



Proceedings of the 17th Workshop on Models and
Algorithms for Planning and Scheduling Problems,
MAPSP 2026

June 21-26, 2026, Redworth Hall Hotel, County Durham, United Kingdom

Sponsored by



ACM Special Interest Group on Algorithms
and Computation Theory

Program committee

Alexander Lindermayr	TU Berlin
Alison Liu	Utrecht University
Arindam Khan	IISC Bangalore
Franziska Eberle	TU Berlin
Jens Schlöter	CWI
Lars Rohwedder (chair)	University of Southern Denmark
Marek Elias	Bocconi University
Moritz Buchem	Bremen University
Ruilong Zhang	TU Munich
Sami Davis	Simons Institute and UC Berkeley
Sander Borst	MPI Saarbrücken
Tjark Vredeveld	Maastricht University

Organizing committee

Prudence Wong	University of Liverpool
Thomas Erlebach	Durham University

Preface

This volume contains abstracts of talks presented at the 17th Workshop on Models and Algorithms for Planning and Scheduling Problems (MAPSP 2026), held from June 21 to June 26, 2026, in County Durham, United Kingdom.

MAPSP is a biennial workshop dedicated to all theoretical and practical aspects of scheduling, planning, and timetabling. Previous MAPSP meetings have been held in Menaggio, Italy (1993), Wernigerode, Germany (1995), Cambridge, UK (1997), Renesse, Netherlands (1999), Aussois, France (2001), Aussois, France (2003), Siena, Italy (2005), Istanbul, Turkey (2007), Kerkrade, Netherlands (2009), Nymburk, Czech Republic (2011), Pont á Mousson, France (2013), La Roche-en-Ardenne, Belgium (2015), Seeon, Germany (2017), Renesse, Netherlands (2019), Oropa, Italy (2022), and Kolding, Denmark (2024).

The abstracts in this volume include 3 invited talks by Franziska Eberle, Andreas Wiese, and Jannik Matuschke, plus 61 contributed talks. Each submission went through a light review process. The submitted papers are presented in two parallel tracks.

We thank our sponsor SIGACT for their generous support and the members of the program and organizing committees for their work.

Lars Rohwedder
Prudence Wong
Thomas Erlebach

Contents

Preface	4
Invited talks	10
Session 1 (Monday morning)	11
Franziska Eberle: Stochastic Load Balancing and Related Problems	11
Session 5 (Tuesday morning)	12
Jannik Matuschke: When Nicolas and Vilfredo Join Hands: Condorcet Dimension and Pareto Optimality for Matchings and Beyond	12
Session 11 (Thursday morning)	13
Andreas Wiese: Practical scheduling problems and where to find them ... at my university	13
Contributed talks	13
Session 2A (Monday morning)	15
Ya-Chun Liang, Jian-Xi Shao and Chung-Shou Liao: Revisit the Online TSP on the Line	15
Anna Hu and Hsiang-Hsuan Liu: Power of knowing the full neighborhood of online vertex cover	16
David Shmoys, Varun Suriyanarayana and Seeun William Umboh: Improved On-line Algorithms for the JRP with Holding & Delay Costs: Riding the Wave Makes Things Simpler, Stronger, & More General	19
Session 2B (Monday morning)	24
Sanjoy Baruah and Pontus Ekberg: Rethinking efficiency in real-time schedulability analysis	24
Sanjoy Baruah and Pontus Ekberg: Efficient explainability of schedulability analysis	26
Mario Günzel, Marion Sudvarg, Max Deppert, Ao Li, Ning Zhang and Jian-Jia Chen: Optimal Priority Assignment for Synchronous Harmonic Tasks With Dynamic Self-Suspension	29

Session 3A (Monday afternoon)	33
Maximilian von Aspern, Felix Buld and Michael Pinedo: Flow Shop Scheduling with Stochastic Reentry	33
Sven Jäger and Daniel Schmidt Genannt Waldschmidt: A Novel IP Formulation for Stochastic Non-Preemptive Scheduling	36
Session 3B (Monday afternoon)	40
Martijn van Ee and Rene Sitters: Approximation algorithms for graph search problems with imperfect detection	40
Sunny Atalig, Marek Chrobak, Christoph Dürr, Petr Kolman, Huong Luu, Jiri Sgall and Gregory Zhu: Two Complexity Results on Spanning-Tree Congestion Problems	42
Session 4A (Monday afternoon)	46
Mong-Jen Kao: On the Integrality Gap of MFN Relaxation for the Capacitated Facility Location Problem	46
Sander Borst, Golnoosh Shahkarami and Rohit Vaish: Interval Scheduling under Approximate Envy-Freeness	49
Sami Davies, Venkatesan Guruswami and Xuandi Ren: Scheduling Problems with Constrained Rejections	52
Session 4B (Monday afternoon)	56
Moritz Buchem, Nicole Megow, Marc Uetz and Leoni Winschermann: Approximating Fair Repetitive Scheduling	56
Mirabel Mendoza-Cadena, Arturo Merino, Mads Anker Nielsen and Kevin Schewior: Combinatorial Perpetual Scheduling	58
Kalina Jasinska, John Kuzmaul and Gyudong Lee: Strengths and Limitations of Greedy in Cup games	61
Session 6A (Tuesday morning)	68
Antonios Antoniadis, Denise Graafsma, Ruben Hoeksma and Maria Vlasiou: Refining the Complexity Landscape of Speed Scaling: Hardness and Algorithms	68
Peter Gyorgyi and Tamas Kis: Resource leveling problems with precedence constraints and convex cost functions	70
Klaus Jansen and Felix Ohnesorge: A Practical 73/50 Approximation for Contiguous Monotone Moldable Job Scheduling	73
Session 6B (Tuesday morning)	77
Benjamin Moseley, Heather Newman, Kirk Pruhs and Rudy Zhou: Robust Gittins for Stochastic Scheduling	77
Benjamin Moseley, Kirk Pruhs, Marc Uetz and Rudy Zhou: Minimizing Completion Times of Stochastic Jobs on Parallel Machines is Hard	79
P.J. van Mill, André Berger and Tjark Vredeveld: A further investigation of Learning-SEPT	82
Session 7A (Tuesday afternoon)	86
Eyüp Ensar Işık, Z. Caner Taşkın and Semra Ağralı: A Decomposition-Based Exact Solution Approach for Lot-sizing and Scheduling Problem in Co-production Systems	86

Tanvi Hisaria, Neel Karia, Clifford Stein, Asser Tantawi, Olivier Tardieu and Wenqing Yu: SMART-MIG: A Learning Framework for Scalable and Energy-Efficient GPU Scheduling	88
Session 7B (Tuesday afternoon)	92
Steven Miltenburg: On the Complexity of the Euclidean Capacitated Vehicle Routing Problem	92
Steven Miltenburg, Tim Oosterwijk and René Sitters: Capacitated Vehicle Routing with Order Restrictions: Models, Algorithms, and Limits	94
Session 8A (Tuesday afternoon)	98
Mateusz Basiak, Marcin Bienkowski, Martin Böhm, Marek Chrobak, Łukasz Jeż, Jiří Sgall and Agnieszka Tatarczuk: A 3.3904-Competitive Online Algorithm for List Update with Uniform Costs	98
Michael A. Bender, Alex Conway, Daniel DeLayo, Martin Farach-Colton, Jaehyun Han, Linfeng He, Rob Johnson, Sudarsun Kannan, William Kuzmaul, Donald Porter and Evan West: Don't Melt Your Cache: Low-Associativity with Heat-Sink	100
Enoch Peserico and Michele Scquizzato: Is competitive paging an artifact?	104
Session 8B (Tuesday afternoon)	108
Stephen Arndt, Benjamin Moseley, Kirk Pruhs and Michael Zlatin*: Edmonds++: Efficiently Coloring more Matroid Intersections	108
Stephen Arndt, Benjamin Moseley, Kirk Pruhs, Chaitanya Swamy and Michael Zlatin: Efficiently Coloring the Intersection of General Matroids	111
Lisa Hellerstein, Benedikt M. Plank and Kevin Schewior: Approximating Matroid Basis Testing for Partition Matroids using Budget-In-Expectation . .	115
Session 9A (Wednesday morning)	119
Spyros Angelopoulos, Mathis Degrise, Christoph Dürr and Imrane Sakkour: Randomized online bidding with prediction	119
Samuel McCauley, Benjamin Moseley, Helia Niaparast and Shikha Singh: Stable Matching with Predictions: Robustness and Efficiency under Pruned Preferences	121
Session 9B (Wednesday morning)	126
Jesper Nederlof, Celine Swennenhuis and Karol Wegrzycki: A Subexponential Time Algorithm for Makespan Scheduling of Unit Jobs with Precedence Constraints	126
Klaus Jansen, Felix Ohnesorge and Lis Piroton: A Tight Double-Exponentially Lower Bound for High-Multiplicity Bin Packing	127
Session 10A (Wednesday morning)	131
Samir Khuller, Mozhengfu Liu and Xueyan Tang: To delay or not to delay – Online Span Minimization	131
Ekin Ergen: Online Makespan Scheduling under Two Scenarios	133
Kunal Agrawal, Sanjoy Baruah, Gregory Kehne, Jubayer Nirjhor, Kei Rockwell and Nicole Wein: Scheduling Jobs to Maximize Fractional Value	136

Session 10B (Wednesday morning)	139
Sven Jäger, Alexander Lindermayr and Bart Zondervan: Tight Analysis of Proportional Fairness for Minimizing Weighted Flow Time in Monotone Polytope Scheduling	139
Frits Spieksma and Sten Wessel: Fair Incomplete Round-Robin Tournaments . .	141
Frits Spieksma, Mark de Berg and Andrés López Martínez: The Price of Diversity of the Traveling Salesman Problem	144
Session 13A (Thursday afternoon)	148
Max Hugen, Bob Krekelberg and Alison Hsiang-Hsuan Liu: Online Firefighting on Cactus Graphs	148
Kunal Agrawal, Benjamin Moseley, Heather Newman and Kirk Pruhs: Scheduling Out-Trees Online to Optimize Maximum Flow	150
Session 13B (Thursday afternoon)	154
Debajyoti Kar, Arindam Khan and Malin Rau: Improved Approximation Algorithms for Three-Dimensional Bin Packing	154
Debajyoti Kar, Arindam Khan, Malin Rau, Ann-Brith Strömberg and Albert Vesterlund: A Tight 2-Approximation for Demand Bin Packing	157
Session 14A (Thursday afternoon)	161
Lars Rohwedder and Leander Schnaars: Graph Scheduling with Group Completion Times	161
Thomas Erlebach, Natalia Shakhlevich, Akiyoshi Shioura and Jie Xu: Scheduling Framework for Edge-to-Cloud Task Offloading	164
Thomas Erlebach, Naveen Garg, Sukriti Gupta and Amitabh Trehan: Approximating optimal broadcast of files in a hose-model network	166
Session 14B (Thursday afternoon)	170
Aflatoun Amouzandeh, Klaus Jansen, Lis Piroton, Rob van Stee and Corinna Wambsganz: Online and offline algorithms for weighted makespan minimization	170
Alexander Lindermayr and Morten Weber: Online Weighted Flow Time with Equal-Size Jobs	172
Felix Buld and Andreas S. Schulz: Scheduling with Testing: Competitive Algorithms for Minimizing the Total Weighted Completion Time in the Adversarial Model	175
Session 15A (Friday morning)	180
Vipin Ravindran Vijayalakshmi, Marc Schroder and Tami Tamir: Interval Scheduling Games	180
Etienne Bamas, Shi Li and Lars Rohwedder: Randomized Rounding over Dynamic Programs	182
Session 15B (Friday morning)	186
Ziyad Benomar, Romain Cosson, Alexander Lindermayr and Jens Schlöter: Non-Clairvoyant Scheduling with Progress Bars	186
Anupam Gupta, Haim Kaplan, Alexander Lindermayr, Jens Schlöter* and Sorachai Yingchareonthawornchai: A Little Clairvoyance Is All You Need . . .	188

Session 16A (Friday morning)	192
Evrripidis Bampis, Bruno Escoffier, Dimitris Fotakis, Giorgos Mitropoulos and Michalis Xeferis: What to Predict for Efficient Scheduling on Multiple Machines?	192
Eric Balkanski, Vasilis Gkatzelis and Xizhi Tan: Strategyproof Scheduling with Predictions	195
Eric Balkanski, Jingwei Li*, Clifford Stein and Cherlin Zhu: Speed Predictions for Online Energy-Efficient Scheduling	198
Session 16B (Friday morning)	203
Federico Della Croce and Quentin Schau: Revisiting Johnson’s rule for minimizing makespan in the two-machine flow shop scheduling problem	203
Lin Chen, Yixiong Gao, Minming Li, Guohui Lin and Kai Wang: Revisit the Scheduling Problem with Calibrations	205

Invited talks

Stochastic Load Balancing and Related Problems

Franziska Eberle *

In many classic scheduling results, it is typically assumed that all problem parameters such as the number of machines or a job's processing requirement are known upfront. Often, this assumption is too optimistic, and there are multiple ways how to model uncertainty, online arrival of jobs and/or stochastic processing times being among the most prominent rules. Stochastic scheduling has seen many exciting new results over the last few years, especially when it comes to adaptive policies for minsum objectives.

In this talk, we will focus on two other objective functions: minimizing the expected makespan as well as as minimizing a monotone symmetric norm of the vector of machine loads in expectation. In order to obtain approximation ratios with bounded guarantees, one typically splits jobs into their truncated and their exceptional parts: For the former, strong concentration bounds hold, while the latter behave reasonably well in expectation. We show that this approach also works when comparing a non-adaptive policy to the adaptive optimum with respect to the expected makespan. We obtain a $\mathcal{O}\left(\frac{\log m}{\log \log m}\right)$ -approximation algorithm, which matches the known adaptivity gap.

However, when minimizing an arbitrary monotone symmetric norm $\|\cdot\|$, we need to split each job into up to $\mathcal{O}(\log(\|\mathbf{1}_m\|)) \in \mathcal{O}(\log m)$ many parts, where $\mathbf{1}_m$ is the m -dimensional all-ones vector. Instead of losing an additional factor of $\mathcal{O}(\log m)$ in the approximation ratio, we develop a new counting scheme, which allows us to simultaneously compare the *number* of machines with high load due to “medium” job parts of our non-adaptive policy to that of an adaptive optimum. Combining this with a new concentration bound, we obtain a best-possible $\mathcal{O}\left(\frac{\log m}{\log \log m}\right)$ -approximation algorithm.

Based on joint works with Anupam Gupta, Nicole Megow, Ben Moseley, and Rudy Zhou as well as with Thomas Kesselheim and Rudy Zhou.

*f.eberle@tu-berlin.de. Institute of Mathematics, TU Berlin, Straße des 17. Juni 136, 10623 Berlin, Germany

When Nicolas and Vilfredo Join Hands: Condorcet Dimension and Pareto Optimality for Matchings and Beyond

Jannik Matuschke *

In this talk, we discuss matching problems in which agents form one side of a bipartite graph and have preferences over items on the other side. A central solution concept in this setting is popularity: a matching is a (*weak*) *Condorcet winner* if no other matching is preferred by a strict majority of agents. It is well known, however, that such Condorcet winners need not exist. We first discuss how to find a Condorcet winner if one exists, by exploiting a characterization based on LP duality. For the case that no Condorcet winner exist, we then turn to a natural and prominent relaxation: A set of matchings is a *Condorcet-winning set* if, for every competing matching, a majority of agents prefers their favorite matching in the set over the competitor. Our results reveal that every Pareto-optimal (w.r.t. agent's preferences) set of two matchings is a Condorcet-winning set. This result continues to hold when we impose matroid constraints on the set of matched objects. When agents' preferences are weak rankings, this implies a tight upper bound of 2 on the *Condorcet dimension*, i.e., the size of a smallest Condorcet-winning set, as the necessary Pareto-optimal solutions exist and can be easily computed. Surprisingly, however, for general partial-order preferences, Pareto-optimal matchings may fail to exist—and both determining the Condorcet dimension and deciding the existence of a Pareto-optimal matching are NP-hard problems under such preferences.

This talk is based on the following collaborations:

- Telikepalli Kavitha, Tamás Király, Jannik Matuschke, Ildikó Schlotter, and Ulrike Schmidt-Kraepelin: *The popular assignment problem: when cardinality is more important than popularity* (SODA 2022)
- Telikepalli Kavitha, Jannik Matuschke, and Ulrike Schmidt-Kraepelin: *Condorcet Dimension and Pareto Optimality for Matchings and Beyond* (arXiv:2602.16289)

*jannik.matuschke@kuleuven.be. Research Center for Operations Management, KU Leuven, Belgium.

Practical scheduling problems and where to find them ... at my university

Andreas Wiese (Speaker) *

In this presentation, I will talk about applied scheduling problems at my university and how scheduling theory helped us develop practical algorithms for them. First, we study a problem that arises when a new employee is hired. Our colleagues in the administration need to assign one or more internal positions to the new hire. However, these positions may already be partially assigned to other staff members and additional administrative constraints must be taken into account. This leads to a multi-objective optimization problem that can be modeled as a scheduling problem with machine non-availability periods. Next, we discuss an optimization tool for assigning graders and supervisors to exams. This problem can be modeled as a scheduling problem in which some machines are better suited to process certain jobs, where job preemptions should be minimized, and fairness conditions across machines are desired. Finally, I will show an online tool that we developed to help our students plan their courses. It takes the students' preferences and all rules of their degree program into account and computes a corresponding study plan. This project also has a strong scheduling flavor; for example, the selected lectures must not overlap in time. For all of these projects, I will show how insights and viewpoints from scheduling theory helped us design better practical algorithms.

*andreas.wiese@tum.de. Boltzmannstraße 3. 85748 Garching bei München, Germany.

