Report from Dagstuhl Seminar 16081

Scheduling

Edited by

Nikhil Bansal[1], Nicole Megow[2], and Clifford Stein[3]

1 **TU Eindhoven, NL,** n.bansal@tue.nl
2 **TU München, DE,** nmegow@ma.tum.de
3 **Columbia University, USA,** cliff@ieor.columbia.edu

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 16081 "Scheduling". The seminar was centered around recent new developments, discussion of open problems and exploring future research directions within the broader scheduling community.

Seminar February 21–26, 2016 – http://www.dagstuhl.de/16081
1998 ACM Subject Classification C.4 Performance of Systems
Keywords and phrases approximation algorithms, scheduling, optimization
Digital Object Identifier 10.4230/DagRep.6.2.97
Edited in cooperation with Bouke Cloostermans

1 Executive Summary

Nikhil Bansal
Nicole Megow
Clifford Stein

This fourth meeting in a series of Dagstuhl "Scheduling" seminars had two major objectives. Firstly, it offered a forum for presenting recent scheduling results of high impact and new techniques which may be useful for solving important and long-standing open problems. The second major objective was to debate and explore future research directions, discuss important open problems, and foster new collaborations with a particular attention to interactions with application areas, both in academia and industry.

The organization of the meeting differed from the previous Dagstuhl "Scheduling" seminars by not inviting a different community to interact. Despite (or perhaps because of) the success of the cross-discipline events, there was an explicit desire to dedicate a seminar explicitly to recent advances and new research trends within the algorithmics/math programming scheduling community. This setting allowed for very high technical level talks and deep discussions on recent scheduling results, new techniques, and discussions on important open problems. The program included 15 invited main talks, 10 short spot-light talks, open problem sessions in the beginning of the week, and ample unstructured time for research and interaction. The overall atmosphere among the 45 participants was very interactive and oriented towards solving problems (also initiated by the few well-chosen application-driven talks) within new collaborations.

2 Table of Contents

3 Overview of Talks

3.1 Online Scheduling: New and Old Problems

Susanne Albers (TU München, DE)

We study problems in online scheduling, where jobs arrive incrementally over time and sequencing decisions must be made without knowledge of any future input. In the first part of the talk we investigate energy-efficient scheduling, where jobs are processed by variable-speed processors with the objective to minimize energy consumption. We focus on problems where jobs have deadlines and present results for multi-processor environments. Platforms with homogeneous and heterogeneous processors are considered. In the second part of the talk we revisit classical online makespan minimization. We review and develop results for various advanced problem settings in which online strategies are given extra power or information to process the incoming job sequence. In the scenarios addressed in the talk an algorithm (a) has a reordering buffer, (b) may migrate jobs, (c) is allowed to construct several candidate schedules, or (d) has some information about the job sequence.

3.2 On Speed Scaling with a Sleep State

Antonios Antoniadis (MPI für Informatik – Saarbrücken, DE)

Joint work of Susanne Albers, Antonios Antoniadis, Chien-Chung Huang and Sebastian Ott
Main reference A. Antoniadis, C. Huang, S. Ott, "A Fully Polynomial-Time Approximation Scheme for Speed Scaling with Sleep State", in Proc. of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'15), pp. 1102–1113, SIAM, 2015; pre-print available from author's webpage.
URL http://dx.doi.org/10.1137/1.9781611973730.74
URL http://www.cse.chalmers.se/~huangch/work/energysleep.pdf

We consider classical deadline-based preemptive scheduling of jobs in a computing environment equipped with both dynamic speed scaling and sleep state capabilities: Each job is specified by a release time, a deadline and a processing volume, and has to be scheduled on a single, speed-scalable processor that is supplied with a sleep state. In the sleep state, the processor consumes no energy, but a constant wake-up cost is required to transition back to the active state. In contrast to speed scaling alone, the addition of a sleep state makes it sometimes beneficial to accelerate the processing of jobs in order to transition the processor to the sleep state for longer amounts of time and incur further energy savings. The goal is to output a feasible schedule that minimizes the energy consumption.

We shortly exhibit the proof that the optimization problem is NP-hard and present a fully polynomial-time approximation scheme for the problem, which is based on a transformation to a non-preemptive variant of the problem. We conclude by discussing challenges that arise when trying to extend existing algorithms for the problem to the multiprocessor case.

3.3 Packing Small Vectors

Yossi Azar (Tel Aviv University, IL)

Joint work of Yossi Azar, Ilan Reuven Cohen, Amos Fiat, Alan Roytman
Main reference Y. Azar, I. R. Cohen, A. Fiat, A. Roytman, "Packing Small Vectors", in Proc. of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'16), pp. 1511–1525, SIAM, 2016; pre-print available from author's webpage.
URL http://dx.doi.org/10.1137/1.9781611974331.ch103
URL http://cs.tau.ac.il/~alanr/publications/FracOnVBinPack.pdf

Online d-dimensional vector packing models many settings such as minimizing resources in data centers where jobs have multiple resource requirements (CPU, Memory, etc.). However, no online d-dimensional vector packing algorithm can achieve a competitive ratio better than d. Fortunately, in many natural applications, vectors are relatively small, and thus the lower bound does not hold. For sufficiently small vectors, an $O(\log d)$-competitive algorithm was known. We improve this to a constant competitive ratio, arbitrarily close to $e \approx 2.718$, given that vectors are sufficiently small. We give improved results for the two dimensional case. For arbitrarily small vectors, the First Fit algorithm for two dimensional vector packing is no better than 2-competitive. We present a natural family of First Fit variants, and for optimized parameters get a competitive ratio ≈ 1.48 for sufficiently small vectors. We improve upon the 1.48 competitive ratio – not via a First Fit variant – and give a competitive ratio arbitrarily close to 4/3 for packing small, two dimensional vectors. We show that no algorithm can achieve better than a 4/3 competitive ratio for two dimensional vectors, even if one allows the algorithm to split vectors among arbitrarily many bins.

