$$\mathbf{E}[X \mid A] := \sum_{x \in \mathbb{R}} x\mathbf{P}[X = x \mid A].$$

Analogously, for a random variable Y and a fixed real number $y \in \mathbb{R}$ with $\mathbf{P}[Y = y] > 0$, we define

$$\mathbf{E}[X \mid Y = y] := \sum_{x \in \mathbb{R}} x\mathbf{P}[X = x \mid Y = y].$$

The following theorem is very helpful for computing the expectation of a random variable.

Theorem 2.4.6 (total expectation theorem). *Let $A_1, \ldots, A_n$ be a partition of Ω, and let $\mathbf{P}[A_i] > 0$ for each $i \in [n]$. Then, for any random variable X,*

$$\mathbf{E}[X] = \sum_{i \in [n]} \mathbf{E}[X \mid A_i]\mathbf{P}[A_i]\,.$$

The *variance* of a random variable X measures the deviation of X from its expectation $\mathbf{E}[X]$. It is defined as $\mathbf{Var}[X] := \mathbf{E}[(X - \mathbf{E}[X])^2]$. The *standard deviation* σ of X is defined as $\sigma := \sqrt{\mathbf{Var}[X]}$.

Theorem 2.4.7. *Let X and Y be two random variables.*

1. $\mathbf{Var}[X] = \mathbf{E}[X^2] - \mathbf{E}[X]^2$.
2. *For any real value $c \in \mathbb{R}$,* $\mathbf{Var}[cX] = c^2\mathbf{Var}[X]$.
3. *If X and Y are independent,* $\mathbf{Var}[X + Y] = \mathbf{Var}[X] + \mathbf{Var}[Y]$.

2.4.4 Moment Inequalities and Concentration of Measure

We state some inequalities that will be used in subsequent chapters.

Theorem 2.4.8 (Markov's inequality). *Let X be a non-negative random variable. Then, for every $t \in \mathbb{R}^+$,*

$$\mathbf{P}[X \geq t] \leq \frac{\mathbf{E}[X]}{t}.$$

Theorem 2.4.9 (Chebyshev's inequality). *Let X be a random variable with standard deviation σ. Then, for every $t \in \mathbb{R}^+$,*

$$\mathbf{P}[|X - \mathbf{E}[X]| \geq t\sigma] \leq \frac{1}{t^2}.$$

Consider a sequence $X_1, \ldots, X_n$ of n independent binary random variables with $\mathbf{P}[X_i = 1] := p_i$ and $\mathbf{P}[X_i = 0] := 1 - p_i$ for each $i \in [n]$. In the following we are interested in the question of how much the sum $X := \sum_{i \in [n]} X_i$ deviates from its expectation $\mu := \mathbf{E}[X]$.

Theorem 2.4.10 (Chernoff bound). *Let $X_1, \ldots, X_n$ be independent binary random variables. Define $X := \sum_{i \in [n]} X_i$ and $\mu := \mathbf{E}[X]$.*

1. *For any $0 \leq \varepsilon \leq 1$,*

$$\mathbf{P}[X \leq (1 - \varepsilon)\mu] \leq \mathrm{e}^{-\mu\varepsilon^2/2}. \tag{2.2}$$

2. *For any* $\varepsilon > 0$,

$$\mathbf{P}[X \geq (1+\varepsilon)\mu] \leq \left(\frac{\mathrm{e}^{\varepsilon}}{(1+\varepsilon)^{1+\varepsilon}}\right)^{\mu}. \tag{2.3}$$

We can further bound the right hand side of (2.3) as follows

$$\left(\frac{\mathrm{e}^{\varepsilon}}{(1+\varepsilon)^{1+\varepsilon}}\right)^{\mu} \leq \left(\frac{\mathrm{e}}{1+\varepsilon}\right)^{(1+\varepsilon)\mu}.$$

If $\varepsilon > 2\mathrm{e} - 1$, the latter term is at most $2^{-(1+\varepsilon)\mu}$.

We may use the following bounds if the X_i's are non-binary, independent random variables in $[0,1]$.

Theorem 2.4.11 (Chernoff–Hoeffding bound). *Let $X_1, \ldots, X_n$ be independent random variables with $X_i \in [0,1]$ for each $i \in [n]$. Define $X := \sum_{i\in[n]} X_i$ and $\mu := \mathbf{E}[X]$.*

1. *For any $t > 0$,*

$$\mathbf{P}[X \leq \mathbf{E}[X] - t] \leq \mathrm{e}^{-2t^2/n} \quad \textit{and} \quad \mathbf{P}[X \geq \mathbf{E}[X] + t] \leq \mathrm{e}^{-2t^2/n}.$$

2. *For any $0 < \varepsilon < 1$,*

$$\mathbf{P}[X \leq (1-\varepsilon)\mu] \leq \mathrm{e}^{-\mu\varepsilon^2/2} \quad \textit{and} \quad \mathbf{P}[X \geq (1+\varepsilon)\mu] \leq \mathrm{e}^{-\mu\varepsilon^2/3}.$$

Theorem 2.4.12 (Hoeffding Bound). *Let $X_1, \ldots, X_n$ be independent random variables. Define $X := \sum_{i\in[n]} X_i$ and $\mu := \mathbf{E}[X]$.*

1. *If for each $i \in [n]$ and some $k > 0$, $X_i \in [0,k]$, then, for any t,*

$$\mathbf{P}[X \geq t] \leq \left[\left(\frac{t}{\mathrm{e}\mu}\right)^{-t} \mathrm{e}^{-\mu}\right]^{1/k}.$$

2. *If for each $i \in [n]$ and some constants a_i and b_i, $X_i \in [a_i, b_i]$, then, for any $t > 0$,*

$$\mathbf{P}[X \leq \mathbf{E}[X] - t] \leq \exp\left(\frac{-2t^2}{\sum_{i\in[n]}(b_i - a_i)^2}\right) \quad \textit{and} \tag{2.4}$$

$$\mathbf{P}[X \geq \mathbf{E}[X] + t] \leq \exp\left(\frac{-2t^2}{\sum_{i\in[n]}(b_i - a_i)^2}\right). \tag{2.5}$$

Theorem 2.4.13 (method of bounded differences). *Let $X_1, \ldots, X_n$ be independent random variables with $X_i \in I_i$ for each $i \in [n]$. Suppose that the (measurable) function $f : I_1 \times \cdots \times$*

$I_n \to \mathbb{R}$ *satisfies*

$$|f(\vec{x}) - f(\vec{x}')| \leq c_i,$$

whenever the vectors $\vec{x}$ *and* $\vec{x}'$ *differ only in the* i*th component. Define* $Y := f(X_1, \ldots, X_n)$*. Then, for any* $t > 0$*,*

$$\mathbf{P}[|Y - \mathbf{E}[Y]| \geq t] \leq 2\exp\left(\frac{-2t^2}{\sum_{i\in[n]} c_i^2}\right).$$

Theorem 2.4.14 (Kolmogorov's inequality). *Let* $X_1, \ldots, X_n$ *be independent random variables such that* $\mathbf{E}[X_j] = 0$ *for each* $j \in [n]$*. Define* $S_0 := 0$ *and* $S_i := \sum_{j=1}^{i} X_j$*. Then, for any* $\varepsilon > 0$*,*

$$\mathbf{P}\left[\max_{0\leq k\leq n} |S_k| \geq \varepsilon\right] \leq \frac{\mathbf{E}[S_n^2]}{\varepsilon^2}.$$

2.4.5 Common Discrete Random Variables

We list some commonly used random variables.

Discrete Uniform Distribution

A *discrete uniform* random variable X over $[a, \ldots, b]$ takes each value k, $a \leq k \leq b$, with equal probability, i.e., $\mathbf{P}[X = k] := \frac{1}{b-a+1}$. We have $\mathbf{E}[X] = \frac{a+b}{2}$ and $\mathbf{Var}[X] = \frac{(b-a)(b-a+2)}{12}$.

Bernoulli Distribution

A binary random variable X with $\mathbf{P}[X = 1] := p$ and $\mathbf{P}[X = 0] := 1-p$ is called a *Bernoulli* random variable. We can think of p and $1 - p$ being the *success* and *failure* probability of a trial. We have $\mathbf{E}[X] = p$ and $\mathbf{Var}[X] = p(1 - p)$.

Binomial Distribution

We perform n independent Bernoulli trials $X_1, \ldots, X_n$ (with success probability p) and define $X := \sum_{i\in[n]} X_i$ as the number of successes. The variable X is called a *binomial* random variable. We will also use the notation $\mathbf{Bin}[n, p]$ to refer to a binomial distribution with parameters n and p. The probability that the number of successes equals $k \in \{0, \ldots, n\}$ is

$$\mathbf{P}[X = k] := \binom{n}{k} p^k (1 - p)^{n-k}.$$

We have $\mathbf{E}[X] = np$ and $\mathbf{Var}[X] = np(1 - p)$.

Poisson Distribution

A *Poisson* variable X with parameter $\lambda > 0$ is a random variable such that

$$\mathbf{P}[X = k] := \mathrm{e}^{-\lambda}\frac{\lambda^k}{k!}, \quad k = 0, 1, 2, \ldots$$

We have $\mathbf{E}[X] = \lambda$ and $\mathbf{Var}[X] = \lambda$. If $\lambda = np$ for very large n and very small p, $\mathbf{Bin}[n, p]$ approximates X, i.e.,

$$\mathbf{P}[X = k] \approx \binom{n}{k} p^k (1-p)^{n-k}.$$

Geometric Distribution

We perform n independent Bernoulli trials (with success probability p) and define X as the number of trials until a *success* occurs for the first time. X is a *geometric* random variable with parameter p and

$$\mathbf{P}[X = k] := (1-p)^{k-1}p, \quad k = 1, 2, 3, \ldots$$

We have $\mathbf{E}[X] = \frac{1}{p}$ and $\mathbf{Var}[X] = \frac{1-p}{p^2}$.

2.4.6 A Powerful Technique to Prove Correlations

Two events A and B are *positively correlated* if $\mathbf{P}[A \mid B] \geq \mathbf{P}[A]$. A and B are *negatively correlated* if $\mathbf{P}[A \mid B] \leq \mathbf{P}[A]$.

Next, we review a technique to prove positive or negative correlation of two events A and B. The technique will turn out to be extremely powerful in Chapters 3 and 4. Essentially, this technique enables to prove that two variables are correlated if the following conditions hold:

1. The probability space forms a *distributive lattice*.
2. The probability distribution is *log-supermodular*.
3. The events A and B are *monotone increasing* or *monotone decreasing*.

The technique is described in the book by Alon and Spencer [AS00].

Definition 2.4.8 (distributive lattice). *A* lattice $(L, \leq, \vee, \wedge)$ *is a partially*[1] *ordered set* $(L, \leq)$ *in which every two elements* x *and* y *have a unique minimal upper bound,* $x \vee y$*, called the* join *of* x *and* y*, and a unique maximal lower bound,* $x \wedge y$*, called the* meet *of* x *and* y*. A lattice* $(L, \leq, \vee, \wedge)$ *is* distributive *if for all* $x, y, z \in L$*,*

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z).$$

[1] A relation $\leq$ partially orders L if it is (i) *reflexive*, i.e., $a \leq a$ for all $a \in L$, (ii) *transitive*, i.e., if $a \leq b$ and $b \leq c$ then $a \leq c$ for any $a, b, c \in L$, and (iii) *antisymmetric*, i.e., if $a \leq b$ and $b \leq a$ then $a = b$ for any $a, b \in L$.

Example 2.4.1. *Let 2^N denote the power set of $N := [n]$. Every subset $L \subseteq 2^N$ which is closed under union and intersection forms a distributive lattice $(L, \subseteq, \cup, \cap)$.*

Example 2.4.2. *Let $L := \{0,1\}^n$ be the set of all binary vectors of length n. For any two vectors $x, y \in L$ we write $x \leq y$ if x is componentwise less than or equal to y, i.e., if $x_i \leq y_i$ for each $i \in [n]$. Moreover, we define $x \vee y$ as the componentwise* or *of x and y, i.e., $(x \vee y)_i := x_i \vee y_i$ for each $i \in [n]$, and analogously $x \wedge y$ as the componentwise* and *of x and y. Observe that $x, y \leq x \vee y$ and $x, y \geq x \wedge y$. It can easily be verified that $(L, \leq, \vee, \wedge)$ is a distributive lattice.*

Definition 2.4.9 (log-supermodular function). *Let $(L, \leq, \vee, \wedge)$ be a distributive lattice. A function $\xi : L \to \mathbb{R}^+$ is* log-supermodular *if for all $x, y \in L$*

$$\xi(x) \cdot \xi(y) \leq \xi(x \vee y) \cdot \xi(x \wedge y).$$

Example 2.4.3. *Let $(L, \leq, \vee, \wedge)$ be a distributive lattice as defined in Example 2.4.2. Moreover, let $\xi : L \to \mathbb{R}^+$ be a function on L defined as $\xi(x) := p^{\sum_i x_i}(1-p)^{n-\sum_i x_i}$ for each $x \in L$ and some $0 \leq p \leq 1$. Intuitively, $\xi(x)$ is the probability of x if each component of x is independently set to 1 with probability p and to 0 with probability $1-p$. Now,*

$$\xi(x) \cdot \xi(y) = \xi(x \vee y) \cdot \xi(x \wedge y),$$

which follows from the observation that for all $x, y \in L$

$$\sum_i (x_i + y_i) = \sum_i (x_i \vee y_i) + (x_i \wedge y_i).$$

Thus, ξ is log-supermodular.

A function $f : L \to \mathbb{R}^+$ is *increasing* if for all $x \leq y$, $f(x) \leq f(y)$; f is *decreasing* if for all $x \leq y$, $f(x) \geq f(y)$. The following theorem is due to Fortuin, Kasteleyn, and Ginibre [FKG71] and is known as the FKG inequality.

Theorem 2.4.15 (FKG inequality). *Let $(L, \leq, \vee, \wedge)$ be a finite distributive lattice, and let $\xi : L \to \mathbb{R}^+$ be a log-supermodular function. Then, for any two increasing functions $f, g : L \to \mathbb{R}^+$, we have*

$$\left(\sum_{x \in L} f(x)\xi(x)\right) \cdot \left(\sum_{x \in L} g(x)\xi(x)\right) \leq \left(\sum_{x \in L} f(x)g(x)\xi(x)\right) \cdot \left(\sum_{x \in L} \xi(x)\right).$$

The inequality holds also if f and g are both decreasing. If f is increasing and g is decreasing (or vice versa), the opposite inequality holds.

Applying the FKG Inequality to Probabilities

Let Ω be a sample space, and let $\mathbf{P} : \Omega \to \mathbb{R}^+$ be a probability distribution on Ω. Assume that $(\Omega, \leq, \vee, \wedge)$ is a distributive lattice and that $\mathbf{P}$ is log-supermodular.

Definition 2.4.10 (monotone increasing/decreasing events). *An event A is* monotone increasing *if $\omega \in A$ and $\omega' \geq \omega$ imply $\omega' \in A$. A is* monotone decreasing *if $\omega \in A$ and $\omega' \leq \omega$ imply that $\omega' \in A$.*

Let X_A and X_B be the indicator variables of A and B, respectively. Observe that X_A is increasing or decreasing if A is monotone increasing or decreasing, respectively; the same holds for B and X_B. If both A and B are monotone increasing events then by applying the FKG inequality to X_A and X_B we obtain

$$\begin{aligned}\mathbf{P}[A] \cdot \mathbf{P}[B] &= \left(\sum_{\omega\in\Omega} X_A(\omega)\mathbf{P}[\omega]\right) \cdot \left(\sum_{\omega\in\Omega} X_B(\omega)\mathbf{P}[\omega]\right) \\ &\leq \left(\sum_{\omega\in\Omega} X_A(\omega)X_B(\omega)\mathbf{P}[\omega]\right) \cdot \left(\sum_{\omega\in\Omega} \mathbf{P}[\omega]\right) = \mathbf{P}[A \cap B]\,.\end{aligned}$$

We summarize the result in the following theorem.

Theorem 2.4.16. *Let Ω be a sample space with probability distribution $\mathbf{P}$ such that $(\Omega, \leq, \vee, \wedge)$ constitutes a distributive lattice and $\mathbf{P}$ is log-supermodular. Let $A, B \subseteq \Omega$ be two events. A and B are positively correlated if both A and B are monotone increasing, or both are monotone decreasing. A and B are negatively correlated if A is monotone increasing and B is monotone decreasing, or vice versa.*

2.5 Random Graph Models

We define two models of random graphs that are due to Erdős and Rényi [ER59]: the $G(n,p)$ model and the $G(n,m)$ model. Let $G(n)$ be the set of all undirected graphs (without self-loops) on n vertices and define $M := \binom{n}{2}$; M is the number of edges in a complete graph on n vertices. $G(n)$ has precisely 2^M elements. The sample spaces underlying $G(n,p)$ and $G(n,m)$ are both subsets of $G(n)$. However, these models differ in the way we define their probability distributions on $G(n)$.

We extend these two models also to bipartite graphs. Let $G(n,n)$ denote the set of all bipartite graphs with n nodes on each side. For bipartite graphs we define $M := n^2$. We refer to the respective bipartite random graph models by $G(n,n,p)$ and $G(n,n,m)$.

$G(n,p)$ *Model*

In the $G(n,p)$ model each of the M potential edges is present with probability p, independently of other edges. That is, the sample space of $G(n,p)$ is the entire set $G(n)$, and the probability

of a graph $G \in G(n)$ with m edges is $p^m(1-p)^{M-m}$. If $p = \frac{1}{2}$ all graphs in $G(n)$ are equiprobable. We define $G(n,n,p)$ analogously.

$G(n,m)$ ***Model***

The sample space of $G(n,m)$ consists of all $\binom{M}{m}$ graphs from $G(n)$ that have exactly m edges and each graph in $G(n,m)$ is equiprobable. Put differently, a graph G from $G(n,m)$ is chosen independently uniformly at random with probability $1/\binom{M}{m}$. We define the $G(n,n,m)$ model analogously.

A *graph property* P is a subset of $G(n)$ containing all graphs having property P. For example, $P := \{G \in G(n) : G \text{ is Hamiltonian}\}$ is the graph property of being Hamiltonian. A graph property P is *monotone increasing* if $G_1 \in P$, $G_2 \in G(n)$ and $G_1 \subset G_2$ imply $G_2 \in P$. For example, the graph property of being Hamiltonian is monotone increasing.

Let G_p be a graph chosen from $G(n,p)$ with $p = m/M$. Then the expected number of edges in G_p is $Mp = m$. The following theorem, due to Angluin and Valiant [AV79], relates the occurrence of a graph property P in G_p to its occurrence in a graph G_m from $G(n,m)$ with $p = m/M$.

Theorem 2.5.1. *Let P be some monotone graph property. Moreover, let $G_p \in G(n,p)$ with $p = m/M$ and $G_m \in G(n,m)$. Then*

$$\mathbf{P}[G_m \in P] = O(n\mathbf{P}[G_p \in P]).$$

3. A HEURISTIC FOR DIJKSTRA'S ALGORITHM WITH MANY TARGETS

Abstract

We consider the *single-source many-targets shortest path* (SSMTSP) problem in directed graphs with non-negative edge costs. We are given a source node s and a target set T, and the objective is to compute a shortest path from s to a node in T. Dijkstra's algorithm can be used to solve the SSMTSP problem. Our interest in the SSMTSP problem originates from its use in weighted bipartite matching algorithms. A weighted bipartite matching in a graph with n nodes on each side reduces to n SSMTSP problems, where the number of nodes in the target set varies between n and 1.

In this chapter we describe a simple heuristic that is easy to implement and significantly reduces the number of queue operations performed by Dijkstra's algorithm. In our experiments on random graphs a speed-up by a factor of up to 12 was observed for the weighted matching algorithm. We also present a partial analysis that gives some theoretical support to our experimental findings.

Publication Notes. A preliminary version of this chapter was first published together with Kurt Mehlhorn in the Conference Proceedings of the Nineth Annual European Symposium on Algorithms (ESA 2001) [MS01]. A journal version appeared in Algorithmica in 2003 [BMST03] and is joint work with Holger Bast, Kurt Mehlhorn, and Hisao Tamaki. Holger Bast and Hisao Tamaki helped us to resolve an error in the preliminary version.

3.1 Introduction

In the *single-source many-targets shortest path* (SSMTSP) problem we are given a directed graph $G = (V, E)$, a non-negative cost function $c : E \to \mathbb{R}^+$ on the edges of G, and a source node s. Moreover, every node in V is designated as either *free* or *non-free*. We are interested in finding a shortest path from s to a free node.

The SSMTSP problem is solved by Dijkstra's algorithm. Dijkstra's algorithm maintains a tentative distance for each node and a partition of the nodes into *settled* and *unsettled*. At the beginning all nodes are unsettled. The algorithm operates in phases. In each phase, an unsettled node with smallest tentative distance is declared settled and its outgoing edges are relaxed in order to improve tentative distances of other unsettled nodes. The unsettled nodes are kept in a priority queue. The algorithm can be stopped once the first free node becomes settled.

We describe a heuristic improvement. The improvement maintains an upper bound on the tentative distances of free nodes and only performs queue operations with values smaller than the bound. All other queue operations are suppressed. The heuristic significantly reduces the number of queue operations and therefore the running time of the algorithm.

We first used this heuristic in a jump-start routine to compute an initial matching for the general weighted matching algorithm [Sch00, MS00, MS02]. The jump-start routine computes a maximum weight matching if it is applied to bipartite graphs. When we compared the running time of the jump-start routine with LEDA's bipartite matching algorithms [MN99, Sec. 7.8], we found that the jump-start routine is consistently faster. We traced the superiority to the heuristic described in this chapter.

The experiments that we present in this chapter were performed with the *Tool Set for Computational Experiments*; see ⟨*http://exptools.sourceforge.net*⟩. The tool provides a simple way to set up, run, and analyze experiments. Moreover, it facilitates the documentation of the environment in which the experiments were performed and also enables to reproduce the experiments at a later time.

This chapter is organized as follows. In Section 3.2 we discuss Dijkstra's algorithm for many targets and describe our heuristic. In Section 3.3 we give an analysis of the heuristic for random graphs and report about experiments on random graphs. In Section 3.4 we discuss the application to weighted bipartite matching algorithms and present our experimental findings for the matching problem.

3.2 Dijkstra's Algorithm with Many Targets

It is useful to introduce some more notation. For a node $v \in V$, let $d(v)$ be the shortest path distance from s to v, and let $d_0 := \min\{d(v) : v \text{ is free}\}$. If there is no free node reachable from s, $d_0 = +\infty$. Our goal is to compute (i) a node v_0 with $d(v_0) = d_0$ (or an indication that there is no such node), (ii) the subset V' of nodes with $d(v) < d_0$, more precisely, $v \in V'$ if $d(v) < d_0$ and $d(v) \geq d_0$ if $v \notin V'$, and (iii) the value $d(v)$ for every node $v \in \{v_0\} \cup V'$, i.e., a partial function $\tilde{d}$ with $\tilde{d}(v) = d(v)$ for any $v \in \{v_0\} \cup V'$. (Observe that nodes v with $d(v) = d_0$ may or may not be in V'.) We refer to the problem just described as the single-source many-targets shortest path (SSMTSP) problem. It is easily solved by an adapted version of Dijkstra's algorithm as shown in Figure 3.1.

We maintain a priority queue PQ for the nodes of G. The queue is empty initially. For each node $u \in V$ we compute a tentative distance $dist(u)$ of a shortest path from s to u. Initially, we set $dist(s)$ to zero and insert the item $\langle s, 0\rangle$ into the priority queue. For each $u \in V, u \neq s$, we set $dist(u)$ to $+\infty$ (no path from s to u has been encountered yet). In the main loop we delete a node u with minimal $dist$-value from the priority queue. If u is free, we stop: $v_0 = u$ and V' is the set of nodes removed in preceding iterations. Otherwise, we relax all edges out of u. Consider an edge $e = (u, v)$ and let $\delta = dist(u) + c(e)$. We check whether δ is smaller than the current tentative distance of v. If so, we distinguish two cases. (i) If e is

```
DIJKSTRA'S ALGORITHM (ADAPTED VERSION):
dist(s) = 0 and dist(u) = +∞ for all u ∈ V, u ≠ s
PQ.insert(s, 0)                                   (insert ⟨s, 0⟩ into PQ)
while not PQ.empty() do
   u = PQ.del_min()                               (remove node u from PQ with minimal priority)
   if u is free then STOP fi
   RELAX ALL OUTGOING EDGES OF u
od

RELAX ALL OUTGOING EDGES OF u:
for all e = (u, v) ∈ E do
   δ = dist(u) + c(e)
   if δ < dist(v) then
      if dist(v) = +∞                             (v is not contained in PQ)
         then PQ.insert(v, δ)                     (insert ⟨v, δ⟩ into PQ)
         else PQ.decrease_p(v, δ)                 (decrease priority of v in PQ to δ)
      fi
      dist(v) = δ
   fi
od
```

Figure 3.1: Dijkstra's algorithm adapted for many targets. When the first free node is removed from the queue, the algorithm is stopped: v_0 is the node removed last and V' consists of all non-free nodes removed from the queue.

the first edge into v that is relaxed (this is the case iff $dist(v)$ equals $+\infty$) we insert an item $\langle v, \delta \rangle$ into PQ. (ii) Otherwise, we decrease the priority of v in PQ to δ. If a queue operation is performed, we also update $dist(v)$.

Observe that the single-source many-targets shortest path problem can alternatively be solved by a single-source single-target shortest path computation from s to a target node t, where all target nodes in T are contracted to a single target node t. The adapted version essentially does the same but without performing these contractions explicitly.

We next describe a heuristic improvement of the algorithm above. Let B be the smallest $dist$-value of a free node encountered by the algorithm; $B := +\infty$ initially. We claim that queue operations $PQ.op(\cdot, \delta)$ with $\delta \geq B$ may be skipped without affecting correctness. This is clear, since the algorithm stops when the first free node is removed from the queue and since the $dist$-value of this node is certainly no larger than B. Thus all $dist$-values less than $d(v_0)$ will be computed correctly. The modified algorithm may output a different node v_0 and a different set V'. However, if all distances are pairwise distinct the same node v_0 and the same set V' as in the basic algorithm are computed. The pruning heuristic can conceivably save on queue operations, since fewer insert and decrease priority operations may be performed. Figure 3.2 shows the algorithm with the heuristic added.

Note that the changes which are necessary to incorporate our heuristic into the adapted version of Dijkstra's algorithm are trivial and computationally negligible. Moreover, if the underlying priority queue is stable, i.e., items with the same priority are removed from the

```
DIJKSTRA'S ALGORITHM WITH PRUNING HEURISTIC:
dist(s) = 0 and dist(u) = +∞ for all u ∈ V, u ≠ s
B = +∞                                          (initialize upper bound to +∞)
PQ.insert(s, 0)                                 (insert ⟨s, 0⟩ into PQ)
while not PQ.empty() do
   u = PQ.del_min()                             (remove node u from PQ with minimal priority)
   if u is free then STOP fi
   RELAX ALL OUTGOING EDGES OF u
od

RELAX ALL OUTGOING EDGES OF u:
for all e = (u, v) ∈ E do
   δ = dist(u) + c(e)
   if δ ≥ B then continue fi                    (prune edge if bound is exceeded)
   if v is free then B = min(δ, B) fi           (try to improve bound)
   if δ < dist(v) then
      if dist(v) = +∞                           (v is not contained in PQ)
         then PQ.insert(v, δ)                   (insert ⟨v, δ⟩ into PQ)
         else PQ.decrease_p(v, δ)               (decrease priority of v in PQ to δ)
      fi
      dist(v) = δ
   fi
od
```

Figure 3.2: Dijkstra's algorithm for many targets with a pruning heuristic. An upper bound B for $d(v_0)$ is maintained and queue operations $PQ.op(\cdot, \delta)$ with $\delta \geq B$ are not performed.

queue in the order of their insertions, it is clear that the heuristic will never use more queue operations than the adapted Dijkstra algorithm.

Subsequently, we refer to the adapted version of Dijkstra's algorithm as *standard scheme*, while we refer to our heuristic as *refined scheme*.

3.3 Analysis

We perform a partial analysis of the standard and the refined scheme on random graphs with random real-valued edge costs. We use n for the number of nodes, m for the expected number of edges, and f for the expected number of free nodes. Throughout the analysis we make the following probability assumptions:

1. The underlying graph G is a directed random graph chosen from the $G(n, p)$ model with $p := m/n^2$, i.e., each of the n^2 potential edges is picked independently at random with probability p.
2. A node is free with probability $q := f/n$, independently of the other nodes.
3. The edge costs are random reals chosen independently and uniformly from $[0, 1]$.

Let c be the expected outdegree of a node, i.e., $c := pn = m/n$. We are mainly interested in the case where $p = c/n$ for a small constant c, say $2 \leq c \leq 10$, and q a constant, i.e., the expected number of free nodes is a fixed fraction of the nodes. Alternatively, we could choose our random graph from the $G(n, m)$ model and let free nodes form a random subset of f nodes. The results would be similar.

Number of Deletions from the Queue

We first analyze the number of nodes removed from the queue. If our graph were infinite and all nodes were reachable from s, the expected number would be $1/q$, namely the expected number of trials until the first head occurs in a sequence of coin tosses with success probability q. However, our graph is finite (not really a serious difference if n is large) and only a subset of the nodes is reachable from s. Observe that the probability that s has no outgoing edge is $(1-p)^n \approx e^{-c}$. This probability is not negligible. We proceed in two steps. We first analyze the number of nodes removed from the queue given the number R of nodes reachable from s, and in a second step review results about the number R of reachable nodes.

Lemma 3.3.1. *Let R be the number of nodes reachable from s in G and let T be the number of iterations, i.e., in iteration T the first free node is removed from the queue or there is no free node reachable from s and $T = R$. Then, $\mathbf{P}[T = t \mid R = r] = q(1-q)^{t-1}$ for $1 \leq t < r$, and $\mathbf{P}[T = t \mid R = r] = (1-q)^{t-1}$ for $t = r$. Moreover, for the expected number of iterations we have $\mathbf{E}[T \mid R = r] = 1/q - (1-q)^r/q$.*

Proof. Since each node is free with probability $q = f/n$ and since the property of being free is independent from the order in which nodes are removed from the queue, we have $\mathbf{P}[T = t \mid R = r] = q(1-q)^{t-1}$ and $\mathbf{P}[T \geq t \mid R = r] = (1-q)^{t-1}$ for $1 \leq t < r$. If $t = r$, $\mathbf{P}[T = t \mid R = r] = (1-q)^{t-1} = \mathbf{P}[T \geq t \mid R = r]$.

The expected number of iterations is

$$\begin{aligned} \mathbf{E}[T \mid R = r] &= \sum_{t \geq 1} \mathbf{P}[T \geq t \mid R = r] = \sum_{t=1}^{r-1} (1-q)^{t-1} + (1-q)^{r-1} \\ &= \frac{1-(1-q)^r}{1-(1-q)} = \frac{1}{q} - \frac{(1-q)^r}{q}. \end{aligned}$$

□

The expected number of edges relaxed is $c\mathbf{E}[(T-1) \mid R = r]$ since $T-1$ non-free nodes are removed from the queue and since the expected outdegree of every node is $c = m/n$. We conclude that the number of edges relaxed is about $((1/q) - 1)(m/n)$.

Now, how many nodes are reachable from s? This quantity is analyzed in the book by Alon and Spencer [ASE92, Sec. 10.5]. Let $\alpha > 0$ be such that $\alpha = 1 - \exp(-c\alpha)$, and let R be the number of nodes reachable from s. Then R is bounded by a constant with probability

c	2	5	8	8
α	0.7968	0.9930	0.9997	0.9997
MS	15	2	1	1
ML	714	981	996	1995
R	796.5	993	999.7	1999.3
F	7958	9931	9997	9995

Table 3.1: For all experiments (except the one in the last column) we used random graphs with $n = 1000$ nodes and $m = cn$ edges. For the last column we chose $n = 2000$ in order to illustrate that the dependency on n is weak. The following quantities are shown; for each value of c we performed 10^4 trials.
α: the solution of the equation $\alpha = 1 - \exp(-c\alpha)$.
MS: the maximal number of nodes reachable from s when few nodes are reachable.
ML: the minimal number of nodes reachable from s when many nodes are reachable.
R: the average number of nodes reachable from s when many nodes are reachable.
F: the number of times many nodes are reachable from s.

about $1-\alpha$ and is approximately αn with probability about α. More precisely, for every $\epsilon > 0$ and $\delta > 0$, there is a t_0 such that for all sufficiently large n, we have

$$1 - \alpha - \epsilon \;\le\; \mathbf{P}[R \le t_0] \;\le\; 1 - \alpha + \epsilon$$

and

$$\alpha - \epsilon \;\le\; \mathbf{P}[(1-\delta)\alpha n < R < (1+\delta)\alpha n] \;\le\; \alpha + \epsilon.$$

Table 3.1 indicates that small values of ϵ and δ work even for moderate n. For $c = 2$, we have $\alpha \approx 0.7968$. We generated 10000 graphs with $n = 1000$ nodes and 2000 edges and determined the number of nodes reachable from a given source node s. This number was either smaller than 15 or larger than 714. The latter case occurred in $7958 \approx \alpha \cdot 10000$ trials. Moreover, the average number of nodes reachable from s in the latter case was $796.5 \approx \alpha \cdot 1000 = \alpha n$.

We are only interested in the case that many nodes are reachable from s. We fix δ rather arbitrarily at 0.01 and restrict attention to the set of graphs with more than $(1-\delta)\alpha n$ nodes reachable from s. In this situation, the probability that all reachable nodes are removed from the queue is

$$(1-q)^{\alpha n} \le \exp(-\alpha n q) = \exp(-\alpha f).$$

This is less than $1/n^2$ if $c \ge 2$ and $f \ge 4 \ln n$; observe that $c \ge 2$ implies $\alpha > 1/2$. We thus require our parameters to satisfy $c \ge 2$ and $f \ge 4 \ln n$ and assume that more than $(1-\delta)\alpha n$ nodes are reachable from s. We use the phrase "R is large" to refer to this assumption.

Number of Insertions into the Queue

We next analyze the number of insertions into the queue, first for the standard scheme.

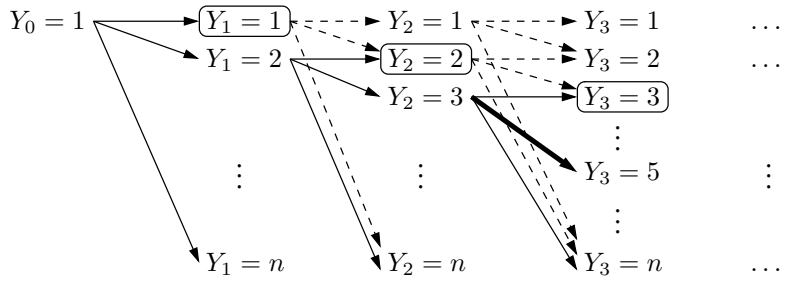

Figure 3.3: The probability of the bold edge is the probability of having two successes in $n-3$ Bernoulli trials with success probability p. We can view the process with and without the dashed edges. The process with the dashed edges corresponds to the recursive definition of the variables Y_t given in (3.1), and the process without the dashed edges corresponds to graph exploration. In the latter case the process dies as soon as $Y_t = t$ (i.e., a box is hit). We are interested in the latter process. The transition probabilities in the latter process differ in a non-trivial way from the transition probabilities in the original process.

Lemma 3.3.2. *Let IS be the number of insertions into the queue in the standard scheme. Then $\mathbf{E}[IS \mid T = t$ and R is large$] \geq n - (n-1)(1-p)^{t-1}$ for $t < (1-\delta)\alpha n$ and*

$$\mathbf{E}[IS \mid R \text{ is large}] \geq \frac{c(1-q)}{q+(1-q)c/n} - \frac{(1-q)c/n}{q+(1-q)c/n} + 1 - o(1) \approx \frac{c}{q} - c + 1 - o(1).$$

Proof. We need to review some more material from [ASE92, Sec. 10.5]. Let $\mathbf{Bin}[k,p]$ denote the binomial distribution with k trials and success probability p; see Section 2.4.5. Consider the following sequence of random variables:

$$Y_0 = 1 \quad \text{and} \quad Y_t = Y_{t-1} + \mathbf{Bin}[n - Y_{t-1}, p] \quad \text{for } 1 \leq t \leq n, \tag{3.1}$$

and let R denote the least t such that $Y_t = t$. Then R is the number of reachable nodes as a simple induction shows: Observe that precisely s is reachable before the first removal and that at the time the tth node is removed from the queue, each of the $n - Y_{t-1}$ remaining (i.e., non-reached) nodes is reached with probability p. Figure 3.3 illustrates the process.