Contributed talks

Revisit the Online TSP on the Line

Ya-Chun Liang * Jian-Xi Shao † Chung-Shou Liao (Speaker) ‡

1 Introduction

With a rapidly growing attention on the development of learning-augmented algorithms, we revisit the Online Traveling Salesman Problem on the Line (OLTSP) from the perspective of such learning approaches. The objective of the OLTSP is to minimize the completion time of a server that begins at the origin, moves at unit speed to serve all requests arriving online along the real line, and then returns to the origin. Bjelde et al. [2] proposed the optimal deterministic algorithm, achieving a competitive ratio of $(9 + \sqrt{17})/8 \approx 1.640$ for OLTSP. When restricting to zealous algorithms which never allow the server to wait if there are still unserved requests, Blom et al. [3] and Ausiello et al. [1] show matching upper and lower bounds of 1.75 on the competitive ratio for any deterministic zealous algorithms.

The problem is fundamentally motivated by the operational demands of smart warehouses, such as the parcel picking problem where Kiva robots must efficiently navigate linear aisles to fulfill orders. In these real-world environments, request patterns are often not entirely random but follow certain distributions. Traditional worst-case analysis may be overly pessimistic; therefore, we define the OLTSP in a framework where each request σ_i is associated with a release time t_i and a location x_i , but the algorithm may now leverage external “advice” or predictions to overcome the information bottleneck inherent in purely online settings.

2 Algorithm

This study considers how a learning-augmented online algorithm, or simply an online algorithm with predictions, improves its competitive performance for the OLTSP. In particular, we present a specific forecasting strategy, called online predictions, which makes a sequence of predictions one by one in an online manner.

*ycliang@cs.nthu.edu.tw. Department of Computer Science, National Tsing Hua University, Hsinchu 30013, Taiwan.

†shaocharlotte3314@gapp.nthu.edu.tw. Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Hsinchu 30013, Taiwan.

‡csliao@ntu.edu.tw. Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan.

This stands in contrast to predicting the entire information of future requests, such as all their release times or locations, in advance. While offline predictors typically forecast the entire set of requests before an online algorithm begins, the online prediction model proposed here generates a forecast for the next request only upon the arrival of the current one. This sequential approach is often based on the assumption that the total number of requests is not necessarily known a priori, allowing the predictor to adjust its accuracy over time.

To ensure the advice is theoretically and practically applicable, we introduce the concept of “feasible predictions”. A prediction is considered feasible if the server is capable of reaching the predicted location y_i by the predicted release time \hat{t}_i , assuming it moves directly toward the target upon receiving the prediction without serving other requests. This framework for incorporating future information further inspires an exploration into whether randomization could potentially facilitate a better upper bound as a byproduct of the learning-augmented approach.

The presented zealous algorithm, PQRP (PQR with Predictions), integrates a confidence parameter $\lambda \in (0, 1)$ to manage the trade-off between robustness and consistency. Under this framework, the algorithm is characterized by a robustness of $\max\{\frac{6\lambda+1}{4\lambda}, \frac{2\lambda+1}{2\lambda}\}$ and a consistency related to the prediction errors η_t and η_p . While a theoretical lower bound of 1.75 remains for zealous algorithms even with perfect online predictions in specific scenarios, numerical experiments indicate that the proposed framework leads to enhanced efficiency in practical, compared to classical deterministic approaches.

References

- [1] Giorgio Ausiello, Esteban Feuerstein, Stefano Leonardi, Leen Stougie, and Maurizio Talamo. Algorithms for the on-line travelling salesman. *Algorithmica*, 29(4):560–581, 2001.
- [2] Antje Bjelde, Jan Hackfeld, Yann Disser, Christoph Hansknecht, Maarten Lipmann, Julie Meißner, Miriam SchlÖter, Kevin Schewior, and Leen Stougie. Tight bounds for online tsp on the line. *ACM Transactions on Algorithms (TALG)*, 17(1):1–58, 2020.
- [3] Michiel Blom, Sven O Krumke, Willem E de Paepe, and Leen Stougie. The on-line tsp against fair adversaries. *INFORMS Journal on Computing*, 13(2):138–148, 2001.

Power of knowing the full neighborhood of online vertex cover

Anna Hu * Alison Hsiang-Hsuan Liu (Speaker) †

In this work, we study the competitive complexity of the *online vertex cover* problem when the online algorithm has knowledge of the *full neighborhood*, as proposed by Harutyunyan et al. [3].

Competitive complexity of online problems. Given an online algorithm ALG, its competitive ratio $\mathcal{CR}(\text{ALG})$ is defined as $\sup_{\mathcal{I}} \frac{\text{ALG}(\mathcal{I})}{\text{OPT}(\mathcal{I})}$ for any instance \mathcal{I} , where $\text{ALG}(\mathcal{I})$ and $\text{OPT}(\mathcal{I})$ are the cost incurred by the algorithm and the optimal cost, respectively. The competitive ratio of an online problem is then defined as $\inf_{\text{ALG}} \mathcal{CR}(\text{ALG})$ for any possible deterministic online algorithm ALG.

Online vertex cover problem. The (classic) online vertex cover problem is not competitive (that is, any deterministic online algorithm has a competitive ratio that grows as the graph size grows), even when the underlying graph is a tree [2]. The interesting question is, what type of adversary power makes the online algorithm so trapped? There is an area of research on the power of uncertainty. By using different semi-online models, one can consider that some type of uncertainty is restricted in such a model. The improvement of the competitive ratio in the semi-online model against the classic online model can then be seen as the power of the restricted uncertainty. For example, Boyar et al. studied the power of *recourse* [1], and Liu and Toole-Charignon studied the power of *amortized recourse* [4], where the idea of recourse captures how rapidly the structure of the offline optimal solution can change with regard to the size of the input. With recourse and amortized recourse, the online vertex cover problem becomes 2-competitive and $(2 - \frac{2}{\text{OPT}})$ -competitive, respectively.

In this work, we investigate how much a (restricted) extra knowledge influence the competitive complexity of the online vertex cover problem. Specifically, if the adversary lost the power of arbitrarily designing the neighborhood of the revealed vertices, how much the competitive ratio of the online vertex cover problem can be improved?

Vertex arrival knowing the full neighborhood. Given a graph $G = (V, E)$, which is initially *unknown* to the online algorithm. The vertices in V are revealed one-by-one. Once a vertex $v \in V$ is *revealed*, every edge $(v, u) \in E$ and neighbor u is known to the algorithm. If u is not revealed yet, it becomes *visible*. The algorithm then has to make an irrevocable decision if it wants to include v into its vertex cover. Namely, a vertex can only be *accepted* as a member of the vertex cover at the round when it becomes revealed. Afterwards, the vertex cannot be *rejected*. Similarly, once a vertex is

*Department of Information and Computing Sciences, Utrecht University, The Netherlands

†h.h.liu@uu.nl. Department of Information and Computing Sciences, Utrecht University, The Netherlands

rejected in the round when it becomes revealed, it cannot be accepted later. This model restricts the power of the adversary of uncertain neighborhood.

Previous work on online dominating set. This model was first proposed by Hovhannes Harutyunyan et al. [3]. The authors studied the online dominating set problem on this model. While the ordinary online dominating set problem is not competitive even on trees, the authors found that the problem becomes 2-competitive on trees. When the underlying graph G is a cactus graph, the problem becomes 2.5-competitive. The problem is $(t - 1)$ -competitive if G is $K_{1,t}$ -free and $\Theta(\sqrt{\Delta})$ -competitive if G has maximum degree Δ . The authors also showed that the problem remains not competitive on threshold, bipartite planar, and series-parallel graphs.

Our results. We study the online vertex cover problem with full neighborhood knowledge across different graph classes. Interestingly, our results show that the power of designing an arbitrary neighborhood is more powerful in the online vertex cover problem than in the online dominating set problem.

Specifically, the knowledge helps improve the competitive ratio of the online vertex cover problem on trees (DS: 2, VC: 1.5) and cactus graphs (DS: 2.5, VC: 1.666) more than the online dominating set problem. It implies that the power of controlling the uncertain neighborhood is more critical for the adversary in the online vertex cover problem.

Theorem 1 *The online vertex cover problem on trees is 1.5-competitive in the full neighborhood model.*

Theorem 2 *The online vertex cover problem on cactus graphs is $\frac{5}{3}$ -competitive in the full neighborhood model.*

Note that trees have treewidth of 1, and cactus graphs have treewidth of 2. The above results can be interpreted as $(2 - \frac{1}{\text{tw}})$ -competitive, where tw is the treewidth of the graph classes (trees or cactus graphs). We are curious which other graph classes have this property.

Theorem 3 *The online vertex cover problem on outerplanar graphs is at most 2.5-competitive and at least 1.666-competitive in the full neighborhood model.*

Finally, our result shows that on *threshold graphs*, the level of critical of this knowledge of the full neighborhood is much higher: while the dominating set problem remains not competitive even with this knowledge, the vertex cover problem becomes 2-competitive once the adversary is restricted from designing the (unseen) neighborhood with all freedom.

Theorem 4 *The online vertex cover problem on threshold graphs is at most 2-competitive and at least 1.5-competitive in the full neighborhood model.*

References

- [1] J. Boyar, L. M. Favrholdt, M. Kotrbčik, and K. S. Larsen. Relaxing the irrevocability requirement for online graph algorithms. In *WADS 2017*.

- [2] M. Demange and V. T. Paschos. On-line vertex-covering. *Theor. Comput. Sci.*, 2005.
- [3] H. A. Harutyunyan, D. Pankratov, and J. Racicot. Online domination: The value of getting to know all your neighbors. In *MFCS 2021*.
- [4] A. H. Liu and J. Toole-Charignon. The power of amortized recourse for online graph problems. In *WAOA 2022*.

Improved On-line Algorithms for the JRP with Holding & Delay Costs:

Riding the Wave Makes Things Simpler, Stronger, & More General

D. Shmoys (speaker) * V. Suriyanarayana * S. W. Umboh†

The Joint Replenishment Problem (JRP) is a classical inventory management problem that aims to model the trade-off between coordinating orders for multiple commodities (and their ordering cost) with holding costs incurred by meeting demand in advance. In the off-line JRP, there is a collection of demands, each with its own desired service time and holding cost function, as well as a specified demand type. We serve these demands by placing replenishment orders that have an associated fixed cost for placing any order whatsoever, as well as a fixed cost for each type included in the order (independent of the positive quantity that is ordered).

There is a long line of research devoted to approximation algorithms for the JRP and for on-line variants in which the demands arrive over time. The first constant approximation algorithm for the off-line problem is due to Levi, Roundy, & Shmoys [14], who gave a primal-dual 2-approximation algorithm. This was subsequently improved in [15], and the best performance guarantee currently known is a 1.791-approximation algorithm due to Bienkowski et al. [8], based on first solving the natural LP relaxation and rounding the optimal LP solution. Nonner & Souza showed that the JRP is APX-Hard [18], and this was subsequently strengthened by Bienkowski et al. [7], who showed that it is APX-Hard even when each holding cost is either 0 or ∞ .

Buchbinder, Kimbrel, Levi, K. Makarychev, & Sviridenko [10] considered a closely related model in an on-line setting; in their model, demands arrive over time, and rather than having holding costs, there is a delay cost, where the cost incurred is a monotone function of the time that elapses between the arrival time of a demand and when the order is completed. Buchbinder et al. [10] give a 3-competitive algorithm for this setting.

Recently, Moseley, Niaparast and Ravi [16, 17] introduced a natural generalization of the on-line JRP of [10], in which inventory demands arrive over time, and each demand also specifies a desired service time (no earlier than its arrival time) — replenishment orders can be placed earlier than this desired service time (incurring a holding cost) or later (incurring a delay cost). Moseley et al. [16, 17]

*Cornell University. Emails: david.shmoys@cornell.edu, vs478@cornell.edu

†The University of Melbourne, ARC Training Centre in Optimisation Technologies, Integrated Methodologies, and Applications (OPTIMA) (Email: william.umbroh@unimelb.edu.au). Supported by the Australian Government through the Australian Research Council DP240101353.

established that when the holding and delay costs are monotone and uniform across demands, there is a 30-competitive algorithm that employs a greedy strategy and a dual-fitting-based analysis; notably, they left as an open problem whether analogous results could be obtained without the assumption that the cost functions are uniform. Moreover, Moseley et al. [17] showed that their algorithm requires the uniformity assumption: when the delay and holding cost functions are not uniform, then their algorithm's competitive ratio can be linear in the number of requests even for the special case where there is only a single item type, and when the delay and holding cost functions are linear. Thus, additional ideas would be needed to remove the uniformity assumption.

We develop a 5-competitive algorithm that handles arbitrary monotone demand-specific holding and delay cost functions, thereby overcoming the restriction of [16, 17] to uniform cost functions, while simultaneously improving the performance guarantee substantially. We assume only that the cost functions are monotone; it is important to note that without the monotonicity assumption, the problem generalizes the set-cover problem, and hence, even in the off-line setting, only logarithmic performance guarantees can be achieved (unless $\mathcal{P} = \mathcal{NP}$). In fact, for the off-line setting, the algorithm of Bienkowski et al. [8] and many of its predecessors [14, 15] work with no (or minimal) loss in approximation factor when there are both arbitrary monotone holding and delay costs.

It is important to note that the uniformity assumption made by [17] is a significant limitation. For example, when there are only delay costs, the JRP and its various generalizations (e.g., concave JRP [12], multi-level aggregation [5, 6, 9], network design with delay [3, 4], and set cover/aggregation with delay [11, 1]) do not require this uniformity assumption to be amenable to strong competitive analysis results. In addition, our analysis is much simpler than the intricate analysis employed by [17]; this suggests that our results might extend to the more general inventory management settings mentioned above.

For the single-item lot-sizing problem (i.e., the special case of the JRP in which there is only one demand type), the on-line variant with holding and delay costs is already challenging. This is due to the fact that, when we place an order, while it is clear that we should serve all demands whose desired service time has passed, since their delay cost can only increase, it is unclear which of the demands whose service time has not yet passed we should also serve, since their holding cost can decrease in the future. Moseley, Niaparast and Ravi [16, 17] give a 3-competitive algorithm for this special case, again limited to the assumption of uniform cost functions across all demands. We generalize this to arbitrary monotone cost functions, giving a $(\phi + 1)$ -competitive algorithm (where ϕ denotes the golden ratio) for the single-item lot-sizing problem with both holding and delay costs. This significantly narrows the gap to the best-known lower bound of 2; this lower bound stemmed from previous work on TCP acknowledgment [13], which is equivalent to single-item lot-sizing with only delay costs. Although the focus of our work has been in the design of on-line algorithms, we have leveraged work on off-line algorithms to obtain our results. One surprising aspect of our results for the single-item lot-sizing

problem with holding and delay costs is that we give a primal-dual optimization algorithm. One might presume that this means that we have given a complete polyhedral description of the feasible region, but the optimality depends on the cost functions being monotonic.

We adapt the on-line primal-dual algorithm for delay costs of Buchbinder et al. [10] to obtain our 5-competitive algorithm, raising the dual variables in a wave-front manner and placing replenishment orders when unserved demands' dual variables are part of a tight constraint. To handle the difficulty of determining which demands with a desired service time in the future we should add, we rank the demands with desired service times in the future based on when their delay cost would equal their current holding cost and serve those for whom this condition occurs first; this contrasts with the approach of Moseley et al. [16, 17] that ranks demands in ascending order of desired service time. The intuition behind our approach is that we only regret not serving a request early if we end up serving it later when its delay cost is higher than the current holding cost.

This idea of considering the time at which a demand's delay cost would equal its current holding cost is also employed by the concurrent, independent work of Azar and Lewkowicz [2], who call this future timestep the 'virtual deadline' of a demand. However, their approach to determine if/when a replenishment order ought to be placed relies more directly on the delay cost accrued by unserved demands.

References

- [1] Y. Azar, A. Chiplunkar, S. Kutten, and N. Touitou. Set cover with delay - clairvoyance is not required. In F. Grandoni, G. Herman, and P. Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPICs*, pages 8:1–8:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
- [2] Y. Azar and S. Lewkowicz. Online joint replenishment problem with arbitrary holding and backlog costs. In K. Larsen and B. Saha, editors, *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2026, New Orleans, LA, USA, January 11-14, 2026*. SIAM, 2026.
- [3] Y. Azar and N. Touitou. General Framework for Metric Optimization Problems with Delay or with Deadlines . In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 60–71, Los Alamitos, CA, USA, Nov. 2019. IEEE Computer Society.
- [4] Y. Azar and N. Touitou. Beyond tree embeddings - a deterministic framework for network design with deadlines or delay. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1368–1379. IEEE, 2020.
- [5] M. Bienkowski, M. Böhm, J. Byrka, M. Chrobak, C. Dürr, L. Folwarczny, L. Jez, J. Sgall, K. T. Nguyen, and P. Veselý. Online algorithms for multilevel aggregation. *Oper. Res.*, 68(1):214–232, 2020.
- [6] M. Bienkowski, M. Böhm, J. Byrka, M. Chrobak, C. Dürr, L. Folwarczny, L. Jez, J. Sgall, K. T. Nguyen, and P. Veselý. New results on multi-level aggregation. *Theor. Comput. Sci.*, 861:133–143, 2021.