3.4 Applications of scheduling theory to realtime computing

Sanjoy K. Baruah (University of North Carolina at Chapel Hill, US)

Scheduling theory is one of the foundational cornerstones of the discipline of real-time computing. Prior collaborative research between the real-time computing and scheduling theory communities has yielded some important and interesting results; I will attempt to identify additional open problems in real-time computing that may be of interest to researchers in scheduling theory, and that are perhaps best solved via collaboration between the communities.

3.5 The Primal-Dual Approach to Online Optimization Problems

Niv Buchbinder (Tel Aviv University, IL)

The primal-dual method is one of the fundamental design methodologies in the areas of linear programming, combinatorial optimization, and approximation algorithms. In the area of online algorithms the primal-dual method emerged as a unifying framework to the design and

analysis of online algorithms. The method has been shown to be applicable to many central online problems such as paging, routing, scheduling, online set cover, and graph connectivity problems.

In this talk I will provide a short introduction to the primal-dual method for online combinatorial optimization. I will then discuss several recent extensions of the method to online learning and convex objective functions, and will also present several open questions.

3.6 Probabilistic RT scheduling

Liliana Cucu-Grosjean (INRIA – Paris, FR)

Probabilistic real-time scheduling considers systems that have some parameters described by probability distribution. As expected this makes the problem harder while not any probabilistic description is appropriate for calculating the response time.

3.7 A Brief History of Speedup Factors for Uniprocessor EDF and Fixed Priority Scheduling

Robert Davis (University of York, GB)

The performance of real-time scheduling algorithms can be compared using the resource augmentation bound or speedup factor. The Speedup Factor comparing two real-time scheduling algorithms X and Y is given by the minimum factor by which the speed of the processor needs to be increased so as to ensure that any task set that is schedulable under algorithm Y is guaranteed to also be schedulable under algorithm X. In this talk we give a brief tour of the exact speedup factors which have been derived for comparisons between pre-emptive and non-pre-emptive, Earliest Deadline First and Fixed Priority scheduling (EDF-P, EDF-NP, FP-P, and FP-NP) on a single processor. The scope of the problem is uniprocessor systems, assuming the sporadic (real-time) task model, and considering three different classes of task set (with implicit, constrained and arbitrary deadlines).

We begin with results for FP-P v EDF-P published in 2009 and results for FP-NP v. EDF-NP published in 2010. In both cases, recent work published in 2015 has closed the gap between upper and lower bounds providing exact speedup factors, finally closing these problems for all three classes of task set. Further, we note that all of the exact speedup factors for FP-P v EDF-P and FP-NP v. EDF-NP continue to hold when simple linear (sufficient) schedulability tests are used for FP, along with Deadline Monotonic priority assignment (even when it is not optimal). These latter results are under review.

Recent work comparing the non-pre-emptive and pre-emptive paradigms has resulted in exact speedup factors for EDF-NP v EDF-P, FP-NP v. EDF-P, and FP-NP v. FP-P. (Since non-pre-emptive scheduling suffers from the so called long task problem, these speedup factors are parametric in C_{max}/D_{min}, the ratio of the longest execution time to the shortest deadline). To the best of our knowledge, proving the exact speedup factor for FP-P v FP-NP remains an open problem and is discussed in another talk.

3.8 Wireless Scheduling

Magnus M. Halldórsson (Reykjavik University, IS)

We examine scheduling in wireless networking, focusing on throughput and latency in link scheduling at the MAC layer in physical (or SINR) models of interference. These can be captured as packing and partitioning problems in independence systems with complicated feasibility predicate. While throughput (or Capacity) has been well solved in most contexts, much less has been achieved for latency (or Link Scheduling). We discuss recent approach based on abstracting the independence system as a graph, show that it achieves improved approximations, but that it also has inherent limitations. We also touch on the challenging question of how best to model real wireless environments with algorithmically usable models.

3.9 Fair Scheduling via Iterative Quasi-Uniform Sampling

Sungjin Im (University of California – Merced, US)

Joint work of Benjamin Moseley

We consider minimizing the ℓ_k-norms of flow time on a single machine offline using a preemptive scheduler for $k \geq 1$. We show the first constant approximation for the problem, improving upon the previous best $O(\log \log P)$-approximation by Bansal and Pruhs (FOCS 09 and SICOMP 14) where P is the ratio of the maximum job size to the minimum. Our main technical ingredient is a novel combination of quasi-uniform sampling and iterative rounding, which is of interest in its own right.

3.10 Approximation Schemes for Machine Scheduling with Resource (in-)dependent Processing Times

Klaus Jansen (Universität Kiel, DE)

Joint work of Klaus Jansen, Marten Maack and Malin Rau
Main reference K. Jansen, M. Maack, M. Rau, "Approximation schemes for machine scheduling with resource (in-)dependent processing times", in Proc. of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'16), pp. 1526–1542, SIAM, 2016.
URL http://dx.doi.org/10.1137/1.9781611974331.ch104

We consider two related scheduling problems: resource constrained scheduling on identical parallel machines and a generalization with resource dependent processing times. In both problems, jobs require a certain amount of an additional resource and have to be scheduled minimizing the makespan, while at every point in time a given resource capacity is not exceeded. We present a method to obtain asymptotic fully polynomial approximation schemes (AFPTAS) for the problems.

3.11 On the strength of the Configuration-LP for the Maximum Budgeted Allocation Problem

Christos Kalaitzis (EPFL Lausanne, CH)

Joint work of Aleksander Madry, Alantha Newman, Lukáš Poláček and Ola Svensson
Main reference C. Kalaitzis, A. Madry, A. Newman, L. Polácek, O. Svensson, "On the configuration LP for maximum budgeted allocation", Math. Progr., 154(1–2):427–462, Springer, 2015.
URL http://dx.doi.org/10.1007/s10107-015-0928-8

The MBA problem is the problem of assigning items to budget-constrained agents (each of which is willing to pay up to a certain amount for each item), such that the budgets of the agents are respected and the total amount the players pay is maximized. In this talk, I will give a few insights as to why the Configuration-LP is stronger than the natural LP-relaxation for the problem, a fact which also leads to the best approximation ratio we know for the problem.