An inductive argument (see [ASE92, Sec. 10.5]) shows $Y_t = 1 + \mathbf{Bin}[n-1, 1-(1-p)^t]$ and hence $\mathbf{E}[Y_t] = n-(n-1)(1-p)^t$ for $0 \leq t \leq n$. We cannot directly use this result as we are interested in the process without the dashed edges. Let $E_t = (Y_0 \geq 1 \cap \ldots \cap Y_{t-1} \geq t)$ be the event that there are at least t reachable nodes. Then $E_{(1-\delta)\alpha n}$ is tantamount to "R is large". Also, $\mathbf{E}[Y_t \mid R \text{ is large}]$ is the expected value of Y_t for the process without the dashed edges. The following claim that the event "R is large" (i.e., the exclusion of the dashed edges) biases Y_t towards larger values is intuitively plausible, but not at all trivial to prove.

Proposition 3.3.1. *For $t \leq (1-\delta)\alpha n$ we have*

$$\mathbf{E}[Y_t \mid R \text{ is large}] \geq \mathbf{E}[Y_t] = n - (n-1)(1-p)^t.$$

Proof. Let $N = (1-\delta)\alpha n$. Recall that $Y_0 = 1$ and $Y_t = Y_{t-1} + \mathbf{Bin}[n - Y_{t-1}, p]$ for $1 \le t \le n$. It is convenient to view the underlying probability space Ω as $\{0,1\}^{n^2}$, where entries are independently 1 with probability p. An elementary event is $\omega = (\omega_1, \ldots, \omega_n)$, where $\omega_i \in \{0,1\}^n$, and $Y_t(\omega) - Y_{t-1}(\omega)$ is the number of ones among the first $n - Y_{t-1}(\omega)$ entries of ω_t.

Let E be the event $(Y_0 \ge 1 \cap \ldots \cap Y_{N-1} \ge N)$ (i.e., the event "R is large"), and let A be the event $Y_t \ge a$ for some arbitrary t and a. We will prove that A and E are positively correlated using the technique described in Section 2.4.6. The reader may verify that Ω forms a distributive lattice and that $\mathbf{P}$ is log-supermodular; see also Examples 2.4.2 and 2.4.3. Both events are monotone increasing, i.e., if ω is componentwise less than or equal to ω' then $\omega \in E$ implies $\omega' \in E$ and $\omega \in A$ implies $\omega' \in A$. Thus, by Theorem 2.4.16,

$$\mathbf{P}[A \cap E] \ge \mathbf{P}[A] \cdot \mathbf{P}[E] \quad \text{or, equivalently,} \quad \mathbf{P}[Y_t \ge a \mid R \text{ is large}] \ge \mathbf{P}[Y_t \ge a]\,.$$

Thus

$$\mathbf{E}[Y_t \mid R \text{ is large}] \ge \mathbf{E}[Y_t].$$

□

We can now derive our bound on the number of insertions. Let T be the number of removals from the queue. Then

$$\begin{aligned}\mathbf{E}[IS \mid R \text{ is large}] = {} & \mathbf{E}[IS \mid T \le (1-\delta)\alpha n \text{ and } R \text{ is large}]\mathbf{P}[T \le (1-\delta)\alpha n \mid R \text{ is large}] \\ & + \mathbf{E}[IS \mid T > (1-\delta)\alpha n \text{ and } R \text{ is large}]\mathbf{P}[T > (1-\delta)\alpha n \mid R \text{ is large}]\,.\end{aligned}$$

If R is large, the probability that we have more than $(1-\delta)\alpha n$ removals from the queue is $O(1/n^2)$. Thus

$$\begin{aligned}\mathbf{E}[IS \mid R \text{ is large}] &\ge \mathbf{E}[IS \mid T \le (1-\delta)\alpha n \text{ and } R \text{ is large}](1 - O(n^{-2})) \\ &\ge \mathbf{E}[IS \mid T \le (1-\delta)\alpha n \text{ and } R \text{ is large}] - o(1).\end{aligned}$$

If R is large and $T \le (1-\delta)\alpha n$, the procedure stops when the first free node is reached (and not because it runs out of edges). The number of insertions into the queue equals the number of nodes which are reached until the first free node is removed from the queue. Thus, for $t \le (1-\delta)\alpha n$, we obtain (recall that the outgoing edges of the free node removed are not relaxed)

$$\mathbf{E}[IS \mid T = t \text{ and } R \text{ is large}] \ge n - (n-1)(1-p)^{t-1}.$$

Thus

$$\mathbf{E}[IS \mid R \text{ is large}] \ge \sum_{t=1}^{(1-\delta)\alpha n} \mathbf{E}[IS \mid T = t \text{ and } R \text{ is large}]\mathbf{P}[T = t \mid R \text{ is large}] - o(1)$$

$$
\begin{aligned}
&= \sum_{t\geq 1} \left(n - (n-1)(1-p)^{t-1}\right)(1-q)^{t-1}q - o(1)\\
&= n - q(n-1)\sum_{t\geq 0}(1-q)^t(1-p)^t - o(1)\\
&= n - q(n-1)\frac{1}{1-(1-p)(1-q)} - o(1)\\
&= n - 1 - (n-1)\frac{q}{p+q-pq} + 1 - o(1)\\
&= (n-1)\frac{p-pq}{p+q-pq} + 1 - o(1)\\
&= \frac{c(1-q)}{q+(1-q)c/n} - \frac{(1-q)c/n}{q+(1-q)c/n} + 1 - o(1)\\
&\approx \frac{c}{q} - c + 1 - o(1).
\end{aligned}
$$

□

The final approximation is valid if $c/n \ll q$. The approximation makes sense intuitively: By Lemma 3.3.1, we relax the edges out of $1/q - 1$ nodes and hence relax about c times as many edges. There is hardly any sharing of targets between these edges if n is large (and c is small). We conclude that the number of insertions into the queue is $\frac{c}{q} - c + 1$.

Observe that the standard scheme makes about c/q insertions into but only $1/q$ removals from the queue. This is where the refined scheme saves.

Number of Nodes Inserted but Never Removed.

Lemma 3.3.3. *Let $INRS$ be the number of nodes which are inserted into the queue but never removed in the standard scheme. Then, by the above,*

$$\mathbf{E}[INRS \mid R \text{ is large}] \quad \approx \quad \frac{c}{q} - c + 1 - \frac{1}{q} \approx \frac{c-1}{q}.$$

□

The standard scheme also performs some $decrease\_p$ operations on the nodes inserted but never removed. This number is small since the expected number of incoming edges per node is c, which we assumed to be a small constant. Observe that the expected number of insertions is basically the same as the expected number of edge relaxations.

We turn to the refined scheme. We have three kinds of savings.

- Nodes that are removed from the queue may incur fewer queue operations because they are inserted later or because some distance decreases do not lead to a queue operation.

This saving is small since the number of distance decreases is small (recall that only few incoming edges per node are scanned).

- Nodes that are never removed from the queue in the standard scheme are not inserted in the refined scheme. This saving is significant and we will estimate it below.
- Nodes that are never removed from the queue in the standard scheme are inserted in the refined scheme but fewer decreases of their distance labels lead to a queue operation. This saving is small for the same reason as in the first item.

We concentrate on the set of nodes that are inserted into but never removed from the queue in the standard scheme. How many of these $INRS$ insertions are also performed in the refined scheme? We use $INRR$ to denote their number.

Lemma 3.3.4. *Let $INRR$ denote the number of insertions which are also performed in the refined scheme. Then*

$$\mathbf{E}[INRR \mid R \text{ is large}] \leq \frac{1}{q} \cdot (1 + \ln(c-1)).$$

Proof. We first compute the expectation of $INRR$ conditional on an arbitrary fixing of the edges of the graph and of the nodes removed from the queue by the standard scheme. More precisely, what is fixed in this event is the edges of the graph, the sequence of nodes removed from the queue, their distance labels, and whether they are free or not.

Then what is still random in this conditional probability space? It is the weights of the edges going from a node removed from the queue to a node that is (thus inserted but) not removed from the queue, and it is whether the nodes these edges are going to are free or not. And still random are, of course, the weights of all edges with neither node looked at by the standard scheme, and whether these nodes are free or not.

Let $e_1 = (u_1, v_1), \ldots, e_l = (u_l, v_l)$ be the edges going from a node removed from the queue to a node that is inserted but not removed from the queue, in the order in which they are relaxed, that is, $d(u_i) \leq d(u_{i+1})$, for $i = 1, \ldots, l-1$. Note that the sequence $u_1, \ldots, u_l$ may contain repetitions of the same node, corresponding to edges relaxed from the same node, whereas the $v_1, \ldots, v_l$ are all different.

The key observation is that in the conditional probability space the edge costs $c(e_1), \ldots, c(e_l)$ are still independent, and the distance label $d(u_i) + c(e_i)$ with which v_i is inserted into the queue is uniformly from $[d(u_l), d(u_i) + 1]$. This is because the fixing of the nodes removed from the queue by the standard scheme implies that $d(u_i) + c(e_i) \geq d(u_l)$ but reveals nothing else about the value of $d(u_i) + c(e_i)$.

In the refined scheme e_i leads to an insertion only if $d(u_i) + c(e_i)$ is smaller than $d(u_j) + c(e_j)$ for every free v_j with $j < i$. The probability for this event is at most $1/(k+1)$, where k is the number of free v_j preceding v_i. The probability would be exactly $1/(k+1)$ if the values $d(u_h) + c(e_h)$, $1 \leq h \leq i$, were all contained in the same interval. Since the upper bound of the interval containing $d(u_h) + c(e_h)$ increases with h, the probability is at most $1/(k+1)$.

We thus obtain that, for any event E_l that fixes the edges of the graph and a sequence of l nodes removed by the standard scheme,

$$\begin{aligned}\mathbf{E}[INRR \mid E_l] &\leq \sum_{i=1}^{l}\sum_{k=0}^{i-1}\binom{i-1}{k}q^k(1-q)^{i-1-k}\frac{1}{k+1}\\ &= \sum_{i=1}^{l}\frac{1}{iq}\sum_{k=0}^{i-1}\binom{i}{k+1}q^{k+1}(1-q)^{i-(k+1)}\\ &= \sum_{i=1}^{l}\frac{1}{iq}\sum_{k=1}^{i}\binom{i}{k}q^k(1-q)^{i-k}\\ &= \sum_{i=1}^{l}\frac{1}{iq}\left(1-(1-q)^i\right),\end{aligned}$$

where the first equality follows from $\binom{i-1}{k}\frac{1}{k+1} = \frac{1}{i}\binom{i}{k+1}$. The final formula can also be interpreted intuitively. There are about iq free nodes preceding v_i and hence v_i is inserted with probability about $1/(iq)$.

In order to estimate the final sum we split the sum at a yet to be determined index i_0. For $i < i_0$ we estimate $(1-(1-q)^i) \leq iq$, and for $i \geq i_0$ we use $(1-(1-q)^i) \leq 1$. We obtain

$$\mathbf{E}[INRR \mid E_l] \leq i_0 + \frac{1}{q}\sum_{i=i_0}^{l}\frac{1}{i} \approx i_0 + \frac{1}{q}\ln\left(\frac{l}{i_0}\right).$$

For $i_0 = 1/q$ (which minimizes the final expression[1]) we have

$$\mathbf{E}[INRR \mid E_l] \leq \frac{1}{q}\cdot(1+\ln(lq)).$$

Now of all the parameters that constitute E_l, this upper bound depends solely on l, the number of nodes that are inserted but not removed from the queue by the standard scheme, so that we may conclude

$$\mathbf{E}[INRR \mid INRS = l \text{ and } R \text{ is large}] \leq \frac{1}{q}\cdot(1+\ln(lq)).$$

Since $\ln(lq)$ is a convex function of l (its first derivative is positive and its second derivative is negative), we obtain an upper bound on the expectation of $INRR$ conditioned on R being large if we replace $INRS$ by its expectation; see Jensen's inequality (Theorem 2.4.4). We obtain

$$\mathbf{E}[INRR \mid R \text{ is large}] \leq \frac{1}{q}\cdot(1+\ln(q\mathbf{E}[INRS \mid R \text{ is large}]))$$

[1] Take the derivative with respect to i_0 ...

$$\approx \frac{1}{q} \cdot \left(1 + \ln\left(q \cdot \frac{c-1}{q}\right)\right)$$
$$= \frac{1}{q} \cdot (1 + \ln(c-1)).$$

□

Number of Saved Queue Operations.

We can now finally lower bound the number S of queue operations saved by the refined scheme.

Theorem 3.3.1. *Let S denote the number of queue operations saved by the refined scheme. Then*

$$\mathbf{E}[S \mid R \text{ is large}] \geq \frac{c}{q}\left(1 - \frac{2 + \ln(c-1)}{c}\right).$$

That is, if the refined scheme is used to solve the single-source many-targets shortest path problem on random graphs drawn from the $G(n,p)$ model, with $p = c/n$ and $c = m/n$, then we are guaranteed to save at least the fraction $1 - (2 + \ln(c-1))/c$ of the queue operations performed by the standard scheme.

Proof. By the above the saving is at least $INRS - INRR$. Thus

$$\mathbf{E}[S \mid R \text{ is large}] \geq \frac{c-1}{q} - \frac{1}{q}(1 + \ln(c-1)) = \frac{c}{q}\left(1 - \frac{2 + \ln(c-1)}{c}\right).$$

□

For example, if $c = 8$, we will save at least a fraction of $1 - (2 + \ln 7)/8 \approx 0.51$ of the queue operations. The actual savings are higher, see Table 3.2. Also, there are substantial savings even if the assumption of R being large does not hold (e.g., for $c = 2$ and $q = 0.02$).

It is interesting to observe how our randomness assumptions were used in the argument above. G is a random graph and hence the number of nodes reachable from s is either bounded or very large. The fact that a node is free with fixed probability gives us the distribution of the number of deletions from the queue. In order to estimate the savings resulting from the refined scheme we use that every node has the same chance of being free and that edge weights are random. For this part of the argument we do not need that our graph is random.

3.4 Bipartite Matching Problems

A matching M in a graph G is a subset of the edges such that no two edges of M share an endpoint. In the *weighted bipartite matching problem* we want to compute a matching M of maximum weight in a bipartite graph $G = (A \cup B, E, w)$, where $w : E \to \mathbb{R}$ is a function

c	2	2	2	5	5	5	8	8	8	8
q	0.02	0.06	0.18	0.02	0.06	0.18	0.02	0.06	0.18	0.18
D	49.60	16.40	5.51	49.33	16.72	5.50	50.22	16.79	5.61	5.53
D^*	50.00	16.67	5.56	50.00	16.67	5.56	50.00	16.67	5.56	5.56
IS	90.01	31.40	10.41	195.20	73.71	22.98	281.30	112.90	36.45	36.52
IS^*	90.16	31.35	10.02	197.60	73.57	23.25	282.30	112.30	36.13	36.77
$INRS$	40.41	15.00	4.89	145.80	56.99	17.49	231.00	96.07	30.85	30.99
$INRS^*$	40.16	14.68	4.46	147.60	56.90	17.69	232.30	95.60	30.57	31.22
$INRR$	11.00	4.00	1.00	35.00	12.00	4.00	51.00	18.00	5.00	5.00
$INRR^*$	39.05	14.56	4.34	104.10	37.13	11.99	126.80	45.78	15.03	15.15
DP_s	1.42	0.19	0.02	13.78	1.90	0.19	36.55	5.28	0.56	0.28
DP_r	0.71	0.09	0.01	2.63	0.31	0.03	4.60	0.50	0.05	0.03
Q_s	140.00	46.98	14.94	257.30	91.33	27.67	367.00	133.90	41.62	41.34
Q_r	110.40	36.12	11.52	134.50	45.33	13.97	154.40	50.85	16.00	15.77
S	29.58	10.86	3.42	122.80	46.00	13.69	212.70	83.08	25.62	25.57
S^*	1.12	0.13	0.12	43.47	19.77	5.70	105.50	49.82	15.54	16.07
P	21.12	23.11	22.87	47.74	50.37	49.50	57.94	62.03	61.55	61.85

Table 3.2: For all experiments (except the one in the last column) we used random graphs with $n = 1000$ nodes and $m = cn$ edges. For the last column we chose $n = 2000$ in order to illustrate that the dependency on n is weak. Nodes were free with probability q. The following quantities are shown; for each value of q and c we performed 10^4 trials. Trials where only a small number of nodes were reachable from s were ignored, i.e., about $(1-\alpha)\cdot 10^4$ trials were ignored.
D: the number of deletions from the queue.
$D^* = 1/q(1-(1-q)^{\alpha n})$: the predicted number of deletions from the queue.
IS: the number of insertions into the queue in the standard scheme.
$IS^* = \frac{c(1-q)}{q+(1-q)c/n} - \frac{(1-q)c/n}{q+(1-q)c/n} + 1$: the predicted number of insertions into the queue.
$INRS$: the number of nodes inserted but never removed.
$INRS^* = IS^* - D^*$: the predicted number.
$INRR$: the number of extra nodes inserted by the refined scheme.
$INRR^* = \frac{1}{q}\cdot(1+\ln(qINRS^*))$: the predicted number.
DP_s: the number of decrease priority operations in the standard scheme.
DP_r: the number of decrease priority operations in the refined scheme.
Q_s: the total number of queue operations in the standard scheme.
Q_r: the total number of queue operations in the refined scheme.
$S = Q_s - Q_r$: the number of saved queue operations.
S^*: the lower bound on the number of saved queue operations.
$P = S/Q_s$: the percentage of queue operations saved.

that assigns a real weight to each edge. The weight of a matching M is simply the sum of the weights of the edges in the matching, i.e., $w(M) := \sum_{e\in M} w(e)$. One may either ask for a perfect matching of maximum weight (known as the weighted *perfect* matching problem or the *assignment problem*) or simply for a matching of maximum weight. Both versions of the problem can be reduced to solving n, where $n = \min(|A|, |B|)$, SSMTSP problems. In this section we discuss the reduction for the assignment problem.

A popular algorithm for the assignment problem follows the primal dual paradigm [AMO93, Sec. 12.4], [MN99, Sec. 7.8], [Gal86]. The algorithm constructs a perfect matching and a dual solution simultaneously. A dual solution is simply a function $\pi : V \to \mathbb{R}$ that assigns a real

Unit Weights

n	c	LEDA	MS	c	LEDA	MS	c	LEDA	MS
10000	2	0.60	0.47	5	42.51	10.80	8	93.07	8.21
20000	2	1.32	1.03	5	152.82	39.31	8	336.24	28.20
40000	2	2.94	2.33	5	550.54	138.88	8	1255.05	97.97

Random Weights $[1, \ldots, 1000]$

n	c	LEDA	MS	c	LEDA	MS	c	LEDA	MS
10000	2	0.57	0.50	5	2.33	1.41	8	11.22	4.87
20000	2	1.20	1.05	5	5.25	3.14	8	25.41	10.79
40000	2	2.63	2.31	5	11.09	6.80	8	56.00	23.63

Random Weights $[1000, \ldots, 1005]$

n	c	LEDA	MS	c	LEDA	MS	c	LEDA	MS
10000	2	0.66	0.57	5	11.42	7.02	8	20.13	11.00
20000	2	1.39	1.22	5	36.56	22.69	8	59.36	31.59
40000	2	3.07	2.71	5	112.05	68.29	8	181.85	99.17

Table 3.3: Effect of the pruning heuristic. LEDA stands for LEDA's bipartite matching algorithm (up to version LEDA-4.2) as described in [MN99, Sec. 7.8] and MS stands for a modified implementation with the pruning heuristic. We created random graphs with n nodes on each side and each edge is present with probability $p = c/n$. The running time is stated in CPU-seconds and is an average of 10 trials.

potential to every node. Let $V := A \cup B$. The algorithm maintains a matching M and a potential function π with the property that

(a) $w(e) \leq \pi(a) + \pi(b)$ for every edge $e = (a, b)$,

(b) $w(e) = \pi(a) + \pi(b)$ for every edge $e = (a, b) \in M$, and

(c) $\pi(b) = 0$ for every *free*[2] node $b \in B$.

Initially, $M := \emptyset$, $\pi(a) := \max_{e \in E} w(e)$ for every $a \in A$, and $\pi(b) := 0$ for every $b \in B$. The algorithm stops when M is a perfect matching[3] or when it discovers that there is no perfect matching. The algorithm works in phases. In each phase the size of the matching is increased by one (or it is determined that there is no perfect matching).

A phase consists of the search for an augmenting path of minimum reduced cost. An augmenting path is a path starting at a free node in A, ending at a free node in B, and using alternately edges not in M and in M. The reduced cost of an edge $e = (a, b)$ is defined as

[2] A node is free if no edge in M is incident to it.

[3] It is easy to see that M has maximum weight among all perfect matchings. Observe that if M' is any perfect matching and π is any potential function such that (a) holds then $w(M') \leq \sum_{v \in V} \pi(v)$. If (b) also holds, we have a pair (M', π) with equality and hence the matching has maximum weight (and the node potential has minimal weight among all potentials satisfying (a)).

$\bar{w}(e) := \pi(a) + \pi(b) - w(e)$; observe that edges in M have reduced cost zero and that all edges have non-negative reduced cost. The reduced cost of a path is simply the sum of the reduced costs of the edges contained in it. There is no need to search for augmenting paths from all free nodes in A; it suffices to search for augmenting paths from a single arbitrarily chosen free node $a_0 \in A$.

If no augmenting path starting in a_0 exists, there is no perfect matching in G and the algorithm stops. Otherwise, for every $v \in V$, let $d(v)$ be the minimal reduced cost of an alternating path from a_0 to v. Let $b_0 \in B$ be a free node in B which minimizes $d(b)$ among all free nodes b in B. We update the potential function according to the rules (we use π' to denote the new potential function):

(d) $\pi'(a) = \pi(a) - \max(d(b_0) - d(a), 0)$ for all $a \in A$,

(e) $\pi'(b) = \pi(b) + \max(d(b_0) - d(b), 0)$ for all $b \in B$.

It is easy to see that this change maintains (a), (b), and (c) and that all edges on the least cost alternating path p from a_0 to b_0 become tight[4]. We complete the phase by switching the edges on p: matching edges on p become non-matching and non-matching edges become matching edges. This increases the size of the matching by one. The correctness of the algorithm can be seen as follows. The algorithm maintains properties (a), (b), and (c) and hence the current matching M is optimal in the following sense. Let $A(M)$ be the nodes in A that are matched. Then M is a maximum weight matching among the matchings that match the nodes in $A(M)$ and leave the nodes in $A \setminus A(M)$ unmatched. Indeed if M' is any such matching then $w(M') \leq \sum_{a \in A(M)} \pi(a) + \sum_{b \in B} \pi(b) = w(M)$, where the inequality follows from (a) and (c) and the equality follows from (b) and (c).

A phase is tantamount to a SSMTSP problem: a_0 is the source and the free nodes are the targets. We want to determine a target (i.e., free node) b_0 with minimal distance from a_0 and the distance values of all nodes v with $d(v) < d(b_0)$. For nodes v with $d(v) \geq d(b_0)$ there is no need to know the exact distance. It suffices to know that the distance is at least $d(b_0)$.

Table 3.3 shows the effect of the pruning heuristic for the bipartite matching algorithm. (The improved code is part of LEDA, Version 4.3 or higher.)

3.5 Concluding Remarks

We presented a simple heuristic for the single-source many-targets shortest path problem. The incorporation of the heuristic into an existing implementation of Dijkstra's algorithm is trivial. In our experiments on random graphs, we observed a substantial improvement in running time for the bipartite weighted matching algorithm. Our analysis supports this observation showing that on random input a significant fraction of queue operations performed by Dijkstra's algorithm is saved.

[4] An edge is called tight if its reduced cost is zero.

At this point we would like to remark that the heuristic can also be used in a capacity scaling implementation of the min-cost flow algorithm; see [AMO93, Sec. 10.2]. It would be interesting to investigate the impact of the heuristic on real-world instances.

4. SMOOTHED COMPETITIVE ANALYSIS OF THE MULTI-LEVEL FEEDBACK ALGORITHM

Abstract

We consider an online problem which is known as the *non-clairvoyant scheduling problem*. Jobs are released over time and have to be scheduled on a machine while the processing times of these jobs are not known. The objective is to minimize the *average flow time*, i.e., the average time spent by jobs in the system. An algorithm for this problem, which is successfully used in practice, is the *multi-level feedback algorithm* (MLF). Although MLF performs very well in practice, its competitive ratio is exponential; more specifically, if the processing times are in $[1, 2^K]$ for some $K \geq 0$, its competitive ratio is $\Omega(2^K)$.

In this chapter, we introduce the notion of *smoothed competitive analysis* of online algorithms and apply it to the multi-level feedback algorithm. We use a partial bit randomization model, where the initial processing times are perturbed by changing the k least significant bits under a quite general class of probability distributions. We show that MLF has smoothed competitive ratio $O((2^k/\sigma)^3 + (2^k/\sigma)^2 2^{K-k})$, where σ denotes the standard deviation of the distribution; in particular, we obtain a competitive ratio of $O(2^{K-k})$ if $\sigma = \Theta(2^k)$. We also prove an $\Omega(2^{K-k})$ lower bound for any deterministic algorithm that is run on processing times smoothed according to the partial bit randomization model. For various other smoothing models we establish a higher lower bound of $\Omega(2^K)$. A direct consequence of our analysis is also the first average case analysis of MLF. We show that MLF has constant expected competitive ratio under several distributions, including the uniform distribution.

Publication Notes. The results presented in this chapter are joint work with Luca Becchetti, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Tjark Vredeveld. An extended abstract appeared in the Conference Proceedings of the Forty-Fourth Annual IEEE Symposium on Foundations of Computer Science (FOCS 2003) [BLMS+03a]. A complete version of the paper was published as a MPII research report [BLMS+03b]. The exposition in this chapter differs significantly from these articles; in particular, we clearly identify the necessary properties of the underlying smoothing distribution for our analysis to hold.

4.1 Introduction

One of the most successful online algorithms used in practice is the *multi-level feedback algorithm* (MLF) for processor scheduling in a time sharing multitasking operating system. MLF

is a *non-clairvoyant* scheduling algorithm, i.e., scheduling decisions are taken without knowledge of the time a job needs to be executed. Windows NT [Nut99] and Unix [Tan92] have MLF at the very basis of their scheduling policies. The objective is to provide a fast response to users. A widely used measure for the responsiveness of a system is the *average flow time* of the jobs, i.e., the average time spent by jobs in the system between release and completion. Job preemption is also widely recognized as a key feature to improve the responsiveness of a system.

The basic idea of MLF is to organize jobs into a hierarchy of queues $Q_0, Q_1, \ldots$. If a job has been processed for a total of 2^i time units it is promoted to queue Q_{i+1}, if not completed. At any time, MLF processes the job at the front of the lowest queue. For the single machine case, if the processing times of the jobs are known, there exists a simple optimal online algorithm, called *Shortest Remaining Processing Time* (SRPT), which always processes a job having smallest remaining processing time. Roughly speaking, MLF tries to simulate SRPT by guessing the processing times of the jobs, giving precedence to jobs that are assumed to have small remaining processing time.

Competitive Analysis. Competitive analysis attempts to characterize the quality of an online algorithm by comparing the performance of the algorithm to that of an optimal offline algorithm. The *competitive ratio* [ST85] of an algorithm is defined as the maximum over all input instances of the ratio between the cost of the online algorithm and the cost of an optimal offline algorithm. While MLF performs very well in practice, it behaves poorly if its performance is measured in terms of its competitive ratio. Assuming that processing times are in $[1, 2^K]$, Motwani, Phillips, and Torng [MPT94] proved a lower bound of $\Omega(2^K)$ on the competitive ratio of any deterministic non-clairvoyant scheduling algorithm. MLF is therefore an example of an algorithm, where the traditional notion of competitiveness fails to explain the good performance of an algorithm in practice.

Smoothed Competitive Analysis. The analysis of online algorithms is a natural field for the application of the idea of smoothed analysis. In this chapter, we propose a new kind of analysis for online algorithms, namely *smoothed competitive analysis*. Roughly speaking, in smoothed competitive analysis we measure the quality of an algorithm by its competitive ratio on randomly perturbed adversarial input instances. In this setting, we also define two different types of adversaries: an *oblivious adversary*, which cannot react to the execution of the algorithm, and a stronger *adaptive adversary*, which may make decisions based on the execution of the algorithm.

We apply this new notion of competitiveness to analyze the multi-level feedback algorithm. We smoothen the input by means of a partial bit randomization model; see also Section 2.3. We assume that the adversarial processing times are K-bit integers in $[1, 2^K]$. (For technical reasons we do not allow zero processing times; we therefore let the all-zero bit string represent 2^K.) For each job we perturb its processing time by replacing the k least significant bits by

some random number in $[1, 2^k]$ drawn from a smoothing distribution f. We use σ to denote the standard deviation of f. For k varying from 0 to K we "smoothly" move from worst case to average case analysis. Our analysis holds for a large class of smoothing distributions, to which we refer to as *well-shaped distributions*, including, for example, the uniform and the normal distribution.

In detail, our contributions are the following:

1. We show that MLF has smoothed competitive ratio $O((2^k/\sigma)^3 + (2^k/\sigma)^2 2^{K-k})$. The competitive ratio therefore improves exponentially with k and as the distribution becomes less sharply concentrated around its mean. In particular, we obtain an expected competitive ratio of $O(2^{K-k})$ for smoothing distributions with $\sigma = \Theta(2^k)$, e.g., for the uniform distribution. We remark that our analysis holds for both the oblivious and the adaptive adversary.

2. As a consequence of our analysis we also obtain an average case analysis of MLF. By choosing $k = K$, our result implies that MLF has constant expected competitive ratio for various distributions of processing times with $\sigma = \Theta(2^k)$ and arbitrarily fixed release dates. Very surprisingly, to the best of our knowledge, this is the first average case analysis of MLF.

3. We prove a lower bound of $\Omega(2^{K-k})$ against an adaptive adversary and a slightly weaker bound of $\Omega(2^{K/6-k/2})$, for every $k \leq K/3$, against an oblivious adversary for any deterministic algorithm if the processing times are smoothed according to the partial bit randomization model.

4. Spielman and Teng [ST01] proposed symmetric smoothing models (see also Section 2.3), where each input value is smoothed symmetrically around its initial value. By using the partial bit randomization model we do not smoothen the processing times symmetrically around their initial values. Therefore, a natural question is whether or not symmetric smoothing models are more suitable to analyze MLF. We answer this question in the negative. In fact, we prove that MLF admits a poor competitive ratio of $\Omega(2^K)$ under symmetric smoothing models.

Related Work. A randomized version of the multi-level feedback algorithm (RMLF) was first proposed by Kalyanasundaram and Pruhs [KP97] for a single machine achieving an $O(\log n \log\log n)$ competitive ratio against the adaptive adversary, where n is the number of jobs that are released. Becchetti and Leonardi [BL01] present a version of RMLF having an $O(\log n \log(n/m))$ competitive ratio on m parallel machines and a tight $O(\log n)$ competitive ratio on a single machine against the oblivious adversary, therefore matching for the single machine case the randomized lower bound of Motwani et al. [MPT94].

Recently, Scharbrodt, Schickinger, and Steger [SSS02] performed an analysis of the average competitive ratio of the Shortest Expected Processing Time First heuristic, minimizing the average completion time, where the processing times of the jobs follow a gamma distribution. Our result is stronger in the following aspects: (i) The analysis of Scharbrodt et al. applies when the algorithm knows the distribution of the processing times, while in our analysis we require no knowledge about the distribution of the processing times. (ii) Our result applies to average flow time, a stronger quality measure than average completion time.

Concerning the average case competitiveness of MLF, Michel and Coffman considered in an early work [MC74] only the problem of synthesizing a feedback queue system under Poisson arrivals and a known discrete probability distribution on processing times so that prespecified mean flow time criteria are met.

Organization of this Chapter. In Section 4.2 we introduce smoothed competitive analysis. Then, in Section 4.3, we define the non-clairvoyant scheduling problem, and in Section 4.4 present the smoothing model that we use. In Section 4.5 we describe the multi-level feedback algorithm. In Section 4.6 we introduce some more notation that is used throughout the analysis presented in Section 4.7. Finally, in Section 4.8, we present lower bounds on the smoothed competitive ratio of MLF, and in Section 4.9 we give some concluding remarks.

4.2 Smoothed Competitive Analysis

Competitive analysis [ST85] measures the quality of an online algorithm by comparing its performance to that of an optimal offline algorithm that has full knowledge of the input. The (worst case) competitive ratio c of a deterministic online algorithm ALG for a cost minimization problem is defined as the maximum over all input instances $\check{I} \in \mathcal{I}$ of the ratio between the cost of the algorithm ALG and the cost of an optimal offline algorithm OPT, i.e.,

$$c := \max_{\check{I} \in \mathcal{I}} \frac{\text{ALG}(\check{I})}{\text{OPT}(\check{I})}.$$

Competitive analysis often provides an overly pessimistic estimation of the performance of an algorithm, or fails to distinguish between algorithms that perform differently in practice due to the presence of pathological bad instances that rarely occur.

The analysis of online algorithms is a natural field for the application of the idea of smoothed analysis. We therefore carry the notion of smoothed analysis over to the area of online algorithms. Following definition (2.1) in Section 2.2, we define the *smoothed competitive ratio* $c(\sigma)$ of an online algorithm ALG as

$$c(\sigma) := \max_{\check{I} \in \mathcal{I}} \mathbf{E}_{I \xleftarrow{f} N(\check{I},\sigma)} \left[\frac{\text{ALG}(I)}{\text{OPT}(I)} \right]. \tag{4.1}$$

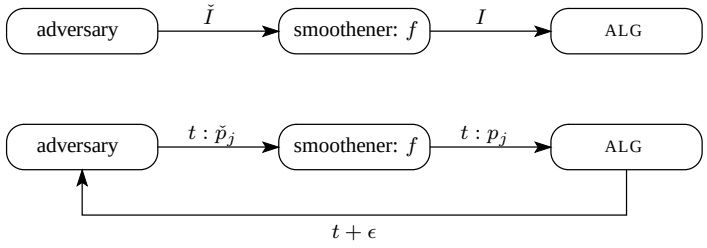

Figure 4.1: Interaction of adversary, smoothing process, and online algorithm. Top: Oblivious adversary. Bottom: Adaptive adversary.

Observe that we might alternatively define the smoothed competitive ratio as the ratio of the expectations, i.e.,

$$c(\sigma) := \max_{\check{I} \in \mathcal{I}} \frac{\mathbf{E}_{I \xleftarrow{f} N(\check{I},\sigma)}[\mathrm{ALG}(I)]}{\mathbf{E}_{I \xleftarrow{f} N(\check{I},\sigma)}[\mathrm{OPT}(I)]}. \tag{4.2}$$

It is difficult to state, which of the two definitions is stronger or more meaningful. Some prefer the definition in (4.1), others prefer (4.2). We believe that definition (4.1) gives a stronger notion of smoothed competitiveness, since (i) the performance is compared instance-wise, and (ii) techniques from probability theory, such as second moment methods, etc., can be used to obtain deviation results.

This kind of analysis results in having the algorithm and the smoothing process together play a game against an adversary, in a way similar to the game played by a randomized online algorithm against its adversary. As for the analysis of randomized online algorithms [BEY98], we define different types of adversaries; see Figure 4.1. The *oblivious adversary* constructs the input instance only based on the knowledge of the algorithm and of the smoothing function f. That is, the oblivious adversary specifies the entire input instance which is then smoothed and presented to the online algorithm. We also define a stronger adversary, the *adaptive adversary*, that reveals the input instance over time, thereby taking decisions made by the online algorithm in the past into account. Said differently, the adaptive adversary constructs the input instance revealed to the algorithm after time t also on the basis of the execution of the algorithm up to time t. Both adversaries are charged with the optimal offline cost on the smoothed instance. Considering the instance space, in the oblivious case $N(\check{I}, \sigma)$ is defined at the beginning, once the adversary has fixed $\check{I}$, while in the adaptive case $N(\check{I}, \sigma)$ is itself a random variable, since it depends on the evolution of the algorithm.

Several other attempts were made in the past to refine the notion of competitiveness so as to characterize the performance of an online algorithm more adequately than by its competitive ratio. One idea was to enhance the capability of the online algorithm by allowing a limited lookahead [Alb97, Alb98]. Another idea was to restrict the power of the adversary. A partial list of these efforts includes the access graph model of Borodin et al. [BIRS95] to restrict the input sequences in online paging problems to specific patterns, and the resource augmentation

model of Kalyanasundaram and Pruhs [KP00] for analyzing online scheduling algorithms.

More related to our proposal of smoothed competitive analysis is the *diffuse adversary model* of Koutsoupias and Papadimitriou [KP94]. In this model, the distribution of the input is chosen by an adversary from a known class of possible distributions. However, smoothed competitive analysis is substantially different from the diffuse adversary model. In the latter model the probability distribution of the input instances is selected by a worst case adversary, while in smoothed competitive analysis the input instance is determined by a worst case adversary and then perturbed according to a specific distribution.

We strongly believe that smoothed competitive analysis is a natural alternative to competitive analysis and that it will help to characterize the actual performance of online algorithms.