- [7] M. Bienkowski, J. Byrka, M. Chrobak, N. Dobbs, T. Nowicki, M. Sviridenko, G. Świrszcz, and N. E. Young. Approximation algorithms for the joint replenishment problem with deadlines. *Journal of Scheduling*, 18(6):545–560, 2015.
- [8] M. Bienkowski, J. Byrka, M. Chrobak, L. Jez, D. Nogneng, and J. Sgall. Better approximation bounds for the joint replenishment problem. In C. Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 42–54. SIAM, 2014.
- [9] N. Buchbinder, M. Feldman, J. S. Naor, and O. Talmon. $O(\text{depth})$ -competitive algorithm for online multi-level aggregation. In P. N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1235–1244. SIAM, 2017.
- [10] N. Buchbinder, T. Kimbrel, R. Levi, K. Makarychev, and M. Sviridenko. Online make-to-order joint replenishment model: Primal-dual competitive algorithms. *Oper. Res.*, 61(4):1014–1029, 2013.
- [11] R. A. Carrasco, K. Pruhs, C. Stein, and J. Verschae. The online set aggregation problem. In M. A. Bender, M. Farach-Colton, and M. A. Mosteiro, editors, *LATIN 2018: Theoretical Informatics - 13th Latin American Symposium, Buenos Aires, Argentina, April 16-19, 2018, Proceedings*, volume 10807 of *Lecture Notes in Computer Science*, pages 245–259. Springer, 2018.
- [12] R. Chen, J. Khatkar, and S. W. Umboh. Online weighted cardinality joint replenishment problem with delay. In M. Bojanczyk, E. Merelli, and D. P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPICs*, pages 40:1–40:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
- [13] D. R. Dooly, S. A. Goldman, and S. D. Scott. Tcp dynamic acknowledgment delay (extended abstract): theory and practice. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, page 389–398, New York, NY, USA, 1998. Association for Computing Machinery.
- [14] R. Levi, R. Roundy, and D. B. Shmoys. Primal-dual algorithms for deterministic inventory problems. *Math. Oper. Res.*, 31(2):267–284, 2006.
- [15] R. Levi, R. Roundy, D. B. Shmoys, and M. Sviridenko. A constant approximation algorithm for the one-warehouse multiretailer problem. *Manag. Sci.*, 54(4):763–776, 2008.
- [16] B. Moseley, A. Niaparast, and R. Ravi. Putting off the catching up: Online joint replenishment problem with holding and backlog costs, 2024.
- [17] B. Moseley, A. Niaparast, and R. Ravi. Putting off the catching up: Online joint replenishment problem with holding and backlog costs. In Y. Azar and D. Panigrahi, editors, *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, pages 3865–3883. SIAM, 2025.
- [18] T. Nonner and A. Souza. Approximating the joint replenishment problem with deadlines. *Discrete Mathematics, Algorithms and Applications*, 01(02):153–173, 2009.

Rethinking efficiency in real-time schedulability analysis*

Sanjoy Baruah (Speaker) †

Pontus Ekberg ‡

Whereas the algorithms community has traditionally associated computational intractability with NP-hardness, real-time systems has a more relaxed view. In particular, real-time schedulability-analysis algorithms that are designed for use prior to system runtime are considered acceptably efficient if they have pseudo-polynomial running time – examples of widely-used such pseudo-polynomial time algorithms include Response-Time Analysis [3] for fixed-priority scheduling and Processor-Demand Analysis [4, 1] for EDF scheduling of recurrent task systems upon preemptive uniprocessors. More recently, the presence of excellent ILP solvers has led to approaches based on transforming a schedulability analysis problem to a (reasonably small, i.e., polynomial sized) ILP to often also considered acceptably efficient.

Recall that showing a problem to be NP-hard implies it is unlikely to have a polynomial-time solution, and showing it to be strongly NP-hard implies it is unlikely to have a pseudo-polynomial solution. In a similar vein, showing a problem to be hard for the complexity class NP^{NP} (also denoted as Σ_2^{P}) offers strong evidence that it cannot be efficiently solved even with a highly optimized ILP solver¹.

Summarizing the discussion above, notions of efficiency in real-time schedulability analysis have converged upon three beliefs. (i). Algorithms with pseudo-polynomial running times are efficient. (ii). “Polynomial-time + ILP-solver” algorithms – algorithms with polynomial running time that are additionally allowed to make calls to an ILP solver – are also coming to be considered efficient. (iii). Showing a problem to be hard for Σ_2^{P} shows it to be truly intractable.

However, the following recent observation brings the third of these beliefs into question, by showing that there are problems that are hard for Σ_2^{P} (and indeed, even more general complexity classes) that have pseudo-polynomial time solutions.

Observation 1. *If \mathcal{C} is a complexity class contained in EXP and \mathcal{C} is closed under polynomial-time reductions, then there exist \mathcal{C} -complete problems with pseudo-polynomial time solutions. (Here EXP is the complexity class of all decision problems that are solvable using exponential-time algorithms.)* □

A finer-grained view of pseudo-polynomial time. Pseudo-polynomial-time algorithms are considered efficient for real-time scheduling problems because their numerical

*Proofs of the results presented here appear in [2] (and the reference cited there).

†baruah@wustl.edu. Washington University in St. Louis, USA

‡pontus.ekberg@it.uu.se. Uppsala University, Sweden

¹Woeginger [5] explains the implications in this manner: “If you hit a Σ_2^{P} -complete problem, then there is no way of formulating it (in polynomial time) as an integer program (of polynomial size)” and concludes that “ Σ_2^{P} -complete problems are much, much, much, much, much harder than any problem [...] that can be attacked via ILP solvers.”

parameters typically have a direct physical interpretation, most commonly as measures of time, that are unlikely to be too large in practice. When the largest numerical parameter N greatly exceeds the total input size n , the running time of a pseudo-polynomial algorithm is dominated by its dependence on N . Despite this, little attention has been paid in the real-time scheduling literature to how pseudo-polynomial algorithms scale with respect to N , treating all such algorithms as essentially equivalent. This motivates a more fine-grained notion that makes the dependence on N explicit:

Definition 1 (Pseudo- f). *An algorithm's running time is pseudo- f if it is $O(n^k \times f(N))$, where n is the size of the representation of the problem instance, N is its largest numerical parameter value, k is a constant, and f is some function.*

Thus an algorithm's running time is pseudo-polynomial if and only if it is pseudo- f for some polynomial f . We will say that an algorithm with pseudo- f running time is *pseudo-linear* if $f(N) \in O(N)$, *pseudo-quadratic* if $f(N) \in O(N^2)$, etc. The result below states that any difference in polynomial degree between f and g differentiates pseudo- f from pseudo- g :

Theorem 2. *For all $a > b > 0$, there are problems that can be solved in time $O(\text{poly}(n) \times N^a)$, but not in time $O(\text{poly}(n) \times N^b)$, where n is the size of the input and N is the value of its largest numerical parameter.* \square

Scale invariance. In addition to distinguishing pseudo-polynomial-time algorithms by their polynomial dependence on numerical input values, we also wish to distinguish them by whether their running time is invariant under uniform scaling of those values. Ideally, a pseudo-polynomial algorithm should become slower only when the relationships among numerical parameters become more complex, not when all values are scaled by a common factor. In scheduling problems where numerical parameters often represent time, this corresponds to insensitivity to the choice of time unit: an algorithm should have the same running time whether times are expressed in milliseconds, microseconds, or any other unit, provided the instance structure is unchanged and all values remain integral. The following definition formalizes this notion of scale invariance for pseudo-polynomial-time algorithms.

Definition 3 (Scale-invariant pseudo-polynomial time). *An algorithm is scale-invariant pseudo-polynomial if it runs in time that is polynomial in n and in N/G , where n is the size of the input, N is its largest numerical parameter, and G is the greatest common divisor of all its numerical parameters*

We note that it seems useful to retain some flexibility regarding which numerical parameters to include in the computation of the greatest common divisor G in Definition 3. For instance, in a multiprocessor scheduling problem with a multitude of numerical parameters denoting time (deadlines, periods, execution times) and a single numerical parameter denoting the number of processors, it may make sense to include only the parameters that share a unit (i.e., the parameters representing time), but not the processor count, when calculating G .

Not all pseudo-polynomial time algorithms are scale invariant, and in the following we see that this differentiates computational problems as well.

Theorem 4. *There are problems that can be solved in pseudo-polynomial time, but not in scale-invariant pseudo-polynomial time.* \square

References

- [1] S. Baruah, A. Mok, and L. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Proceedings of the 11th Real-Time Systems Symposium*, pages 182–190, Orlando, Florida, 1990. IEEE Computer Society Press.
- [2] Sanjoy Baruah and Pontus Ekberg. A closer look at pseudo-polynomial time and its use in real-time scheduling theory. In Susanne Graf, Paul Pettersson, and Bernhard Steffen, editors, *Real Time and Such: Essays Dedicated to Wang Yi to Celebrate His Scientific Career*, pages 120–134. Springer Nature Switzerland, Cham, 2025.
- [3] M. Joseph and P. Pandya. Finding response times in a real-time system. *The Computer Journal*, 29(5):390–395, October 1986.
- [4] Joseph Y.-T Leung and M. Merrill. A note on the preemptive scheduling of periodic, real-time tasks. *Information Processing Letters*, 11:115–118, 1980.
- [5] Gerhard J Woeginger. The trouble with the second quantifier. *4OR: A Quarterly Journal of Operations Research*, 19(2):157–181, 2021.

Efficient explainability of schedulability analysis

Sanjoy Baruah *

Pontus Ekberg (Speaker) †

Even after a successful analysis establishing a system’s schedulability (e.g., that all deadlines are always met in a safety-critical real-time system), one may still have to prove this fact to a third party, such as a Certification Authority (CA). The third party may mistrust the veracity of the analysis and may be unable or unwilling to carry out the costly analysis on their own. We would like to be able to convince them instead via a (relatively) simple and trustworthy proof. Ideally, such proofs are formal and machine-checkable, such as the formally verifiable response-time certificates produced by the POET tool [6], which use the PROSA framework [3] on top of Coq/Rocq.

One of the fundamental challenges to overcome in such a setup is that not all computational problems are amenable to efficiently verifiable proofs. Indeed, if we were to define “efficient” as polynomial time, and wanted proofs to be static certificates that can be handed over to a third party for separate verification without further interaction, then the computational problems amenable to this would be exactly those in complexity class NP. Unfortunately, most interesting schedulability problems for real-time systems are not in NP. It is, of course, possible to create verifiable proofs also for problems outside NP, but such proofs would not be both polynomial in size and checkable in polynomial time. In the SAT community, it is common for SAT solvers to be able to produce verifiable proofs of *unsatisfiability*. But since UNSAT is (probably) not in NP, such proofs cannot be small in general, and indeed it has been reported that solvers have created UNSAT proofs greater than 100 GB in size even for benchmark problems used in SAT solver competitions¹. In an interesting use of such UNSAT certificates to settle a long-standing number-theoretic problem [5], the produced proof was reported to be 2 PB in size and to take over 15 CPU-years to verify. Though seemingly much less used than in SAT solving, similar certificates also exist for mixed-integer linear programming: VIPR [4] is a certificate format that supports verifiable proofs of MIP infeasibility and optimality.

In the work outlined here, we try to sidestep inherent scalability challenges in verifying solutions (and also aim to address problems beyond coNP where UNSAT and MIP infeasibility is found) by outlining principles to carve out efficiently verifiable subsets of a problem (outside NP) that we are interested in solving. In essence

*baruah@wustl.edu. Washington University in St. Louis, USA

†pontus.ekberg@it.uu.se. Uppsala University, Sweden

¹See <https://satcompetition.github.io/2023/certificates.html>

this is done by identifying special cases that are in NP or are otherwise amenable to some other type of efficient verification. Carving out an NP subproblem from a larger non-NP problem is not much different from carving out a polynomial time subproblem from a larger non-polynomial time problem, a very common practice in most fields of algorithms. But, at least from our experience in real-time scheduling, identifying NP subproblems is a much more unusual take on the same basic principle and may require thinking about solutions slightly differently. Indeed, it may well happen that producing certificates to an NP subproblem is significantly more computationally demanding than just solving the original problem, but the benefit we are looking for is the easily verifiable certificate and not reducing the initial solution time. In [2] we present a small proof of principle by identifying some simple NP subproblems of the coNP-complete EDF-schedulability problem.

Further, we lift the definition of efficient approximation schemes from solving to verifying problems. As an example, in [2] we extended a well-known FPTAS for EDF-schedulability [1] to take also a certificate as input, which (disregarding the effort to find the certificate) can make it more efficient, exponentially so for some instances. We dubbed the resulting type of approximation scheme FPTVAS, for fully polynomial-time verification approximation scheme and defined it as follows.²

Definition 1 (FPTVAS) *A fully polynomial-time **verification** approximation scheme (FPTVAS) for a schedulability analysis problem is an algorithm that, given as input an instance Γ , a parameter $\delta > 0$, **and a certificate**, returns “unschedulable” if Γ is unschedulable on a speed-1 processor, and returns “schedulable” if Γ is schedulable on a speed- $(1/(1 + \delta))$ processor. Its running time, and the certificate size, is bounded by a polynomial in the two parameters $|\Gamma|$ and $\frac{1}{\delta}$.*

FPTVASs can not only have improved efficacy compared to FPTASs, but there also exist problems that have an FPTVAS that provably have no FPTAS (assuming $P \neq NP$). An example of this is the partitioned multiprocessor EDF-schedulability problem for which the FPTVAS mentioned above is easily adapted, but for which no FPTAS exists as it trivially generalizes bin packing.

Finally, recognizing that pseudo-polynomial time (and not polynomial time) is the typical gold standard of efficiency in real-time scheduling, we observe that it makes sense to aim also for pseudo-polynomial time verification. We are not aware of any previous work directly focusing on this concept, but a *nondeterministic pseudo-polynomial time* (pseudoNP) class can readily be defined. While a very straightforward concept, it seems meaningful as pseudoNP contains real-world problems that are neither in NP nor can be solved in (deterministic) pseudo-polynomial time. An example of this is again the (bounded-utilization) partitioned multiprocessor EDF-schedulability problem.

²We recognize that Definition 1 may seem a bit clunky to the approximation algorithms community. It mirrors the way FPTASs are usually thought of in real-time scheduling theory: as δ -gap promise problems that are always safe (no false positives) and in which processor speed is the augmented resource. This definition can certainly be generalized to fit other settings.

The astute reader will have noticed that we have failed to mention interactive proofs, perhaps the most elegant computer science concept targeting verification by an agent with bounded resources. It may well turn out that this is the best approach also in our imagined use cases, but as it stands, it is precisely the interaction that we would like to avoid by working on simpler first principles.

References

- [1] K. Albers and F. Slomka. An event stream driven approximation for the analysis of real-time systems. In *Proceedings of the 16th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 187–195, June 2004.
- [2] Sanjoy Baruah and Pontus Ekberg. Towards efficient explainability of schedulability properties in real-time systems. In *Proceedings of the 35th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 2:1–2:20, 2023.
- [3] Felipe Cerqueira, Felix Stutz, and Björn B. Brandenburg. PROSA: A case for readable mechanized schedulability analysis. In *Proceedings of the 28th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 273–284, 2016.
- [4] Kevin K. H. Cheung, Ambros Gleixner, and Daniel E. Steffy. Verifying integer programming results. In *Integer Programming and Combinatorial Optimization*, pages 148–160, Cham, 2017. Springer International Publishing.
- [5] Marijn J. H. Heule. Schur number five. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, AAAI’18*. AAAI Press, 2018.
- [6] Marco Maida, Sergey Bozhko, and Björn B. Brandenburg. Foundational Response-Time Analysis as Explainable Evidence of Timeliness. In *Proceedings of the 34th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 19:1–19:25, 2022.

Optimal Priority Assignment for Synchronous Harmonic Tasks With Dynamic Self-Suspension

Mario Günzel ^{*} Marion Sudvarg [†] Max Deppert [‡] Ao Li [§]
 Ning Zhang [¶] Jian-Jia Chen ^{||}

1 Introduction

We consider periodic real-time tasks with *dynamic self-suspension*, where a task τ_i is assigned a maximum total self-suspension time parameter S_i . Specifically, under the dynamic self-suspension model, a task τ_i may suspend itself at any moment before it finishes and infinitely often as long as its total suspension time does not exceed S_i . This model can be found in many real-world applications. For example, many real-time autonomous automotive and aerial systems include real-time perception pipelines with tasks that self-suspend, e.g., while offloading computation to GPUs or other accelerators.

For tasks with self-suspension, the literature has focused on sporadic task systems, where each task has a minimum inter-arrival time between two consecutive job releases. There are only four dedicated results for periodic task models. To the best of our knowledge, there is not a single result on exact schedulability tests or optimal priority assignments of dynamic self-suspending tasks under preemptive Task-level Fixed-Priority (T-FP) scheduling.

2 Our Results

We explore optimal preemptive T-FP scheduling algorithms to deal with dynamic self-suspension. To that end, we focus on periodic real-time task systems, to step away from the obstacles of sporadic arrivals of real-time jobs. We make the following contributions:

- We show that the dynamics of self-suspension can be handled in polynomial time by Suspension-Aware Deadline-Monotonic (SADM) priority assignment when the periodic tasks have the same period and release their first jobs synchronously (namely, *frame-based* real-time tasks). This holds for both implicit-deadline and constrained-deadline task systems.

^{*}mario.guenzel@tu-dortmund.de Department of Computer Science, TU Dortmund University, Germany

[†]msudvarg@wustl.edu Washington University in St. Louis, USA

[‡]made@cs.uni-kiel.de Kiel University, Germany

[§]ao@wustl.edu Washington University in St. Louis, USA

[¶]zhang.ning@wustl.edu Washington University in St. Louis, USA

^{||}jian-jia.chen@tu-dortmund.de Department of Computer Science, TU Dortmund University, Germany

	ordinary (no suspension)			dynamic self-suspension			segmented self-suspension
Release Model	Implicit	Constrained	Arbitrary	Implicit	Constrained	Arbitrary	
synchronous, frame-based	any	DM	DM	SADM (poly.-time, our result)		no analysis or optimization known	\mathcal{NP} -hard in the strong sense [5]
synchronous, periodic, harmonic	RM [10]	DM [11]	OPA [1] (expo.-time exists)	OPA (poly.-time, our result)			
synchronous periodic	RM [11, 12]	DM [11]	OPA [1] (expo.-time exists)	no analysis or optimization known			
asynchronous periodic	strongly \mathcal{NP} -hard [2, 8, 11] ¹						
sporadic	the same as <i>synchronous periodic</i>			non-existence of bounded speedup factors for T-FP if suspension time cannot be sped up [4], unknown computational complexity [6]			

Table 1: The scheduler design problem under T-FP.

- For synchronous periodic tasks with harmonic periods and dynamic self-suspension, we show that SADM is no longer optimal even for implicit-deadline tasks and develop a polynomial-time algorithm based on Audsley’s approach [1].
- We note that our positive results for frame-based tasks and harmonic tasks are based on exact schedulability tests dedicated for these scenarios. Any extension should be done with care.