3.12 The department chair's scheduling problem

Samir Khuller (University of Maryland – College Park, US)

The objective function is simple. I'd like to schedule 1hr meetings with release times and deadlines and minimize the number of days I have to go into campus for. We called in "Active time minimization".

3.13 Practice Driven Scheduling Models

Retsef Levi (MIT – Camridge, US)

In this overview talk, we would discuss several practical examples of scheduling decision in various application domains, such as maintenance, inventory management, warehousing, and hospital operations. The practical examples are chosen to illustrate some potentially new theoretical models and algorithmic challenges with the hope to inspire new directions in the academic work.

3.14 The Lasserre/Sum-of-Squares hierarchy for friends

Monaldo Mastrolilli (IDSIA – Manno, CH)

The Lasserre/Sum-of-Squares hierarchy is a systematic procedure to strengthen LP relaxations by constructing a sequence of increasingly tight formulations. For a wide variety

of optimization problems, this approach captures the convex relaxations used in the best available approximation algorithms.

The goal of this tutorial is to introduce the non-expert to this technique by giving the essential intuition behind the definition and key properties. We discuss some simple examples, applications and limitations.

3.15 Scheduling Parallel DAG Jobs Online

Benjamin J. Moseley (Washington University – St. Louis, US)

Joint work of Kunal Agrawal, Jing Li, Kefu Lu
Main reference K. Agrawal, J. Li, K. Lu, B. Moseley, "Scheduling Parallel DAG Jobs Online to Minimize Average Flow Time", in Proc. of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'16), pp. 176–189, SIAM, 2016.
URL http://dx.doi.org/10.1137/1.9781611974331.ch14

In this talk, we discuss the problem of scheduling parallelizable jobs online. Each parallel job is modeled as a DAG where each node is a sequential task and each edge represents dependence between tasks. Previous work online has focused on a model of parallelizability known as the arbitrary speed-up curves setting. However, the DAG model is more widely used by practitioners, since many jobs generated from parallel programming languages and libraries can be represented in this model. Little is known for this model in the online setting with multiple jobs. The DAG model and the speed-up curve models are incomparable and algorithmic results from one do not imply results for the other.

Previous work has left open the question of whether an online algorithm can be $O(1)$-competitive with $O(1)$-speed for average flow time and maximum flow time in the DAG setting. In this talk will give the first scalable algorithms which are $(1+\epsilon)$-speed $O(1)$-competitive for average flow time and maximum flow time for any $\epsilon > 0$.

3.16 A Lasserre-based $(1+\epsilon)$-approximation for Makespan Scheduling with Precedence Constraints

Thomas Rothvoss (University of Washington – Seattle, US)

Joint work of Elaine Levey
Main reference E. Levey, T. Rothvoss, "A Lasserre-based $(1+\epsilon)$-approximation for $\mathrm{Pm}|\mathrm{p}_j = 1;\ \mathrm{prec}\ |\mathrm{Cmax}$", arXiv:1509.07808 [cs.DS], 2015.
URL http://arxiv.org/abs/1509.07808v1

In a classical problem in scheduling, one has n unit size jobs with a precedence order and the goal is to find a schedule of those jobs on m identical machines as to minimize the makespan. It is one of the remaining four open problems from the book of Garey & Johnson whether or not this problem is NP-hard for $m = 3$.

We prove that for any fixed epsilon and m, a Sherali-Adams/Lasserre lift of the time-index LP with slightly super poly-logarithmic number of rounds provides a $(1+\epsilon)$-approximation.

The previously best approximation algorithms guarantee a $2 - 7/(3m+1)$-approximation in polynomial time for $m >= 4$ and $4/3$ for $m = 3$. Our algorithm is based on a recursive scheduling approach where in each step we reduce the correlation in form of long chains.

Our method adds to the rather short list of examples where hierarchies are actually useful to obtain better approximation algorithms.

3.17 Online Algorithms for Multi-Level Aggregation

Jiri Sgall (Charles University – Prague, CZ)

Joint work of Marcin Bienkowski, Martin Böhm, Jaroslaw Byrka, Marek Chrobak, Christoph Dürr, Lukáš Folwarczný, Lukasz Jez, Nguyen Kim Thang, and Pavel Veselý
Main reference M. Bienkowski, M. Böhm, J. Byrka, M. Chrobak, C. Dürr, L. Folwarczny, L. Jez, J. Sgall, K. T. Nguyen, P. Veselý, "Online Algorithms for Multi-Level Aggregation", arXiv:1507.02378 [cs.DS], 2015.
URL http://arxiv.org/abs/1507.02378v2

In the Multi-Level Aggregation Problem (MLAP), requests arrive at the nodes of an edge-weighted tree T, and have to be served eventually. A service is defined as a subtree X of T that contains its root. This subtree X serves all requests that are pending in the nodes of X, and the cost of this service is equal to the total weight of X. Each request also incurs waiting cost between its arrival and service times. The objective is to minimize the total waiting cost of all requests plus the total cost of all service subtrees.

MLAP is a generalization of some well-studied optimization problems; for example, for trees of depth 1, MLAP is equivalent to the TCP Acknowledgment Problem, while for trees of depth 2, it is equivalent to the Joint Replenishment Problem. Aggregation problem for trees of arbitrary depth arise in multicasting, sensor networks, communication in organization hierarchies, and in supply-chain management. The instances of MLAP associated with these applications are naturally online, in the sense that aggregation decisions need to be made without information about future requests.

Constant-competitive online algorithms are known for MLAP with one or two levels. However, it has been open whether there exist constant competitive online algorithms for trees of depth more than 2. Addressing this open problem, we give the first constant competitive online algorithm for networks of arbitrary (fixed) number of levels. The competitive ratio is $O(D^4 2^D)$, where D is the depth of T. The algorithm works for arbitrary waiting cost functions, including the variant with deadlines.