4.3 Problem Definition

The adversary releases a set $J = [n]$ of n jobs over time. For each job $j \in J$ the adversary specifies its *release time* r_j and its *initial processing time* $\check{p}_j$. We consider the single machine case. The machine can process at most one job at a time, and a job cannot be processed before its release time. We allow *preemption* of jobs, i.e., a job that is being processed can be interrupted and resumed later on the machine. A scheduling algorithm decides which uncompleted job should be executed at each time. For a generic schedule $\mathcal{S}$, let $C_j^{\mathcal{S}}$ denote the *completion time* of job j. The *flow time* of job j is given by $F_j^{\mathcal{S}} := C_j^{\mathcal{S}} - r_j$, i.e., the total time that j is in the system. The *total flow time* of a schedule $\mathcal{S}$ is defined as $F^{\mathcal{S}} := \sum_{j \in J} F_j^{\mathcal{S}}$ and the *average flow time* is given by $\frac{1}{n} F^{\mathcal{S}}$. A *non-clairvoyant* scheduling algorithm knows about the existence of a job only at the release time of the job, and the processing time of a job is only known when the job is completed. The objective is to find a schedule that minimizes the average flow time.

4.4 Smoothing Model

We smoothen the processing times of the jobs. We remark that we could additionally smoothen the release dates. However, for our analysis to hold it is sufficient to only smoothen the processing times. Furthermore, from a practical point of view, each job is released at a certain time, while processing times are estimates. Therefore, it is more natural to smoothen the processing times and to leave the release dates intact.

We use a partial bit randomization model. We assume that the initial processing times are K-bit integers in $[1, 2^K]$. For each job $j \in J$ we perturb the initial processing times $\check{p}_j$ by replacing the k least significant bits by some random number ε_j that is chosen independently according to a smoothing distribution f from $[1, 2^k]$. More precisely, we define the *smoothed*

processing time p_j of a job $j \in J$ as

$$p_j := 2^k \left\lfloor \frac{\check{p}_j - 1}{2^k} \right\rfloor + \varepsilon_j, \quad \text{where } \varepsilon_j \overset{f}{\leftarrow} [1, 2^k].$$

Note that ε_j is at least 1 and therefore 1 is subtracted from $\check{p}_j$ before applying the modification. For $k = 0$ the smoothed processing times are equal to the initial processing times; for $k = K$ the processing times are chosen entirely at random from $[1, 2^K]$. A similar model is used by Banderier, Beier, and Mehlhorn [BBM03] and by Beier et al. [BCKV04]. However, in [BBM03] and [BCKV04] only the uniform distribution was considered, while our analysis holds for a large class of smoothing distributions. At first glance, it may seem odd to allow distributions other than the uniform distribution. However, the advantage is that for $k = K$ we obtain processing times that are chosen entirely at random according to f.

4.4.1 Feasible Smoothing Distributions

Our analysis holds for any smoothing distribution f that satisfies properties (P1), (P2), and (P3) below. Let ε be a random variable that is chosen according to density function f from $[1, 2^k]$.

(P1) $\mathbf{P}[\varepsilon \geq (1+\gamma)2^{k-1}] \geq \alpha$ for some $0 < \alpha \leq 1$ and $0 < \gamma \leq 2^{k-K-1}$.

(P2) $\sum_{i=0}^{k} \mathbf{P}[\varepsilon \leq 2^i] \leq \beta$ for some $1 \leq \beta \leq k+1$.

(P3) $\mathbf{E}[\varepsilon] \geq \delta \cdot 2^k$ for some $0 < \delta \leq 1$.

We give some intuition; see also Figure 4.2. (P1) states that the upper tail probability of f is at least α. Supposed β is small, (P2) means that f is slowly increasing from 1. (P3) states that the expectation of f is not too close to 1. We remark that our analysis holds for both discrete and continuous distributions. Subsequently, however, we assume that f is discrete. We use μ and σ to denote the expectation and standard deviation of f, respectively.

For distributions satisfying (P1)–(P3) we prove that MLF has smoothed competitive ratio

$$O\left(\frac{K-k+\beta}{\alpha} + \frac{1}{\alpha\gamma} + \frac{1}{\delta^2}\right).$$

Ideally, if α, β, and δ are constants and $\gamma = 2^{k-K-1}$, we obtain a smoothed competitive ratio of $O(2^{K-k})$. It is difficult to give a generic characterization for distributions that satisfy (P1)–(P3) with reasonable values α, γ, β, and δ. We propose the following class of distributions and refer the reader to Section 4.4.2 for further characterizations. We call a distribution f *well-shaped* if the following conditions hold:

1. f is symmetric around μ,
2. $\mu \geq 2^{k-1}$, and

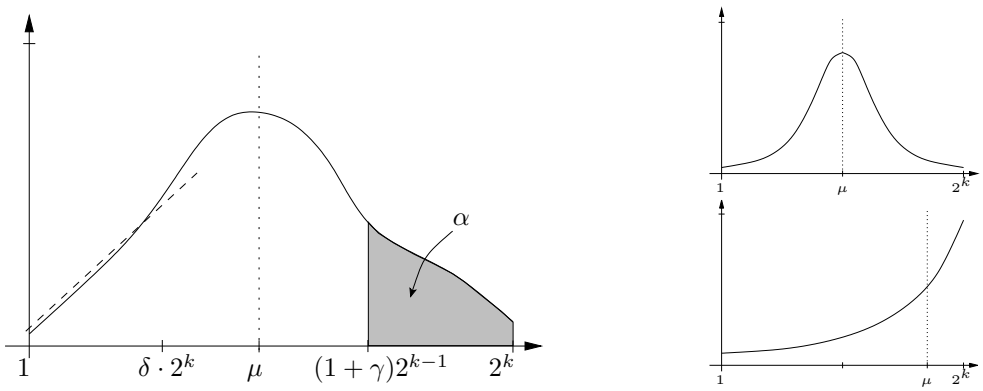

Figure 4.2: Illustration of properties (P1)–(P3).

3. f is non-decreasing in $[1, 2^{k-1}]$.

For example, the uniform, the normal, and the double exponential distribution with $\mu = 2^{k-1} + \frac{1}{2}$ are well-shaped distributions. In Section 4.4.2 we show that well-shaped distributions satisfy (P1)–(P3) with

$$\alpha = \left(\frac{\sigma}{2^k}\right)^2, \quad \gamma = \min\left(\frac{1}{\sqrt{2}}\left(\frac{\sigma}{2^{k-1}}\right),\ 2^{k-K-1}\right), \quad \beta = 2, \quad \text{and} \quad \delta = \frac{1}{2}.$$

Therefore, for a well-shaped distribution we obtain a smoothed competitive ratio of

$$O\left(\left(\frac{2^k}{\sigma}\right)^3 + \left(\frac{2^k}{\sigma}\right)^2 2^{K-k}\right).$$

From the discussion in Section 4.4.2 it will also become apparent that we obtain the same competitive ratio for any distribution with $\mu \geq 2^{k-1}$ and which is non-decreasing in $[1, 2^k]$, e.g., for the exponential distribution.

4.4.2 Characterization of Feasible Smoothing Distributions

In the following we attempt to characterize distributions that satisfy properties (P1)–(P3). The reader may prefer to proceed to subsequent sections first and come back to these characterizations later.

We start with (P1). A trivial lower bound on the tail probability $\mathbf{P}[\varepsilon \geq (1+\gamma)2^{k-1}]$ is given by the following lemma, where we assume a uniform distribution over $[1, (1+\gamma)2^{k-1})$. We remark that although Lemma 4.4.1 is straightforward, it might be indeed tight, e.g., for the uniform distribution.

Lemma 4.4.1. *Let ε be a random variable chosen according to a distribution f over $[1, 2^k]$. Moreover, let M be such that $\mathbf{P}[\varepsilon = x] \leq M$ for each $x \in [1, (1+\gamma)2^{k-1})$. Then, $\mathbf{P}[\varepsilon \geq (1+\gamma)2^{k-1}] \geq 1 - M(1+\gamma)2^{k-1}$.*

Proof.

$$\mathbf{P}[\varepsilon \geq (1+\gamma)2^{k-1}] = 1 - \mathbf{P}[\varepsilon < (1+\gamma)2^{k-1}] \geq 1 - M(1+\gamma)2^{k-1}.$$

□

We also obtain two other lower bounds on the tail probability of f. Both use an "inverse" version of Chebyshev's inequality. We first prove the following lemma; see also [GS01].

Lemma 4.4.2. *Let ε be a random variable and let $h(\varepsilon)$ be a non-negative function such that $h(\varepsilon) \leq M$ for each ε. Then*

$$\mathbf{P}[h(\varepsilon) > \lambda] \geq \frac{\mathbf{E}[h(\varepsilon)] - \lambda}{M - \lambda}.$$

Proof. Let $X_{h(\varepsilon)}$ be 1 if $(h(\varepsilon) > \lambda)$ and 0 otherwise. We have

$$h(\varepsilon) \leq M \cdot X_{h(\varepsilon)} + \lambda \cdot (1 - X_{h(\varepsilon)}),$$

and by linearity of expectation

$$\mathbf{E}[h(\varepsilon)] \leq M \cdot \mathbf{E}[X_{h(\varepsilon)}] + \lambda \cdot (1 - \mathbf{E}[X_{h(\varepsilon)}]).$$

The proof now follows from the fact that $\mathbf{E}[X_{h(\varepsilon)}] = \mathbf{P}[h(\varepsilon) > \lambda]$. □

We are now in a position to obtain our first inverse Chebyshev inequality.

Lemma 4.4.3 (inverse Chebyshev inequality I). *Let ε be a random variable chosen according to a distribution f over $[1, 2^k]$ with mean μ and standard deviation σ. Then, for each $0 < \lambda < 2^k$,*

$$\mathbf{P}[\varepsilon > \lambda] \geq \frac{\sigma^2 + \mu^2 - \lambda^2}{2^{2k} - \lambda^2}.$$

Proof. Define $h(\varepsilon) := \varepsilon^2$. Then $h(\varepsilon) \leq 2^{2k}$ for each ε. The bound now follows from Lemma 4.4.2, where we exploit that $\sigma^2 = \mathbf{E}[\varepsilon^2] - \mu^2$. □

The following lemma shows that for $\gamma := 2^{k-K-1}$ we obtain $\alpha = (\sigma/2^k)^2$, if only the expectation of f is large enough. We remark that the requirement on δ is always satisfied if $\mu \geq \frac{3}{4} \cdot 2^k$.

Lemma 4.4.4. *Let ε be a random variable chosen according to a distribution f over $[1, 2^k]$ with mean $\mu \geq \delta \cdot 2^k$ and standard deviation σ. Define $\gamma := 2^{k-K-1}$. If $\delta \geq \frac{1}{2}(1+\gamma)$ then $\mathbf{P}[\varepsilon \geq (1+\gamma)2^{k-1}] \geq (\sigma/2^k)^2$.*

Proof. The proof follows from Lemma 4.4.3 and since $\mu \geq \delta \cdot 2^k \geq (1+\gamma)2^{k-1}$. □

We derive our second inverse Chebyshev inequality.

Lemma 4.4.5 (inverse Chebyshev inequality II). *Let ε be a random variable chosen according to a distribution f over $[1, 2^k]$ with mean μ and standard deviation σ. Then, for each $0 < \lambda < 2^k - \mu$,*

$$\mathbf{P}[|\varepsilon - \mu| \geq \lambda] \geq \frac{\sigma^2 - \lambda^2}{(2^k - \mu)^2 - \lambda^2}.$$

Proof. Define $h(\varepsilon) := (\varepsilon - \mu)^2$. Then $h(\varepsilon) \leq (2^k - \mu)^2$ for each ε. The proof follows from Lemma 4.4.2. □

The next lemma applies if the underlying distribution f satisfies $\mathbf{P}[\varepsilon \geq \mu + \frac{\sigma}{\sqrt{2}}] \geq \mathbf{P}[\varepsilon \leq \mu - \frac{\sigma}{\sqrt{2}}]$. For example, this condition holds if f is symmetric around μ or if f is non-decreasing over $[1, 2^k]$.

Lemma 4.4.6. *Let ε be a random variable chosen according to a distribution f over $[1, 2^k]$ with mean $\mu \geq \delta \cdot 2^k$ and standard deviation σ, and assume $\mathbf{P}[\varepsilon \geq \mu + \frac{\sigma}{\sqrt{2}}] \geq \mathbf{P}[\varepsilon \leq \mu - \frac{\sigma}{\sqrt{2}}]$. Define*

$$\gamma := \min\left(2\delta - 1 + \frac{1}{\sqrt{2}}\left(\frac{\sigma}{2^{k-1}}\right),\ 2^{k-K-1}\right).$$

Then

$$\mathbf{P}[\varepsilon \geq (1+\gamma)2^{k-1}] \geq \frac{1}{4}\left(\frac{\sigma}{(1-\delta)2^k}\right)^2.$$

Proof. If $\gamma \leq 2\delta - 1 + \frac{1}{\sqrt{2}}\left(\frac{\sigma}{2^{k-1}}\right)$, we obtain

$$\mathbf{P}[\varepsilon \geq (1+\gamma)2^{k-1}] \geq \mathbf{P}\left[\varepsilon \geq \mu + \frac{\sigma}{\sqrt{2}}\right] \geq \frac{1}{2} \cdot \mathbf{P}\left[|\varepsilon - \mu| \geq \frac{\sigma}{\sqrt{2}}\right],$$

where the last inequality holds because $\mathbf{P}[\varepsilon \geq \mu + \frac{\sigma}{\sqrt{2}}] \geq \mathbf{P}[\varepsilon \leq \mu - \frac{\sigma}{\sqrt{2}}]$. Since $2^k - \mu \leq (1-\delta)2^k$, we obtain from Lemma 4.4.5

$$\mathbf{P}[\varepsilon \geq (1+\gamma)2^{k-1}] \geq \frac{1}{2} \cdot \frac{\sigma^2 - \frac{1}{2}\sigma^2}{((1-\delta)2^k)^2} = \frac{1}{4}\left(\frac{\sigma}{(1-\delta)2^k}\right)^2.$$

□

Note that we have to make sure that $\gamma > 0$. Therefore, for $\delta < \frac{1}{2}$ the definition of γ in Lemma 4.4.6 makes sense only if we require $(\sigma/2^{k-1}) > (1-2\delta) \cdot \sqrt{2}$.

Corollary 4.4.1. *If f is a well-shaped distribution, we have $\delta = \frac{1}{2}$ and thus*

$$\alpha = \left(\frac{\sigma}{2^k}\right)^2, \quad \textit{where} \quad \gamma = \min\left(\frac{1}{\sqrt{2}}\left(\frac{\sigma}{2^{k-1}}\right),\ 2^{k-K-1}\right).$$

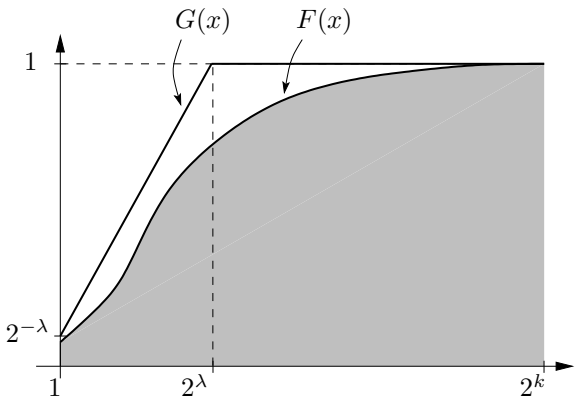

Figure 4.3: $F(x) := \mathbf{P}[\varepsilon \leq x]$, $G(x) := \min(x \cdot \left(\frac{1}{2}\right)^{\lambda}, 1)$, where $\lambda := k - l$.

We come to property (P2). The next lemma characterizes distributions that satisfy (P2).

Lemma 4.4.7. *Let ε be a random variable chosen according to a distribution f over $[1, 2^k]$. Let l be some integer, $0 \leq l \leq k$, such that for each i, $0 \leq i \leq k - l$, $\mathbf{P}[\varepsilon \leq 2^i] \leq 2^i \cdot \left(\frac{1}{2}\right)^{k-l}$. Then $\sum_{i=0}^{k} \mathbf{P}[\varepsilon \leq 2^i] \leq 2 + l$.*

Proof.

$$\sum_{i=0}^{k} \mathbf{P}[\varepsilon \leq 2^i] = \sum_{i=0}^{k-l} \mathbf{P}[\varepsilon \leq 2^i] + \sum_{i=k-l+1}^{k} \mathbf{P}[\varepsilon \leq 2^i] \leq \sum_{i=0}^{k-l} \left(\frac{1}{2}\right)^{k-l-i} + \sum_{i=k-l+1}^{k} 1$$
$$= \sum_{i=0}^{k-l} \left(\frac{1}{2}\right)^{i} + l \leq 2 + l.$$

□

Corollary 4.4.2. *If f is a well-shaped distribution then $\beta = 2$.*

Proof. Since f is non-decreasing in $[1, 2^{k-1}]$, the distribution function $F(x) := \mathbf{P}[\varepsilon \leq x]$ of f is *strictly* increasing once $F(x) > 0$. Moreover, since f is symmetric around μ and $\mu \geq 2^{k-1}$, $F(2^{k-1}) \leq \frac{1}{2}$. Thus, $F(2^i) \leq 2^i \cdot \left(\frac{1}{2}\right)^k$ for each i, $0 \leq i \leq k - 1$. Clearly, $F(2^k) \leq 1$. □

Finally, consider property (P3). We remark that $\mathbf{P}[\varepsilon \geq (1 + \gamma)2^{k-1}] \geq \alpha$ implies $\mathbf{E}[\varepsilon] \geq \frac{1}{2}(1 + \gamma)\alpha 2^k$. However, this bound on δ might be a too weak. In Lemma 4.4.7 we require $\mathbf{P}[\varepsilon \leq x] \leq x \cdot (1/2)^{k-l}$ only for each $x = 2^i$, where $0 \leq i \leq k - l$. If we instead require that this relation holds for every $x \in [1, 2^{k-l}]$, we obtain a characterization for (P3).

Lemma 4.4.8. *Let ε be a random variable chosen according to a distribution f over $[1, 2^k]$. Let l be some integer, $0 \leq l \leq k$, such that for each $x \in [1, 2^{k-l}]$, $\mathbf{P}[\varepsilon \leq x] \leq x \cdot \left(\frac{1}{2}\right)^{k-l}$. Then $\mathbf{E}[\varepsilon] \geq \frac{1}{2^{l+1}} \cdot 2^k$.*

Proof. Consider a uniform random variable U over $[1, 2^{k-l}]$. We have $G(x) := \mathbf{P}[U \leq x] = \min(x \cdot \left(\frac{1}{2}\right)^{k-l}, 1)$; see also Figure 4.3. By definition, $\mathbf{P}[\varepsilon > x] \geq \mathbf{P}[U > x]$ for each $x \in [1, 2^k]$. That is, ε stochastically dominates U, and therefore $\mathbf{E}[\varepsilon] \geq \mathbf{E}[U] = \frac{2^{k-l}+1}{2}$. □

For example, well-shaped distributions satisfy Lemma 4.4.8 with $l = 1$, which yields $\mathbf{E}[\varepsilon] \geq \frac{1}{4} \cdot 2^k$.

4.4.3 Properties of Smoothed Processing Times

We state two crucial properties of smoothed processing times. Define $\phi_j := 2^k \lfloor (\check{p}_j - 1)/2^k \rfloor$. We have $p_j = \phi_j + \varepsilon_j$. Consider a job j with initial processing time $\check{p}_j \in [1, 2^k]$. Then the initial processing time of j is entirely replaced by some random processing time in $[1, 2^k]$ that is chosen according to the probability distribution f.

Fact 4.4.1. *For each job j with $\check{p}_j \in [1, 2^k]$ we have $\phi_j = 0$ and thus $p_j \in [1, 2^k]$. Moreover, $\mathbf{P}[p_j \leq x] = \mathbf{P}[\varepsilon_j \leq x]$ for each $x \in [1, 2^k]$.*

Next, consider a job j with initial processing time $\check{p}_j \in (2^{i-1}, 2^i]$ for some integer $i > k$. Then the smoothed processing time p_j is randomly chosen from a subrange of $(2^{i-1}, 2^i]$ according to the probability distribution f.

Fact 4.4.2. *For each job j with $\check{p}_j \in (2^{i-1}, 2^i]$, for some integer $i > k$, we have $\phi_j \in [2^{i-1}, 2^i - 2^k]$ and thus $p_j \in (2^{i-1}, 2^i]$.*

4.5 Multi-Level Feedback Algorithm

In this section we describe the multi-level feedback algorithm. We say that a job is *alive* or *active* at time t in a generic schedule $\mathcal{S}$, if it has been released but not completed at this time, i.e., $r_j \leq t < C_j^{\mathcal{S}}$. Denote by $x_j^{\mathcal{S}}(t)$ the amount of time that has been spent on processing job j in schedule $\mathcal{S}$ up to time t. We define $y_j^{\mathcal{S}}(t) := p_j - x_j^{\mathcal{S}}(t)$ as the *remaining processing time* of job j in schedule $\mathcal{S}$ at time t. Subsequently, we denote by MLF the schedule produced by the multi-level feedback algorithm.

The set of active jobs is partitioned into a set of priority queues $Q_0, Q_1, \ldots$. Within each queue, the priority is determined by the release dates of the jobs: the job with smallest release time has highest priority. For any two queues Q_h and Q_i, we say that Q_h is lower than Q_i if $h < i$. At any time t, MLF behaves as follows.

1. Job j released at time t enters queue Q_0.

2. Schedule on the machine the alive job that has highest priority in the lowest non-empty queue.
3. For a job j in a queue Q_i at time t, if $x_j^{\text{MLF}}(t) = p_j$, assign $C_j^{\text{MLF}} = t$ and remove the job from the queue.
4. For a job j in a queue Q_i at time t, if $x_j^{\text{MLF}}(t) = 2^i < p_j$, job j is moved from Q_i to Q_{i+1}.

Observe that if the processing times are in $[1, 2^K]$ then at most $K+1$ queues $Q_0, \ldots, Q_K$ are used during the execution of MLF. Moreover, at any time t and for any queue Q_i at most one job in Q_i has been executed. Put differently, if we consider all jobs that are in queue Q_i at time t then at most one of these jobs satisfies $x_j^{\text{MLF}}(t) > 2^{i-1}$, while for all other jobs we have $x_j^{\text{MLF}}(t) = 2^{i-1}$.

Fact 4.5.1. *At any time t and for any queue Q_i at most one job, alive at time t, has been executed in Q_i but has not been promoted to Q_{i+1}.*

Under which circumstances does MLF achieve a good performance guarantee? We offer some intuition. As mentioned in the introduction, Shortest Remaining Processing Time (SRPT) is an optimal algorithm for the single machine case. We can view MLF as trying to simulate SRPT by using estimates for the processing times of the jobs in the system. When a new job arrives its estimated processing time is 1; if a job is enqueued into queue Q_i, for some $i > 0$, MLF assumes that it has processing time 2^i. Put differently, whenever a job has been executed for its estimated processing time and is not completed, MLF doubles its estimate. Observe that if a job j is enqueued into queue Q_i, $i > 0$, MLF assumes that it takes 2^{i-1} additional time to complete j. Therefore, MLF gives precedence to jobs in lower queues.

Consider a job j with processing time $p_j \in (2^{i-1}, 2^i]$. The final estimate of j's processing time in MLF is 2^i. Intuitively, if the actual processing time of j is not too far from its final estimate then the decisions made by MLF with respect to j are not too different from those made by SRPT. However, if the final estimate is far off from the actual processing time then MLF and SRPT may indeed perform very differently. For example, suppose that the actual processing time of j is $2^{i-1} + 1$. When j enters queue Q_i, MLF defers j until all jobs of processing time at most 2^{i-1} are completed. On the other hand, SRPT completes j after one additional time unit.

In fact, it can easily be seen that MLF may perform arbitrarily bad on jobs of the latter kind: We release jobs in two phases. In the first phase, at time $t = 0$, we release $N := 2^{K-1} + 1$ jobs with processing time $2^{K-1} + 1$. Let $\hat{t}$ be the first time when a job, say j^*, has been completed by MLF. At time $\hat{t}$, all remaining $N - 1$ jobs have remaining processing time 1. Now, consider another algorithm ALG that does not schedule j^* and therefore can allocate $2^{K-1} + 1$ time units on the other jobs. ALG will have completed all jobs except j^* by time $\hat{t}$. In the second phase, starting at time $\hat{t}$, we release one after another a long sequence of jobs with processing time 1. If we choose this sequence sufficiently long then the total flow time will be dominated

by the contribution of the second phase. Since during the second phase MLF has at least N jobs in the system while ALG has only two jobs in the system, we obtain a competitive ratio of $\Omega(N) = \Omega(2^K)$.

4.6 Preliminaries

We use MLF and OPT to denote the schedules produced by the multi-level feedback algorithm and by an optimal algorithm, respectively. We use $\mathcal{S}$ to refer to a generic schedule.

We partition jobs into classes: a job $j \in J$ is of *class* i, $0 \leq i \leq K$, if $p_j \in (2^{i-1}, 2^i]$. We use Cl_j to denote the class of a job j. Note that if $\check{p}_j \in (2^{i-1}, 2^i]$ for some integer $i > k$ then Cl_j is *not* a random variable; see Fact 4.4.2. Note that in MLF a job of class i is completed in queue Q_i.

We denote by $\delta^{\mathcal{S}}(t)$ the number of jobs that are active at time t in $\mathcal{S}$. For each job j and any time t we define a binary random variable $X_j^{\mathcal{S}}(t)$ which is 1 if job j is active at time t, and 0 otherwise. We have $\delta^{\mathcal{S}}(t) = \sum_{j \in J} X_j^{\mathcal{S}}(t)$. Moreover, we use $S^{\mathcal{S}}(t)$ to refer to the set of active jobs at time t.

The total flow time $F^{\mathcal{S}}$ of a schedule $\mathcal{S}$ is defined as the sum of the flow times of all jobs. Equivalently, we can express the total flow time as the integral over time of the number of active jobs. We state this as a fact; see also [LR97].

Fact 4.6.1. $F^{\mathcal{S}} = \sum_{j \in J} F_j^{\mathcal{S}} = \int_{t \geq 0} \delta^{\mathcal{S}}(t) dt$.

The following obvious fact states that the sum of the processing times of all jobs is a lower bound on the flow time of any schedule $\mathcal{S}$.

Fact 4.6.2. $F^{\mathcal{S}} \geq \sum_{j \in J} p_j$.

An important notion in our analysis is the notion of *lucky* and *unlucky* jobs. It serves to distinguish between jobs that are good and those which are bad for the performance of MLF.

Definition 4.6.1. *A job j of class i is called* lucky *if $p_j \geq (1+\gamma)2^{i-1}$; otherwise, it is called* unlucky.

For each job j we define a binary random variable X_j^l which is 1 if j is lucky, and 0 otherwise.

Note that for MLF a lucky job of class i is a job that still has a remaining processing time of at least $\gamma 2^{i-1}$ when it enters its queue Q_i of completion. We use $\delta^l(t)$ to denote the number of lucky jobs that are active at time t in MLF. We also define a binary random variable $X_j^l(t)$ that indicates whether or not a job j is lucky and alive at time t in MLF, i.e., $X_j^l(t) := X_j^l \cdot X_j^{\text{MLF}}(t)$. We have $\delta^l(t) = \sum_{j \in J} X_j^l(t)$.

At time t, the job with highest priority among all jobs in queue Q_i (if any) is said to be the *head* of Q_i. A head job of queue Q_i is *ending* if it will be completed in Q_i. We denote by $h(t)$ the total number of ending head jobs at time t.

Let X be a generic random variable. For an input instance I, X_I denotes the value of X for this particular instance I. Note that X_I is uniquely determined by the execution of the algorithm.

4.7 Smoothed Competitive Analysis of MLF

The intuition behind our analysis is as follows. We argued that MLF tries to simulate SRPT by using estimates of the processing times and that the performance of MLF can be related to the one of SRPT if the final estimates are not too far from the actual processing times of the jobs. We make this relation explicit by proving that at any time t the number of lucky jobs is at most the number of ending head jobs plus $6/\gamma$ times the number of active jobs in an optimal schedule. This argument is purely deterministic. We also prove an upper bound of $K - k + \beta$ on the expected number of ending head jobs at any time t.

We write the total flow time as the integral over time of the number of active jobs. At any time t, we distinguish between (i) the number of active jobs in MLF is at most $2/\alpha$ times the number of lucky jobs, and (ii) where this is not the case. We prove that case (i) occurs with high probability so that we can use the deterministic bound to relate MLF to the optimal algorithm. The contribution of case (ii) is compensated by the exponentially small probability of its occurrence.

The high probability argument is presented in Section 4.7.1. Our analysis holds both for the oblivious adversary and for the adaptive adversary. For the sake of clarity, we first concentrate on the oblivious adversary and discuss the differences for the adaptive adversary in Section 4.7.2.

Lemma 4.7.1 provides a deterministic bound on the number of lucky jobs in the schedule of MLF for a specific instance I. The proof is similar to the one given by Becchetti and Leonardi [BL01] and can be found in Appendix 4.A of this chapter.

Lemma 4.7.1. *For any input instance I, at any time t, $\delta_I^l(t) \leq h_I(t) + \frac{6}{\gamma}\delta_I^{\mathrm{OPT}}(t)$.*

Clearly, at any time t the number of ending head jobs is at most $K + 1$. The following lemma gives a better upper bound on the expected number of ending head jobs.

Lemma 4.7.2. *At any time t, $\mathbf{E}[h(t)] \leq K - k + \beta$.*

Proof. Let $h'(t)$ denote the number of ending head jobs in the first $k + 1$ queues. Clearly $\mathbf{E}[h(t)] \leq K - k + \mathbf{E}[h'(t)]$, since the last $K - k$ queues can contribute at most $K - k$ to the expected value of $h(t)$.

We next consider the expected value of $h'(t)$. Let $H(t)$ denote the ordered sequence $(q_0, \ldots, q_k)$ of jobs that are at time t at the head of the first $k + 1$ queues $Q_0, \ldots, Q_k$, respectively. We use $q_i = \times$ to denote that Q_i is empty at time t. Let $H_i(t)$ be a binary random variable indicating whether or not the head job of queue Q_i (if any) is ending, i.e., $H_i(t) = 1$ if $q_i \neq \times$ and q_i is in its final queue, and $H_i(t) = 0$ otherwise. Let $H \in (J \cup \times)^k$ denote any

possible configuration for $H(t)$. Observe that by definition $\mathbf{P}[H_i(t) = 1 \,|\, H(t) = H] = 0$ if $q_i = \times$. Let $q_i \neq \times$. We have

$$\mathbf{P}[H_i(t) = 1 \,|\, H(t) = H] = \mathbf{P}[p_{q_i} \leq 2^i \,|\, H(t) = H]\,.$$

In Appendix 4.B we show that the events $(p_{q_i} \leq 2^i)$ and $(H(t) = H)$ are negatively correlated. Thus, $\mathbf{P}[H_i(t) = 1 \,|\, H(t) = H] \leq \mathbf{P}[p_{q_i} \leq 2^i]$. We obtain

$$\mathbf{E}[h'(t) \,|\, H(t) = H] = \sum_{i=0}^{k} \mathbf{P}[H_i(t) = 1 \,|\, H(t) = H] \leq \sum_{i=0}^{k} \mathbf{P}[p_{q_i} \leq 2^i]\,.$$

If a job q_i is of class larger than k we have $\mathbf{P}[p_{q_i} \leq 2^i] = 0$. Therefore, the sum is maximized if we assume that each q_i is of class at most k. Since the processing times are chosen identically, independently, and (under the above assumption) entirely at random, we have

$$\mathbf{E}[h'(t) \,|\, H(t) = H] \leq \sum_{i=0}^{k} \mathbf{P}[\varepsilon_{q_i} \leq 2^i] \leq \sum_{i=0}^{k} \mathbf{P}[\varepsilon \leq 2^i] \leq \beta,$$

where ε is a random variable chosen according to f from $[1, 2^k]$, and the last inequality follows from property (P2) of our distribution. We conclude

$$\mathbf{E}[h'(t)] = \sum_{H \in (J \cup \times)^k} \mathbf{E}[h'(t) \,|\, H(t) = H]\, \mathbf{P}[H(t) = H] \leq \beta.$$

□

We define a random variable R as the sum of the random parts of all processing times, i.e., $R := \sum_{j \in J} \varepsilon_j$. We need the following bound on the probability that R is at least a constant fraction of its expectation.

Lemma 4.7.3. $\mathbf{P}[R \geq \frac{1}{2}\mathbf{E}[R]] \geq 1 - \mathrm{e}^{-n\delta^2/2}$.

Proof. Observe that $\mathbf{E}[R] = n\mu$, where μ denotes the expectation of f. We use Hoeffding's bound (see also Theorem 2.4.12 (2.4)) and property (P3) to obtain

$$\mathbf{P}[R \leq \tfrac{1}{2}\mathbf{E}[R]] \leq \exp\left(-\frac{\frac{1}{2}\mathbf{E}[R]^2}{n(2^k-1)^2}\right) \leq \exp\left(-\frac{\frac{1}{2}n\mu^2}{2^{2k}}\right) \leq \exp\left(-n\delta^2/2\right).$$

□

We are now in a position to prove Theorem 4.7.1. We introduce the following notation. For an

instance I, we define

$$\mathcal{D}_I := \{t : \delta_I^{\text{MLF}}(t) \leq \tfrac{2}{\alpha}\delta_I^l(t)\} \quad \text{and} \quad \bar{\mathcal{D}}_I := \{t : \delta_I^{\text{MLF}}(t) > \tfrac{2}{\alpha}\delta_I^l(t)\}.$$

Moreover, we define the event $\mathcal{E} := \big(R \geq \frac{1}{2}\mathbf{E}[R]\big)$ and use $\bar{\mathcal{E}}$ to refer to the complement of $\mathcal{E}$.

