Online Problems and Two-Player Games: Algorithms and Analysis

Naveen Sivadasan

Online Problems and Two-Player Games: Algorithms and Analysis

Online Problems and Two-Player Games: Algorithms and Analysis

Naveen Sivadasan

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

Saarbrücken
2004

Tag des Kolloquiumsm:	09. July 2004
Dekan:	Prof. Dr. Jörg Eschmeier
Gutachter:	Prof. Dr. Dr.-Ing. E. h. Kurt Mehlhorn
	Prof. Dr. Stefano Leonardi

Dedicated to my parents.

Abstract

In this thesis we study three problems that are adversarial in nature. Such problems can be viewed as a game between an algorithm and an adversary, where the adversary always tries to force the algorithm into worst-case scenarios during its execution. Many real world problems with inherent uncertainty or lack of information fit into this model. For instance, it includes the vast field of online problems where the input is only partially available and an adversary reveals the complete input gradually over time (online fashion). The algorithm has to perform efficiently under this uncertainty. In contrast to the online setting, in an offline setting, the complete input is available in the beginning. The first problem that we investigate is a classical online scheduling problem where a sequence of jobs that arrive online have to be assigned to a set of identical machines with the objective of minimizing the maximum load. We study a natural generalization of this problem where we allow migration of already scheduled jobs to other machines upon the arrival of a new job, thus bridging the gap between online and offline setting. Already for a small amount of migration, our result compares with the best results to date in both online and offline settings. From the point of view of sensitivity analysis, our results imply that, only small changes are to be made to the current schedule to accommodate a new job, if we are satisfied with near optimal solution. The other online problem that we study is the well-known metrical task systems problem. We present a probabilistic analysis of the well-known text book algorithm called the work function algorithm. Besides average-case analysis we also present smoothed analysis, which is a notion introduced recently as a hybrid between worst-case and average-case analysis. Our analysis reveals that the performance of this algorithm is much better than worst-case for a large class of inputs. This motivates us to support smoothed analysis as an alternative model for evaluating the performance of online algorithms. The third problem that we investigate is a pursuit-evasion game: an algorithm (the pursuer) has to find/catch an adversary that is ‘hiding’ in a graph where both players can travel in the graph. This problem belongs to the rich field of search games and it addresses the question of how long it takes for the pursuer to find the evader in a given graph that, for example, corresponds to a computer network or a geographic terrain. Such game models are also used to design efficient communication protocols. We present improved results against adversaries with varying power and also present tight lower bounds.

Kurzzusammenfassung

In der vorliegenden Arbeit beschäftigen wir uns mit drei Problemen, welche als eine Art Spiel zwischen einem Algorithmus und seinem Gegenspieler interpretiert werden können. In diesem Spiel versucht der Gegenspieler, den Algorithmus während seiner Ausführung in sein Worst-Case Verhalten zu zwingen. Eine Vielzahl von praxisrelevanten Problemen, in denen nicht von Beginn an die volle Information über die Eingabeinstanz zur Verfügung steht, lassen sich als derartige Spiele modellieren. Zu dieser Klasse von Problemen gehören z. B. auch online Probleme, in denen der Gegenspieler die Eingabeinstanz für den Algorithmus *online*, d. h. während der Ausführung des Algorithmus, spezifiziert. Das Ziel des Algorithmus ist es, auf dieser so spezifizierten Instanz möglichst effizient zu sein. Im Gegensatz zum online Szenario kennt der Algorithmus im offline Szenario die gesamte Eingabeinstanz gleich von Beginn an. Im online Szenario wird die Effizienz eines (online) Algorithmus anhand seines *Competitive Ratio* gemessen. Ein Algorithmus ist *c-competitive*, wenn die Kosten, die der Algorithmus auf einer beliebigen online Eingabe verursacht, maximal einen Faktor c von den Kosten eines optimalen (offline) Algorithmus, der die gesamte Eingabe kennt, entfernt ist.

Das erste Problem, dass wir betrachten, ist ein klassisches Scheduling Problem, in dem Jobs online eintreffen und auf identischen parallelen Maschinen verteilt werden müssen. Das Ziel ist es, die maximale Maschinenlast zu minimieren. Wir erweitern dieses Problem, indem wir erlauben, dass beim Eintreffen eines neuen Jobs die Zuweisungen von Jobs zu ihren entsprechenden Maschinen nachträglich geändert werden dürfen (sog. Neuzuweisungen), unter der Bedingung, dass die Gesamtgrösse der bewegten Jobs maximal β mal die Grösse des neu eingetroffenen Jobs ist. Für $\beta = 0$ erhalten wir das klassische online Problem, in dem getroffene Entscheidungen nicht rückgängig gemacht werden können. Für $\beta \to \infty$ erhalten wir hingegen das entsprechende Problem in einem offline Szenario. Bereits für kleine Werte für β, lässt sich unser Scheduling Algorithmus mit den derzeit besten Resultaten vergleichen. Schon für $\beta = 4/3$ erreicht er ein Competitive Ratio von 1.5 und schlägt damit die untere Schranke von 1.88 (1.58) für deterministische (randomisierte) Algorithmen des online Problems. Für das offline Problem präsentieren wir einen Linearzeit-Algorithmus, der für ein gegebenes $\epsilon > 0$ ein Competitive Ratio von $(1+\epsilon)$ erzielt und dabei nur einen konstanten Faktor $\beta(\epsilon)$ von Neuzuweisungen benötigt. Im Sinne einer Sensitivitätsanalyse kann man unsere Ergebnisse dahingehend interpretieren, dass man nur geringfügige Veränderungen vornehmen muss, um einen neuen Job in einem bereits vorhandenen Schedule einzupassen, wenn man sich mit guten approximativen Lösungen zufrieden gibt.

Das zweite online Problem, dass wir betrachten, ist das Metrical Task System Problem.

Ein online Algorithmus befindet sich in einem Graphen G mit n Knoten und kann sich in diesem Graphen bewegen, wobei er Kosten in Höhe der zurückgelegten Distanz verursacht. Der Algorithmus muss eine Sequenz von Tasks abarbeiten, welche online erscheinen. Jeder Task spezifiziert für jeden Knoten Kosten, die entstehen, wenn der Algorithmus den Task in diesem Knoten abarbeitet. Die Aufgabe ist es, die Gesamtkosten zu minimieren. Wir präsentieren eine probabilistische Analyse des Work Function Algorithmus (WFA) von Borodin, Linial und Saks, welcher ein optimales Competitive Ratio von $2n - 1$ hat. Das Competitive Ratio eines Algorithmus stellt allerdings oftmals eine zu pessimistische Abschätzung seiner tatsächlichen Effizienz dar. Kürzlich stellten Spielman und Teng ein neues Komplexitätsmass vor, die *Smoothed Complexity*. Die Idee ist es, die Eingabeinstanz zufällig zu perturbieren und die Effizienz des Algorithmus auf den perturbierten Instanzen zu messen. Wir präsentieren eine Smoothed Analyse für WFA. Unsere Ergebnisse zeigen, dass das Smoothed Competitive Ratio von WFA asymptotisch sehr viel besser als $O(n)$ ist und dass er von verschiedenen topologischen Parametern des zugrundeliegenden Graphen G abhängt. Als Beispiel erreicht WFA schon für geringfügige Perturbationen ein Smoothed Competitive Ratio von $O(\log n)$ auf einer Clique oder einem vollständigen binären Baum und von $O(\sqrt{n})$ auf einem beliebigen Graphen mit konstantem Grad. Desweiteren zeigen wir, dass unsere Schranken für eine grosse Klasse von Graphen bis auf einen konstanten Faktor scharf sind.

Als drittes Problem analysieren wir ein "Katz-und-Maus-Spiel": eine Katze (der Algorithmus) und eine Maus (der Gegenspieler) befinden sich in einem Graphen und die Katze versucht, die Maus zu fangen. Das Spiel wird in Runden gespielt und in jeder Runde kann sich sowohl die Katze als auch die Maus im Graphen bewegen. Die Katze und die Maus sehen sich nicht, es sei denn, sie befinden sich im selben Knoten. Wir nehmen an, dass die möglichen Bewegungen der Katze durch den Graphen vorgegeben sind, d. h. in jeder Runde kann die Katze sich entlang einer Kante im Graphen bewegen. Für die Maus betrachten wir zwei verschiedene Modelle: im ersten Modell ist die Maus auf den Graphen beschränkt und im zweiten Modell kann sich die Maus unbeschränkt im Graphen bewegen, d. h. sie kann in jeder Runde zu einem beliebigen Knoten springen. Beide Spieler können sich randomisierter Strategien bedienen. Wir betrachten daher die erwartete Anzahl von Runden (Fluchtlänge) bis sie sich in einem Knoten treffen als Zielfunktion, die von der Katze zu minimieren und von der Maus zu maximieren versucht wird. Wir präsentieren eine Strategie für die Katze, die auf allgemeinen Graphen eine Fluchtlänge von nur $O(n \log(diam))$ sowohl gegen eine auf den Graphen beschränkte als auch unbeschränkte Maus garantiert, wobei $diam$ den Durchmesser des Graphen bezeichnet. Diese Schranke ist beinahe scharf, da $\Omega(n)$ eine triviale untere Schranke für die Fluchtlänge in beiden Modellen ist. Ferner beweisen wir, dass unsere obere Schranke für den unbeschränkten Fall bis auf einen konstanten Faktor optimal ist. Desweiteren zeigen wir mögliche Strategien für die Katze, die eine Fluchtlänge von $O(n)$ garantieren, wenn es sich bei dem zugrundeliegenden Graphen um einen Spezialfall handelt, wie z. B. um einen vollständig binären Baum. Schliesslich präsentieren wir für stark verbundene Graphen Strategien für die Katze, die eine optimale Fluchtlänge von $O(n^2)$ erzeugen.

Acknowledgements

There are many people who have contributed either directly or indirectly to the completion of this thesis. First of all, I am grateful to my advisor Prof. Dr. Kurt Mehlhorn for his guidance, encouragement, patience and above all for being a great teacher. He also gave me sufficient scientific freedom to explore things on my own. He was always approachable for any academic/ non-academic issues that I faced. I learned a lot through discussions with him and by attending his lectures. Being his teaching assistant was also a great learning experience for me. I very much enjoyed the excellent research atmosphere in Max-Planck Institute für Informatik. I thank all my colleagues for this. My research was financially supported by the Max-Planck Society's fellowship for Ph.D. students associated to the International Max-Planck Research School.

I am grateful to Prof. Dr. Stefano Leonardi for being the external referee to my thesis. I thank him again for hosting me during my scientific visit to the University of Roma "La Sapienza".

I am indebted to all the administrative and support staffs who were responsible for my comfortable stay in the Max-Planck Institute. My sincere gratitude to Petra Mayer, Ingrid Finkler-Paul, Roxane Wetzel, Sabine Krott, Kerstin Meyer-Ross, Anja Becker and Michael Bentz.

I am grateful to Peter Sanders, Prof. Berthold Vöcking, Martin Skutella and Prof. Edgar Ramos for advising me and for the numerous discussions I had with them. It was a great learning experience for me working with them. I thank Prof. Vöcking also for his lessons on probabilistic analysis, especially the analysis of random process. I thank all my co-authors for the nice scientific collaborations we had.

I thank Surendar Baswana, Kavitha Telikepalli, Guido Schäfer, Martin Skutella and Rene Beier for carefully reading parts of this thesis and giving me valuable suggestions. I thank Guido Schäfer and Rene Beier again for the German translation of my thesis abstract. I also thank them for all the memorable moments we had at MPI. I am grateful to Narahari Rao, Sunil Chandran, Venkatesh Srinivasan, Surendar Baswana, Kavitha Telikepalli, Rahul Ray, Piotr Krysta, Anil Kumar, Tobias Polzin, Sai Pradeep and Venkata Krishna for their companionship and the support they gave me. I thank Venkatesh, Sunil and Surendar again for their lessons on approximation algorithms, graph theory and probabilistic techniques. I once again thank Narahari Rao for all the wonderful discussions we had about earth,water,life,people,art,history, institution,culture,... I made many new friends after coming to Saarbrücken. I thank them all

for their warmth and friendship.

I am indebted to my parents, Sivadasan and Rema Devi, for their love, understanding and belief in me. I dedicate this thesis to them.

I also thank my elder brother Praveen Sivadasan, my teachers and all my friends for shaping my scientific career either directly or indirectly. I also thank all my family members and relatives, especially Rasmi, Remya, my in-laws, Kaladharan, Gopinathan, Divakaran, for their love and affection.

Beyond words, I am grateful to Renjini for making this venture possible. As always, I am indebted to her for her unconditional love and support, and for being with me all along...

Contents

Chapter 1

Introduction

Problems that are *adversarial* in nature form an important part of the algorithmic problems. In such problems, there are two parties involved, an algorithm and an adversary. Such problems can be usually viewed as a *game* between the algorithm and the adversary, where the adversary always tries to force the algorithm into worst-case scenarios during its execution. Many real world problems can be represented more appropriately in such a framework. For example, such a framework is suitable for modeling situations with uncertainty or lack of information, which occur quite often in practice. In this thesis, we investigate three such problems.

Online Problems. Following Ben-David et al. [BDBK$^+$94], online problems can be defined as a *request-answer* game between an adversary and an algorithm. The adversary generates a sequence of requests $\sigma = \sigma(1), \ldots, \sigma(m)$, where each request $\sigma(i)$, $i = 1, \ldots, m$, belongs to a set of possible requests R. An *online algorithm* in response serves (answers) the requests one at a time. While serving request $\sigma(i)$, the online algorithm does not know the later requests. In serving the request sequence σ, the online algorithm incurs a non-negative cost and the objective is to minimize this cost.

For example, consider the following scheduling problem on a set of identical machines. A sequence of jobs, each with a possibly different size, arrive in an online fashion. Each job has to be assigned to exactly one machine. The objective is to minimize the maximum total size of jobs assigned to a machine. Once a job is assigned to a machine, the decision cannot be revoked. The offline version of this problem is to come up with the best schedule if the set of all jobs with respective sizes are known before-hand. In the online setting, the achievable performance is not determined by limited computing power but mainly by the lack of information about parts of the input that will only be revealed in the future. Many well-known problems like paging, list update, metrical task systems, load balancing etc., can be formulated in this framework. We refer to [Alb03] for a survey on online problems and algorithms.

To analyze the performance of the online algorithm in such a framework, Sleator and Tarjan [ST85], Karlin et al. [KMRS88] proposed a new comparison measure called *competitive analysis*: compare the cost incurred by the online algorithm for serving σ to the cost of an

optimal offline algorithm, which knows the entire sequence σ in advance. Formally, let $\mathrm{alg}(\sigma)$ and $\mathrm{opt}(\sigma)$ respectively denote the cost incurred by an online algorithm A and the offline optimum to process σ. The algorithm A is called c–competitive, if there exists a constant a such that

$$\mathrm{alg}(\sigma) \;\leq\; c \cdot \mathrm{opt}(\sigma) + a$$

for all request sequences σ. The *competitive factor* or the *competitive ratio* of A is simply the minimum c such that A is c–competitive. Equivalently, if the algorithm A is randomized then A is c–competitive in the *expected* sense if

$$E\big[\mathrm{alg}(\sigma)\big] \;\leq\; c \cdot \mathrm{opt}(\sigma) + a$$

Randomized strategies give rise to different adversarial notions like *adaptive adversary* and *oblivious adversary*. An adaptive adversary generates the next request based on all the random choices made by the algorithm so far. An oblivious (weak) adversary does not know the random choices made by the algorithm, it only knows the specification of the algorithm. In other words, the algorithm has a secure random source which is not visible to the adversary.

Search Games. Yet another adversarial framework that we consider in this thesis is the problem of searching/finding an evader (adversary) by a set of pursuers, also called the *pursuit-evasion* problem. Search games have a long history in the fields of game theory and algorithms [Isa65, Gal80]. There are many variants of such games studied, like a game played on a simple polygon, both players having specified amount of visibility, players moving on a closed curve, clearing an evader (like a poisonous gas) from a system of tunnels, etc. There are also basic graph parameters like *search number* of a graph that is associated with the number of pursuers needed to find an evader in a deterministic setting. We specifically focus on a *two-player game* played on a graph between a *pursuer* and an *evader*. The pursuer is trying to catch the evader, while they both travel from node to node of a connected, undirected graph. Both players may use a randomized strategy where each player has a secure source of randomness which cannot be observed by the other player. That is, both players are oblivious. The objective of the pursuer is to minimize the expected number of rounds until the evader is caught.

Outline of the thesis. In this thesis, we present results for three problems. In Chapters 2 and 3, we focus on two online problems. In the third chapter we investigate a pursuit-evasion game on graphs. Each chapter is mostly self-contained.

Chapter 2: Online scheduling with bounded migration. In this chapter, we consider a classical online scheduling problem where jobs that arrive one by one are assigned to identical parallel machines with the objective of minimizing the maximum machine load, also known as *makespan*. We generalize this problem by allowing the current assignment to be changed whenever a new job arrives, subject to the constraint that the total size of moved jobs is bounded by β times the size of the arriving job. Observe that if $\beta = 0$ then it is the classical online setting, where decisions once made cannot be revoked, and if $\beta \to \infty$ then it is the offline setting.

For the classical online setting ($\beta = 0$), a series of results [BFKV95, KPT96, Alb99] improve the achievable competitive ratio from 2, initially achieved by Graham [Gra66], to 1.9201, which is the best known result to date due to Fleischer and Wahl [FW00]. Already a lower bound of 1.88 (1.58) on the achievable competitive ratio for deterministic (randomized) algorithms is known. For the offline case, a linear time *polynomial time approximation scheme* (PTAS), that is, a family of polynomial time approximation algorithms with performance guarantee within a factor $1 + \epsilon$ of the optimum for all fixed $\epsilon > 0$ is known.

Our main result is a linear time 'online approximation scheme', that is, a family of online algorithms with competitive ratio $1 + \epsilon$ and constant migration factor $\beta(\epsilon)$, for any fixed $\epsilon > 0$. This result is of particular importance if considered in the context of sensitivity analysis: While a newly arriving job may force a complete change of the entire structure of an optimal schedule, only very limited 'local' changes suffice to preserve near-optimal solutions. We believe that this concept will find wide application in its own right. We also present simple deterministic online algorithms with migration factors $\beta = 2$ and $\beta = 4/3$, respectively. Their competitive ratio $3/2$ beats the lower bound on the performance of any online algorithm in the classical setting without migration. Furthermore, with migration factor $\beta = 4$, the competitive ratio further drops down to $4/3$. For two machines we obtain tight competitive ratio of $7/6$ already for migration factor 1. We also present improved algorithms and similar results for closely related problems. In particular, we consider the objective of maximizing the minimum load of a machine.

Chapter 3: Smoothed competitiveness of metrical task systems. In this chapter we present a probabilistic analysis of a standard text book algorithm for a very well-known online problem called *metrical task systems*. Metrical task systems can be described as follows. 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.

Borodin, Linial and Saks [BLS92] presented a deterministic *work function algorithm* (WFA) for metrical task systems having a *tight* competitive ratio of $2n - 1$. However, the competitive ratio often is an over-pessimistic estimation of the true performance of an online

algorithm. Several other attempts were made in the past to overcome this problem: by allowing limited lookahead for the algorithm or by considering restricted adversarial models like access graphs or diffuse adversary. Spielman and Teng [ST01] recently proposed a new complexity measure, called *smoothed complexity*, which is a hybrid between average-case and worst-case complexity. Becchetti et al. [BLMS$^+$03] recently extended this idea to competitive analysis and proposed *smoothed competitive analysis* as an alternative to worst-case competitive analysis of online algorithms. The idea is to randomly perturb, or *smoothen*, an adversarial input instance and to analyze the performance of the algorithm on the perturbed instances. Such a process can be equivalently viewed as drawing an instance probabilistically from the neighborhood of the adversarial input instance and estimating the average performance of the algorithm in this neighborhood. An improved average performance in this neighborhood implies that the worst-case instances are 'sparse', i.e., isolated peaks in the instance/performance space.

We present a *smoothed competitive analysis* of WFA. We smoothen the adversarial request costs and analyze the performance of WFA. 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. Assuming that the ratio between the maximum and the minimum edge length of G is bounded by a constant, the smoothed competitive ratio of WFA becomes $O(\mathrm{diam}(U_{\min}/\sigma + \log(D)))$ and $O(\sqrt{n \cdot (U_{\min}/\sigma + \log(D))})$, where σ denotes the standard deviation of the smoothing distribution. For example, already for perturbations with $\sigma = \Theta(U_{\min})$ the competitive ratio reduces to $O(\log n)$ on a clique or a complete binary tree and to $O(\sqrt{n})$ on any constant degree graph. We also prove that for a large class of graphs these bounds are asymptotically tight. Our analysis holds for various probability distributions, including the uniform and the normal distribution. 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 $\sigma = \Theta(U_{\min})$. Thus our analysis gives a strong indication that the asymptotic competitive ratio of WFA is much better than $\Theta(n)$ for a large class of inputs.

Chapter 4: Randomized pursuit-evasion game on graphs. In this chapter, we analyze a randomized pursuit-evasion game on graphs. This game is played by two players, a *pursuer* and an *evader* (adversary). We also refer to them as *hunter* and *rabbit* respectively. Let G be any connected, undirected graph with n nodes. The game is played in rounds and in each round both the hunter and the rabbit are located at a node of the graph. Between rounds both the hunter and the rabbit can stay at the current node or move to another node. They do not 'see' each other unless they meet in the same node. The hunter is assumed to be *restricted* to the graph G: in every round, the hunter can move using at most one edge. For the rabbit we investigate two models: in one model the rabbit is restricted to the same graph as the hunter, and in the other model the rabbit is *unrestricted*, i.e., it can jump to an arbitrary node in every round.

We say that the rabbit is *caught* as soon as hunter and rabbit are located at the same node in a round. The goal of the hunter is to catch the rabbit in as few rounds as possible, whereas the rabbit aims to maximize the number of rounds until it is caught. Given a randomized hunter strategy for G, the *escape length* for that strategy is the worst-case expected number of rounds it takes the hunter to catch the rabbit, where the worst-case is with regards to all (possibly randomized) rabbit strategies.

The problem we address is motivated by the question of how long it takes a single pursuer to find an evader on a given graph that, for example, corresponds to a computer network or to a map of a terrain in which the evader is hiding. Such search games have a long history in the field of game theory. Our problem is a discrete version of the *Princess-Monster* game introduced in 1965 [Isa65]. But the adversary that we consider is more powerful (can take short-cuts). Considerable amount of research was also done on the geometric version of the pursuit-evasion problem. *Deterministic* pursuit-evasion games in graphs are also well-studied [LaP93, KP86, MHG$^+$88]. In the area of mobile ad-hoc networks, such pursuit-evasion models are used to design communication protocols [CNS01]. A hunter strategy based on random walks was first studied in [AKL$^+$79]. It was shown that the hunter catches an unrestricted rabbit within $O(nm^2)$ rounds, where n and m denote the number of nodes and edges, respectively.

One of our main results is a hunter strategy for general graphs with an escape length of only $O(n \log(\mathrm{diam}))$ against restricted as well as unrestricted rabbits, where diam is the diameter of G. This bound is close to optimal since $\Omega(n)$ is a trivial lower bound on the escape length in both models. Furthermore, we prove that our upper bound is optimal up to constant factors against unrestricted rabbits. We show a non-standard random walk for the rabbit such that, for any positive integers n and $d < n$, there is a graph G with n nodes and diameter d, such that the escape length is $\Omega(n \log(d))$ for any hunter strategy against this random-walk based rabbit. Furthermore, we show using a different hunter strategy that on special graphs like complete binary tree on n nodes, the escape length is only $O(n)$. Finally, we also discuss the case of strongly connected graphs and show a hunter strategy with tight escape length of $O(n^2)$.

Discussions and open problems related to each of these problems can be found at the end of the respective chapters. We assume that the reader is familiar with the basics of probability theory and randomized algorithms, which can, for instance, be found in [MR95]. For an introduction to online algorithms, we refer to the book by Borodin and El-Yaniv [BEY98], and the survey articles by [Alb03, Sga98]. For a detailed account of linear and integer programming theory we refer to the books [Sch86, NW88]. We refer to [Hoc96a] for an introduction to approximation algorithms in combinatorial optimization problems, including scheduling problems. An introduction to the basic complexity theory can be found in the books [GJ79, Pap94]. Definitions of some of the basic concepts used in this thesis can be found at the end of this thesis.

Chapter 2

Online Scheduling with Bounded Migration

2.1 Introduction

One of the most fundamental scheduling problems asks for an assignment of jobs to m identical parallel machines so as to minimize the makespan. The makespan is the completion time of the last job that finishes in the schedule; it also equals the maximum machine load. In the standard classification scheme of Graham, Lawler, Lenstra and Rinnooy Kan [GLLR79], this scheduling problem is denoted by $\mathrm{P}\,||C_{\max}$ and it is well-known to be strongly NP-hard [GJ78], i.e., it is still NP-hard even if all numbers appearing in the input are bounded by some polynomial in the length of the input.

The *offline* variant of this problem assumes that all jobs are known in advance whereas in the *online* variant the jobs are incrementally revealed by an adversary and the online algorithm can only choose the machine for the new job without being allowed to move other jobs. Note that dropping this radical constraint on the online algorithm yields the offline situation.

A new online scheduling paradigm. We study a natural generalization of both offline and online problems. Jobs arrive incrementally but, upon arrival of a new job j, we are allowed to migrate *some* previous jobs to other machines. The total size of the migrated jobs however must be bounded by βp_j where p_j is the size of the new job. For *migration factor* $\beta = 0$ we get the online setting and for $\beta = \infty$ we get the offline setting.

For a number of offline optimization problems, a PTAS, i.e., a family of polynomial time approximation algorithms with performance guarantee $1 + \epsilon$ for all fixed $\epsilon > 0$ is known. In contrast to the offline approximation results, the achievable competitive ratios in online settings are not determined by limited computing power but by the lack of information about parts of the input that will only be revealed in the future. As a consequence, for all interesting classical online problems it is rather easy to come up with lower bounds that create a gap between the best possible competitive ratio α and 1. In particular, it is usually impossible to

Publication Notes. A preliminary version of this joint work, together with Peter Sanders and Martin Skutella, appeared in the proceedings of the 31st International Colloquium on Automata, Languages, and Programming (ICALP), 2004 [SSS04].

construct a family of $(1+\epsilon)$-competitive online algorithms for such problems.

2.2 Related Work

For the online machine scheduling problem, Graham's *list scheduling* algorithm keeps the makespan within a factor $2-1/m$ of the offline optimum [Gra66]: Schedule a newly arriving job on the least loaded machine. It can also easily be seen that this bound is tight: adversarial sequence consists of $m(m-1)$ jobs of size $\frac{1}{m}$ followed by one job of size 1. The optimal makespan in this case is 1.

For the offline setting, Graham showed three years later that sorting the jobs in the order of non-increasing size before feeding them to the list scheduling algorithm yields an approximation algorithm with performance ratio $4/3-1/(3m)$ [Gra69]. Later, exploiting the relationship between the machine scheduling problem under consideration and the binpacking problem, algorithms with improved approximation ratios have been obtained in a series of works [CGJ78, Fri84, Lan81].

Finally, polynomial time approximation schemes for a constant number of machines and for an arbitrary number of machines are given in [Gra69, Sah76] and by Hochbaum and Shmoys [HS87], respectively. The latter PTAS partitions jobs into large and small jobs. The sizes of large jobs are rounded such that an optimum schedule for the rounded jobs can be obtained via dynamic programming. The small jobs are then added greedily using Graham's list scheduling algorithm. This approach can be refined to an algorithm with linear running time (see, e.g., [Hoc96b]): replace the dynamic program with an integer linear program on a fixed number of variables and constraints which can be solved in constant time [Len83].

In a series of papers, increasingly complicated online algorithms with better and better competitive ratios beating the Graham bound 2 have been developed [BFKV95, KPT96, Alb99]. The best result known to date is a 1.9201-competitive algorithm due to Fleischer and Wahl [FW00]. The best lower bound 1.88 on the competitive ratio of any deterministic online algorithm currently known is due to Rudin [Rud01]. For randomized online algorithms there is a lower bound of $e/(e-1) \approx 1.58$ [CvVW94, Sga97]. For more results on online algorithms for scheduling we refer to the recent survey articles by Albers [Alb03] and Sgall [Sga98].