3.18 Bounded Preemptions in Real-time Scheduling

Hadas Shachnai (Technion – Haifa, IL)

Scheduling problems are traditionally classified as either preemptive, or non-preemptive. However, practical scenarios call for a bounded-preemptive scheduling model, which allows to reduce costs and the runtime overhead caused by preemptions. We study the bounded preemptions model in real-time scheduling, where the total number of preemptions, or the number of preemptions per job is bounded by an integer parameter, k. The goal is to feasibly schedule on a single machine a subset of the jobs of maximum total weight. Our theoretical results establish an interesting distinction between the hardness status of non-preemptive, or (unbounded) preemptive real-time scheduling, and k-bounded preemptive

scheduling, already for uniform-length jobs with unit weights, for a wide range of values of k. Constant-factor approximation algorithms are obtained for subclasses of instances. Our experimental study shows that while k-bounded preemptive scheduling is hard to solve already on highly restricted instances, simple priority-based heuristics yield almost optimal schedules for realistic inputs and arbitrary values of k.

3.19 Smarter tools for (Citi)bike-sharing

David Shmoys (Cornell University, US)

New York launched the largest bike-sharing system in North America in May 2013. We have worked with Citibike, using analytics and optimization to change how they manage the system. Huge rush-hour usage imbalances the system; we answer both of the questions: where should bikes be at the start of a rush hour and how can we mitigate the imbalances that develop? We will outline the analytics we have employed for the former question, where we developed an approach based continuous-time Markov chains combined with integer programming models to compute daily stocking levels for the bikes, as well as methods employed for optimizing the capacity of the stations. For the latter problem, we present a 3-approximation algorithm used to target limited available rebalancing resources during rush-hour periods. The goal is to ensure that users are never too far from a station that will be rebalanced when looking for a bike or a spot to return one. We formulate this as a variant of the k-center problem and provide a linear programming-based algorithm with a performance guarantee of 3. Finally, we briefly discuss two further planning questions related to their operations, the placement of a limited number of so-called corrals, and scheduling the maintenance of the batteries powering the stations.

3.20 No-wait scheduling for locks

Frits C. R. Spieksma (KU Leuven, BE)

Joint work of Ward Passchyn and Dirk Briskorn.
Main reference W. Passchyn, D. Briskorn, F. C. R. Spieksma, "Mathematical programming models for lock scheduling with an emission objective", European Journal of Operational Research (EOR), 248(3):802–814, 2016.
URL http://dx.doi.org/10.1016/j.ejor.2015.09.012

We investigate a problem inspired by the practical setting of scheduling a lock with multiple chambers. We show how this problem relates to known interval scheduling problems. In particular, for a lock consisting of two chambers we are able to characterize the feasible instances, and use this result to obtain efficient algorithms. We also provide an efficient algorithm for the special case with identical lock chambers. Furthermore, we describe a dynamic programming approach for the more general case with arbitrary chambers, and prove that the problem is strongly NP-complete when the number of chambers is part of the input.

3.21 Lift-and-Round to Improve Scheduling on Unrelated Machines

Ola Svensson (EPFL – Lausanne, CH)

Joint work of Nikhil Bansal and Aravind Srinivasan
Main reference N. Bansal, O. Svensson, A. Srinivasan, "Lift-and-Round to Improve Weighted Completion Time on Unrelated Machines", arXiv:1511.07826 [cs.DS], 2015.
URL http://arxiv.org/abs/1511.07826v2

Weighted completion time and makespan are some of the most relevant and well-studied measures of quality of service in scheduling and resource allocation problems. While these objectives have been studied since the 50's, a systematic study of their approximability was started in the 90's. Since then, impressive progress has led to a complete understanding of these objectives in simple machine models, such as identical machines.

In contrast, it remains a notorious problem to understand the approximability in the more general unrelated machine setting: the classic algorithms developed in the 90's resisted any improvements while having guarantees that are far from the strongest known lower bounds.

In this talk, we overview recent developments with a focus on our recent approximation algorithm that improves upon the barrier of 3/2 for the weighted completion time objective. The improvement is based on a new lift-and-project based SDP relaxation and a general bipartite-rounding procedure that produces an assignment with certain strong negative correlation properties.

This is based on joint work with Nikhil Bansal and Aravind Srinivasan.

3.22 Closing the gap for makespan scheduling via sparsification techniques

Jose Verschae (Pontifical Catholic University of Chile – Santiago, CL)

Joint work of Klaus Jansen and Kim-Manuel Klein
Main reference K. Jansen, K.-M. Klein, J. Verschae, "Closing the Gap for Makespan Scheduling via Sparsification Techniques", in Proc. of the 43rd Int'l Colloquium on Automata, Languages and Programming (ICALP 2016), 2016; to appear.

Makespan scheduling on identical machines is one of the most basic and fundamental packing problems studied in the discrete optimization literature. It asks for an assignment of n jobs to a set of m identical machines that minimizes the makespan. The problem is strongly NP-hard, and thus we cannot expect a $(1+\epsilon)$-approximation algorithm to have polynomial dependency on $1/\epsilon$. Very recently, Chen et al. showed that the dependency has to be exponential if we assume the Exponential Time Hypothesis. A long sequence of algorithms have been developed that try to obtain low dependencies on $1/\epsilon$, but they are all super-exponential on $1/\epsilon$. In this talk I will discuss an improved algorithm that almost matches the lower bound by Chen et al., essentially closing this long line of research. The new ingredient of our approach is an observation that guarantees the existence of a highly symmetric and sparse almost-optimal solution. This structure can then be exploited by integer programming techniques and enumeration. The same idea helps us to obtain improved algorithms for a number of different related problems, including the minimum makespan problem on related machines.