Consider a set Γ of periodic tasks, where T_i is the period of task τ_i and D_i is the relative deadline of τ_i . We say that Γ is an *implicit-deadline* task set if $D_i = T_i, \forall \tau_i \in \Gamma$, a *constrained-deadline* task set if $D_i \leq T_i, \forall \tau_i \in \Gamma$, or an *arbitrary-deadline* task set, otherwise. Table 1 and Table 2 summarize the results in the literature as well as our findings in our paper [9] (marked in blue) for self-suspending tasks regarding the scheduler design problem and the (exact) schedulability test problem.

References

- [1] N. C. Audsley. Optimal priority assignment and feasibility of static priority tasks with arbitrary start times. Technical Report YCS-164, Department of Computer Science, University of York, 1991.
- [2] S. K. Baruah, R. R. Howell, and L. Rosier. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Systems*, 2:301–324, 1990.
- [3] V. Bonifaci, A. Marchetti-Spaccamela, N. Megow, and A. Wiese. Polynomial-time exact schedulability tests for harmonic real-time tasks. In *IEEE 34th Real-Time Systems Symposium. RTSS*. pages 236–245. 2013.

Release Model	ordinary (no suspension)			dynamic self-suspension		
	Implicit	Constrained	Arbitrary	Implicit	Constrained	Arbitrary
synchronous, frame-based	poly.-time			poly.-time (our paper)		no analysis or optimization known
synchronous, periodic, harmonic	poly.-time [10]	poly.-time [3, 13]	expo.-time exists	poly.-time (our paper)		
synchronous periodic	weakly \mathcal{NP} -complete [7, 8] (pseudo-poly.-time exists)	weakly \mathcal{NP} -complete [7, 8] (pseudo-poly.-time exists)	weakly \mathcal{NP} -hard [7, 8] (expo.-time exists)	no exact schedulability test known, at least as difficult as the corresponding task model without self-suspension		
asynchronous periodic	weakly \mathcal{NP} -hard and strongly co- \mathcal{NP} -hard [2, 8, 11]					
sporadic	the same as <i>synchronous periodic</i>					

Table 2: Schedulability test for ordinary and self-suspending tasks under T-FP.

- [4] J.-J. Chen. Computational complexity and speedup factors analyses for self-suspending tasks. In *Real-Time Systems Symposium (RTSS)*, pages 327–338, 2016.
- [5] J.-J. Chen, T. Hahn, R. Hoeksma, N. Megow, and G. von der Brüggen. Scheduling self-suspending tasks: New and old results. In S. Quinton, editor, *31st Euromicro Conference on Real-Time Systems, ECRTS*, volume 133, pages 16:1–16:23, 2019.
- [6] J.-J. Chen, G. Nelissen, W.-H. Huang, M. Yang, B. Brandenburg, K. Bletsas, C. Liu, P. Richard, F. Ridouard, Neil, Audsley, R. Rajkumar, D. de Niz, and G. von der Brüggen. Many suspensions, many problems: a review of self-suspending tasks in real-time systems. *Real Time Syst.*, 55(1):144–207, 2019.
- [7] P. Ekberg and W. Yi. Fixed-priority schedulability of sporadic tasks on uniprocessors is np-hard. In *Real-Time Systems Symposium, RTSS*, pages 139–146, 2017.
- [8] P. Ekberg and W. Yi. *Complexity of Uniprocessor Scheduling Analysis*, pages 1–18. Springer Singapore, Singapore, 2022.
- [9] M. Günzel, M. Sudvarg, M. A. Deppert, A. Li, N. Zhang, and J. Chen. Optimal priority assignment for synchronous harmonic tasks with dynamic self-suspension. In *Real-Time and Embedded Technology and Applications Symposium, RTAS*, pages 40–53, 2025.
- [10] T.-W. Kuo and A. K. Mok. Load adjustment in adaptive real-time systems. In *IEEE Real-Time Systems Symposium*, pages 160–171, 1991.
- [11] J. Y.-T. Leung and J. Whitehead. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Perform. Eval.*, 2(4):237–250, 1982.
- [12] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.
- [13] T. H. C. Nguyen, W. Grass, and K. Jansen. Exact polynomial time algorithm for the response time analysis of harmonic tasks. In *International Workshop on Combinatorial Algorithms (IWOCA)*. volume 13270. pages 451–465. 2022.

Flow Shop Scheduling with Stochastic Reentry

Maximilian von Aspern (Speaker) * Felix Buld * Michael Pinedo †

1 Introduction

We consider a natural model of flow shop scheduling motivated by the manufacturing of semiconductor computer chips. Such computer chips consist of dozens of individual layers that must be applied in series, and all require several production steps on specialized machinery. Similarly, the jobs in our scheduling problem must complete multiple loops through the flow shop, each requiring a visit to every machine in the correct order. Each job is described by the number of loops it requires, which we assume are independent random variables capturing uncertainty in the production process. We assume that each operation of each loop requires unit processing time on every machine. In three-field notation, our problem can be denoted as $F|nrnr, p_{ijk} = 1, Y_j \sim \text{stoch}|E[\gamma]$, where $\gamma \in \{C_{\max}, \sum_j C_j, \sum_j w_j C_j\}$ is the objective function which is to be minimized in expectation.

Reentrant flow shops were first considered by Graves, Meal, Stefek, and Zeghmi [6], and since then, a number of other authors have studied the problem, for example, by analyzing different reentry patterns. We refer the reader to Emmons and Vairaktarakis [3] for an overview. The reentry pattern considered in this work is the full loop pattern, in which each job must pass through every machine in each loop. Most relevant to our work are the studies by Yu and Pinedo [11] and von Aspern, Buld, Klein, and Pinedo [1], which develop optimal priority rules for the deterministic variant of our problem.

In this work, we present a reduction to a classical scheduling problem on identical parallel machines, which provides deep insights into the problem described above as well as the structure of its (near-) optimal solutions. In particular, this reduction immediately yields optimal and approximately optimal policies under certain assumptions, by drawing on earlier results by Weber [10], Glazebrook [5], and Jäger and Skutella [8]. We hope that our insights can help derive approximation or optimality guarantees for simple policies under further random distributions that better capture real-world uncertainties.

*{maximilian.aspern@tum.de, felix.buld@tum.de}. School of Computation, Information and Technology & School of Management, Technical University of Munich, Arcisstraße 21, 80333 München, Germany.

†mlp5@stern.nyu.edu. Leonard N. Stern School of Business, New York University, 44 West Fourth Street, New York, NY 10012, USA.

2 Model

There are n jobs to be processed on an m -machine flow shop with reentry. Each job $j \in [n]$ must complete Y_j loops, where Y_j is randomly distributed. The processing time of each loop $k \in [Y_j]$ of job $j \in [n]$ on machine $i \in [m]$ is 1, i.e., $p_{ijk} = 1$. Each machine can process at most one job at a time, and each job can be processed by at most one machine simultaneously. Crucially, the loops of each job must be completed in series, and each loop must pass first through machine 1, then machine 2, etc. We denote an instance of this problem as I .

Under the unit-time assumption, each loop can be processed contiguously, and the only decision to be made is which job to start on machine 1 at each time. A solution to this problem is a scheduling policy, which maps all possible states of the dynamic system to a scheduling decision. While such a policy can be exponentially large in general, we are interested in polynomial-size policies that are approximately optimal within the class of non-anticipatory policies, i.e., those that do not know the realizations of random variables before the corresponding job is completed. Of those, priority policies – which consist of an ordering of all jobs and always schedule an available job with the highest priority – are of particular interest. We call a policy π an α -approximation if $\mathbb{E}[\gamma(\pi)] \leq \alpha \mathbb{E}[\gamma(OPT)]$.

3 Reduction

While the contiguity of the individual job loops appears to make the problem more tractable in general, it also seems to make the problem resistant to the inductive proof strategies that have been used to establish classical optimality results for parallel machine scheduling problems [2, 4, 7, 9] because jobs cannot be interrupted and succeeded by arbitrary jobs at arbitrary times. In particular, a new loop must always start on machine 1. To circumvent this difficulty, we construct a set of auxiliary problem instances on identical parallel machines. As part of the construction, we show that a policy π for our flow shop problem induces policies π_k for the auxiliary problems, preserving completion times. Consequently, we can derive sufficient conditions (on the induced policies) for establishing approximation guarantees for the flow shop policy.

Our set of auxiliary problems consists of m individual instances I_k of the problem $P|m(t) \nearrow, X_j \sim \text{stoch} | \mathbb{E}[\gamma]$, where $m(t) \nearrow$ denotes that the number of available machines is non-decreasing over time. Specifically, I_k contains n jobs with random processing times $X_j = Y_j$ and the number of machines is defined by $m_k(0) = m + 1 - k$ and $m_k(t) = m$ for $t \in \mathbb{N}_{>0}$. W.l.o.g., we may assume that machines $1, 2, \dots, m + 1 - k$ are available at time 0, while the remaining machines become available at time 1. The construction is illustrated in Figure 1.

Theorem 1 *A priority policy π is an α -approximation for I if for all $k \in [m]$, π_k is an α -approximation for auxiliary problem I_k .*

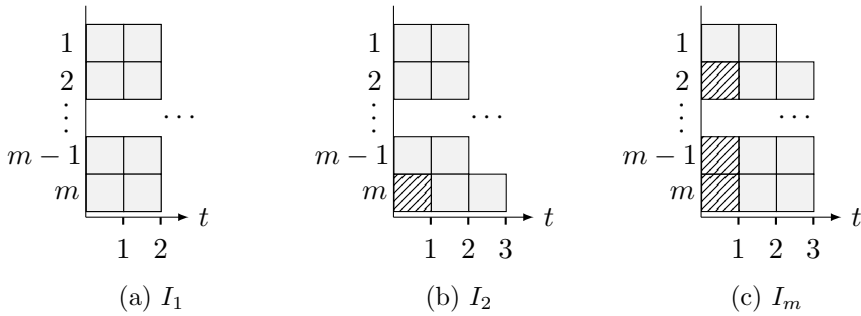


Figure 1: Illustration of machine unavailabilities in the auxiliary problems (shaded).

4 Results

We exemplify the potency of our approach by presenting optimality and approximation guarantees for three simple priority policies defined below. While these results can all be shown to hold in more general settings, we will only present them in the case of geometrically distributed $Y_j \sim G(q_j)$ for the sake of brevity.

Let us define the priority policies MERL (“most expected remaining loops first”), LERL (“least expected remaining loops first”), and WLERL (“weighted least expected remaining loops first”) as the policies that sort the jobs by increasing q_j , decreasing q_j , and decreasing $w_j q_j$, respectively. Here, q_j is the parameter of the geometric distribution of Y_j , and it holds that $\mathbb{E}[Y_j] = 1/q_j$.

Theorem 2 *MERL is optimal for $\gamma = C_{\max}$ and LERL is optimal for $\gamma = \sum_j C_j$.*

Theorem 3 *WLERL is a $\sqrt{2}$ -approximation for $\gamma = \sum_j w_j C_j$.*

References

- [1] M. VON ASPERN, F. BULD, N. KLEIN, AND M. PINEDO *Flow Shops with Reentry: The Total Weighted Completion Time Objective*. Journal of Scheduling, 2025.
- [2] J. BRUNO, P. DOWNEY, AND G.N. FREDERICKSON *Sequencing Tasks with Exponential Service Times to Minimize the Expected Flow Time or Makespan*. Journal of the ACM, 1981.
- [3] H. EMMONS AND G. VAIRAKTARAKIS *Flow Shop Scheduling: Theoretical Results, Algorithms, and Applications*. Springer, 2013.
- [4] J.C. GITTINS *Multiserver Scheduling of Jobs with Increasing Completion Rates*. Journal of Applied Probability, 1981.
- [5] K.D. GLAZEBROOK *Scheduling Tasks with Exponential Service Times on Parallel Processors*. Journal of Applied Probability, 1979.

- [6] S.C. GRAVES, H.C. MEAL, D. STEFEK, AND A.H.ZEGHMI *Scheduling of Re-entrant Flow Shops*. Journal of Operations Management, 1983.
- [7] L. VAN DER HEYDEN *Scheduling Jobs with Exponential Processing and Arrival Times on Identical Processors so as to Minimize the Expected Makespan*. Mathematics of Operations Research, 1981.
- [8] S. JÄGER AND M. SKUTELLA *Generalizing the Kawaguchi-Kyan Bound to Stochastic Parallel Machine Scheduling*. Leibniz International Proceedings in Informatics, Schloss Dagstuhl, 2018.
- [9] M. PINEDO AND G. WEISS *Scheduling of Stochastic Tasks on Two Parallel Processors*. Naval Research Logistics Quarterly, 1979.
- [10] R.R. WEBER *Optimal Organization Of Multi-Server Systems*. PhD thesis, University of Cambridge, 1979.
- [11] T.S. YU AND M. PINEDO *Flow Shops with Reentry: Reversibility Properties and Makespan Optimal Schedules*. European Journal of Operations Research, 2020.

A Novel IP Formulation for Stochastic Non-Preemptive Scheduling

Sven Jäger* Daniel Schmidt genannt Waldschmidt (Speaker)*

1 Introduction

The scheduling problem of minimizing the sum of weighted completion times is a widely studied challenge in combinatorial optimization that originated exactly 70 years ago with the publication of the first result. Jobs $1, \dots, n$ with positive integral processing times $p_1, \dots, p_n \in \mathbb{N}$ and non-negative weights $w_1, \dots, w_n \in \mathbb{Q}_{\geq 0}$ have to be processed non-preemptively on m parallel identical machines so as to minimize the sum of weighted completion times. In the deterministic setting the problem is well understood [13, 2, 10, 12]. As in reality not all data is certain, it is natural to consider the generalization in which the processing times are not deterministic. In the very pessimistic non-clairvoyant setting in which no information of the processing times is known and only revealed upon completion there can be no constant competitive strategy already in the unweighted single-machine case, even when we allow for randomization [5]. A less pessimistic and more realistic assumption on the instance is that the processing times are modeled as independent random variables $\mathbf{P} = (P_1, \dots, P_n)$ defined on a sample set Ω . In this work we restrict ourselves to positive integral random variables following finite probability distributions. When embedding this stochastic scheduling problem into a dynamic stochastic optimization framework, we must specify an action for every given state that leads to a next state depending on the realization. The resulting solution concept is called a dynamic policy that for any realization induces a schedule in the deterministic sense. The goal is to compute a dynamic policy that minimizes the expected sum of weighted completion times. For more details, we refer to [7]. On a single machine, this stochastic scheduling problem is optimally solved by the WSEPT strategy that processes the jobs in non-increasing order of $w_j/\mathbb{E}[P_j]$ [9]. However, when we consider two or more machines, the problem is to date not well understood. There is various work for upper bounds on $\mathbb{P} \|\mathbb{E}[\sum w_j C_j]$ that either depend on the squared coefficient of variation [8, 11, 6] or on the number of jobs and machines [4, 3]. Moreover, there is a PTAS for a constant number of types of Bernoulli jobs [1]. On the other hand, there are no trivial inapproximability results

*{jaeger,dschmidt}@math.tu-berlin.de Institute of Mathematics, Technische Universität Berlin, Straße des 17. Juni 136, 10623 Berlin, Germany.

known for this problem. The crucial challenge for showing improved performance guarantees is to find good lower bounds on the cost of an optimal policy, which is the starting point of our work.

2 Integer Program

We introduce a new integer linear program that merges the time-indexed formulation from the deterministic setting [14] with the realization-dependent point of view inherent to stochastic problems. However, this is not sufficient to obtain an exact formulation as the schedules for different realizations can be defined independently and hence, the realization-wise optimal schedule would be feasible. In other words, the non-anticipatory character of a policy is not reflected in the integer program. Our contribution is to formulate the non-anticipativity condition as a linear constraint that yields an exact formulation.

Theorem 1 *Let $\mathcal{T} = \{0, \dots, \max_{\mathbf{p} \in \mathbf{P}(\Omega)} \sum_{j=1}^n p_j\}$. The set of policies starting jobs only at times $t \in \mathcal{T}$ is in one-to-one correspondence to the set of integer solutions in the polytope \mathcal{P} given by*

$$\begin{aligned} \sum_{t \in \mathcal{T}} x_{jt}(\mathbf{p}) &= 1 && \forall j \in [n], \mathbf{p} \in \mathbf{P}(\Omega) \\ \sum_{j \in [n]} \sum_{s=t-p_j+1}^t x_{js}(\mathbf{p}) &\leq m && \forall t \in \mathcal{T}, \mathbf{p} \in \mathbf{P}(\Omega) \\ x_{jt}(\mathbf{p}) - x_{jt}(\tilde{\mathbf{p}}) &\leq \sum_{\substack{k: p_k \neq \tilde{p}_k \\ k \neq j}}^{t - \min(p_k, \tilde{p}_k)} \sum_{s=0}^{t - \min(p_k, \tilde{p}_k)} x_{ks}(\mathbf{p}) && \forall j \in [n], t \in \mathcal{T}, \mathbf{p}, \tilde{\mathbf{p}} \in \mathbf{P}(\Omega) \\ x_{jt}(\mathbf{p}) &\geq 0 && \forall j \in [n], t \in \mathcal{T}, \mathbf{p} \in \mathbf{P}(\Omega). \end{aligned}$$

The formulation can also be generalized to the setting of unrelated machines, non-trivial release dates, precedence constraints, and more. One obvious downside is that its size is exponential in the size of the input. However, its relaxation by dropping the integrality conditions has the potential to be used as a lower bound on the cost of an optimal policy.

3 LP Relaxation

Let LP denote the relaxation of IP in which we replace the integrality constraints by non-negativity constraints. To the best of our knowledge, the best linear programming relaxation of $\mathbf{P} \parallel \mathbb{E}[\sum w_j C_j]$ in terms of integrality gap so far is the completion time formulation based on the work by Möhring, Schulz, and Uetz [8]. Its optimal value on a single machine coincides with the cost of the optimal policy computed by WSEPT. We show that our formulation is at least as strong as the one in [8].

Theorem 2 *Let $\mathbf{x} \in \mathcal{P}$ be an arbitrary (fractional) solution. Then*

$$C_j := \sum_{\mathbf{p} \in \mathcal{P}(\Omega)} \mathbb{P}(\mathbf{p}) \left[\left(\sum_{t \in \mathcal{T}} t x_{jt}(\mathbf{p}) \right) + p_j \right], \quad j \in [n],$$

defines a feasible solution to the completion time formulation of [8].