Strategies that reassign jobs were studied in the context of online load balancing, jobs arrive in and depart from a system of m machines online and the scheduler has to assign each incoming job to one of the machines. Deviating from the usual approach of comparing against the optimal *peak load* seen so far, Westbrook [Wes00] introduced the notion of competitiveness against *current load*: An algorithm is α–competitive if after every round the makespan is within α factor of the optimal makespan for the current set of jobs. Each incoming job u has size p_u and reassignment cost r_u. For a job, the reassignment cost has to be paid for its initial assignment and then every time it is reassigned. Observe that the optimal strategy has to pay this cost once for each job for its initial assignment. Thus the optimal (re)assignment cost S is simply the sum of reassignment costs of all jobs scheduled till now. Westbrook showed

a 6-competitive strategy for identical machines with reassignment cost $3S$ for proportional reassignments, i.e., r_u is proportional to p_u, and $2S$ for unit reassignments, i.e., $r_u = 1$ for all jobs. Later Andrews et al. [AGZ99] improved it to 3.5981 with the same reassignment factors. They also showed $3 + \epsilon$ and $2 + \epsilon$ competitive strategies respectively for the proportional and unit case, the reassignment factor depending only on ϵ. For arbitrary reassignment costs they achieve 3.5981 competitiveness with 6.8285 reassignment factor. They also present a 32–competitive strategy with constant reassignment factor for related machines. Job deletions is an aspect that we do not consider in our work, our focus is primarily on achieving competitive ratios close to 1. Our results can also be interpreted in the framework of online load balancing with proportional reassignments and without job deletions. We show strategies with better competitive ratios, at the same time achieving reassignment factor strictly less than three. We also show $(1+\epsilon)$–competitive strategies, for any $\epsilon > 0$, with constant reassignment factor $f(\epsilon)$. Our results are also stronger in the sense that a strategy with reassignment factor β ensures that when a job u arrives, the total reassignment cost incurred (for scheduling it) is at most βr_u. This is different from the more relaxed constraint that after t rounds, the total reassignment cost incurred is at most $\beta \sum r_u$ (summing over all jobs seen till round t). Most of our strategies are *robust*, they convert *any* α–competitive schedule to an α–competitive schedule after assigning the newly arrived job, whereas in [Wes00, AGZ99] it is required that the schedule so far is carefully constructed in order to ensure the competitiveness after assigning/deleting a job in the next round.

2.3 Our Contribution

In Section 2.5 we describe a simple online algorithm which achieves approximation ratio $3/2$ using a moderate migration factor $\beta = 2$. Notice that already this result beats the lower bound 1.88 (1.58) on the competitive ratio of any classical (randomized) online algorithm without migration. Using a more sophisticated analysis, the migration factor can be decreased to $4/3$ while maintaining competitive ratio $3/2$. On the other hand we show that our approach does not allow for migration factor 1 and competitive ratio $3/2$. Furthermore, an improved competitive ratio $4/3$ can be achieved with migration factor 4. For two machines, we can achieve competitive ratio $7/6$ with a migration factor of one. This ratio is tight for migration factor one.

In Section 2.8 we discuss an application of bounded migration to configuring storage servers. This was the original motivation for our work. In this application, the objective is to maximize the minimum load. It is well-known [AE98] that any online deterministic algorithm for this *machine covering problem* has competitive ratio at least m (the number of machines). There is also a lower bound of $\Omega(\sqrt{m})$ for any randomized online algorithm. We develop a simple deterministic online strategy which is 2-competitive already for migration factor $\beta = 1$.

Our main result can be found in Section 2.6. We present a family of online algorithms with

competitive ratio $1+\epsilon$ and constant migration factor $\beta(\epsilon)$, for any fixed $\epsilon > 0$. On the negative side, no constant migration factor suffices to maintain competitive ratio one, i.e., optimality. We provide interpretations of these results in several different contexts:

Online algorithms. Online scheduling with bounded job migration is a relaxation of the classical online paradigm. Obviously, there is a tradeoff between the desire for high quality solutions and the requirement to compute them online, that is, to deal with a lack of information. Our result can be interpreted in terms of the corresponding tradeoff curve: Any desired quality can be guaranteed while relaxing the online paradigm only moderately by allowing for a constant migration factor.

Sensitivity analysis. Given an optimum solution to an instance of an optimization problem and a slightly modified instance, can the given solution be turned into an optimum solution for the modified instance without changing the solution too much? This is the impelling question in sensitivity analysis. As indicated above, for the scheduling problem under consideration one has to answer in the negative. Already one additional job can change the entire structure of an optimum schedule. However, our result implies that the answer is positive if we only require near-optimum solutions.

Approximation results. Our result yields a new PTAS for the scheduling problem under consideration. Due to its online background, this PTAS constructs the solution incrementally. That is, it reads the input little by little always maintaining a $(1+\epsilon)$-approximate solution. Indeed, it follows from the analysis of the algorithm that every update only takes constant time. In particular, the overall running time is linear and thus matches the previously best known approximation result.

We believe that each of these interpretations constitutes an interesting motivation for results like the one we present here in its own right and can therefore lead to interesting results for many other optimization problems.

The underlying details of the presented online approximation scheme have the same roots as the original PTAS by Hochbaum and Shmoys [HS87] and its refinements [Hoc96b]. We distinguish between small and large jobs; a job is called large if its size is of the same order of magnitude as the optimum makespan. Since this optimum can change when a new job arrives, the classification of jobs must be updated dynamically. The size of every large job is rounded such that the problem of computing an optimum schedule for the subset of large jobs can be formulated as an integer linear program of constant size. A newly arriving job causes a small change in the right hand side of this program. This enables us to use results from sensitivity analysis of integer programs in order to prove that the schedule of large jobs needs to be changed only slightly. Our PTAS is very simple, it uses only this structural result and does not use any algorithms from integer programming theory.

2.4 Preliminaries

Let the set of *machines* be denoted by $M = \{1, \ldots, m\}$. The set of *jobs* is $\{1, \ldots, n\}$ where job j arrives in round j. Let p_j denote the positive *processing time* or the *size* of job j. For a subset of jobs N, the *total processing time* of jobs in N is $p(N) := \sum_{j \in N} p_j$; let $p_{\max}(N) := \max_{j \in N} p_j$. For a schedule on the set of jobs N, let $S(i)$ denote the set of *jobs scheduled on machine* i. For a subset of machines $Y \subseteq M$, let $S(Y) := \bigcup_{i \in Y} S(i)$. For a subset of jobs N, let $\mathrm{opt}(N)$ denote the *optimal makespan*. If the subset of jobs N and a newly arrived job j are clear from the context, we sometimes also use the shorter notation $\mathrm{opt} := \mathrm{opt}(N)$ and $\mathrm{opt}' := \mathrm{opt}(N \cup \{j\})$. It is easy to observe that $\mathrm{lb}(N) := \max\{p(N)/m, p_{\max}(N)\}$ is a lower bound on $\mathrm{opt}(N)$ satisfying

$$\mathrm{lb}(N) \;\leq\; \mathrm{opt}(N) \;\leq\; 2\mathrm{lb}(N) \;. \tag{2.1}$$

The following well-known fact is used frequently in the subsequent sections.

Observation 1. *For a set of jobs N, consider an arbitrary schedule with makespan κ. Assigning a new job j to the least loaded machine yields a schedule with makespan at most $\max\{\kappa, \mathrm{opt}(N \cup \{j\}) + (1 - 1/m)p_j\}$.*

Proof. We need to show that if the makespan changes after scheduling job j, then the new makespan is at most $\mathrm{opt}(N \cup \{j\}) + p_j(1 - 1/m)$. Since job j is scheduled on the least loaded machine, the new makespan is at most $p(N)/m + p_j$. This combined with the following inequality yields the fact; $\mathrm{opt}(N \cup \{j\}) \geq p(N \cup \{j\})/m = p(N)/m + p_j/m$. □

2.5 Strategies with Small Migration Factor

We consider the problem of scheduling jobs arriving one after another on m parallel machines so as to minimize the makespan. We first show a very simple $3/2$-competitive algorithm with migration factor 2. The algorithm is as follows:

Procedure FILL_1:
Upon arrival of a new job j, choose one of the following two options minimizing the resulting makespan.

Option 1: Assign job j to the least loaded machine.

Option 2: Let i be the machine minimizing the maximum job size. Repeatedly remove jobs from this machine; stop before the total size of removed jobs exceeds $2p_j$. Assign job j to machine i. Assign the removed jobs successively to the least loaded machine.

Theorem 1. *Procedure* FILL_1 *is $\left(\frac{3}{2} - \frac{1}{2m}\right)$-competitive with migration factor 2.*

Proof. From the description of FILL_1, it is clear that the migration factor is at most 2. In order to prove competitiveness, we consider an arbitrary $\left(\frac{3}{2}-\frac{1}{2m}\right)$-approximate schedule for a set of jobs N and show that incorporating a new job j according to FILL_1 results in a new schedule which is still $\left(\frac{3}{2}-\frac{1}{2m}\right)$-approximate. In the following, a job is called *small* if its processing time is at most $\mathrm{opt}'/2$, otherwise it is called *large*. If the new job j is small, then the first option yields makespan at most $\left(\frac{3}{2}-\frac{1}{2m}\right)\mathrm{opt}'$ by Observation 1. Thus we can assume from now on that j is large.

Since there can be at most m large jobs in $N \cup \{j\}$, all jobs on the machine chosen in the second option are small. Thus, after removing jobs from this machine as described above, the machine is either empty or the total size of removed jobs exceeds the size of the large job j. In both cases, assigning job j to this machine cannot increase its load above $\left(\frac{3}{2}-\frac{1}{2m}\right)\mathrm{opt}'$. Thus, using the same argument as above, assigning the removed small jobs successively to the least loaded machine yields again a $\left(\frac{3}{2}-\frac{1}{2m}\right)$-approximate schedule. □

Next we show that the migration factor can be decreased to $4/3$ without increasing the competitive ratio above $3/2$. This result is achieved by carefully modifying FILL_1.

Procedure FILL_2 :
Upon arrival of j, choose the one of the following $m+1$ options that minimizes the resulting makespan. (Break ties in favor of option 0.)

Option 0: Assign job j to the least loaded machine.

Option i [for $i \in \{1,\ldots,m\}$]: Ignoring the largest job on machine i, consider the remaining jobs in the order of non-increasing size and repeatedly remove them from the machine; stop before the total size of removed jobs exceeds $\frac{4}{3}p_j$. Assign job j to machine i. Assign the removed jobs successively to the least loaded machine.

Theorem 2. *Procedure* FILL_2 *is* $\frac{3}{2}$*-competitive with migration factor* $\frac{4}{3}$.

Proof. The migration factor is clear from the description of FILL_2 . To show the competitive ratio, we consider an arbitrary $\frac{3}{2}$–approximate schedule of N, also denoted as *input schedule*, that additionally satisfies the following property; the total load on any machine excluding its largest job is at most opt. We remark that this is not an unreasonable requirement on the input schedule as even the list scheduling algorithm ensures this. We show that incorporating the new job j results in a $\frac{3}{2}$–approximate schedule. As shown by the following fact, the resulting schedule also satisfies the above additional property.

Fact 1. *After scheduling job* j*, the total load on any machine excluding its largest job is at most* opt'.

Proof. Let M' be the subset of machines 'touched' by FILL_2 for scheduling job j. It suffices to show that the total load in such a machine excluding its 'last' job (the job that entered last) is at most opt'. Fix any machine in M' and consider its last job. If it is not j then it was

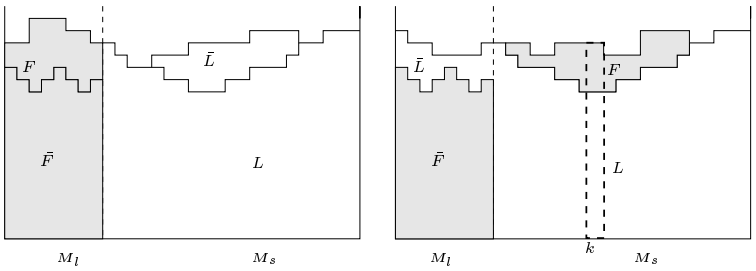


Fig. 2.1: Figure showing the differences between the optimal schedule and the input schedule of jobs in N. The sets $F, \bar{F}, L$ and $\bar{L}$ capture these differences. The 'common sets' in both schedules are $\bar{F}$ and L. On both schedules, exactly the machines in M_l contain one *large* job each, and such jobs belong to $\bar{F}$.

assigned as part of the redistribution phase. Since redistribution is always performed on the current least loaded machine, the load in this machine excluding this last job is at most opt'. If the last job is j then, as option 0 is always preferred, the total load of this machine can only be smaller than the load in the least loaded machine of the input schedule together with the size of job j. Thus the fact follows. □

We distinguish three cases depending on p_j. If $p_j \leq \mathrm{opt}'/2$ then option 0 already yields a schedule of makespan at most $\frac{3}{2}\mathrm{opt}'$ by Observation 1.

Case $\mathrm{opt}'/2 < p_j \leq \frac{3}{4}\mathrm{opt}'$: let

$$p_j = \mathrm{opt}'/2 + \delta \quad \text{where} \quad 0 < \delta \leq \mathrm{opt}'/4 \tag{2.2}$$

To handle this case, it suffices find a $k \in \{1, \ldots, m\}$ such that Option k yields a $\frac{3}{2}$–approximate schedule. We call it a 'good' option. We fix k as follows. Denote a job as *large* if its size is more than $\mathrm{opt}' - \delta$ and *small* otherwise. With respect to the input schedule, partition the set of machines M as machines with only small jobs denoted as M_s, and the rest denoted as M_l. That is,

$$M_s = \{i \in M \mid p_{\max}(S(i)) \leq \mathrm{opt}' - \delta\} \quad \text{and} \quad M_l = M - M_s \tag{2.3}$$

Observe that on any $\frac{3}{2}$–approximate schedule of N, a machine has at most one large job. Hence there are $|M_l|$ large jobs; one on each machine in M_l. Consequently, we fix an optimal schedule of N such that large jobs are scheduled on M_l. We now compare the input schedule with this optimal schedule in the following way. As shown in Figure 2.1, partition N as $F, \bar{F}, L, \bar{L}$. In the optimal schedule, the jobs scheduled on M_l and M_s are $F \cup \bar{F}$ and $L \cup \bar{L}$ respectively and, the respective sets in the input schedule are $\bar{F} \cup \bar{L}$ and $F \cup L$. It follows that in the input schedule there is a machine in M_s with total load at most opt with respect to L.

We fix k as this machine. That is,

$$p(S(k) - F) \;=\; p(S(k) \cap L) \;\leq\; \mathrm{opt} \tag{2.4}$$

It remains to show that option k is good.

By the choice of optimal schedule, every large job belongs to the 'common set' $\bar{F}$. Hence each job in set F has size at most δ; as it is scheduled along with a large job (from $\bar{F}$), in the optimal schedule. As a consequence, the set of jobs in machine k with each having size at most δ includes the subset of jobs induced by F. That is, if we partition the set $S(k)$ as K_δ and K_r, where

$$K_\delta = \{\text{All jobs in machine } k \text{ with size at most } \delta\}; \quad K_r = S(k) - K_\delta$$

then

$$p(K_r) \;\leq\; \mathrm{opt} \qquad \big(\text{as } F \subseteq K_\delta \text{ and } p(S(k) - F) \leq \mathrm{opt} \text{ by (2.4)}\big) \tag{2.5}$$

This readily implies that there is at most one job with size greater than $\mathrm{opt}'/2$ in machine k; as such jobs belongs to set K_r. Consequently, every job removed during option k has size at most $\mathrm{opt}'/2$; as the largest job is untouched. This ensures by Observation 1 that, reassignment of the removed jobs still yields a $\frac{3}{2}$–approximate schedule, if the schedule before reassignment is $\frac{3}{2}$–approximate. Hence we are done with this case analysis if we show that the total load of machine k *before redistribution*, i.e., after removal of jobs and assignment of job j, is at most $\frac{3}{2}\mathrm{opt}'$. That is, it is enough to ensure that for machine k,

$$p(\text{initial jobs}) - p(\text{removed jobs}) + p_j \;\leq\; \frac{3}{2}\mathrm{opt}' \tag{2.6}$$

The interesting case is when the set of removed jobs excludes more jobs in addition to the largest job; the largest job has size at most $\mathrm{opt}' - \delta$ (recall that machine k belongs to M_s). Even then, it is not a problem if an unremoved job belongs to K_δ as this readily implies that the total size of removed jobs is at least $\frac{4}{3}p_j - \delta \geq p_j$, where $\frac{4}{3}p_j$ is the total removal limit. The remaining situation is that every unremoved job (apart from the largest job) is from K_r. An unremoved job from K_r, by the greedy removal, immediately implies that there is a removed job from K_r. This is because, (a) the removal limit, i.e. $\frac{4}{3}p_j$, is at least $\mathrm{opt}'/2$ and (b) all jobs except the largest job have size at most $\mathrm{opt}'/2$ in machine k. Thus $p(\text{removed jobs}) \geq p(K_\delta) + \delta$, which in turn satisfies (2.6). The additional δ accounts for the removed job from K_r.

Case $p_j > \frac{3}{4}\mathrm{opt}'$: let

$$p_j \;=\; \frac{3}{4}\mathrm{opt}' + \delta \quad \text{where} \quad \delta \;\leq\; \mathrm{opt}'/4 \tag{2.7}$$

Since there can be at most m jobs in $N \cup \{j\}$ each with size greater than $\mathrm{opt}'/2$, there is a machine k in the input schedule where all jobs have size at most $\mathrm{opt}'/2$. We show that option k is good. Observe that the reassignment of removed jobs yield $\frac{3}{2}\mathrm{opt}'$ makespan schedule by Observation 1, if the schedule before has $\frac{3}{2}\mathrm{opt}'$ makespan. Recall that in the input schedule, on any machine, the load excluding its largest job is at most opt. Hence all jobs, except the largest job, are removed and thus yielding a total load of at most $\frac{3}{2}\mathrm{opt}'$ after assigning job j. This concludes the proof. □

Robustness

Most of our scheduling strategies for minimizing the makespan discussed in this chapter are robust in the following sense. The only invariant that we require in their analyses is that before the arrival of a new job the current schedule is α-approximate. Job j can then be incorporated yielding again an α-approximate schedule. In other words, we do not require that the current schedule is carefully constructed so far, to maintain the competitiveness in the next round.

2.5.1 Negative Results

Theorems 1 and 2 raise the question of which migration factor is really necessary to achieve competitive ratio $3/2$. We can prove that any robust strategy needs migration factor greater than 1 in order to maintain competitive ratio $3/2$.

Lemma 1. *There exists a $3/2$-approximate schedule such that, upon arrival of a particular job, migration factor 1.114 is needed to achieve $3/2$-competitiveness. Moreover, migration factor 1 only allows for competitive ratio 1.52 in this situation.*

Proof. The situation is depicted in Figure 2.2. There are 37 machines. Machines 1 to 8 are identically packed. Each of them has one job of size 84 and three jobs of size 4. Machines 9 to 12 are also identical. Each of them has one job of size 68 and two jobs of size 16. Machine 13 contains four jobs of size 32. Machines 14 to 37 contain one job of size 96 each. The size of the newly arriving job is 86.1. The optimal makespan is 100. To achieve a makespan of at most 150, it is necessary to migrate at least 3 jobs of size 32 from machine 13 to other machines. Hence, the migration factor is at least $96/86.1 > 1.114$. To show the lower bound on the competitiveness (the second part of the lemma), we set the size of the newly arriving job to 88 instead of 86.1. It is straightforward to check that the final optimal makespan is still 100 and the best possible makespan achievable for any strategy with migration factor $\beta \leq 1$ is 152. □

An additional feature of FILL_1 and FILL_2 is that they are *local* in the sense that they migrate jobs only from the machine where the newly arrived job is assigned to. There is a class of optimal schedules for which, upon arrival of a new job, it is not possible to achieve a better competitive ratio than $3/2$ using only local migration. This holds even if an arbitrary migration factor is allowed. The following optimal schedule on m machines, upon the arrival

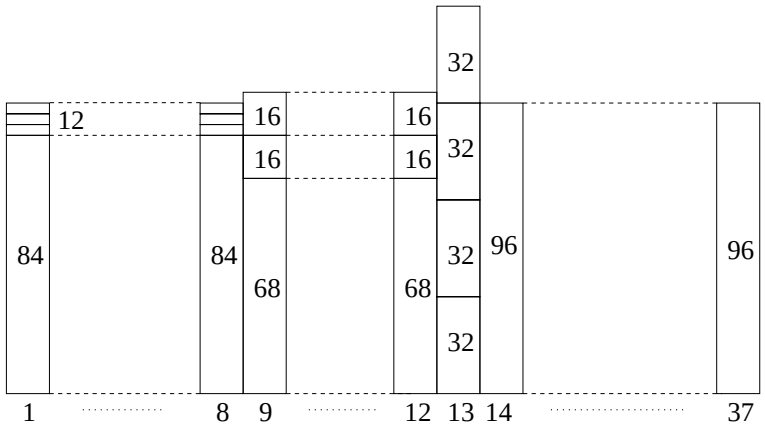


Fig. 2.2: A $\frac{3}{2}$-approximate schedule ($m = 37$). If a new job of size 86.1 arrives, jobs of total size at least 96 have to be moved in order to construct a schedule that is still $\frac{3}{2}$-approximate.

of a new job, enforces a competitive ratio of at least $3/(2+\frac{2}{m})$ for any amount of migration. This bound converges to $3/2$ for large m. The example looks as follows: Machines 1 and 2 each contain one job of size $1/2$ and $m/2$ jobs of size $1/m$. All other machines contain a single job of size 1. The newly arriving job has size 1. The optimum makespan is $1 + 1/m$ and the makespan achievable by any local strategy is $3/2$ (by scheduling the new job on say, machine 1 and migrating all small jobs to other machines).

2.5.2 A $\frac{4}{3}$–Competitive Strategy

In this section, we show that an improved competitive ratio of $\frac{4}{3}$ can be achieved by a more sophisticated algorithm, FILL_3, with migration factor 4.

Procedure FILL_3:
Upon arrival of j, choose one of the following $m + 1$ options that minimizes the resulting makespan.

Option 0: Assign job j to the least loaded machine.

Option i [for $i \in \{1, \ldots, m\}$]: Skip phase one if either $2p_j < p(S(i))$ or $p_j < p_{\max}(S(i))$.

Phase one: Let ℓ denote the largest job in machine i. Remove all jobs from machine i and schedule job j there. Except job ℓ, assign the removed jobs successively in the least loaded machine.

Phase two: We assign the unassigned job ℓ in this phase. If phase one was skipped then ℓ is simply job j. Consider $m + 1$ sub-options and choose the one that minimizes the resulting makespan.

Sub-option 0: Assign job ℓ to the least loaded machine.

Sub-option k [for $k \in \{1, \ldots, m\}$]: Ignoring the largest job in machine k, consider the remaining jobs in the order of non-increasing size and repeatedly remove them; stop before the total size of removed jobs exceeds $2p_j$. Assign job ℓ to machine k and assign the removed jobs successively to the least loaded machine.

Theorem 3. FILL_3 *is* $\frac{4}{3}$*–competitive with factor* 4 *migration.*

Proof. The migration factor is clear from the description of FILL_3. In both phases the migration factor is 2. To show the competitive ratio, we consider any arbitrary *input schedule* on the set of jobs N that is $\frac{4}{3}$–approximate. We show that FILL_3 yields a $\frac{4}{3}$–approximate schedule on $N \cup \{j\}$. We call a job b as either *small*, *medium* or *large* where

$$\text{small}: \ p_b \leq \text{opt}'/3 \qquad \text{medium}: \ \text{opt}'/3 \ < p_b \ \leq \frac{2}{3}\text{opt}' \qquad \text{large}: \ p_b > \frac{2}{3}\text{opt}'$$

If job j is small then option 0 yields a $\frac{4}{3}$–approximate schedule by Observation 1.

Case $p_j > \text{opt}'/3$: With respect to the input schedule, we partition the set of machines M as M_L and M_S, where

$$M_L \ = \ \{i \in M \mid \text{machine } i \text{ contains a large job}\} \ \text{ and } \ M_S = M - M_L$$

Since $p_j > \text{opt}'/3$, the number of large jobs in N is at most $m - 1$; otherwise the optimal makespan of $N \cup \{j\}$ exceeds opt'.

Observation 2. *Consider the set of jobs* $A \cup \{b\}$ *with optimal makespan* opt' *on* m *machines. Let* $p_b > \text{opt}'/3$. *If there are* m' *large jobs (*$m' < m$*) in* A *then there are at most* $2(m - m') - 1$ *medium jobs in* A.

Proof. Otherwise the optimal makespan of $A \cup \{b\}$ exceeds opt'. □

Clearly Observation 2 implies that there is a machine z with at most one medium job and no large jobs. Thus

$$p_{\max}(S(z)) \ \leq \ \frac{2}{3}\text{opt}' \tag{2.8}$$

We show that option z is *good*, i.e., it yields a $\frac{4}{3}$–approximate schedule for $N \cup \{j\}$.

By Observation 2, all jobs in machine z, except possibly the largest, are small jobs. Hence reassigning them during phase one after scheduling job j is fine by Observation 1. Thus after phase one of option z, the 'intermediate schedule' is $\frac{4}{3}$–approximate.

We now show that the same holds after phase two of option z. Recall that the job to be assigned in this phase is denoted as ℓ. If phase one was skipped then ℓ is simply job j. Second phase is entered either

– skipping phase one because $2p_j < p(S(z)) \leq \frac{4}{3}\mathrm{opt}$ or $p_j < p_{\max}(S(z)) \overset{(2.8)}{\leq} \frac{2}{3}\mathrm{opt}'$. Job j is same as ℓ for phase two.

– or after phase one. Hence $p_\ell = p_{\max}(S(z)) \overset{(2.8)}{\leq} \frac{2}{3}\mathrm{opt}'$ and $p_\ell \leq p_j$.

In either case it follows that $p_\ell \leq p_j$ and job ℓ is either small or medium. If job ℓ is small then the sub-option 0 yields $\frac{4}{3}$–approximate schedule by Observation 1.

We complete the proof by handling the case that job ℓ is medium. As $p_\ell > \mathrm{opt}'/3$, by Observation 2, in the intermediate schedule after phase one there is a machine z' with at most one medium job and no large jobs. We show that sub-option z' yields a $\frac{4}{3}$–approximate schedule. Let the largest job in machine z' be ℓ'. Since ℓ' is untouched in sub-option z', all the removed jobs are small. Hence by Observation 1, the makespan of schedule after reassignment of removed jobs is $\frac{4}{3}\mathrm{opt}'$ if the schedule before has $\frac{4}{3}\mathrm{opt}'$ makespan. Clearly this is true if all jobs except ℓ' were removed, as both ℓ and ℓ' are non-large jobs. At least one unremoved job in addition to ℓ' also ensures this as the total size of removed jobs in this case is at least $2p_j - \mathrm{opt}'/3$, where $2p_j$ is the removal limit. Thus the total load in machine z' after assigning job ℓ is at most $\frac{4}{3}\mathrm{opt}'$ as $p_j \geq p_\ell$ and $p_\ell > \mathrm{opt}'/3$. □

Even better results are possible for two machines. In section 2.7, we discuss a specialized algorithm with competitive ratio $\frac{7}{6}$ and migration factor of one. We also show that this ratio is tight for any deterministic strategy with migration factor one.

2.6 An Online Approximation Scheme with Constant Migration

The results presented in the last section raise the question how far the competitive ratio for online algorithms with constant migration factor can be decreased. We first prove that optimality (i.e., competitive ratio 1) cannot be achieved. However, for any fixed $\epsilon > 0$ we can get down to competitive ratio $1 + \epsilon$.

Lemma 2. *Any online algorithm computing optimal solutions needs migration factor* $\Omega(m)$.

Proof. Consider a scheduling instance with m machines and $2m - 2$ jobs, two of size i/m for all $i = 1, \ldots, m-1$. Up to permutations of machines, any optimum schedule has the structure depicted in the left part of Figure 2.3. The optimum makespan is $(m-1)/m$. When a new job of size 1 arrives, the optimum makespan increases to 1. Again, the structure of an optimum schedule for the enlarged instance is unique; see the right hand side of Figure 2.3. From each machine in $\{2, \ldots, m-1\}$, at least one job from the pair has to move. Hence the minimum total size of jobs that have to be migrated is at least

$$\frac{1}{m}\sum_{i=1}^{m-2} \min\{i,\ m-1-i\} \ \geq\ \frac{1}{m}\sum_{i=1}^{\lfloor (m-2)/2 \rfloor} i\ ,$$

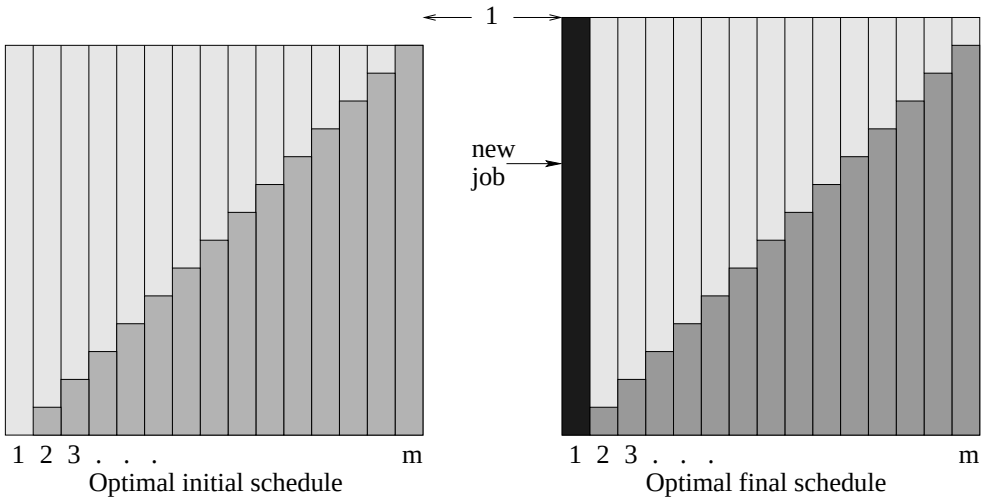


Fig. 2.3: An instance where all machine configurations have to change to maintain optimality.

which is $\Omega(m)$. □

In the following, $\epsilon > 0$ is a fixed constant. We assume without loss of generality that $\epsilon \leq 1$. The following observation belongs by now to the folklore in the field of scheduling; see, e.g., [ABC+99].