3.23 Approximation algorithms for geometric packing problems

Andreas Wiese (MPI für Informatik – Saarbrücken, DE)

Joint work of Giorgi Nadiradze and Anna Adamaszek

A common setting in geometric packing problems is that we are given a set of two-dimensional items, e.g., rectangles, and a rectangular container and our goal is to pack these items or a subset of them items into the container to optimize objective functions like the total profit of the packed items or the necessary height of the container. A typical obstacle in these problem settings is that in the input there are different types of items, i.e., items that are wide and thin, that are high and narrow, or items that are large in both dimensions. In this talk, I will present a method to handle this obstacle. In a nutshell, the key is to prove that there are near-optimal solutions in which the given container can be partitioned into few rectangular boxes such that in each box there are only items of one of the mentioned types. This leads to better approximation guarantees for two specific problems: a $(1+\epsilon)$-approximation algorithm in quasi-polynomial time for the two-dimensional knapsack problem and a $(1.4+\epsilon)$-approximation algorithm in pseudo-polynomial time for the strip-packing problem. Note that the latter bound is strictly smaller than the lower bound of 3/2 that holds for (non-pseudo-)polynomial time algorithms for the problem.

3.24 Scheduling for Electricity Cost in Smart Grid

Prudence W. H. Wong (University of Liverpool, GB)

Joint work of Mihai Burcea, Wing-Kai Hon, Hsiang-Hsuan Liu, David K. Y. Yau
Main reference M. Burcea, W.-K. Hon, H. H. Liu, P. W. H. Wong, D. K. Y. Yau, "Scheduling for Electricity Cost in Smart Grid", in Proc. of the 7th Int'l Conf. on Combinatorial Optimization and Applications (COCOA 2013), LNCS, Vol. 8287, pp. 306–317, Springer, 2013; pre-print available from author's webpage.
URL http://dx.doi.org/10.1007/978-3-319-03780-6_27
URL https://www.cs.purdue.edu/homes/lans/20061129/publications/burcea2013cocoa.pdf

We study a scheduling problem arising in demand response management in smart grid. Consumers send in power requests with a flexible feasible time interval during which their requests can be served. The grid controller, upon receiving power requests, schedules each request within the specified interval. The electricity cost is measured by a convex function of the load in each timeslot. The objective is to schedule all requests with the minimum total electricity cost. We study different variants and obtain exact algorithms and online algorithms.

4 Open problems

4.1 Multiprocessor Sleep-State Scheduling

Antonios Antoniadis

Consider a set of processors each of which has a unit power consumption per time unit while being active and can transition to sleep at a fixed energy cost α. We are given a set of tasks each associated with a release time, a deadline and a length, and wish to preemptively schedule them so that we minimize one of the following two objectives: (i) The total power consumption. (ii) The number of gaps. Here, a gap is defined as a maximal interval during which a processor is not processing any task. ((ii) can be seen as a special case of (i).)

Questions:

1. Are the two problems NP-hard? Polynomial-time solvable?
2. Develop a constant-factor approximation algorithm for any of the problems?
3. With respect to (ii), is the Longest Viable Gap (LVG) algorithm a constant factor approximation algorithm? Roughly speaking, LVG is a greedy algorithm that repeatedly identifies and fixes the longest possible gap in the instance.

4.2 Exact Speedup Factor for Fixed Priority Pre-emptive versus Fixed Priority Non-pre-emptive Scheduling

Rob I. Davis

The Speedup Factor comparing two real-time scheduling algorithms X and Y is given by the minimum factor by which the speed of the processor needs to be increased so as to ensure that any task set that is schedulable under algorithm Y is guaranteed to also be schedulable under algorithm X. Earliest Deadline First Pre-emptive (EDF-P) scheduling is an optimal single processor scheduling algorithm and so dominates Fixed Priority Pre-emptive (FP-P) and Fixed Priority Non-Pre-emptive (FP-NP) scheduling. There is; however, no such dominance relationships between fixed priority pre-emptive and fixed priority non-pre-emptive scheduling. Determining the exact speedup factor for FP-P v. FP-NP is the focus of this talk. The scope of the problem is uniprocessor systems, assuming the sporadic (real-time) task model. We show, via an example, that the speedup factor is lower bounded by $\sqrt{2}$. We then present the results of experiments using a genetic algorithm to search for high speedup factor task sets. These results indicate that the speedup factor appears to also be upper bounded by $\sqrt{2}$. We conjecture that the exact speedup factor is $\sqrt{2}$. Proof is needed!

4.3 Approximate Mixed-Criticality Scheduling

Christoph Dürr

We consider a single machine scheduling problem, where we want to minimize the makespan, but the exact processing times of jobs is not known at the moment we commit to the starting times of the jobs. But we have the possibility at execution time to drop some less important jobs when the long execution of some previous jobs prevent their executions. This problem falls into the area of mixed criticality scheduling, which has been studied mostly in the real time scheduling community [1, 2].

Formally for this problem there is a single machine and k levels of criticality[1]. Every given job j has a priority level $\ell_j \in [k]$ and processing times $p_j^1 \leq \ldots \leq p_j^{\ell_j}$.

The goal is to produce a schedule defined by starting times S_j for each job, such that for every criticality $c \in [k]$ and jobs i, j with $\ell_i, \ell_j \geq c$ we have $S_i + p_i^c \leq S_j$ or $S_j + p_j^c \leq S_i$. The goal is to minimize the *makespan*, which is defined as $\max_{c\in[k]} \max_{j:\ell_j\geq c} S_j + p_j^c$.

job j	priority level	p_j^1	p_j^2	p_j^3
1	3	2	3	5
2	1	1		
3	2	1	2	

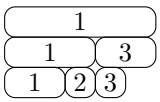

Figure 1 Example of an optimal schedule: the rows correspond to criticality levels and the columns to time slots.

The problem could be strongly NP-hard as for $k = 2$ by a possible reduction from the 3-Partition problem. What is the best achievable approximation ratio by a polynomial time algorithm? Could it be a constant?

In [3] a greedy algorithm is analyzed for the special case $p_j^c = c$, and showed to have ratio between 1.05 and 1.5. It should be possible to tighten the bounds for this special case.