Theorem 4.7.1. *For any instance $\check{I}$ and any smoothing distribution f that satisfies (P1), (P2), and (P3),*

$$\mathbf{E}_{I \xleftarrow{f} N(\check{I},\sigma)}\left[\frac{F_I^{\text{MLF}}}{F_I^{\text{OPT}}}\right] = O\left(\frac{K-k+\beta}{\alpha} + \frac{1}{\alpha\gamma} + \frac{1}{\delta^2}\right).$$

Proof. Throughout the proof we omit I and that the expectation is taken according to f over $N(\check{I},\sigma)$.

$$\mathbf{E}\left[\frac{F^{\text{MLF}}}{F^{\text{OPT}}}\right] = \mathbf{E}\left[\frac{F^{\text{MLF}}}{F^{\text{OPT}}}\,\middle|\,\mathcal{E}\right]\mathbf{P}[\mathcal{E}] + \mathbf{E}\left[\frac{F^{\text{MLF}}}{F^{\text{OPT}}}\,\middle|\,\bar{\mathcal{E}}\right]\mathbf{P}[\bar{\mathcal{E}}] \;\leq\; \mathbf{E}\left[\frac{F^{\text{MLF}}}{F^{\text{OPT}}}\,\middle|\,\mathcal{E}\right]\mathbf{P}[\mathcal{E}] + n\mathrm{e}^{-n\delta^2/2},$$

where the inequality follows from Lemma 4.7.3 and the fact that n is an upper bound on the competitive ratio of MLF. Define $c := 2/\delta^2$. Since $\mathrm{e}^{-x} < \frac{1}{x}$ for $x > 0$, we have $n\mathrm{e}^{-n\delta^2/2} < c$. We partition the flow time $F^{\text{MLF}} = \int_t \delta^{\text{MLF}}(t)dt$ into the contribution of time instants $t \in \mathcal{D}$ and $t \in \bar{\mathcal{D}}$, i.e., $F^{\text{MLF}} = \int_{t\in\mathcal{D}} \delta^{\text{MLF}}(t)dt + \int_{t\in\bar{\mathcal{D}}} \delta^{\text{MLF}}(t)dt$, and bound these contributions separately.

$$\begin{aligned}
\mathbf{E}\left[\frac{\int_{t\in\mathcal{D}} \delta^{\text{MLF}}(t)dt}{F^{\text{OPT}}}\,\middle|\,\mathcal{E}\right]\mathbf{P}[\mathcal{E}] &\leq \mathbf{E}\left[\frac{\int_{t\in\mathcal{D}} \frac{2}{\alpha}\delta^l(t)dt}{F^{\text{OPT}}}\,\middle|\,\mathcal{E}\right]\mathbf{P}[\mathcal{E}] \\
&\leq \mathbf{E}\left[\frac{\int_{t\in\mathcal{D}} \frac{2}{\alpha}h(t)dt + \int_{t\in\mathcal{D}} \frac{2}{\alpha}\cdot\frac{6}{\gamma}\delta^{\text{OPT}}(t)dt}{F^{\text{OPT}}}\,\middle|\,\mathcal{E}\right]\mathbf{P}[\mathcal{E}] \\
&\leq \mathbf{E}\left[\frac{\int_{t\in\mathcal{D}} \frac{2}{\alpha}h(t)dt}{F^{\text{OPT}}}\,\middle|\,\mathcal{E}\right]\mathbf{P}[\mathcal{E}] + \frac{12}{\alpha\gamma},
\end{aligned}$$

where we use the deterministic bound of Lemma 4.7.1 on $\delta^l(t)$ and the fact that $F^{\text{OPT}} \geq \int_{t\in\mathcal{D}} \delta^{\text{OPT}}(t)dt$. By Fact 4.6.2 and the definition of event $\mathcal{E}$ we have $F^{\text{OPT}} \geq \sum_j p_j \geq \sum_j \phi_j + \frac{1}{2}\mathbf{E}[R]$. Hence,

$$\begin{aligned}
\mathbf{E}\left[\frac{\int_{t\in\mathcal{D}} \delta^{\text{MLF}}(t)dt}{F^{\text{OPT}}}\,\middle|\,\mathcal{E}\right]\mathbf{P}[\mathcal{E}] &\leq \frac{\mathbf{E}\big[\int_{t\in\mathcal{D}} \frac{2}{\alpha}h(t)dt \,\big|\, \mathcal{E}\big]\,\mathbf{P}[\mathcal{E}]}{\sum_j \phi_j + \frac{1}{2}\mathbf{E}[R]} + \frac{12}{\alpha\gamma} \\
&\leq \frac{\frac{2}{\alpha}(K-k+\beta)\mathbf{E}[\sum_j p_j]}{\sum_j \phi_j + \frac{1}{2}\mathbf{E}[R]} + \frac{12}{\alpha\gamma},
\end{aligned}$$

where we use Lemma 4.7.2 together with the fact that for any input instance $h(t)$ contributes only in those time instants where at least one job is in the system, so at most $\sum_j p_j$. Since

$\mathbf{E}[\sum_j p_j] = \sum_j \phi_j + \mathbf{E}[R]$, we obtain

$$\mathbf{E}\left[\frac{\int_{t\in\mathcal{D}} \delta^{\text{MLF}}(t)dt}{F^{\text{OPT}}} \,\middle|\, \mathcal{E}\right] \mathbf{P}[\mathcal{E}] \;\leq \frac{4(K-k+\beta)}{\alpha} + \frac{12}{\alpha\gamma}.$$

Next, consider the contribution of time instants $t \in \bar{\mathcal{D}}$. Given $\mathcal{E}$, we have $F^{\text{OPT}} \geq \sum_j \phi_j + \frac{1}{2}\mathbf{E}[R]$. Exploiting Lemma 4.7.4, which is given below, we obtain

$$\mathbf{E}\left[\frac{\int_{t\in\bar{\mathcal{D}}} \delta^{\text{MLF}}(t)dt}{F^{\text{OPT}}} \,\middle|\, \mathcal{E}\right] \mathbf{P}[\mathcal{E}] \;\leq \frac{\mathbf{E}[\int_{t\in\bar{\mathcal{D}}} \delta^{\text{MLF}}(t)dt \,|\, \mathcal{E}]\mathbf{P}[\mathcal{E}]}{\sum_j \phi_j + \frac{1}{2}\mathbf{E}[R]} \;\leq\; \frac{\frac{8}{\alpha}\,\mathbf{E}[\sum_j p_j]}{\sum_j \phi_j + \frac{1}{2}\mathbf{E}[R]} \;\leq\; \frac{16}{\alpha}.$$

Putting everything together, we obtain

$$\mathbf{E}\left[\frac{F^{\text{MLF}}}{F^{\text{OPT}}}\right] \leq \frac{4(K-k+\beta)}{\alpha} + \frac{12}{\alpha\gamma} + \frac{16}{\alpha} + \frac{2}{\delta^2}.$$

□

Lemma 4.7.4. $\mathbf{E}\big[\int_{t\in\bar{\mathcal{D}}} \delta^{\text{MLF}}(t)dt \,|\, \mathcal{E}\big]\, \mathbf{P}[\mathcal{E}] \;\leq \frac{8}{\alpha}\,\mathbf{E}[\sum_j p_j]$.

Proof. We use Lemma 4.7.5, the proof of which is subject of Section 4.7.1. We have

$$\begin{aligned}
\mathbf{E}\left[\int_{t\in\bar{\mathcal{D}}} \delta^{\text{MLF}}(t)dt \,\middle|\, \mathcal{E}\right] \mathbf{P}[\mathcal{E}] \;&\leq \mathbf{E}\left[\int_{t\in\bar{\mathcal{D}}} \delta^{\text{MLF}}(t)dt\right] \\
&= \int_{t\geq 0} \mathbf{E}\big[\delta^{\text{MLF}}(t) \,|\, t\in\bar{\mathcal{D}}\big]\, \mathbf{P}[t\in\bar{\mathcal{D}}]\, dt \\
&= \int_{t\geq 0} \sum_{s=1}^{n} s\, \mathbf{P}[\delta^{\text{MLF}}(t) = s \,|\, t\in\bar{\mathcal{D}}]\, \mathbf{P}[t\in\bar{\mathcal{D}}]\, dt \\
&= \int_{t\geq 0} \sum_{s=1}^{n} s\, \mathbf{P}[t\in\bar{\mathcal{D}} \,|\, \delta^{\text{MLF}}(t) = s]\, \mathbf{P}[\delta^{\text{MLF}}(t) = s]\, dt \\
&\leq \int_{t\geq 0} \sum_{s=1}^{n} s\, \mathrm{e}^{-\alpha s/8}\, \mathbf{P}[\delta^{\text{MLF}}(t) = s]\, dt \\
&\leq \frac{8}{\alpha} \int_{t\geq 0} \sum_{s=1}^{n} \mathbf{P}[\delta^{\text{MLF}}(t) = s]\, dt \\
&= \frac{8}{\alpha} \int_{t\geq 0} \mathbf{P}[\delta^{\text{MLF}}(t) \geq 1]\, dt \\
&= \frac{8}{\alpha}\, \mathbf{E}[\textstyle\sum_j p_j],
\end{aligned}$$

where the fifth inequality is due to Lemma 4.7.5 and the sixth inequality follows since $\mathrm{e}^{-x} < \frac{1}{x}$ for $x > 0$. □

4.7.1 High Probability Bound

To complete the proof we are left to show that with high probability at any time t the number of lucky jobs is a good fraction of the overall number of jobs in the system.

Lemma 4.7.5. *For any $s \in [n]$, at any time t, $\mathbf{P}[\delta^l(t) < \frac{1}{2}\alpha\delta^{\text{MLF}}(t) \mid \delta^{\text{MLF}}(t) = s] \leq \mathrm{e}^{-\alpha s/8}$.*

Let $S \subseteq J$. We condition the probability space on the event that (i) the set of jobs that are alive at time t in MLF is equal to S, i.e., $(S^{\text{MLF}}(t) = S)$, and (ii) the processing times of all jobs not in S are fixed to values that are specified by a vector $\mathbf{x}_{\bar{S}}$, which we denote by $(\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}})$. We define the event $\mathcal{F}(t, S, \mathbf{x}_{\bar{S}}) := ((S^{\text{MLF}}(t) = S) \cap (\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}}))$.

Recall that we defined $X_j^l(t) = X_j^l \cdot X_j^{\text{MLF}}(t)$. Since we condition on $(S^{\text{MLF}}(t) = S)$, we have for each $j \in J$

$$X_j^l(t) = \begin{cases} X_j^l & \text{if } j \in S, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Thus,

$$\mathbf{E}[\delta^l(t) \mid \mathcal{F}(t, S, \mathbf{x}_{\bar{S}})] = \sum_{j \in J} \mathbf{P}[X_j^l(t) = 1 \mid \mathcal{F}(t, S, \mathbf{x}_{\bar{S}})] = \sum_{j \in S} \mathbf{P}[X_j^l = 1 \mid \mathcal{F}(t, S, \mathbf{x}_{\bar{S}})] .$$

In order to prove Lemma 4.7.5 we proceed as follows. We first prove that, conditioned on $\mathcal{F}(t, S, \mathbf{x}_{\bar{S}})$, the random variables $(X_j^l \mid \mathcal{F}(t, S, \mathbf{x}_{\bar{S}}))$, $j \in S$, are independent. After that, we prove that the expected number of jobs that are lucky and alive at time t is at least α times the number jobs that are active at this time, i.e.,

$$\mathbf{E}[\delta^l(t) \mid \mathcal{F}(t, S, \mathbf{x}_{\bar{S}})] \geq \alpha|S|.$$

We can then prove the above lemma simply by using a Chernoff bound argument.

Proof of Lemma 4.7.5. For each $j \in S$ we define $Y_j := (X_j^l \mid \mathcal{F}(t, S, \mathbf{x}_{\bar{S}}))$. Then the Y_j's are independent. Moreover, $\mathbf{E}[\sum_{j \in S} Y_j] = \mathbf{E}[\delta^l(t) \mid \mathcal{F}(t, S, \mathbf{x}_{\bar{S}})] \geq \alpha|S|$. Applying Chernoff's bound (see Theorem 2.4.10 (2.2)), we obtain

$$\begin{aligned} \mathbf{P}[\delta^l(t) < \tfrac{1}{2}\alpha\delta(t) \mid \mathcal{F}(t, S, \mathbf{x}_{\bar{S}})] &= \mathbf{P}[\textstyle\sum_{j \in S} Y_j < \frac{1}{2}\alpha|S|] \\ &\leq \mathbf{P}[\textstyle\sum_{j \in S} Y_j < \frac{1}{2}\mathbf{E}[\sum_{j \in S} Y_j]] \;\leq\; \mathrm{e}^{-\alpha|S|/8}. \end{aligned}$$

Finally, summing over all possible subsets $S \subseteq J$ with $|S| = s$ and all possible assignments $\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}}$, the lemma follows. □

In the rest of this section we only consider properties of the schedule produced by MLF. We therefore omit the superscript MLF in the notation below.

Independence of Being Lucky

We first investigate the probability space conditioned on the event $\mathcal{F}(t, S, \mathbf{x}_{\bar{S}}) = ((S(t) = S) \cap (\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}})$ more closely and then prove that the random variables $Y_j = (X_j^l \mid \mathcal{F}(t, S, \mathbf{x}_{\bar{S}}))$, $j \in S$, are independent.

Lemma 4.7.6. *Assume $S(t) = S$ and $\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}}$. Then the schedule of MLF up to time t is uniquely determined.*

Proof. Assume otherwise. Then there exist two different schedules $\mathcal{S}_1$ and $\mathcal{S}_2$ such that $S^{\mathcal{S}_1}(t) = S^{\mathcal{S}_2}(t) = S$. Let I_1 and I_2 be the corresponding instances. Since the processing times of jobs not in S are fixed, I_1 and I_2 differ in the processing times of some subset of the jobs in S. Let $t' \leq t$ be the first time where $\mathcal{S}_1$ and $\mathcal{S}_2$ differ. MLF changes its scheduling decision if either (i) a new job is released or (ii) an active job is completed. Since the release dates are the same in I_1 and I_2, a job j was completed at time t' in one schedule, say $\mathcal{S}_1$, but not in the other. Since j must belong to S and $t' \leq t$, this contradicts the hypothesis that $S^{\mathcal{S}_1}(t) = S$. □

Corollary 4.7.1. *Assume $S(t) = S$ and $\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}}$. Then, for each $j \in S$, $x_j(t)$ is a uniquely determined constant.*

Subsequently, given that $S(t) = S$ and $\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}}$, we set $\pi_j := x_j(t)$ for all $j \in S$. We state the following important fact.

Fact 4.7.1. *Let I be an instance such that $S(t) = S$ and $\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}}$. Then every instance I', with $\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}}$ and $p_{jI'} \geq p_{jI}$ for each $j \in S$, satisfies $x_{jI'}(t) = x_{jI}(t)$ for each $j \in J$.*

In particular, we can generate all instances satisfying $S(t) = S$ and $\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}}$ as follows. Let I_0 be defined as $\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}}$ and $p_{jI_0} := \pi_j$ for each $j \in S$. Note that I_0 is not contained in $\mathcal{F}(t, S, \mathbf{x}_{\bar{S}})$, since $S_{I_0}(t) = \emptyset$; but every instance I with $\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}}$ and $p_{jI} > p_{jI_0}$, for each $j \in S$, is contained in $\mathcal{F}(t, S, \mathbf{x}_{\bar{S}})$.

Lemma 4.7.7. *Assume $S(t) = S$ and $\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}}$. Moreover, let $\pi_j = x_j(t)$ for all $j \in S$. Then the following events are equivalent:*

$$(S(t) = S) \cap (\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}}) \equiv \bigcap_{j \in S} (p_j > \pi_j) \cap (\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}}).$$

Proof. Let I be an instance such that $S(t) = S$ and $\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}}$. By Lemma 4.7.6, the time spent by MLF on $j \in S$ up to time t is $x_j(t) = \pi_j$. Since j is active at time t, $p_j > x_j(t) = \pi_j$.

Next, let I be an instance such that $p_{jI} > \pi_j$ for each $j \in S$ and $\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}}$. Let I_0 be defined as $\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}}$ and $p_{jI_0} := \pi_j$ for each $j \in S$. For each $j \in S$ we have $p_{jI} > \pi_j = p_{jI_0}$. From the discussion above we conclude that $I \in \mathcal{F}(t, S, \mathbf{x}_{\bar{S}})$. □

Lemma 4.7.8. *The variables $Y_j = (X_j^l \mid \mathcal{F}(t, S, \mathbf{x}_{\bar{S}}))$, $j \in S$, are independent.*

Proof. Let $R \subseteq S$. For each $j \in R$ let $a_j \in \{0,1\}$ and let L_j denote the set of processing times such that $(p_j \in L_j)$ if and only if $(X_j^l = a_j)$. From Lemma 4.7.7 we obtain

$$\begin{aligned}\mathbf{P}\left[\bigcap_{j\in R} X_j^l = a_j \,\middle|\, \mathcal{F}(t,S,\mathbf{x}_{\bar{S}})\right] &= \mathbf{P}\left[\bigcap_{j\in R} p_j \in L_j \,\middle|\, \bigcap_{j\in S}(p_j > \pi_j) \cap (\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}})\right] \\ &= \frac{\mathbf{P}[\bigcap_{j\in R}(p_j \in L_j) \cap \bigcap_{j\in S}(p_j > \pi_j) \cap (\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}})]}{\mathbf{P}[\bigcap_{j\in S}(p_j > \pi_j) \cap (\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}})]} \\ &= \frac{\mathbf{P}[\bigcap_{j\in R}(p_j \in L'_j) \cap \bigcap_{j\in S\setminus R}(p_j > \pi_j) \cap (\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}})]}{\mathbf{P}[\bigcap_{j\in S}(p_j > \pi_j) \cap (\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}})]},\end{aligned}$$

where L'_j is defined as the intersection of L_j and $(\pi_j, 2^K]$. Using the fact that processing times are perturbed independently, we obtain

$$\begin{aligned}\mathbf{P}\left[\bigcap_{j\in R} X_j^l = a_j \,\middle|\, \mathcal{F}(t,S,\mathbf{x}_{\bar{S}})\right] &= \frac{\prod_{j\in R}\mathbf{P}[p_j \in L'_j]\,\mathbf{P}[\bigcap_{j\in S\setminus R}(p_j > \pi_j) \cap (\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}})]}{\prod_{j\in R}\mathbf{P}[p_j > \pi_j]\,\mathbf{P}[\bigcap_{j\in S\setminus R}(p_j > \pi_j) \cap (\mathbf{p}_{\bar{S}} = \mathbf{x}_{\bar{S}})]} \\ &= \prod_{j\in R}\frac{\mathbf{P}[p_j \in L'_j]}{\mathbf{P}[p_j > \pi_j]} \;=\; \prod_{j\in R}\mathbf{P}[X_j^l = a_j \,|\, p_j > \pi_j]\,. \qquad (4.3)\end{aligned}$$

The above equality holds for any subset $R \subseteq S$. In particular, for a singleton set $\{j\}$ we obtain

$$\mathbf{P}[X_j^l = a_j \,|\, \mathcal{F}(t,S,\mathbf{x}_{\bar{S}})] = \mathbf{P}[X_j^l = a_j \,|\, p_j > \pi_j]\,. \qquad (4.4)$$

Finally, combining (4.3) and (4.4), we obtain

$$\mathbf{P}\left[\bigcap_{j\in R} X_j^l = a_j \,\middle|\, \mathcal{F}(t,S,\mathbf{x}_{\bar{S}})\right] = \prod_{j\in R}\mathbf{P}[X_j^l = a_j \,|\, \mathcal{F}(t,S,\mathbf{x}_{\bar{S}})]\,.$$

□

Expected Number of Lucky and Alive Jobs

From Equation (4.4) in the proof of Lemma 4.7.8 we learn that if we concentrate on the probability space conditioned on the event $\mathcal{F}(t,S,\mathbf{x}_{\bar{S}})$ then

$$\mathbf{P}[X_j^l = a_j \,|\, \mathcal{F}(t,S,\mathbf{x}_{\bar{S}})] = \mathbf{P}[X_j^l = a_j \,|\, p_j > \pi_j] \quad \text{for each } j \in S.$$

This relation is very useful in proving the following lemma.

Lemma 4.7.9. *For every $j \in S$, $\mathbf{P}[X_j^l = 1 \,|\, \mathcal{F}(t,S,\mathbf{x}_{\bar{S}})] \geq \alpha$. Thus, $\mathbf{E}[\delta^l(t) \,|\, \mathcal{F}(t,S,\mathbf{x}_{\bar{S}})] \geq \alpha|S|$.*

Proof. First, let $\check{p}_j \in (2^{i-1}, 2^i]$ for some integer $i > k$. Due to Fact 4.4.2 the processing time p_j is chosen randomly from a subrange of $(2^{i-1}, 2^i]$. Hence,

$$\mathbf{P}[X_j^l = 1 \,|\, \mathcal{F}(t, S, \mathbf{x}_{\bar{S}})] = \mathbf{P}[p_j \geq (1+\gamma)2^{i-1} \,|\, p_j > \pi_j] \;\geq\; \mathbf{P}[\varepsilon_j \geq \gamma 2^{i-1} \,|\, p_j > \pi_j]\,,$$

where the second inequality is due to the fact that $\phi_j \geq 2^{i-1}$. In Appendix 4.B we show that the events $(\varepsilon_j \geq \gamma 2^{i-1})$ and $(p_j > \pi_j)$ are positively correlated. We have

$$\mathbf{P}[X_j^l = 1 \,|\, \mathcal{F}(t, S, \mathbf{x}_{\bar{S}})] \geq \mathbf{P}[\varepsilon_j \geq \gamma 2^{i-1}] \;\geq\; \mathbf{P}[\varepsilon_j \geq (1+\gamma)2^{k-1}]\,,$$

where the last inequality holds for every i, $k < i \leq K$, if we choose $\gamma \leq 2^{k-K}$.

Next, let $\check{p}_j \in [1, 2^k]$. Due to Fact 4.4.1 the processing time p_j is chosen completely at random from $[1, 2^k]$. Let L_j denote the set of all processing times such that $(X_j^l = 1)$ holds. Then

$$\mathbf{P}[X_j^l = 1 \,|\, \mathcal{F}(t, S, \mathbf{x}_{\bar{S}})] = \mathbf{P}[\varepsilon_j \in L_j \,|\, \varepsilon_j > \pi_j] \geq \mathbf{P}[\varepsilon_j \geq (1+\gamma)2^{k-1}]\,.$$

To prove that the last inequality holds, we distinguish two cases:

(a) Let $\pi_j < (1+\gamma)2^{k-1}$. Since $\mathbf{P}[\varepsilon_j > \pi_j] \leq 1$,

$$\mathbf{P}[\varepsilon_j \in L_j \,|\, \varepsilon_j > \pi_j] \geq \mathbf{P}[(\varepsilon_j \in L_j) \cap (\varepsilon_j > \pi_j)] \geq \mathbf{P}[\varepsilon_j \geq (1+\gamma)2^{k-1}]\,.$$

(b) Let $\pi_j \geq (1+\gamma)2^{k-1}$. Then

$$\mathbf{P}[\varepsilon_j \in L_j \,|\, \varepsilon_j > \pi_j] = 1 \geq \mathbf{P}[\varepsilon_j \geq (1+\gamma)2^{k-1}]\,.$$

Assuming that the smoothing distribution f satisfies (P1), the lemma follows. □

4.7.2 Adaptive Adversary

Recall that the adaptive adversary may change the input instance on basis of the outcome of the random process. This additional power may affect the correlation technique that we used in Lemmas 4.7.2 and 4.7.9. However, as discussed in Appendix 4.B these lemmas also hold for an adaptive adversary. Thus, the upper bound on the smoothed competitive ratio given in Theorem 4.7.1 also holds against an adaptive adversary.

4.8 Lower Bounds

4.8.1 Lower Bounds for the Partial Bit Randomization Model

The first bound is an $\Omega(2^{K/6-k/2})$ one on the smoothed competitive ratio for any deterministic algorithm against an oblivious adversary. We advise the reader to first read the proof for the

adaptive adversary since this bound is more intuitive. In the lower bound proofs, we assume that the smoothing distribution is well-shaped with $\mu = 2^{k-1} + \frac{1}{2}$.

Theorem 4.8.1. *Any deterministic algorithm* ALG *has smoothed competitive ratio* $\Omega(2^{K/6-k/2})$ *for every* $k \leq K/3$ *against an oblivious adversary in the partial bit randomization model.*

Proof. For notational convenience, we assume that K is even. The input sequence for the lower bound is divided into two phases.

Phase 1: At time $t = 0$, the adversary releases $N := 2^{K/2} + \lfloor (2^{K-k} - 2)/3 \rfloor$ jobs and runs ALG on these jobs up to the first time $\hat{t}$ when one of the following two events occurs: (i) $2^{K/2}$ jobs, denoted by $j_1^*, j_2^*, \ldots, j_{2^{K/2}}^*$, have been processed for at least $2^{K/2}$ time units, or (ii) one job, say j^*, has been processed for $2^K - 2^{k+1}$ time units. Subsequently, we call jobs released in the first phase *phase-1 jobs*.

Let $x_j^{\mathrm{ALG}}(\hat{t})$ denote the amount of time spent by algorithm ALG on job j up to time $\hat{t}$. We fix the initial processing time of each job j to $\check{p}_j := x_j^{\mathrm{ALG}}(\hat{t}) + 2^{k+1}$. Note that after smoothing the $\check{p}_j$'s we have $x_j^{\mathrm{ALG}}(\hat{t}) + 2^k < p_j < x_j^{\mathrm{ALG}}(\hat{t}) + 3 \cdot 2^k$ for each j. That is, in the schedule produced by ALG, each job has a remaining processing time between 2^k and $3 \cdot 2^k$ at time $\hat{t}$. Moreover, ALG has not completed any job at this time, i.e., $\delta^{\mathrm{ALG}}(\hat{t}) = N$.

Instead of considering an optimal scheduling algorithm, we consider a scheduling algorithm $\mathcal{S}$ that schedules the jobs as described below. Clearly, the total flow time of OPT is upper bounded by the total flow time of $\mathcal{S}$.

Let $\hat{t}$ be determined by case (i). Then $\mathcal{S}$ does not process jobs $j_1^*, j_2^*, \ldots, j_{2^{K/2}}^*$ before all other jobs are completed. Therefore, at least 2^K time units can be allocated on the other jobs. Since each of these $N - 2^{K/2}$ jobs has remaining processing time at most $3 \cdot 2^k$, $\mathcal{S}$ has completed at least

$$\min\left(N - 2^{K/2}, \left\lfloor \frac{2^K}{3 \cdot 2^k} \right\rfloor\right) \geq N - 2^{K/2}$$

jobs, i.e., all these jobs. In case (ii), by not processing job j^*, $\mathcal{S}$ completes at least

$$\min\left(N - 1, \left\lfloor \frac{2^K - 2^{k+1}}{3 \cdot 2^k} \right\rfloor\right) \geq N - 2^{K/2}$$

of the other jobs. Thus, we obtain $\delta^{\mathcal{S}}(\hat{t}) \leq 2^{K/2}$.

Phase 2: Starting from time $\hat{t}$, the adversary releases a sequence of $L := 2^{5K/3-k}$ jobs, denoted by $N+1, N+2, \ldots, N+L$, for a period of $\tilde{t} := \mu L$, where $\mu := 2^{k-1} + \frac{1}{2}$. The release time of job $j = N+i$ is $r_j := \hat{t} + (i-1)\mu$, for $i = 1, \ldots, L$. Each such job j has initial processing time $\check{p}_j := 1$ and its smoothed processing time satisfies $p_j \leq 2^k$. Subsequently, we call jobs released in the second phase *phase-2 jobs*.

To analyze the number of jobs in the system of ALG and $\mathcal{S}$ during the second phase, we define the random variables $X_j := p_{N+j} - \mu$, for $j = 1, \ldots, L$. Note that the X_j's are independently distributed random variables with zero mean. Define $S_0 := 0$ and $S_i :=$

$\sum_{j=1}^{i} X_j$, for $i = 1, \ldots, L$. Applying Kolmogorov's inequality (see Theorem 2.4.14), we obtain

$$\mathbf{P}\left[\max_{0\leq i\leq L}|S_i| \geq \mu\sqrt{L}\right] \leq \frac{\mathbf{E}\left[S_L^2\right]}{\mu^2 L} \leq \frac{1}{3} \tag{4.5}$$

The last inequality follows since $\mathbf{E}[S_L^2] = \mathbf{Var}[S_L]$ and the variance of the random variable S_L for the uniform distribution is $L(2^{2k}-1)/12$. The bound holds for any well-shaped distribution, since among these distributions the variance is maximized by the uniform distribution.

Consider a schedule $\mathcal{Q}$ only processing phase-2 jobs. The amount of idle time up to time $\hat{t} + i\mu$ is given by

$$I_0 := 0 \quad \text{and} \quad I_i := \max\left(I_{i-1}, i\mu - \sum_{j=1}^{i} p_{N+j}\right).$$

Hence, the total idle time up to time $\hat{t} + i\mu$ for this algorithm is

$$I_i = \max_{0\leq j\leq i} -S_j.$$

By (4.5) we know that with probability at least $\frac{2}{3}$ the total idle time at any time $\hat{t} + i\mu$ stays below $\mu\sqrt{L}$.

We first derive a lower bound on the number of jobs that are in the system of ALG during the second phase.

Lemma 4.8.1. *With probability at least $\frac{2}{3}$, at any time $t \in [\hat{t}, \hat{t}+\tilde{t}]$: $\delta^{\text{ALG}}(t) \geq N - \frac{1}{2}\sqrt{L} - 1$.*

Proof. ALG can do no better than the SRPT rule during the second phase. Each phase-1 job has remaining processing time larger than 2^k. Therefore, ALG follows $\mathcal{Q}$ using the idle time to schedule phase-1 jobs, unless a phase-1 job has received so much processing time that its remaining processing time is less than the processing time of the newly released job. This leads to at most an additional 2^k time spent on phase-1 jobs. Hence, with probability at least $\frac{2}{3}$, at most $\frac{1}{2}\sqrt{L} + 1$ phase-1 jobs are finished by ALG during the second phase. □

$\mathcal{S}$ also follows $\mathcal{Q}$ during the second phase using the idle time to schedule phase-1 jobs. We next give an upper bound on the number of jobs in the system of $\mathcal{S}$ during the second phase.

Lemma 4.8.2. *With probability at least $\frac{2}{3}$, at any time $t \in [\hat{t}, \hat{t}+\tilde{t}]$: $\delta^{\mathcal{S}}(t) \leq 2^{K/2} + 2\sqrt{L} + 2$.*

Proof. Consider the amount of additional volume brought into the system. Just before time $t = \hat{t} + i\mu$ this is

$$\sum_{j=1}^{i} p_j - (i\mu - I_i),$$

i.e., the total processing time of phase-2 jobs released before time t minus the amount of time processed on phase-2 jobs. Hence, the maximum amount of additional volume before the release of a phase-2 job is given by

$$\Delta V = \max_{0\le i\le L}(S_i + I_i) = \max_{0\le i\le L}(S_i + \max_{0\le j\le i} -S_j) = \max_{0\le j\le i\le L}(S_i - S_j).$$

The probability that this value exceeds some threshold value is bounded by

$$\mathbf{P}[\Delta V > 2\lambda] \le \mathbf{P}\left[\max_{0\le i,j\le L}(S_i - S_j) > 2\lambda\right] \le \mathbf{P}\left[\max_{0\le i\le L}|S_i| > \lambda\right]$$

Setting λ to $\mu\sqrt{L}$, by (4.5) this probability is at most $\frac{1}{3}$.

To conclude the proof we need the following fact, which can easily be proven by induction on the number of phase-2 jobs released.

Fact 4.8.1. *Just before the release of a phase-2 job, $\mathcal{S}$ has no more than one phase-2 job with remaining processing time less than μ.*

Assume ΔV attains its maximum just before time $t' := \hat{t} + i\mu$. Due to Fact 4.8.1 no more than one phase-2 job has remaining processing time less than μ. At time t' a new phase-2 job is released. Therefore, with probability at least $\frac{2}{3}$, the number of phase-2 jobs that are in the system is bounded by

$$\frac{2\mu\sqrt{L}}{\mu} + 2 = 2\sqrt{L} + 2.$$

□

By the above two lemmas, with constant probability the total flow time of the two schedules is bounded by

$$\begin{aligned} F^{\mathrm{ALG}} &\ge (N - \sqrt{L}/2 - 1)\tilde{t}, \\ F^{\mathcal{S}} &\le N\hat{t} + (2^{K/2} + 2\sqrt{L} + 2)\tilde{t} + (2^{K/2} + 2\sqrt{L} + 2)(3N2^k + 2\mu\sqrt{L}), \end{aligned}$$

where the contribution of the period after time $\hat{t} + \tilde{t}$ for $\mathcal{S}$ is bounded by the number of jobs at time $\hat{t} + \tilde{t}$ times the remaining processing time at the start of this phase.

To bound the ratio between F^{ALG} and $F^{\mathcal{S}}$, we note that from the upper bounds on N and $\hat{t}$ it follows that $N\hat{t} \le 2(2^{K/2} + 2\sqrt{L} + 2)\mu L$. Moreover, we know from the definition of N and μ that $3N2^k + 2\mu\sqrt{L} \le 8\mu L$. Hence, by restricting $k \le K/3$, we have that

$$\mathbf{E}\left[\frac{F^{\mathrm{ALG}}}{F^{\mathrm{OPT}}}\right] = \Omega\left(\frac{N - \sqrt{L}/2 - 1}{2^{K/2} + 2\sqrt{L} + 2}\right) = \Omega\left(\frac{2^{K-k} + 2^{K/2} - 2^{5K/6-k/2}}{2^{5K/6-k/2}}\right) = \Omega\left(2^{K/6-k/2}\right).$$

□

As mentioned before, the adaptive adversary is stronger than the oblivious one as it may construct the input instance revealed to the algorithm after time t also on the basis of the execution of the algorithm up to time t. The next theorem gives an $\Omega(2^{K-k})$ lower bound on the smoothed competitive ratio of any deterministic algorithm under the partial bit randomization model, thus showing that MLF achieves up to a constant factor the best possible ratio in this model. The lower bound is based on ideas similar to those used by Motwani et al. [MPT94] for an $\Omega(2^K)$ non-clairvoyant deterministic lower bound.

Theorem 4.8.2. *Any deterministic algorithm* ALG *has smoothed competitive ratio* $\Omega(2^{K-k})$ *against an adaptive adversary in the partial bit randomization model.*

Proof. The input sequence for the lower bound is divided into two phases.

Phase 1: At time $t = 0$, the adversary releases $N := \lfloor (2^{K-k} - 2)/3 \rfloor + 1$ jobs. We run ALG on these jobs up to the first time $\hat{t}$ when a job, say j^*, has been processed for $2^K - 2^{k+1}$ time units, $k \leq K - 2$. The adversary makes sure that none of the N jobs is completed up to time $\hat{t}$. Let $x_j^{\text{ALG}}(\hat{t})$ denote the amount of time spent by algorithm ALG on job j up to time $\hat{t}$. We fix the initial processing time of each job j to $\check{p}_j := x_j^{\text{ALG}}(\hat{t}) + 2^{k+1}$. Note that after smoothing the $\check{p}_j$'s we have $x_j^{\text{ALG}}(\hat{t}) + 2^k < p_j < x_j^{\text{ALG}}(\hat{t}) + 3 \cdot 2^k$ for each j. That is, each job has a remaining processing time between 2^k and $3 \cdot 2^k$. Therefore, ALG will not complete any job at time $\hat{t}$, i.e., $\delta^{\text{ALG}}(\hat{t}) = N$.

Consider the optimal algorithm OPT. If OPT does not process j^* until time $\hat{t}$, $2^K - 2^{k+1}$ time units can be allocated on the other jobs. Thus, at least

$$\frac{2^K - 2^{k+1}}{3 \cdot 2^k} \geq \left\lfloor \frac{2^{K-k} - 2}{3} \right\rfloor = N - 1$$

of these jobs are completed by OPT until time $\hat{t}$, i.e., $\delta^{\text{OPT}}(\hat{t}) = 1$.

Phase 2: The adaptive adversary releases a sequence $N + 1, N + 2, \ldots$ of jobs. The release time of job $j = N + i$ is $r_j := \hat{t}$ for $i = 1$ and $r_j := r_{j-1} + p_{j-1}$ for $i > 1$. Each such job j has initial processing time $\check{p}_j := 1$ and therefore its smoothed processing time satisfies $p_j \leq 2^k$.

OPT will then complete every job released in the second phase before the next one is released. The optimal strategy for ALG is also to process the jobs released in the second phase to completion as soon as they are released since every job left uncompleted from the first phase has remaining processing time larger than 2^k.

The second phase goes on for a time interval larger than 2^{3K-2k} which is an upper bound on the contribution to the total flow time of any algorithm in the first phase of the input sequence. Therefore, in terms of total flow time, the second phase dominates the first phase for both ALG and OPT. Since in the second phase ALG has $\Omega(N)$ jobs and OPT has $O(1)$ jobs in the system, we obtain a competitive ratio of $\Omega(N) = \Omega(2^{K-k})$. □

4.8.2 Lower Bounds for Symmetric Smoothing Models

Since we are using the partial bit randomization model we do not smoothen the processing times symmetrically around their initial values. Therefore, a natural question is whether or not symmetric smoothing models (see also Section 2.3) are more suitable to analyze MLF. We answer this question in the negative by providing a lower bound of $\Omega(2^K)$ on the performance of MLF under the following symmetric smoothing model.

Consider a function $\vartheta : \mathbb{R}^+ \to \mathbb{R}^+$ which is continuous and non-decreasing. In the symmetric smoothing model according to ϑ we smoothen the initial processing times as follows:

$$p_j := \max(1, \check{p}_j + \varepsilon_j), \quad \text{where } \varepsilon_j \overset{f}{\leftarrow} [-\vartheta(\check{p}_j)/2, \vartheta(\check{p}_j)/2],$$

and f is the uniform distribution. As will be discussed below, the symmetric smoothing model according to ϑ captures the additive smoothing model, a variant of the additive relative smoothing model, and the relative smoothing model.

We prove the following lower bound for a symmetric smoothing model according to ϑ.

Theorem 4.8.3. *Let $\vartheta : \mathbb{R}^+ \to \mathbb{R}^+$ be function such that there exists a $x^* \in \mathbb{R}^+$ satisfying $x^* - \vartheta(x^*)/2 > 2^{K-2}$ and $x^* + \vartheta(x^*)/2 = 2^{K-1} + a$ for some constant $a \geq 1$. Then there exists an $\Omega(2^K/a)$ lower bound on the smoothed competitive ratio of MLF against an oblivious adversary in the symmetric smoothing model according to ϑ.*

The additive symmetric smoothing model over $[-c, c]$ is equivalent to the above defined model with $\vartheta(x) := 2c$. Since Theorem 4.8.3 requires $x^* - c > 2^{K-2}$ and x^* is defined as $x^* = 2^{K-1} + a - c$, we obtain $c < 2^{K-3} + a/2$. By fixing $a := 1$, Theorem 4.8.3 yields an $\Omega(2^K)$ lower bound for the symmetric additive smoothing model against an oblivious adversary.

We can use the symmetric smoothing model according to ϑ to simulate a variant of the additive relative symmetric smoothing model. We define $\vartheta(x) := 2x^c$ for some $c \geq 0$. The processing times are then smoothed according to a symmetric smoothing model over $[-x^c, x^c]$. Define $c := c(y) = y/\log(x^*)$ as a function of $y \in \mathbb{R}^+$, and fix $a := 1$. Then, $x^* = 2^{K-1} + 1 - 2^y$. The condition $x^* - (x^*)^c > 2^{K-2}$ is satisfied if $y \leq K - 3$. Since $c(y)$ is monotone increasing, we obtain the restriction $c \leq c(K-3) = (K-3)/\log(3 \cdot 2^{K-3} + 1)$. From Theorem 4.8.3, we obtain an $\Omega(2^K)$ lower bound for this additive relative symmetric smoothing model.

The relative smoothing model is equivalent to the symmetric smoothing model according to ϑ with $\vartheta(x) := 2\epsilon x$. The conditions in Theorem 4.8.3 are fulfilled if $0 \leq \epsilon \leq (2^{K-2} + a)/(3 \cdot 2^{K-2} + a)$. Hence, for $a := 1$, we obtain an $\Omega(2^K)$ lower bound for the relative smoothing model.