Observation 3. *Rounding up each job's processing time to the nearest integer power of* $1+\epsilon$ *increases the makespan of an arbitrary schedule at most by a factor* $1+\epsilon$*. In particular, in specifying a* $(1+O(\epsilon))$*-competitive algorithm we can assume that all processing times are integer powers of* $1+\epsilon$*.*

The current set of jobs is denoted by N. A job in N is called *large* if its processing time is at least $\epsilon \operatorname{lb}(N)$; otherwise, it is called *small*. The subset of large and small jobs is denoted by N_L and N_S, respectively. We partition N into classes N_i, $i \in \mathbb{Z}$, with

$$N_i \;:=\; \{j \in N \mid p_j = (1+\epsilon)^i\} \;.$$

Let $I := \{i \in \mathbb{Z} \mid \epsilon \operatorname{lb}(N) \leq (1+\epsilon)^i \leq p_{\max}(N)\}$ such that $N_L = \bigcup_{i \in I} N_i$. Thus the number of different job sizes for large jobs is bounded by $|I|$ and therefore constant:

$$|I| \;\leq\; 1 + \log_{1+\epsilon} \frac{p_{\max}(N)}{\epsilon \operatorname{lb}(N)} \;\leq\; 1 + \log_{1+\epsilon} \frac{1}{\epsilon} \;\leq\; \frac{2}{\epsilon} \log\left(\frac{1+\epsilon}{\epsilon}\right) \;. \tag{2.9}$$

Given an assignment of jobs N_L to machines, we say that a particular machine obeys *configuration* $k : I \to \mathbb{N}_0$ if, for all $i \in I$, exactly $k(i)$ jobs from N_i are assigned to this machine.

The set of configurations that can occur in any schedule for N_L is

$$\mathbf{K} := \{k : I \to \mathbb{N}_0 \mid k(i) \leq |N_i| \text{ for all } i \in I\} .$$

Up to permutations of machines, an arbitrary schedule for N_L can be described by specifying, for each $k \in \mathbf{K}$, the number y_k of machines that obey configuration k. Conversely, a vector $y \in \mathbb{N}_0{}^{\mathbf{K}}$ specifies a feasible m-machine-schedule for N_L if and only if

$$\sum_{k \in \mathbf{K}} y_k = m \qquad \text{and} \tag{2.10}$$

$$\sum_{k \in \mathbf{K}} k(i)\, y_k = |N_i| \qquad \text{for all } i \in I. \tag{2.11}$$

We denote the set of vectors $y \in \mathbb{N}_0{}^{\mathbf{K}}$ satisfying (2.10) and (2.11) by $\mathbf{S}$. Thus, $\mathbf{S}$ represents the set of all schedules (up to permutations of machines and up to permutations of equal size jobs) for N_L. For a configuration $k \in \mathbf{K}$ let

$$\text{load}(k) := \sum_{i \in I} (1+\epsilon)^i\, k(i)$$

denote the load of a machine obeying configuration k. The makespan of a schedule $y \in \mathbf{S}$ is equal to $\max\{\text{load}(k) \mid y_k > 0\}$. For $\mu \geq 0$, let

$$\mathbf{K}(\mu) := \{k \in \mathbf{K} \mid \text{load}(k) \leq \mu\} .$$

The set of all schedules with makespan at most μ is denoted by

$$\mathbf{S}(\mu) := \{y \in \mathbf{S} \mid y_k = 0 \text{ if } \text{load}(k) > \mu\} .$$

In the following, we usually interpret a schedule $y \in \mathbf{S}(\mu)$ as a vector in $\mathbb{N}_0{}^{\mathbf{K}(\mu)}$ by ignoring all zero-entries corresponding to configurations not contained in $\mathbf{K}(\mu)$.

The minimum makespan for N_L can be obtained by determining the minimum value μ with $\mathbf{S}(\mu) \neq \emptyset$. Checking whether $\mathbf{S}(\mu)$ is empty and, otherwise, finding a schedule $y \in \mathbf{S}(\mu)$ can be done by finding a feasible solution to an integer linear program. We can write

$$\mathbf{S}(\mu) = \{y \in \mathbb{N}_0{}^{\mathbf{K}(\mu)} \mid A(\mu)\, y = b\} ,$$

where $A(\mu)$ is a matrix in $\mathbb{N}_0{}^{(1+|I|)\times|\mathbf{K}(\mu)|}$ and b is a vector in $\mathbb{N}_0{}^{1+|I|}$. The first row of the linear system $A(\mu)\, y = b$ corresponds to constraint (2.10); the remaining $|I|$ rows correspond to constraints (2.11).

Lemma 3. *Let N be a set of jobs and let j be a new job of size $p_j \geq \epsilon\, \text{lb}(N)$. Any schedule for N_L with makespan $\mu \leq (1+\epsilon)\text{opt}(N)$ can be turned into a schedule for $N_L \cup \{j\}$ by*

touching only a constant number of machines such that the makespan of the resulting schedule is at most $\max\{\mu, \mathrm{opt}(N_L \cup \{j\})\}$.

Proof. We distinguish two cases. If $p_j \geq 2\mu$, then it is easy to observe that $\mathrm{opt}(N_L \cup \{j\}) = p_j$ and an optimal schedule for $N_L \cup \{j\}$ can be obtained by assigning job j to an arbitrary machine and moving all jobs that are currently on this machine to any other machine.

In the remainder of the proof we can assume that $p_j = (1+\epsilon)^{i'} < 2\mu$ and therefore $\mathrm{opt}(N_L \cup \{j\}) \leq 2\mu$. Let $y \in \mathbf{S}(\mu)$ denote the given schedule for N_L. Then y satisfies

$$A(\mu)\, y \;=\; b \;, \qquad y \in {\mathbb{N}_0}^{\mathbf{K}(\mu)} \;. \tag{2.12}$$

Let $I' := I \cup \{i'\}$ and let $\mathbf{K}'$ denote the set of configurations $k : I' \to \mathbb{N}_0$ that can occur in any schedule for $N_L \cup \{j\}$. Then $\mathbf{K}'(\mu)$, $\mathbf{S}'(\mu)$, $A'(\mu)$, and b' are defined analogously to $\mathbf{K}(\mu)$, $\mathbf{S}(\mu)$, $A(\mu)$, and b, respectively, with $\mathbf{K}$ replaced by $\mathbf{K}'$ and I replaced by I'.

Let $\mu' := \max\{\mu, \mathrm{opt}(N_L \cup \{j\})\} \leq 2\mu$. We are looking for a schedule $y' \in \mathbf{S}'(\mu')$, that is, y' must satisfy

$$A'(\mu')\, y' \;=\; b' \;, \qquad y' \in {\mathbb{N}_0}^{\mathbf{K}'(\mu')} \;. \tag{2.13}$$

Moreover, y' should be 'similar' to y. In order to compare the two vectors, we first 'lift' y to a vector in ${\mathbb{N}_0}^{\mathbf{K}'(\mu')}$ as follows. A configuration $k \in \mathbf{K}(\mu)$ can be interpreted as an element of $\mathbf{K}'(\mu')$ by defining $k(i) := 0$ for all $i \in I' \setminus I$. We then define

$$y_k \;:=\; 0 \qquad\qquad \text{for all } k \in \mathbf{K}'(\mu') \setminus \mathbf{K}(\mu) \;.$$

It follows from (2.12) that the extended vector y satisfies

$$A'(\mu')\, y \;=\; \hat{b} \;, \qquad y \in {\mathbb{N}_0}^{\mathbf{K}'(\mu')} \;. \tag{2.14}$$

The right hand side $\hat{b} \in {\mathbb{N}_0}^{1+|I'|}$ is defined as follows: If $I' = I$, then $\hat{b} = b$; otherwise, $I' = I \cup \{i'\}$ and we define the entry of vector $\hat{b}$ corresponding to i' to be zero and all other entries as in vector b.

Thus y and y' are solutions to essentially the same integer linear program ((2.14) and (2.13), respectively) with slightly different right hand sides $\hat{b}$ and b', respectively. More precisely, the right hand sides are equal for all but one entry (the one corresponding to i) where they differ by 1.

Theorem 4 ([Sch86, Corollary 17.2a]). *Let A be an integral $m \times n$-matrix, such that each sub-determinant of A is at most Δ in absolute value, let $\hat{b}$ and b' be column m-vectors, and let c be a row n-vector. Suppose $\max\{cx \mid Ax \leq \hat{b};\ x \text{ integral}\}$ and $\max\{cx \mid Ax \leq b';\ x \text{ integral}\}$ are finite. Then for each optimum solution y of the first maximum there exists*

an optimum solution y' of the second maximum such that

$$\|y - y'\|_\infty \;\leq\; n\Delta\,(\|\hat{b} - b'\|_\infty + 2).$$

By Theorem 4 (choosing c to be zero vector), there exists a solution y' to (2.13) satisfying

$$\|y - y'\|_\infty \;\leq\; 3\left|\mathbf{K}'(\mu')\right|\Delta \;\;, \tag{2.15}$$

where Δ is an upper bound on the absolute value of any sub-determinant of the matrix $A(\mu')$. To complete the proof, we have to show that the right hand side of (2.15) is constant.

First we give an upper bound on the number of columns $|\mathbf{K}'(\mu')|$, i.e., on the number of machine configurations with load at most μ'. Since each job has size at least

$$\epsilon\,\mathrm{lb}(N) \overset{(2.1)}{\geq} \frac{\epsilon}{2}\,\mathrm{opt}(N) \;\geq\; \frac{\epsilon}{2(1+\epsilon)}\,\mu \;\geq\; \frac{\epsilon}{4(1+\epsilon)}\,\mu' \;\;,$$

there are at most $\gamma := \lfloor 4(1+\epsilon)/\epsilon \rfloor \leq \frac{8}{\epsilon}$ jobs in any configuration $k \in \mathbf{K}'(\mu')$. In particular, $k(i) \leq \gamma$ for all $i \in I'$. This yields

$$\left|\mathbf{K}'(\mu')\right| \;\leq\; \gamma^{|I'|} \;\leq\; \gamma^{|I|+1} \leq \left(\frac{8}{\epsilon}\right)^{|I|+1} \overset{(2.9)}{\leq} \left(\frac{8}{\epsilon}\right)^{\frac{3}{\epsilon}\log(\frac{1+\epsilon}{\epsilon})} \leq \left(\frac{1+\epsilon}{\epsilon}\right)^{\frac{3}{\epsilon}\log(\frac{8}{\epsilon})} \tag{2.16}$$

Finally, all entries in the first row of $A'(\mu')$ are 1 and the remaining entries are of the form $k(i) \leq \gamma$. Hence we bound Δ as follows. The maximum dimension of a square sub-matrix inside $A'(\mu')$ is at most the number of rows, i.e., $2 + |I|$ and, each entry in it has value at most γ. Hence the value of its determinant is upper-bound by $((2+|I|)\gamma)^{2+|I|}$. Thus

$$\Delta \;\leq\; \left((2+|I|)\,\gamma\right)^{2+|I|} \overset{(2.9)}{\leq} \left(\frac{8}{\epsilon^2}\right)^{2+|I|} \cdot \left(\frac{8}{\epsilon}\right)^{2+|I|} \overset{(2.9)}{\leq} \left(\frac{1+\epsilon}{\epsilon}\right)^{\frac{12}{\epsilon}\log(\frac{4}{\epsilon})}$$

Hence the number of machines touched is at most

$$\left|\mathbf{K}'(\mu')\right| \cdot \|y - y'\|_\infty \overset{(2.15)}{\leq} 3\left|\mathbf{K}'(\mu')\right|^2\Delta \overset{(2.16)}{\leq} 3\left(\frac{1+\epsilon}{\epsilon}\right)^{\frac{18}{\epsilon}\log(\frac{8}{\epsilon})} \tag{2.17}$$

and therefore is a constant. This concludes the proof. □

Theorem 5. *Let N be a set of jobs and let j be a new job not contained in N. Any $(1+\epsilon)$-approximate schedule for N can be turned into a $(1+\epsilon)$-approximate schedule for $N \cup \{j\}$ such that the total size of jobs that have to be moved is bounded by a constant $\beta(\epsilon)$ times p_j.*

Proof. We distinguish two cases. If the newly arrived job is small, i.e., $p_j < \epsilon\,\mathrm{lb}(N)$, then j can simply be assigned to the least loaded machine by Observation 1 and no job in N has to be moved.

It remains to consider the case $p_j \geq \epsilon \operatorname{lb}(N)$. The given schedule for N induces a schedule for N_L with makespan $\mu \leq (1+\epsilon)\operatorname{opt}(N) \leq (1+\epsilon)\operatorname{opt}(N \cup \{j\})$. By Lemma 3, the latter schedule can be turned into a schedule for $N_L \cup \{j\}$ with makespan at most

$$\max\{\mu, \operatorname{opt}(N_L \cup \{j\})\} \;\leq\; (1+\epsilon)\operatorname{opt}(N \cup \{j\}) \;\;,$$

by touching only a constant number of machines. In the following, this subset of machines of constant size is denoted by M'. We construct a schedule for $N \cup \{j\}$ as follows:

i) Start with the schedule for $N_L \cup \{j\}$ discussed above.

ii) The jobs in N_S that were assigned, by the given schedule for N, to one of the machines in $M \setminus M'$ are assigned to the same machine again.

iii) The remaining jobs in N_S are assigned one after another to the least loaded machine.

The makespan of the partial schedule constructed in steps i) and ii) is bounded by the maximum of the makespan of the given schedule for N and the optimal makespan of the schedule for $N_L \cup \{j\}$. It is thus bounded by $(1+\epsilon)\operatorname{opt}(N \cup \{j\})$. Assigning small jobs greedily to the least loaded machine in step iii) therefore results in a $(1+\epsilon)$-approximate schedule for $N \cup \{j\}$ by Observation 1.

Finally, notice that, in the whole process, only jobs that have initially been scheduled on machines M' are moved. The total size of these jobs is at most

$$\begin{aligned}(1+\epsilon)\operatorname{opt}(N)\,|M'| \;&\overset{(2.1)}{\leq}\; 2(1+\epsilon)\operatorname{lb}(N)\,|M'| \;\leq\; 2p_j\left(\frac{1+\epsilon}{\epsilon}\right)|M'| \\ &\overset{(2.17)}{=}\; p_j\left(\frac{1+\epsilon}{\epsilon}\right)^{O\left(\frac{\log(2/\epsilon)}{\epsilon}\right)} \;.\end{aligned}$$

This concludes the proof. □

Theorem 6. *There exists a $(1+\epsilon)$-competitive online algorithm with constant migration factor $\beta(\epsilon)$ such that the running time for scheduling a newly arrived job is constant.*

In particular, it follows from the last property mentioned in the theorem that the algorithm has linear running time.

Proof. The result on the competitive ratio follows from Theorem 5. It remains to show that upon arrival of a new job j, the schedule can be updated in constant time. We consider only the non-trivial case $\epsilon < 1$. We assume that for the current set of jobs N the following information is given:

– The total size of jobs $p(N)$, the maximum job size $p_{\max}(N)$, and the lower bound $\operatorname{lb}(N)$.

– For each machine, its load rounded down to the nearest integer power of $1+\frac{\epsilon}{2}$.

We argue that this information can be updated in constant time for the new set of jobs $N \cup \{j\}$. It is certainly easy to compute $p(N \cup \{j\}) := p(N) + p_j$, $p_{\max}(N \cup \{j\}) := \max\{p_{\max}(N), p_j\}$, and $\mathrm{lb}(N \cup \{j\}) := \max\{p(N \cup \{j\})/m, p_{\max}(N \cup \{j\})\}$. Since only constant number of machines are touched to incorporate the new job j, approximating the modified machine loads can also be done in constant time.

From Lemma 3, we recall the notion of a machine configuration $k(\cdot)$ with respect to the set of large jobs. In the following, we call a job small if its size is less than $\frac{\epsilon}{2}\mathrm{lb}(N)$ and large otherwise. This slightly modified definition is just a technicality and it only affects the bound on I and γ in Lemma 3, by a constant factor. Thus it only changes the bound (2.17), by a constant factor.

Similar to the arguments in Lemma 3, we argue that the number of machine configurations with each configuration having load at most $4\mathrm{lb}(N)$ is a constant. That is $|\mathbf{K}(4\mathrm{lb}(N))|$ is a constant. Each large job (belonging to N_L) has size at least $\frac{\epsilon}{2}\mathrm{lb}(N)$. Hence a machine configuration with total load at most $4\mathrm{lb}(N)$ has at most $\frac{8}{\epsilon}$ jobs from N_L. Each such job belongs to one of the job classes from I. Since the total number of large job classes $|I|$ is also a constant and the job classes are indexed from $\lceil \log_{1+\epsilon}(\frac{\epsilon}{2}\mathrm{lb}(N)) \rceil =: i$ to $i + |I|$, we get:

Observation 4. *There are at most a constant number (say c) of configurations with each configuration having total load at most $4\mathrm{lb}(N)$, and we can enumerate them in constant time.*

The given schedule on N is represented using the following simple data structure. We assume that initially the schedule is given to us in this form. Later we show how to update it in constant time while scheduling job j. The machine configurations are represented using structures as shown in Figure 2.4. There is an array of *Config Head*s of dimension c, one for each possible configuration, and each *Config Head* points to the list of machines (list of *Machine Node*s) obeying that configuration. A *Machine Node* points to the list of jobs in it grouped as batches. The small jobs (belonging to N_S) in it are grouped into batches of size at most $\frac{\epsilon}{2}\mathrm{lb}(N)$ and the large jobs remain as a batch with single node. Clearly there are only a constant number of such batches in any machine. Each batch has a *Batch Head* that points to the list of jobs (*Job Node*s) in it.

Since $\mathrm{lb}(\cdot)$ is monotonically increasing, the machines belonging to the same configuration list still belong together in future, possibly in a different configuration list, as long as they are untouched. Thus while incorporating job j, *Config Head* array and its associated machine lists can be updated in constant time if a constant number of machines are touched and the pointers to their corresponding *Machine Node*s are available.

To find a machine with load at most $(1 + \frac{\epsilon}{2})\mathrm{lb}(N)$ in constant time, we use bucketing. This is needed for assigning small jobs. The machines are partitioned into buckets based on the exponent of $1+\frac{\epsilon}{2}$ to which machine loads are rounded. All machines with approximate load at most $\mathrm{lb}(N)$ belong to bucket 0. Each machine belonging to bucket $i > 0$ has its approximate load ℓ such that $\frac{\mathrm{lb}(N)}{1+\epsilon/2} < \frac{\ell}{(1+\epsilon/2)^i} \leq \mathrm{lb}(N)$. Since the maximum load of a machine is at most $2\mathrm{opt}(N) \leq 4\mathrm{lb}(N)$, the number of buckets is at most a constant. A machine from

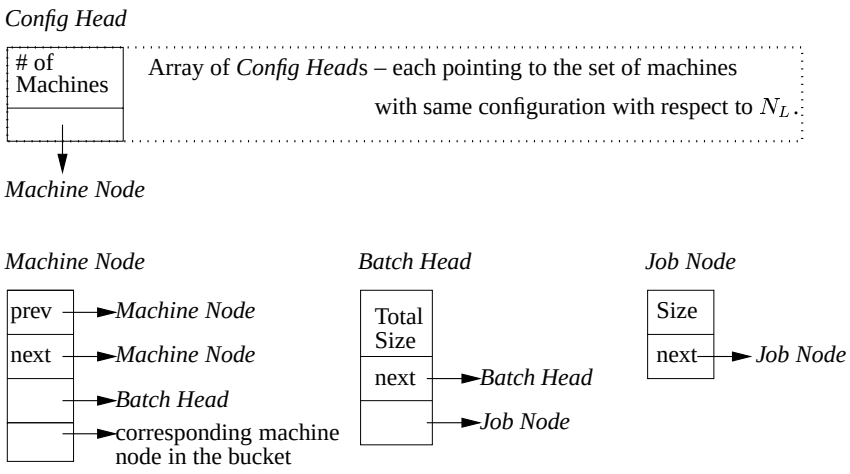


Fig. 2.4: The different structures used in representing the schedule.

bucket 0 can be found in constant time (bucket 0 is always non-empty). Observe that the untouched machines in a bucket stay together even in future (possibly in a new bucket) as $\mathrm{lb}(\cdot)$ is monotonically increasing. Thus the bucket structure can be updated in constant time while assigning job j if only a constant number of machines are touched. We assume that the bucket representation for N is also available in the beginning.

If the new job j is small ($p_j < \frac{\epsilon}{2}\mathrm{lb}(N)$), then

i) Consider any machine from bucket 0. Assigning job j to it changes its load to at most $(1+\epsilon)\mathrm{lb}(N) \leq (1+\epsilon)\mathrm{opt}(N \cup \{j\})$, as any machine in bucket 0 has load at most $(1+\frac{\epsilon}{2})\mathrm{lb}(N)$.

ii) Assign j to a batch of small jobs such that the total load in it does not exceed $\frac{\epsilon}{2}\mathrm{lb}(N)$. If no such batch exists, create a new batch for job j. Update the batches (merge small batches) with respect to $N \cup \{j\}$. There are only constant number of batches initially.

iii) Update the bucket structure with respect to $N \cup \{j\}$.

If the new job j is large then with respect to the input schedule y

a) Generate feasible schedules $z \in \mathbf{S}'\big(4\,\mathrm{lb}(N \cup \{j\})\big)$ by enumerating all vectors with constant distance $\|y - y'\|_\infty$ (see (2.15) and (2.17) of Lemma 3), where $y' \in \mathbf{S}'\big((1+\epsilon)\,\mathrm{opt}(N \cup \{j\})\big)$. Observe that $\mathbf{S}'\big((1+\epsilon)\,\mathrm{opt}(N \cup \{j\})\big) \subseteq \mathbf{S}'\big(4\,\mathrm{lb}(N \cup \{j\})\big)$. There are only constant number of such vectors and they can be generated in constant time by Observation 4. For each feasible vector do the following and choose the one minimizing the makespan.

b) The component wise difference between y and z specifies a subset of configurations and non-zero number of machines from each such configuration that should be modified. For

each such configuration, we remove the required number of machines from the front of the machine list pointed to by the respective *Config Head*.

- Remove small job batches (with respect to $N \cup \{j\}$) from these machines.
- Reschedule the remaining jobs among these machines.
- Assign these machines to the appropriate configuration lists.
- Update the bucket structure with respect the new machine loads.
- Each of the small batches are reassigned as in the small job case discussed above.

It is straightforward to verify that all of the above steps take only constant time. Thus we conclude the proof. □

2.7 The Two Machine Case

In this section we show a tight competitive ratio of $\frac{7}{6}$ for the two machine case. Consider the following procedure FILL_4.

Procedure FILL_4:
Upon arrival of j, choose the option minimizing the makespan from the following options.

> For each fixed machine $i \in \{1, 2\}$, we define multiple options in the following way. Let L be the largest 3 jobs in machine i. Set L could possibly have less than three jobs. Let the remaining jobs in machine i be B. That is $B = S(i) \setminus L$. Let $\mathcal{L} \subseteq 2^L$ be set of all possible subsets (including ϕ) of L such that for each $L_k \in \mathcal{L}$, $p(L_k) \leq p_j$. Each set $L_k \in \mathcal{L}$ gives rise to a new
>
> *Option* $(i.k)$: Migrate jobs in L_k to the other machine. Consider the jobs in B in non-increasing order order of size and repeatedly remove them; stop before the total size of removed jobs exceeds $p_j - p(L_k)$. Let B_k denote these removed jobs. Assign job j to machine i and assign the removed jobs successively to the least loaded machine.
>
> *Option* 0: Assign job j to the least loaded machine.

Theorem 7. FILL_4 *is* $\frac{7}{6}$*–competitive with factor 1 migration.*

Proof. The migration factor is clear from the FILL_4 description. To show the competitive ratio, we consider an arbitrary *input schedule* on the set of jobs N that is $\frac{7}{6}$–approximate. We show that FILL_4 yields a $\frac{7}{6}$–approximate schedule for $N \cup \{j\}$.

If job j is such that $p_j \leq \mathrm{opt}'/3$ then option 0 already yield a $\frac{7}{6}$–competitive schedule by Observation 1. It remains to handle the *case* $p_j > \mathrm{opt}'/3$. From now on we assume that Option 0 does not suffice. As the largest three jobs are not included in set B we have,

Observation 5. *For any machine i consider the set B as defined in* FILL_4*. Every job in it has size at most* $\mathrm{opt}'/3$.

Observation 6. *Migrating jobs of total size at most $p_j - \mathrm{opt}'/3$ from any fixed machine and assigning job j there yields a $\frac{7}{6}$–approximate schedule for $N \cup \{j\}$.*

Proof. Let machine 2 be the higher loaded machine and let $p(S(2)) = \mathrm{opt}' - p_j/2 + y$. This implies that $p(S(1)) \leq \mathrm{opt}' - p_j/2 - y$. Merely assigning job j to machine 1 fail only if the final load difference exceeds $\mathrm{opt}'/3$. For this it is necessary that $y < p_j/2 - \mathrm{opt}'/6$ and $p_j > \mathrm{opt}'/3$. The total migration amount needed is either $p_j/2 - \mathrm{opt}'/6$ (from machine 1 to 2) or $p_j/2 + y - \mathrm{opt}'/6$ (from 2 to 1), which is at most $p_j - \mathrm{opt}'/3$. □

Though above observation is true, it is possible that every feasible set of jobs that needs to be migrated have total size more than $p_j - \mathrm{opt}'/3$.

Fact 2. *There is a subset of jobs of total size at most p_j residing in a machine such that scheduling job j here and migrating this subset to the other machine yields a $\frac{7}{6}$–approximate schedule.*

It might take exponential time in the worst case to identify the subset that needs to be migrated. Using Observation 6 we show that our polynomial time strategy also works. Let X denote the *set with the smallest total size* that needs to be migrated from machine i according to Fact 2. Consider the sets L and B for machine i as defined in FILL_4. Let $X \cap L = L_k$. Observe that $L_k \in \mathcal{L}$. We show that Option $(i.k)$ yields a $\frac{7}{6}$–approximate schedule. The set of jobs that are either migrated or removed from machine i before assigning j is $L_k \cup B_k$. Observe that $p(L_k \cup B_k)$ is at least $p_j - \mathrm{opt}'/3$ unless $B_k = B$ as size of any job in B is at most $\mathrm{opt}'/3$ by Observation 5. Thus in any case $p(X) \leq p(L_k \cup B_k)$ as $p(X) \leq p_j - \mathrm{opt}'/3$ by Observation 6. Thus by Observation 6 the total load in machine i after assigning job j, i.e., $p(S(i) \setminus (L_k \cup B_k)) + p_j$, is at most $\frac{7}{6}\mathrm{opt}'$. The reassignment of jobs in B_k (each have size at most $\mathrm{opt}'/3$) is also fine by Observation 1. □

Proof. (Fact 2) It is clear that there is no need to migrate a total size more than p_j (simply assign p_j on the destination machine instead). Fix any optimal schedule of $N \cup \{j\}$. We capture the difference between this optimal schedule and the input schedule on N by sets $\delta_1, \delta_2, \Delta_1$ and Δ_2. As shown in Figure 2.5, the above four sets define a partition of N. The set δ_1 is the set of all jobs assigned to machine 1 on both schedules. Similarly, set Δ_2 is the set of all jobs assigned to machine 2 on both schedules. The remaining two sets capture the differences between these two schedules except for job j, which is only present in the optimal schedule.