References

1 S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. M. Spaccamela, N. Megow, and L. Stougie. Scheduling real-time mixed-criticality jobs. *IEEE Transactions on Computers*, 61(8):1140–1152, 2012.
2 A. Burns and R. I. Davis. Mixed criticality systems – a review. 7th edition, 2016.
3 C. Dürr, Z. Hanzálek, C. Konrad, R. Sitters, O. C. Vásquez, and G. J. Woeginger. The triangle scheduling problem. *CoRR*, abs/1602.04365, 2016.

[1] $[k]$ stands for the set $\{1, \ldots, k\}$.

4.4 Scheduling and Load Balancing with Recourse

Anupam Gupta and Amit Kumar

In many online scheduling settings, we require jobs to be dispatched immediately to a machine. Consider the example of load balancing. Jobs arrive online, and each job j specifies a size p_j and a subset S_j of machines in which it can be scheduled (the so-called "restricted assignment" model). When a job arrives, it needs to be dispatched to a machine. The goal is to minimize the maximum load on a machine, where the load on machine is defined as the total processing time of jobs dispatched to it. The greedy algorithm, which dispatches a job to the least loaded machine, has competitive ratio of $O(\log m)$, where m is the number of machines; and it is known that this is tight even when all jobs sizes are same [2]. If we were allowed to change the allocation of jobs to machines (instead of immediate dispatch), we could solve the off-line load balancing problem at each time instant. However, this may completely disrupt the current assignment of jobs to machines. In the scheduling with recourse model, we ask the following question : Can we improve the competitive ratio if we are allowed to make small changes in the job assignment? As an example, we can show the following: consider the load balancing problem described above in the restricted assignment setting with unit size jobs. We can get a constant competitive algorithm if we are allowed to change the assignment of one existing job (in an amortized manner) whenever a new job arrives [3]. In general, we seek to study how much advantage recourse gives us over traditional competitive analysis. Here are some specific questions:

- Consider the setting of load balancing as above for general job sizes. Can we get constant competitive algorithms with constant amount of job reassignment per job arrival? We can only get an online algorithm which is O(log log n)-competitive (with constant recourse) [3].
- Suppose jobs are arriving over time (in restricted assignment setting), and we want to minimize the maximum flow-time of a job. If we insist on immediate dispatch, it is known that there is no constant competitive algorithm [1]. What happens when we allow recourse?

References

1 S. Anand, K. Bringmann, T. Friedrich, N. Garg, and A. Kumar. Minimizing maximum (weighted) flow-time on related and unrelated machines. In *Proceedings of the 40th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 13–24, 2013.

2 Y. Azar, J. S. Naor, and R. Rom. The competitiveness of on-line assignments. In *Proceedings of the Third Annual ACM-SIAM symposium on Discrete Algorithms (SODA)*, pages 203–210, 1992.

3 A. Gupta, A. Kumar, and C. Stein. Maintaining assignments online: matching, scheduling, and flows. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 468–479, 2014.

4.5 Wireless Scheduling

Magnus M. Halldórsson

Let $p_1, p_2, \ldots, p_n$ be points on the real line with *capacities* $c_1, \ldots, c_n$. The problem is to partition $P = \{p_i\}$ into fewest sets $P_1, \ldots, P_t$, such that

$$\sum_{p' \in P_i, p' \neq p} |p - p'|^3 \leq c_i, \qquad \text{for each } i \text{ and each } p \in P_i.$$

We seek a $O(1)$-approximation. This problem statement captures the most basic open question in scheduling wireless links under the physical (or, SINR) model. Normally, links are given as sender-receiver pairs, but it is known that when messages are all transmitted with the same uniform power, we can blur the distinction between sender and receiver, by paying a constant factor. The problem is usually specified on the plane, or in a general distance metric, but results for the one-dimensional case can typically be generalized relatively easily. The exponent "3", known as the path-loss constant, is situation dependent, and can be any number between 2 and 6.

A $O(1)$-approximation is known for the throughput problem of finding a single set P_1 of maximum cardinality within which all points satisfy the inequality above [1]. This immediately give a $O(\log n)$-factor, but no better is known.

References

1 O. Goussevskaia, R. Wattenhofer, M. M. Halldórsson, and E. Welzl. Capacity of arbitrary wireless networks. In *28th IEEE International Conference on Computer Communications (INFOCOM)*, pages 1872–1880, 2009.

4.6 Stochastic and Multi-Dimensional Scheduling, Generalized Min-Sum Set Cover

Sungjin Im

1. Stochastic scheduling. We are asked to schedule jobs on parallel identical machines non-preemptively with the goal of minimizing total completion time in expectation when the probability distribution on the size of each job is known a priori. The distributions can be arbitrary. Is there a constant approximation for this problem? Even constant approximations with $O(1)$ machine augmentation would be interesting.
2. Generalized min-sum set cover. There is a collection of sets over a universe of elements. Each set S_i has a requirement k_i. We are asked to order elements linearly. A set is said to be covered at time t if t is the first time t' such that k_i elements in S_i appear in the first t' elements of the ordering. The goal is to minimize the total cover time of all sets. The question is if there exists a 4-approximation for this problem. When $k_i = 1$ for all i, there are 4-approximations known. If $k_i = |S_i|$ for all i, there are 2-approximations known. But for the general k_i, the best known approximation factor is greater than 12.
3. In multi-dimensional scheduling, each job j has a demand vector over multiple resources r_j. One can process jobs at rates of $\{x_j\}$ if $\sum_j r_j x_j \leq 1$. The goal is to minimize total

flow time. A scalable algorithm is known for this problem, but the competitive ratio has an exponential dependency on D where D is the number of dimensions. Is it possible to get a $O(1)$-speed $O(poly(D))$-competitive algorithm?

