Diss. ETH No. 20164

Advice and Randomization in Online Computation

A dissertation submitted to
ETH Zurich

for the degree of
Doctor of Sciences

presented by
Dennis Komm
Dipl.-Inform., RWTH Aachen University
born on May 2, 1982
citizen of Germany

accepted on the recommendation of
Prof. Dr. Juraj Hromkovič, examiner
Prof. Dr. Peter Widmayer, co-examiner
Prof. Dr. Georg Schnitger, co-examiner

2012

Abstract

Online computation is both of theoretical interest and practical relevance as numerous computational problems require a model in which algorithms do not know the whole input at every time step during runtime. The established measurement for the output quality of these *online algorithms* is the so-called *competitive analysis*, introduced by Sleator and Tarjan in 1985. Similar to the decrease in accuracy we have to accept when efficiently (i. e., in polynomial time) solving $\mathcal{NP}$-hard problems, the *competitive ratio* describes what we have to pay for not knowing the future.

In this thesis, we want to measure how much additional information is both necessary and sufficient to escape from this dilemma, i. e., we want to understand what causes this, for the majority of problems, vast amount of precision we lose due to facing an online scenario. More specifically, we want to study the *advice complexity* of online problems, which describes the amount of information online algorithms lack, causing them to fail (compared to hypothetical offline algorithms that know all yet unrevealed parts of the input from the start). In the model used throughout this thesis, we equip online algorithms with an additional *advice tape* onto which an *oracle*, which sees the whole input before the algorithm is executed, may write binary information. The algorithm can then use these *advice bits* during computation. We call the minimum number of advice bits needed to compute an optimal solution for some online problem the *information content* of this problem. This information is what needs to be extracted from the instance in order to overcome the drawback of not completely knowing it in advance. We know that there exist well-studied online problems for which any solution computed by a deterministic online algorithm is (asymptotically) half as good as the optimal solution, because it does not know future input parts. However, a single bit of advice suffices to perform optimally. For many other problems, measuring the information content proves to be a more complicated task. Moreover, generalizing this idea, we study the tradeoff between obtaining high-quality results (i. e., creating online algorithms with a reasonable competitive ratio) and the number of advice bits both necessary and sufficient for this.

We study five online problems within the framework described above, the *job shop scheduling problem* with two jobs and unit-length tasks, the *disjoint path allocation problem*, the *k-server problem*, the *set cover problem*, and the *knapsack problem*. It turns out that online problems may behave very differently in terms of advice complexity. There are, for instance, problems that achieve very good results with a constant number of advice bits and other ones that, if given less advice than linear in the input size, are doomed to fail. Specifically, we ask how many advice bits are necessary and sufficient to (i) be optimal, (ii) to improve over purely deterministic strategies, or (iii) to be on par with (or better than) randomized strategies.

Since we may look at computing with advice as supplying the best possible random string for any input, we are particularly interested in the last point and the further rela-

tion between advice and randomization. Obviously, lower bounds on the advice complexity carry over to randomization and upper bounds on randomization carry over to computing with advice. For some particular problems, we further show how to construct *barely random* algorithms (i. e., randomized algorithms that only use a constant number of random bits) from algorithms with advice. Also, we introduce an online problem for which we study the combination of advice and randomization, showing how every additional advice bit significantly improves the output quality.

Zusammenfassung

Online-Berechnungen sind sowohl von theoretischem Interesse als auch von grossem Nutzen für die Praxis, da viele Berechnungsprobleme eine Analyse erfordern, bei der die gesamte Eingabe dem entsprechenden Algorithmus zur Laufzeit nicht bekannt ist. Um die Güte von diesen *Online-Algorithmen* zu messen, wird meist die so genannte *Competitive Analysis* verwendet. So wie die Approximationsgüte von Algorithmen für $\mathcal{NP}$-schwere Probleme angibt, was wir zahlen, um ein Problem effizient (in polynomieller Laufzeit) zu lösen, zeigt die *Competitive Ratio* auf, welchen Preis wir dafür zahlen, die Zukunft nicht zu kennen.