We consider only the interesting case that assigning j to any machine without migration fails. From now, for convenience, we denote the size of any set X as simply X. For job j we denote its size as j. We also normalize opt' to 1. The optimal schedule (Figure 2.5) and the

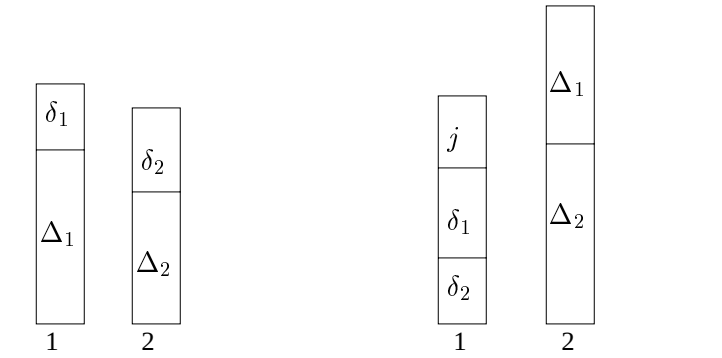


Fig. 2.5: Comparison of input schedule on N and optimal schedule on $N \cup \{j\}$, for two machines.

fact that assigning j without migration fails implies

$(1)\ \delta_1 + \delta_2 + j \leq 1$ $(3)\ \delta_1 + \Delta_1 + j > 7/6$

$(2)\ \Delta_1 + \Delta_2 \leq 1$ $(4)\ \delta_2 + \Delta_2 + j > 7/6$

It is straightforward to verify that these four inequalities yield $j > 2/3$ and hence $\delta_1 + \delta_2 \leq 1/3$ (Inequalities 1 and 3 imply $\Delta_1 > 1/6 + \delta_2$. Inequalities 2 and 4 imply $\delta_2 + j > 1/6 + \Delta_1$). W.l.o.g let $\delta_1 \leq 1/6$. Hence the migration strategy is; migrate Δ_2 to machine 1 and schedule j on machine 2. □

Tight Lower Bound

Theorem 8. *Let A be any deterministic algorithm that is c–competitive on two machines with migration factor at most one. Then $c \geq \frac{7}{6(1+\epsilon)}$ for any sufficiently small positive $\epsilon \in \mathbb{R}^+$.*

Proof. The adversary initially issues four jobs with the following size: $1/6 + \epsilon/2, 1/6 + \epsilon/2, 1/2$ and $1/2$. It is easy to verify that the only $\frac{7}{6(1+\epsilon)}$–approximate way of scheduling them is to assign in each machine one job of size $1/6 + \epsilon/2$ and one job of size $1/2$.

Assume that both machines contain one job of size $1/2$ and one job of size $1/6+\epsilon/2$. Now adversary issues a new job of size $2/3$. The optimal makespan is $1 + \epsilon$. But if the migration factor is restricted to 1, then the best possible makespan by A is $7/6$. Hence $c = \frac{7}{6(1+\epsilon)}$. □

2.8 Maximizing the Minimum Machine Load

An alternative, yet less frequently used objective for machine scheduling is to maximize the minimum load. However, we have a concrete application using this objective function that was the original motivation for our interest in bounded migration: Storage area networks (SAN)

usually connect many disks of different capacity and grow over time. A convenient way to hide the complexity of a SAN is to treat it as a single big, fault tolerant disk of huge capacity and throughput [BSS02, San04]. A simple scheme with many nice properties implements this idea if we manage to partition the SAN into several sub-servers [San04] of about equal size. Mapping to our scheduling framework, the contents of disks correspond to jobs and the sub-servers correspond to machines. Each sub-server stores the same amount of data. For example, if we have two sub-servers, each of them stores all the data to achieve a fault tolerance comparable to mirroring in ordinary RAID level 0 arrays [PGK88]. More sub-servers allow for a more flexible tradeoff between fault tolerance, redundancy, and access granularity. In any case, the capacity of the server is determined by the *minimum* capacity of a sub-server. Moreover, it is not acceptable to completely reconfigure the system when a new disk is added to the system or when a disk fails. Rather, the user expects a "proportionate response", i.e., if she adds a disk of x GByte she will not be astonished if the system moves data of this order of magnitude but she would complain if much more is moved. Our theoretical investigation confirms that this 'common sense' expectation is indeed reasonable.

We concentrate on the case without job departures (disk failures). We show that the following simple strategy, which is very similar to FILL_1, is 2–competitive already for migration factor $\beta = 1$.

Procedure FILL_5:
Upon arrival of a new job j, do the following. Repeatedly remove jobs from the least loaded machine $i_{\min}$; stop before the total size of removed jobs exceeds p_j. Assign job j to machine $i_{\min}$. Assign the removed jobs successively to the least loaded machine.

Theorem 9. FILL_5 *is* 2*–competitive with migration factor* 1.

Proof. The migration factor is clear from the description of FILL_5. In the input schedule on N, consider the *maximum loaded machine among those containing multiple jobs*. If there is no such machine (i.e, every machine has at most one job) then the schedule after assigning job j is 1–approximate (optimal). Hence the interesting case is that such machines exist. We call such machines as *multi-job* machines.

We assume that the following property holds for the input schedule;

$$\text{maximum load of a multi-job machine} \;\leq\; 2 \cdot \text{ minimum load} \tag{2.18}$$

Later we show that the above property is preserved on the output schedule. An output schedule with above property is a 2–approximate schedule as the optimal *minimum* load is at most the maximum load of a multi-job machine. Thus it remains to show that property (2.18) holds for the output schedule. If $p_j \leq p(S(i_{\min}))$ then this is straightforward. In case $p_j > p(S(i_{\min}))$, initially all jobs from the least loaded machine are removed and j is assigned there. This intermediate schedule (before reassigning the removed jobs) satisfies property (2.18). Observe that each of the removed jobs has size at most the intermediate minimum

load. Hence reassigning them still preserves the property as shown above. □

Negative Result

The following lemma shows that it is not possible to start with an arbitrary 2–approximate schedule on N and obtain a 2–approximate schedule for $N \cup \{j\}$ with constant migration factor.

Lemma 4. *There is a* 2*–approximate schedule on* m *machines such that upon the arrival of a new job, it is not possible for any strategy to obtain* 2*–approximate schedule with migration factor less than* $m - 2$.

Proof. In the initial schedule machine 1 has $2m$ jobs of size 1. All the remaining $m - 1$ machines have one job of size 1. In total there are $3m - 1$ jobs. The optimal minimum load is 2. Hence this is a 2–approximate schedule. A new job of size 1 arrives. The new optimal minimum is 3. To achieve minimum load greater than 1, any strategy has to move at least $m - 2$ jobs from machine 1. □

2.9 Summary and Open Problems

We presented strategies for online scheduling problem where we allow migration of already assigned jobs. We show that already with migration factor 2 the competitive ratio becomes 1.5, which beats the lower bounds for the classical online scheduling problem. We bring down the migration factor to $4/3$ for 1.5–competitiveness and also obtain 4/3–competitiveness with factor 4 migration. For the case of two machines we obtain a tight competitive ratio of $7/6$ with migration factor 1. We presented an online PTAS (yielding competitive ratio of $1 + \epsilon$ for any $\epsilon > 0$ with constant migration factor $f(\epsilon)$) with linear running time, which compares with the PTAS for the offline version. We also discussed the setting of maximizing the minimum load and gave a 2–competitive algorithm with unit migration factor.

One open problem is to bring down the migration factors while achieving same or better competitive ratios. It is an interesting question to show better lower bounds on the migration factors needed, possibly as a function of the desired competitive ratio. Another open problem is to come up with better strategies for the maximizing minimum load objective. It would also be interesting to investigate the cases of related parallel machines, i.e., the job size gets scaled differently in different machines, and unrelated parallel machines, i.e., processing time for job j on machine i is given separately as p_{ij}.

Chapter 3

Smoothed Competitiveness of Metrical Task Systems

3.1 Introduction

Metrical Task Systems, introduced by Borodin, Linial and Saks [BLS92], is one of the very well-known and well-studied problems in the field of online algorithms. Metrical task systems is formulated as follows. We are given an undirected and connected graph $G := (V, E)$, with node set V 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 in node u after serving task τ_{t-1}, 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 on any given metric space on n nodes *every* deterministic online algorithm for metrical task systems has competitive ratio at least $2n - 1$. On the other hand, the competitive ratio often is an over-pessimistic estimation of the performance of an online algorithm, since instances that force the online algorithm in its worst-case behavior are

Publication Notes. This is a joint work with Guido Schäfer. A preliminary version of this work appeared in the proceedings of the 21st International Symposium on Theoretical Aspects of Computer Science (STACS), 2004 [SS04].

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

highly artificial.

Smoothed Competitive Analysis. Spielman and Teng [ST01] proposed a new complexity measure, called *smoothed complexity*, which is a hybrid between average-case and worst-case complexity. The basic idea of *smoothed analysis* is to *randomly perturb*, or *smoothen*, the input instances and to analyze the performance of the algorithm on the perturbed instances. The smoothed complexity of an algorithm is defined as the supremum over all input instances of the expected running time on the perturbed instances.

More formally, let $\mathcal{I}_n \in \mathbb{R}^n$ denote the set of all inputs of length n for a deterministic algorithm A. Let $\check{I} = (i_1, \ldots, i_n) \in \mathcal{I}_n$ be one such input. Let $A(\check{I})$ denote the performance of A on input $\check{I}$. Consider the following random variable $I = (i_1 + \varepsilon_1, i_2 + \varepsilon_2, \ldots, i_n + \varepsilon_n)$, where each ε_i is a random number chosen from a *symmetric distribution* f with zero mean. The process of drawing an instance from the *neighborhood* of $\check{I}$ in the above fashion is called the perturbation or smoothing process. The extend of perturbation ('size' of the neighborhood) depends on the *smoothing parameter* σ, which relates to the distribution f in the sense that it reflects how much the smoothed input deviate from the initial input values stochastically. The worst-case complexity and the smoothed complexity of A can be written as,

$$\text{Worst-case} = \sup_{\check{I} \in \mathcal{I}_n} A(\check{I}) \qquad \text{Smoothed Complexity} = \sup_{\check{I} \in \mathcal{I}_n} \mathbf{E}_{I \xleftarrow{f} \check{I}} \left[A(\check{I}) \right].$$

Intuitively, the smoothed performance of an algorithm on a given instance is its performance 'averaged' over the instance 'neighborhood' (because of the random perturbation). Hence the smoothed complexity of an algorithm is small if the worst case instances are isolated peaks in the instance/performance space, which in a sense reflects the 'sparse distribution' of worst-case instances.

Based on this idea underlying *smoothed analysis*, Becchetti et al. [BLMS+03] recently proposed *smoothed competitive analysis* as an alternative to worst-case competitive analysis of online algorithms. The idea is to randomly perturb, or *smoothen*, an adversarial input instance $\check{\mathcal{S}}$ and to analyze the performance of the algorithm on the perturbed instances. Let $\mathrm{alg}[\mathcal{S}]$ and $\mathrm{opt}[\mathcal{S}]$, respectively, be the cost of the online and the optimal offline algorithm on a smoothed instance $\mathcal{S}$ obtained from $\check{\mathcal{S}}$. The *smoothed competitive ratio* c of alg with respect to a smoothing distribution f is defined as

$$c := \sup_{\check{\mathcal{S}}} \mathbf{E}_{\mathcal{S} \xleftarrow{f} \check{\mathcal{S}}} \left[\frac{\mathrm{alg}[\mathcal{S}]}{\mathrm{opt}[\mathcal{S}]} \right].$$

We use the notion of smoothed competitiveness to characterize the asymptotic performance of WFA. We ignore the constants in the competitive ratio and also ignore short request sequences. Each cost entry in the request vector is smoothed by adding a random number chosen from a probability distribution f, whose expectation coincides with the original cost

entry. The underlying graph is unchanged. Our analysis reveals that the smoothed competitive ratio of WFA is much better than its worst-case competitive ratio and it depends on certain *topological parameters* of the underlying graph. Our analysis holds for various probability distributions, including the uniform and the normal distribution. We also show that our bounds on smoothed competitiveness are tight for a large class of graphs, against *any deterministic* algorithm.

Relevant Topological Parameters: 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 $\mathrm{Diam}(G)$ 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 $\mathrm{diam}(G)$ if any two nodes are connected by a path of at most $\mathrm{diam}(G)$ edges. We use shorter notation $\mathrm{diam} := \mathrm{diam}(G)$ and $\mathrm{Diam} := \mathrm{Diam}(G)$ when the graph G is clear from the context. Observe that $\mathrm{diam}U_{\min} \leq \mathrm{Diam} \leq \mathrm{diam}U_{\max}$. We emphasize that these topological parameters are defined with respect to G and its length function λ and not with respect to the resulting metric, which is a complete graph.

3.1.1 The 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 ensure that the smoothed costs are non-negative. Thus, the probability for an original zero cost entry to remain zero is amplified to $\frac{1}{2}$.

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. We remark that the general upper bounds on the smoothed competitive ratio of WFA do not improve by choosing σ much larger than $U_{\min}$. There are input instances where lower bounds matching the stated upper bounds can be shown even if $\sigma = 2^{\Theta(\sqrt{n})}U_{\max}$ (See Remark 3 in Section 3.7.2). Thus, throughout this chapter, we restrict σ to the interesting range of $(0, cU_{\min}]$, for some constant c, for both upper and lower bounds. In particular, for all our analysis we assume that $\sigma \leq 2U_{\min}$, though the same analysis holds for any fixed constant. Moreover, we use c_f to denote a constant

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

Tab. 3.1: Upper bounds on the smoothed competitive ratio of WFA.

depending on f such that for a random ε chosen from f, $\mathbf{Pr}[\varepsilon \geq \sigma/c_f] \geq \frac{1}{4}$.

Random tasks are generated by smoothing zero tasks (all-zero entries). Here we take care that we only allow distributions over $[0,\infty)$.

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

In the terminology of [ST01], our smoothing model falls into the category of *additive symmetric smoothing models*, where a random number chosen from a symmetric distribution is added to the original input. We refer to [ST01] for a review of other smoothing models.

3.2 Our Contribution

We prove several upper bounds; see Table 3.1.

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 a competitive ratio of $O(\log(n))$ on a clique and of $O(1)$ on constant degree graphs like a line graph, 2-d grid, binary tree etc.

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

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

   For example, if $\sigma=\Theta(U_{\min})$ and $U_{\max}/U_{\min}=\Theta(1)$, WFA has smoothed competitive ratio $O(\log(n))$ on any graph with constant edge diameter and $O(\sqrt{n})$ on any graph

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

Tab. 3.2: Lower bounds on the smoothed competitive ratio of any deterministic online algorithm.

with constant maximum degree. Note that we obtain an $O(\log(n))$ bound on a complete binary tree and $O(\log(n)\,\mathrm{loglog}(n))$ on an n-node hypercube.

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. A 1-elementary task is also called an *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.

Remark 1. The upper bound expressions for arbitrary tasks given above hold even for the cases of β-elementary tasks and random tasks. For example, the final upper bound for β-elementary tasks is the minimum of the arbitrary tasks upper bound, $2n-1$ and the β-elementary upper bound.

We also present lower bounds; see Table 3.2. All our lower bounds hold for *any* deterministic online algorithm and if the request costs are smoothed according to the additive symmetric smoothing model as described in Section 3.1.1. 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 the parameters 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 our 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:

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

   Assume that $U_{\max}/U_{\min} = \Theta(1)$. Then, the first bound matches the first upper bound for arbitrary tasks if the edge diameter diam is constant, e.g., for a clique. The second bound matches the second upper bound for arbitrary tasks if $\mathrm{diam} = \Omega(n)$ 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 upper bound for β-elementary tasks is tight up to a factor of $(U_{\max}/U_{\min})\log(D)$.

Constrained Balls into Bins 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 balls into bins experiment with dependencies between the heights of the bins, which are specified by a constraint graph. We call it the *constrained balls into bins experiment*. We believe that the constrained balls into bins is also interesting independently of the context of this work.

Lower Bound for Zero-Retaining Smoothing Models

A major criticism to the additive smoothing model is that zero cost entries are destroyed. However, 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. The proof 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 (see [BLS92, MMS88, BEY98]). 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 and hence the lower bound still holds.

Theorem 10. *Every deterministic online algorithm for metrical task systems has a smoothed competitive ratio of at least* $2n-1$ *under a zero-retaining smoothing model.*

3.3 Related Work

Several other attempts were made in the past to overcome the over-pessimistic estimation of the performance of an online algorithm by its competitive ratio. One idea was to enhance the capability of the online algorithm by allowing a limited lookahead [Alb93, Alb98]. Another idea was to restrict the power of the adversary. For example, Borodin et al. [BIRS95] used an access graph model to restrict the input sequences in online paging problems to specific patterns. The diffuse adversary model by Koutsoupias and Papadimitriou [KP94] is another attempt to refine the notion of competitiveness. In this model, the actual distribution of the input is chosen by an adversary from a known class of possible distributions. We strongly believe that smoothed competitive analysis is a natural alternative to adequately characterize the performance guarantee of an online algorithm.

Chapter Organization. In Section 3.4 we first define the commonly used notations and also the work functions. In Section 3.4.2, we review the work function algorithm and state some of its properties. The lower bound on the cost of an optimal offline algorithm and the related balls into bins game are presented in Section 3.5. Then, in Section 3.6.1 and Section 3.6.2, we prove the upper bounds on the smoothed competitive ratio of WFA. After that, in Section 3.6.4 we present an upper bound on the competitive ratio of WFA against random tasks, and in Section 3.6.5 we develop the bound for β-elementary tasks. In Section 3.7.1 we prove the existential lower bound for β-elementary tasks. Then we proceed to show the two universal lower bounds in Section 3.7.2. This is followed by the existential lower bounds in Section 3.7.3. We conclude this chapter with concluding remarks and open problems in Section 3.8. Proofs of some of the simple facts and lemmas are moved to Appendices 3.A, 3.B and 3.C, for better readability.

3.4 Preliminaries

Let the sequence of requests for ℓ rounds be denoted as $\check{\mathcal{S}} = \langle \tau_1, \ldots, \tau_\ell \rangle$. Let $\mathrm{wfa}[\check{\mathcal{S}}]$ and $\mathrm{opt}[\check{\mathcal{S}}]$ denote costs incurred by WFA and the optimal offline strategy respectively on $\check{\mathcal{S}}$. The smoothed outcome of the request sequence is denoted by the random variable $\mathcal{S}$. Let s_0 denote the initial position. We denote by $s_0, \ldots, s_\ell$ the sequence of nodes visited by WFA. For the lower bound proofs we use $\mathrm{alg}[\mathcal{S}]$ to denote the cost incurred on $\mathcal{S}$ by the algorithm under consideration.

3.4.1 Work Function

Let $\check{\mathcal{S}}_t$ denote the subsequence of the first t tasks of $\check{\mathcal{S}}$. 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 $\check{\mathcal{S}}_t$ starting in s_0 and ending in u. The function w_t is called the *work function* at time t with respect to $\check{\mathcal{S}}$ and s_0.

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

$$w_0(u) := \delta(s_0, u), \quad \text{and} \quad w_t(u) := \min_{v\in V}\{w_{t-1}(v) + r_t(v) + \delta(v, u)\}. \tag{3.1}$$

3.4.2 Work Function Algorithm

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 $s_0, \ldots, s_t$ denote the sequence of nodes visited by wfa to process $\check{\mathcal{S}}_t$. Then, to process task τ_{t+1}, wfa moves to a node s_{t+1} that minimizes $w_{t+1}(v) + \delta(s_t, v)$ for all $v \in V$. There is always a choice for s_{t+1} such that in addition $w_{t+1}(s_{t+1}) = w_t(s_{t+1}) + r_{t+1}(s_{t+1})$. More formally,

$$s_{t+1} := \arg\min_{v\in V}\{w_{t+1}(v) + \delta(s_t, v)\} \quad \text{such that} \quad w_{t+1}(s_{t+1}) = w_t(s_{t+1}) + r_{t+1}(s_{t+1}). \tag{3.2}$$

Lemma 5 ([BEY98]). *Let A be the set of all the states that satisfy both $s_{t+1} = \arg\min_{x\in V}\{w_{t+1}(x) + \delta(s_t, x)\}$ and $w_{t+1}(s_{t+1}) = w_t(s_{t+1}) + r_{t+1}(s_{t+1})$. Then A is not empty (i.e.,* wfa *can always choose an appropriate state s_{t+1}).*

We duplicate the proof of the above lemma in the Appendix 3.C for the sake of completeness. We continue by observing a few properties of work functions and of the online algorithm wfa (see Appendix 3.A for the corresponding proofs).

Fact 3. *For any node u and any time t, $w_t(u) \geq w_{t-1}(u)$.*

Fact 4. *For any node u and any time t, $w_t(u) \leq w_{t-1}(u) + r_t(u)$.*

Fact 5. *For any two nodes u and v and any time t, $|w_t(u) - w_t(v)| \leq \delta(u, v)$.*

Fact 6. *At any time t, $w_t(s_t) = w_t(s_{t-1}) - \delta(s_{t-1}, s_t)$.*

Fact 7. *At any time t, $r_t(s_t) + \delta(s_{t-1}, s_t) = w_t(s_{t-1}) - w_{t-1}(s_t)$.*

3.4.3 Tail Inequalities

We used the following well-known tail bounds quite often in the analysis. We refer to [MR95, HR90] for their proofs.

Lemma 6 (Markov Inequality). *Let Y be a random variable assuming only non-negative values, Then for all $t \in \mathbb{R}^+$, $\mathbf{Pr}[Y \geq t] \leq \frac{\mathbf{E}[Y]}{t}$.*

Lemma 7 (Chebyshev Inequality). *Let X be a random variable with expectation $\mathbf{E}[X]$ and standard deviation σ. Then for any $t \in \mathbb{R}^+$, $\mathbf{Pr}[|X - \mathbf{E}[X]| \geq t\,\sigma] \leq \frac{1}{t^2}$.*

Lemma 8 (Chernoff Bound). *Let $X_1, \ldots, X_k$ be independent binary random variables. Let $X = \sum_{j=1}^{k} X_j$. Then for any $\delta > 0$,*

$$\mathbf{Pr}[X \geq (1+\delta)\mathbf{E}[X]] \leq e^{-\min\{\delta^2, \delta\}\cdot\mu/3} .$$

Furthermore, it holds that for all $0 < \delta < 1$,

$$\mathbf{Pr}[X \leq (1-\delta)\mathbf{E}[X]] \leq e^{-\delta^2\mu/2} .$$

3.5 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 experiment*.

3.5.1 Constrained Balls into Bins

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 node 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.

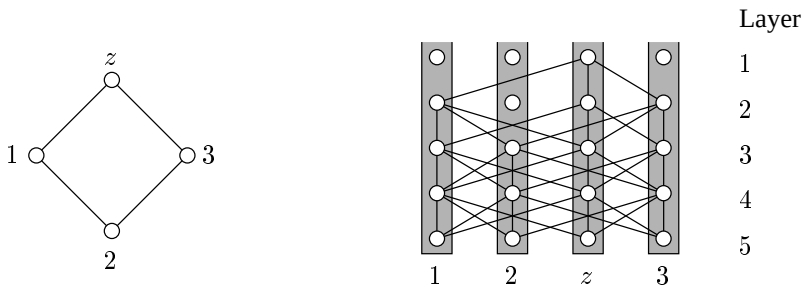


Fig. 3.1: Illustration of the "unfolding" for $Q = 1$ and $h = 5$. Left: constraint graph G_c. Right: layered dependency graph $\mathcal{D}_h$.

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 11. *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,* $\mathbf{Pr}[R_z > c_3 h\,(1 + \log(D)/Q)] \leq 1/n^4$.

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, but essentially equivalent, experiment 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 3.1 for an example.

Now, the following experiment on $\mathcal{D}_h$ is equivalent to the balls and bins experiment. 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 and only 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 11 . Let $\mathcal{D}_h$ be a layered dependency graph constructed from G_c as described above. As argued above, R_z is stochastically dominated by the random variable denoting the number of rounds needed till root node becomes FINISHED. That is, the probability that $(R_z \leq t)$ is at least the probability that root node becomes FINISHED within t rounds. 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., $\alpha(t) := \mathbf{Pr}[X \leq h-1]$. 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, $\mathbf{Pr}[R_z > t] \leq \alpha(t)(D+1)^{h'}$. 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 bound (Lemma 8) on X, we obtain for $h \geq \log(n)$

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

□

3.5.2 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 9. *Let the random variable $\mathcal{S}$ denote the smoothed outcome of 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, $\mathbf{Pr}[\mathrm{opt}[\mathcal{S}] < n\gamma U_{\min}] \leq 1/n^3$.*

We will use Lemma 9 several times as follows.

Corollary 1. *Let the random variable $\mathcal{S}$ denote the smoothed outcome of 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 $\gamma \geq 1$. Let $\mathcal{E}$ denote the event that $\mathrm{opt}[\mathcal{S}] \geq n\gamma U_{\min}$. Then*

$$\mathbf{E}\left[\frac{\mathrm{wfa}[\mathcal{S}]}{\mathrm{opt}[\mathcal{S}]}\right] \leq \mathbf{E}\left[\frac{\mathrm{wfa}[\mathcal{S}]}{\mathrm{opt}[\mathcal{S}]}\,\middle|\,\mathcal{E}\right] + o(1) \leq \frac{\mathbf{E}[\mathrm{wfa}[\mathcal{S}]]}{n\gamma U_{\min}} + o(1).$$

Proof. From Lemma 9 we know that $\mathbf{Pr}[\neg\mathcal{E}] \leq 1/n^3$. Thus,

$$\begin{aligned}\mathbf{E}\left[\frac{\mathrm{wfa}[\mathcal{S}]}{\mathrm{opt}[\mathcal{S}]}\right] &= \mathbf{E}\left[\frac{\mathrm{wfa}[\mathcal{S}]}{\mathrm{opt}[\mathcal{S}]}\,\middle|\,\mathcal{E}\right]\mathbf{Pr}[\mathcal{E}] + \mathbf{E}\left[\frac{\mathrm{wfa}[\mathcal{S}]}{\mathrm{opt}[\mathcal{S}]}\,\middle|\,\neg\mathcal{E}\right]\mathbf{Pr}[\neg\mathcal{E}] \\ &\leq \frac{\mathbf{E}[\mathrm{wfa}[\mathcal{S}] \mid \mathcal{E}]\,\mathbf{Pr}[\mathcal{E}]}{n\gamma U_{\min}} + \frac{2n-1}{n^3} \leq \frac{\mathbf{E}[\mathrm{wfa}[\mathcal{S}]]}{n\gamma U_{\min}} + o(1),\end{aligned}$$

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 9 . The cost of opt on a smoothed sequence $\mathcal{S}$ of length ℓ is $\mathrm{opt}[\mathcal{S}] = \min_{u\in V}\{w_\ell(u)\}$. 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 $Q := \lfloor U_{\min}/\xi\rfloor$, where $\xi := \min\{U_{\min}, \sigma/c_f\}$. 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 $\mathbf{Pr}[\varepsilon \geq \sigma/c_f] \geq \frac{1}{4}$, the smoothed request cost $r_t(v_i)$ is at least σ/c_f with probability at least $\frac{1}{4}$, for any v_i and any t. This holds irrespective 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 10 given below, the number of rounds needed until a bin B_i has height h stochastically dominates the time t needed until $w_t(v_i) \geq h\xi$. Applying Theorem 11, we obtain that for any bin B_i, after $\ell \geq c_3 h(1 + \log(D)/Q)$ rounds, $\mathbf{Pr}[h_\ell(i) < h] \leq 1/n^4$. Consequently, after ℓ rounds all bins have height at least h with probability at least $1 - 1/n^3$. Choosing $h := 2n\gamma Q$, this implies that after ℓ rounds, with probability at least $1 - 1/n^3$, $w_\ell(v_i) \geq 2n\gamma Q\xi \geq n\gamma U_{\min}$ for all v_i of G. Finally, we make sure that $\ell := c_2 n\gamma(U/\sigma + \log(D)) \geq c_3 h(1 + \log(D)/Q)$ by fixing $c_2 := 4c_3\lceil c_f\rceil$. □

Lemma 10. *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 $t \geq 0$, $w_t(v_i) \geq h_t(i)\,\xi$.*

Proof. We prove 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 and correspondingly all bins are empty initially. 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 $w_t(v_i) \geq h_t(i)\xi$. 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, $r_{t+1}(v_i) \geq \xi$. Thus, we have $w_{t+1}(v_i) \geq w_t(v_i) + \xi \geq (h_t(i)+1)\xi = h_{t+1}(i)\xi$, 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 ξ and rearranging terms, we obtain

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

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

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

□

Remark 2. We note that the above lower bound on opt holds even against an adaptive adversary. This is because the bound holds irrespective of the costs on the request vectors before smoothing. Moreover, the bound holds even if the distribution is restricted to $[0, \infty)$, which is the case while analyzing competitiveness if the inputs are random tasks.