As a consequence, the linear program with feasible set \mathcal{P} can also be shown to have no integrality gap for $1 \parallel \mathbb{E}[\sum w_j C_j]$. We also study modifications of our formulation and obtain further results.

References

- [1] A. ANTONIADIS, R. HOEKSMAS, K. SCHEWIOR, AND M. UETZ (2025). *Stochastic scheduling with Bernoulli-type jobs through policy stratification*. To appear in Proc. of FOCS.
- [2] J. BRUNO, E.G. COFFMAN JR., AND R. SETHI (1974). *Scheduling independent tasks to reduce mean finishing time*. J. ACM, 17(7):382–387.
- [3] A. GUPTA, B. MOSELEY, AND R. ZHOU (2023). *Minimizing Completion Times for Stochastic Jobs via Batched Free Times*. Proc. of SODA, 1905–1930.
- [4] S. IM, B. MOSELEY, AND K. PRUHS (2015). *Stochastic Scheduling of Heavy-tailed Jobs*. Proc. of STACS, 474–486.
- [5] S. JÄGER, G. SAGNOL, D. SCHMIDT GENANNT WALDSCHMIDT, AND P. WARODE (2025). *Competitive Kill-and-Restart and Preemptive Strategies for Non-Clairvoyant Scheduling*. Math. Program., 210(1):457–509.
- [6] S. JÄGER AND M. SKUTELLA (2018). *Generalizing the Kawaguchi-Kyan Bound to Stochastic Parallel Machine Scheduling*. Proc. of STACS, 43:1–43:14.
- [7] R.H. MÖHRING, F.J. RADERMACHER, AND G. WEISS (1984). *Stochastic scheduling problems I - General strategies*. Z. Oper. Res., 28(7):193–260.
- [8] R.H. MÖHRING, A.S. SCHULZ, AND M. UETZ (1999). *Approximation in stochastic scheduling: the power of LP-based priority policies*. J. ACM, 46(6):924–942.
- [9] M.H. ROTHKOPF (1966). *Scheduling with Random Service Times*. Manag. Sci., 12(9):707–713.
- [10] S.K. SAHNI (1976). *Algorithms for scheduling independent tasks*. J. ACM, 23(1):116–127.
- [11] M. SKUTELLA, M. SVIRIDENKO, AND M. UETZ (2016). *Unrelated Machine Scheduling with Stochastic Processing Times*. Math. Oper. Res., 41(3):851–864.
- [12] M. SKUTELLA AND G.J. WOEGINGER (2000). *A PTAS for Minimizing the Total Weighted Completion Time on Identical Parallel Machines*. Math. Oper. Res., 25(1):63–75.
- [13] W.E. SMITH (1956). *Various optimizers for single-stage production*. Nav. Res. Logist. Q., 3(1-2):59–66.
- [14] J.P. SOUSA AND L.A. WOLSEY (1992). *A time indexed formulation of non-preemptive single machine scheduling problems*. Math. Program., 54:353–367.

Approximation algorithms for graph search problems with imperfect detection

Martijn van Ee (Speaker) * René Sitters †

1 Introduction

Imperfect detection is a common feature in search theory [2], and is generally motivated by inaccuracies in sensor measurements or the inability to perfectly sweep an area. In search problems, one is usually given a set of cells, and the goal is to design a search strategy that minimizes the expected time until the target has been found. Alternatively, a common objective is the probability of finding the target, given a certain deadline. However, travel times are usually ignored in this field. On the other hand, searching with travel times is well studied in graph search problems, such as the traveling repairman problem [3] and the expanding search problem [5]. In these problems, detection is assumed to be perfect. In this study, we combine both fields, with a focus on approximation algorithms. In particular, we extend the approximability results for graph search problems with perfect detection to the case with imperfect detection. These results were published in [4].

We consider the case where the target is hidden at random, and the case where the target is hidden adversarially. If the target is hidden at random, our goal is to find an (infinite) search path that minimizes the expected time until the target is found. If the target is hidden adversarially, meaning that it can observe our search strategy before choosing a vertex to hide, our goal is to find an (infinite) search path that minimizes the maximum expected time until detection over the vertices.

We consider two ways to search the graph: *pathwise search* (as is standard in routing problems) and *expanding search*, where we only incur the cost of an edge if we use it for the first time and any following traversal has no cost. Natural applications of searching with imperfect detection include search and rescue missions, where a local search (such as an excavation) as well as moving equipment takes substantial time. Other applications can be found in testing the state of a system by investigating its components. The reliability of a test depends on the thoroughness and precision and hence on the time invested. Moreover, switching between tests takes a setup time. This is closely related to time-critical testing problems [1].

*M.v.Ee.01@mindef.nl. Faculty of Military Sciences, Netherlands Defence Academy, Den Helder, The Netherlands.

†r.a.sitters@vu.nl. Department of Operations Analytics, Vrije Universiteit Amsterdam, The Netherlands.

2 Problem definitions

In imperfect graph search problems, we are given a graph $G = (V, E)$ with metric distances $d(i, j)$ for $v_i, v_j \in V$, and a root $r \in V$ where the search starts. We assume the searcher moves at unit speed, and that searching a vertex takes 1 time unit. It is allowed to search a vertex multiple times consecutively. The target is hidden in vertex v_i with probability p_i . The probability that the target is detected while it is at the vertex that is being searched is $\gamma \in (0, 1]$. We consider the imperfect graph search problem with either pathwise search (IGS-P) or expanding search (IGS-E).

Let C_{ij} be the arrival time at the j -th visit to vertex v_i . This includes the search time in v_i and consecutive visits to the same vertex are considered separately. In both IGS-P and IGS-E, we aim to minimize the expected time until target detection, i.e.,

$$\sum_{v_i \in V} \sum_{j=1}^{\infty} (1 - \gamma)^{j-1} \gamma p_i C_{ij}.$$

We also study the problem of maximizing the probability of finding the target, given a deadline T . Since the case with perfect detection is known as the orienteering problem, we will refer to this problem as IGS-O. Here, a solution is given by, for each vertex v_i the number of times k_i it is searched, together with a path visiting all v_i with $k_i \geq 1$ such that the search is done within deadline T . The objective for IGS-O is to maximize the probability of detection, i.e.,

$$\sum_{v_i \in V} \left(1 - (1 - \gamma)^{k_i}\right) p_i.$$

In imperfect graph search problems with adversarial targets, we have the same setting with the only difference that the target is not hidden at random, but instead, we assume the hider can observe our search strategy before choosing the vertex where it will hide. Hence, it chooses the vertex with the maximum expected time until detection. Therefore, we need to design a search strategy that minimizes the maximum expected time until detection over the vertices, where the expectation is taken only w.r.t. the uncertainty in detecting the target, i.e.,

$$\max_{v_i \in V} \sum_{j=1}^{\infty} (1 - \gamma)^{j-1} \gamma C_{ij}.$$

We consider imperfect graph search problems with adversarial targets in both the pathwise (AIGS-P) and the expanding search paradigm (AIGS-E).

3 Our contributions

We give the first approximation algorithms for imperfect graph search problems, with either a random or adversarially hidden target, in either the pathwise search or the expanding search paradigm. Although our algorithms produce infinite search paths, all the resulting solutions can be represented in polynomial space. This is because our search paths are repetitions of sequences of polynomial length, possibly preceded by an initial sequence of polynomial length.

Our first result is that IGS-P reduces, with a loss of a factor $1 + \varepsilon$, to weighted-TRP. Hence, given the $(3.59 + \varepsilon)$ -approximation for weighted-TRP, we obtain an $(3.59 + \varepsilon)$ -approximation for IGS with pathwise search.

It turns out that roughly the same reduction also works for IGS-E. Hence, we obtain an $(2e + \varepsilon)$ -approximation. As a side result, we note that a similar reduction also gives the first constant factor approximation for IGS-O.

For imperfect graph search problems with adversarial targets, we present simple algorithms with constant factor approximation guarantees. For pathwise search (AIGSP), the algorithm’s solution walks back and forth along the (approximate) shortest path that visits all vertices, and either searches for one time unit or for $\lceil \frac{1}{\gamma} \rceil$ time units when visiting a vertex. This gives a 3.2826-approximation. For expanding search (AIGSE), we construct the minimum spanning tree in an expanding manner, and search each vertex for one time unit. Afterwards, the solution searches the vertices in reversed order, and so on. This gives a 2.0876-approximation. Here, we note that the complexity of AIGSE is open. It is even unclear whether the decision version of the problem is contained in NP. In Table 1 is a summary of our results, together with known approximability results for perfect detection.

Table 1: Approximability results for perfect and imperfect detection.

	Random target		Adversarial target	
	Perfect	Imperfect	Perfect	Imperfect
Pathwise search	$3.59 + \varepsilon$	$3.59 + \varepsilon$	$1.5 - \varepsilon_0$	3.2826
Expanding search	$2e + \varepsilon$	$2e + \varepsilon$	1	2.0876
Orienteering	$0.5 - \varepsilon$	$0.5 - \varepsilon$		

References

- [1] Alessandro Agnetis, Ben Hermans, Roel Leus, and Salim Rostami. Time-critical testing and search problems. *European Journal of Operational Research*, 296(2):440–452, 2022.
- [2] Stanley J. Benkoski, Michael G. Monticino, and James R. Weisinger. A survey of the search theory literature. *Naval Research Logistics*, 38(4):469–494, 1991.
- [3] Kamalika Chaudhuri, Brighten Godfrey, Satish Rao, and Kunal Talwar. Paths, trees, and minimum latency tours. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 36–45, 2003.
- [4] Martijn van Ee and René Sitters. Approximation algorithms for graph search problems with imperfect detection. In *Proceedings of the 23rd International Workshop on Approximation and Online Algorithms*, pages 213–227, 2025.
- [5] Svenja M. Griesbach, Felix Hommelsheim, Max Klimm, and Kevin Schewior. Improved approximation algorithms for the expanding search problem. In *Proceedings of the 31st Annual European Symposium on Algorithms*, pages 54:1–54:15, 2023.

Two Complexity Results on Spanning-Tree Congestion Problems

Sunny Atalig * Marek Chrobak * Christoph Dürr †
 Petr Kolman ‡ Huong Luu § Jiří Sgall (Speaker) ‡ Gregory Zhu *

1 Introduction

We focus on *the spanning-tree congestion problem (STC)*, that arises naturally in some network design and routing problems. The problem can be viewed as a special case of the graph sparsification problem where a graph G is embedded into its spanning tree T by mapping each edge (x, y) of G to the unique x -to- y path in T . The congestion of an edge $e \in T$ is defined as the number of edges of G whose corresponding path in T traverses e , and the congestion of T is the maximum congestion of its edges. In the STC problem, we are given a graph G and the objective is to compute a spanning tree with minimum congestion. This minimum congestion value is referred to as the *spanning-tree congestion of G* and denoted by $\text{stc}(G)$.

The concept of spanning-tree congestion was introduced under different names in the late 1990s and in 2004 formalized by Ostrovskii [7]. The STC problem is NP-hard, as first proven by Löwenstein [5] in 2010.

In the decision version of STC, in addition to a graph G we are also given an integer K , and the goal is to determine if $\text{stc}(G) \leq K$. A natural variant of STC, when the congestion parameter K is a fixed constant (rather than given as input) is denoted K -STC. The K -STC problem was shown to be NP-complete for $K \geq 5$ by Luu and Chrobak [6], building on earlier results for larger constants. On the other hand, K -STC is solvable in linear time for $K \leq 3$ [8]. The complexity status of 4-STC remains an intriguing open problem.

Kolman [3] observed that the existing NP-hardness proofs of the STC problem used graphs of unbounded degree, and raised the question about the complexity of for graphs of constant degree. Recently, Lampis et al. [4] proved that, for any constant $\Delta \geq 8$, STC is NP-hard for graphs with maximum degree Δ , leaving open the complexity of STC for degree bounds between 3 and 7.

*University of California at Riverside, USA

†Sorbonne Université, CNRS, LIP6, France

‡Charles University, Prague, Czech Republic

§California Polytechnic University at Pomona, USA

Addressing the problem left open by Lampis et al. [4], we fully resolve the status of STC for bounded-degree graphs:

Theorem 1 *Problem STC is NP-hard for graphs of maximum degree at most 3.*

In this presentation we focus on our second result that studies the K -STC problem for K -edge-connected graphs:

Theorem 2 *There is an $\tilde{O}(|E|)$ -time algorithm that, given a K -edge-connected graph $G = (V, E)$, determines whether $\text{stc}(G) = K$. The $\tilde{O}(|E|)$ time bound is actually independent of K .*

Our solution is based on the so-called cactus representation of K -cuts in K -edge-connected graphs that was developed by Dinic et al. [1] (see also [2]). We further refine this characterization for graphs with congestion K , to obtain additional properties that lead to a fast algorithm. Besides its own interest, this result sheds new light on the complexity of 4-STC, showing that its difficulty is related to the presence of cuts of size less than 4 in the graph.

2 K -edge-connected graphs with congestion K

We assume that $G = (V, E)$ is a given graph that is K -edge-connected, i.e., it remains connected even after removing any $K - 1$ edges.

For any subset $X \notin \{\emptyset, V\}$ of vertices, by ∂X we denote the set of edges between X and $\bar{X} = V \setminus X$, and we call it a *cut*. We refer to X and \bar{X} as the *shores* of cut ∂X . If $|\partial X| = K$, we say that ∂X is a K -cut. Two cuts ∂X and ∂Y are called *nested* if one of the four pair-wise shore intersections $X \cap Y$, $\bar{X} \cap Y$, $X \cap \bar{Y}$ and $\bar{X} \cap \bar{Y}$ is empty. If all four intersections are non-empty then we say that cuts ∂X and ∂Y *cross*.

We call a K -cut *basic* if it does not cross any other K -cut. Thus the basic K -cuts form a laminar family. For odd values of K , it may be shown that no two K -cuts cross, i.e., every K -cut is a basic one. For even K the structure is more complex and we use the cactus representation of K -cuts.

We now define cactus graphs; we will use terminology of *nodes* and *links* (instead of vertices and edges) for them. A connected multigraph is called a *cactus graph* if every link belongs to exactly one cycle. (Equivalently, it is a 2-edge-connected graph whose biconnected components are cycles.) Cactus cycles of length 2 are called *trivial*. If all its cycles are trivial, the cactus forms a tree whose adjacent nodes are connected by two parallel links.

A cactus representation of $G = (V, E)$ consists of a cactus graph (U, F) and an associated mapping $\phi : V \rightarrow U$ of the vertices of the graph onto the cactus nodes such that the following holds: For each set $X \subseteq V$, ∂X is a K -cut if and only if $X = \phi^{-1}(Q)$ for some $Q \subseteq U$ such that ∂Q is a 2-cut of the cactus graph.

In other words, any pair of cactus links on a common cycle disconnects the cactus and consequently also defines a partition of the vertices of the original graph—and this partition is a pair of shores of a K -cut: moreover these are exactly all

K -cuts of G . As a typical example, the cactus (U, F) may be a single cycle. Then each cactus node represents a highly connected subgraph of G and each cactus link represents $K/2$ edges between these adjacent connected subgraphs.

The algorithm proceeds by dynamic programming guided by the tree-like structure of the cactus representation and tries to reconstruct the spanning tree T of congestion K . The key structural insight is that for any spanning tree T of congestion K and any basic K -cut ∂X , all edges in $T \cap \partial X$ share a common endpoint. This enables the dynamic program to remember only these plausible common endpoints for each basic K -cut.

Further insights are needed to show how the information for basic cuts can be combined in cycles of a cactus representation. Some technical refinements are necessary to characterize the partial solutions properly.

The full proofs can be found in ArXiv report arXiv:2601.10881. The results are accepted for a presentation at LATIN 2026.

References

- [1] E. A. DINIC, A. V. KARZANOV, AND M. V. LOMONOSOV. The structure of a system of minimal edge cuts of a graph. In *Issledovani po diskretnoi optimizacii (Studies in discrete optimization)*, pages 290–306. Nauka, Moscow, 1976. In Russian.
- [2] T. FLEINER AND A. FRANK. A quick proof for the cactus representation of mincuts. Technical Report QP-2009-03, Egerváry Research Group, Budapest, 2009.
- [3] P. KOLMAN. Approximating spanning tree congestion on graphs with polylog degree. In *Proc. 35th IWOCA*, pages 497–508, 2024.
- [4] M. LAMPIS, V. MITSOU, E. NEMERY, Y. OTACHI, M. VASILAKIS, AND D. VAZ. Parameterized spanning tree congestion. In *Proc. 50th MFCS*, pages 65:1–65:20, 2025.
- [5] C. LÖWENSTEIN. *In the Complement of a Dominating Set*. PhD thesis, Technische Universität at Ilmenau, 2010.
- [6] H. LUU AND M. CHROBAK. Better hardness results for the minimum spanning tree congestion problem. *Algorithmica*, 87(1):148–165, 2025.
- [7] M. OSTROVSKII. Minimal congestion trees. *Discrete Mathematics*, 285(1):219–226, 2004.
- [8] Y. OTACHI, H. L. BODLAENDER, AND E. J. VAN LEEUWEN. Complexity results for the spanning tree congestion problem. In *Proc. 36th WG*, pages 3–14, 2010.

On the Integrality Gap of MFN Relaxation for the Capacitated Facility Location Problem

Mong-Jen Kao (Speaker) *

Abstract

The Multicommodity Flow Network (MFN) relaxation, originally developed in [An, Singh, Svensson, FOCS 2014], is the only known relaxation that leads to an LP-based algorithm with $O(1)$ -guarantee for the classic capacitated facility location (CFL) problem. In this work, we show that the MFN relaxation has an integrality gap at most $(10 + \sqrt{67})/2 \approx 9.0927$ for the CFL problem.