Proof of Theorem 4.8.3. The input sequence of the adversary consists of two phases. Let $\mathcal{S}$ be the algorithm that during the first phase schedules the jobs to completion in the order in which

they are released, and during the second phase schedules the jobs that are released in this phase to completion in the order in which they are released. After having completed all phase-2 jobs, $\mathcal{S}$ finishes the remaining phase-1 jobs. We upper bound OPT by $\mathcal{S}$. To prove the theorem, we show that with constant probability $F^{\text{MLF}}/F^{\mathcal{S}} = \Omega(2^K/a)$. Then $\mathbf{E}[F^{\text{MLF}}/F^{\text{OPT}}] = \Omega(2^K/a)$. Without loss of generality, we assume that $K \geq 3$, and we define $L := \vartheta(x^*)$.

Phase 1: At time $t = 0$, $M := 8\max(L^3/2^K, 1)$ jobs are released with initial processing time $\check{p}_1 := x^*$ and then every $\check{p}_1$ time units one job with the same initial processing time is released. The total number of jobs released in the first phase is $N := \max(L^4, 2^{2K}/L^2)$. Note that by definition of x^*, the smoothed processing time of each phase-1 job is at least 2^{K-2}.

Let $T_1(i)$ be the total processing time of jobs released in phase 1 at or before time $i\check{p}_1$, for $i = 0, 1, \ldots, N - M$. Define $S_0 := 0$ and $S_i := S_{i-1} + \varepsilon_i = \sum_{j=1}^{i} \varepsilon_j$, for $i = 1, \ldots, N$. As $\mathbf{E}[\varepsilon_j] = 0$ and all ε_j are drawn independently, we have $\mathbf{E}[S_i] = 0$ and $\mathbf{E}[S_i^2] = iL^2/12$, for all $i = 0, \ldots, N$. Applying Kolmogorov's inequality (see Theorem 2.4.14), we obtain

$$\mathbf{P}\left[\max_{0 \leq k \leq N} |S_k| > L\sqrt{N}\right] \leq \frac{1}{12}.$$

Hence, we have with probability at least $11/12$ that for all $i = 0, \ldots, N - M$

$$(i + M)\check{p}_1 - L\sqrt{N} \leq T_1(i) \leq (i + M)\check{p}_1 + L\sqrt{N}. \tag{4.6}$$

Subsequently, we assume that (4.6) holds.

Let $\hat{t} := (N - M + 1)\check{p}_1$, and consider a $t \in [0, \hat{t})$. Then the remaining processing time for $\mathcal{S}$ as well as MLF at time t is

$$\begin{aligned} T_1(\lfloor t/\check{p}_1 \rfloor) - t &\geq (\lfloor t/\check{p}_1 \rfloor + M)\check{p}_1 - L\sqrt{N} - t \\ &\geq t - 1 + M\check{p}_1 - L\sqrt{N} - t \;\geq\; M2^{K-2} - L\sqrt{N} - 1 \\ &\geq 2\max(L^3, 2^K) - \max(L^3, 2^K) - 1 > 0. \end{aligned} \tag{4.7}$$

Hence, $\mathcal{S}$ and MLF do not have any idle time during the first phase. Moreover, the remaining processing time for both algorithms is at most $M\check{p}_1 + L\sqrt{N}$.

Consider some $t \in [0, \hat{t})$. There is at most one job that has been processed on by $\mathcal{S}$ but is not yet completed. Hence,

$$\delta^{\mathcal{S}}(t) \leq \frac{M\check{p}_1 + L\sqrt{N}}{2^{K-2}} + 1 = O(M).$$

Consider the schedule produced by MLF up to time $\hat{t}$. The probability that a job released in phase 1 is of class K is at least a/L. The expected number of phase-1 class K jobs is at least aN/L. Applying Chernoff's bound (see Theorem 2.4.10), we know that with probability at least $1 - \mathrm{e}^{aN/(8L)} \geq (\mathrm{e} - 1)/\mathrm{e}$ there are at least $aN/(2L)$ class K phase-1 jobs. Subsequently, we assume that this property holds. Note that the probability that both (4.6) and the bound on

the number of class K jobs hold is at least $(\mathrm{e}-1)/\mathrm{e}-1/12$.

If MLF does not finish any class K job up to time $\hat{t}$ then

$$\delta^{\mathrm{MLF}}(\hat{t}) \geq \frac{aN}{2L}.$$

Otherwise, consider the last time $t \in [0,\hat{t})$ that MLF was processing a job in queue Q_K. By definition of MLF, we know that at this time all lower queues were empty. Moreover, we know that the remaining processing time of each job in this queue is at most a, and we also know from (4.7) that the total remaining processing time is at least $\max(L^3, 2^K) - 1 = L\sqrt{N} - 1$. Hence, at this time the number of alive jobs in the schedule of MLF is at least $(L\sqrt{N}-1)/a$ and also

$$\delta^{\mathrm{MLF}}(\hat{t}) \geq \frac{L\sqrt{N}-1}{a}.$$

Phase 2: At time $\hat{t}$, M jobs with $\check{p}_2 := 2^{K-2}$ are released and then every $\check{p}_2$ time units one job with the same $\check{p}_2$ is released. The total number of jobs released in this phase is $2N$. Note that no job released in the second phase enters queue Q_K.

Let $T_2(i)$ be the total processing time of the phase-2 jobs released at or before time $\hat{t}+i\check{p}_2$. Applying Kolmogorov's inequality yields that with probability at least $11/12$ we have

$$(i+M)\check{p}_2 - L\sqrt{2N} \leq T_2(i) \leq (i+M)\check{p}_2 + L\sqrt{2N}. \tag{4.8}$$

Subsequently, we assume that also (4.8) holds. The probability that the bound on the number of class K jobs and (4.6) and (4.8) hold is at least $(\mathrm{e}-1)/\mathrm{e} - 1/6 > 0.46$.

Using the same arguments as before, we now show that MLF continuously processes phase-2 jobs until time $\bar{t} := \hat{t} + (2N - M + 1)\check{p}_2$. Namely, consider a $t \in [\hat{t},\bar{t})$. Then the remaining processing time for $\mathcal{S}$ as well as MLF at time t is

$$\begin{aligned} T_2(\lfloor (t-\hat{t})/\check{p}_2 \rfloor) - (t-\hat{t}) &\geq (\lfloor (t-\hat{t})/\check{p}_2 \rfloor + M)\check{p}_2 - L\sqrt{2N} - (t-\hat{t}) \\ &\geq M\check{p}_2 - L\sqrt{2N} - 1 \;\geq\; M2^{K-2} - L\sqrt{2N} - 1 \\ &\geq 2\max(L^3, 2^K) - \sqrt{2}\max(L^3, 2^K) - 1 \;>\; 0. \end{aligned}$$

Thus, if MLF does not finish any phase-1 job of class K up to time $\hat{t}$, we have

$$\delta^{\mathrm{MLF}}(t) \geq \frac{aN}{2L}, \quad \text{for } t \in [\hat{t},\bar{t}), \quad \text{and} \quad F^{\mathrm{MLF}} = \Omega\left(\frac{aN}{2L}(2N - M + 1)\check{p}_2\right).$$

Otherwise, we have

$$\delta^{\mathrm{MLF}}(t) \geq \frac{L\sqrt{N}-1}{a}, \quad \text{for } t \in [\hat{t},\bar{t}), \quad \text{and} \quad F^{\mathrm{MLF}} = \Omega\left(\frac{L\sqrt{N}}{a}(2N - M + 1)\check{p}_2\right).$$

Moreover, using the same argumentation as for phase 1, we know that during $[\hat{t}, \bar{t})$, $\mathcal{S}$ has at most $(M\check{p}_2 + L\sqrt{2N})/2^{K-3} + 1 = (2+\sqrt{2})M + 1$ phase-2 jobs in its system. Hence,

$$\delta^{\mathcal{S}}(t) = O(M) \quad \text{for } t \in [\hat{t}, \bar{t}).$$

After time $\bar{t}$, the time needed by $\mathcal{S}$ to finish all jobs is at most

$$M\check{p}_1 + L\sqrt{N} + M\check{p}_2 + L\sqrt{2N} \leq \left(\frac{9+\sqrt{2}}{2} + 1\right) M\check{p}_2 \leq \left(\frac{9+\sqrt{2}}{2} + 1\right)(2N - M + 1)\check{p}_2.$$

Hence,

$$F^{\mathcal{S}} = O(M(2N - M + 1)\check{p}_2).$$

If $N = L^4$ then $M = 8L^3/2^K$ and

$$F^{\text{MLF}}/F^{\mathcal{S}} = \Omega\left(\frac{aN}{2LM}\right) = \Omega(a2^K) \quad \text{or} \quad F^{\text{MLF}}/F^{\mathcal{S}} = \Omega\left(\frac{L\sqrt{N}}{M}\right) = \Omega\left(\frac{2^K}{a}\right).$$

If $N = 2^{2K}/L^2$ then $L^3 \leq 2^K$ and $M = 8$. Moreover,

$$F^{\text{MLF}}/F^{\mathcal{S}} = \Omega\left(\frac{aN}{2LM}\right) = \Omega(a2^K) \quad \text{or} \quad F^{\text{MLF}}/F^{\mathcal{S}} = \Omega\left(\frac{L\sqrt{N}}{M}\right) = \Omega\left(\frac{2^K}{a}\right).$$

Since the probability that (4.6), (4.8), and the bound on the number of class K jobs hold is constant and $a \geq 1$, we have

$$\mathbf{E}\left[\frac{F^{\text{MLF}}}{F^{\text{OPT}}}\right] = \Omega\left(\frac{2^K}{a}\right).$$

□

Obviously, Theorem 4.8.3 also holds for the adaptive adversary. Finally, we remark that we can generalize the theorem to the case that f is a well-shaped function.

4.9 Concluding Remarks

We analyzed the performance of the multi-level feedback algorithm using the novel approach of smoothed analysis. Smoothed competitive analysis provides a unifying framework for worst case and average case analysis of online algorithms. We considered several smoothing models, including the additive symmetric smoothing model proposed by Spielman and Teng [ST01]. The partial bit randomization model yields the best upper bound. In particular, we proved that the smoothed competitive ratio of MLF using this model is $O((2^k/\sigma)^3 + (2^k/\sigma)^2 2^{K-k})$, where σ denotes the standard deviation of the smoothing distribution. The analysis holds for

various distributions. For distributions with $\sigma = \Theta(2^k)$, e.g., for the uniform distribution, we obtain a smoothed competitive ratio of $O(2^{K-k})$. By choosing $k = K$, the result implies a constant upper bound on the average competitive ratio of MLF. We also proved that any deterministic algorithm has smoothed competitive ratio $\Omega(2^{K-k})$. Hence, under this model, the smoothed competitive ratio of MLF is optimal up to a constant factor. For various other symmetric smoothing models we have obtained lower bounds of $\Omega(2^K)$. Thus, these models do not seem to capture the good performance of MLF in practice.

A natural question that arises is whether or not the smoothing of the release dates helps to further reduce the smoothed competitive ratio of MLF. We provide a partial answer to this question: If the initial processing times in $[1, 2^K]$ are smoothed according to the partial bit randomization model and the release dates of the jobs are smoothed by means of a smoothing model that does not disrupt the initial release dates by more than 2^{K-1}, i.e., $|\check{r}_j - r_j| \leq 2^{K-1}$ for each job $j \in J$, we can prove a lower bound of $\Omega(2^{K-k})$ on the smoothed competitive ratio of MLF.

As mentioned in the introduction, one could alternatively define the smoothed competitive ratio as the ratio between the expected cost of the algorithm and the expected optimal cost; see definition (4.2). We remark that from Lemmas 4.7.1, 4.7.2, and 4.7.9 we obtain the same bound under this alternative definition, without the need for any high probability argument.

An interesting open problem is to improve the lower bound against the oblivious adversary in the partial bit randomization model. It can also be of some interest to extend our analysis to the multiple machine case. Following the work of Becchetti and Leonardi [BL01], we can extend Lemma 4.7.1 having an extra factor of K, which will also be in the smoothed competitive ratio. Finally, we hope that this framework of analysis will be extended to other online problems.

4.A Proof of Lemma 4.7.1

We introduce some additional notation. The volume $V^{\mathcal{S}}(t)$ is the sum of the remaining processing times of the jobs that are active at time t. $L^{\mathcal{S}}(t)$ denotes the total work done prior to time t, i.e., the overall time the machine has been processing jobs until time t. For a generic function ϑ ($= \delta$, V, or L), we define $\Delta\vartheta(t) = \vartheta^{\text{MLF}}(t) - \vartheta^{\text{OPT}}(t)$. For ϑ ($= \delta$, V, ΔV, L, or ΔL), the notation $\vartheta_{=k}(t)$ will denote the value of ϑ at time t when restricted to jobs of class k. We use $\vartheta_{\geq h,\leq k}(t)$ to denote the value of ϑ at time t when restricted to jobs of classes between h and k.

Lemma 4.7.1. *For any input instance I, at any time t, $\delta_I^l(t) \leq h_I(t) + \frac{6}{\gamma}\delta_I^{\text{OPT}}(t)$.*

Proof. In the following we omit I when clear from the context. Denote by k_1 and k_2, respectively, the lowest and highest class such that at least one job of that class is in the system at time t. We bound the number of lucky jobs that are active at time t as follows:

$$\delta^l(t) \leq h(t) + \frac{1}{\gamma}\sum_{i=k_1}^{k_2}\frac{V_{=i}^{\text{MLF}}(t)}{2^{i-1}}. \tag{4.9}$$

The bound follows since every job that is lucky at time t is either an ending head job or not. An ending head job might have been processed and therefore we cannot assume anything about its remaining processing time. However, the number of ending head jobs is $h(t)$. For all other lucky jobs we can bound the remaining processing time from below: a job of class i has remaining processing time at least $\gamma 2^{i-1}$. We have

$$\begin{aligned}
\sum_{i=k_1}^{k_2}\frac{V_{=i}^{\text{MLF}}(t)}{2^{i-1}} &= \sum_{i=k_1}^{k_2}\frac{V_{=i}^{\text{OPT}}(t) + \Delta V_{=i}(t)}{2^{i-1}} \\
&\leq 2\delta_{\geq k_1,\leq k_2}^{\text{OPT}}(t) + \sum_{i=k_1}^{k_2}\frac{\Delta V_{=i}(t)}{2^{i-1}} \\
&= 2\delta_{\geq k_1,\leq k_2}^{\text{OPT}}(t) + 2\sum_{i=k_1}^{k_2}\frac{\Delta V_{\leq i}(t) - \Delta V_{\leq i-1}(t)}{2^i} \\
&= 2\delta_{\geq k_1,\leq k_2}^{\text{OPT}}(t) + 2\frac{\Delta V_{\leq k_2}(t)}{2^{k_2}} + 2\sum_{i=k_1}^{k_2-1}\frac{\Delta V_{\leq i}(t)}{2^{i+1}} \\
&\leq 2\delta_{\geq k_1,\leq k_2}^{\text{OPT}}(t) + \delta_{\leq k_1-1}^{\text{OPT}}(t) + 4\sum_{i=k_1}^{k_2}\frac{\Delta V_{\leq i}(t)}{2^{i+1}} \\
&\leq 2\delta_{\leq k_2}^{\text{OPT}}(t) + 4\sum_{i=k_1}^{k_2}\frac{\Delta V_{\leq i}(t)}{2^{i+1}},
\end{aligned} \tag{4.10}$$

where the second inequality follows since a job of class i has size at most 2^i, while the fourth inequality follows since $\Delta V_{\leq k_1-1}(t) = 0$ by definition.

We are left to study the sum in (4.10). For any $t_1 \leq t_2 \leq t$ and a generic function ϑ, denote by $\vartheta^{[t_1,t_2]}(t)$ the value of ϑ at time t when restricted to jobs released between t_1 and t_2, e.g., $L_{\leq i}^{[t_1,t_2]}(t)$ is the work done by time t on jobs of class at most i released between time t_1 and t_2. Denote by $t_i < t$ the maximum between 0 and the last time prior to time t in which a job was processed in queue Q_{i+1} or higher in this specific execution of MLF. Observe that, for $i = k_1, \ldots, k_2$, $[t_{i+1}, t) \supseteq [t_i, t)$.

At time t_i, either the algorithm was processing a job in queue Q_{i+1} or higher, or $t_i = 0$. Thus, at time t_i no jobs were in queues $Q_0, \ldots, Q_i$. Therefore,

$$\Delta V_{\leq i}(t) \leq \Delta V_{\leq i}^{(t_i,t]}(t) \leq L_{>i}^{\text{MLF}(t_i,t]}(t) - L_{>i}^{\text{OPT}(t_i,t]}(t) = \Delta L_{>i}^{(t_i,t]}(t).$$

In the following we adopt the convention $t_{k_1-1} = t$. From the above, we have

$$\begin{aligned}
\sum_{i=k_1}^{k_2} \frac{\Delta L_{>i}^{(t_i,t]}(t)}{2^{i+1}} &= \sum_{i=k_1}^{k_2} \frac{L_{>i}^{\text{MLF}(t_i,t]}(t) - L_{>i}^{\text{OPT}(t_i,t]}(t)}{2^{i+1}} \\
&= \sum_{i=k_1}^{k_2} \sum_{j=k_1-1}^{i-1} \frac{L_{>i}^{\text{MLF}(t_{j+1},t_j]}(t) - L_{>i}^{\text{OPT}(t_{j+1},t_j]}(t)}{2^{i+1}} \\
&= \sum_{j=k_1-1}^{k_2-1} \sum_{i=j+1}^{k_2} \frac{L_{>i}^{\text{MLF}(t_{j+1},t_j]}(t) - L_{>i}^{\text{OPT}(t_{j+1},t_j]}(t)}{2^{i+1}},
\end{aligned}$$

where the second equality follows by partitioning the work done on the jobs released in the interval $(t_i, t]$ into the work done on the jobs released in the intervals $(t_{j+1}, t_j]$, $j = k_1 - 1, \ldots, i-1$.

Let $\bar{i}(j) \in \{j+1, \ldots, k_2\}$ be the index that maximizes $L_{>i}^{\text{MLF}(t_{j+1},t_j]} - L_{>i}^{\text{OPT}(t_{j+1},t_j]}$. Then

$$\begin{aligned}
\sum_{i=k_1}^{k_2} \frac{\Delta L_{>i}^{(t_i,t]}(t)}{2^{i+1}} &\leq \sum_{j=k_1-1}^{k_2-1} \sum_{i=j+1}^{k_2} \frac{L_{>\bar{i}(j)}^{\text{MLF}(t_{j+1},t_j]}(t) - L_{>\bar{i}(j)}^{\text{OPT}(t_{j+1},t_j]}(t)}{2^{i+1}} \\
&\leq \sum_{j=k_1-1}^{k_2-1} \frac{L_{>\bar{i}(j)}^{\text{MLF}(t_{j+1},t_j]}(t) - L_{>\bar{i}(j)}^{\text{OPT}(t_{j+1},t_j]}(t)}{2^{j+1}} \\
&\leq \sum_{j=k_1-1}^{k_2-1} \delta_{>\bar{i}(j)}^{\text{OPT}(t_{j+1},t_j]}(t) \;\leq\; \delta_{\geq k_1}^{\text{OPT}(t_{k_2},t]}(t) \;\leq\; \delta_{\geq k_1}^{\text{OPT}}(t).
\end{aligned}$$

To prove the third inequality observe that every job of class larger than $\bar{i}(j) > j$ released in the time interval $(t_{j+1}, t_j]$ is processed by MLF in the interval $(t_{j+1}, t]$ for at most 2^{j+1} time units. Order the jobs of this specific set by increasing $x_j^{\text{MLF}}(t)$. Now, observe that each

of these jobs has initial processing time at least $2^{\bar{i}(j)} \geq 2^{j+1}$ at their release and we give to the optimum the further advantage that it finishes every such job when processed for an amount $x_j^{\text{MLF}}(t) \leq 2^{j+1}$. To maximize the number of finished jobs the optimum places the work $L_{>\bar{i}(j)}^{\text{OPT}(t_{j+1},t_j]}$ on the jobs with smaller $x_j^{\text{MLF}}(t)$. The optimum is then left at time t with a number of jobs

$$\delta_{>\bar{i}(j)}^{\text{OPT}(t_{j+1},t_j]}(t) \geq \frac{L_{>\bar{i}(j)}^{\text{MLF}(t_{j+1},t_j]}(t) - L_{>\bar{i}(j)}^{\text{OPT}(t_{j+1},t_j]}(t)}{2^{j+1}}.$$

Altogether, we obtain from (4.9), (4.10), and (4.11)

$$\delta^l(t) \leq h(t) + \frac{2}{\gamma}\delta_{\leq k_2}^{\text{OPT}}(t) + \frac{4}{\gamma}\delta_{\geq k_1}^{\text{OPT}}(t) \;\leq\; h(t) + \frac{6}{\gamma}\delta^{\text{OPT}}(t).$$

□

4.B Proving Positive and Negative Correlations

In Lemmas 4.7.2 and 4.7.9 we use the technique described in Section 2.4.6 to prove that two events are negatively or positively correlated. We give some more details in this section.

In both Lemmas we need to prove that two events A' and B' are correlated; in Lemma 4.7.2, $A' := (p_{q_i} \leq 2^i)$ and $B' := (H(t) = H)$, and in Lemma 4.7.9, $A' := (\varepsilon_j \geq \gamma 2^{i-1})$ and $B' := (p_j > \pi_j)$. In both cases, A' is an event that solely depends on the perturbation of some job j, e.g., $j := q_i$ in Lemma 4.7.2 and j itself in Lemma 4.7.9. We condition the probability space in order to make sure that only the processing time of j is random. That is, we fix the processing times of all jobs other than j to $\mathbf{x}_{\bar{j}}$, which we denote by $(\mathbf{p}_{\bar{j}} = \mathbf{x}_{\bar{j}})$. Define $A = (A' \,|\, \mathbf{p}_{\bar{j}} = \mathbf{x}_{\bar{j}})$ and $B = (B' \,|\, \mathbf{p}_{\bar{j}} = \mathbf{x}_{\bar{j}})$. Let Ω denote the conditioned probability space and let $\mathbf{P}$ denote the underlying conditioned probability distribution. The following two statements are easy to verify.

1. Ω together with the partial order $\leq$ and the standard $\max$ and $\min$ operations constitutes a distributive lattice.
2. $\mathbf{P}$ is log-supermodular. The inequality holds even with equality and does not depend on the underlying probability distribution.

We next argue that the events A and B are monotone increasing or decreasing.

Lemma 4.7.2. Let the processing time p_{jI} of job $j = q_i$ in I be fixed such that $I \in A = (p_{q_i} \leq 2^i \,|\, \mathbf{p}_{\bar{j}} = \mathbf{x}_{\bar{j}})$. Define an instance I' with $p_{jI'} \leq p_{jI}$. Then $I' \in A$. Hence, A is monotone decreasing. On the other hand, if the processing time p_{jI} in I is chosen such that $I \in B = (H(t) = H \,|\, \mathbf{p}_{\bar{j}} = \mathbf{x}_{\bar{j}})$, i.e., j is a head job at time t, then j remains a head job in any instance I' with $p_{jI'} \geq p_{jI}$. Therefore, B is monotone increasing. By Theorem 2.4.16, A and B are negatively correlated.

Lemma 4.7.9. Let I be an instance with processing time p_{jI} of j being such that $I \in A = (\varepsilon_j \geq \gamma 2^{i-1} \,|\, \mathbf{p}_{\bar{j}} = \mathbf{x}_{\bar{j}})$. Consider an instance I' with processing time $p_{jI'} \geq p_{jI}$. Clearly, $I' \in A$ and thus A is monotone increasing. Similarly, let p_{jI} be fixed such that $I \in B = (p_j > \pi_j \,|\, \mathbf{p}_{\bar{j}} = \mathbf{x}_{\bar{j}})$. If we consider an instance I' with $p_{jI'} \geq p_{jI}$ then j also satisfies $(p_{jI'} > \pi_j)$ and thus $I' \in B$. That is, B is monotone increasing. By Theorem 2.4.16, we conclude that A and B are positively correlated.

Since the processing times of all jobs are perturbed independently, A' and $(\mathbf{p}_{\bar{j}} = \mathbf{x}_{\bar{j}})$ are independent, i.e., $\mathbf{P}[A' \,|\, \mathbf{p}_{\bar{j}} = \mathbf{x}_{\bar{j}}] = \mathbf{P}[A']$. We exploit this fact as follows in order to prove that the events A' and B' are also correlated. (The second inequality is due to the correlation of A and B.)

$$\begin{aligned}
\mathbf{P}[A' \cap B'] &= \sum_{\mathbf{x}_{\bar{j}}} \mathbf{P}[A' \cap B' \,|\, \mathbf{p}_{\bar{j}} = \mathbf{x}_{\bar{j}}]\, \mathbf{P}[\mathbf{p}_{\bar{j}} = \mathbf{x}_{\bar{j}}] \\
&\lesseqgtr \sum_{\mathbf{x}_{\bar{j}}} \mathbf{P}[A' \,|\, \mathbf{p}_{\bar{j}} = \mathbf{x}_{\bar{j}}]\, \mathbf{P}[B' \,|\, \mathbf{p}_{\bar{j}} = \mathbf{x}_{\bar{j}}]\, \mathbf{P}[\mathbf{p}_{\bar{j}} = \mathbf{x}_{\bar{j}}] \\
&= \mathbf{P}[A'] \sum_{\mathbf{x}_{\bar{j}}} \mathbf{P}[B' \,|\, \mathbf{p}_{\bar{j}} = \mathbf{x}_{\bar{j}}]\, \mathbf{P}[\mathbf{p}_{\bar{j}} = \mathbf{x}_{\bar{j}}] \;=\; \mathbf{P}[A']\, \mathbf{P}[B'] \,.
\end{aligned}$$

The above reasoning clearly holds for the oblivious adversary. Observe, however, that it also holds in the adaptive case: The event A' only depends on the random outcome ε_j of job j, which the adaptive adversary cannot control. In principle, the event B' might be influenced by a change in the processing time of j. However, since p_j is increased in both cases, this change is revealed to the adversary only after the completion of j itself. So, up to time t, the behavior of the adaptive adversary will be the same.

5. TOPOLOGY MATTERS: SMOOTHED COMPETITIVENESS OF METRICAL TASK SYSTEMS

Abstract

We consider *metrical task systems*, a general framework to model online problems. An online algorithm resides in a graph G of n nodes and may move in this graph at a cost equal to the distance. The algorithm has to service a sequence of *tasks* that arrive online; each task specifies for each node a *request cost* that is incurred if the algorithm services the task in this particular node. The objective is to minimize the total request cost plus the total travel cost. A deterministic online algorithm for metrical task systems is the *work function algorithm* (WFA), which has an optimal competitive ratio of $2n - 1$.

In this chapter, we present a smoothed competitive analysis of WFA. Given an adversarial task sequence, we smoothen the request costs by means of a symmetric additive smoothing model and analyze the competitive ratio of WFA on the smoothed task sequence. Our analysis reveals that the smoothed competitive ratio of WFA is much better than $O(n)$ and that it depends on several topological parameters of the underlying graph G, such as the minimum edge length $U_{\min}$, the maximum degree D, and the edge diameter $diam$. For example, supposed that the ratio between the maximum and the minimum edge length of G is bounded by a constant, WFA has smoothed competitive ratio $O(diam(U_{\min}/\sigma + \log(D)))$ and $O(\sqrt{n(U_{\min}/\sigma + \log(D))})$, where σ denotes the standard deviation of the smoothing distribution. That is, already for perturbations with $\sigma = \Theta(U_{\min})$ the competitive ratio reduces to $O(\log(n))$ on a clique and to $O(\sqrt{n})$ on a line. We also prove that for a large class of graphs these bounds are asymptotically tight. Furthermore, we provide lower bounds for arbitrary graphs. We obtain a better bound of $O(\beta(U_{\min}/\sigma + \log(D)))$ on the smoothed competitive ratio of WFA if each adversarial task contains at most β non-zero entries. We also provide the first average case analysis of WFA. We prove that WFA has $O(\log(D))$ expected competitive ratio if the request costs are chosen randomly from an arbitrary non-increasing distribution with standard deviation $\sigma = \Theta(U_{\min})$.

Publication Notes. This chapter is joint work with Naveen Sivadasan. An extended abstract will appear in the Conference Proceedings of the Twenty-First International Symposium on Theoretical Aspects of Computer Science (STACS 2004) [SS04]. A complete version of the paper was published as a MPII research report [SS03].

Naveen Sivadasan is a Ph. D. student at the Max-Planck-Institut für Informatik at Saarbrücken. The results presented in this chapter will also become part of his thesis. My own contribution to the contents of this chapter is 50%.

5.1 Introduction

Borodin, Linial, and Saks [BLS92] introduced a general framework, which is commonly known as *metrical task systems*, to model online problems. Many online problems can be formulated as metrical task systems; for example, the paging problem, the static list accessing problem, and the k-server problem. Due to its generality, the competitive ratio of an algorithm for metrical task systems is usually weak compared to the one of an online algorithm that is designed for a particular problem, such as the k-server problem. However, precisely because of its generality we believe that it is interesting to analyze WFA.

Metrical task systems are formulated as follows. We are given an undirected and connected graph $G := (V, E)$ with node set $V := \{v_1, \ldots, v_n\}$ and edge set E, and a positive length function $\lambda : E \to \mathbb{R}^+$ on the edges of G. We extend λ to a metric δ on G. Let $\delta : V \times V \to \mathbb{R}^+$ be a distance function such that $\delta(u, v)$ denotes the shortest path distance (with respect to λ) between any two nodes u and v in G. A *task* τ is an n-vector $(r(v_1), \ldots, r(v_n))$ of *request costs*. The cost to process task τ in node v_i is $r(v_i) \in \mathbb{R}^+ \cup \{\infty\}$. The online algorithm starts from a given initial position $s_0 \in V$ and has to service a sequence $\mathcal{S} := \langle \tau_1, \ldots, \tau_r \rangle$ of tasks, arriving one at a time. If the online algorithm resides after task τ_{t-1} in node u, the cost to service task τ_t in node v is $\delta(u, v) + r_t(v)$; $\delta(u, v)$ is the *transition cost* and $r_t(v)$ is the *processing cost*. The objective is to minimize the total transition plus processing cost.

Borodin, Linial, and Saks [BLS92] gave a deterministic online algorithm, known as the *work function algorithm* (WFA), for metrical task systems. WFA has a competitive ratio of $2n - 1$, which is optimal. Borodin, Linial, and Saks [BLS92] and Manasse, McGeoch, and Sleator [MMS88] proved that *every* deterministic online algorithm for metrical task systems has competitive ratio at least $2n - 1$.

We use the notion of smoothed competitiveness to characterize the asymptotic performance of WFA. We smoothen the request costs of each task according to an additive symmetric smoothing model. Each cost entry is smoothed by adding a random number chosen from a symmetric probability distribution f with mean zero. Our analysis holds for various probability distributions, including the uniform and the normal distribution. We use σ to refer to the standard deviation of f. Our analysis reveals that the smoothed competitive ratio of WFA is much better than its worst case competitive ratio suggests and that it depends on certain *topological parameters* of the underlying graph.

Definition of Topological Parameters: Throughout this chapter, we assume that the underlying graph G has n nodes, minimum edge length $U_{\min}$, maximum edge length $U_{\max}$, and maximum degree D. Furthermore, we use $Diam$ to refer to the *diameter* of G, i.e., the maximum length of a shortest path between any two nodes. Similarly, a graph has *edge* diameter $diam$ if any two nodes are connected by a path of at most $diam$ edges. Observe that $diam U_{\min} \leq Diam \leq diam U_{\max}$. We emphasize that these topological parameters are defined with respect to G and its length function λ—not with respect to the resulting metric.

Upper Bounds			
random tasks	$O\big(\frac{\sigma}{U_{\min}}\big(\frac{U_{\min}}{\sigma}+\log(D)\big)\big)$		
arbitrary tasks	$O\big(\frac{Diam}{U_{\min}}\big(\frac{U_{\min}}{\sigma}+\log(D)\big)\big)$	and	$O\Big(\sqrt{n\cdot\frac{U_{\max}}{U_{\min}}\big(\frac{U_{\min}}{\sigma}+\log(D)\big)}\Big)$
β**-elementary tasks**	$O\big(\beta\cdot\frac{U_{\max}}{U_{\min}}\big(\frac{U_{\min}}{\sigma}+\log(D)\big)\big)$		

Table 5.1: Upper bounds on the smoothed competitive ratio of WFA.

We prove several upper bounds; see Table 5.1.

1. We show that if the request costs are chosen randomly from a distribution f, which is non-increasing in $[0,\infty)$, the expected competitive ratio of WFA is

$$O\big(1+\tfrac{\sigma}{U_{\min}}\cdot\log(D)\big).$$

   In particular, WFA has an expected competitive ratio of $O(\log(D))$ if $\sigma=\Theta(U_{\min})$. For example, we obtain an expected competitive ratio of $O(\log(n))$ on a clique and of $O(1)$ on a binary tree.

2. We prove two upper bounds on the smoothed competitive ratio of WFA:

$$O\Big(\tfrac{Diam}{U_{\min}}\big(\tfrac{U_{\min}}{\sigma}+\log(D)\big)\Big)\quad\text{and}\quad O\Big(\sqrt{n\cdot\tfrac{U_{\max}}{U_{\min}}\big(\tfrac{U_{\min}}{\sigma}+\log(D)\big)}\Big).$$

   For example, if $\sigma=\Theta(U_{\min})$ and $U_{\max}/U_{\min}=\Theta(1)$, WFA has smoothed competitive ratio $O(\log(D))$ on any graph with constant edge diameter and $O(\sqrt{n})$ on any graph with constant maximum degree. Note that we obtain an $O(\log(n))$ bound on a complete binary tree.

3. We obtain a better upper bound on the smoothed competitive ratio of WFA if the adversarial task sequence only consists of β*-elementary tasks*. A task is β-elementary if it has at most β non-zero entries. (We will use the term *elementary task* to refer to a 1-elementary task.) We prove a smoothed competitive ratio of

$$O\big(\beta\cdot\tfrac{U_{\max}}{U_{\min}}\big(\tfrac{U_{\min}}{\sigma}+\log(D)\big)\big).$$

   For example, if $\sigma=\Theta(U_{\min})$ and $U_{\max}/U_{\min}=\Theta(1)$, WFA has smoothed competitive ratio $O(\beta\log(D))$ for β-elementary tasks.

We also present lower bounds; see Table 5.2. All our lower bounds hold for *any* deterministic online algorithm and if the request costs are smoothed according to the additive symmetric

Lower Bounds			
arbitrary tasks			
– existential	$\Omega\big(\frac{Diam}{U_{\min}}\big(\frac{U_{\min}}{\sigma}+\log(D)\big)\big)$	and	$\Omega\Bigg(\sqrt{n\cdot\frac{U_{\max}}{U_{\min}}\big(\frac{U_{\min}}{\sigma}+\log(D)\big)}\Bigg)$
– universal	$\Omega\big(\frac{U_{\min}}{\sigma}+\frac{U_{\min}}{U_{\max}}\log(D)\big)$	and	$\Omega\Bigg(\sqrt{diam\cdot\frac{U_{\min}}{U_{\max}}\big(\frac{U_{\min}}{\sigma}+1\big)}\Bigg)$
β-**elementary tasks**	$\Omega\big(\beta\cdot\big(\frac{U_{\min}}{\sigma}+1\big)\big)$	(existential)	

Table 5.2: Lower bounds on the smoothed competitive ratio of any deterministic online algorithm.

smoothing model. We distinguish between *existential* and *universal* lower bounds. An existential lower bound, say $\Omega(f(n))$, means that there *exists* a class of graphs such that *every* deterministic algorithm has smoothed competitive ratio $\Omega(f(n))$ on these graphs. On the other hand, a universal lower bound $\Omega(f(n))$ states that for *any arbitrary* graph, *every* deterministic algorithm has smoothed competitive ratio $\Omega(f(n))$. Clearly, for metrical task systems, the best lower bound we can hope to obtain is $\Omega(n)$. Therefore, if we state a lower bound of $\Omega(f(n))$, we actually mean $\Omega(\min(n, f(n)))$.

4. For a large range of values for $Diam$ and D, we present existential lower bounds that are asymptotically tight to the upper bounds stated in 2. This means (a) that the stated smoothed competitive ratio of WFA is asymptotically tight, and (b) that WFA is asymptotically optimal under the additive smoothing model—no other deterministic algorithm can achieve a better smoothed competitive ratio.

5. We also prove two universal lower bounds on the smoothed competitive ratio:

$$\Omega\big(\tfrac{U_{\min}}{\sigma}+\tfrac{U_{\min}}{U_{\max}}\log(D)\big) \quad \text{and} \quad \Omega\Bigg(\min\Bigg(diam, \sqrt{diam\cdot\tfrac{U_{\min}}{U_{\max}}\big(\tfrac{U_{\min}}{\sigma}+1\big)}\Bigg)\Bigg).$$

   Suppose that $U_{\max}/U_{\min} = \Theta(1)$. Then the first bound matches the first upper bound stated in 2 if the edge diameter $diam$ is constant, e.g., for a clique. The second bound matches the second upper bound in 2 if $diam = \Omega(n)$ and the maximum degree D is constant, e.g., for a line.