3.6 Upper Bounds

3.6.1 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 11. *Let $\mathcal{K}$ be any request sequence of length ℓ. Then,* $\mathrm{wfa}[\mathcal{K}] \leq \mathrm{opt}[\mathcal{K}] + Diam \cdot \ell$.

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 7, we obtain $C(t) = w_t(s_{t-1}) - w_{t-1}(s_t)$. Hence,

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

where the last two inequalities follow from Fact 5. Since $\mathrm{opt}[\mathcal{K}] \geq \min_{v\in V} w_\ell(v)$, the lemma follows. □

Theorem 12. *The smoothed competitive ratio of* wfa *is*

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

Proof. Let the random variable $\mathcal{S}$ denote the smoothed sequence of length $\ell := \lceil c_2 n\gamma\,(U_{\min}/\sigma + \log(D))\rceil$ for some $\gamma \geq 1$. By Lemma 11, we have for any sequence $\mathcal{K}$ of ℓ tasks, $\mathrm{wfa}[\mathcal{K}] \leq \mathrm{opt}[\mathcal{K}] + \mathrm{Diam}\cdot\ell$. Thus by Corrollary 1, the upper bound follows from the following bound

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

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

3.6.2 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 8 can be found in Appendix 3.A.

Fact 8. *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 processing 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 6 we have $w_t(s_{t-1}) = w_t(s_t) + \delta(s_{t-1}, s_t)$. Therefore, the above definition is equivalent to

$$T := \left\{t : w_t(s_t) - w_{t-1}(s_{t-1}) \geq -\frac{1}{2}\delta(s_{t-1}, s_t)\right\}. \tag{3.3}$$

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

Proof. We have $w_\ell(s_\ell) - w_0(s_0) \geq -\mathrm{Diam}$, due to Fact 5 and since $w_0(s_0) \leq w_\ell(s_0)$. Thus,

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

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

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

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

$$\sum_{t\in \bar{T}}(w_{t-1}(s_{t-1}) - w_t(s_t)) \leq \mathrm{Diam} + \sum_{t\in R^+}(w_t(s_t) - w_{t-1}(s_{t-1})). \tag{3.4}$$

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 and $r_t(s_t) = w_t(s_t) - w_{t-1}(s_t)$ by

(3.2). 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 (3.3) 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 because $w_t(s_t) = w_{t-1}(s_t) + r_t(s_t)$ by (3.2) and $w_{t-1}(s_t) - w_{t-1}(s_{t-1}) \leq \delta(s_{t-1}, s_t)$ by Fact 5. 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, (3.4) implies

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

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

We further partition T into T^1 and T^2, where

$$T^1 := \{t \in T : w_t(s_t) - w_{t-1}(s_t) \leq 4U_{\max}\mathrm{diam}\}\,, \quad \text{and} \quad T^2 = T \setminus T^1.$$

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

Proof. Let $H \leq 4U_{\max}\mathrm{diam}$. 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 $d \leq \lfloor \mathrm{diam}/2 \rfloor$, 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 $\lceil \mathrm{diam}/2 \rceil$.) By Fact 5, we have for each i, $0 \leq i \leq d$,

$$w_t(u_i) \;\geq\; w_t(u_0) - iU_{\max} \quad \text{and} \quad w_{t-1}(u_i) \;\leq\; w_{t-1}(u_0) + iU_{\max}.$$

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

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

□

Lemma 14. *Let $\mathcal{K}$ be a sufficiently long task sequence such that $\mathrm{opt}[\mathcal{K}] \geq 2\mathrm{Diam}$. There exists a constant b such that*

$$\mathrm{opt}[\mathcal{K}] \geq \frac{1}{bn}\left(\frac{1}{U_{\max}}\sum_{t\in T^1} C(t)^2 + n\sum_{t\in T^2} C(t)\right).$$

Proof. For every node v_i, $w_\ell(v_i) \leq \min_{u\in V}\{w_\ell(u)\} + \mathrm{Diam}$ (by Fact 5). Moreover, $\mathrm{opt}[\mathcal{K}] \geq \min_{u\in V}\{w_\ell(u)\}$. We obtain

$$\sum_{i=1}^{n} w_\ell(v_i) \leq n\cdot\mathrm{opt}[\mathcal{K}] + n\mathrm{Diam}, \quad \text{or, equivalently,} \quad \mathrm{opt}[\mathcal{K}] \geq \frac{1}{n}\left(\sum_{i=1}^{n} w_\ell(v_i) - n\mathrm{Diam}\right).$$

Since $\mathrm{opt}[\mathcal{K}] \geq 2\mathrm{Diam}$, the latter reduces to

$$\mathrm{opt}[\mathcal{K}] \geq \frac{2}{3n}\sum_{i=1}^{n} w_\ell(v_i). \tag{3.5}$$

Claim 1. *For any $t \in T^1$, $\Delta W_t \geq C(t)^2/(160U_{\max})$.*

Proof. By (3.2) we have $r_t(s_t) = w_t(s_t) - w_{t-1}(s_t)$. Below, we will show that

$$\Delta W_t \geq \left(\delta(s_{t-1}, s_t)^2 + r_t(s_t)^2\right)/(80U_{\max}). \tag{3.6}$$

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 $\Delta W_t \geq C(t)^2/(160U_{\max})$. 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 13 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 $r_t(s_t) \leq 4U_{\max}\mathrm{diam}$ by the definition of T_1, using Lemma 13 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 2. *For any* $t \in T^2$, $\Delta W_t \geq 4nC(t)/10$.

Proof. Since $t \in T^2$, definition of T^2 together with (3.2) imply that $r_t(s_t)/4 > \mathrm{diam} U_{\max} \geq \delta(s_{t-1}, s_t)$. Thus, $C(t) = r_t(s_t) + \delta(s_{t-1}, s_t) < 5r_t(s_t)/4$. Furthermore, by (3.2) we have $r_t(s_t) = w_t(s_t) - w_{t-1}(s_t)$. Applying Lemma 13 with $H := r_t(s_t)$, we obtain $\Delta W_t \geq nr_t(s_t)/2 \geq 4nC(t)/10$. □

Claim 1 and Claim 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 (3.5). □

Theorem 13. *The smoothed competitive ratio of* wfa *is*

$$\mathcal{O}\left(\sqrt{n \cdot \frac{U_{\max}}{U_{\min}} \left(\frac{U_{\min}}{\sigma} + \log(D)\right)}\right).$$

Proof. Let the random variable $\mathcal{S}$ denote the smoothed sequence of length $\ell := \lceil c_2 n\gamma(U_{\min}/\sigma + \log(D))\rceil$, for an appropriate γ. Due to Corrollary 1 it suffices to bound $\mathbf{E}[\mathrm{wfa}[\mathcal{S}]/\mathrm{opt}[\mathcal{S}] \,|\, \mathcal{E}]$, where $\mathcal{E}$ is the event $(\mathrm{opt}[\mathcal{S}] \geq n\gamma U_{\min})$. Consider any smoothing outcome $\tilde{\mathcal{S}}$ such that the event $\mathcal{E}$ holds. We fix γ sufficiently large such that $\mathrm{opt}[\tilde{\mathcal{S}}] \geq 6\mathrm{Diam}$. Observe that $\mathrm{wfa}[\tilde{\mathcal{S}}] \geq \mathrm{opt}[\tilde{\mathcal{S}}] \geq 6\mathrm{Diam}$.

First, assume $\sum_{t \in T^1} C(t) < \sum_{t \in T^2} C(t)$. Then, due to Lemma 12 and Lemma 14,

$$\mathrm{wfa}[\tilde{\mathcal{S}}] \leq 16 \sum_{t \in T^2} C(t) \quad \text{and} \quad \mathrm{opt}[\tilde{\mathcal{S}}] \geq \frac{1}{b} \sum_{t \in T^2} C(t).$$

Hence, $\mathbf{E}[\mathrm{wfa}[\mathcal{S}]/\mathrm{opt}[\mathcal{S}] \,|\, \mathcal{E}] = \mathcal{O}(1)$.

Next, assume $\sum_{t \in T^1} C(t) \geq \sum_{t \in T^2} C(t)$. By Lemma 12 and Lemma 14 we have

$$\mathrm{wfa}[\tilde{\mathcal{S}}] \leq 16 \sum_{t \in T^1} C(t) \quad \text{and} \quad \mathrm{opt}[\tilde{\mathcal{S}}] \geq \frac{1}{bn}\left(\frac{1}{U_{\max}} \sum_{t \in T^1} C(t)^2\right). \tag{3.7}$$

Thus,

$$\frac{\mathrm{wfa}[\tilde{\mathcal{S}}]}{\mathrm{opt}[\tilde{\mathcal{S}}]} \leq 16bnU_{\max}\left(\frac{\sum_{t \in T^1} C(t)}{\sum_{t \in T^1} C(t)^2}\right). \tag{3.8}$$

Since $\mathcal{E}$ holds, we also have

$$\frac{\mathrm{wfa}[\tilde{\mathcal{S}}]}{\mathrm{opt}[\tilde{\mathcal{S}}]} \leq \frac{\ell \cdot 16 \sum_{t\in T^1} C(t)}{\ell \cdot n\gamma U_{\min}} \leq \frac{c}{U_{\min}} \left(\frac{U_{\min}}{\sigma} + \log(D)\right) \left(\frac{\sum_{t\in T^1} C(t)}{|T^1|}\right), \tag{3.9}$$

where the latter inequality holds for an appropriate constant c since $\ell \geq |T^1|$. Observe that (3.9) is well-defined since $\sum_{t\in T^1} C(t) \geq \frac{1}{16}\mathrm{wfa}[\tilde{\mathcal{S}}]$ (by (3.7)) and $\mathrm{wfa}[\tilde{\mathcal{S}}] \geq 6\mathrm{Diam}$ imply that $|T^1| \geq 1$.

Applying Fact 8 to (3.8) and (3.9), these two bounds are combined to

$$\frac{\mathrm{wfa}[\tilde{\mathcal{S}}]}{\mathrm{opt}[\tilde{\mathcal{S}}]} \leq \sqrt{16bcn \cdot \frac{U_{\max}}{U_{\min}} \left(\frac{U_{\min}}{\sigma} + \log(D)\right)} = \mathcal{O}\left(\sqrt{n \cdot \frac{U_{\max}}{U_{\min}} \left(\frac{U_{\min}}{\sigma} + \log(D)\right)}\right),$$

which concludes the proof. □

3.6.3 Potential Function

In this section we use the standard potential method [CLR90] to derive an upper bound on the expected cost of wfa. This is useful for proving the upper bounds for the cases of random tasks and β-elementary tasks.

Lemma 15. *Let the random variable $\mathcal{S}$ denote the smoothed sequence of length ℓ. For each t, $1 \leq t \leq \ell$, and a given node s, 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 $\mathbf{E}[\mathrm{wfa}[\mathcal{S}]] \leq 4\kappa\ell + \mathrm{Diam}$.*

Before we proceed to prove the lemma, 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 15 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 alg plus some additive term.

Proof of Lemma 15 . 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

$$\Phi(t) := w_t(s_t) + t\mathrm{Diam}/\ell.$$

Observe that

$$\Phi(\ell) - \Phi(0) = w_\ell(s_\ell) - w_0(s_0) + \mathrm{Diam} \geq w_\ell(s_\ell) - w_\ell(s_0) + \mathrm{Diam} \geq 0,$$

where the last inequality follows from Fact 5 and since $\delta(s_\ell, s_0) \leq \mathrm{Diam}$.

We define the *amortized cost* $C_a(t)$ incurred by wfa to process task π_t as

$$\begin{aligned} C_a(t) &:= r_t(s_t) + \delta(s_{t-1}, s_t) + \Phi(t) - \Phi(t-1) \\ &= r_t(s_t) + \delta(s_{t-1}, s_t) + w_t(s_t) - w_{t-1}(s_{t-1}) + \mathrm{Diam}/\ell \\ &= w_t(s_t) - w_{t-1}(s_t) + w_t(s_{t-1}) - w_{t-1}(s_{t-1}) + \mathrm{Diam}/\ell, \end{aligned} \tag{3.10}$$

where the last equality follows from Fact 7. Using Fact 5 and (3.1) 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 (3.10) as

$$C_a(t) \;\leq\; 2\Delta_t(s_t) + 2\Delta_t(s_{t-1}) + \mathrm{Diam}/\ell.$$

Since $\mathrm{wfa}[\mathcal{S}] = \sum_{t=1}^{\ell} C_a(t) - \Phi(\ell) + \Phi(0)$ and $\Phi(\ell) - \Phi(0) \geq 0$, we obtain

$$\mathbf{E}[\mathrm{wfa}[\mathcal{S}]] \leq \mathbf{E}\left[\sum_{t=1}^{\ell} C_a(t)\right] \leq 2\mathbf{E}\left[\sum_{t=1}^{\ell} (\Delta_t(s_t) + \Delta_t(s_{t-1}))\right] + \mathrm{Diam} \leq 4\kappa\ell + \mathrm{Diam}.$$

□

If $\ell \geq \mathrm{Diam}$ then the above bound reduces to $O(\kappa\ell)$. Corrollary 1 together with the upper bound of Lemma 15 yield the following corollary.

Corollary 2. *Let the random variable $\mathcal{S}$ denote the smoothed 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 $\ell \geq \mathrm{Diam}$, the smoothed competitive ratio of* wfa *is*

$$\mathbf{E}\left[\frac{\mathrm{wfa}[\mathcal{S}]}{\mathrm{opt}[\mathcal{S}]}\right] = \mathcal{O}\left(\frac{\kappa\ell}{n\gamma U_{\min}}\right) = \mathcal{O}\left(\kappa\left(\frac{1}{\sigma} + \frac{\log(D)}{U_{\min}}\right)\right).$$

3.6.4 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 3.A.

Fact 9. *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 14. *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*

$$\mathcal{O}\left(1 + \frac{\sigma}{U_{\min}} \log(D)\right).$$

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 $\ell \geq \mathrm{Diam}$. 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 9 implies that $\mathbf{E}[\Delta_t(s)] \leq \kappa := \sqrt{12}\sigma$. Thus, by Lemma 15, we have $\mathbf{E}[\mathrm{wfa}[\mathcal{S}]] = 4\sqrt{12}\sigma\ell + \mathrm{Diam} = O(\sigma\ell)$.

Note that we can use the lower bound established in Section 3.5 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. As we remarked earlier, the lower bound on opt holds even for such distributions. Thus the theorem follows from Corrollary 2. □

3.6.5 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 15. *If the adversarial task sequence only consists of β-elementary tasks then the smoothed competitive ratio of* wfa *is*

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

We state the following fact. The proof is given in Appendix 3.A.

Fact 10. *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 16. *Let s be an arbitrary node of G. In round t 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 request 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 by Fact 10 and since $r_t(v^*) = \max\{0, \varepsilon(v^*)\}$, $\varepsilon(v^*)$ is a random variable chosen from f. □

Proof of Theorem 15 . Let random variable $\mathcal{S}$ denote the smoothed sequence of length $\ell := \lceil c_2 n\gamma(U_{\min}/\sigma + \log(D))\rceil$, for an appropriate $\gamma \geq U_{\max}$. By Lemma 16, $\mathbf{E}[\Delta_t(s)] \leq \kappa := \sigma + \beta U_{\max}$, which is $O(\beta U_{\max})$ since $\sigma \leq 2U_{\min}$. The theorem now follows from Lemma 15 and Corrollary 2. □

3.7 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.

3.7.1 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 15 is tight up to a factor of $U_{\max}/U_{\min}$ if the underlying graph is a line. For such a comparison to the β-elementary upper bound (upper bound on β-elementary tasks), it is necessary that the β-elementary upper bound is stronger (lesser) than the general upper bounds of Theorem 12 and Theorem 13 for arbitrary tasks. We use this to fix the minimum term in the following lower bound expression of Theorem 16, which finally yields the necessary bound. Theorem 16 is also used later to obtain the universal lower bound of Theorem 17.

Theorem 16. *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 the standard 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 $\mathrm{opt}[\mathcal{S}]$.

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 16. Assume S_j does not receive any ∞ request cost. Then, $\mathbf{E}[C_j(t)] \leq \sigma$ by Fact 10.

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 inequality (Lemma 6), $\mathbf{Pr}[\tilde{\mathbf{B}}[\mathcal{S}] < 2\mathbf{E}[\tilde{\mathbf{B}}[\mathcal{S}]]] \geq \frac{1}{2}$. Since in each round, alg is forced to travel at least a distance of $\lceil \beta/2 \rceil$, we have $\mathrm{alg}[\mathcal{S}] \geq \ell\beta U_{\min}/2$.

We conclude

$$\mathbf{E}\left[\frac{\mathrm{alg}[\mathcal{S}]}{\mathrm{opt}[\mathcal{S}]}\right] \geq \left(\frac{1}{2}\right)\frac{\ell\beta U_{\min}/2}{2\ell\left(\frac{2(\beta U_{\max}+\sigma)}{h} + \sigma\right)} = \Omega\left(\frac{\beta U_{\min}}{\beta^2 U_{\max}/n + \sigma}\right).$$

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 a $\Omega(\beta\cdot(U_{\min}/\sigma+1))$ bound. □

Observe that on a line the β-elementary bound of Theorem 15 is stronger than the general upper bound of Theorem 13 only if

$$\beta \leq \sqrt{\frac{nU_{\min}}{U_{\max}(U_{\min}/\sigma+1)}}.$$

In this case, Theorem 16 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 $U_{\max}/U_{\min}$.

3.7.2 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 16.

Corollary 3. *Let G be a line graph. Any deterministic algorithm* alg *has smoothed competitive ratio* $\Omega\big(\min\{n, \sqrt{n(U_{\min}/U_{\max})(U_{\min}/\sigma+1)}\}\big)$ *against an adaptive adversary.*

Proof. Fix $\beta := \sqrt{nU_{\min}/(U_{\max}(U_{\min}/\sigma+1))}$ and use the lower bound given in Theorem 16. □

Theorem 17. Any *deterministic algorithm* alg *has a smoothed competitive ratio of*

$$\Omega\left(\min\left\{\mathrm{diam},\ \sqrt{\mathrm{diam}\cdot\frac{U_{\min}}{U_{\max}}\cdot\left(\frac{U_{\min}}{\sigma}+1\right)}\right\}\right).$$

Proof. We extend Theorem 16 to arbitrary graphs in a straightforward way. Consider a path in G of edge length at least 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 3. □

Next, we prove the following universal lower bound.

Theorem 18. 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 $\Omega(\min\{n, U_{\min}/\sigma\})$ when $U_{\min}/\sigma \ge 1$. 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 the 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 $\mathrm{opt}[\mathcal{S}]$. It suffices to prove that with constant probability $\mathrm{alg}[\mathcal{S}]/\tilde{\mathbf{B}}[\mathcal{S}] = \Omega(\min\{n, U_{\min}/\sigma\})$.

For the analysis, we view the smoothing process as being done in 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 $Z := \sum_{t=1}^{\ell} Z_t$. In the sequel, we condition the smoothing outcome $\mathcal{S}$ on the conjunction of following three events:

$$\mathcal{E} := \left(\sum_{v \in V} \mathbf{R}'(v) \leq 2n\ell\sigma\right) \qquad \mathcal{F} := \left(\sum_{t=1}^{\ell} U_t \leq 4\ell U_{\min}\right) \qquad \mathcal{G} := (Z \geq \ell/4)$$

We first argue that the event $(\mathcal{E} \cap \mathcal{F} \cap \mathcal{G})$ occurs with at least constant probability. (i) Due to Fact 10, $\mathbf{E}[\mathbf{R}'(v)] \leq \ell\sigma$ for each $v \in V$. By Markov inequality, we thus have $\mathbf{Pr}[\mathcal{E}] \geq 1/2$. (ii) By Fact 10 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 inequality, $\mathbf{Pr}[\sum_{t=1}^{\ell} U_t \geq 4\ell U_{\min}] \leq 1/2$. (iii) Since the smoothing distribution f is a symmetric, we have $\mathbf{Pr}[U_t \geq U_{\min}] \geq 1/2$ for each $1 \leq t \leq \ell$. Thus, $\mathbf{E}[Z_t] \geq 1/2$. Moreover, the Z_t's are independent. Applying Chernoff bound we obtain $\mathbf{Pr}[Z \leq \ell/4] \leq e^{-\ell/16}$.

Since event $\mathcal{E}$ is defined with respect to $\mathcal{S}'$, it is independent of the event $(\mathcal{F} \cap \mathcal{G})$. Therefore,

$$\mathbf{Pr}[\mathcal{E} \cap \mathcal{F} \cap \mathcal{G}] \geq \frac{1}{2} \cdot \left(1 - \left(\frac{1}{2} + e^{-\ell/16}\right)\right) \geq \frac{1}{8},$$

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,

$$\mathbf{B}[\mathcal{S}] \leq \mathrm{alg}[\mathcal{S}] + \sum_{t \in T_k} U_t + \sum_{v \in V} \mathbf{R}'(v) \leq \mathrm{alg}[\mathcal{S}] + 4\ell U_{\min} + 2n\ell\sigma,$$

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

$$\frac{\mathrm{alg}[\mathcal{S}]}{\tilde{\mathbf{B}}[\mathcal{S}]} \geq \frac{\mathrm{alg}[\mathcal{S}]}{17\mathrm{alg}[\mathcal{S}]/n + 2\ell\sigma} \geq c \cdot \min\left\{n, \frac{U_{\min}}{\sigma}\right\}.$$

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_0 by specifying in each round a request cost of ∞ for each $v \notin V_0$. 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 17 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 17 here since with respect to G_s the request sequence is elementary.) Applying Markov inequality, we obtain $\mathbf{Pr}[\mathbf{B}[\mathcal{S}] \geq 4c\ell U_{\max}/\log(D)] \leq 1/2$. 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

$$\mathbf{E}\left[\frac{\mathrm{alg}[\mathcal{S}]}{\mathrm{opt}[\mathcal{S}]}\right] \geq \mathbf{E}\left[\frac{\mathrm{alg}[\mathcal{S}]}{\mathbf{B}[\mathcal{S}]}\right] \geq \left(\frac{1}{2}\right) \cdot \frac{\ell U_{\min}}{4c\ell U_{\max}/\log(D)} = \Omega\left(\frac{U_{\min}}{U_{\max}} \cdot \log(D)\right).$$

□

Lemma 17. *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 $\mathbf{Pr}[X_i = 1] \geq (1/2)^k \geq 1/\sqrt{m}$. 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 bound we obtain

$$\mathbf{Pr}[\neg\mathcal{E}] = \mathbf{Pr}[\mathbf{X} \leq \sqrt{m}/2] \leq e^{-\sqrt{m}/8}.$$

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

$$\mathbf{E}[\mathbf{B}[\mathcal{S}]] = \mathbf{E}[\mathbf{B}[\mathcal{S}] \mid \mathcal{E}]\ \mathbf{Pr}[\mathcal{E}]\ + \mathbf{E}[\mathbf{B}[\mathcal{S}] \mid \neg\mathcal{E}]\ \mathbf{Pr}[\neg\mathcal{E}]\ \leq U_{\max} + \mathbf{E}[\mathbf{B}[\mathcal{S}] \mid \neg\mathcal{E}] \cdot e^{-\sqrt{m}/8}.$$

Next, we bound $\mathbf{E}[\mathbf{B}[\mathcal{S}] \mid \neg\mathcal{E}]$. Clearly, the transition cost in each round is at most $U_{\max}$. The expected request cost incurred by $\mathbf{B}$ in round t is $\mathbf{E}[\min_{u\in V}\{r_t(u)\} \mid \neg\mathcal{E}]$. 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 $\mathbf{E}[\min_{u\in V}\{r_t(u)\} \mid \neg\mathcal{E}]\ \leq\ \mathbf{E}[r_t(v_i) \mid \neg\mathcal{E}]$. 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

$$\mathbf{E}[r_t(v_i) \mid X_1 = x_1, \ldots, X_{m+1} = x_{m+1}]\ =\ \mathbf{E}[r_t(v_i) \mid X_i = x_i].$$

If $x_i = 1$ then $\mathbf{E}[r_t(v_i) \mid X_i = x_i] = 0$, since all request costs at v_i must be zero. If $x_i = 0$ then $\mathbf{E}[r_t(v_i) \mid X_i = x_i] \leq \mathbf{E}[r_t(v_i) \mid r_t(v_i) > 0]$. (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 10, the expected cost $\mathbf{E}[r_t(v_i)]$ is at most σ. Moreover, $\mathbf{Pr}[r_t(v_i) > 0]\ \geq \mathbf{Pr}[r_t(v_i) \geq \sigma/c_f]\ \geq \frac{1}{4}$. Hence, $\mathbf{E}[r_t(v_i) \mid r_t(v_i) > 0] \leq 4\mathbf{E}[r_t(v_i)] \leq 4\sigma$. Putting everything together, we obtain

$$\mathbf{E}[\mathbf{B}[\mathcal{S}] \mid \neg\mathcal{E}] \leq \sum_{t=1}^{k}(\mathbf{E}[\min_{u\in V}\{r_t(u)\} \mid \neg\mathcal{E}] + U_{\max}) \leq k(4\sigma + U_{\max}) \leq 9kU_{\max},$$

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. □

Remark 3. The above proof goes through even if $\sigma = e^{\Theta(\sqrt{m})}U_{\max}$, yielding the same upper bound on $\mathbf{E}[\mathbf{B}[\mathcal{S}]]$. This implies that the existential lower bounds in the next section and the universal lower bound of Theorem 18, where the above lemma is used, hold even if $\sigma = e^{\Theta(\sqrt{m})}U_{\max}$.

Remark 4. The universal lower bound of Theorem 18 is established using only elementary (1-elementary) tasks as the adversarial sequence. Hence this lower bound compares more sharply against the β-elementary upper bound if $\beta = O(1)$, than the general β-elementary lower bound.

3.7.3 Existential Lower Bounds

We provide two existential lower bounds showing that for a large range of values for the 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 12 and Theorem 13. In order to prove these existential lower bounds, we first show the following lemma.

Lemma 18. *Given, number of nodes n, minimum edge cost $U_{\min}$, maximum edge cost $U_{\max}$, maximum degree $D \geq 3$, and diameter Diam such that*

$$\mathrm{Diam} \geq 4U_{\min}\log_{D-1}(n), \quad \textit{and} \quad \mathcal{D} := \min\{\mathrm{Diam}/U_{\max}, D\} \geq 17,$$

there exists a graph such that the smoothed competitive ratio of any deterministic algorithm alg *is*

$$\Omega\left(\min\left\{\frac{nU_{\max}}{\mathrm{Diam}}, \frac{\mathrm{Diam}}{U_{\min}}\cdot\left(\frac{U_{\min}}{\sigma}+\log(\mathcal{D})\right)\right\}\right).$$

Note. We would like to point out that in any graph of n nodes and maximum degree D, $\mathrm{Diam} \geq U_{\min}\log_{D-1}(n)$. Hence the restriction on Diam in the above lemma is only slightly stronger, i.e., by a constant factor.

Proof of Lemma 18 . We construct a graph G as depicted in Figure 3.2. The graph consists of $m := \frac{1}{2}nU_{\max}/\mathrm{Diam}$ 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 $2(U_{\min} + X\cdot U_{\max} + (h-1)U_{\min}) = \mathrm{Diam}$, i.e., $X := \lceil(\mathrm{Diam}/2 - hU_{\min})/U_{\max}\rceil$. Moreover, we want that the minimum distance between any two nodes in different cliques is at least $\frac{1}{4}\mathrm{Diam}$, i.e., $X\cdot U_{\max} \geq \frac{1}{8}\mathrm{Diam}$. If $\mathrm{Diam} \geq 4U_{\min}\log_{D-1}(n)$, this condition holds. (Also observe that in any graph of n

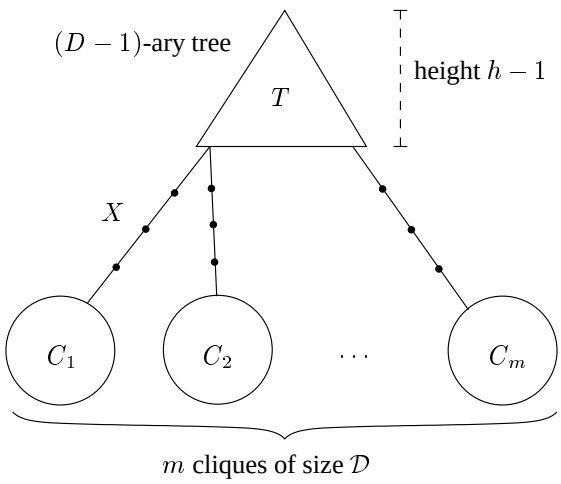


Fig. 3.2: The lower bound graph. Each C_i is a $\mathcal{D}$-clique which is connected to the central tree T by a path having X edges.

nodes and maximum degree D, $\mathrm{Diam}/U_{\min} \geq \log_{D-1}(n)$, i.e., our condition is only slightly stronger.)