1 Introduction

The capacitated facility location (CFL) is a classic cover problem that takes the distance measure of assignments and the concept of limited supply into consideration. In this problem, we are given a set of facilities, a set of clients, and a distance metric defined on them. Each facility is associated with an open cost and a capacity, which is the number of clients it can serve when opened up. The cost of serving a client using a facility equals the distance between them. The goal of this problem is to determine a set of facilities to open up and an assignment of the clients to the opened facilities that respects the capacity limits so as to minimize that the facility open cost plus the service cost.

The CFL problem was first considered by Shmoys, Tardos, and Aardal in [16] back in the 90s. Since then, the majority of the results known for this problem were based on local search algorithms, see, e.g., [15, 14, 18, 5, 11, 7, 2].

As opposed to the rich LP-based toolboxes developed for the uncapacitated facility location (UFL) problem, e.g., [13, 6, 16, 8, 9, 10, 12], the fact that no LP relaxations were known to provide a bounded integrality gap for CFL was intriguing and surprising for a long time. In particular, deriving an LP-based algorithm with a constant approximation guarantee for CFL was highlighted as one of ten open problems in the textbook due to Williamson and Shmoys [17]. This problem was resolved by the notable work of An, Singh, and Svensson [3, 4], and a strong multi-commodity flow network (MFN) relaxation was designed. The

*mjkao@nycu.edu.tw. Department of Computer Science, National Yang-Ming Chiao-Tung University, 1001 University Road, Hsinchu City 300, Taiwan.

idea of this relaxation is to impose Knapsack-cover type constraints, formulated as reassignable partial assignments to be amended in each qualifying test. This relaxation is known to have an integrality gap of at least 2, and the authors showed that the integrality gap is at most 288.

The main challenge in designing a rounding algorithm for the MFN relaxation, as opposed to the various LP-based algorithms for UFL, is in the necessity to meet the various types of capacity constraints imposed in the relaxation. Due to this reason, dealing with this relaxation has been a challenging task, since the rounding operations are insensitive in nature to such types of capacity constraints. Up to date, it still remains an open question to determine the exact integrality gap of the MFN relaxation and develop LP-based algorithms with better guarantees.

Our Result and Technique. We present a rounding algorithm for the MFN relaxation that proves an improved upper-bound on the integrality gap.

Theorem 1 *There is an LP-rounding algorithm for the MFN relaxation that computes an integral solution with a guarantee of $(10 + \sqrt{67})/2 \approx 9.0927$ for the capacitated facility location problem in polynomial-time.*

Our algorithm is built on an iterative rounding scheme that combines new insights and ideas with techniques developed in the past [1, 3, 4]. The power of the MFN relaxation lies in its capability to remove the large facilities, namely, facilities with large fractional values, from consideration, using the cost of the assignments made to them as the extra price [3, 4], and what remains is the rounding problem for an instance consisting of small facilities and the assignments made to them. In [3, 4], the small instance is created by proving the existence of a *nice* structured flow which sends a firm fraction of demand from each client of interest to the small facilities. Then the work due to Abraham et al. [1] is applied.

In our algorithm, we take a different philosophy in creating and handling the small instance. Our rounding procedure aims at fractionally serving the clients while making sure that the rounded facilities are sparsely-loaded by the assignments, so that a final round-up on the rounded assignments can be made to guarantee the feasibility. The sparsity of the small facilities is guaranteed by default. In our algorithm, we observe that the large facilities are sparsely-loaded by the flow sent to them, and a reasonable final round-up can be made when necessary. This distinguishable characteristic allows our improvement on the guarantee.

Our rounding procedure for the small instance is inspired by the work due to Abraham et al. [1] and prior works developed for uncapacitated facility location problem. In each iteration, the facility with the least average rerouting cost is selected to be rounded, and all the flow along with the facility value in the vicinity is rerouted simultaneously to the selected facility.

To bound the extra cost incurred, one essential element is to guarantee a low assignment radius for each client. In [1], this is done by applying a so-called

filtering technique, which uses the Markov inequality and unconditional round-up. This inevitably compensates the guarantee for the assignment radius.

In our algorithm, we use a carefully designed LP to handle the fractional solution for the residual instance. In a nutshell, the LP allows in a subtle and crucial way for our rounding algorithm to balance the facility cost and the assignment cost we spend in each iteration. To obtain our final guarantee, we then resort to the cost of the flow that is sent in the MFN network.

References

- [1] Z. ABRAMS, K. MUNAGALA, AND S. PLOTKIN, *On the integrality gap of capacitated facility location*. Technical Report CMU-CS-02-199, Carnegie Mellon University, 2002.
- [2] A. AGGARWAL, A. LOUIS, M. BANSAL, N. GARG, N. GUPTA, S. GUPTA, AND S. JAIN, *A 3-approximation algorithm for the facility location problem with uniform capacities*, Math. Program., 141 (2013), pp. 527–547.
- [3] H. AN, M. SINGH, AND O. SVENSSON, *Lp-based algorithms for capacitated facility location*, in 55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014, IEEE Computer Society, 2014, pp. 256–265.
- [4] H.-C. AN, M. SINGH, AND O. SVENSSON, *Lp-based algorithms for capacitated facility location*, SIAM Journal on Computing, 46 (2017), pp. 272–306.
- [5] M. BANSAL, N. GARG, AND N. GUPTA, *A 5-approximation for capacitated facility location*, in Algorithms - ESA 2012 - 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings, L. Epstein and P. Ferragina, eds., vol. 7501 of Lecture Notes in Computer Science, Springer, 2012, pp. 133–144.
- [6] F. A. CHUDAK AND D. B. SHMOYS, *Improved approximation algorithms for the uncapacitated facility location problem*, SIAM Journal on Computing, 33 (2003), pp. 1–25.
- [7] F. A. CHUDAK AND D. P. WILLIAMSON, *Improved approximation algorithms for capacitated facility location problems*, in Proceedings of the 7th International IPCO Conference on Integer Programming and Combinatorial Optimization, Berlin, Heidelberg, 1999, Springer-Verlag, pp. 99–113.
- [8] K. JAIN, M. MAHDIAN, E. MARKAKIS, A. SABERI, AND V. V. VAZIRANI, *Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp*, J. ACM, 50 (2003), p. 795–824.

- [9] K. JAIN, M. MAHDIAN, AND A. SABERI, *A new greedy approach for facility location problems*, in Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing, STOC '02, New York, NY, USA, 2002, Association for Computing Machinery, pp. 731–740.
- [10] K. JAIN AND V. V. VAZIRANI, *Approximation algorithms for metric facility location and k -median problems using the primal-dual schema and lagrangian relaxation*, J. ACM, 48 (2001), pp. 274–296.
- [11] M. R. KORUPOLU, C. G. PLAXTON, AND R. RAJARAMAN, *Analysis of a local search heuristic for facility location problems*, in Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '98, USA, 1998, Society for Industrial and Applied Mathematics, pp. 1–10.
- [12] S. LI, *A 1.488 approximation algorithm for the uncapacitated facility location problem*, in Proceedings of the 38th International Conference on Automata, Languages and Programming - Volume Part II, ICALP'11, Berlin, Heidelberg, 2011, Springer-Verlag, pp. 77–88.
- [13] J.-H. LIN AND J. S. VITTER, *Approximation algorithms for geometric median problems*, Inf. Process. Lett., 44 (1992), p. 245–249.
- [14] M. MAHDIAN AND M. PÁL, *Universal facility location*, in Algorithms - ESA 2003, 11th Annual European Symposium, Budapest, Hungary, September 16–19, 2003, Proceedings, G. D. Battista and U. Zwick, eds., vol. 2832 of Lecture Notes in Computer Science, Springer, 2003, pp. 409–421.
- [15] M. PÁL, E. TARDOS, AND T. WEXLER, *Facility location with nonuniform hard capacities*, in Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science, FOCS '01, USA, 2001, IEEE Computer Society, p. 329.
- [16] D. B. SHMOYS, E. TARDOS, AND K. AARDAL, *Approximation algorithms for facility location problems (extended abstract)*, in Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, STOC '97, New York, NY, USA, 1997, Association for Computing Machinery, p. 265–274.
- [17] D. P. WILLIAMSON AND D. B. SHMOYS, *The Design of Approximation Algorithms*, Cambridge University Press, USA, 1st ed., 2011.
- [18] J. ZHANG, B. CHEN, AND Y. YE, *A multi-exchange local search algorithm for the capacitated facility location problem: (extended abstract)*, in Integer Programming and Combinatorial Optimization, 10th International IPCO Conference, New York, NY, USA, June 7–11, 2004, Proceedings, G. L. Nemhauser and D. Bienstock, eds., vol. 3064 of Lecture Notes in Computer Science, Springer, 2004, pp. 219–233.

Interval Scheduling under Approximate Envy-Freeness

Sander Borst* Golnoosh Shahkarami (Speaker) † Rohit Vaish‡

1 Introduction

We study interval scheduling from the perspective of fair allocation. The input consists of a set of intervals, each specified by a start time, an end time, and a nonnegative weight, together with m identical machines. A feasible schedule assigns intervals to machines so that no two intervals overlap on the same machine. The classical objective is to maximize the total weight of accepted intervals, a problem extensively studied in both offline and online settings [1, 2].

In many applications, however, purely efficiency-driven schedules can be highly unbalanced across machines. To address this, we incorporate a fairness requirement inspired by the fair division literature [3]. Viewing machines as agents and intervals as items, we require the resulting allocation to be *envy-free up to one item* (EF1), a widely studied relaxation of envy-freeness for indivisible goods [5, 4]. In the unweighted setting, where all intervals have unit weight, EF1 coincides with the stronger notion of envy-freeness up to any item (EFX) [6].

We consider both offline and online variants of the problem. In the offline model, the entire set of intervals is known in advance. In the online model, intervals arrive sequentially in nondecreasing order of start times; each interval must be accepted or rejected upon arrival, rejections are irrevocable, and accepted intervals may be revoked before their end time. Efficiency is measured against the optimal offline solution without fairness constraints. In the offline setting, we quantify the loss due to fairness via the *price of fairness* [7, 8]; in the online setting, we measure the *fair competitive ratio*, capturing the combined loss due to fairness and online uncertainty [9].

Our main question is whether EF1 can be enforced while retaining constant-factor efficiency guarantees. We answer this affirmatively and provide tight bounds in both offline and online settings.

*sborst@mpi-inf.mpg.de Max Planck Institute for Informatics

†gshahkar@mpi-inf.mpg.de Max Planck Institute for Informatics, University of Bremen

‡rvaish@iitd.ac.in Indian Institute of Technology Delhi

2 Results and Techniques

Offline EF1 Scheduling. In the unweighted setting, we design a polynomial-time algorithm that computes an EF1 allocation achieving a $3/2$ price of fairness. We also prove a matching lower bound, showing that this factor is unavoidable even when all intervals have identical weights. Thus, $3/2$ precisely characterizes the intrinsic efficiency loss caused by EF1 in the offline model.

We further extend this result beyond uniform valuations. When weights take a constant number of distinct values, we obtain constant-factor EF1 approximations. At the same time, we show that the $3/2$ lower bound persists even in the highly structured unit-length weighted setting, demonstrating that the efficiency loss is driven by the fairness constraint itself rather than by interval lengths or valuation complexity.

Our offline algorithm follows a two-phase approach. We begin with an efficiency-optimal schedule without fairness constraints [11, 12] and then enforce EF1 by redistributing intervals across machines. This is achieved through a combination of *rebalancing* and carefully chosen *swaps* of intervals between machines. Each swap may discard a single interval, and we tightly control the number of such losses to match the $3/2$ lower bound.

Online EF1 Scheduling. In the online unweighted setting, we propose a deterministic algorithm, *Greedy-Balanced*, which maintains EF1 at all times while achieving a competitive ratio of $2 - \frac{1}{m}$ with respect to the offline optimum without fairness. We also prove a matching lower bound for deterministic algorithms, establishing the tightness of this guarantee.

The online analysis builds on two key ideas. First, we separate feasibility from fairness by filtering intervals through an efficiency-optimal online benchmark [10] and enforcing EF1 through load balancing. Second, we analyze the execution using a block-based charging argument, in which intervals that cannot be accepted due to fairness are charged to accepted intervals within the same block. This reveals an inherent dependence on the number of machines: fairness becomes increasingly restrictive as m grows, leading to the tight factor $2 - \frac{1}{m}$.

Together, our offline and online results cleanly separate the efficiency loss due to fairness from the additional loss caused by online uncertainty. Although we present our results for nonnegative weights (goods), the same arguments extend to the chore setting with negative weights.

Empirical Evaluation. We complement our theoretical analysis with experiments on real-world scheduling traces from the Parallel Workloads Archive. Across all benchmark instances and machine counts, Greedy-Balanced consistently performs significantly better than its worst-case guarantee. These results suggest that while the theoretical bounds are tight in adversarial settings, the practical efficiency loss due to EF1 is substantially smaller on natural workloads.

References

- [1] A.W.J. KOLEN, J.K. LENSTRA, C.H. PAPADIMITRIOU, AND F.C.R. SPIEKSMAN (2007). Interval Scheduling: A Survey. *Naval Research Logistics*, 54(5):530–543.
- [2] R.J. LIPTON AND A. TOMKINS (1994). Online Interval Scheduling. In *Proceedings of SODA*.
- [3] H. MOULIN (2004). Fair Division and Collective Welfare. *MIT Press*.
- [4] E. BUDISH (2011). The Combinatorial Assignment Problem: Approximate Competitive Equilibrium from Equal Incomes. *Journal of Political Economy*, 119(6):1061–1103.
- [5] R.J. LIPTON, E. MARKAKIS, E. MOSSEL, AND A. SABERI (2004). On Approximately Fair Allocations of Indivisible Goods. In *Proceedings of EC*.
- [6] I. CARAGIANNIS, D. KUROKAWA, H. MOULIN, A.D. PROCACCIA, N. SHAH, AND J. WANG (2019). The Unreasonable Fairness of Maximum Nash Welfare. *ACM Transactions on Economics and Computation*, 7(3).
- [7] N. BANSAL, A. FRIGGSTAD, AND R. THAKUR (2011). On the Price of Fairness. In *Proceedings of ESA*.
- [8] D. CHAKRABARTY, J. KÖNEMANN, AND D. PRITCHARD (2012). The Efficiency of Fair Division. *ACM Transactions on Algorithms*, 8(4).
- [9] D.D. SLEATOR AND R.E. TARJAN (1985). Amortized Efficiency of List Update and Paging Rules. *Communications of the ACM*, 28(2):202–208.
- [10] U. FAIGLE AND W. KERN (1995). A Note on Online Interval Scheduling. *Information Processing Letters*, 55(2):73–77.
- [11] E.M. ARKIN AND R.E. TARJAN (1987). Scheduling to Minimize Maximum Lateness. *Discrete Applied Mathematics*, 18:1–12.
- [12] A. BOUZINA AND C. EMMONS (1996). Interval Scheduling on Identical Parallel Machines. *Operations Research Letters*, 19(3):125–129.

Scheduling Problems with Constrained Rejections

Sami Davies (Speaker) * Venkatesan Guruswami† Xuandi Ren‡

1 Introduction

Two central problems in scheduling theory are Makespan Minimization on Unrelated Machines and its dual, the Santa Claus problem (also called Max-Min Fair Allocation). In **MAKESPAN**, a set of jobs J are available to schedule on a set of machines M , where the processing time of job $j \in J$ on machine $i \in M$ is $p_{i,j}$. We let $p_{i,j} = \infty$ if job j cannot be processed on machine i . The jobs must be assigned non-preemptively, and the goal is to minimize the makespan. In **SANTA CLAUS**, a set of items I are available to assign to a set of agents A , where the value agent $i \in A$ receives from item $j \in I$ is $p_{i,j} \geq 0$. Each item can be allocated to at most one agent, and the total value agent i receives is the sum of the values from individual items. The goal is to find an allocation of items to agents so that the minimum total value of any agent is maximized. For both problems in their most general versions, progress on the hardness and algorithmic fronts has been stuck.

In this work, we study a different type of relaxation for these problems. For **MAKESPAN** we allow a $1 - \alpha$ fraction of jobs to be rejected (i.e., not scheduled on any machine), for **SANTA CLAUS** we allow a $1 - \alpha$ fraction of agents to be rejected (i.e., receive 0 value). Then we study how this affects the approximation factor β for makespan/value guarantee. While a standard approximation algorithm fixes $\alpha = 1$ and optimizes β , we study the trade-off between the two parameters.

This relaxation of **MAKESPAN** is the unweighted case of the Maximum General Assignment Problem, for which Feige and Vondrák [FV06] gave an algorithm that achieved $\alpha = 1 - 1/e + 10^{-180}$ and $\beta = 1$. Nutov, Beniaminy and Yuster [NBY06] showed the NP-hardness of $\alpha = 1 - \varepsilon$ and $\beta = 1$ for some $\varepsilon > 0$. In this work, we give an algorithm that reaches $\alpha > 1 - 1/e + 0.02$ for $\beta = 3/2$, which we hope could be a starting point for further interest in this problem.

Theorem 1. *For the **MAKESPAN** problem, there is a polynomial-time randomized algorithm, which given as input an instance where the optimal makespan for scheduling all n jobs on the m machines is T , schedules in expectation $\frac{6e-5}{6e+1} \cdot n > 0.6533 \cdot n > (1 - 1/e + 0.02) \cdot n$ many jobs within makespan $\frac{3}{2} \cdot T$.*

*samidavies@berkeley.edu. Department of EECS, UC Berkeley and RelationalAI.

†venkatg@berkeley.edu. Simons Institute for the Theory of Computing, and Departments of EECS and Mathematics, UC Berkeley.

‡xuandi_ren@berkeley.edu. Department of EECS, UC Berkeley.

To the best of our knowledge, this is the first result examining the tradeoff between makespan and the fraction of scheduled jobs when the makespan is not T or $2T$. Allowing some jobs to be rejected is not only a natural variant, but we believe it still captures much of the difficulty in the original problems. We conjecture there are some “transition points” on the curve between α and β , where when one wishes to schedule a slightly larger fraction of jobs (say from α to $\alpha + \delta$, for a small constant $\delta > 0$), a large jump in the approximation factor, β , is needed.