4.7 Generalized assignment

Christos Kalaitzis

The Generalized Assignment Problem is the problem of assigning items, which have a certain size and value, to unit-capacity bins , such that the capacity of the bins is respected, and the sum of the values of the assigned items is maximized. For this problem, it is known that an LP-relaxation called the Configuration-LP has an integrality gap of $1 - 1/e + c$, for some small $c > 0$, which also constitutes the best approximation ratio we know. It would be interesting to look into specific restrictions of the problem, and come up with new techniques to tighten the gap between the upper and lower bounds we have for the Configuration-LP.

The Restricted Max-Min Allocation Problem is the problem of assigning jobs (each of which has a certain processing time) to machines (where each job can only be assigned to a specific subset of the machines), such that all jobs are processed and we maximize the minimum of the sum of processing times over all machines. For this problem, we know that the Configuration-LP has an integrality gap between 1/4 and 1/2, and it would be interesting to investigate its exact value (even if the resulting algorithm is inefficient).

4.8 Flow Time Scheduling on a Line Network

Benjamin Moseley

Say you are given a graph $G = (V, E)$ that is a line. Each node of the graph is a router that forwards packets. Each router can forward up to one packet each time step. Assume that the routers always forward packets to the router to the right for simplicity. There are no buffers and routers can store any number of packets, but can only forward one per time unit.

Requests arrive over time where a request i has an arrival time r_i, a source node s_i and a destination node d_i. Each request is for a single packet to be routed from s_i to d_i. Under an algorithm A, let C_i^A be the time that the packet for request i reaches the designation router d_i. The goal is to decide how packets should be scheduled (routed) to minimize $\sum_i C_i^A - r_i$, the average flow time of the schedule.

See [1, 2] for related work. It is known that no $O(1)$ competitive online algorithm exists. There are no non-trivial upper bounds, even with resource augmentation. A clear goal would be to find a $O(1)$-speed $O(1)$-competitive algorithm. The work of [1] discusses possible algorithms for the problems and their strengths, weaknesses and some properties. The work also gives a $1+\epsilon$ speed $O(1)$-competitive algorithm for the same problem, but the objective is minimizing the maximum flow time. The work of [2] considers the same objective, but looks at tree networks where the source is always the same and gives $O(1)$-speed $O(1)$-competitive algorithms.

References

1 A. Antoniadis, N. Barcelo, D. Cole, K. Fox, B. Moseley, M. Nugent, and K. Pruhs. Packet forwarding algorithms in a line network. In *Proceedings of the 11th Latin American Symposium (LATIN)*, pages 610–621, 2014.

2 S. Im and B. Moseley. Scheduling in bandwidth constrained tree networks. In *Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 171–180, 2015.

4.9 Are Existentially Scalable Algorithms Really Necessary?

Kirk Pruhs

A scheduling algorithm A is universally scalable if it is $(1+\epsilon)$- speed $O_\epsilon(1)$-competitive, where the subscript of ϵ denotes that the constant can depend upon ϵ. A collection $\{A_\epsilon\}$ of scheduling algorithms, one for each $\epsilon > 0$, is existentially scalable if each A_ϵ is $(1+\epsilon)$-speed $O_\epsilon(1)$-competitive. Thus in this collection the algorithm that 1.1-speed $O(1)$-competitive may well not be the same algorithm that is 1.01-speed $O(1)$-competitive. There are many online scheduling problems where existentially scalable algorithms are known, but no universally scalable online algorithm is known. Examples include broadcast scheduling to minimize total flow time [1], scheduling jobs with arbitrary speed-up curves to minimize total flow[3], broadcast scheduling to minimize maximum flow [2], scheduling unrelated machines to minimize p-norms of flow [4]. A natural, and I believe important, open question is whether universally scalable algorithms exist for these problems. In this short talk, I want to publicize this problem, and briefly discuss some of the difficulties one immediately encounters.

References

1 N. Bansal, R. Krishnaswamy, and V. Nagarajan. Better scalable algorithms for broadcast scheduling. *ACM Transactions on Algorithms (TALG)*, 11(1):3, 2014.

2 C. Chekuri, S. Im, and B. Moseley. Online scheduling to minimize maximum response time and maximum delay factor. *Theory of Computing*, 8(1):165–195, 2012.

3 J. Edmonds and K. Pruhs. Scalably scheduling processes with arbitrary speedup curves. *ACM Transactions on Algorithms (TALG)*, 8(3):28, 2012.

4 S. Im and B. Moseley. An online scalable algorithm for minimizing ℓ_k-norms of weighted flow time on unrelated machines. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms(SODA)*, pages 95–108, 2011.

4.10 Precedence-Constrained Scheduling on Identical Machines to minimize weighted completion time

Ola Svensson

To minimize weighted sum of completion times in the presence of precedence constraints is well understood in the case of a single machine: there are plenty of 2-approximation algorithms and the factor 2 is tight assuming a variant of the UGC.

The natural question that follows is what happens if we consider parallel identical machines? In this case, the best algorithm is due to Queyranne and Schulz who gave a 4-approximation algorithm. However the best known integrality gap of the known LP is 2 which is the hardness of the single machine problem.

So the research question is to design a better algorithm for $P|prec|\sum w_jC_j$ or at least give a stronger integrality gap.

4.11 Scheduling with state-dependent machine speed

Tjark Vredeveld

During the 2013 Dagstuhl seminar on Scheduling, Urtzi Ayesta proposed the following open problem. Given are n jobs that need to be processed on a single machine. The jobs have a processing time p_j and weight w_j. The machine may preempt the job. Moreover, the speed of the machine is dependent on the number of jobs in the system. Hereto, speeds $s_1, \ldots, s_n$ are given, where speed s_i denotes the speed of the machine when $i-1$ jobs have been completed. The goal is to minimize the total weighted completion time.

When all speeds are equal, $s_i = 1$ for all $i = 1, \ldots, n$, then the preemption is not used and this scheduling problem is solved by Smith's rule [1], also known as the WSPT rule: process jobs in order of non-increasing ratio $w_j = p_j$. On the other hand, for arbitrary varying speeds, preemption may be useful. The preemption model can be viewed as a processor sharing model in which the processing capacity of the machine is divided over several jobs that can be processed simultaneously.