6. For β-elementary tasks we prove an existential lower bound of

$$\Omega\big(\beta\cdot\big(\tfrac{U_{\min}}{\sigma}+1\big)\big).$$

   This implies that the bound in 3 is tight up to a factor of $(U_{\max}/U_{\min})\log(D)$.

Constrained Balls into Bins Game. Our analysis crucially relies on a lower bound on the cost of an optimal offline algorithm. We therefore study the growth of the work function values on

a sequence of random requests. It turns out that the increase in the work function values can be modeled by a version of a balls into bins game with dependencies between the heights of the bins, which are specified by a constraint graph. We call this game the *constrained balls into bins game*. The dependencies between the heights of the bins make it difficult to analyze this stochastic process. We believe that the constrained balls into bins game is also interesting independently of the context of this work.

Organization of this Chapter. In Section 5.2 we first review the work function algorithm and state some of its properties. In Section 5.3 we define the smoothing model that we use. The lower bound on the cost of an optimal offline algorithm and the related balls into bins game are presented in Section 5.4. Then, in Section 5.5 and Section 5.6, we prove the upper bounds on the smoothed competitive ratio of WFA. After that, in Section 5.7, we present upper bounds for random and β-elementary tasks. Finally, in Section 5.8, we prove existential and universal lower bounds. We give some concluding remarks in Section 5.9.

5.2 Work Function Algorithm

Let $\mathcal{S} = \langle \tau_1, \ldots, \tau_\ell \rangle$ be a request sequence, and let $s_0 \in V$ denote the initial position of the online algorithm. Let $\mathcal{S}_t$ denote the subsequence of the first t tasks of $\mathcal{S}$. For each t, $0 \leq t \leq \ell$, we define a function $w_t : V \to \mathbb{R}$ such that for each node $u \in V$, $w_t(u)$ is the minimum offline cost to process $\mathcal{S}_t$ starting in s_0 and ending in u. The function w_t is called the *work function* at time t with respect to $\mathcal{S}$ and s_0.

Let OPT denote an optimal offline algorithm. Clearly, the optimal offline cost $\text{OPT}[\mathcal{S}]$ on $\mathcal{S}$ is equal to the minimum work function value at time ℓ, i.e., $\text{OPT}[\mathcal{S}] = \min_{u \in V} w_\ell(u)$. We can compute $w_t(u)$ for each $u \in V$ by dynamic programming:

$$w_0(u) := \delta(s_0, u) \quad \text{and} \quad w_t(u) := \min_{v \in V}(w_{t-1}(v) + r_t(v) + \delta(v, u)) \quad \text{for } t \geq 1. \tag{5.1}$$

We next describe the online work function algorithm; see also [BLS92, BEY98]. Intuitively, a good strategy for an online algorithm to process task τ_t is to move to a node where OPT would reside if τ_t would be the final task. However, the competitive ratio of an algorithm that solely sticks to this policy can become arbitrarily bad. A slight modification gives a $2n - 1$ competitive algorithm: Instead of blindly (no matter at what cost) traveling to the node of minimum work function value, we additionally take the transition cost into account. Essentially, this is the idea underlying the work function algorithm.

Work Function Algorithm (WFA): Let $s_0, \ldots, s_{t-1}$ denote the sequence of nodes visited by WFA to process $\mathcal{S}_{t-1}$. Then, to process task τ_t, WFA moves to a node s_t that minimizes $w_t(v) + \delta(s_{t-1}, v)$ for all $v \in V$. There is always a choice for s_t such that in addition

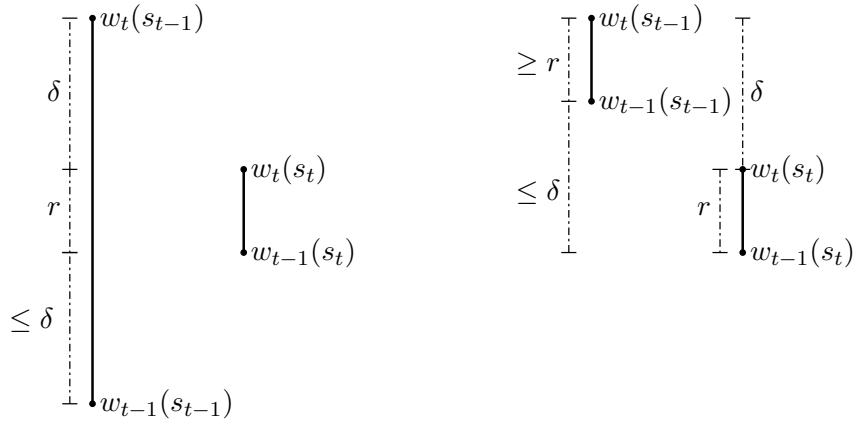

Figure 5.1: Illustration of facts. Let $r := r_t(s_t)$ and $\delta := \delta(s_{t-1}, s_t)$.

$w_t(s_t) = w_{t-1}(s_t) + r_t(s_t)$. More formally,

$$s_t := \arg\min_{v \in V}(w_t(v) + \delta(s_{t-1}, v)) \quad \text{such that} \quad w_t(s_t) = w_{t-1}(s_t) + r_t(s_t). \tag{5.2}$$

Subsequently, we use WFA and OPT, respectively, to denote the work function and the optimal offline algorithm. For a given sequence $\mathcal{S} = \langle \tau_1, \ldots, \tau_\ell \rangle$ of tasks, WFA$[\mathcal{S}]$ and OPT$[\mathcal{S}]$ refer to the cost incurred by WFA and OPT on $\mathcal{S}$, respectively. By $s_0, \ldots, s_\ell$ we denote the sequence of nodes visited by WFA.

We continue by observing a few properties of work functions and of the online algorithm WFA; see also Figure 5.1. The corresponding proofs are given in Appendix 5.A.

Fact 5.2.1. *For any node u and any time t, $w_t(u) \geq w_{t-1}(u)$.*

Fact 5.2.2. *For any node u and any time t, $w_t(u) \leq w_{t-1}(u) + r_t(u)$.*

Fact 5.2.3. *For any two nodes u and v and any time t, $|w_t(u) - w_t(v)| \leq \delta(u, v)$.*

Fact 5.2.4. *At any time t, $w_t(s_t) = w_t(s_{t-1}) - \delta(s_{t-1}, s_t)$.*

Fact 5.2.5. *At any time t, $r_t(s_t) + \delta(s_{t-1}, s_t) = w_t(s_{t-1}) - w_{t-1}(s_t)$.*

5.3 Smoothing Model

Let the *adversarial task sequence* be given by $\check{\mathcal{S}} := \langle \check{\tau}_1, \ldots, \check{\tau}_r \rangle$. We smoothen each task vector $\check{\tau}_t := (\check{r}_t(v_1), \ldots, \check{r}_t(v_n))$ by perturbing each *original cost* entry $\check{r}_t(v_j)$ according to some probability distribution f as follows

$$r_t(v_j) := \max(0, \check{r}_t(v_j) + \varepsilon(v_j)), \quad \text{where } \varepsilon(v_j) \leftarrow f.$$

That is, to each original cost entry we add a random number which is chosen from f. The obtained *smoothed* task is denoted by $\tau_t := (r_t(v_1), \ldots, r_t(v_n))$. We use μ and σ, respectively, to denote the expectation and the standard deviation of f. We assume that f is symmetric around $\mu := 0$. We take the maximum of zero and the smoothing outcome in order to assure that the smoothed costs are non-negative. Observe that the probability for an original zero cost entry to remain zero is amplified to $\frac{1}{2}$.

A major criticism to the additive model is that zero cost entries are destroyed. However, as we will argue in the next subsection, one can easily verify that the lower bound proof of $2n-1$ on the competitive ratio of any deterministic algorithm for metrical task systems goes through for any smoothing model that does not destroy zeros.

Our analysis holds for a large class of probability distributions, which we call *permissible*. We say f is permissible if (i) f is symmetric around $\mu = 0$ and (ii) f is non-increasing in $[0, \infty)$. For example, the uniform and the normal distribution are permissible. The concentration of f around μ is given by its standard deviation σ. Since the stated upper bounds on the smoothed competitive ratio of WFA do not further improve by choosing σ much larger than $U_{\min}$, we assume that $\sigma \leq 2U_{\min}$. Moreover, we use c_f to denote a constant depending on f such that for a random ε chosen from f, $\mathbf{P}[\varepsilon \geq \sigma/c_f] \geq \frac{1}{4}$.

All our results hold against an *adaptive adversary*. An adaptive adversary reveals the task sequence over time, thereby taking decisions made by the online algorithm in the past into account.

5.3.1 Lower Bound for Zero-Retaining Smoothing Models

The proof of the $2n-1$ lower bound on the competitive ratio of any deterministic algorithm, see [BLS92, MMS88, BEY98], is based only on the use of elementary tasks and the fact that the cost of the online algorithm is monotone increasing with the length of the input sequence. Assume we consider a zero-retaining smoothing model, i.e., a model in which zero cost entries are invariant to the smoothing. In such a model, elementary tasks are smoothed to elementary tasks. In particular this means that the above two properties still hold. Therefore, the lower bound proof also goes through for sequences that are smoothed according to any zero-retaining smoothing model.

Theorem 5.3.1. *Every deterministic online algorithm* ALG *for metrical task systems has a smoothed competitive ratio of at least* $2n-1$ *under a zero-retaining smoothing model.*

5.4 A Lower Bound on the Optimal Offline Cost

In this section, we establish a lower bound on the cost incurred by an optimal offline algorithm OPT when run on tasks smoothed according to the additive smoothing model. For the purpose of proving the lower bound, we first investigate an interesting version of a balls into bins experiment, which we call the *constrained balls into bins game*.

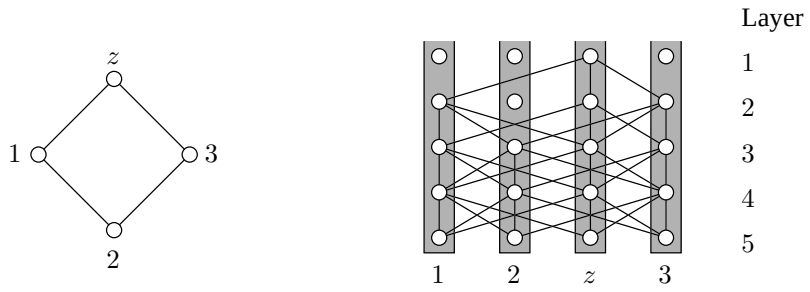

Figure 5.2: Illustration of the "unfolding" for $Q = 1$ and $h = 5$. Left: Constraint graph G_c. Right: Layered dependency graph $\mathcal{D}_h$.

5.4.1 Constrained Balls into Bins Game

We are given n bins $B_1, \ldots, B_n$. In each round, we place a ball independently in each bin B_i with probability p; with probability $1-p$ no ball is placed in B_i. We define the *height* $h_t(i)$ of a bin B_i as the number of balls in B_i after round t. We have dependencies between the heights of different bins that are specified by an (undirected) *constraint graph* $G_c := (V_c, E_c)$. The node set V_c of G_c contains n nodes $u_1, \ldots, u_n$, where each node u_i corresponds to a bin B_i. All edges in E_c have uniform edge lengths equal to Q. Let D be the maximum degree of a vertex in G_c. Throughout the experiment, we maintain the following invariant.

Invariant: The difference in height between two bins B_i and B_j is at most the shortest path distance between u_i and u_j in G_c.

If the placement of a ball into a bin B_i would violate this invariant, the ball is *rejected*; otherwise we say that the ball is *accepted*. Observe that if two bins B_i and B_j do not violate the invariant in round t then, in round $t+1$, B_i and B_j might cause a violation only if one bin, say B_i, receives a ball, and the other, B_j, does not receive a ball; if both receive a ball or both do not receive a ball, the invariant remains true.

Theorem 5.4.1. *Fix any bin B_z. Let R_z be the number of rounds needed until the height of B_z becomes $h \geq \log(n)$. Then, $\mathbf{P}[R_z > c_3 h\,(1 + \log(D)/Q)] \leq 1/n^4$.*

We remark that constraint graphs with $Q = 1$ exist, e.g., a clique on n nodes, such that the expected number of rounds needed for the height of a bin to become h is $\Omega(h \log(n))$. Moreover, for any given maximum degree D one can create graph instances with $Q = 1$ such that the expected number of rounds is $\Omega(h \log(D))$.

We next describe how one can model the growth of the height of B_z by an alternative game on a *layered dependency graph*. A layered dependency graph $\mathcal{D}_h$ consists of h layers, $V_1, \ldots, V_h$, and edges are present only between adjacent layers. The idea is to "unfold" the constraint graph G_c into a layered dependency graph $\mathcal{D}_h$.

We first describe the construction for $Q = 1$: Each layer of $\mathcal{D}_h$ corresponds to a subset of nodes in G_c. Layer 1 consists of z only, the node corresponding to bin B_z. Assume we have constructed layers $V_1, \ldots, V_i$, $i < h$. Then V_{i+1} is constructed from V_i by adding all nodes, $\Gamma_{G_c}(V_i)$, that are adjacent to V_i in G_c, i.e., $V_{i+1} := V_i \cup \Gamma_{G_c}(V_i)$. For every pair $(u, v) \in V_i \times V_{i+1}$, we add an edge (u, v) to $\mathcal{D}_h$ if $(u, v) \in E_c$, or $u = v$. See Figure 5.2 for an example.

Now, we consider the following game on $\mathcal{D}_h$. Each node in $\mathcal{D}_h$ is in one of three states, namely UNFINISHED, READY, or FINISHED. Initially, all nodes in layer h are READY and all other nodes are UNFINISHED. In each round, all READY nodes independently toss a coin; each coin turns up *head* with probability p and *tail* with probability $1 - p$. A READY node changes its state to FINISHED if the outcome of its coin toss is *head*. At the end of each round, an UNFINISHED node in layer j changes its state to READY if all its neighbors in layer $j + 1$ are FINISHED.

Note that the nodes in layer V_j are FINISHED if the corresponding bins B_i, $i \in V_j$, have height at least j. Consequently, the number of rounds needed until the root node z in $\mathcal{D}_h$ becomes FINISHED dominates the number of rounds needed for the height of B_z to become h.

We use a similar construction if $Q > 1$. For simplicity, let h be a multiple of Q and define $h' = h/Q$. We construct a dependency graph $\mathcal{D}_{h'}$ with h' layers as described above (replace h by h' in the description above). Then we transform $\mathcal{D}_{h'}$ into a layered graph $\mathcal{D}_h$ with h layers as follows. Let v be a node in layer j of $\mathcal{D}_{h'}$. We replace v by a path $(v_1, \ldots, v_k)$, where $k = |Q|$. Node v_1 is connected to all neighbors of v in layer $j - 1$ and node v_k is connected to all neighbors of v in layer $j + 1$. This replacement makes sure that the number of rounds needed until the root node becomes FINISHED in $\mathcal{D}_h$ dominates the number of rounds needed for the height of B_z to become h.

Proof of Theorem 5.4.1 . Let $\mathcal{D}_h$ be a layered dependency graph constructed from G_c as described above. As argued above, the event $(R_z \leq t)$ is stochastically dominated by the event that the root node becomes FINISHED within t rounds in $\mathcal{D}_h$. Consider the event that the root node z does not become FINISHED after t rounds. Then there exists a *bad* path $P := (v_1, \ldots, v_h)$ from $z = v_1$ to some node v_h in the bottom layer h such that no node v_i of P was delayed by nodes other than $v_{i+1}, \ldots, v_h$. Put differently, P was delayed independently of any other path. Consider the outcome of the coin flips only for the nodes along P. If P is bad then the number of coin flips, denoted by X, that turned up *head* within t rounds is at most $h - 1$. Let $\alpha(t)$ denote the probability that P is bad, i.e., $\alpha(t) := \mathbf{P}[X \leq h - 1]$. Clearly, $\mathbf{E}[X] = tp$.

Observe that in $\mathcal{D}_h$ (i) at most h' layers contain nodes of degree larger than 2 and (ii) these nodes have at most $D + 1$ neighbors in the next larger layer. That is, the number of possible paths from z to any node v in layer h is bounded by $(D + 1)^{h'}$.

Thus, $\mathbf{P}[R_z > t] \leq \alpha(t)(D + 1)^{h'}$. We want to choose t such that this probability is at most $1/n^4$. If we choose $t \geq (32/p)(h + h' \log(D))$ and use Chernoff's bound (see

Theorem 2.4.10) on X, we obtain for $h \geq \log(n)$

$$\alpha(t) = \mathbf{P}[X \leq h-1] \leq \mathbf{P}[X \leq pt/2] \leq e^{-pt/8} \leq \frac{1}{n^4(D+1)^{h'}}.$$

□

5.4.2 Lower Bound

We are now in a position to prove that an optimal offline algorithm incurs with high probability a cost of at least $n\gamma U_{\min}$ on a sequence of $\Theta(n\gamma\,(U_{\min}/\sigma + \log(D)))$ tasks.

Lemma 5.4.1. *Let $\check{\mathcal{S}}$ be an adversarial sequence of $\ell := \lceil c_2 n\gamma(U_{\min}/\sigma + \log(D))\rceil$ tasks, for a fixed constant c_2 and some $\gamma \geq 1$. Then, $\mathbf{P}[\text{OPT}[\mathcal{S}] < n\gamma U_{\min}] \leq 1/n^3$.*

We will use Lemma 5.4.1 several times as follows.

Corollary 5.4.1. *Let $\check{\mathcal{S}}$ be an adversarial sequence of $\ell := \lceil c_2 n\gamma(U_{\min}/\sigma + \log(D))\rceil$ tasks for a fixed constant c_2 and an some $\gamma \geq 1$. Then the smoothed competitive ratio of* WFA *is at most $\mathbf{E}[\text{WFA}[\mathcal{S}]]/(n\gamma U_{\min}) + o(1)$.*

Proof. Let $\mathcal{S}$ be a random variable denoting a smoothed sequence obtained from $\check{\mathcal{S}}$. We define $\mathcal{E}$ as the event that OPT incurs a cost of at least $n\gamma U_{\min}$ on $\mathcal{S}$. By Lemma 5.4.1, $\mathbf{P}[\neg\mathcal{E}] \leq 1/n^3$. Thus

$$\begin{aligned}\mathbf{E}\left[\frac{\text{WFA}[\mathcal{S}]}{\text{OPT}[\mathcal{S}]}\right] &= \mathbf{E}\left[\frac{\text{WFA}[\mathcal{S}]}{\text{OPT}[\mathcal{S}]}\,\middle|\,\mathcal{E}\right]\,\mathbf{P}[\mathcal{E}] \;+\; \mathbf{E}\left[\frac{\text{WFA}[\mathcal{S}]}{\text{OPT}[\mathcal{S}]}\,\middle|\,\neg\mathcal{E}\right]\mathbf{P}[\neg\mathcal{E}] \\ &\leq \frac{\mathbf{E}[\text{WFA}[\mathcal{S}]\,|\,\mathcal{E}]\,\mathbf{P}[\mathcal{E}]}{n\gamma U_{\min}} + \frac{2n-1}{n^3} \;\leq\; \frac{\mathbf{E}[\text{WFA}[\mathcal{S}]]}{n\gamma U_{\min}} + o(1),\end{aligned}$$

where the second inequality follows from the definition of $\mathcal{E}$ and the fact that the (worst case) competitive ratio of WFA is $2n-1$. □

Proof of Lemma 5.4.1 . The cost of OPT on a smoothed sequence $\mathcal{S}$ of length ℓ is $\text{OPT}[\mathcal{S}] = \min_{u\in V} w_\ell(u)$. Therefore, it suffices to prove that with probability at least $1 - 1/n^3$, $w_\ell(u) \geq n\gamma U_{\min}$ for each $u \in V$. Observe that we can assume that the initial work function values are all set to zero, since this can only reduce the cost of OPT.

We relate the growth of the work function values to the balls and bins experiment. For each node v_i of G we have a corresponding bin B_i. The constraint graph G_c is obtained from G by setting all edge lengths to $Q := \lfloor U_{\min}/\Delta\rfloor$, where $\Delta := \min(U_{\min}, \sigma/c_f)$. The placement of a ball in B_i in round t corresponds to the event $(r_t(v_i) \geq \sigma/c_f)$. Since our smoothing distribution satisfies $\mathbf{P}[\varepsilon \geq \sigma/c_f] \geq \frac{1}{4}$, we have that for any v_i and any t the smoothed request cost $r_t(v_i)$ is at least σ/c_f with probability at least $\frac{1}{4}$, irrespectively of its original cost

entry and independently of the other request costs. Therefore, in each round t we place a ball into each bin with probability $p = \frac{1}{4}$.

By Lemma 5.4.2 given below, the number of rounds needed until a bin B_i has height $h \geq \log(n)$ stochastically dominates the time t needed until $w_t(v_i) \geq h\Delta$. Applying Theorem 5.4.1, we obtain that for any bin B_i, after $\ell \geq c_3 h(1 + \log(D)/Q)$ rounds, $\mathbf{P}[h_\ell(i) < h] \leq 1/n^4$. Consequently, after ℓ rounds with probability at least $1 - 1/n^3$ all bins have height at least h. By choosing $h := 2n\gamma Q$, this implies that at time ℓ with probability at least $1 - 1/n^3$, $w_\ell(v_i) \geq 2n\gamma Q\Delta \geq n\gamma U_{\min}$ for all v_i of G. Finally, we make sure that $\ell := \lceil c_2 n\gamma(U_{\min}/\sigma + \log(D))\rceil \geq c_3 h(1 + \log(D)/Q)$ by fixing $c_2 := 4c_3\lceil c_f\rceil$. □

Lemma 5.4.2. *Consider any node v_i and its corresponding bin B_i. Let $h_t(i)$ denote the number of balls in bin B_i after t rounds. Then, for any $t \geq 0$, $w_t(v_i) \geq h_t(i)\,\Delta$.*

Proof. We proof the lemma by induction on the number of rounds t. For $t = 0$, the lemma clearly holds. (We can assume that the initial work function values are all zero.) Assume that the induction hypothesis holds after t rounds. In round $t + 1$, if no ball is accepted in any bin then clearly the hypothesis remains true. Consider the case where at least one ball is accepted by some bin B_i. By the induction hypothesis, we have $w_t(v_i) \geq h_t(i)\Delta$. Let v_k be the node that determines the work function value $w_{t+1}(v_i)$, i.e.,

$$w_{t+1}(v_i) = w_t(v_k) + r_{t+1}(v_k) + \delta(v_i, v_k).$$

Assume that $v_k = v_i$. Then the work function value of v_i increases by the request cost $r_{t+1}(v_i)$, and since a ball was accepted in B_i, $r_{t+1}(v_i) \geq \Delta$. Thus, we have $w_{t+1}(v_i) \geq w_t(v_i) + \Delta \geq (h_t(i) + 1)\Delta = h_{t+1}(i)\Delta$, and we are done.

Next, assume that $v_k \neq v_i$. Let d be the shortest path distance between v_i and v_k in the constraint graph. Since in round $t + 1$ a ball was accepted in B_i, B_i and B_k do not violate the invariant. Therefore,

$$h_t(i) - h_t(k) \leq d - 1 + [\text{ball accepted in } B_k \text{ in round } t + 1],$$

where "[*statement*]" is 1 if *statement* is true, and 0 otherwise. By multiplying both sides with Δ and rearranging terms, we obtain

$$(h_t(k) + d)\Delta \geq (h_t(i) + 1 - [\text{ball accepted in } B_k \text{ in round } t + 1])\Delta.$$

Observe that $d\Delta \leq \delta(v_i, v_k)$ by the definition of d and the edge lengths Q of the constraint graph. Moreover, $r_{t+1}(v_k) \geq [\text{ball accepted in } B_k \text{ in round } t + 1]\Delta$. Thus,

$$\begin{aligned} w_{t+1}(v_i) &= w_t(v_k) + r_{t+1}(v_k) + \delta(v_i, v_k) \\ &\geq h_t(k)\Delta + [\text{ball accepted in } B_k \text{ in round } t + 1]\Delta + d\Delta \\ &\geq (h_t(i) + 1)\Delta = h_{t+1}(i)\Delta. \end{aligned}$$

□

5.5 First Upper Bound

We can use the lower bound obtained in the last section to derive our first upper bound on the smoothed competitive ratio of WFA. We prove the following deterministic bound on the cost of WFA.

Lemma 5.5.1. *Let $\mathcal{K}$ be any task sequence of length ℓ. Then,* $\text{WFA}[\mathcal{K}] \leq \text{OPT}[\mathcal{K}] + Diam \cdot \ell$.

Proof. Let $s_0, \ldots, s_\ell$ denote the sequence of nodes visited by WFA. For any t, the cost incurred by WFA to process task t is $C(t) := r_t(s_t) + \delta(s_{t-1}, s_t)$. By Fact 5.2.5, we have $C(t) = w_t(s_{t-1}) - w_{t-1}(s_t)$. Hence,

$$\begin{aligned}\text{WFA}[\mathcal{K}] = \sum_{t=1}^{\ell} C(t) \;&=\; w_\ell(s_{\ell-1}) - w_0(s_1) + \sum_{t=1}^{\ell-1} w_t(s_{t-1}) - w_t(s_{t+1}) \\ &\leq w_\ell(s_{\ell-1}) + (\ell-1)\cdot Diam \;\leq\; \min_{v\in V} w_\ell(v) + \ell \cdot Diam,\end{aligned}$$

where the last two inequalities follow from Fact 5.2.3. Since $\text{OPT}[\mathcal{K}] \geq \min_{v\in V} w_\ell(v)$, the lemma follows. □

Theorem 5.5.1. *The smoothed competitive ratio of* WFA *is*

$$O\left(\frac{Diam}{\sigma} + \frac{Diam}{U_{\min}} \cdot \log(D)\right).$$

Proof. Consider an adversarial task sequence $\check{\mathcal{S}}$ of length $\ell := \lceil c_2 n\gamma\,(U_{\min}/\sigma + \log(D))\rceil$ for some $\gamma \geq 1$. Let $\mathcal{S}$ be a random variable denoting a smoothed sequence obtained from $\check{\mathcal{S}}$. Due to the proof of Corrollary 5.4.1 it suffices to bound $\mathbf{E}[\text{WFA}[\mathcal{S}]/\text{OPT}[\mathcal{S}] \mid \mathcal{E}]$, where $\mathcal{E}$ is the event $(\text{OPT}[\mathcal{S}] \geq n\gamma U_{\min})$. Using Lemma 5.5.1, we have for any sequence $\mathcal{K}$ of ℓ tasks, $\text{WFA}[\mathcal{K}] \leq \text{OPT}[\mathcal{K}] + Diam \cdot \ell$. Thus,

$$\begin{aligned}\mathbf{E}\left[\frac{\text{WFA}[\mathcal{S}]}{\text{OPT}[\mathcal{S}]}\,\middle|\,\mathcal{E}\right] \leq \mathbf{E}\left[\frac{\text{OPT}[\mathcal{S}] + Diam\cdot\ell}{\text{OPT}[\mathcal{S}]}\,\middle|\,\mathcal{E}\right] \;&\leq\; 1 + \frac{Diam\cdot\ell}{n\gamma U_{\min}} \\ &= O\left(\frac{Diam}{U_{\min}}\left(\frac{U_{\min}}{\sigma} + \log(D)\right)\right),\end{aligned}$$

where the last equality follows from the definition of ℓ. □

5.6 Second Upper Bound

We prove a second upper bound on the smoothed competitive ratio of WFA. The idea is as follows. We derive two upper bounds on the smoothed competitive ratio of WFA. The first one is a deterministic bound, and the second one uses the probabilistic lower bound on OPT. We then combine these two bounds using the following fact. The proof of Fact 5.6.1 can be found in Appendix 5.A.

Fact 5.6.1. *Let A, B, and X_i, $1 \leq i \leq m$, be positive quantities. We have*

$$\min\left(\frac{A\sum_{i=1}^{m} X_i}{\sum_{i=1}^{m} X_i^2}, \frac{B\sum_{i=1}^{m} X_i}{m}\right) \leq \sqrt{A\,B}.$$

Consider any deterministic task sequence $\mathcal{K}$ of length ℓ. Let $s_0, s_1, \ldots, s_\ell$ denote the sequence of nodes visited by WFA. Define $C(t) := r_t(s_t) + \delta(s_{t-1}, s_t)$ as the service cost plus the transition cost incurred by WFA in round t.

With respect to $\mathcal{K}$ we define T as the set of rounds, where the increase of the work function value of s_{t-1} is at least one half of the transition cost, i.e., $t \in T$ if and only if $w_t(s_{t-1}) - w_{t-1}(s_{t-1}) \geq \delta(s_{t-1}, s_t)/2$. Due to Fact 5.2.4 we have $w_t(s_{t-1}) = w_t(s_t) + \delta(s_{t-1}, s_t)$. Therefore, the above definition is equivalent to

$$T := \left\{t : w_t(s_t) - w_{t-1}(s_{t-1}) \geq -\frac{1}{2}\delta(s_{t-1}, s_t)\right\}. \tag{5.3}$$

We use $\bar{T}$ to denote the complement of T.

We first prove that the total cost of WFA on $\mathcal{K}$ is bounded by a constant times the total cost contributed by rounds in T.

Lemma 5.6.1. *Let $\mathcal{K}$ be a sufficiently long task sequence such that* $\text{WFA}[\mathcal{K}] \geq 6Diam$. *Then,* $\text{WFA}[\mathcal{K}] \leq 8\sum_{t\in T} C(t)$.

Proof. We have $w_\ell(s_\ell) - w_0(s_0) \geq -Diam$, since $w_0(s_0) \leq w_\ell(s_0)$ and due to Fact 5.2.3. Thus,

$$\sum_{t=1}^{\ell}(w_t(s_t) - w_{t-1}(s_{t-1})) \geq -Diam.$$

Let R^- be the set of rounds where $w_t(s_t) - w_{t-1}(s_{t-1}) < 0$, and let R^+ be the set of rounds where $w_t(s_t) - w_{t-1}(s_{t-1}) \geq 0$. The above inequality can be rewritten as

$$\sum_{t\in R^-}(w_{t-1}(s_{t-1}) - w_t(s_t)) \leq Diam + \sum_{t\in R^+}(w_t(s_t) - w_{t-1}(s_{t-1})).$$

Since $\bar{T} \subseteq R^-$ and each term on the left hand side is non-negative, we have

$$\sum_{t\in\bar{T}}(w_{t-1}(s_{t-1}) - w_t(s_t)) \leq Diam + \sum_{t\in R^+}(w_t(s_t) - w_{t-1}(s_{t-1})). \tag{5.4}$$

For any $t \in \bar{T}$ we have $C(t) < 3\,(w_{t-1}(s_{t-1}) - w_t(s_t))$. This can be seen as follows. We have $w_{t-1}(s_t) \geq w_{t-1}(s_{t-1}) - \delta(s_{t-1}, s_t)$ (by Fact 5.2.3) and $r_t(s_t) = w_t(s_t) - w_{t-1}(s_t)$ (by (5.2)). Therefore, $r_t(s_t) \leq \delta(s_{t-1}, s_t) - w_{t-1}(s_{t-1}) + w_t(s_t)$. Moreover, since $t \in \bar{T}$ and by the definition (5.3) of T, $\delta(s_{t-1}, s_t) < 2(w_{t-1}(s_{t-1}) - w_t(s_t))$. Hence, $C(t) = r_t(s_t) + \delta(s_{t-1}, s_t) < 3\,(w_{t-1}(s_{t-1}) - w_t(s_t))$.

Furthermore, for any t we have $w_t(s_t) - w_{t-1}(s_{t-1}) \leq C(t)$. This follows from $w_t(s_t) = w_{t-1}(s_t) + r_t(s_t)$ (by (5.2)) and $w_{t-1}(s_t) - w_{t-1}(s_{t-1}) \leq \delta(s_{t-1}, s_t)$ (by Fact 5.2.3). Since $R^+ \subseteq T$, we conclude

$$\sum_{t\in R^+}(w_t(s_t) - w_{t-1}(s_{t-1}) \leq \sum_{t\in R^+} C(t) \leq \sum_{t\in T} C(t).$$

Therefore, (5.4) implies

$$\frac{1}{3}\sum_{t\in\bar{T}} C(t) \leq Diam + \sum_{t\in T} C(t).$$

Exploiting the fact that $\text{WFA}[\mathcal{K}] = \sum_{t\in\bar{T}} C(t) + \sum_{t\in T} C(t)$ and $\text{WFA}[\mathcal{K}] \geq 6Diam$, we obtain $\text{WFA}[\mathcal{K}] \leq 8\sum_{t\in T} C(t)$. □

We partition T into T^1 and T^2, where $T^1 := \{t \in T : w_t(s_t) - w_{t-1}(s_t) \leq 4U_{\max} diam\}$, and $T^2 := T \setminus T^1$. For any round t, we define $W_t := \sum_{i=1}^n w_t(v_i)$ and $\Delta W_t := W_t - W_{t-1}$.

Lemma 5.6.2. *Fix a round t and consider any node u such that $w_t(u) - w_{t-1}(u) \geq H$. If $H \leq 4U_{\max} diam$ then $\Delta W_t \geq H^2/(10U_{\max})$; otherwise, $\Delta W_t \geq nH/2$.*

Proof. Let $H \leq 4U_{\max} diam$. Define $d := \lfloor H/(8U_{\max})\rfloor$. For $d = 0$ the claim clearly holds. Assume $d > 0$. Consider a shortest path $P := (u_0, u_1, \ldots, u_d)$ of edge length d starting from $u_0 := u$. Since $d \leq \lfloor diam/2\rfloor$, there always exists a shortest path of length d. (Consider a breadth-first search tree rooted at u_0; the depth of this tree is at least $\lceil diam/2\rceil$.) By Fact 5.2.3, we have for each i, $0 \leq i \leq d$, $w_t(u_i) \geq w_t(u_0) - iU_{\max}$ and $w_{t-1}(u_i) \leq w_{t-1}(u_0) + iU_{\max}$.

Therefore,

$$\begin{aligned}\sum_{i=0}^{d}(w_t(u_i) - w_{t-1}(u_i)) &\geq \sum_{i=0}^{d}(w_t(u_0) - w_{t-1}(u_0)) - 2U_{\max}\sum_{i=1}^{d} i \\ &\geq (d+1)H - (d+1)dU_{\max} \geq (d+1)(H - dU_{\max}) \geq \frac{H^2}{10U_{\max}},\end{aligned}$$

where the last inequality holds since $d \leq H/(8U_{\max}) \leq d+1$.

Let $H > 4U_{\max} diam$. Since for any node v_i, $w_{t-1}(v_i) \leq w_{t-1}(u) + U_{\max} diam$ and $w_t(v_i) \geq w_t(u) - U_{\max} diam$, we have

$$\sum_{i=1}^{n} (w_t(v_i) - w_{t-1}(v_i)) \geq nH - 2nU_{\max} diam \geq nH/2.$$

□

Lemma 5.6.3. *Let $\mathcal{K}$ be a sufficiently long task sequence such that $\text{OPT}[\mathcal{K}] \geq 2Diam$. There exists a constant b such that*

$$\text{OPT}[\mathcal{K}] \geq \frac{1}{bn}\left(\frac{1}{U_{\max}}\sum_{t\in T^1} C(t)^2 + n\sum_{t\in T^2} C(t)\right).$$

Proof. For every node v_i, $w_\ell(v_i) \leq \min_{u\in V} w_\ell(u) + Diam$ (by Fact 5.2.3). Moreover, $\text{OPT}[\mathcal{K}] \geq \min_{u\in V} w_\ell(u)$. We obtain

$$\sum_{i=1}^{n} w_\ell(v_i) \leq n\text{OPT}[\mathcal{K}] + nDiam \quad \text{or, equivalently,} \quad \text{OPT}[\mathcal{K}] \geq \frac{1}{n}\left(\sum_{i=1}^{n} w_\ell(v_i) - nDiam\right).$$

Since $\text{OPT}[\mathcal{K}] \geq 2Diam$, the latter reduces to

$$\text{OPT}[\mathcal{K}] \geq \frac{2}{3n}\sum_{i=1}^{n} w_\ell(v_i). \tag{5.5}$$

Claim 5.6.1. *For any $t \in T^1$, $\Delta W_t \geq C(t)^2/(160U_{\max})$.*

Proof. By (5.2) we have $r_t(s_t) = w_t(s_t) - w_{t-1}(s_t)$. Below, we will show that

$$\Delta W_t \geq \left(\delta(s_{t-1}, s_t)^2 + r_t(s_t)^2\right)/(80U_{\max}). \tag{5.6}$$

Since $C(t)^2 = (\delta(s_{t-1}, s_t) + r_t(s_t))^2 \leq 2(\delta(s_{t-1}, s_t)^2 + r_t(s_t)^2)$, we conclude that $\Delta W_t \geq C(t)^2/(160U_{\max})$. Now, all that remains to be shown is (5.6). We distinguish two cases.