SANTA CLAUS with constrained rejections has been studied algorithmically by Golovin [Gol05], who gave an algorithm which for any $k \in \mathbb{Z}^+$, finds an allocation for SANTA CLAUS where at least a $(1 - 1/k)$ fraction of agents obtain value at least $1/k \cdot T$. We show the hardness of this problem in the high-value regime.

Theorem 2. *For the SANTA CLAUS problem, there are universal constants $\delta, \varepsilon > 0$, such that given the existence of an assignment where all n agents receive total value at least T , it's NP-hard to find an assignment where $(1 - \delta) \cdot n$ agents receive total value $(1 - \varepsilon) \cdot T$.*

To prove Theorem 2, we introduce bicriteria SET PACKING as an intermediate problem (see Definition 3), which we believe may be of independent interest. We give initial hardness and algorithmic results for this problem too.

2 Proof Overviews

Overview of the proof of Theorem 1 We define the set of large edges as $E_L = \{(i, j) \mid i \in M, j \in J, \frac{T}{2} < p_{i,j} \leq T\}$ and $E_S = \{(i, j) \mid i \in M, j \in J, 0 \leq p_{i,j} \leq \frac{T}{2}\}$, and then define G_L to be the bipartite graph induced by E_L .

Our algorithm proving Theorem 1 has three sub-procedures:

1. Algorithm 1: computes a maximum matching (of size $m^* \leq m$) on G_L and schedules these m^* jobs within makespan T .
2. Algorithm 2: rounds the solution to the Configuration LP for MAKESPAN by allowing each machine to sample a configuration of jobs with makespan T , according to the LP solution. We show if all jobs can fit into makespan T , the algorithm schedules a $(1 - \frac{1}{e})$ fraction of jobs in expectation within makespan T .
3. Algorithm 3: greedily schedules small jobs (i.e., only assigning j to i if $(i, j) \in E_S$), and in total schedules $\frac{1}{6}(n - m^*)$ many jobs within makespan $\frac{T}{2}$ using only edges in E_S .

While each sub-procedure on its own is simple, the salient point is that they can be combined in a way that is stronger than any of them individually. In total, we either schedule m^* jobs from Algorithm 1 (resulting in a schedule of makespan T), or we first run Algorithm 3 to obtain a schedule with makespan $T/2$ and then run Algorithm 2 on the remaining jobs (resulting in a schedule with makespan $\frac{3}{2} \cdot T$).

Overview of the proof of Theorem 2 We reduce from the following hardness of $(\frac{\beta}{\alpha}, 1 - \varepsilon)$ -bicriteria SET PACKING, a problem we define next.

Definition 3 (SET PACKING Problem). *In the SET PACKING problem, we are given as input a collection of sets \mathcal{S} over a universe U , and the goal is to find the maximum number of disjoint sets in \mathcal{S} .*

For $0 \leq \alpha, \beta \leq 1$, (α, β) -bicriteria SET PACKING is the task where given the guarantee that there are m sets in \mathcal{S} partitioning U , the goal is to find $m' = \alpha \cdot m$ many sets $S_1, \dots, S_{m'}$ in \mathcal{S} together with their subsets $A_1 \subseteq S_1, \dots, A_{m'} \subseteq S_{m'}$, such that for every $i \in [m']$, $|A_i| \geq \beta \cdot |S_i|$, and the sets $A_1, \dots, A_{m'}$ are disjoint.

Then to prove Theorem 2 for SANTA CLAUS, we prove the following hardness for SET PACKING.

Theorem 4. *For the SET PACKING problem, there are universal constants $\varepsilon > 0, \alpha > \beta > 0$, such that given as input a collection \mathcal{S} of n sets over a universe U and the existence of $m = \alpha \cdot n$ sets in \mathcal{S} that form a partition of U , it is NP-hard to find $m' = \beta \cdot n$ many sets $S_1, \dots, S_{m'}$ in \mathcal{S} together with their subsets $A_1 \subseteq S_1, \dots, A_{m'} \subseteq S_{m'}$, such that for every $i \in [m']$, $|A_i| \geq (1 - \varepsilon) \cdot |S_i|$, and the sets $A_1, \dots, A_{m'}$ are disjoint.*

We use the canonical reduction from q -CSP to SET PACKING [HSS06, LST24], while replacing the gadget in [LST24] with Feige’s hypercube partition system [Fei98], and plugging in Feige’s hardness of degree-5 3-CNF [Fei98].

References

- [Fei98] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998.
- [FV06] Uriel Feige and Jan Vondrák. Approximation algorithms for allocation problems: Improving the factor of $1 - 1/e$. In *47th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2006, Berkeley, California, USA, October 21-24, 2006, Proceedings*, pages 667–676. IEEE Computer Society, 2006.
- [Gol05] Daniel Golovin. *Max-min fair allocation of indivisible goods*. School of Computer Science, Carnegie Mellon University, 2005.
- [HSS06] Elad Hazan, Shmuel Safra, and Oded Schwartz. On the complexity of approximating k -set packing. *Comput. Complex.*, 15(1):20–39, 2006.
- [LST24] Euiwoong Lee, Ola Svensson, and Theophile Thiery. Asymptotically optimal hardness for k -set packing and k -matroid intersection, 2024.
- [NBY06] Zeev Nutov, Israel Beniaminy, and Raphael Yuster. A $(1-1/e)$ -approximation algorithm for the generalized assignment problem. *Oper. Res. Lett.*, 34(3):283–288, 2006.

Approximating Fair Repetitive Scheduling

Moritz Buchem (Speaker) ^{*} Nicole Megow [†] Marc Uetz [‡]
 Leoni Winschermann [§]

1 Introduction

In many real-world applications, such as workforce scheduling, cloud computing, or service systems, the same agents or tasks recur across multiple periods, often with varying characteristics. In such repetitive settings, optimizing each planning period in isolation may lead to systematic disadvantage for individual agents, even if every single schedule is efficient. As a result, fairness among agents must be measured across schedules rather than within a single one. Such scenarios naturally lead to the question of how to compute schedules that ensure “global fairness” for more than just one planning horizon. Recently, Hermelin et al. [4] introduced the model of fair repetitive scheduling capturing fairness among agents who repeatedly share a scheduling resource over multiple days or periods. We describe the model in a (possibly day dependent) unrelated parallel machine environment.

We consider a repetitive scheduling problem over a finite set of days $\mathcal{D} = \{1, \dots, D\}$ and a repeating set of jobs $\mathcal{J} = \{1, \dots, n\}$. Each day $d \in \mathcal{D}$, all jobs must be processed on a machine environment \mathcal{M}_d , which may vary arbitrarily across days. The processing time of job j on machine $i \in \mathcal{M}_d$ on day d is denoted by p_{jdi} . Furthermore, job j has release date r_{jdi} indicating the earliest possible start time of j on day d and machine i , and it has a weight w_{jd} on day d . A schedule for day d assigns each job j to a machine $i(j) \in \mathcal{M}_d$ and specifies the exact order in which all jobs assigned to a machine are scheduled. The cumulative weighted completion time of job j over the planning horizon is defined as $C_j := \sum_{d \in \mathcal{D}} w_{jd} C_{jd}$. The objective is to minimize the maximum cumulative weighted completion time over all jobs, that is,

$$\min \max_{j \in \mathcal{J}} C_j.$$

^{*}mbuchem@uni-bremen.de. Faculty of Mathematics and Computer Science, University of Bremen, Bremen, Germany.

[†]nicole.megow@uni-bremen.de. Faculty of Mathematics and Computer Science, University of Bremen, Bremen, Germany.

[‡]m.uetz@utwente.nl. Faculty of Electrical Engineering, Mathematics, and Computer Science, University of Twente, Enschede, The Netherlands

[§]llewin@dtu.dk. Department of Wind and Energy Systems, Technical University of Denmark, Kongens Lyngby, Denmark

We present the first non-trivial approximation results for fair repetitive scheduling on multiple machines, even in the very general unrelated parallel machine environment in which job parameters may depend on job, machine and day. Previous work on the fair repetitive scheduling problem has mainly focused on its computational complexity [4] as well as practical results [5] in a single machine environment without weights and release dates. Hermelin et al. [4] proved that problem is strongly NP-hard, even without weights. It remains weakly NP-hard even for $D \geq 4$ days. They complement these results with a polynomial-time algorithm for $D = 2$, and also give parameterized complexity results. Further, Plotkin et al. [5] present a polynomial-time algorithm for unit jobs, i.e., $p_{jd} = 1$ for all $d \in \mathcal{D}$ and $j \in \mathcal{J}$, and perform an experimental study for a MILP formulation and greedy heuristics. Hermelin et al. [3] present a 2-approximation based on a linear programming relaxation using daily completion times as decision variables and a PTAS for a constant number of days. Furthermore, they consider the day-invariant model ($p_{jd} = p_j$ for every $d \in \mathcal{D}$) and present a $((1 + \sqrt{2})/2 + \epsilon)$ -approximation as well as a Quasi-PTAS.

2 The job-invariant setting

We first focus on the *job-invariant setting*, in which processing times and weights depend solely on day and machine, i.e., $p_{jdi} = p_{di}$ and $w_{jd} = w_d$, for all $j \in \mathcal{J}$ and $d \in \mathcal{D}$. Although this is arguably a most basic variant of the model, it was recently shown to be NP-hard even on a single machine [4]. Our first main result is a PTAS even for uniformly related machines.

Theorem 1 (Informal) *There exists a polynomial-time approximation scheme for fair repetitive scheduling in the job-invariant setting on uniform machines.*

This result is based on a non-trivial reduction to the problem of makespan minimization on identical parallel machines with bag constraints, introduced by Das and Wiese [1]. Here, jobs are partitioned into bags and each machine is only allowed to process one job per bag. The key observation of our reduction is that we can restrict to a well-structured class of optimal schedules with a makespan-optimal schedule each day. Therefore, one can interpret our problem of fair repetitive scheduling as an assignment problem between jobs and time slots of such a well-structured schedule with the additional constraint that each job needs to be assigned to exactly one time slot per day. The idea of the reduction is to give jobs the role of machines and time slots the role of jobs and define a bag for each day which contains all time slots of this day.

3 The power of time-indexed LPs

When going beyond the job-invariant setting, we show that the time-indexed LP relaxation introduced by Dyer and Wolsey [2] for minimizing the sum of weighted

completion time proves to be powerful for fair repetitive scheduling on different machine environments. In particular, we show how to formulate it for fair repetitive scheduling and how to utilize different (known) rounding strategies such as slow-motion [6] and alpha-point scheduling [7] to obtain the following results.

Theorem 2 (Informal) *Fair repetitive scheduling can be approximated in polynomial-time with a factor of:*

- $(4 + \epsilon)$ for preemptive on unrelated machines with release dates.
- $(4 + \epsilon)$ for non-preemptive scheduling on identical parallel machines without release dates, and $(5.83 + \epsilon)$ with release dates.
- $(2 + \epsilon)$ on a single machine without release dates, and $(3 + \epsilon)$ with release dates both with and without preemption.

References

- [1] DAS, SYAMANTAK AND WIESE, ANDREAS (2017). *On Minimizing the Makespan When Some Jobs Cannot Be Assigned on the Same Machine*. 25th Annual European Symposium on Algorithms (ESA 2017). 31:1–31:14.
- [2] DYER, MARTIN E AND WOLSEY, LAURENCE A (1990). *Formulating the single machine sequencing problem with release dates as a mixed integer program*. Discrete Applied Mathematics, 26(2-3), 255-270.
- [3] DANNY HERMELIN AND DANNY SEGEV AND DVIR SHABTAY (2025). *Approximation Algorithms for Fair Repetitive Scheduling*. arXiv preprint arXiv:2512.25020.
- [4] DANNY HERMELIN AND HENDRIK MOLTER AND ROLF NIEDERMEIER AND MICHAEL PINEDO AND DVIR SHABTAY (2025). *Fairness in repetitive scheduling*. European Journal of Operational Research, 323(3), 724-738.
- [5] ANDREI PLOTKIN AND YALI FINK AND DVIR SHABTAY (2026). *Algorithms for fair repetitive scheduling based on total completion time criterion*. Computers & Industrial Engineering, 212, 111659.
- [6] ANDREAS S. SCHULZ AND MARTIN SKUTELLA (1997). *Random-Based Scheduling: New Approximations and LP Lower Bounds*. In International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM 2006) (pp. 119-133).
- [7] MARTIN SKUTELLA (2006). *List Scheduling in Order of α -Points on a Single Machine*. In Efficient Approximation and Online Algorithms: Recent Progress on Classical Combinatorial Optimization Problems and New Applications (pp. 250-291).

Combinatorial Perpetual Scheduling

Mirabel Mendoza-Cadena* Arturo Merino†
 Mads Anker Nielsen (Speaker)‡ Kevin Schewior§

Certain scheduling problems require tasks to be executed not only once but perpetually under certain restrictions on how frequently each task must/may be executed. These problems, often appealingly clean, include the pinwheel scheduling (PS) problem (and variants) [3, 7, 6, 10] and the bamboo-garden trimming (BGT) problem (and variants) [8, 5, 2].

To a large extent, the literature has only considered settings in which, at any time, only a single task can be scheduled. In many applications, however, it may be possible to schedule multiple tasks at the same time, subject to a combinatorial constraint. For instance, there may be multiple workers, each of which has the expertise to do a certain subset of the tasks. In this case, a set of tasks can be executed simultaneously if and only if it is the independent set of a transversal matroid.

In our paper [9], we introduce combinatorial variants of BGT and PS, which essentially unify the few combinatorial variants that have been considered in the literature. Namely, we introduce the *Combinatorial Bamboo Garden Trimming* (CBGT) problem. In this problem, there is a set E of bamboos, with each $e \in E$ having an individual *growth rate* $g(e)$. Furthermore, there is a family $\mathcal{I} \subseteq 2^E$ of schedulable subsets of E . The following two steps alternate ad infinitum: First, all bamboos grow according to their growth rates. Then, a set $I \in \mathcal{I}$ is selected and its elements are cut back down to 0. This process leads to an infinite *schedule* $\pi = I_1, I_2, \dots$, with $I_i \in \mathcal{I}$ for all $i \in \mathbb{N}$. The goal is to minimize the maximum height $h(\pi)$ that *ever* occurs among any of the bamboos.

The maximum height ever reached by any bamboo obviously depends not only on the schedule π but also on the “magnitude” of g . In particular, scaling g by a constant simply scales the height $h(\pi)$ of any schedule π by that same constant, and thus some normalization is in order. Here, we assume that g is scaled such that g is upper-bounded by a convex combination of the incidence vectors of the sets in \mathcal{I} . That is, there exist coefficients $\lambda(I) \geq 0$ for $I \in \mathcal{I}$ with $\sum_{I \in \mathcal{I}} \lambda(I) = 1$

*Centro de Modelamiento Matemático (CNRS IRL2807), Universidad de Chile, Santiago, Chile. lmendoza@cmm.uchile.cl.

†Department of Computer Science, University of Chile, Chile. amerino@dcc.uchile.cl.

‡Department of Mathematics and Computer Science, University of Cologne, Germany. m.nielsen@uni-koeln.de.

§Department of Mathematics and Computer Science, University of Cologne, Germany, and Department of Mathematics and Computer Science, University of Southern Denmark, Denmark. schewior@cs.uni-koeln.de.

such that $g \leq \sum_{I \in \mathcal{I}} \lambda(I) \cdot \mathbf{1}_I$ where $\mathbf{1}_I \in \{0, 1\}^E$ is the incidence vector of I . This normalization coincides with the standard assumption that $\sum_{e \in E} g(e) \leq 1$ for BGT.

We are primarily concerned with two aspects of CBGT instances. The first is: For a given family \mathcal{F} of set systems, what is the least h such that a schedule π with $h(\pi) \leq h$ always exists for CBGT instances on a set system from \mathcal{F} ? While this is purely an existential question, we are also interested in algorithms that *implement* such schedules in polynomial time. By this, we mean algorithms that output the set of bamboos to cut each day in an online fashion, using only polynomial time each day.

Contributions

Our main theorem is the following.

Theorem 1 *Any CBGT instance (E, \mathcal{I}, g) where (E, \mathcal{I}) is a matroid has a schedule π with $h(\pi) < 2$.*

Note that this upper bound is tight even if (E, \mathcal{I}) is restricted to 1-uniform matroids (BGT), as can be seen from the instance with $|E| = 2$ and growth rates $1 - \varepsilon, \varepsilon$ [3]. While our proof is constructive, the running time of the suggested algorithm is not polynomial. We therefore complement this result with

- a polynomial-time algorithm that implements a height-2 schedule for uniform matroids and
- polynomial-time algorithms that implement height-4 schedules for graphic and laminar matroids.

Our algorithm for uniform matroids is based on *tree schedules* [1], while the algorithms for graphic and laminar matroids are based on computing a suitable *rainbow-circuit-free* coloring (see e.g. [4]) of the ground set such that each color class can be scheduled in parallel.

Lastly, we also consider CBGT for arbitrary set systems. We obtain the following negative result.

Theorem 2 *For every $k \in \mathbb{N}$ there is a CBGT instance (E, \mathcal{I}, g) with $|E| = 2^k - 1$ for which every schedule has height at least $\frac{1}{2} \log_2(|E| + 1) - \frac{1}{2}$.*

We also obtain the following positive result. Note that the height guarantee is asymptotically best possible due to the previous theorem.