Consider the case $U_{\min}/\sigma > \log(\mathcal{D})$. We need to prove a lower bound of $\Omega\left(\min\{nU_{\max}/\mathrm{Diam}, \mathrm{Diam}/\sigma\}\right)$. 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 $U_{\min} = \frac{1}{4}\mathrm{Diam}$ and $U_{\max} = \mathrm{Diam}$. Applying the universal lower bound of Theorem 18 to this clique we obtain the desired lower bound of $\Omega(\min\{m, \mathrm{Diam}/\sigma\})$.

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.

We use the standard 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 17. We may assume without loss of generality that each $\mathbf{B}_i$ starts in a different clique (see Appendix 3.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 $\ell\mathrm{Diam}$. The expected cost of each algorithm in a clique with zero adversarial request cost is, due to Lemma 17, 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,

$$\mathbf{E}[\tilde{\mathbf{B}}[\mathcal{S}]] \leq \frac{\ell \mathrm{Diam}}{m-1} + \frac{1}{m-1}\mathbf{E}\left[\sum_{i=1}^{m-1} \mathbf{B}_i[\mathcal{S}]\right] \leq \frac{\ell \mathrm{Diam}}{m-1} + \frac{c\ell U_{\min}}{\log(\mathcal{D}-1)}.$$

By Markov inequality, $\mathbf{Pr}[\tilde{\mathbf{B}}[\mathcal{S}] < 2\mathbf{E}[\tilde{\mathbf{B}}[\mathcal{S}]]] \geq \frac{1}{2}$. Clearly, $\mathrm{alg}[\mathcal{S}] \geq \frac{1}{4}\ell\mathrm{Diam}$. Therefore,

$$\mathbf{E}\left[\frac{\mathrm{alg}[\mathcal{S}]}{\mathrm{opt}[\mathcal{S}]}\right] \geq \left(\frac{1}{2}\right) \frac{\frac{1}{4}\ell\mathrm{Diam}}{2(\frac{\ell\mathrm{Diam}}{m-1} + \frac{c\ell U_{\min}}{\log(\mathcal{D}-1)})} = \Omega\left(\min\left\{m, \frac{\mathrm{Diam}}{U_{\min}} \cdot \log(\mathcal{D})\right\}\right).$$

□

The next bound shows that if Theorem 12 gives a better upper bound than Theorem 13 then this bound is tight up to a factor of $\log(D)/\log(\mathcal{D}) \leq \log(n)$ for a large class of graphs.

Theorem 19. *There exists a class of graphs such that the smoothed competitive ratio of any deterministic algorithm* alg *is*

$$\Omega\left(\min\left\{n, \frac{\mathrm{Diam}}{U_{\min}}\left(\frac{U_{\min}}{\sigma} + \log(\mathcal{D})\right)\right\}\right),$$

where $\mathcal{D} = \min\{\mathrm{Diam}/U_{\min}, D\}$.

Proof. If Theorem 12 gives a better upper bound than Theorem 13, we have

$$\frac{\mathrm{Diam}}{U_{\min}}\left(\frac{U_{\min}}{\sigma} + \log(D)\right) \leq \sqrt{n \cdot \frac{U_{\max}}{U_{\min}}\left(\frac{U_{\min}}{\sigma} + \log(D)\right)},$$

which is equivalent to

$$\frac{nU_{\max}}{\mathrm{Diam}} \geq \frac{\mathrm{Diam}}{U_{\min}}\left(\frac{U_{\min}}{\sigma} + \log(D)\right).$$

Since $\log(D) \geq \log(\mathcal{D})$, we obtain from Lemma 18 the desired lower bound. □

Theorem 20. *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 $\mathcal{D} = \min\{\mathrm{Diam}/U_{\min}, D\}$.

Proof. Let $U_{\min}/\sigma > \log(\mathcal{D})$. We fix Diam such that $nU_{\max}/\mathrm{Diam} = \mathrm{Diam}/\sigma$, i.e., $\mathrm{Diam} = \sqrt{n\sigma U_{\max}}$. The lower bound of Lemma 18 then reduces to $\Omega(\sqrt{nU_{\max}/\sigma})$.

Assume $U_{\min}/\sigma \leq \log(\mathcal{D})$. We fix Diam such that $nU_{\max}/\mathrm{Diam} = (\mathrm{Diam}/U_{\min})\log(\mathcal{D})$, i.e., $\mathrm{Diam} = \sqrt{nU_{\max}U_{\min}/\log(\mathcal{D})}$. The lower bound of Lemma 18 then reduces to $\Omega\big(\sqrt{n(U_{\max}/U_{\min})\log(\mathcal{D})}\big)$. □

3.8 Concluding Remarks

In this chapter, we focused on 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 already for $\sigma = \Theta(U_{\min})$, the smoothed competitive ratio of WFA is much better than its worst-case competitive ratio suggests and that it depends on topological parameters of the underlying graph. Moreover, all our bounds, except the one for β-elementary tasks, are tight up to constant factors on a large class of graphs when σ ranges between 0 and $\Theta(U_{\min})$. We believe that our analysis gives a strong indication that the performance of WFA in practice is much better than $2n-1$. An open problem is to strengthen the universal lower bounds.

3.A Proofs of Facts

Proof of Fact 5 . 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 6 . By (3.2), 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, $w_t(s_t) \geq w_t(s_{t-1}) - \delta(s_{t-1},s_t)$. □

Proof of Fact 7 . Using (3.2) and Fact 6, 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 8 . Define $\mathcal{X} := \min\left\{\frac{A\sum_{i=1}^m X_i}{\sum_{i=1}^m X_i^2},\ \frac{B\sum_{i=1}^m X_i}{m}\right\}$. First, note that

$$m(X_1^2 + X_2^2 + \cdots + X_m^2) \ \geq\ (X_1 + X_2 + \cdots + X_m)^2, \tag{3.11}$$

because

$$\frac{1}{2}\sum_{i,j}\left(X_i^2 + X_j^2\right) \geq \sum_{i=1}^m X_i^2 \ + \sum_{i,j,i\neq j} X_iX_j.$$

Define $Y := \sum_{i=1}^m X_i/m$. Note that Y is positive. Due to (3.11), 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 $\mathcal{X} \leq \sqrt{AB}$. □

Proof of Fact 9. Let X be a random variable chosen from f. Define $\mathcal{E}$ as the event $(|X-\mu| \geq \mu/2)$. Using Chebyshev inequality (Lemma 7), we obtain

$$\mathbf{Pr}[\mathcal{E}] = \mathbf{Pr}\left[|X-\mu| \geq \frac{\mu}{2}\right] \leq \frac{4\sigma^2}{\mu^2}. \tag{3.12}$$

Since f is continuous and non-increasing in $[0,\infty)$,

$$\mathbf{Pr}[\mathcal{E}] = \mathbf{Pr}\left[|X-\mu| \geq \frac{\mu}{2}\right] \geq \mathbf{Pr}\left[X \leq \frac{\mu}{2}\right] \geq \frac{1}{2}\mathbf{Pr}\left[\frac{\mu}{2} < X \leq \frac{3\mu}{2}\right] \geq \frac{1}{2}\mathbf{Pr}[\neg\mathcal{E}]\,.$$

This implies that $\mathbf{Pr}[\mathcal{E}] \geq \frac{1}{3}$. Hence, (3.12) gives $\mu^2 \leq 12\sigma^2$. □

Proof of Fact 10 . 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 $\mathbf{E}[Y] \leq \sigma/\sqrt{2}$. □

3.B Constant Additive Cost for the Offline Algorithm

We note 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. This 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 $\Omega((Y \cdot \ell)/(X \cdot \ell + Z)) = \Omega(Y/X)$.

3.C Proof that WFA is well-defined

Proof of Lemma 5. [BEY98] Define the set $A' = \{y : y \in V \text{ and } y = \arg\min_x\{w_{t+1}(x) + \delta(s_t, x)\}\}$. Clearly, A' is not empty because the set of nodes V is finite ad not empty. We will prove that there is an element of A' that also satisfies the second condition in the lemma. This will prove that the set A is not empty.

For each $x \in V$, $w_{t+1}(x) \leq w_t(x) + r_{t+1}(x)$. This is because the optimal way to process the sequence $\check{\mathcal{S}}_{t+1}$ ending in state x is surely no more costly than optimally processing $\check{\mathcal{S}}_t$ ending in x and then processing r_{t+1} from x. Adding $\delta(x, s_t)$ to both sides of this inequality, we get,

$$w_{t+1}(x) + \delta(x, s_t) \leq w_t(x) + r_{t+1}(x) + \delta(x, s_t). \tag{3.13}$$

Let z be an element of A and let x^* be the state for which the minimum in equation (3.1) with $v = z$ is obtained. That is,

$$w_{t+1}(z) = w_t(x^*) + r_{t+1}(x^*) + \delta(x^*, z). \tag{3.14}$$

Adding $\delta(x^*, s_t)$ to above equation and rearranging terms, we obtain,

$$w_t(x^*) + r_{t+1}(x^*) + \delta(x^*, s_t) \;\; = \;\; w_{t+1}(z) + \delta(x^*, s_t) - \delta(x^*, z). \tag{3.15}$$

Using the triangle inequality $\delta(x^*, s_t) - \delta(x^*, z) \leq \delta(z, s_t)$, inequality (3.13), equation (3.15), we get,

$$w_{t+1}(x^*) + \delta(x^*, s_t) \;\leq\; w_{t+1}(z) + \delta(z, s_t). \tag{3.16}$$

Therefore, since $z = \arg\min_x\{w_{t+1}(z) + \delta(x, s_t)\}$, it must be that $x^* \in A'$. Moreover, inequality (3.16) must be an equality. Therefore, when $x = x^*$, inequality (3.13) must also be an equality. Hence x^* is an element of A. □

Chapter 4

Randomized Pursuit-Evasion in Graphs

4.1 Introduction

In this chapter we study a pursuit-evasion game on graphs. In this round-based game, a pursuer tries to catch an evader (the adversary) while they both travel from node to node of a connected, undirected graph G. We also refer to these players as *Hunter* and *Rabbit* respectively. The hunter catches the rabbit when in some round the hunter and the rabbit are both located on the same node of the graph. We assume that both players know the graph in advance but they cannot see each other until the rabbit gets caught. Both players may use a randomized (also called *mixed*) strategy, where each player is oblivious to the random choices made by the other player. That is, each player has a secure source of randomness which cannot be observed by the other player. In this setting we study upper bounds (i.e., good hunter strategies) as well as lower bounds (i.e., good rabbit strategies) on the expected number of rounds until the hunter catches the rabbit.

The problem we address is motivated by the question of how long it takes a single pursuer to find an evader on a given graph that, for example, corresponds to a computer network or to a map of a terrain in which the evader is hiding. A natural assumption is that both the pursuer and the evader have to follow the edges of the graph. In some cases however it might be that the evader has more advanced possibilities than the pursuer in the terrain where he is hiding. Therefore we additionally consider a stronger adversarial model in which the evader is allowed to jump arbitrarily between nodes of the graph. Such a jump between nodes corresponds to a short-cut between two places which is only known to the evader (like a rabbit using rabbit holes). Obviously, a strategy that is efficient against an evader that can jump is efficient as well against an evader who may only move along the edges of the graph.

Publication Notes. A preliminary version of this joint work, together with Micah Adler, Harald Räcke, Christian Sohler and Berthold Vöcking, was published in the proceedings of the 29th International Colloquium on Automata, Languages, and Programming (ICALP), 2002 [ARS+02]. A journal version of this work appeared in Combinatorics, Probability and Computing [ARS+03].

One approach to use for a hunter strategy would be to perform a random walk on the graph G. Unfortunately, the hitting time of a random walk, i.e., the expected number of rounds needed to reach another node, can be as large as $\Omega(n^3)$ with n denoting the number of nodes [MR95]. Thus it would require at least $\Omega(n^3)$ rounds to find a rabbit even if the rabbit does not move at all. We show that one can do significantly better. In particular, we prove that for any graph G with n nodes there is a hunter strategy such that the expected number of rounds until a rabbit that is not necessarily restricted to the graph is caught is $O(n \log n)$ rounds. Furthermore we show that this result cannot be improved in general as there is a graph with n nodes and an unrestricted rabbit strategy such that the expected number of rounds required to catch this rabbit is $\Omega(n \log n)$ for any hunter strategy.

4.2 Related Work

Search games have a long history in the field of game theory: In 1965 Isaacs introduced the so-called *Princess-Monster* game [Isa65]. In this game a (highly intelligent) monster tries to capture a princess in a totally dark room $\mathcal{D}$ with arbitrary shape. Both the monster and the princess are aware of the boundary of the room and the monster catches the princess if their mutual distance becomes smaller than some threshold (which is small in comparison with the extension of $\mathcal{D}$). The monster moves at a known speed using simple motion, that is, the monster moves along continuous trajectories inside $\mathcal{D}$. The princess moves along continuous trajectories but at arbitrary speed.

Since the general game seemed to be hard to analyze Isaacs also introduced a simpler Princess-Monster game where both the princess and the monster are moving on a closed curve taken as a circle. This game has been analyzed several years later by Alpern [Alp74] and Zelekin [Zel72]. Finally, Gal presented an analysis of the Princess-Monster game in a convex multidimensional region [Gal79].

The Hunter vs. Rabbit game is a *discrete* variant of the Princess-Monster game that is played in rounds. The most important difference between the two variants is that in our case the rabbit (the princess) can use short-cuts not known to the hunter (the monster), that is, the rabbit is allowed to 'jump' from a node to any other node of the graph. Further the rabbit is only caught if at the end of a round it is on the same node as the hunter. During the motion the rabbit cannot be caught.

A first study of the Hunter vs. Rabbit game can be found in [AKL$^+$79]. The presented hunter strategy is based on a random walk on the graph and it is shown that the hunter catches an unrestricted rabbit within $O(nm^2)$ rounds, where n and m denote the number of nodes and edges, respectively. In fact, the authors place some additional restrictions on the space requirements for the hunter strategy, which is an aspect that we do not consider.

In the area of mobile ad-hoc networks, related models are used to design communication protocols (see e.g. [CNP$^+$01, CNS01, BK87]). In this scenario, some mobile users (the "hunters") aid in transmitting messages to the receivers (the "rabbits"). The expected number

of rounds needed to catch the rabbit in our model corresponds directly to the expected time needed to deliver a message. We improve the deliver time of known protocols, which are based on random walks.

Deterministic pursuit-evasion games in graphs are well-studied. In the early work by Parsons [Par76, Par78] the graph was considered to be a system of tunnels in which a fugitive is hiding. Parsons introduced the concept of the *search number* of a graph which is, informally speaking, the minimum number of guards needed to capture a fugitive who can move with arbitrary speed. LaPaugh [LaP93] showed that if ℓ guards are sufficient to capture the fugitive then this can be done without re-contamination, i.e., if at any point of time the fugitive is known not to be in edge e then there is no chance for him to enter edge e without being caught in the remainder of the game. Meggido et al. [MHG$^+$88] proved that the computation of the search number of a graph is an NP-hard problem which implies its NP-completeness because of LaPaugh's result.

If an edge can be cleared without moving along it, but it suffices to 'look into' an edge from a node, then the minimum number of guards needed to catch the fugitive is called the *node search number* of a graph [KP86]. Connections between the search number and the fundamental graph parameters like vertex separation were studied in [EST94].

Pursuit evasion problems in the plane were introduced by Suzuki and Yamashita [SY92]. They gave necessary and sufficient conditions for a simple polygon to be searchable by a single pursuer. Later Guibas et al. [GLL$^+$99] presented a complete algorithm and showed that the problem of determining the minimal number of pursuers needed to clear a polygonal region with holes is NP-hard. Recently, Park et al. [PLC01] gave three necessary and sufficient conditions for a polygon to be searchable and showed that there is an $O(n^2)$ time algorithm for constructing a search path for an n-sided polygon.

Efrat et al. [EGHP$^+$00] gave a polynomial time algorithm for the problem of clearing a simple polygon with a chain of k pursuers when the first and last pursuer have to move on the boundary of the polygon.

In our setting the hunter and the rabbit have no visibility, they can see each other only if they are in the same node. Hunter strategies in case of full-visibility have also been studied (with restricted rabbit) [NW83, BW00]. The number of hunters needed to find the rabbit is called the cop number. It is known that the cop number of planar graphs is at most 3 [AF84] but the case of general graphs is still open [NN98, FN01].

4.3 Our Contribution

We present a hunter strategy for general networks that improves significantly on the results obtained by using random walks. Let $G = (V, E)$ denote a connected graph with n nodes and diameter $\mathrm{diam}(G)$, which is the maximum over all pairwise shortest path distances in G. Observe that $\Omega(n)$ is a lower bound on the escape length against restricted as well as against unrestricted rabbit strategies on every graph with n nodes (the rabbit chooses its first

node uniformly at random and does not move during the game). Our hunter strategy achieves escape length close to this lower bound. In particular, we present a hunter strategy that has an expected escape length of only $O(n \log(\mathrm{diam}(G)))$ against any unrestricted rabbit strategy. Clearly, an upper bound on the escape length against unrestricted rabbit strategies implies the same upper bound against restricted strategies.

Our general hunter strategy is based on a hunter strategy for cycles which is then simulated on general graphs. Observe that if hunter and rabbit are restricted to a cycle, then there is a simple, efficient hunter strategy with escape length $O(n)$ - in every nth round, the hunter chooses a *direction* at random, either clockwise or counterclockwise, and then he follows the cycle in this direction for the next n rounds. Against unrestricted rabbits, however, the problem of devising efficient hunter strategies becomes much more challenging. For example, for the hunter strategy given above, the following simple rabbit strategy results in an escape length of $\Theta(n\sqrt{n})$. In each phase of n rounds, the rabbit first chooses a direction and a starting position at random. Now it sweeps the cycle for $\sqrt{n}$ rounds and then jump back by a length of $2\sqrt{n}$ and then repeats this. For unrestricted rabbits on cycles of length n, we present a hunter strategy with escape length $O(n \log n)$. Furthermore, we prove that this result is optimal by devising an unrestricted rabbit strategy with escape length $\Omega(n \log n)$ against any hunter strategy on the cycle.

Generalizing the lower bound for cycles, we can show that our general hunter strategy is optimal in the sense that for any positive integers n, d with $d < n$ there exists a graph G with n nodes and diameter d such that any hunter strategy on G has escape length $\Omega(n \cdot \log(d))$. This gives rise to the question whether $n \cdot \log(\mathrm{diam}(G))$ is a universal lower bound on the escape length in any graph. We can answer this question negatively. In fact, we present a hunter strategy with escape length $O(n)$ for complete binary trees against unrestricted rabbits.

Finally, we investigate the Hunter vs. Rabbit game on strongly connected directed graphs. We show that there exists a directed graph for which every hunter needs $\Omega(n^2)$ rounds to catch a restricted rabbit. Furthermore, for every strongly connected directed graph, there is a hunter strategy with escape length $O(n^2)$ against unrestricted rabbits.

4.4 Preliminaries

Definition of the game. The Hunter vs. Rabbit game is a round-based game that is played on an undirected connected graph $G = (V, E)$ without self loops and multiple edges. In this game there are two players – the hunter and the rabbit – moving on the nodes of G. The hunter tries to catch the rabbit, i.e., he tries to move to the same node as the rabbit, and the rabbit tries not to be caught.

During the game both players cannot “see” each other, i.e., a player has no information about the movement decisions made by his opponent and thus does not know his position in the graph. The only interaction between both players occurs when the game ends because the hunter and the rabbit move to the same node in G and the rabbit is caught. Therefore

the movement decisions of both players do not depend on each other. We want to find good strategies for both hunter and rabbit. Strategies are defined as follows:

Definition 1. *A pure strategy for a player in the Hunter vs. Rabbit game on a graph $G = (V, E)$ is a sequence $\mathcal{S} = \mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2, \ldots$, where $\mathcal{S}_t \in V$ denotes the position of the player in round $t \in \mathbb{N}_0$ of the game. A mixed strategy $\mathfrak{S}$ for a player is a probability distribution over the set of pure strategies.*

Note that both players may use mixed strategies, i.e., we assume that they both have a secure source of random bits for randomizing their movements on the graph.

For two pure strategies $\mathcal{H}$ and $\mathcal{R}$ of hunter and rabbit, respectively, the *escape length* $\mathrm{el}(\mathcal{H}, \mathcal{R}) := \min\{t \in \mathbb{N}_0 \mid \mathcal{H}_t = \mathcal{R}_t\}$ is the number of rounds until the rabbit is caught. Similarly, $\mathrm{el}(\mathfrak{H}, \mathfrak{R})$ denotes the *expected escape length* for two mixed strategies $\mathfrak{H}$ and $\mathfrak{R}$.

We analyze for both players the best expected escape length the player can guarantee for himself, regardless of what the other player does. This means we give asymptotically tight bounds on $\min_{\mathfrak{H}} \max_{\mathfrak{R}} \mathrm{el}(\mathfrak{H}, \mathfrak{R})$ for the hunter and on $\max_{\mathfrak{R}} \min_{\mathfrak{H}} \mathrm{el}(\mathfrak{H}, \mathfrak{R})$ for the rabbit, where the maxima and minima are taken over all mixed hunter and rabbit strategies, respectively.

As mentioned in the previous section we assume that the hunter cannot change his position arbitrarily between two consecutive rounds but has to follow the edges of G. To model this we call a pure strategy $\mathcal{S}$ *restricted* (to G) if either $(\mathcal{S}_t, \mathcal{S}_{t+1}) \in E$ or $\mathcal{S}_t = \mathcal{S}_{t+1}$ holds for every $t \in \mathbb{N}_0$. A (mixed) strategy is called restricted if it is a probability distribution over the set of restricted pure strategies. For the analysis we will consider only restricted strategies for the hunter and both restricted and unrestricted strategies for the rabbit.

Notice that in our definition, the hunter may start his walk on the graph at an arbitrary node. However, we want to point out that defining a fixed starting position for the hunter would not asymptotically affect the results.

Basic Concepts

The strategies will be analyzed in phases. A phase consists of m consecutive rounds, where m will be defined depending on the context. Suppose that we are given an m-round hunter strategy $\mathfrak{H}$ and an m-round rabbit strategy $\mathfrak{R}$ for a phase. We want to determine the probability that the rabbit is caught during the phase. Therefore we introduce the indicator random variables $hit(t), 0 \leq t < m$ for the event $\mathcal{H}_t = \mathcal{R}_t$ that the pure hunter strategy $\mathcal{H}$ and the pure rabbit strategy $\mathcal{R}$ chosen according to $\mathfrak{H}$ and $\mathfrak{R}$, respectively, meet in round t of the phase. Furthermore, we define indicator random variables $\mathit{fhit}(t), 0 \leq t < m$ describing first hits, i.e., $\mathit{fhit}(t) = 1$ iff $hit(t) = 1$ and $hit(t') = 0$ for every $t' \in \{0, \ldots, t-1\}$. Finally we define $hits = \sum_{t=0}^{m-1} hit(t)$.

The goal of our analysis is to derive upper and lower bounds for $\mathbf{Pr}[hits \geq 1]$, the probability that the rabbit is caught in the phase. To analyze the quality of an m-round rabbit strategy

we fix a pure hunter strategy $\mathcal{H}$ and derive a lower bound on the probability $\mathbf{Pr}[hits \geq 1]$ using the following proposition which follows trivially from the definitions.

Proposition 1. *Let $\mathfrak{R}$ be an m-round rabbit strategy and let $\mathcal{H}$ be a pure m-round hunter strategy. Then*

$$\mathbf{Pr}\left[hits \geq 1\right] = \frac{\mathbf{E}\left[hits\right]}{\mathbf{E}\left[hits \mid hits \geq 1\right]} \ .$$

Similarly, to analyze the quality of an m-round hunter strategy we fix a pure rabbit strategy and apply the following proposition, which is known as the Second Moment method.

Proposition 2. *Let $\mathfrak{H}$ be an m-round hunter strategy and let $\mathcal{R}$ be a pure m-round rabbit strategy. Then*

$$\mathbf{Pr}\left[hits \geq 1\right] \geq \frac{\mathbf{E}\left[hits\right]^2}{\mathbf{E}\left[hits^2\right]} \ .$$

Proof. We consider the conditional expectations $\mathbf{E}\left[hits \mid hits \neq 0\right]$ and $\mathbf{E}\left[hits^2 \mid hits \neq 0\right]$. For these we have

$$\mathbf{E}\left[hits^2 \mid hits \neq 0\right] - \mathbf{E}\left[hits \mid hits \neq 0\right]^2 = \mathbf{Var}\left[hits \mid hits \neq 0\right] \geq 0 \ .$$

By using $\mathbf{E}\left[hits \mid hits \neq 0\right] = \frac{\mathbf{E}[hits]}{\mathbf{Pr}[hits \neq 0]}$ and $\mathbf{E}\left[hits^2 \mid hits \neq 0\right] = \frac{\mathbf{E}[hits^2]}{\mathbf{Pr}[hits \neq 0]}$ we get

$$\frac{\mathbf{E}\left[hits^2\right]}{\mathbf{Pr}\left[hits \neq 0\right]} \geq \frac{\mathbf{E}\left[hits\right]^2}{\mathbf{Pr}\left[hits \neq 0\right]^2}$$

which yields the lemma since $\mathbf{Pr}[hits \geq 1] = \mathbf{Pr}[hits \neq 0]$. □

Note that in both cases a bound against *all* pure strategies of the other player implies the same bound against mixed strategies, as well.

4.5 Efficient Hunter Strategies

In this section we prove that for a graph G with n nodes and diameter $\mathrm{diam}(G)$, there exists a hunter strategy such that for every rabbit strategy the expected escape length is $O(n \cdot \log(\mathrm{diam}(G)))$. For this general strategy we cover G with a set of small cycles and then use a subroutine for searching these cycles. We first describe this subroutine: an efficient hunter strategy for catching the rabbit on a cycle. The general strategy is described in section 4.5.2.

4.5.1 Strategies for Cycles and Circles

We prove that there is an $O(n)$-round hunter strategy on an n-node cycle that has a probability of catching the rabbit of at least $\frac{1}{2H_n+1} = \Omega(\frac{1}{\log(n)})$, where H_n is the n^{th} harmonic number, which is defined as $\sum_{i=1}^{n} \frac{1}{i}$. Clearly, by repeating this strategy until the rabbit is caught we get a hunter strategy such that for every rabbit strategy the expected escape length is $O(n \cdot \log(n))$. In order to keep the description of the strategy as simple as possible, we introduce a continuous version of the Hunter vs. Rabbit game for cycles. In this version the hunter tries to catch the rabbit on the boundary of a circle with circumference n. The rules are as follows. In every round the hunter and the rabbit reside at arbitrary, i.e., continuously chosen points on the boundary of the circle. The rabbit is allowed to jump, i.e., it can change its position arbitrarily between two consecutive rounds whereas the hunter can cover at most a distance of one. For the notion of *catching*, we partition the boundary of the circle into n distinct half open intervals of length one. The hunter catches the rabbit if and only if there is a round in which both the hunter and the rabbit reside in the same interval. Since each interval of the boundary corresponds directly to a node of the cycle and vice versa we can make the following observation.

Observation 7. *Every hunter strategy for the Hunter vs. Rabbit game on the circle with circumference n can be simulated on the n-node cycle, achieving the same expected escape length.*

The $O(n)$-round hunter strategy for catching the rabbit on the circle consists of two phases that work as follows. In an *initialization phase* that lasts for $\lceil n/2 \rceil$ rounds the hunter first selects a random position on the boundary as the *starting position* of the following *main phase*. Then the hunter goes to this position. Note that $\lceil n/2 \rceil$ rounds suffice for the hunter to reach any position on the circle boundary. We will not care whether the rabbit gets caught during the initialization phase. Therefore there is no need for specifying the exact route taken by the hunter to get to the starting position.

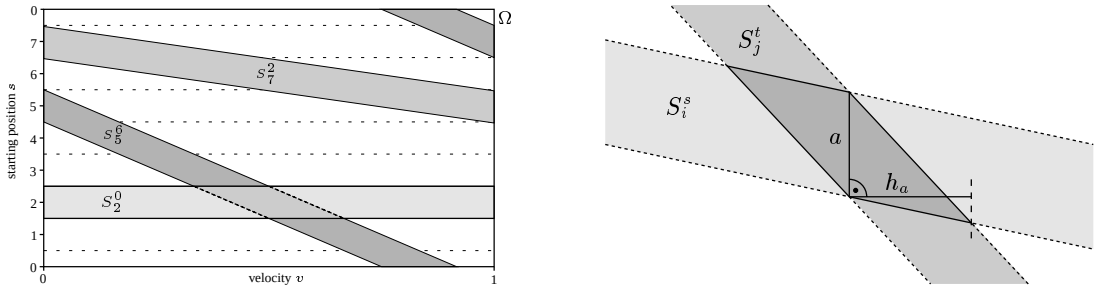
After the first $\lceil n/2 \rceil$ rounds the *main phase* starts, which lasts for n rounds. The hunter selects a velocity uniformly at random between 0 and 1 and proceeds in clockwise direction according to this velocity. This means that a hunter with starting position $s \in [0, n)$ and velocity $v \in [0, 1]$ resides at position $(s + t \cdot v) \bmod n$ in the tth round of the main phase. This strategy is called the RANDOMSPEED-strategy. Clearly, it takes exactly $\lceil \frac{3}{2} n \rceil = O(n)$ rounds. The following analysis shows that it achieves the desired probability of catching the rabbit when simulated on the n-node cycle.