Let $\delta(s_{t-1}, s_t) \geq r_t(s_t)$. By the definition of T, we have $w_t(s_{t-1}) - w_{t-1}(s_{t-1}) \geq \delta(s_{t-1}, s_t)/2$. Using Lemma 5.6.2 with $H := \delta(s_{t-1}, s_t)/2$, we obtain

$$\Delta W_t \geq \delta(s_{t-1}, s_t)^2/(40U_{\max}) \geq \left(\delta(s_{t-1}, s_t)^2 + r_t(s_t)^2\right)/(80U_{\max}).$$

Let $\delta(s_{t-1}, s_t) < r_t(s_t)$. Since $w_t(s_t) - w_{t-1}(s_t) = r_t(s_t)$ and $r_t(s_t) \leq 4U_{\max} diam$ by the definition of T_1, using Lemma 5.6.2 with $H := r_t(s_t)$, we obtain

$$\Delta W_t \geq r_t(s_t)^2/(10U_{\max}) \geq (\delta(s_{t-1}, s_t)^2 + r_t(s_t)^2)/(20U_{\max}).$$

□

Claim 5.6.2. *For any $t \in T^2$, $\Delta W_t \geq 4nC(t)/10$.*

Proof. Since $t \in T^2$ and by (5.2), $r_t(s_t)/4 > diam U_{\max} \geq \delta(s_{t-1}, s_t)$. Thus, $C(t) = r_t(s_t) + \delta(s_{t-1}, s_t) < 5r_t(s_t)/4$. Furthermore, by (5.2) we have $r_t(s_t) = w_t(s_t) - w_{t-1}(s_t)$. Applying Lemma 5.6.2 with $H := r_t(s_t)$, we obtain $\Delta W_t \geq nr_t(s_t)/2 \geq 4nC(t)/10$. □

Claim 5.6.1 and Claim 5.6.2 together imply that

$$\sum_{i=1}^{n} w_\ell(v_i) \geq \sum_{t=1}^{\ell} \Delta W_t \geq \sum_{t \in T} \Delta W_t \geq \frac{1}{160 U_{\max}} \sum_{t \in T^1} C(t)^2 + \frac{4n}{10} \sum_{t \in T^2} C(t).$$

The proof now follows for an appropriate constant b from (5.5). □

Theorem 5.6.1. *The smoothed competitive ratio of* WFA *is*

$$O\left(\sqrt{n \cdot \frac{U_{\max}}{U_{\min}} \left(\frac{U_{\min}}{\sigma} + \log(D)\right)}\right).$$

Proof. Consider an adversarial task sequence $\check{\mathcal{S}}$ of length $\ell := \lceil c_2 n\gamma(U_{\min}/\sigma + \log(D)) \rceil$, for an appropriate γ, and let $\mathcal{S}$ be a random variable denoting a smoothed sequence obtained from $\check{\mathcal{S}}$. Due to the proof of Corrollary 5.4.1 it suffices to bound $\mathbf{E}[\text{WFA}[\mathcal{S}]/\text{OPT}[\mathcal{S}] \mid \mathcal{E}]$, where $\mathcal{E}$ is the event $(\text{OPT}[\mathcal{S}] \geq n\gamma U_{\min})$. Consider a smoothing outcome $\mathcal{S}$ such that the event $\mathcal{E}$ holds. We fix γ sufficiently large such that $\text{OPT}[\mathcal{S}] \geq 6Diam$. Observe that $\text{WFA}[\mathcal{S}] \geq \text{OPT}[\mathcal{S}] \geq 6Diam$.

First, assume $\sum_{t \in T^1} C(t) < \sum_{t \in T^2} C(t)$. Then, due to Lemma 5.6.1 and Lemma 5.6.3,

$$\text{WFA}[\mathcal{S}] \leq 16 \sum_{t \in T^2} C(t) \quad \text{and} \quad \text{OPT}[\mathcal{S}] \geq \frac{1}{b} \sum_{t \in T^2} C(t).$$

Hence, $\mathbf{E}[\text{WFA}[\mathcal{S}]/\text{OPT}[\mathcal{S}] \mid \mathcal{E}] = O(1)$.

Next, assume $\sum_{t \in T^1} C(t) \geq \sum_{t \in T^2} C(t)$. By Lemma 5.6.1 and Lemma 5.6.3 we have

$$\text{WFA}[\mathcal{S}] \leq 16 \sum_{t \in T^1} C(t) \quad \text{and} \quad \text{OPT}[\mathcal{S}] \geq \frac{1}{bn}\left(\frac{1}{U_{\max}} \sum_{t \in T^1} C(t)^2\right). \tag{5.7}$$

Thus,

$$\frac{\text{WFA}[\mathcal{S}]}{\text{OPT}[\mathcal{S}]} \leq 16bnU_{\max}\left(\frac{\sum_{t \in T^1} C(t)}{\sum_{t \in T^1} C(t)^2}\right). \tag{5.8}$$

Since $\mathcal{E}$ holds, we also have

$$\frac{\text{WFA}[\mathcal{S}]}{\text{OPT}[\mathcal{S}]} \leq \frac{\ell \cdot 16 \sum_{t \in T^1} C(t)}{\ell \cdot n\gamma U_{\min}} \leq \frac{c}{U_{\min}} \left(\frac{U_{\min}}{\sigma} + \log(D) \right) \left(\frac{\sum_{t \in T^1} C(t)}{|T^1|} \right), \tag{5.9}$$

where the latter inequality holds for an appropriate constant c and since $\ell \geq |T^1|$. Observe that (5.9) is well-defined since $\sum_{t \in T^1} C(t) \geq \frac{1}{16}\text{WFA}[\mathcal{S}]$ (by (5.7)) and $\text{WFA}[\mathcal{S}] \geq 6Diam$ imply that $|T^1| \geq 1$.

Applying Fact 5.6.1 to (5.8) and (5.9), these two bounds are combined to

$$\frac{\text{WFA}[\mathcal{S}]}{\text{OPT}[\mathcal{S}]} \leq \sqrt{16bcn \cdot \frac{U_{\max}}{U_{\min}} \left(\frac{U_{\min}}{\sigma} + \log(D) \right)} = O\left(\sqrt{n \cdot \frac{U_{\max}}{U_{\min}} \left(\frac{U_{\min}}{\sigma} + \log(D) \right)} \right),$$

which concludes the proof. □

5.7 Better Bounds for Random and β-Elementary Tasks

We obtain better bounds for random and β-elementary tasks. Both bounds exploit the following potential function argument.

5.7.1 Potential Function

In this section we use a potential function argument to derive an upper bound on the expected cost of WFA.

Lemma 5.7.1. *Let $\check{\mathcal{S}}$ be an adversarial task sequence of length ℓ, and let $\mathcal{S} = \langle \tau_1, \ldots, \tau_\ell \rangle$ be a smoothed sequence obtained from $\check{\mathcal{S}}$. For a given node s and a time t, $1 \leq t \leq \ell$, define a random variable $\Delta_t(s) := \min_{u \in V}(r_t(u) + \delta(u, s))$. Let $\kappa > 0$. If $\mathbf{E}[\Delta_t(s)] \leq \kappa$ for each $s \in V$ and for each t, $1 \leq t \leq \ell$, then $\mathbf{E}[\text{WFA}[\mathcal{S}]] \leq 4\kappa\ell + Diam$.*

Before we proceed to prove Lemma 5.7.1, we provide some intuition. Assume we consider a simple greedy online algorithm ALG that always moves to a node which minimizes the transition plus request cost. That is, ALG services task τ_t by moving from its current position, say s'_{t-1}, to a node s'_t that minimizes the expression $\min_{u \in V}(r_t(u) + \delta(u, s'_{t-1}))$. Clearly, if the requirement of Lemma 5.7.1 holds, the total expected cost of ALG on $\mathcal{S}$ is $\sum_{t=1}^{\ell} \mathbf{E}[\Delta_t(s_{t-1})] \leq \ell\kappa$. The above lemma shows that the expected cost of the work function algorithm WFA is at most 4 times the expected cost of the greedy algorithm ALG plus some additive term. In the analysis, it will sometimes be convenient to consider ALG instead of WFA.

Proof of Lemma 5.7.1 . For $1 \leq t \leq \ell$, we denote by s_t the node in which WFA resides after task τ_t has been processed; we use s_0 to refer to the node in which WFA resides initially.

We define a potential function Φ as

$$\Phi(t) := w_t(s_t) + tDiam/\ell.$$

Observe that

$$\Phi(\ell) - \Phi(0) = w_\ell(s_\ell) - w_0(s_0) + Diam \geq w_\ell(s_\ell) - w_\ell(s_0) + Diam \geq 0,$$

where the last inequality follows from Fact 5.2.3 and since $\delta(s_\ell, s_0) \leq Diam$.

We define the *amortized cost* $C_a(t)$ incurred by WFA to process task τ_t as

$$\begin{aligned} C_a(t) &:= r_t(s_t) + \delta(s_{t-1}, s_t) + \Phi(t) - \Phi(t-1) \\ &= r_t(s_t) + \delta(s_{t-1}, s_t) + w_t(s_t) - w_{t-1}(s_{t-1}) + Diam/\ell \\ &= w_t(s_t) - w_{t-1}(s_t) + w_t(s_{t-1}) - w_{t-1}(s_{t-1}) + Diam/\ell, \end{aligned} \tag{5.10}$$

where the last equality follows from Fact 5.2.5. Using Fact 5.2.3 and (5.1) we obtain that for each $u \in V$

$$w_{t-1}(s_t) \geq w_{t-1}(u) - \delta(u, s_t) \quad \text{and} \quad w_t(s_t) \leq w_{t-1}(u) + r_t(u) + \delta(u, s_t).$$

Combining these two inequalities, we obtain

$$\begin{aligned} & w_t(s_t) - w_{t-1}(s_t) \leq r_t(u) + 2\delta(u, s_t) \quad \text{for each } u \in V, \\ \text{and hence} \quad & w_t(s_t) - w_{t-1}(s_t) \leq 2 \min_{u \in V} (r_t(u) + \delta(u, s_t)) \;=\; 2\Delta_t(s_t). \end{aligned}$$

A similar argument shows that $w_t(s_{t-1}) - w_{t-1}(s_{t-1}) \leq 2\Delta_t(s_{t-1})$. Hence, we can rewrite (5.10) as

$$C_a(t) \;\leq\; 2\Delta_t(s_t) + 2\Delta_t(s_{t-1}) + Diam/\ell.$$

Since $\text{WFA}[\mathcal{S}] = \sum_{t=1}^{\ell} C_a(t) - \Phi(\ell) + \Phi(0)$ and $\Phi(\ell) - \Phi(0) \geq 0$, we obtain

$$\mathbf{E}[\text{WFA}[\mathcal{S}]] \leq \mathbf{E}\left[\sum_{t=1}^{\ell} C_a(t)\right] \leq 2\mathbf{E}\left[\sum_{t=1}^{\ell} (\Delta_t(s_t) + \Delta_t(s_{t-1}))\right] + Diam \leq 4\kappa\ell + Diam.$$

□

If $\ell \geq Diam$ then the above bound reduces to $O(\kappa\ell)$. Corrollary 5.4.1 together with the upper bound of Lemma 5.7.1 yield the following corollary.

Corollary 5.7.1. *Let $\check{S}$ be an adversarial sequence of $\ell := \lceil c_2 n\gamma(U_{\min}/\sigma + \log(D))\rceil$ tasks for a fixed constant c_2. If $\gamma \geq U_{\max}$, and therefore $\ell \geq Diam$, the smoothed competitive ratio*

of WFA *is*

$$O\left(\frac{\kappa\ell}{n\gamma U_{\min}}\right) = O\left(\kappa\left(\frac{1}{\sigma} + \frac{\log(D)}{U_{\min}}\right)\right).$$

5.7.2 Random Tasks

We derive an upper bound on the expected competitive ratio of WFA if each request cost is chosen independently from a probability distribution f which is non-increasing in $[0, \infty)$.

We need the following fact; the proof is given in Appendix 5.A.

Fact 5.7.1. *Let f be a continuous, non-increasing distribution over $[0, \infty)$ with mean μ and standard deviation σ. Then, $\mu \leq \sqrt{12}\sigma$.*

Theorem 5.7.1. *If each request cost is chosen independently from a non-increasing probability distribution f over $[0, \infty)$ with standard deviation σ then the expected competitive ratio of* WFA *is*

$$O\left(1 + \frac{\sigma}{U_{\min}}\log(D)\right).$$

Proof. Let $\mathcal{S}$ be a random task sequence of length $\ell := \lceil c_2 n\gamma(U_{\min}/\sigma) + \log(D))\rceil$, for an appropriate $\gamma \geq U_{\max}$, generated from f. Observe that since $\gamma \geq U_{\max}$, we have $\ell \geq Diam$. For any t and any node s, we have

$$\Delta_t(s) = \min_{u \in V}(r_t(u) + \delta(u, s)) \leq r_t(s).$$

Since $r_t(s)$ is chosen from f, Fact 5.7.1 implies that $\mathbf{E}[\Delta_t(s)] \leq \kappa := \sqrt{12}\sigma$. Thus, by Lemma 5.7.1, we have $\mathbf{E}[\text{WFA}[\mathcal{S}]] = 4\sqrt{12}\sigma\ell + Diam = O(\sigma\ell)$.

Note that we can use the lower bound established in Section 5.4 to bound the cost of OPT: The generation of $\mathcal{S}$ is equivalent to smoothing (according to f) an adversarial task sequence consisting of all-zero request vectors only. Here, we do not need that the distribution f is symmetric around its mean. The theorem now follows from Corrollary 5.7.1. □

5.7.3 β-Elementary Tasks

We can strengthen the upper bound on the smoothed competitive ratio of WFA if the adversarial task sequence only consists of β-elementary tasks. Recall that in a β-elementary task the number of non-zero request costs is at most β.

Theorem 5.7.2. *If the adversarial task sequence only consists of β-elementary tasks then the smoothed competitive ratio of* WFA *is*

$$O\left(\beta \cdot \frac{U_{\max}}{U_{\min}}\left(\frac{U_{\min}}{\sigma} + \log(D)\right)\right).$$

We state the following fact; the proof is given in Appendix 5.A.

Fact 5.7.2. *Let f be a permissible probability distribution. Then, $\mathbf{E}[\max(0,\varepsilon)] \leq \sigma$, where ε is a random variable chosen from f.*

We first prove the following lemma.

Lemma 5.7.2. *Let s be an arbitrary node of G. Consider a β-elementary adversarial task $\check{\tau}_t := (\check{r}_t(v_1), \ldots, \check{r}_t(v_n))$, where $\beta < n$. Then, $\mathbf{E}[\Delta_t(s)] \leq \sigma + \beta U_{\max}$.*

Proof. Let $V_0 \subseteq V$ be the set of all nodes with original cost zero, i.e., $V_0 := \{u \in V : \check{r}_t(u) = 0\}$. Then, $|V_0| \geq n - \beta$, and V_0 is non-empty if $\beta < n$. Let v^* be a node from V_0 which is closest to s. We have $\delta(v^*, s) \leq \beta U_{\max}$. (Otherwise, there must exist at least $\beta + 1$ nodes with non-zero original cost, a contradiction.) Thus,

$$\mathbf{E}[\Delta_t(s)] \leq \mathbf{E}[\min_{u \in V_0}(r_t(u) + \delta(u,s))] \leq \mathbf{E}[r_t(v^*) + \delta(v^*, s)] \leq \sigma + \beta U_{\max},$$

where the last inequality follows since $r_t(v^*) = \max(0, \varepsilon(v^*))$, $\varepsilon(v^*)$ is a random variable chosen from f, and Fact 5.7.2. □

Proof of Theorem 5.7.2 . Consider an adversarial task sequence $\check{\mathcal{S}}$ of length $\ell := \lceil c_2 n\gamma(U_{\min}/\sigma + \log(D))\rceil$, for an appropriate $\gamma \geq U_{\max}$, and let $\mathcal{S}$ be a random variable denoting a smoothed sequence obtained from $\check{\mathcal{S}}$. By Lemma 5.7.2, $\mathbf{E}[\Delta_t(s)] \leq \kappa := \sigma + \beta U_{\max}$, which, since we assume that $\sigma \leq 2U_{\min}$, is $O(\beta U_{\max})$. The theorem now follows from Corollary 5.7.1. □

5.8 Lower Bounds

In this section we present existential and universal lower bounds. All our lower bounds hold for any deterministic online algorithm ALG and against an adaptive adversary.

5.8.1 Existential Lower Bound for β-Elementary Tasks

We show an existential lower bound for β-elementary tasks on a line. We prove that the upper bound $O(\beta(U_{\max}/U_{\min})(U_{\min}/\sigma + \log(D)))$ established in Theorem 5.7.2 is tight up to a factor of $U_{\max}/U_{\min}$ if the underlying graph is a line. Later, we will use Theorem 5.8.1 to obtain our first universal lower bound.

Theorem 5.8.1. *Let G be a line graph. If the adversarial task sequence only consists of β-elementary tasks then the smoothed competitive ratio of any deterministic online algorithm* ALG *is*

$$\Omega\left(\min\left(\beta \cdot \left(\frac{U_{\min}}{\sigma} + 1\right), \frac{n}{\beta} \cdot \frac{U_{\min}}{U_{\max}}\right)\right).$$

Proof. We use an averaging technique (see [BLS92]). Divide the line into $h := n/(2\beta)$ contiguous segments of 2β nodes. For simplicity assume that h is an integer. (This does not affect the asymptotic lower bound.) We refer to these segments by $S_1, S_2, \ldots, S_h$.

Let s_t be the node in which ALG resides after the tth task. In round t, the adversary issues a β-elementary task by placing ∞ cost on each node that is within distance $\lceil \beta/2 \rceil - 1$ from s_{t-1} and zero cost on all other nodes. Let the random variable $\mathcal{S}$ denote a smoothed task sequence.

We consider a set $\mathbf{B}$ of h offline algorithms, one for each segment. Let $\mathbf{B}_j$ denote the offline algorithm that resides in segment S_j; $\mathbf{B}_j$ always stays in S_j. In each round t, each $\mathbf{B}_j$ moves to a node v in S_j minimizing the transition cost plus the request cost. Define $\mathbf{B}[\mathcal{S}] := \sum_{j=1}^{h} \mathbf{B}_j[\mathcal{S}]$ as the total cost incurred by the offline algorithms on $\mathcal{S}$; $\mathbf{B}_j[\mathcal{S}]$ is a random variable denoting the total cost incurred by $\mathbf{B}_j$ on $\mathcal{S}$. Clearly, $\tilde{\mathbf{B}}[\mathcal{S}] := \mathbf{B}[\mathcal{S}]/h$ is an upper bound on $\text{OPT}[\mathcal{S}]$.

Consider any round t. At most two consecutive line segments can have ∞ request costs. Moreover, in each segment at most β of the 2β nodes may have ∞ costs. Let $C_j(t)$ be the cost incurred by $\mathbf{B}_j$ in round t. Consider a segment S_j that receives a ∞ request cost. Then, $\mathbf{E}[C_j(t)] \leq \beta U_{\max} + \sigma$ by Lemma 5.7.2. Assume S_j does not receive any ∞ request cost. Then, $\mathbf{E}[C_j(t)] \leq \sigma$ by Fact 5.7.2.

Since in any round at most two segments may receive ∞ costs, we conclude

$$\mathbf{E}[\tilde{\mathbf{B}}[\mathcal{S}]] = \frac{1}{h}\mathbf{E}\left[\sum_{j=1}^{h} \mathbf{B}_j[\mathcal{S}]\right] = \frac{1}{h}\mathbf{E}\left[\sum_{j=1}^{h}\sum_{t=1}^{\ell} C_j(t)\right] \leq \ell\left(\frac{2(\beta U_{\max} + \sigma)}{h} + \sigma\right).$$

By Markov's inequality, $\mathbf{P}[\tilde{\mathbf{B}}[\mathcal{S}] < 2\mathbf{E}[\tilde{\mathbf{B}}[\mathcal{S}]]] \geq \frac{1}{2}$. Since in each round, ALG is forced to travel at least a distance of $\lceil \beta/2 \rceil$, we have $\text{ALG}[\mathcal{S}] \geq \ell\beta U_{\min}/2$.

We conclude

$$\mathbf{E}\left[\frac{\text{ALG}[\mathcal{S}]}{\text{OPT}[\mathcal{S}]}\right] \geq \left(\frac{1}{2}\right)\frac{\ell\beta U_{\min}/2}{2\ell\left(\frac{2(\beta U_{\max}+\sigma)}{h} + \sigma\right)} = \Omega\left(\frac{\beta U_{\min}}{\beta^2 U_{\max}/n + \sigma}\right).$$

That is, we obtain a lower bound of $\Omega((n/\beta)\cdot(U_{\min}/U_{\max}))$ if $\beta \geq \sqrt{n/(U_{\max}/\sigma)}$ and of $\Omega(\beta\cdot(U_{\min}/\sigma))$ if $\beta \leq \sqrt{n/(U_{\max}/\sigma)}$. In the latter case, exploiting that $\sigma \leq 2U_{\min}$, we obtain an $\Omega(\beta\cdot(U_{\min}/\sigma + 1))$ bound. □

Observe that on a line the β-elementary bound of Theorem 5.7.2 is stronger than the general upper bound of Theorem 5.6.1 only if

$$\beta \leq \sqrt{\frac{nU_{\min}}{U_{\max}(U_{\min}/\sigma + 1)}}.$$

In this case, Theorem 5.8.1 provides a lower bound of $\Omega(\beta\cdot(U_{\min}/\sigma + 1))$. That is, for a line graph these bounds differ by a factor of at most $U_{\max}/U_{\min}$.

5.8.2 Universal Lower Bounds

We derive two universal lower bounds on the smoothed competitive ratio of any deterministic algorithm. The first universal bound uses the following corollary of Theorem 5.8.1.

Corollary 5.8.1. *Let G be a line graph. Any deterministic algorithm* ALG *has smoothed competitive ratio $\Omega(\min(n, \sqrt{n(U_{\min}/U_{\max})(U_{\min}/\sigma+1)}))$ against an adaptive adversary.*

Proof. Fix $\beta := \sqrt{nU_{\min}/(U_{\max}(U_{\min}/\sigma+1))}$ and use the lower bound given in Theorem 5.8.1. □

Theorem 5.8.2. Any *deterministic algorithm* ALG *has a smoothed competitive ratio of*

$$\Omega\left(\min\left(diam,\ \sqrt{diam \cdot \frac{U_{\min}}{U_{\max}} \cdot \left(\frac{U_{\min}}{\sigma}+1\right)}\right)\right).$$

Proof. We extend Theorem 5.8.1 to arbitrary graphs in a straightforward way. Consider a path in G of edge length $diam$. The adversary enforces that ALG and OPT never leave this path by specifying ∞ cost for each node that is not part of the path. The desired lower bound now follows from Corrollary 5.8.1. □

Next, we prove the following universal lower bound.

Theorem 5.8.3. *Any deterministic algorithm* ALG *has a smoothed competitive ratio of*

$$\Omega\left(\min\left(n, \frac{U_{\min}}{\sigma} + \frac{U_{\min}}{U_{\max}} \cdot \log(D)\right)\right).$$

Proof. The adversary issues a sequence of ℓ tasks as described below. For each t, $1 \le t \le \ell$, let s_t denote the node at which the deterministic online algorithm ALG resides after the tth task; we use s_0 to refer to the initial position of ALG.

We prove two different lower bounds. Combining these two lower bounds, we obtain the bound stated above.

We first obtain a lower bound of $\Omega(\min(n, U_{\min}/\sigma))$ assuming that $U_{\min}/\sigma \ge 1$. In round t, the adversary enforces a request cost of $U_{\min}$ on s_{t-1} and zero request cost on all other nodes. Recall that the adversary is adaptive and therefore knows the position of ALG.

We use an averaging technique to relate the cost of ALG to the average cost of a collection of offline algorithms. Let $\mathbf{B}$ be a collection of n offline algorithms. We place one offline algorithm at each node, and each offline algorithm remains at its node during the processing of the task sequence. Let $\mathcal{S}$ be a random variable denoting a smoothing outcome of $\check{\mathcal{S}}$. We define $\mathbf{B}[\mathcal{S}]$ as the total cost incurred by the n algorithms to process $\mathcal{S}$. Clearly, the average cost $\tilde{\mathbf{B}}[\mathcal{S}] := \mathbf{B}[\mathcal{S}]/n$ is an upper bound on OPT$[\mathcal{S}]$. It suffices to prove that with constant probability ALG$[\mathcal{S}]/\tilde{\mathbf{B}}[\mathcal{S}] = \Omega(\min(n, U_{\min}/\sigma))$.

For the analysis, we view the smoothing process as being done into two stages.

Stage 1: Initially, we smoothen ℓ zero tasks (all request costs are zero) according to the given smoothing distribution. Let the smoothed sequence be $\mathcal{S}' := \langle \tau'_1, \ldots, \tau'_\ell \rangle$.

Stage 2: For each t, $1 \leq t \leq \ell$, we replace the request cost of s_{t-1} in τ'_t by the outcome of smoothing $U_{\min}$. We use τ_t to refer to the obtained task.

Let $\mathbf{R}'(v) := \sum_{t=1}^{\ell} r'_t(v)$ be the total request cost accumulated in v with respect to $\mathcal{S}'$. Moreover, we define ℓ random variables $U_1, \ldots, U_\ell$: U_t refers to the smoothed request cost $r_t(s_{t-1})$ of task τ_t obtained in Stage 2. For each $1 \leq t \leq \ell$, let Z_t be a $0/1$ random variable which is 1 if and only if $U_t \geq U_{\min}$. We define $Z := \sum_{t=1}^{\ell} Z_t$. Subsequently, we condition the smoothing outcome $\mathcal{S}$ on the following three events: (i) $\mathcal{E} := (\sum_{v \in V} \mathbf{R}'(v) \leq 2n\ell\sigma)$, (ii) $\mathcal{F} := (\sum_{t=1}^{\ell} U_t \leq 4\ell U_{\min})$, and (iii) $\mathcal{G} := (Z \geq \ell/4)$.

We first argue that the event $(\mathcal{E} \cap \mathcal{F} \cap \mathcal{G})$ occurs with at least constant probability. (i) Due to Fact 5.7.2, $\mathbf{E}[\mathbf{R}'(v)] \leq \ell\sigma$ for each $v \in V$. By Markov's inequality, we thus have $\mathbf{P}[\mathcal{E}] \geq 1/2$. (ii) By Fact 5.7.2 and since $\sigma \leq U_{\min}$, we also have $\mathbf{E}[U_t] \leq U_{\min} + \sigma \leq 2U_{\min}$ for each $1 \leq t \leq \ell$. Hence by Markov's inequality, $\mathbf{P}[\sum_{t=1}^{\ell} U_t \geq 4\ell U_{\min}] \leq 1/2$. (iii) Since the smoothing distribution f is a symmetric, we have $\mathbf{P}[U_t \geq U_{\min}] \geq 1/2$ for each $1 \leq t \leq \ell$. Thus, $\mathbf{E}[Z_t] \geq 1/2$. Moreover, the Z_t's are independent. Applying Chernoff's bound (see Theorem 2.4.10), we obtain $\mathbf{P}[Z \leq \ell/4] \leq e^{-\ell/16}$.

Since event $\mathcal{E}$ is defined with respect to $\mathcal{S}'$, it is independent of the event $(\mathcal{F} \cap \mathcal{G})$. Therefore,

$$\mathbf{P}[\mathcal{E} \cap \mathcal{F} \cap \mathcal{G}] \geq \frac{1}{2} \cdot \left(1 - \left(\frac{1}{2} + e^{-\ell/16}\right)\right) \geq \frac{1}{8},$$

where the last inequality holds if $\ell \geq 64$.

Let $\mathcal{S}$ be any fixed outcome of the smoothing such that $(\mathcal{E} \cap \mathcal{F} \cap \mathcal{G})$ holds. Assume that to process sequence $\mathcal{S}$, ALG changes its position in k of the ℓ rounds. Let T_k refer to the set of rounds where ALG changes its position. We bound the cost of the offline algorithms as follows. In any round t, the total cost incurred by the offline algorithms at nodes different from s_{t-1} is at most $\sum_{v \in V} r'_t(v)$. If ALG does not move in round t, both ALG and $\mathbf{B}$ incur a cost of U_t. If ALG moves in round t, $\mathbf{B}$ incurs an additional cost of U_t, since one algorithm resides in s_{t-1}. Thus,

$$\mathbf{B}[\mathcal{S}] \leq \text{ALG}[\mathcal{S}] + \sum_{t \in T_k} U_t + \sum_{v \in V} \mathbf{R}'(v) \leq \text{ALG}[\mathcal{S}] + 4\ell U_{\min} + 2n\ell\sigma,$$

where the last inequality follows from $\mathcal{F}$ and $\mathcal{E}$.

Since also $\mathcal{G}$ holds, we can conclude that ALG incurs a cost of at least $\ell U_{\min}/4$: In each of the at least $\ell/4$ rounds, we have $r_t(s_{t-1}) = U_t \geq U_{\min}$. That is, no matter whether ALG moves or stays in these rounds, it incurs a cost of at least $U_{\min}$.

Thus, conditioned on the event $(\mathcal{E} \cap \mathcal{F} \cap \mathcal{G})$ we obtain for an appropriate constant c

$$\frac{\text{ALG}[\mathcal{S}]}{\tilde{\mathbf{B}}[\mathcal{S}]} \geq \frac{\text{ALG}[\mathcal{S}]}{17\text{ALG}[\mathcal{S}]/n + 2\ell\sigma} \geq c \cdot \min\left(n, \frac{U_{\min}}{\sigma}\right).$$

Next we obtain a lower bound of $\Omega((U_{\min}/U_{\max})\log(D))$. Consider a node s of G with degree D. Let V_s be the set of nodes containing s and all the neighbors of s in G. Define G_s as the subgraph of G induced by V_s. The adversary makes sure that every reasonable online algorithm will always reside at a node in V_s by specifying in each round a request cost of ∞ for each $v \notin V_s$. In addition, in each round t the adversary enforces the online algorithm to move by placing a request cost of ∞ at s_{t-1}. All other request cost are zero.

Let $\mathcal{S}$ be a smoothed task sequence obtained from $\check{\mathcal{S}}$. Since G_s is a star with $D+1$ nodes and the transition cost between any two nodes is at most $2U_{\max}$, Lemma 5.8.1 implies that there exists a deterministic offline algorithm $\mathbf{B}$ with $\mathbf{E}[\mathbf{B}[\mathcal{S}]] \leq 2c\ell U_{\max}/\log(D)$. (Observe that we can apply Lemma 5.8.1 here since with respect to G_s the request sequence is elementary.) Applying Markov's inequality, we obtain $\mathbf{P}[\mathbf{B}[\mathcal{S}] \geq 4c\ell U_{\max}/\log(D)] \leq 1/2$. Since ALG has to move in each round to avoid ∞ cost, the cost of ALG for any smoothed sequence is at least $\ell U_{\min}$. Putting everything together, we obtain

$$\mathbf{E}\left[\frac{\text{ALG}[\mathcal{S}]}{\text{OPT}[\mathcal{S}]}\right] \geq \mathbf{E}\left[\frac{\text{ALG}[\mathcal{S}]}{\mathbf{B}[\mathcal{S}]}\right] \geq \left(\frac{1}{2}\right) \cdot \frac{\ell U_{\min}}{4c\ell U_{\max}/\log(D)} = \Omega\left(\frac{U_{\min}}{U_{\max}} \cdot \log(D)\right).$$

□

Lemma 5.8.1. *Let G be a clique with $m+1$ nodes and maximum edge length $U_{\max}$. Consider an adversarial sequence $\check{\mathcal{S}}$ of ℓ elementary tasks for a sufficiently large ℓ. Then there exists an offline algorithm $\mathbf{B}$ such that for $m \geq 16$, $\mathbf{E}[\mathbf{B}[\mathcal{S}]] \leq c\ell U_{\max}/\log(m)$ for a constant c.*

Proof. We first consider an adversarial sequence $\check{\mathcal{S}} = \langle \check{\tau}_1, \ldots, \check{\tau}_k \rangle$ of $k := \lfloor \log(m)/2 \rfloor$ elementary tasks. We view the smoothing of the elementary tasks as being done in two stages.

Stage 1: Initially, we smoothen k zero tasks (all request costs are zero) according to the given smoothing distribution. Let the smoothed sequence be $\mathcal{S}' := \langle \tau'_1, \ldots, \tau'_k \rangle$.

Stage 2: For each t, $1 \leq t \leq k$, we obtain a task τ_t from τ'_t as follows. Let v^* be the node with non-zero request cost $\check{r}_t(v^*)$ in $\check{\tau}_t$. We replace the request cost of v^* in τ'_t by the outcome of smoothing $\check{r}_t(v^*)$. Let $\mathcal{S} := \langle \tau_1, \ldots, \tau_k \rangle$ be the resulting task sequence.

For any node v_i, we define a $0/1$ random variable X_i which is 1 if and only if the total request cost accumulated in v_i with respect to $\mathcal{S}'$ is zero. Since for each node v_i the request cost remains zero with probability at least $\frac{1}{2}$, we have $\mathbf{P}[X_i = 1] \geq (1/2)^k \geq 1/\sqrt{m}$. Note that the X_i's are independent. Let $\mathbf{X} := X_1 + \cdots + X_{m+1}$. We have $\mathbf{E}[\mathbf{X}] \geq \sqrt{m}$. Let $\mathcal{E}$ denote the event $(\mathbf{X} > \sqrt{m}/2)$. Using Chernoff's bound (see Theorem 2.4.10), we obtain

$$\mathbf{P}[\neg\mathcal{E}] = \mathbf{P}[\mathbf{X} \leq \sqrt{m}/2] \leq e^{-\sqrt{m}/8}.$$

The offline algorithm $\mathbf{B}$ has two different strategies depending on whether event $\mathcal{E}$ holds or not.

Strategy 1: If event $\mathcal{E}$ holds, $\mathbf{B}$ moves at the beginning to a node v_i whose total accumulated request cost is zero and stays there. (Recall that $\mathbf{B}$ is offline.) Note that since $\mathcal{E}$ holds

there are more than $\sqrt{m}/2 - k$ such nodes; for $m \geq 16$ there exists at least one such node.

Strategy 2: If event $\mathcal{E}$ does not hold, $\mathbf{B}$ always moves to a node with minimum request cost.

Since $\mathbf{B}$ only incurs the initial travel cost of at most $U_{\max}$ if $\mathcal{E}$ holds, we obtain

$$\mathbf{E}[\mathbf{B}[\mathcal{S}]] = \mathbf{E}[\mathbf{B}[\mathcal{S}] \,|\, \mathcal{E}]\ \mathbf{P}[\mathcal{E}]\ + \mathbf{E}[\mathbf{B}[\mathcal{S}] \,|\, \neg\mathcal{E}]\ \mathbf{P}[\neg\mathcal{E}]\ \leq U_{\max} + \mathbf{E}[\mathbf{B}[\mathcal{S}] \,|\, \neg\mathcal{E}] \cdot e^{-\sqrt{m}/8}.$$

Next, we bound $\mathbf{E}[\mathbf{B}[\mathcal{S}] \,|\, \neg\mathcal{E}]$. Clearly, the transition cost in each round is at most $U_{\max}$. The expected request cost incurred by $\mathbf{B}$ in round t is $\mathbf{E}[\min_{u \in V} r_t(u) \,|\, \neg\mathcal{E}]$. Consider a node v_i with $\check{r}_t(v_i) = 0$. The smoothed request cost of v_i is not affected by Stage 2. We have $\mathbf{E}[\min_{u \in V} r_t(u) \,|\, \neg\mathcal{E}] \leq \mathbf{E}[r_t(v_i) \,|\, \neg\mathcal{E}]$. Let $(X_1 = x_1, \ldots, X_{m+1} = x_{m+1})$ be any outcome such that $\neg\mathcal{E}$ holds. Since the request costs are chosen independently, we have $\mathbf{E}[r_t(v_i) \,|\, X_1 = x_1, \ldots, X_{m+1} = x_{m+1}] = \mathbf{E}[r_t(v_i) \,|\, X_i = x_i]$. If $x_i = 1$ then $\mathbf{E}[r_t(v_i) \,|\, X_i = x_i] = 0$, since all request costs at v_i must be zero. If $x_i = 0$ then $\mathbf{E}[r_t(v_i) \,|\, X_i = x_i] \leq \mathbf{E}[r_t(v_i) \,|\, r_t(v_i) > 0]$. (For $r_t(v_i)$ the event $(X_i = 0)$ means that either $r_t(v_i) = 0$ and $r_{t'}(v_i) > 0$ for some $t' \neq t$, or $r_t(v_i) > 0$.) By Fact 5.7.2, the expected cost $\mathbf{E}[r_t(v_i)]$ is at most σ. Moreover, $\mathbf{P}[r_t(v_i) > 0] \geq \mathbf{P}[r_t(v_i) \geq \sigma/c_f] \geq \frac{1}{4}$. Hence, $\mathbf{E}[r_t(v_i) \,|\, r_t(v_i) > 0] \leq 4\mathbf{E}[r_t(v_i)] \leq 4\sigma$. Putting everything together, we obtain

$$\mathbf{E}[\mathbf{B}[\mathcal{S}] \,|\, \neg\mathcal{E}] \leq \sum_{t=1}^{k} (\mathbf{E}[\min_{u \in V} r_t(u) \,|\, \neg\mathcal{E}] + U_{\max}) \leq k(4\sigma + U_{\max}) \leq 9kU_{\max},$$

where the last inequality holds since we assume that $\sigma \leq 2U_{\min} \leq 2U_{\max}$.