In dieser Arbeit beschäftigen wir uns mit der Frage, *was* genau dieses Dilemma verursacht, was also der Grund dafür ist, dass wir für eine grosse Anzahl der bekannten Probleme in Kauf nehmen müssen, in Online-Szenarien Ergebnisse sehr schwacher Qualität zu erhalten. Zu diesem Zweck untersuchen wir die *Advice-Komplexität* von Online-Problemen, das heisst die Menge an Informationen, die Online-Algorithmen (verglichen mit hypothetischen Offline-Algorithmen, die die gesamte Eingabe von Anfang an kennen) für die entsprechenden Probleme fehlt, um bessere Ergebnisse zu erzielen. In dem Modell, das wir hier betrachten, verfügen die Online-Algorithmen über ein *Advice-Band*, auf welchem ein *Orakel*, dem die gesamte Eingabe bekannt ist, binäre Informationen über diese schreiben kann. Diese *Advice-Bits* können dann vom jeweiligen Algorithmus während der Laufzeit benutzt werden. Die Anzahl dieser Bits, die notwendig sind, um ein optimales Ergebnis zu berechnen, nennen wir den *Informationsgehalt* des entsprechenden Problems, denn eben genau diese Information über die jeweilige Eingabe muss bekannt sein, um den Nachteil zu überwinden, mit dem Online-Algorithmen konfrontiert werden, weil sie nicht die gesamte Eingabe kennen. Wir kennen Online-Probleme, für die deterministische Algorithmen Lösungen berechnen, die (asymptotisch) doppelt so schlecht sind wie die optimale Lösung. Allerdings reicht bereits ein einziges Bit an Information, um ein optimales Ergebnis zu erzielen. Für viele andere Probleme ist die Analyse jedoch weitaus komplizierter. Ferner interessiert uns eine Verallgemeinerung obiger Frage, nämlich, wie sich der Tradeoff zwischen der sowohl benötigten als auch hinreichenden Anzahl an Advice-Bits und einer zu erreichenden Competitive Ratio verhält.

Wir untersuchen hier fünf Online-Probleme in diesem Modell, das *Job-Shop-Scheduling-Problem* mit zwei Jobs und Aufgaben mit Einheitslänge, das *Disjoint-Path-Allocation-Problem*, das *k-Server*-Problem, das *Set-Cover-Problem* und das *Rucksack-Problem*. Es zeigt sich, dass sich diese Online-Probleme zum Teil völlig unterschiedlich verhalten. Beispielsweise existieren Probleme, die mit einer konstanten Anzahl von Advice-Bits sehr gute Resultate ermöglichen und solche, die weiterhin nur schlechte Ergebnisse zulassen, solange der entsprechende Algorithmus weniger Bits erhält als eine in der Eingabelänge lineare Anzahl. Im Wesentlichen interessiert uns, wie viele Advice-Bits nötig und ausreichend sind, um (i) optimal zu sein, (ii) besser zu sein als deterministische Strategien oder (iii) genau so gut (oder sogar besser) zu sein als randomisierte Algorithmen.

Da wir den Advice als die jeweils beste zufällige Wahl für jede Eingabe betrachten können, ist der letzte Punkt von besonderem Interesse. Es ist offensichtlich, dass untere Schranken für die Advice-Komplexität sich auf randomisierte Verfahren übertragen und umgekehrt, dass obere Schranken für letztere Strategien ebenfalls gültig für Algorithmen mit Advice sind. Für einige konkrete Probleme zeigen wir ferner, wie aus Advice-Algorithmen *Barely-Random*-Algorithmen (also randomisierte Algorithmen, die nur eine konstante Anzahl von Zufallsbits verwenden) konstruiert werden können. Desweiteren stellen wir ein Problem vor, für das wir eine Kombination aus zufallsgesteuerter Berechnung *und* Advice entwerfen, wobei jedes weitere Advice-Bit, das der Algorithmus erhält, die Qualität der Ausgabe wesentlich verbessert.