Theorem 21. *On an n-node cycle a hunter using the* RANDOMSPEED*-strategy catches the rabbit with probability at least $\frac{1}{2H_n+1} = \Omega(\frac{1}{\log(n)})$.*

Proof. We prove that the bound holds for the Hunter vs. Rabbit game on the circle. The theorem then follows from Observation 7.

Since the rabbit strategy is oblivious in the sense that it does not know the random choices made by the hunter we can assume that the rabbit strategy is fixed in the beginning before the hunter starts. Consider an arbitrary pure rabbit strategy $\mathcal{R} = \mathcal{R}_0, \mathcal{R}_1, \ldots, \mathcal{R}_{n-1}$, i.e., $\mathcal{R}_t$ is the interval containing the rabbit in round t of this phase.

For this rabbit strategy let $hits$ denote a random variable counting how often the hunter catches the rabbit. This means $hits$ is the number of rounds during the main phase in which the hunter and the rabbit reside in the same interval. The theorem follows by showing that for any rabbit strategy $\mathcal{R}$ the probability $\mathbf{Pr}[hits \geq 1] = \mathbf{Pr}[\text{hunter catches rabbit}]$ is at least $\frac{1}{2H_n+1}$. For this purpose we estimate $\mathbf{E}[hits]$ and $\mathbf{E}[hits^2]$ and use Proposition 2 to derive a bound for $\mathbf{Pr}[hits \geq 1]$. Let $\Omega = [0, n) \times [0, 1]$ denote the sample space of the random experiment performed by the hunter. Further let $S_i^t \subset \Omega$ denote the subset of random choices such that the hunter resides in the $i$$th$ interval during the $t$$th$ round of the main phase. The hunter catches the rabbit in round t iff his random choice $\omega \in \Omega$ is in the set $S^t_{\mathcal{R}_t}$, which we denote by the indicator function $S^t_{\mathcal{R}_t}(\omega)$.



(a) (b)

Fig. 4.1: (a) The sample space Ω of the RANDOMSPEED strategy can be viewed as the surface of a cylinder. The sets S_i^t correspond to stripes on this surface. (b) The intersection between two stripes of slope $-s$ and $-t$, respectively.

The following interpretation of the sets S_i^t will help in deriving bounds for $\mathbf{E}[hits]$ and $\mathbf{E}[hits^2]$. We represent Ω as the surface of a cylinder as shown in Figure 4.1(a) (the y-axis wraps around). The y values correspond to the positions on the circle and the x values correspond to the different velocities. A set S_i^t corresponds to a stripe around the cylinder that has slope $\frac{\mathrm{d}s}{\mathrm{d}v} = -t$ and area 1. To see this recall that a point $\omega = (s, v)$ belongs to the set S_i^t iff the hunter position p_t in round t resulting from the random choice ω lies in the ith interval I_i. Since $p_t = (s + t \cdot v) \bmod n$ according to the RANDOMSPEED-strategy we can write S_i^t as $\{(s, v) \mid s = (p_t - t \cdot v) \bmod n \wedge p_t \in I_i\}$ which corresponds to a stripe of slope $-t$. For the area, observe that all n stripes S_i^t of a fixed slope t together cover the whole area of the cylinder which is n. Therefore each stripe has the same area of 1. A pure strategy of the rabbit thus corresponds to covering this surface with n bars, each with a different slope from $\{0, -1, \ldots, -(n-1)\}$. The hunter strategy corresponds to throwing a dart (choosing a point) on this surface uniformly at random. We are interested in the probability that the dart hits the

covered region. For this we first estimate $\mathbf{E}[hits]$ as

$$\mathbf{E}[hits] = \mathbf{E}\left[\sum_{t=0}^{n-1} hit(t)\right] = \sum_{t=0}^{n-1} \mathbf{E}[hit(t)] = \sum_{t=0}^{n-1} \int_\Omega \frac{1}{n} S^t_{\mathcal{R}_t}(\omega)\, d\omega = 1 \tag{4.1}$$

Note that $\int_\Omega S^t_{\mathcal{R}_t}(\omega)\, d\omega$ is the area of a stripe and that $\frac{1}{n}$ is the density of the uniform distribution over Ω.
We now provide an upper bound on $\mathbf{E}\left[hits^2\right]$. By definition of $hits$ we have,

$$\begin{aligned}\mathbf{E}\left[hits^2\right] &= \mathbf{E}\left[\left(\sum_{t=0}^{n-1} hit(t)\right)^2\right] = \mathbf{E}\left[\sum_{s=0}^{n-1}\sum_{t=0}^{n-1} hit(s)\cdot hit(t)\right] \\ &= \sum_{s=0}^{n-1}\sum_{t=0}^{n-1} \int_\Omega \frac{1}{n} S^s_{\mathcal{R}_s}(\omega)\cdot S^t_{\mathcal{R}_t}(\omega)\, d\omega \ .\end{aligned} \tag{4.2}$$

$S^s_{\mathcal{R}_s}(\omega)\cdot S^t_{\mathcal{R}_t}(\omega)$ is the indicator function of the intersection between $S^s_{\mathcal{R}_s}$ and $S^t_{\mathcal{R}_t}$. Therefore $\int_\Omega S^s_{\mathcal{R}_s}(\omega)\cdot S^t_{\mathcal{R}_t}(\omega)\, d\omega$ is the area of the intersection of two stripes and can be bounded using the following lemma.

Lemma 19. *The area of the intersection between two stripes S^s_i and S^t_j with $s,t \in \{0,\dots,n-1\}$, is at most $\frac{1}{|t-s|}$.*

Proof. W.l.o.g. we assume $t > s$. Figure 4.1(b) illustrates the case where the intersection between both stripes is maximal. Note that the limitation for the slope values together with the size of the cylinder surface ensure that the intersection is contiguous. This means the stripes only "meet" once on the surface of the cylinder.

By the definition of S^s_i and S^t_j the length of the straight line a in the Figure corresponds to the length of an interval on the boundary of the circle. Thus $a = 1$. The length of h_a is $\frac{a}{t-s}$ and therefore the area of the intersection is $a \cdot h_a = \frac{a^2}{t-s} = \frac{1}{t-s}$. This yields the lemma. □

Using this Lemma we get

$$\begin{aligned}\sum_{t=0}^{n-1} \int_\Omega S^s_{\mathcal{R}_s}(\omega)\cdot S^t_{\mathcal{R}_t}(\omega)\, d\omega &\le \sum_{t=0}^{s-1} \frac{1}{|t-s|} + \int_\Omega S^s_{\mathcal{R}_s}(\omega)\cdot S^s_{\mathcal{R}_s}(\omega)\, d\omega + \sum_{t=s+1}^{n-1} \frac{1}{|t-s|} \\ &= \sum_{t=1}^{s} \frac{1}{t} + \int_\Omega S^s_{\mathcal{R}_s}(\omega)\, d\omega + \sum_{t=1}^{n-s-1} \frac{1}{t} \le 2H_n + 1 \ .\end{aligned}$$

Plugging the above inequality into Equation 4.2 yields $\mathbf{E}\left[hits^2\right] \le 2H_n + 1$. Combining this with Proposition 2 and Equation 4.1 we get $\mathbf{Pr}\left[\text{hunter catches rabbit}\right] \ge \frac{1}{2H_n+1}$ which yields the theorem. □

4.5.2 Hunter Strategies for General Graphs

In this section we extend the upper bound of the previous section to general graphs.

Theorem 22. *Let $G = (V, E)$ denote a graph and let $\mathrm{diam}(G)$ denote the diameter of this graph. Then there exists a hunter strategy on G that has expected escape length $O(|V| \cdot \log(\mathrm{diam}(G)))$.*

Proof. We *cover* the graph with $r = \Theta(n/d)$ cycles $C_1, \ldots, C_r$ of length d where $d = \Theta(\mathrm{diam}(G))$, that is, each node is contained in at least one of these cycles. (In order to obtain this covering, construct a tour of length $2n - 2$ along an arbitrary spanning tree, cut the tour into subpaths of length $d/2$ and then form a cycle of length d from each of these subpaths). From now on, if hunter or rabbit resides at a node of G corresponding to several cycle nodes, then we assume they *commit* to one of these virtual nodes and the hunter catches the rabbit only if they commit to the same node. This only slows down the hunter.

Now the hunter strategy is to choose one of the r cycles uniformly at random and simulate the RANDOMSPEED-strategy on this cycle. Call this a *phase*. We observe that each phase takes only $\Theta(d)$ rounds. The hunter executes phase after phase, each time choosing a new random cycle, until the rabbit is caught. In the following we will show that the success probability within each phase is $\Omega(d/nH_d)$, which implies the theorem.

Let us focus on a particular phase. For the purpose of analysis we assume that on every cycle the nodes are enumerated consecutively from 1 to d. Instead of directly calculating the probability that the hunter catches the rabbit we first analyze the probability that at some point of time both of them are on a node with the same number. Let X denote the indicator random variable for this event. We observe that the probability for $X = 1$ is identical to the probability that the hunter catches the rabbit on a cycle of length d. Consequently,

$$\mathbf{Pr}\left[X = 1\right] = \Omega(1/H_d) \ .$$

Now we use the fact that the hunter catches the rabbit if and only if they are on a node with the same number *and* they are on the same cycle. If during a phase hunter and rabbit are more than one time on a node with the same number, we consider only the first time. At this time the probability that hunter and rabbit are also on the same cycle is $\frac{1}{r}$. We obtain

$$\mathbf{Pr}\left[\text{hunter catches rabbit} \mid X = 1\right] \geq \frac{1}{r} \ .$$

We conclude

$$\mathbf{Pr}\left[\text{hunter catches rabbit}\right] \geq \mathbf{Pr}\left[\text{hunter catches rabbit} \mid X = 1\right] \cdot \mathbf{Pr}\left[X = 1\right] = \Omega(d/nH_d) \ .$$

□

4.6 Lower Bounds and Efficient Rabbit Strategies

We first prove that the hunter strategy for the cycle described in Section 4.5.1 is tight by giving an efficient rabbit strategy for the cycle. Then we provide lower bounds that match the upper bounds for general graphs given in Section 4.5.2.

4.6.1 An Optimal Rabbit Strategy for the Cycle

In this section we will prove a tight lower bound for any (mixed) hunter strategy on a cycle of length n. In particular, we describe a rabbit strategy such that every hunter needs $\Omega(n \log(n))$ expected time to catch the rabbit. We assume that the rabbit is unrestricted, i.e., can jump between arbitrary nodes, whereas the hunter is restricted to follow the edges of the cycle.

Theorem 23. *For the cycle of length n, there is a mixed, unrestricted rabbit strategy such that for every restricted hunter strategy the escape length is $\Omega(n \log(n))$.*

The rabbit strategy is based on a non-standard random walk. Observe that a standard random walk has the limitation that after n rounds, the rabbit is confined to a neighborhood of about $\sqrt{n}$ nodes around the starting position. Hence the rabbit is easily caught by a hunter that just sweeps across the ring (in one direction) in n steps. Also, the other extreme where the rabbit makes a jump to a node chosen uniformly at random in every round does not work, since in each round the rabbit is caught with probability exactly $1/n$, giving an escape length of $O(n)$. But the following strategy will prove to be good for the rabbit. The rabbit will change to a randomly chosen position every n rounds and then, for the next $n-1$ rounds, it performs a "heavy-tailed random walk". For this n-round strategy and an arbitrary n-round hunter strategy, we will show that the hunter catches the rabbit with probability $O(1/H_n)$. As a consequence, the expected escape length is $\Omega(n \log n)$, which gives the theorem.

A heavy-tailed random walk. We define a random walk on $\mathbb{Z}$ as follows. At time 0 a particle starts at position $X_0 = 0$. In a *step* $t \geq 1$, the particle makes a random jump $x_t \in \mathbb{Z}$ from position X_{t-1} to position $X_t = X_{t-1} + x_t$, where the jump length is determined by the following heavy-tailed probability distribution $\mathcal{P}$.

$$\mathbf{Pr}[x_t = k] \;=\; \mathbf{Pr}[x_t = -k] \;=\; \frac{1}{2(k+1)(k+2)}\ ,$$

for every $k \geq 1$ and $\mathbf{Pr}[x_t = 0] = \frac{1}{2}$. Observe that $\mathbf{Pr}[|x_t| \geq k] = (k+1)^{-1}$, for every $k \geq 0$. The following lemma gives a property of this random walk that will be crucial for the proof of our lower bound.

Lemma 20. *There is a constant $c_0 > 0$, such that, for every $t \geq 1$ and $\ell \in \{-t, \ldots, t\}$, $\mathbf{Pr}[X_t = \ell] \geq c_0/t$.*

Proof. We will prove the lemma using two claims. The first claim shows a simple monotonicity property of the random walk and the second claim shows that, with at least constant probability, the particle does not move more than distance $O(\tau)$ within τ steps. (Observe that this does not imply that the expected distance traveled in τ steps is $O(\tau)$. In fact, it is well-known that, under the heavy-tailed distribution $\mathcal{P}$, $\mathbf{E}[|x_t|] = \infty$ so that even the expected distance traveled in only one step is undefined.)

Claim 3 (monotonicity). *For every* $t \geq 0, \ell \geq 0, \mathbf{Pr}\left[X_t = \ell\right] \geq \mathbf{Pr}\left[X_t = \ell + 1\right]$.

Proof. We use induction on t. For $t = 0$ the claim is obviously true as $X_0 = 0$. Assume by inductive hypothesis that the claim holds for $t = r$. Define

$$P_i(j) \;=\; \mathbf{Pr}\left[X_r = i - j \wedge x_{r+1} = j\right] \;=\; \mathbf{Pr}\left[X_r = i - j\right]\mathbf{Pr}\left[x_{r+1} = j\right] \;\;,$$

for $i \geq 0, j \in \mathbb{Z}$. Then,

$$\mathbf{Pr}\left[X_{r+1} = \ell\right] \;\;=\;\; \sum_{j\in\mathbb{Z}} P_\ell(j) \;=\; \sum_{j\geq 0}[P_\ell(j) + P_\ell(-j-1)] \;\;,$$

and

$$\mathbf{Pr}\left[X_{r+1} = \ell + 1\right] \;\;=\;\; \sum_{j\in\mathbb{Z}} P_{\ell+1}(j) \;=\; \sum_{j\geq 0}[P_{\ell+1}(-j) + P_{\ell+1}(j+1)] \;\;.$$

As a consequence,

$$\mathbf{Pr}\left[X_{r+1} = \ell\right] - \mathbf{Pr}\left[X_{r+1} = \ell + 1\right] \;\;=\;\; \sum_{j\geq 0} Y(j),$$

where $Y(j) = P_\ell(j) + P_\ell(-j-1) - P_{\ell+1}(-j) - P_{\ell+1}(j+1)$. Since $\mathcal{P}$ is symmetric, $Y(j) = \big(\mathbf{Pr}\left[X_r = \ell - j\right] - \mathbf{Pr}\left[X_r = \ell + j + 1\right]\big)\big(\mathbf{Pr}\left[x_{r+1} = j\right] - \mathbf{Pr}\left[x_{r+1} = j + 1\right]\big)$. Observe that both factors are always positive by the induction hypothesis, symmetry, and some shifting. Hence, the claim is shown. □

Claim 4. *For every* $\tau \geq 0$ *and* $t \in \{0, \ldots, \tau\}$, $\mathbf{Pr}\left[|X_t| \leq \tau\right] \geq 1/2e^4$.

Proof. Observe that the variance of $\mathcal{P}$ is unbounded. Nevertheless, one can use the Chebyshev inequality for bounding the distance traveled by the particle as follows. Now we truncate the random variables x_i. For this purpose let us fix τ. For simplicity in notation, assume that τ is a multiple of four. For each random variable x_i, we define an auxiliary random variable y_i taking integer values in the range $[-\frac{\tau}{4}, \frac{\tau}{4}]$ such that $\mathbf{Pr}\left[y_i = k\right] \;=\; \mathbf{Pr}[x_i = k \;\mid\; |x_i| \leq \frac{\tau}{4}]$. We observe that

$$\mathbf{Pr}\left[y_i = k\right] \;=\; \frac{(\tau/4 + 2)}{(\tau/4 + 1)}\mathbf{Pr}\left[x_i = k\right] \;\;.$$

Since y_t is bounded, it has a finite variance, which can be estimated as follows:

$$\begin{aligned}\mathbf{Var}\left[y_t\right] &= \sum_{i=1}^{\tau/4} i^2 \,\mathbf{Pr}\left[|y_t| = i\right] \\ &= \frac{(\tau/4+2)}{(\tau/4+1)} \sum_{i=1}^{\tau/4} i^2 \,\frac{1}{(i+1)(i+2)} \;\leq\; \frac{\tau}{2}\;.\end{aligned}$$

Next define $Y_t = \sum_{i=1}^{t} y_i$. Since the random variables y_j are independent, $\mathbf{Var}\left[Y_t\right] = \sum_{j=1}^{t} \mathbf{Var}\left[y_t\right] \leq \frac{\tau^2}{2}$. Furthermore, we observe that $\mathbf{E}[Y_t] = 0$, for all $t \geq 0$. Hence applying the Chebyshev inequality (Lemma 7) gives

$$\mathbf{Pr}\left[|Y_t| \geq \tau\right] \;\leq\; \mathbf{Pr}\left[|Y_t| \geq \sqrt{2 \cdot \mathbf{Var}\left[Y_t\right]}\right] \;\leq\; \frac{1}{2}.$$

Finally, we apply this bound to the original random variables X_t and obtain

$$\begin{aligned}\mathbf{Pr}\left[|X_t| \leq \tau\right] &\geq \mathbf{Pr}\left[|X_t| \leq \tau \;\middle|\; \forall_{i=1}^{t} |x_i| \leq \frac{\tau}{4}\right] \mathbf{Pr}\left[\forall_{i=1}^{t} |x_i| \leq \frac{\tau}{4}\right] \\ &= \mathbf{Pr}\left[|Y_t| \leq \tau\right] \left(1 - \frac{1}{(\tau/4+2)}\right)^t \\ &\geq \frac{1}{2}\left(\frac{1}{e}\right)^4 \;.\end{aligned}$$

Thus Claim 4 is shown. □

Fix $t \geq 1$. We will use Claim 4 in order to show $\mathbf{Pr}\left[t \leq |X_t| < 4t\right] \geq c_0$, for a suitable constant c_0. Afterwards, we will apply Claim 3 to this bound and obtain the lemma.

The probability that there exists $k \in \{1, \ldots, t\}$ with $|x_k| > 2t$ and the first such k, say k^*, fulfills $|x_{k^*}| \leq 3t$ is at least

$$\left(1 - \left(1 - \frac{1}{2t+2}\right)^t\right)\left(1 - \frac{2t+2}{3t+2}\right) \;\geq\; \frac{1}{5}(1 - e^{-1/4})$$

since for every $m \in \mathbb{N}$ we have $\mathbf{Pr}\left[|x_k| \geq m\right] = (m+1)^{-1}$. Given this event, we observe that there exists $r \in \{1, \ldots, k^*\}$ with $|X_r| \in \{t, \ldots, 4t-1\}$. Let us denote by r^* the smallest such r. Applying Claim 4 gives $\mathbf{Pr}[|X_t - X_{r^*}| \leq t] \geq 1/2e^4$ since $X_t - X_{r^*}$ and X_{t-r^*} are identically distributed. It further follows by symmetry that $\frac{1}{2} \cdot \mathbf{Pr}[|X_t - X_{r^*}| \leq t] = \mathbf{Pr}[0 \leq X_t - X_{r^*} \leq t] = \mathbf{Pr}[-t \leq X_t - X_{r^*} \leq 0]$. Now observe that if $t \leq |X_{r^*}| < 4t$ then $t \leq |X_{r^*} - t| < 4t$ or $1 \leq |X_{r^*} + t| < 4t$. And so we obtain

$$\begin{aligned}&\mathbf{Pr}\left[t \leq |X_t| < 4t\right] \\ \geq\; &\frac{1}{2} \cdot \mathbf{Pr}\left[(|X_t - X_{r^*}| \leq t) \,|\, \exists r \leq t : (t \leq |X_r| < 4t)\right] \cdot \mathbf{Pr}\left[\exists r \leq t : (t \leq |X_r| < 4t)\right]\end{aligned}$$

$$\geq \quad \frac{1}{4e^4} \cdot \mathbf{Pr}\left[\exists r \leq t : (t \leq |X_r| < 4t)\right]$$

$$\geq \quad \frac{1}{20e^4}(1 - e^{-1/4}) \; .$$

If we now define

$$c_0 \stackrel{\text{def}}{=} \frac{1}{60e^4}(1 - e^{-1/4})$$

then applying Claim 3 gives $\mathbf{Pr}\left[|X_t| = t\right] \geq c_0/t$. Finally, applying the same claim again, we obtain $\mathbf{Pr}\left[|X_t| = \ell\right] \geq c_0/t$, for $0 \leq \ell \leq t$. This completes the proof of Lemma 20. □

The rabbit strategy. Our n-round rabbit strategy starts at a random position on the cycle. Starting from this position, for the next $n-1$ rounds, the rabbit simulates the heavy-tailed random walk in a wrap-around fashion on the cycle. The following lemma immediately implies Theorem 23.

Lemma 21. *The probability that the hunter catches the rabbit within n rounds is $O(1/H_n)$.*

Proof. Fix any n-round hunter strategy $\mathcal{H} = \mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_{n-1}$. Because of Proposition 1 we only need to estimate $\mathbf{E}[hits]$ and $\mathbf{E}[hits \mid hits \geq 1]$. First, we observe that $\mathbf{E}[hits] = 1$. This is because the rabbit chooses its starting position uniformly at random so that $\mathbf{Pr}[hit(t) = 1] = 1/n$ for $0 \leq t < n$, and hence $\mathbf{E}[hit(t)] = \mathbf{Pr}[hit(t) = 1] = 1/n$. By linearity of expectation, we obtain $\mathbf{E}[hits] = \sum_{t=0}^{n-1} \mathbf{E}[hit(t)] = 1$. Thus, it remains only to show that $\mathbf{E}\left[hits \mid hits \geq 1\right] \geq c_1 H_n$ for some constant c_1. In fact, the idea behind the following proof is that we have chosen the rabbit strategy in such a way that when the rabbit is hit by the hunter in a round then it is likely that it will be hit additionally in several later rounds as well.

Claim 5. *For every $\tau \in \{0, \ldots, \frac{n}{2} - 1\}$, $\mathbf{E}\left[hits \mid fhit(\tau) = 1\right] \geq c_1 H_n$, for a suitable constant c_1.*

Proof. Assume hunter and rabbit meet at time τ for the first time, i.e., $fhit(\tau) = 1$. Observe that the hunter has to stay somewhere in interval $[\mathcal{H}_\tau - (t-\tau), \mathcal{H}_\tau + (t-\tau)]$ in round $t > \tau$ as he is restricted to the cycle. The heavy-tailed random walk will also have some tendency to stay in this interval. In particular, Lemma 20 implies, for every $t > \tau$, $\mathbf{Pr}[hit(t)] \geq c_0/(t-\tau)$. Consequently, $\mathbf{E}[hits \mid fhit(\tau) \geq 1] \geq 1 + \sum_{t=\tau+1}^{n-1} c_0/(t-\tau)$, which is $\Omega(H_n)$ since $\tau < n/2$. □

With this result at hand, we can now estimate the expected number of repeated hits as follows.

$$\mathbf{E}\left[hits \mid hits \geq 1\right] \quad = \quad \sum_{\tau=0}^{n-1} \mathbf{E}\left[hits \mid fhit(\tau) = 1\right] \cdot \mathbf{Pr}\left[fhit(\tau) = 1 \mid hits \geq 1\right]$$

$$\geq \sum_{\tau=0}^{n/2-1} \mathbf{E}\left[hits \mid fhit(\tau)=1\right] \cdot \mathbf{Pr}\left[fhit(\tau)=1 \mid hits \geq 1\right]$$
$$\geq c_1' H_n \sum_{\tau=0}^{n/2-1} \mathbf{Pr}\left[fhit(\tau)=1 \mid hits \geq 1\right]$$

for some suitable constant $c_1' = 2c_1$. Finally, observe that

$$\sum_{\tau=0}^{n/2-1} \mathbf{Pr}\left[fhit(\tau)=1 \mid hits \geq 1\right] + \sum_{\tau=n/2}^{n-1} \mathbf{Pr}\left[fhit(\tau)=1 \mid hits \geq 1\right] = 1 \ .$$

Thus, one of the two sums must be greater than or equal to $\frac{1}{2}$. If the first sum is at least $\frac{1}{2}$, then we directly obtain $\mathbf{E}\left[hits \mid hits \geq 1\right] \geq c_1 H_n$. In the other case, one can prove the same lower bound by going backward instead of forward in time, that is, by summing over the last hits instead of the first hits. Hence Lemma 21 is shown. □

4.6.2 A Lower Bound In Terms of the Diameter

In this section, we show that the upper bound of Section 4.5.2 is asymptotically tight for the parameters n and $\mathrm{diam}(G)$. We will use the efficient rabbit strategy for cycles as a subroutine on graphs with arbitrary diameter.

Theorem 24. *For every positive integers n, d with $d < n$ there exists a graph G with n nodes and diameter d and a rabbit strategy such that for every hunter strategy on G the escape length is $\Omega(n \cdot \log(d))$.*

Proof. For simplicity, we assume that n is odd, $d = 4d'$ and $N = (n-1)/2$ is a multiple of d'. The graph G consists of a *center* $s \in V$ and N/d' subgraphs called loops. Each *loop* consists of a cycle of length $2d' + 1$ and a simple path of $d' + 1$ nodes such that the first node of the simple path is identified with one of the nodes on the cycle and the last node is identified with s. Thus, all loop subgraphs share the center s, otherwise the node sets are disjoint.

Every d' rounds the rabbit chooses uniformly at random one of the N/d loops and performs the optimal d'-round cycle strategy from Section 4.6.1 on the cycle of this loop graph. Observe that the hunter cannot visit nodes in different cycles during a phase of length d'. Hence, the probability that the rabbit chooses a cycle visited by the hunter is at most d'/N. Provided that the rabbit chooses the cycle visited by the hunter the probability that it is caught during the next d' rounds is $O(\frac{1}{H_{d'}})$ by Lemma 21. Consequently, the probability of being caught in one of the independent d'-round games is $O(\frac{d'}{nH_{d'}})$. Thus, the escape length is $\Omega(nH_{d'})$ which is $\Omega(n \cdot \log(d))$. □

4.7 Trees and Directed Graphs

In the previous sections, we have seen a restricted hunter strategy such that for every unrestricted rabbit strategy the expected escape length is $O(n \cdot \log(\mathrm{diam}(G))$. Furthermore, we have seen that this bound is optimal against unrestricted rabbits on cycles and several other networks of smaller diameter. This gives rise to the question whether for every hunter strategy there is a rabbit strategy such that the escape length is $\Omega(n \cdot \log(\mathrm{diam}(G)))$. We can answer this question negatively. In fact, in the following section we present a hunter strategy on a complete binary tree such that for every unrestricted rabbit strategy the expected escape length is $O(n)$.

Subsequently, in Section 4.7.2 we investigate the Hunter vs. Rabbit game on strongly connected directed graphs. We show that there exists a directed graph and a rabbit strategy such that every restricted hunter needs $\Omega(n^2)$ rounds to catch a restricted rabbit. Furthermore, for every strongly connected directed graph, there is a hunter strategy such that for every unrestricted rabbit strategy the expected escape length is $O(n^2)$.

4.7.1 A Linear Time Algorithm for Binary Trees

In this section, we investigate whether there exist graphs for which there is a hunter strategy against unrestricted rabbits with escape length $o(n \cdot \log(\mathrm{diam}(G)))$. The following theorem answers this question positively. It gives an example of an n-node network with diameter $\Theta(\log n)$ and escape length $O(n)$.

Theorem 25. *For the complete binary tree T of height h and $m = 2^h$ leaf nodes, there is a hunter strategy such that for every (unrestricted) rabbit strategy the expected escape length is $O(m)$.*

Proof. For simplicity, we assume that h is a power of 2. Furthermore, we initially assume that the rabbit visits only leaf nodes. (Finally, we will remove this assumption.)