Altogether, we obtain for a sequence $\mathcal{S}$ of length k and for $m \geq 16$

$$\mathbf{E}[\mathbf{B}[\mathcal{S}]] \leq U_{\max} + 9kU_{\max} \cdot e^{-\sqrt{m}/8} \leq 13U_{\max}.$$

We conclude the proof as follows. We split the entire adversarial sequence $\check{\mathcal{S}}$ of length ℓ into $j \geq 1$ subsequences of length k (the final one might have length less than k). On each subsequence, $\mathbf{B}$ performs as described above. We therefore obtain for the entire sequence $\mathcal{S}$ and an appropriate constant c

$$\mathbf{E}[\mathbf{B}[\mathcal{S}]] \leq \mathbf{E}\left[\sum_{t=1}^{j} 13U_{\max}\right] = 13jU_{\max} \leq \frac{c\ell U_{\max}}{\log(m)},$$

where the last inequality follows from the relation between ℓ and j and definition of k. □

5.8.3 Existential Lower Bounds

We provide two existential lower bounds showing that for a large range of parameters n, $U_{\min}$, $U_{\max}$, D, and $Diam$ there exists a class of graphs on which *any* deterministic algorithm has

a smoothed competitive ratio that asymptotically matches the upper bounds stated in Theorem 5.5.1 and Theorem 5.6.1. In order to prove these existential lower bounds, we first show the following lemma.

Lemma 5.8.2. *Given a number of nodes n, minimum edge cost $U_{\min}$, maximum edge cost $U_{\max}$, maximum degree $D \geq 3$, and diameter $Diam$ such that*

$$Diam \geq 4U_{\min}\log_{D-1}(n) \quad \textit{and} \quad \mathcal{D} := \min\left(Diam/U_{\max}, D\right) \geq 17,$$

there exists a graph such that the smoothed competitive ratio of any deterministic algorithm ALG *is*

$$\Omega\left(\min\left(\frac{nU_{\max}}{Diam}, \frac{Diam}{U_{\min}} \cdot \left(\frac{U_{\min}}{\sigma} + \log(\mathcal{D})\right)\right)\right).$$

Observe that in any graph of n nodes and maximum degree D, $Diam/U_{\min} \geq \log_{D-1}(n)$, i.e., the condition in Lemma 5.8.2 is slightly stronger.

Proof of Lemma 5.8.2 . We construct a graph G as depicted in Figure 5.3. The graph consists of $m := \frac{1}{2}nU_{\max}/Diam$ cliques. Each clique has $\mathcal{D}$ nodes and the length of an edge between any two nodes is $U_{\min}$. We need to ensure that the maximum degree is at most D. Therefore, we connect each clique by a path to a $(D-1)$-ary tree T. Each such path consists of X edges of length $U_{\max}$. We assign a length of $U_{\min}$ to each edge in T. Each clique is attached to a leaf node of T; a leaf node may take up to $D-1$ cliques. Since m cliques need to be connected to T and we can attach at most $(D-1)^h$ cliques to a tree of height $h-1$, we fix $h := \log_{D-1}(m)$. The total number of nodes in T is therefore $((D-1)^h - 1)/(D-2) \leq m$, since $D \geq 3$. It is easy to verify that $m + m \cdot (X-1) + m \cdot \mathcal{D} \leq n$, i.e., the total number of nodes in G is at most n. (If it is less than n, we let the remaining nodes become part of T.) The graph should have diameter $Diam$ and thus we fix X such that $2(U_{\min} + X \cdot U_{\max} + (h-1)U_{\min}) = Diam$, i.e., $X := \lceil (Diam/2 - hU_{\min})/U_{\max} \rceil$. Moreover, we want that the minimum distance between any two nodes in different cliques is at least $\frac{1}{4}Diam$, i.e., $X \cdot U_{\max} \geq \frac{1}{8}Diam$. If $Diam \geq 4U_{\min}\log_{D-1}(n)$, this condition holds.

Consider the case $U_{\min}/\sigma > \log(\mathcal{D})$. We need to prove a lower bound of $\Omega(\min(nU_{\max}/Diam, Diam/\sigma))$. In each round, the adversary imposes an ∞ cost on all nodes of the graph except on those nodes that join a clique with its path. That is, the adversary restricts both ALG and OPT to stay in a "virtual" clique of size m with $U_{\min} = \frac{1}{4}Diam$ and $U_{\max} = Diam$. Applying the universal lower bound of Theorem 5.8.3 to this clique we obtain the desired lower bound of $\Omega(\min(m, Diam/\sigma))$.

Consider the case $U_{\min}/\sigma \leq \log(\mathcal{D})$. In each round, the adversary imposes an ∞ cost on all nodes in T and on all nodes that belong to a connecting path. Furthermore, in each round, the adversary forces the online algorithm ALG to leave its clique by specifying ∞ costs on all nodes of the clique in which ALG resides. All other request costs are zero.

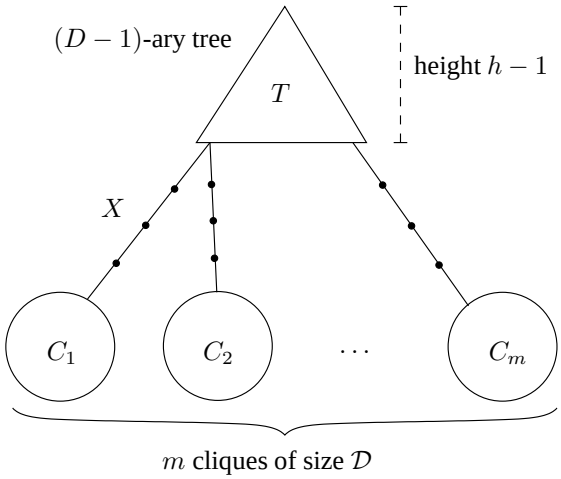

Figure 5.3: Structure of graph constructed in the proof of Lemma 5.8.2.

We use an averaging technique. We define a collection of $m-1$ offline algorithms and compare the cost of ALG with the average cost of the offline algorithms. At most one algorithm resides in each clique. An offline algorithm $\mathbf{B}_i$ remains in its clique C_i until ∞ costs are imposed on C_i; at this point, $\mathbf{B}_i$ moves to the free clique. Within each clique, the offline algorithm follows the strategy as specified in the proof of Lemma 5.8.1. We may assume without loss of generality that each $\mathbf{B}_i$ starts in a different clique (see Appendix 5.B).

Consider a smoothed sequence $\mathcal{S}$ of length ℓ. Let $\mathbf{B}[\mathcal{S}]$ be the total cost incurred by the offline algorithms and define $\mathbf{B}_i[\mathcal{S}]$ as the total cost of $\mathbf{B}_i$ on $\mathcal{S}$. The total cost of the offline algorithms to travel away from cliques with ∞ costs is at most $\ell Diam$. The expected cost of each algorithm in a clique with zero adversarial request cost is, due to Lemma 5.8.1, at most $c\ell U_{\min}/\log(\mathcal{D}-1)$; recall that each clique is of size $\mathcal{D} \geq 17$ and the maximum edge length in each clique is $U_{\min}$. Thus,

$$\mathbf{E}[\tilde{\mathbf{B}}[\mathcal{S}]] \leq \frac{\ell Diam}{m-1} + \frac{1}{m-1}\mathbf{E}\left[\sum_{i=1}^{m-1}\mathbf{B}_i[\mathcal{S}]\right] \leq \frac{\ell Diam}{m-1} + \frac{c\ell U_{\min}}{\log(\mathcal{D}-1)}.$$

By Markov's inequality, $\mathbf{P}[\tilde{\mathbf{B}}[\mathcal{S}] < 2\mathbf{E}[\tilde{\mathbf{B}}[\mathcal{S}]]] \geq \frac{1}{2}$. Clearly, $\text{ALG}[\mathcal{S}] \geq \frac{1}{4}\ell Diam$. Therefore,

$$\mathbf{E}\left[\frac{\text{ALG}[\mathcal{S}]}{\text{OPT}[\mathcal{S}]}\right] \geq \left(\frac{1}{2}\right)\frac{\frac{1}{4}\ell Diam}{2(\frac{\ell Diam}{m-1} + \frac{c\ell U_{\min}}{\log(\mathcal{D}-1)})} = \Omega\left(\min\left(m, \frac{Diam}{U_{\min}} \cdot \log(\mathcal{D})\right)\right).$$

□

The next bound shows that if Theorem 5.5.1 gives a better upper bound than Theorem 5.6.1 then this bound is tight up to a factor of $\log(D)/\log(\mathcal{D}) \leq \log(n)$ for a large class of graphs; moreover, for $D \leq Diam/U_{\min}$ this bound is tight up to a constant factor.

Theorem 5.8.4. *There exists a class of graphs such that the smoothed competitive ratio of any deterministic algorithm* ALG *is*

$$\Omega\left(\min\left(n, \frac{Diam}{U_{\min}}\left(\frac{U_{\min}}{\sigma} + \log(\mathcal{D})\right)\right)\right),$$

where $\mathcal{D} = \min(Diam/U_{\min}, D)$.

Proof. If Theorem 5.5.1 gives a better upper bound than Theorem 5.6.1, we have

$$\frac{Diam}{U_{\min}}\left(\frac{U_{\min}}{\sigma} + \log(D)\right) \leq \sqrt{n \cdot \frac{U_{\max}}{U_{\min}}\left(\frac{U_{\min}}{\sigma} + \log(D)\right)},$$

which is equivalent to

$$\frac{nU_{\max}}{Diam} \geq \frac{Diam}{U_{\min}}\left(\frac{U_{\min}}{\sigma} + \log(D)\right).$$

Since $\log(D) \geq \log(\mathcal{D})$, we obtain from Lemma 5.8.2 the desired lower bound. □

Theorem 5.8.5. *There exist a class of graphs such that the smoothed competitive ratio of any deterministic algorithm* ALG *is*

$$\Omega\left(\min\left(n, \sqrt{n\frac{U_{\max}}{U_{\min}}\left(\frac{U_{\min}}{\sigma} + \log(\mathcal{D})\right)}\right)\right),$$

where $\mathcal{D} = \min(Diam/U_{\min}, D)$.

Proof. Let $U_{\min}/\sigma > \log(\mathcal{D})$. We fix $Diam$ such that $nU_{\max}/Diam = Diam/\sigma$, i.e., $Diam = \sqrt{n\sigma U_{\max}}$. The lower bound of Lemma 5.8.2 then reduces to $\Omega(\sqrt{nU_{\max}/\sigma})$.

Assume $U_{\min}/\sigma \leq \log(\mathcal{D})$. We fix $Diam$ such that $nU_{\max}/Diam = (Diam/U_{\min})\log(\mathcal{D})$, i.e., $Diam = \sqrt{nU_{\max}U_{\min}/\log(\mathcal{D})}$. The lower bound of Lemma 5.8.2 then reduces to $\Omega(\sqrt{n(U_{\max}/U_{\min})\log(\mathcal{D})})$. □

5.9 Concluding Remarks

In this chapter, we investigated the asymptotic behavior of WFA if the request costs of an adversarial task sequence are perturbed by means of a symmetric additive smoothing model. We showed that the smoothed competitive ratio of WFA is much better than $O(n)$ and that it depends on certain topological parameters of the underlying graph. Moreover, all our bounds, except the one for β-elementary tasks, are tight up to constant factors. We believe that our analysis gives a strong indication that the performance of WFA in practice is much better than $2n - 1$.

It might be of some interest to analyze the smoothed competitiveness of WFA using different smoothing models. However, we already showed that zero-retaining smoothing models,

such as the relative smoothing model, cannot yield a smoothed competitive ratio better than $2n - 1$. An open problem would be to strengthen the universal lower bounds. Moreover, it would be interesting to obtain exact bounds on the smoothed competitive ratio of WFA.

5.A Proofs of Facts

Proof of Fact 5.2.3 . Assume x is the node that defines $w_t(v)$, i.e., $w_t(v) = w_{t-1}(x)+r_t(x)+\delta(x,v)$. We have $w_t(u) \leq w_{t-1}(x)+r_t(x)+\delta(x,u) \leq w_{t-1}(x)+r_t(x)+\delta(x,v)+\delta(v,u) = w_t(v)+\delta(v,u)$. □

Proof of Fact 5.2.4 . By (5.2), we have that $w_t(s_t)+\delta(s_{t-1},s_t) \leq w_t(v)+\delta(s_{t-1},v)$ for all $v \in V$. In particular, for $v = s_{t-1}$ this implies $w_t(s_t) \leq w_t(s_{t-1}) - \delta(s_{t-1},s_t)$. On the other hand, due to Fact 5.2.3, $w_t(s_t) \geq w_t(s_{t-1}) - \delta(s_{t-1},s_t)$. □

Proof of Fact 5.2.5 . Using (5.2) and Fact 5.2.4, we obtain

$$r_t(s_t)+\delta(s_{t-1},s_t) = w_t(s_t) - w_{t-1}(s_t) + w_t(s_{t-1}) - w_t(s_t) = w_t(s_{t-1}) - w_{t-1}(s_t).$$

□

Proof of Fact 5.7.1. Let X be a random variable chosen from f. Define $\mathcal{E}$ as the event $(|X-\mu| \geq \mu/2)$. Using Chebyshev's inequality (see Theorem 2.4.9), we obtain

$$\mathbf{P}[\mathcal{E}] = \mathbf{P}\Big[|X-\mu| \geq \frac{\mu}{2}\Big] \leq \frac{4\sigma^2}{\mu^2}. \tag{5.11}$$

Since f is continuous and non-increasing in $[0,\infty)$,

$$\mathbf{P}[\mathcal{E}] = \mathbf{P}\Big[|X-\mu| \geq \frac{\mu}{2}\Big] \geq \mathbf{P}\Big[X \leq \frac{\mu}{2}\Big] \geq \frac{1}{2}\mathbf{P}\Big[\frac{\mu}{2} < X \leq \frac{3\mu}{2}\Big] \geq \frac{1}{2}\mathbf{P}[\neg\mathcal{E}]\,.$$

This implies that $\mathbf{P}[\mathcal{E}] \geq \frac{1}{3}$. Hence, (5.11) gives $\mu^2 \leq 12\sigma^2$. □

Proof of Fact 5.7.2 . Define $Y := \max(0,X)$. Since $\mu = 0$, we have $\sigma^2 = \mathbf{E}[X^2]$. Let σ_Y denote the standard deviation of the distribution of Y. By the definition of $\mathbf{E}[X^2]$, $\mathbf{E}[Y^2] = \frac{1}{2}\mathbf{E}[X^2]$. Since $\sigma_Y^2 = \mathbf{E}[Y^2] - \mathbf{E}[Y]^2$ and $\sigma_Y^2 \geq 0$, we have $\mathbf{E}[Y]^2 \leq \mathbf{E}[Y^2]$. This in turn implies that $\mathbf{E}[Y] \leq \sigma/\sqrt{2}$. □

Proof of Fact 5.6.1 . Define $\mathcal{X} := \min\Big(\frac{A\sum_{i=1}^m X_i}{\sum_{i=1}^m X_i^2}, \frac{B\sum_{i=1}^m X_i}{m}\Big)$. First, note that

$$m(X_1^2+X_2^2+\cdots+X_m^2) \geq (X_1+X_2+\cdots+X_m)^2, \tag{5.12}$$

because

$$\frac{1}{2}\sum_{i,j}(X_i^2+X_j^2) \geq \sum_{i=1}^m X_i^2 + \sum_{i,j,i\neq j} X_iX_j.$$

Define $Y := \sum_{i=1}^m X_i/m$. Note that Y is positive. Due to (5.12), we can write $\mathcal{X} \leq \min(A/Y, BY)$. The latter expression is maximized if $A/Y = BY$, i.e., if $Y = \sqrt{A/B}$. Thus $\mathcal{X} \leq \sqrt{AB}$. □

5.B Constant Additive Cost of the Offline Algorithm

We would like to point out that in our lower bound proofs we can assume without loss of generality that OPT incurs an additional additive cost of Z which is independent of the length of the input sequence. This does not change the asymptotics of the lower bounds, which can be seen as follows. We always prove a lower bound of say $\Omega(Y/X)$ on a task sequence of length ℓ by showing that with constant probability the expected cost of ALG is at least $Y \cdot \ell$ and the cost of OPT is at most $X \cdot \ell$. In order to make sure that the additive cost Z does not influence the competitive ratio, we only have to make sure that the task sequence under consideration is sufficiently long. If we choose ℓ such that $X \cdot \ell \geq Z$, we obtain a lower bound of $\Omega((Y \cdot \ell)/(X \cdot \ell + Z)) = \Omega(Y/X)$.

CONCLUSION

We presented a heuristic improvement for the single-source many-targets shortest path problem. This problem is repeatedly solved by matching algorithms to compute maximum weighted bipartite matchings. Apparently, in the worst case the heuristic might have no effect. However, intuitively, it is clear that the heuristic can only help to reduce the number of queue operations performed by Dijkstra's algorithm to solve this problem. We substantiated this intuition by providing a partial average case analysis showing that on random input a significant fraction of queue operations is saved by the heuristic. Furthermore, in our experiments we observed an improvement in running time for the matching algorithm. The heuristic is simple and can easily be implemented.

A large part of this thesis was devoted to *smoothed competitive analysis*. Based on the ideas underlying smoothed complexity, we proposed to represent the competitiveness of an online algorithm by its *smoothed competitive ratio*. We have seen that smoothed competitive analysis provides a unifying framework of worst case and average case analysis of online algorithms.

We applied this novel notion to the multi-level feedback algorithm (MLF) for the non-clairvoyant scheduling problem and to the work function algorithm (WFA) for metrical task systems. As mentioned in the introduction, smoothed complexity can be interpreted as a measure of fragility of worst case instances. So, one might pose the question:

"How fragile are the worst case instances of these two problems?"

For the multi-level feedback algorithm the answer to this question is subject to interpretation. We proved that MLF has smoothed competitive ratio of essentially $O(2^{K-k})$, if the k least significant bits of the processing times are set at random. One might say that this indicates that worst case instances are rather stable to perturbations; even if we perturb a constant fraction of the K bits, the competitive ratio of MLF remains exponential. However, we would like to put it differently. The competitive ratio of MLF improves exponentially with the amount k of perturbation; therefore perturbing a constant fraction of the bits results in an exponential decrease in its competitive ratio. We also proved an $\Omega(2^{K-k})$ lower bound on the smoothed competitive ratio of *any* deterministic online algorithm under the partial bit randomization model. Besides showing that our analysis is tight, this lower bound also indicates that MLF is asymptotically optimal. For various other smoothing models, including the symmetric additive and relative smoothing models as proposed by Spielman and Teng, we proved a

higher $\Omega(2^K)$ bound on the smoothed competitive ratio of MLF. Put differently, under these smoothing models worst case instances are invariant to perturbations.

For the work function algorithm we can certainly state that worst case instances are fragile if the request costs of the tasks are smoothed according to a symmetric additive smoothing model. Depending on the topology of the underlying graph we have seen that the smoothed competitive ratio of WFA improves significantly, even for moderate perturbations in the order of the minimum edge length. For example, if the underlying graph is a clique of size n, the smoothed competitive ratio reduces from $O(n)$ to $O(\log(n))$. We also provided lower bounds for *any* deterministic online algorithm for metrical task systems. These bounds imply that our analysis is tight up to constant factors and that WFA is asymptotically optimal under the symmetric additive smoothing model. Furthermore, we argued that any deterministic online algorithm has smoothed competitive ratio $\Omega(n)$ if a zero-retaining smoothing model, such as the relative smoothing model, is used. We therefore conclude that under these models worst case instances are invariant to perturbations.

In the introduction we also stated that from the analyses we obtain new insights into the behavior of the algorithms.

"What have we learnt from the analyses?"

From both the smoothed competitive analysis of the multi-level feedback algorithm and the work function algorithm we were able to infer a relation between the performance of the algorithm and certain properties of the input. For instance, we have seen that the competitive ratio of MLF is related to the accuracy of the final estimates of the processing times. Moreover, in the analysis of WFA we clearly established a connection between the competitiveness of the algorithm and the structure of the underlying graph.

Precisely because of the obtaining of these new insights we believe that it is worth to investigate the smoothed competitiveness of online algorithms, and we are confident that this new performance measure will be used in the future to describe the quality of online algorithms.

A. NOTATIONS AND THEIR DEFINITIONS

Notation	Definition
ALG	generic algorithm
OPT	optimal algorithm
$\{a, \ldots, b\}$	set $\{a, \ldots, b\}$
$[n]$	$\{1, \ldots, n\}$
$\mathbb{N}$	natural numbers
$\mathbb{R}$	real numbers
$\mathbb{R}^+$	non-negative reals
$[a, b]$	continuous range $\{x \in \mathbb{R} : a \leq x \leq b\}$
$(a, b]$	continuous range $\{x \in \mathbb{R} : a < x \leq b\}$
$[a, b)$	continuous range $\{x \in \mathbb{R} : a \leq x < b\}$
(a, b)	continuous range $\{x \in \mathbb{R} : a < x < b\}$
2^S for some set S	power set of S
$\mathcal{I}$	set of all input instances
$\mathcal{I}(n)$	set of all input instances of size n
$\check{I}$	adversarial, original, or initial instance
I	perturbed, or smoothed, instance
f	(smoothing) distribution, density function
σ	smoothing parameter, standard deviation (of f)
μ	expectation (of f)
$N(\check{I}, \sigma)$	neighborhood of $\check{I}$ with smoothing parameter σ
$\varepsilon \stackrel{f}{\leftarrow} [a, b]$	ε chosen independently according to f over $[a, b]$
$\mathbf{P}[\cdot]$	probability function
$\mathbf{E}[\cdot]$	expectation
$\mathbf{Var}[\cdot]$	variance
Ω	probability space
ω	elementary event
$F(x)$	distribution function $\mathbf{P}[\varepsilon \leq x]$
$\mathbf{Bin}[n, p]$	binomial distribution
$G(n, p)$	random graph model, n nodes, edge probability p
$G(n, m)$	random graph model, n nodes, m edges, equiprobable

BIBLIOGRAPHY

[Alb97] S. Albers. On the influence of lookahead in competitive paging algorithms. *Algorithmica*, 18:283–305, 1997.

[Alb98] S. Albers. A competitive analysis of the list update problem with lookahead. *Theoretical Computer Science*, 197:95–109, 1998.

[AMO93] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice Hall, 1993.

[AS00] N. Alon and J. H. Spencer. *The Probabilistic Method*. John Wiley & Sons, second edition, 2000.

[ASE92] N. Alon, J. H. Spencer, and P. Erdös. *The Probabilistic Method*. John Wiley & Sons, 1992.

[AV79] D. Angluin and L. Valiant. Fast probabilistic algorithms for hamiltonian circuits and matchings. *Journal of Computer and System Sciences*, 19:155–193, 1979.

[BBM03] C. Banderier, R. Beier, and K. Mehlhorn. Smoothed analysis of three combinatorial problems. In *Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science*, volume 2747, pages 198–207, 2003.

[BCKV04] R. Beier, A. Czumaj, P. Krysta, and B. Vöcking. Computing equilibria for congestion games with (im)perfect information. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 739–748, 2004.

[BD02] A. Blum and J. Dunagan. Smoothed analysis of the perceptron algorithm. In *In Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 905–914, 2002.

[BEY98] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[BIRS95] A. Borodin, S. Irani, P. Raghavan, and B. Schieber. Competitive paging with locality of reference. *Journal of Computer and System Sciences*, 50(2):244–258, 1995.

[BL01] L. Becchetti and S. Leonardi. Non-clairvoyant scheduling to minimize the average flow time on single and parallel machines. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, pages 94–103, 2001.

[BLMS+03a] L. Becchetti, S. Leonardi, A. Marchetti-Spaccamela, G. Schäfer, and T. Vredeveld. Average case and smoothed competitive analysis of the multi-level feedback algorithm. In *Proceedings of the Forty-Fourth Annual IEEE Symposium on Foundations of Computer Science*, pages 462–471, 2003.

[BLMS+03b] L. Becchetti, S. Leonardi, A. Marchetti-Spaccamela, G. Schäfer, and T. Vredeveld. Average case and smoothed competitive analysis of the multi-level feedback algorithm. Technical Report MPI–I–2003–1–014, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 2003.

[BLS92] A. Borodin, N. Linial, and M. Saks. An optimal online algorithm for metrical task systems. *Journal of the ACM*, 39:745–763, 1992.

[BMST03] H. Bast, K. Mehlhorn, G. Schäfer, and H. Tamaki. A heuristic for Dijkstra's algorithm with many targets and its use in weighted matching algorithms. *Algorithmica*, 36(1):75–88, 2003.

[DST02] J. Dunagan, D. A. Spielman, and S. H. Teng. Smoothed analysis of the Renegar's condition number for linear programming. ⟨*http://www-math.mit.edu/~spielman/SmoothedAnalysis*⟩, accessed 2002.

[ER59] P. Erdős and A. Rényi. On random graphs I. *Publ. Math. Debrecen*, 6:290–297, 1959.

[FKG71] C. M. Fortuin, P. W. Kasteleyn, and J. Ginibre. Correlation inequalities on some partially ordered sets. *Commun. Math. Physics*, 22:89–103, 1971.

[Gal86] Z. Galil. Efficient algorithms for finding maximum matching in graphs. *ACM Computing Surveys*, 18(1):23–37, 1986.

[GS01] G. R. Grimmett and D. R. Stirzaker. *Probability and random processes*. Oxford University Press, third edition, 2001.

[KP94] E. Koutsoupias and C. Papadimitriou. Beyond competitive analysis. In *Proceedings of the Twenty-Fifth Annual IEEE Symposium on Foundations of Computer Science*, pages 394–400, 1994.

[KP97] B. Kalyanasundaram and K. Pruhs. Minimizing flow time nonclairvoyantly. In *Proceedings of the Thirty-Eight Annual IEEE Symposium on Foundations of Computer Science*, pages 345–352, 1997. To appear in Journal of the ACM.

[KP00] B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. *Journal of the ACM*, 47(4):617–643, 2000.

[LR97] S. Leonardi and D. Raz. Approximating total flow time on parallel machines. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 110–119, 1997.

[MC74] J. E. Michel and E. G. Coffman, Jr. Synthesis of a feedback queueing discipline for computer operation. *Journal of the ACM*, 21:329–339, 1974.

[MMS88] M. S. Manasse, L. A. McGeoch, and D. D. Sleator. Competitive algorithms for on-line problems. In *Proceedings of the 20th ACM Symposium on Theory of Computing*, pages 322–333, 1988.

[MN99] K. Mehlhorn and S. Näher. *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, 1999.

[MPT94] R. Motwani, S. Phillips, and E. Torng. Non-clairvoyant scheduling. *Theoretical Computer Science*, 130:17–47, 1994.

[MS00] K. Mehlhorn and G. Schäfer. Implementation of $O(nm \log n)$ weighted matchings in general graphs. The power of data structures. In *4th International Workshop on Algorithm Engineering*, volume 1982 of *Lecture Notes in Computer Science*, pages 23–38, 2000.

[MS01] K. Mehlhorn and G. Schäfer. A heuristic for Dijkstra's algorithm with many targets and its use in weighted matching algorithms. In *Proceedings of the 9th Annual European Symposium on Algorithms*, volume 2161 of *Lecture Notes in Computer Science*, pages 242–253, 2001.

[MS02] K. Mehlhorn and G. Schäfer. Implementation of $O(nm \log n)$ weighted matchings in general graphs. The power of data structures. *ACM Journal of Experimental Algorithmics*, 7, 2002.

[Nut99] G. Nutt. *Operating System Projects Using Windows NT*. Addison Wesley, Reading, 1999.

[sah] Smoothed analysis homepage. ⟨*http://www-math.mit.edu/~spielman/SmoothedAnalysis*⟩.

[Sch00] G. Schäfer. Weighted matchings in general graphs. Master's thesis, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, Germany, 2000.

[SS03] G. Schäfer and N. Sivadasan. Topology matters: Smoothed competitiveness of metrical task systems. Research Report MPI-I-2003-1-016, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, December 2003.

[SS04] G. Schäfer and N. Sivadasan. Topology matters: Smoothed competitiveness of metrical task systems. In *Conference Proceedings of the 21st International Symposium on Theoretical Aspects of Computer Science*, 2004. To appear.

[SSS02] M. Scharbrodt, T. Schickinger, and A. Steger. A new average case analysis for completion time scheduling. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*, pages 170–178, 2002.

[SST02] A. Sankar, D. A. Spielman, and S. H. Teng. Smoothed analysis of the condition numbers and growth factors of matrices. ⟨*http://www-math.mit.edu/~spielman/SmoothedAnalysis*⟩, accessed 2002.

[ST85] D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28:202–208, 1985.

[ST01] D. Spielman and S. H. Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, pages 296–305, 2001.

[ST02] D. Spielman and S. H. Teng. Smoothed analysis of property testing. ⟨*http://www-math.mit.edu/~spielman/SmoothedAnalysis*⟩, 2002.

[ST03] D. Spielman and S. H. Teng. Smoothed analysis of interior-point algorithms: Termination. ⟨*http://www-math.mit.edu/~spielman/SmoothedAnalysis*⟩, submitted, 2003.

[Tan92] A. S. Tanenbaum. *Modern Operating Systems*. Prentice-Hall Inc., 1992.

CURRICULUM VITAE

PERSONAL DETAILS

Name	Guido Schäfer
Date of birth	15.11.1974
Place of birth	Köln, Germany
Nationality	German

EDUCATION

10/2000 – present	**Max-Planck-Institut für Informatik, Saarbrücken, Germany** Ph. D. student of computer science; advisor: Prof. Dr. Dr.-Ing. E. h. K. Mehlhorn
10/1994 – 08/2000	**Universität des Saarlandes, Saarbrücken, Germany** student of computer science and computational linguistics
04/1999 – 05/2000	Master's thesis; advisor: Prof. Dr. Dr.-Ing. E. h. K. Mehlhorn; title of thesis: *Weighted Matchings in General Graphs*
08/2000	Diplom ($\approx$ Master's degree) in computer science
08/1985 – 06/1994	**Gymnasium der Stadt Kerpen, Germany**
06/1994	Abitur ($\approx$ high-school diploma)

STAYS ABROAD

04/2003 – 06/2003 & 06/2002 – 12/2002	**Università di Roma, La Sapienza, Italy** visiting student; advisor: Prof. Dr. Stefano Leonardi
09/1997 – 06/1998	**Trinity College, University of Dublin, Ireland** visiting student

WORK AND TEACHING EXPERIENCE

10/2000 – 03/2001 **Max-Planck-Institut für Informatik, Saarbrücken, Germany**
teaching assistant for *Algorithms and Data Structures II*, held by Prof. Dr. Dr.-Ing. E. h. K. Mehlhorn and Priv. Doz. Dr. P. Sanders

10/1998 – 03/1999 **Max-Planck-Institut für Informatik, Saarbrücken, Germany**
teaching assistant for *Algorithms and Data Structures*, held by Prof. Dr. R. Fleischer and Prof. Dr. J. Sibeyn

05/1997 – 09/1997 **Universität des Saarlandes, Saarbrücken, Germany**
member of the CHORUS-Project *Semantic Processing with Concurrent Constraints* (SFB 378); advisor: Prof. Dr. M. Pinkal (department of computational linguistics)

05/1996 – 04/1997 **Max-Planck-Institut für Informatik, Saarbrücken, Germany**
student assistant

SCHOLARSHIP

01/2001 – 12/2003 Member of the Graduiertenkolleg (graduate studies program) "Quality Guarantees for Computer Systems", department of computer science, Universität des Saarlandes, Saarbrücken, Germany; granted by the Deutsche Forschungsgemeinschaft

PUBLICATIONS AND COLLOQUIA

Journals

H. Bast, K. Mehlhorn, G. Schäfer, and H. Tamaki. A heuristic for Dijkstra's algorithm with many targets and its use in weighted matching algorithms. *Algorithmica*, 36(1):75–88, 2003.

K. Mehlhorn, V. Priebe, G. Schäfer, and N. Sivadasan. All-pairs shortest-paths computation in the presence of negative cycles. *Information Processing Letters*, 81(6):341–343, 2002.

K. Mehlhorn and G. Schäfer. Implementation of $O(nm \log n)$ weighted matchings in general graphs. The power of data structures. *ACM Journal of Experimental Algorithmics*, 7, 2002.

Conference Proceedings

S. Leonardi and G. Schäfer. Cross-monotonic cost sharing methods for connected facility location games. In *ACM Conference on Electronic Commerce (EC)*, 2004. To appear.

H. Bast, K. Mehlhorn, G. Schäfer, and H. Tamaki. Matching algorithms are fast in sparse random graphs. In *Conference Proceedings of the 21st International Symposium on Theoretical Aspects of Computer Science (STACS)*, 2004. To appear.

G. Schäfer and N. Sivadasan. Topology matters: Smoothed competitiveness of metrical task systems. In *Conference Proceedings of the 21st International Symposium on Theoretical Aspects of Computer Science (STACS)*, 2004. To appear.

L. Becchetti, S. Leonardi, A. Marchetti-Spaccamela, G. Schäfer, and T. Vredeveld. Average case and smoothed competitive analysis of the multi-level feedback algorithm. In *Proceedings of the Forty-Fourth Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 462–471, 2003.

L. Becchetti, S. Leonardi, A. Marchetti-Spaccamela, and G. Schäfer. Scheduling to minimize flow time metrics. In *3rd Workshop on Wireless, Mobile and Ad Hoc Networks (WMAN), satellite workshop of 17th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2003.

K. Mehlhorn and G. Schäfer. A heuristic for Dijkstra's algorithm with many targets and its use in weighted matching algorithms. In *Proceedings of the 9th Annual European Symposium on Algorithms (ESA)*, volume 2161 of *Lecture Notes in Computer Science*, pages 242–253, 2001.

K. Mehlhorn and G. Schäfer. Implementation of $O(nm \log n)$ weighted matchings in general graphs. The power of data structures. In *4th International Workshop on Algorithm Engineering (WAE)*, volume 1982 of *Lecture Notes in Computer Science*, pages 23–38, 2000.

D. Frigioni, T. Miller, U. Nanni, G. Pasqualone, G. Schäfer, and C. Zaroliagis. An experimental study of dynamic algorithms for directed graphs. In *Proceedings of the 6th Annual European Symposium on Algorithms (ESA)*, volume 1461 of *Lecture Notes in Computer Science*, pages 368–380, 1998.

Master's Thesis

G. Schäfer. Weighted matchings in general graphs. Master's thesis, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, Germany, 2000.

Technical Reports

G. Schäfer. A note on the smoothed complexity of the single-source shortest path problem. Research Report MPI-I-2003-1-018, Max-

Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, November 2003.

S. Hert, T. Polzin, L. Kettner, and G. Schäfer. ExpLab: A tool set for computational experiments. Research Report MPI-I-2002-1-004, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, November 2002.

Colloquia

Matching Algorithms are Fast in Sparse Random Graphs

December 5, 2003, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, Germany.

Cross-Monotonic Cost Sharing Methods for Connected Facility Location Games

Ringvorlesung des Graduiertenkollegs, December 4, 2003, Universität des Saarlandes, Germany.

Average Case and Smoothed Competitive Analysis of the Multi-Level Feedback Algorithm

Forty-Fourth Annual IEEE Symposium on Foundations of Computer Science (FOCS), October 11–14, 2003, Cambridge, MA, USA.

Workshop der Informatik Graduiertenkollegs, July 16–18, 2003, Schloss Dagstuhl, Germany.

Ringvorlesung des Graduiertenkollegs, January 27, 2003, Universität des Saarlandes, Germany.

Seminario Interdipartimentale di Algoritmica, December 09, 2002, Università di Roma, La Sapienza, Italy.

Average Case and Smoothed Analysis of Shortest Paths

AMORE Internal Meeting, July 8–10, 2003, L'Aquila, Italy.

Smoothed Analysis Applied to the SSSP Problem

Seminario Interdipartimentale di Algoritmica, July 25, 2002, Università di Roma, La Sapienza, Italy.

Matchings in Bipartite Random Graphs: A Survey

Ringvorlesung des Graduiertenkollegs, May 6, 2002, Universität des Saarlandes, Germany.

A Heuristic for Dijkstra's Algorithm with Many Targets and its Use in Weighted Matching Algorithms

Nineth Annual European Symposium on Algorithms (ESA), August 28–31, 2001, BRICS, University of Aarhus, Denmark.

Forschungsseminar des Graduiertenkollegs, May 7, 2001, Universität des Saarlandes, Germany.

Saarbrücken, 2nd February 2004