In [2], we proved that there exists an optimal schedule that processes jobs in groups. That is in between two consecutive (different) completion times, the machines processes on a group of one or more jobs that all finish and start at the same time. We furthermore proved that the general problem is strongly NP-hard by a reduction from 3-Partition. In the same paper, we also designed a greedy algorithm that, given the order of job completions, decides the optimal grouping of jobs. That is, the main issue is to determine the right order in which the jobs should complete.

Even in the case of monotone non-decreasing or non-increasing speeds, WSPT does not necessarily give an optimal order on the job completions. This can be seen by the following two examples.

For the case of non-decreasing speeds, the example consists of 2 jobs. The first job has processing time $p_1 = \epsilon$ and weight $w_1 = 0$. The second job has processing time $p_2 = A$ and weight A, for some large A. The speed of the machine, when two jobs are still in the system, is $s_1 = 1$ and when one job has completed the speed becomes $s_2 = A$. Having the jobs finished in WSPT order, we first process job 2 and then job 1, giving a total weighted completion time of WSPT $= A^2$. The optimal ordering would be to finish first the small and unimportant job 1, so that the large job 2 can be processed at speed A. Then the total weighted completion time will be $\mathsf{OPT} = A(1+\epsilon)$. Hence, $\frac{\mathsf{WSPT}}{\mathsf{OPT}} \geq A - \epsilon' = \frac{s_{\max}}{s_{\min}} - \epsilon'$. By giving the optimum the power to process all jobs at maximum speed and WSPT the disadvantage of processing all jobs at minimum speed, it is easy to see that it also holds that for all instances $\frac{\mathsf{WSPT}}{\mathsf{OPT}} \leq \frac{s_{\max}}{s_{\min}}$.

For the case of non-increasing speeds, the bound is better. A lower bound can be given by the following example of two jobs. The first job has processing time and weight $w_1 = p_1 = 1$

and the second job has processing time $w_2 = 1 + \sqrt{2}$ and $p_2 = w_2 + \epsilon$. Hence, in the WSPT order job 1 precedes job 2. The speeds of the machines are $s_1 = \sqrt{2}$ and $s_2 = 1$. It can be shown that when having the jobs finished in WSPT order, it is equally good to first process job 1 and then job 2 as processing both jobs at the same time. This results in a total weighted completion time of $\mathsf{WSPT} = 4 + 3\sqrt{2}$. The optimum would first process job 2 and then job 1, which gives a total weighted completion time of $\mathsf{OPT} = 4 + 2\sqrt{2}$. Hence, we have a lower bound on the approximation ratio of WSPT of $\frac{\mathsf{WSPT}}{\mathsf{OPT}} \geq \frac{1+\sqrt{2}}{2}$. The algorithms that processes all jobs in one group gives a 2-approximation for the case of non-increasing speeds.

We are interested in the following open problems.

- The NP-hardness proof uses the fact that the processing times may change arbitrarily. What is the complexity of the problem when the speeds are monotone non-decreasing or non-increasing?
- What is the complexity of the non-preemptive version of the problem?
- What is the approximation ratio of WSPT for the case of non-increasing speeds?

References

1 W. E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1-2):59–66, 1956.

2 V. Timmermans and T. Vredeveld. Scheduling with state-dependent machine speed. In *Workshop on Approximation and Online Algorithms (WAOA)*, pages 196–208, 2015.

Participants

- Fidaa Abed
TU München – Munich, DE
- Susanne Albers
TU München, DE
- Antonios Antoniadis
MPI für Informatik – Saarbrücken, DE
- Yossi Azar
Tel Aviv University, IL
- Nikhil Bansal
TU Eindhoven, NL
- Sanjoy K. Baruah
University of North Carolina at Chapel Hill, US
- Vincenzo Bonifaci
CNR – Roma, IT
- Niv Buchbinder
Tel Aviv University, IL
- Marek Chrobak
University of California – Riverside, US
- Bouke Cloostermans
TU Eindhoven, NL
- Liliana Cucu-Grosjean
INRIA – Paris, FR
- Robert Davis
University of York, GB
- Christoph Dürr
UPMC – Paris, FR
- Thomas Erlebach
University of Leicester, GB
- Anupam Gupta
Carnegie Mellon University, US
- Magnus M. Halldórsson
Reykjavik University, IS
- Sungjin Im
University of California – Merced, US
- Klaus Jansen
Universität Kiel, DE
- Christos Kalaitzis
EPFL Lausanne, CH
- Samir Khuller
University of Maryland – College Park, US
- Amit Kumar
Indian Inst. of Technology – New Dehli, IN
- Retsef Levi
MIT – Camridge, US
- Alberto Marchetti-Spaccamela
Sapienza University of Rome, IT
- Monaldo Mastrolilli
IDSIA – Manno, CH
- Nicole Megow
TU München, DE
- Rolf H. Möhring
TU Berlin, DE
- Benjamin J. Moseley
Washington University – St. Louis, US
- Seffi Naor
Technion – Haifa, IL
- Kirk Pruhs
University of Pittsburgh, US
- Thomas Rothvoss
University of Washington – Seattle, US
- Jiri Sgall
Charles University – Prague, CZ
- Hadas Shachnai
Technion – Haifa, IL
- David Shmoys
Cornell University, US
- René Sitters
VU University of Amsterdam, NL
- Frits C. R. Spieksma
KU Leuven, BE
- Clifford Stein
Columbia University, US
- Ola Svensson
EPFL – Lausanne, CH
- Marc Uetz
University of Twente, NL
- Suzanne van der Ster
TU München, DE
- Rob van Stee
University of Leicester, GB
- Jose Verschae
Pontifical Catholic University of Chile – Santiago, CL
- Tjark Vredeveld
Maastricht University, NL
- Andreas Wiese
MPI für Informatik – Saarbrücken, DE
- Gerhard J. Woeginger
TU Eindhoven, NL
- Prudence W. H. Wong
University of Liverpool, GB