We define the level of a node v of T as the height of the subtree T_v rooted at v. The hunter strategy is called *sparse random DFS* and is defined as follows. The hunter repeats the following four times (starting at the root of T): he chooses a node with height $h/2$ at random, visits it, and applies the same strategy recursively to the subtree T_v (with respect to its height). The recursion stops at subtrees of height 2, i.e., subtrees with 4 leaf nodes. Here for four times, the hunter chooses a leaf node uniformly at random and checks whether the rabbit hides on this leaf node.

The corresponding 4-ary recursion tree is called the *search tree* T_S. Let h_S denote the height of T_S and let L denote the number of leaf nodes of T_S. It is straightforward to see that $h_S = \log h = \log\log m$ and $L = 4^{h_S} = \log^2 m$. Observe that each leaf of T_S corresponds to a visited leaf of T. Furthermore, each edge of T_S corresponds to a path in T that the hunter has to follow in order to reach the root of the selected subtree on the next recursion level. Figure 4.2 shows a picture of the embedding of the recursion tree T_S into the tree T. Of course, the

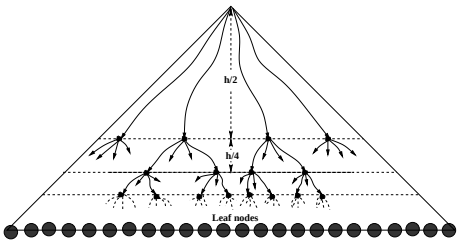


Fig. 4.2: Embedding of the recursion tree T_S into the tree T.

hunter needs some number of rounds in order to follow the paths that simulate the edges of T_S. We observe that the lengths of these paths decrease by a factor of two with every level of recursion. However, the number of edges in T_S per recursion level increases by a factor of four with each level. Hence, the leaf level of T_S dominates the execution time, which leads to the following observation.

Observation 8. *The hunter can perform the sparse random DFS in $O(L)$ rounds, where $L = \log^2 m$ is the number of visited leaf nodes of T.*

Next we investigate the probability that the hunter catches the rabbit within one pass of the described search algorithm.

Lemma 22. *The probability that sparse random DFS finds the rabbit is $\Omega(L/m)$.*

Proof. For $1 \leq i \leq L$, let r_i denote the round in which the ith leaf node is visited. Let $hit(i)$ denote a 0/1 random variable which is one iff the hunter hits the rabbit in round r_i, and $\mathit{fhit}(i) = 1$ if this is the first hit. Clearly, for every i, $\mathbf{E}[hit(i)] = \frac{1}{m}$. Using linearity of expectation, we obtain $\mathbf{E}[hits] = L/m$. Now applying Proposition 1 yields that the lemma can be shown by proving $\mathbf{E}[hits \mid hits \geq 1] = O(1)$. As $\mathbf{E}[hits \mid hits \geq 1] \leq \max_{1 \leq i \leq L} \{\mathbf{E}[hits \mid \mathit{fhit}(i) = 1]\}$, we only need to show $\mathbf{E}[hits \mid \mathit{fhit}(i) = 1] = O(1)$, for $1 \leq i \leq L$.

Fix an arbitrary $i \in \{1, \ldots, L\}$. We assume that $\mathit{fhit}(i) = 1$, that is, the hunter meets the rabbit at leaf i of the search tree T_S and this is the first hit. Let for a level $\ell \in \{1, \ldots, h_S\}$ of the search tree, $T_S(\ell)$ denote the complete 4-ary subtree of height ℓ that contains i. If the mapping of i to a leaf of T is fixed then so is the mapping of the root nodes of the subtrees $T_S(\ell)$, $\ell \in \{1, \ldots, h_S\}$. This partially determines the search tree T_S and hence the leaf nodes visited in addition to i later in the search. We show that the search tree still contains "enough" randomness such that $\mathbf{E}[hits \mid \mathit{fhit}(i) = 1]$ is not too large.

Consider a fixed subtree $T_S(\ell)$ for some value $\ell \in \{1, \ldots, h_S\}$. Let $v(\ell)$ denote the root of $T_S(\ell)$ and let $w(\ell)$ denote the corresponding node in T according to the embedding of T_S in T. We first bound the expected number of hits made by the hunter during the search on the subtree $T_S(\ell)$ *not* including the hits made in $T_S(\ell - 1)$. During this part of the search $3 \cdot 4^{\ell-1}$ leaf nodes of T are visited. These leaf nodes are all contained in the subtree of T rooted at

$w(\ell)$. Altogether, this subtree contains 2^{2^ℓ} leaf nodes since $w(\ell)$ is on level 2^ℓ in T. As each of these nodes is visited with equal probability the expected number of hits is at most $\frac{3 \cdot 4^{\ell-1}}{2^{2^\ell}}$.

We get an upper bound on $\mathbf{E}[hits \mid fhit(i) = 1]$ by summing this value for all subtrees $T_S(1), \ldots, T_S(h_S)$. Hence,

$$\mathbf{E}[hits \mid fhit(i) = 1] \;\leq\; 1 + \sum_{\ell=1}^{h_S} \frac{3 \cdot 4^{\ell-1}}{2^{2^\ell}} \;\leq\; 3 \;.$$

Hence, Lemma 22 is shown. □

Combining Observation 8 and Lemma 22 we conclude that the escape length is $O(n)$. Finally, it remains to show how to deal with rabbit strategies that hide on internal nodes of T. To solve this problem we define a *virtual tree* T' which is a complete binary tree of height $h+1$. We embed T' into T such that every node in T hosts at least one leaf of T' and adjacent nodes in T' are hosted by adjacent nodes in T. (The latter requirement means that the *dilation* of the embedding is one.) Then the hunter simulates the random DFS for T' on T. In this way the rabbit cannot avoid the leaves of T' and Theorem 25 follows.

It remains to describe the embedding of T' into T. Let T'_1 and T'_2 denote the two disjoint subtrees of height h of T'. We map every node of T'_1 to its counterpart in the isomorphic tree T. Additionally, we map the root of T' to the root of T. If T does not consist of a single node we apply the same rule recursively with trees T'_2 and T^*, where T^* denotes the subtree of T induced by its internal nodes, i.e., the subtree obtained by removing all leaf nodes from T. In this way, every node of T receives at least one leaf node of T' (and possibly several other internal nodes). □

4.7.2 Directed Graphs

Now we want to consider the Hunter vs. Rabbit game on directed graphs. We slightly alter the definition of restricted strategy for this purpose. In a directed graph $G = (V, E)$ we call a pure strategy $\mathcal{S}$ *restricted*, if either $[\mathcal{S}_t, \mathcal{S}_{t+1}\rangle \in E$ or $\mathcal{S}_t = \mathcal{S}_{t+1}$ holds for every $t \in \mathbb{N}_0$.

Theorem 26. *Let G denote an arbitrary directed strongly connected graph with n nodes. Then there is a restricted hunter strategy on G such that for every unrestricted rabbit strategy the expected escape length is $O(n^2)$. Furthermore, there is a directed graph with n nodes, where there exists a restricted rabbit strategy such that for every restricted hunter strategy the expected escape length is $\Omega(n^2)$.*

Proof. The hunter strategy is defined as follows. In every n rounds, the hunter goes to a node in the graph chosen uniformly at random (this is possible in n steps because the graph is strongly connected) and the hunter meets the rabbit with probability $\Omega(1/n)$. This proves the claim.

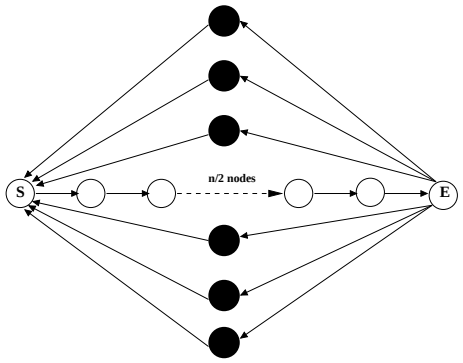


Fig. 4.3: A good graph for the rabbit

We now want to construct a graph and a rabbit strategy such that for every restricted hunter strategy the expected escape length is $\Omega(n^2)$. The graph has a directed path of $n/2$ nodes starting with node S and ending with node E. For each of the remaining nodes (let us call them black nodes), there is an arc from E and an arc to S. Our construction is illustrated in Figure 4.3. The rabbit initially chooses one of the black nodes at random and stays there forever. Now, it is easy to see that, if the hunter fails to find the rabbit in a black node, he has to spend $n/2$ rounds to check another black node. This shows a lower bound of $\Omega(n^2)$ even against a stationary rabbit. Hence the theorem is shown. □

4.8 Summary and Open Problems

In this chapter we studied a pursuit-evasion game called the Rabbit vs. Hunter game. We considered the stronger adversarial model where the rabbit is allowed to jump arbitrarily from one node to the other whereas the hunter can only travel along the edges of the graph. We saw efficient randomized hunter strategies that were based on random speed approach. We also saw good rabbit strategies based on non-standard random walks, which show that the hunter strategy is optimal on certain graphs. We then showed even better hunter strategies on specific graphs (using a different approach) and also discussed the case of directed graphs.

Our results lead to several interesting open problems. One natural question to ask is, are there better hunter strategies against a restricted adversary. That is, are there hunter strategies with $o(n \log(\mathrm{diam}))$ escape length on general graphs if both hunter and rabbit can only move along the edges of the graph. For example, on a cycle the trivial hunter strategy with $O(n)$ escape length is to start from node 0 and sweep the cycle either clockwise or counter-clockwise with equal probability. Interestingly even the following Markovian hunter strategy (the random decisions made in a round do not depend on the past like for e.g., the sweeping direction

that was chosen in the beginning) works well on a cycle. On each node we assign transition probabilities as follows. On any node $v \in \{1, \ldots, n-1\}$, the probability to jump to the successor is 1 (i.e., probability to jump backwards or a self-loop is 0). In node 0, the probability to jump to node 1 is $\frac{1}{n}$ and $1 - \frac{1}{n}$ for the self-loop. The hunter starts from a random node and follows these edge probabilities. If the rabbit ever crosses node 0 after n steps then the hunter is waiting at node 0 with constant probability. If the rabbit crosses node 0 before n steps (and never after n steps) then with constant probability the hunter will go past the rabbit in $4n$ steps. This is because between n and $3n$ steps, the hunter starts again from node 1 with constant probability that will then sweep the cycle in another n steps and meet the rabbit in between.

Another open problem is to study how much randomness does the hunter need. In other words, it would be interesting to investigate the connection between the number of random bits available to the hunter to the escape length achievable. Our hunter strategies need the complete topological information about the underlying graph. It is also interesting to investigate hunter/rabbit strategies if only limited topological information is available and the hunter is provided with some pebbles to be placed in the graph nodes, thereby avoiding routes already taken. In our setting, both hunter and rabbit have zero-visibility (they see each other only if they are in the same node). A different line of research is to study hunter (possibly multiple) and rabbit strategies with more visibility (in case of the restricted rabbit setting). Recently Isler et al. [IKK04] showed that for limited visibility, i.e., both players can see the immediate neighborhood, two hunters suffice to catch the rabbit in general graphs. They also give polynomial time strategies and also characterize graphs where only one hunter is enough.

Bibliography

[ABC+99] F. Afrati, E. Bampis, C. Chekuri, D. Karger, C. Kenyon, S. Khanna, I. Milis, M. Queyranne, M. Skutella, C. Stein, and M. Sviridenko. Approximation schemes for minimizing average weighted completion time with release dates. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 32–43, New York City, NY, 1999.

[AE98] Y. Azar and L. Epstein. On-line machine covering. *Journal of Algorithms*, 1:67–77, 1998.

[AF84] M. Aigner and M. Fromme. A game of cops and robbers. *Discrete Applied Math*, 8:1–12, 1984.

[AGZ99] M. Andrews, M.X. Goemans, and L. Zhang. Improved bounds for on-line load balancing. *Algorithmica*, 23:278–301, 1999.

[AKL+79] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovász, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 218–223, 1979.

[Alb93] S. Albers. The influence of lookahead in competitive paging algorithms (extended abstract). In *European Symposium on Algorithms*, pages 1–12, 1993.

[Alb98] S. Albers. A competitive analysis of the list update problem with lookahead. *Theoretical Computer Science*, 197(1–2):95–109, 1998.

[Alb99] S. Albers. Better bounds for online scheduling. *SIAM Journal on Computing*, 29:459–473, 1999.

[Alb03] S. Albers. Online algorithms: a survey. *Mathematical Programming*, 97:3–26, 2003.

[Alp74] S. Alpern. The search game with mobile hider on the circle. *Differential Games and Control Theory*, pages 181–200, 1974.

[ARS+02] M. Adler, H. Räcke, N. Sivadasan, C. Sohler, and B. Vöcking. Randomized pursuit-evasion in graphs. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 366–376, 2002.

[ARS+03] M. Adler, H. Räcke, N. Sivadasan, C. Sohler, and B. Vöcking. Randomized pursuit-evasion in graphs. *Combinatorics, Probability and Computing*, 12(3):225–244, 2003.

[BDBK+94] S. Ben-David, A. Borodin, R. M. Karp, G. Tardos, and A. Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11(1):2–14, 1994.

[BEY98] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[BFKV95] Y. Bartal, A. Fiat, H. Karloff, and R. Vohra. New algorithms for an ancient scheduling problem. *Journal of Computer and System Sciences*, 51:359–366, 1995.

[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.

[BK87] T. Basar and P. R. Kumar. On worst case design strategies. *Computers and Mathematics with Applications: Special Issue on Pursuit-Evasion Differential Games*, 13(1-3):239–245, 1987.

[BLMS+03] 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.

[BLS92] A. Borodin, N. Linial, and M. Saks. An optimal online algorithm for metrical task systems. *Journal of the ACM*, 39:745–763, 1992.

[BSS02] A. Brinkmann, K. Salzwedel, and C. Scheideler. Compact, adaptive placement schemes for non-uniform requirements. In *14th ACM Symposium on Parallel Algorithms and Architectures*, pages 53–62, 2002.

[BW00] G. Brightwell and P. Winkler. Gibbs measures and dismantlable graphs. *J. Comb. Theory (Series B)*, 78, 2000.

[CGJ78] E. G. Coffman Jr., M. R. Garey, and D. S. Johnson. An application of bin-packing to multiprocessor scheduling. *SIAM Journal on Computing*, 7:1–17, 1978.

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.

[CNP+01] I. Chatzigiannakis, S. Nikoletseas, N. Paspallis, P. Spirakis, and C. Zaroliagis. An experimental study of basic communication protocols in ad-hoc mobile networks. In *Proceedings of the 5thWorkshop on Algorithmic Engineering*, pages 159–171, 2001.

[CNS01] I. Chatzigiannakis, S. Nikoletseas, and P. Spirakis. An efficient communication strategy for ad-hoc mobile networks. In *Proceedings of the 20th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 320–322, 2001.

[CvVW94] B. Chen, A. van Vliet, and G. J. Woeginger. Lower bounds for randomized online scheduling. *Information Processing Letters*, 51:219–222, 1994.

[EGHP+00] A. Efrat, L. J. Guibas, S. Har-Peled, D. C. Lin, J. S. B. Mitchell, and T. M. Murali. Sweeping simple polygons with a chain of guards. In *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 927–936, 2000.

[EST94] J. A. Ellis, I. H. Sudborough, and J. S. Turner. The vertex seperation and search number of a graph. *Information and Computation*, 113(1):50–79, 1994.

[FN01] S. Fitzpatrick and R. Nowakowski. Copnumber of graphs with strong isometric dimension two. *Ars Combinatorica*, 59:65–73, 2001.

[Fri84] D. K. Friesen. Tighter bounds for the multifit processor scheduling algorithm. *SIAM Journal on Computing*, 13:170–181, 1984.

[FW00] R. Fleischer and M. Wahl. Online scheduling revisited. *Journal of Scheduling*, 3:343–353, 2000.

[Gal79] S. Gal. Search games with mobile and immobile hider. *SIAM J. Control Optim.*, 17:99–122, 1979.

[Gal80] S. Gal. *Search Games*. Mathematics in Science and Engineering, 149. Academic Press, Inc. [Harcourt Brace Jovanovich, Publishers], New York-London, 1980.

[GJ78] M. R. Garey and D. S. Johnson. Strong np-completeness results: Motivation, examples and implications. *Journal of the ACM*, 25:499–508, 1978.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP–Completeness*. Freeman, San Francisco, 1979.

[GLL+99] L. J. Guibas, Jean-Claude Latombe, S. M. LaValle, D. Lin, and R. Motwani. A visibility-based pursuit-evasion problem. *International Journal of Computational Geometry and Applications (IJCGA)*, 9(4):471–493, 1999.

[GLLR79] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[Gra66] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.

[Gra69] R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17:263–269, 1969.

[Hoc96a] D. S. Hochbaum, editor. *Approximation Algorithms for NP-Hard Problems*. Thomson, 1996.

[Hoc96b] D. S. Hochbaum. Various notions of approximation: Good, better, best, and more. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, chapter 9, pages 346–398. Thomson, 1996.

[HR90] T. Hagerup and C. Rüb. A guided tour of chernoff bounds. *Information Processing Letters*, 33:305–308, 1990.

[HS87] D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: Theoretical and practical results. *Journal of the ACM*, 34:144–162, 1987.

[IKK04] V. Isler, S. Kannan, and S. Khanna. Randomized pursuit-evasion with limited visibility. In *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1053–1062, 2004.

[Isa65] R. Isaacs. *Differential games. A mathematical theory with applications to warfare and pursuit, control and optimization.* John Wiley & Sons, Inc., New York-London-Sydney, 1965.

[KMRS88] A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator. Competitive snoopy caching. *Algorithmica*, 3(1):79–119, 1988.

[KP86] L. M. Kirousis and C. H. Papadimitriou. Searching and pebbling. *Theoretical Computer Science*, 47:205–218, 1986.

[KP94] E. Koutsoupias and C. Papadimitriou. Beyond competitive analysis. In *Proceedings of the Twenty-Fifth Symposium on Foundations of Computer Science*, pages 394–400, 1994.

[KPT96] D. R. Karger, S. J. Phillips, and E. Torng. A better algorithm for an ancient scheduling problem. *Journal of Algorithms*, 20:400–430, 1996.

[Lan81] M. A. Langston. *Processor scheduling with improved heuristic algorithms*. PhD thesis, Texas A&M University, 1981.

[LaP93] A. S. LaPaugh. Recontamination does not help to search a graph. *Journal of the ACM*, 40(2):224–245, 1993.

[Len83] H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

[MHG+88] N. Megiddo, S. Louis Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou. The complexity of searching a graph. *Journal of the ACM*, 35(1):18–44, 1988.

[MMS88] M. S. Manasse, L. A. McGeoch, and D. D. Sleator. Competitive algorithms for on-line problems. In *ACM Symposium on Theory of Computing*, pages 322–333, 1988.

[MR95] R. Motwani and P. Raghavan. *Randomized algorithms*. Cambridge University Press, 1 edition, 1995.

[NN98] S. Neufeld and R. Nowakowski. A game of cops and robbers played on products of graphs. *Discrete Mathematics*, 186:253–268, 1998.

[NW83] R. Nowakawski and P. Winkler. Vertex-to-vertex pursuit in a graph. *Discrete Math*, 43:235–239, 1983.

[NW88] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, New York, 1988.

[Pap94] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[Par76] T. D. Parsons. Pursuit-evasion in a graph. In Y. Alavi and D.R. Lick, editors, *Theory and Applications of Graphs*, Lecture Notes in Mathematics, pages 426–441. Springer, 1976.

[Par78] T. D. Parsons. The search number of a connected graph. In *Proceedings of the 9th Southeastern Conference on Combinatorics, Graph Theory and Computing*, pages 549–554, 1978.

[PGK88] D. Patterson, G. Gibson, and R. Katz. A case for redundant arrays of inexpensive disks (RAID). *Proceedings of ACM SIGMOD'88*, pages 109–116, 1988.

[PLC01] Sang-Min Park, Jae-Ha Lee, and Kyung-Yong Chwa. Visibility-based pursuit-evasion in a polgonal region by a searcher. In *Proceedings of the 28th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 456–468, 2001.

[Rud01] J. F. Rudin III. *Improved bounds for the on-line scheduling problem*. PhD thesis, The University of Texas at Dallas, 2001.

[Sah76] S. Sahni. Algorithms for scheduling independent tasks. *Journal of the ACM*, 23:116–127, 1976.

[San04] P. Sanders. Algorithms for scalable storage servers. In *SOFSEM 2004: Theory and Practice of Computer Science*, volume 2932, pages 82–101, 2004.

[Sch86] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Chichester, 1986.

[Sga97] J. Sgall. A lower bound for randomized on-line multiprocessor scheduling. *Information Processing Letters*, 63(1):51–55, 14 July 1997.

[Sga98] J. Sgall. On-line scheduling — a survey. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms: The State of the Art*, volume 1442 of *Lecture Notes in Computer Science*, pages 196–231. Springer, Berlin, 1998.

[SS04] G. Schäfer and N. Sivadasan. Topology matters: Smoothed competitiveness of metrical task systems. In *Proceedings of the 21st Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 489–500, 2004.

[SSS04] P. Sanders, N. Sivadasan, and M. Skutella. Online scheduling with bounded migration. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP)*, pages 1111–1122, 2004.

[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 *ACM Symposium on Theory of Computing*, 2001.

[SY92] I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM Journal on Computing*, 21(5):863–888, 1992.

[Wes00] J. Westbrook. Load balancing for response time. *J. Algorithms*, 35(1):1–16, 2000.

[Zel72] M. I. Zelikin. A certain differential game with incomplete information (russian). *Dokl. Akad. Nauk*, 202:998–1000, 1972.

Definitions of Some Basic Concepts

Random Walks on Graphs. Consider a connected undirected graph $G = (V, E)$. A random walk on G is defined as follows. Graph G induces the following Markov chain M_G, the states of M_G being the nodes of G. Define transition probabilities for every $(u, v) \in V \times V$ as

$$P_{uv} = \begin{cases} \frac{1}{d(u)} & \text{if } (u, v) \in E \\ 0 & \text{otherwise;} \end{cases}$$

where $d(u)$ is the degree of node u.

Hitting time. Also denoted as h_{uv}, is the expected number of steps in a random walk that starts at u and ends upon first reaching v.

Commute time. Defined as C_{uv} for a pair of nodes u and v, is the expected time for a random walk starting at u to return to u after at least one visit to v.

Cover time. Let $\mathcal{C}_u$ denote the expected length of a walk that starts at u and ends after visiting every node in G at least once. Then the cover time C for G is $\max_u \mathcal{C}_u$.

Graph Isomorphism. Graphs G_1 and G_2 are *isomorphic* if there is a one-to-one correspondence between the vertices of G_1 and G_2 with the property that two vertices are adjacent in G_1 *iff* their images in G_2 are adjacent.

The l_p-Norm. For $x \in \mathbb{R}^n$, the l_p-norm of x, denoted by $||x||_p$, is defined as

$$||x||_p = \Big(\sum_{i=1}^{n} |x_i|^p \Big)^{1/p} \quad .$$

The l_2-norm is the *Euclidean norm*. The l_∞-norm is simply $\max\{|x_1|, \ldots, |x_n|\}$.

Classification Scheme for Scheduling. For describing a scheduling problem, the following terminology, introduced by Graham, Lawler, Lenstra and Rinnooy Kan [GLLR79], is

usually used. Problems are described by three fields: the first represents the machine environment, the second denotes any special conditions or constraints in the model, and the third is the objective function. Possible machine environments are $1, \mathrm{P}, R$: environment consisting of a single machine, *identical parallel* machines, and *unrelated* machines. For identical parallel machines the processing time for a job j is the same, denoted as p_j. For unrelated machines, processing time for a job can be different in each machine, p_{ij} denoting the processing time of job j in machine i. For example, "$\mathrm{P}|p_j = 1|C_{\max}$" is the problem of scheduling unit-time jobs on identical parallel machines to minimize the makespan.

Linear and Integer Linear Programs. A *linear program*, LP for short, is a problem of minimizing or maximizing a linear function subject to a finite set of linear constraints. An LP can be written as

$$\max\{cx \ : \ Ax \leq b\},$$

where $A \in \mathbb{R}^{m \times n}$ is a matrix of real numbers, with m rows and n columns, and $b \in \mathbb{R}^m$ is a m-dimensional column vector. The above LP has n variables $x_1, \ldots, x_n$, denoted by the column vector x. In the above LP, cx is called an *objective* function, where $c \in \mathbb{R}^n$ is an n-dimensional row vector. A *feasible* solution for the given LP is an assignment to $x_1, \ldots, x_n$, such that the set of constraints, i.e., $Ax \leq b$, is satisfied. A solution to the LP is a feasible solution that maximizes cx.

An *integer linear program*, ILP for short, is a linear program with the additional constraint that the variables can only take integer values. That is, given a rational matrix A, and rational vectors b and c, determine $\max\{cx \ : \ Ax \leq b; x \text{ integral}\}$. Many combinatorial optimization problems can be formulated as ILPs.

Approximation Algorithms. Usually for problems that are computationally hard, there is no polynomial time algorithm to solve it unless P=NP, polynomial time approximation algorithms are designed. An *α-approximation* algorithm is a polynomial time algorithm that computes a feasible solution whose value is always within a factor α of the optimum.

Amortized Analysis and Potential Method. In an *amortized analysis*, the cost involved for a sequence of operations is averaged over all the steps. Even though in the worst-case, the cost for one step may be very high, amortized analysis guarantees *the average cost of each step in the worst-case*.

The *potential method* of amortized analysis represents the pre-paid work/cost as a *potential energy* or just "potential", that can be released to pay for future operations. A potential is defined using a potential function $\Phi(t)$ for round/operation t, with $\Phi(0)$ as the initial potential. The amortized cost $\mathcal{C}_a(t)$ for round t is defined as

$$\mathcal{C}_a(t) = C(t) + \Phi(t) - \Phi(t-1),$$

where $C(t)$ is the actual cost for operation t. Hence the total amortized cost for n operations is

$$\sum_{t=1}^{n} \mathcal{C}_a(i) \;=\; \sum_{t=1}^{n} \big(C(t) + \Phi(t) - \Phi(t-1)\big) \;=\; \sum_{t=1}^{n} C(t) + \Phi(n) - \Phi(0).$$

If we ensure that $\Phi(n) - \Phi(0)$ is non-negative then the total amortized cost is an upper bound on the total cost.

Curriculum Vitae

EDUCATION

Jul 2000 – Jul 2004	Max-Planck-Institut für Informatik, Saarbrücken, Germany. Ph. D. student in the Algorithms and Complexity Group. Advisor: Prof. Dr. Kurt Mehlhorn
Jul 1997 – Jan 1999	Indian Institute Of Science, Dept. of Computer Science, Bangalore, India. Master of Engineering in Computer Science. Master's thesis: *3D Surface Reconstruction.* Advisor: Prof. Dr. Ramesh Hariharan
Jul 1992 – Jul 1996	Regional Engineening College (NIT), Calicut, India. Bachelor of Engineering in Computer Science & Engg.

SCHOLARSHIPS

Max-Planck Society's scholarship for Ph.D students affiliated to the *International Max-Planck Research School*, Max-Planck-Institut für Informatik, Saarbrücken, Germany.

JOURNAL PUBLICATIONS

[1] Randomized Pursuit-Evasion in Graphs. M. Adler, H. Räcke, N. Sivadasan, C. Sohler, B. Vöcking, *Combinatorics, Probability and Computing*, 12(3) page 225–244, 2003.

[2] All-pairs shortest-paths computation in the presence of negative cycles. K. Mehlhorn, V. Priebe, G. Schäfer, N. Sivadasan, *Information Processing Letters*, 81(6) page 341-343, 2002.

CONFERENCE PROCEEDINGS

[1] Online Scheduling with Bounded Migration. P. Sanders, N. Sivadasan, M. Skutella. *Proceedings of the 31st International Colloquium on Automata, Languages, and Programming (ICALP)*, LNCS 3142, pp.1111–1122, 2004. (Invited to special issue of TCS).

[2] Topology Matters: Smoothed Competitiveness of Metrical Task Systems. G. Schäfer, N. Sivadasan, *Proceedings of the 21st International Symposium on Theoretical Aspects of Computer Science (STACS)*, LNCS 2996, pp. 489–500, 2004.

[3] Randomized Pursuit-Evasion in Graphs. M. Adler, H. Räcke, N. Sivadasan, C. Sohler, B. Vöcking, *Proceedings of the 29th International Colloquium on Automata, Languages, and Programming (ICALP)*, LNCS 2380, pp. 366–376, 2002.

[4] Energy Optimal Routing in Radio Networks Using Geometric Data Structures. R. Beier, P. Sanders, N. Sivadasan, *Proceedings of the 29th International Colloquium on Automata, Languages, and Programming (ICALP)*, LNCS 2380, pp. 910–912, 2002.