Theorem 3 *Any CBGT instance has a schedule π with $h(\pi) \in O(\log |E|)$. Furthermore, such a schedule can be implemented in polynomial time given access to a linear-optimization oracle over the convex hull of the incidence vectors of the sets in \mathcal{I} .*

References

- [1] A. Bar-Noy, B. Patt-Shamir, and V. Dreizin. Efficient periodic scheduling by trees. In *IEEE International Conference on Computer Communications (INFOCOM)*, pp. 791–800, 2002.
- [2] Y. Biktairov, L. Gasieniec, W. P. Jiamjitrak, Namrata, B. Smith, and S. Wild. Simple approximation algorithms for Polyamorous Scheduling. In *Symposium on Simplicity in Algorithms (SOSA)*, pp. 290–314, 2025.
- [3] L. Gasieniec et al. Perpetual maintenance of machines with different urgency requirements. *Journal of Computer and System Sciences*, 139:103476, 2024.
- [4] D. Hoffman, P. Horn, P. Johnson, and A. Owens. On rainbow-cycle-forbidding edge colorings of finite graphs. *Graphs and Combinatorics*, 35(6):1585–1596, 2019.
- [5] F. Höhne and R. van Stee. A $10/7$ -Approximation for Discrete Bamboo Garden Trimming and Continuous Trimming on Star Graphs. In *International Conference on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, 16:1–16:19, 2023.
- [6] S. Kanellopoulos, M. Kokkou, E. Markou, A. Pagourtzis, and C. Pergaminielis. Finite Pinwheel Scheduling: the k -Visits Problem. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 355–371, 2026.
- [7] A. Kawamura. Proof of the Density Threshold Conjecture for Pinwheel Scheduling. In *ACM Symposium on Theory of Computing (STOC)*, pp. 1816–1819, 2024.
- [8] J. Kuszmaul. Bamboo Trimming Revisited: Simple Algorithms Can Do Well Too. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pp. 411–417, 2022.
- [9] M. Mendoza-Cadena, A. Merino, M. A. Nielsen, and K. Schewior. Combinatorial Perpetual Scheduling. *arXiv:2602.11826*, 2026.
- [10] A. Mishra. An Optimal Density Bound for Discretized Point Patrolling. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 2846–2875, 2026.

Strengths and Limitations of Greedy in Cup Games

Kalina Jasińska*

John Kuszmaul (Speaker)[†]Gyudong Lee[‡]

1 Introduction

The *cup game* is a classical buffer-management scheduling problem, dating back to the late 1960s [30, 31], which is embodied as a two-player game played on n initially empty cups. During each time step, the *adversary* first distributes 1 unit of water among the cups arbitrarily. The *player* then selects a single cup, and is allowed to remove up to 1 unit of water from that cup. As we will see, in some variations, the player is allowed to remove more than 1 unit of water, but always just from a single cup. The player’s goal is to minimize the *backlog*, the supremum of the amount of water in the fullest cup over all time steps.

Cup games have found ample applications in areas including processor scheduling [1, 5–7, 9, 10, 14, 19, 22, 25–32], network-switch buffer management [4, 17, 20, 34], quality-of-service guarantees [1, 6, 28], and data-structure deamortization [2, 3, 8, 13, 14, 16, 21, 23, 33]. See [25] for a detailed discussion of the related work. Since cup games have arisen independently several times due to their natural modeling of many real-world phenomena in computer systems, they have also been studied under the monikers the *leaky-bucket problem* [1], *Cinderella versus the wicked stepmother* [12] and *bamboo garden trimming* [19] (as well as more restrained names such as *multiprogram scheduling* [31]).

Most variations of the cup game have strengthened the player’s hand [1, 5–7, 9, 10, 13, 18, 19, 22, 24, 27–32]. A first modification is to give the player *resource augmentation*, where the player is allowed to empty more than 1 unit of water. In the limit, the player is allowed to empty cups (reducing their height to 0) rather than only removing a single unit of water. This is known as the *cup flushing game* and has been studied for over 30 years [9, 13, 14, 28]. Additionally, a wide body of work has considered the scheduling problem resulting from the important special case where the adversary is restricted to its initial distribution of water for all subsequent time steps [5–7, 19, 22, 27, 29–32]; the fill rates of each cup are then fixed. This is known as the *fixed-rate cup game*. When both modifications are applied simultaneously, the result is the fixed-rate cup flushing game—this is also known in the literature as the *bamboo garden trimming problem*, introduced by [19].¹

*Cambridge University

[†]jhnkszml@mit.edu. MIT[‡]MIT¹The name stems from a rather creative metaphor in which n bamboos grow, each at a fixed rate, and a gardener (who happens to be a panda) selects 1 bamboo each day to chop down.

The **greedy** algorithm, which always selects the fullest cup, has been a central research topic in every variation of the cup game. **Greedy** is known to achieve optimal backlog $\Theta(\log n)$ in the cup flushing game due to a seminal 1987 paper [13] by Dietz and Sleator. Interestingly, this tight $\Theta(\log n)$ bound on backlog is robust, regardless of how much resource augmentation the player is given. Their proof [13], in under half a page, uses an elegant family of invariants, and can be extended to the canonical cup game, as was independently discovered by [1]. These works show that in both settings the **greedy** algorithm achieves the exact optimal backlog² of $H(n)+1$, where $H(n)$ denotes the n th harmonic number, i.e., $H(n) = \sum_{i=1}^n 1/i$. Remarkably, this past work demonstrates that there is no separation between the optimal worst-case backlog of the cup game and the cup flushing game.

The **greedy** algorithm has also received considerable attention in the bamboo setting, where the optimal backlog for any algorithm is known to be exactly 2 [5–7, 19, 22, 27, 29–32]. D’Emidio, Di Stefano, and Navarra [15] conjecture, with the support of an extensive computer search, that **greedy** is optimal in this setting, achieving backlog 2. If true, this would make **greedy** optimal for both the bamboo trimming problem and the cup game. Considerable effort has been made towards proving D’Emidio et al.’s conjecture, and the best known upper bound in this setting was improved from the baseline of $O(\log n)$ [1, 13] to 9 [11], and then to the current best bound of 4 [24]. Nonetheless, in this paper, we show that the conjecture is actually false – **greedy** does not achieve backlog 2.

We additionally study a new model, the **semi-oblivious cup game**, which captures uncertainty in the player’s estimate of the height of each cup, thus strengthening the adversary’s hand. In contrast to most well-studied variations of the cup game, the semi-oblivious cup game weakens the player. Each time step t , rather than being informed of the exact height $f_j(t)$ of each cup j , the player instead receives an estimate $f'_j(t)$ of the height of each cup t . This estimate must satisfy $f'_j(t) \in [f_j(t), cf_j(t)]$. The **greedy** algorithm simply selects the cup with the maximal $f'_j(t)$. By analyzing the semi-oblivious model, we can gain insight into settings in which the player is given imperfect information regarding the cup heights (i.e., the processor has imperfect information regarding the amount of work in each buffer), and thus the player may end up selecting a cup that is off by at most a constant factor.

In the semi-oblivious cup game model, the **greedy** algorithm always removes water from cups whose height is within an $O(1)$ factor of the fullest cup. What is unclear is how this affects the game. How important is having perfect information for achieving good backlog bounds? Does **greedy** continue to achieve a backlog of $O(\log n)$, or do the guarantees of the algorithm fundamentally require access to high-fidelity measurements of how much water is in the cups?

In summary, our work explores the limitations of the **greedy** algorithm in both the most favorable cup game setting for the player, i.e., the bamboo trimming problem, and in one of the least favorable settings, i.e., the semi-oblivious setting. In doing so, we obtain new upper and lower bounds that extend our understanding of the strengths and limitations of the greedy algorithm, as well as the types of bounds that are possible in each setting.

²Note that the additive 1 disappears if the amount of water in a cup is allowed to be negative.

1.1 Our Results

Our first result is a tight analysis of the **greedy** algorithm in the semi-oblivious cup game. In this setting, the **greedy** algorithm always selects a cup whose height is within a constant multiplicative error c of the fullest cup, and removes up to 1 unit of water from it. We show that **greedy** performs much worse in this setting than in the classical version of the game. In particular, we prove matching upper and lower bounds of the form $\Theta(n^{\frac{c-1}{c}})$ on the algorithm's backlog.

A natural question is whether the bound achieved by semi-oblivious **greedy** might improve with the help of resource augmentation, i.e., in the *semi-oblivious cup flushing game*, where the player removes all of the water from some cup on each step. Our second result shows that resource augmentation does, in fact, make a difference, allowing **greedy** to achieve a sub-polynomial bound on backlog. Specifically, we prove tight upper and lower bounds of the form $2^{\Theta(\sqrt{\log n})}$ on the backlog that **greedy** achieves in the semi-oblivious cup flushing game.

These results demonstrate a separation between the performance of the **greedy** algorithm in the semi-oblivious cup game with and without resource augmentation. This contrasts with results for the canonical cup game, where resource augmentation has no effect on the optimal backlog [1, 13].

These results may give the impression that the $O(\log n)$ bound achieved by **greedy** in the canonical cup game is actually quite fragile. Does **greedy** actually require perfect information to achieve the classic $O(\log n)$ backlog guarantee? To this end, we consider an additive-error version of the semi-oblivious cup game, in which the adversary's estimates are constrained by $f'_j(t) \in [f_j(t), c + f_j(t)]$. Past work [9] established that **greedy** achieves $O(\log n)$ backlog in the cup flushing game, even with $O(1)$ additive error. For the standard (non-flushing) version of the game, we find matching upper and lower bounds on **greedy**'s performance in the additive error setting of

$$(c+1)\ln m \pm \Theta(1) = \Theta(\log n).$$

This result is especially exciting since the known techniques for the cup flushing game, i.e., the potential-function style arguments of [9], do not yield anything non-trivial in this setting. Our backlog bound can be viewed as a robustness result, demonstrating that **greedy** remains resilient against additive errors in the canonical cup game, but that its resiliency decays directly proportionally to the quantity $1+c$.

All of our lower bounds in the semi-oblivious cup game extend to apply not just against the **greedy** algorithm, but against any deterministic algorithm. Thus, as an immediate (and non-trivial) corollary, we get that **greedy** is asymptotically optimal in all of the above settings.

Our final result revisits the behavior of **greedy** in the bamboo flushing version of the cup game. Here, it is conjectured [11, 15, 24] that **greedy** achieves the optimal backlog, which in this version of the game is 2. We show that this conjecture is not true, giving a lower bound of 2.076. We also present a simple **Deadline-Driven/Greedy** hybrid algorithm that performs optimally across both the bamboo trimming problem and several variants of the cup game, always achieving a backlog within a $1+o(1)$ factor of optimal

References

- [1] M. Adler, P. Berenbrink, T. Friedetzky, L. A. Goldberg, P. Goldberg, and M. Paterson. A proportionate fair scheduling rule with good worst-case performance. In *Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 101–108, 2003.
- [2] A. Amir, M. Farach, R. M. Idury, J. A. L. Poutré, and A. A. Schäffer. Improved dynamic dictionary matching. *Inf. Comput.*, 119(2):258–282, 1995.
- [3] A. Amir, G. Franceschini, R. Grossi, T. Kopelowitz, M. Lewenstein, and N. Lewenstein. Managing unbounded-length keys in comparison-driven data structures with applications to online indexing. *SIAM J. Comput.*, 43(4):1396–1416, 2014.
- [4] Y. Azar and A. Litichevsky. Maximizing throughput in multi-queue switches. *Algorithmica*, 45(1):69–90, 2006.
- [5] A. Bar-Noy, A. Nisgav, and B. Patt-Shamir. Nearly optimal perfectly periodic schedules. *Distributed Comput.*, 15(4):207–220, 2002.
- [6] S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel. Proportionate progress: A notion of fairness in resource allocation. *Algorithmica*, 15(6):600–625, Jun 1996.
- [7] S. K. Baruah, J. Gehrke, and C. G. Plaxton. Fast scheduling of periodic tasks on multiple resources. In *Proceedings of IPSPS '95, The 9th International Parallel Processing Symposium, April 25-28, 1995, Santa Barbara, California, USA*, pages 280–288. IEEE Computer Society, 1995.
- [8] M. A. Bender, R. Das, M. Farach-Colton, R. Johnson, and W. Kuszmaul. Flushing without cascades. In S. Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 650–669. SIAM, 2020.
- [9] M. A. Bender, M. Farach-Colton, and W. Kuszmaul. Achieving optimal backlog in multi-processor cup games. In M. Charikar and E. Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1148–1157. ACM, 2019.
- [10] M. A. Bender and W. Kuszmaul. Randomized cup game algorithms against strong adversaries. In D. Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2059–2077. SIAM, 2021.
- [11] D. Bilò, L. Gualà, S. Leucci, G. Proietti, and G. Scornavacca. Cutting bamboo down to size. *Theoretical Computer Science*, 2022.
- [12] M. H. L. Bodlaender, C. A. J. Hurkens, V. J. J. Kusters, F. Staals, G. J. Woeginger, and H. Zantema. Cinderella versus the wicked stepmother. In *IFIP International Conference on Theoretical Computer Science*, pages 57–71, 2012.

- [13] P. Dietz and D. Sleator. Two algorithms for maintaining order in a list. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing (STOC)*, pages 365–372, 1987.
- [14] P. F. Dietz and R. Raman. Persistence, amortization and randomization. In *Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 78–88, 1991.
- [15] M. D’Emidio, G. Di Stefano, and A. Navarra. Bamboo garden trimming problem: Priority schedulings. *Algorithms*, 12(4):74, 2019.
- [16] J. Fischer and P. Gawrychowski. Alphabet-dependent string searching with wexponential search trees. In F. Cicalese, E. Porat, and U. Vaccaro, editors, *Combinatorial Pattern Matching - 26th Annual Symposium, CPM 2015, Ischia Island, Italy, June 29 - July 1, 2015, Proceedings*, volume 9133 of *Lecture Notes in Computer Science*, pages 160–171. Springer, 2015.
- [17] H. R. Gail, G. A. Grover, R. Guérin, S. L. Hantler, Z. Rosberg, and M. Sidi. Buffer size requirements under longest queue first. *Perform. Evaluation*, 18(2):133–140, 1993.
- [18] L. Gaşieniec, T. Jurdziński, R. Klasing, C. Levkopoulos, A. Lingas, J. Min, and T. Radzik. Perpetual maintenance of machines with different urgency requirements. *Journal of Computer and System Sciences*, 139:103476, 2024.
- [19] L. Gaşieniec, R. Klasing, C. Levkopoulos, A. Lingas, J. Min, and T. Radzik. Bamboo garden trimming problem (perpetual maintenance of machines with different attendance urgency factors). In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 229–240. Springer, 2017.
- [20] M. H. Goldwasser. A survey of buffer management policies for packet switches. *SIGACT News*, 41(1):100–128, 2010.
- [21] M. T. Goodrich and P. Pszona. Streamed graph drawing and the file maintenance problem. In S. K. Wismath and A. Wolff, editors, *Graph Drawing - 21st International Symposium, GD 2013, Bordeaux, France, September 23-25, 2013, Revised Selected Papers*, volume 8242 of *Lecture Notes in Computer Science*, pages 256–267. Springer, 2013.
- [22] N. Guan and W. Yi. Fixed-priority multiprocessor scheduling: Critical instant, response time and utilization bound. In *26th IEEE International Parallel and Distributed Processing Symposium Workshops & PhD Forum, IPDPS 2012, Shanghai, China, May 21-25, 2012*, pages 2470–2473. IEEE Computer Society, 2012.
- [23] T. Kopelowitz. On-line indexing for general alphabets via predecessor queries on subsets of an ordered list. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 283–292. IEEE Computer Society, 2012.

- [24] J. Kuszmaul. Bamboo trimming revisited: Simple algorithms can do well too. In *Proceedings of the 34th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA'22, page 411–417, New York, NY, USA, 2022. Association for Computing Machinery.
- [25] W. Kuszmaul. Achieving optimal backlog in the vanilla multi-processor cup game. In S. Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1558–1577. SIAM, 2020.
- [26] W. Kuszmaul. How asymmetry helps buffer management: achieving optimal tail size in cup games. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1248–1261, 2021.
- [27] A. Litman and S. Moran-Schein. On distributed smooth scheduling. In P. B. Gibbons and P. G. Spirakis, editors, *SPAA 2005: Proceedings of the 17th Annual ACM Symposium on Parallelism in Algorithms and Architectures, July 18-20, 2005, Las Vegas, Nevada, USA*, pages 76–85. ACM, 2005.
- [28] A. Litman and S. Moran-Schein. Smooth scheduling under variable rates or the analog-digital confinement game. *Theor. Comp. Sys.*, 45(2):325–354, June 2009.
- [29] A. Litman and S. Moran-Schein. On centralized smooth scheduling. *Algorithmica*, 60(2):464–480, 2011.
- [30] C. L. Liu. Scheduling algorithms for multiprocessors in a hard real-time environment. *JPL Space Programs Summary*, 1969, 1969.
- [31] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61, 1973.
- [32] M. Moir and S. Ramamurthy. Pfair scheduling of fixed and migrating periodic tasks on multiple resources. In *Proceedings of the 20th IEEE Real-Time Systems Symposium, Phoenix, AZ, USA, December 1-3, 1999*, pages 294–303. IEEE Computer Society, 1999.
- [33] C. W. Mortensen. Fully-dynamic two dimensional orthogonal range and line segment intersection reporting in logarithmic time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA*, pages 618–627. ACM/SIAM, 2003.
- [34] M. Rosenblum, M. X. Goemans, and V. Tarokh. Universal bounds on buffer size for packetizing fluid policies in input queued, crossbar switches. In *Proceedings IEEE INFOCOM 2004, The 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, Hong Kong, China, March 7-11, 2004*, pages 1126–1134. IEEE, 2004.

Refining the Complexity Landscape of Speed Scaling: Hardness and Algorithms *

Antonios Antoniadis [†] Denise Graafsma (speaker) [‡]
 Ruben Hoeksma [§] Maria Vlasiou [¶]

1 Introduction

We study the computational complexity of scheduling jobs on a single speed-scalable processor with the objective of capturing the trade-off between the (weighted) flow time and the energy consumption. This trade-off has been extensively explored in the literature through a number of problem formulations that differ in the specific job characteristics and the precise objective function. Nevertheless, the computational complexity of four important problem variants has remained unresolved and was explicitly identified as an open question in prior work (see [4]). We settle the complexity of these variants.

Consider the following scheduling problem: Given a set of n preemptable jobs, each job j is associated with a *release time* r_j , a *processing volume* v_j , and a *weight* w_j . The jobs are to be processed on a speed-scalable processor and the power-consumption corresponding to each allowable speed is given by a power function P . The set of different allowable speeds can be discrete or continuous (in the latter case, P is usually considered to be a continuous convex function of the speed). When the processor runs at a *speed* s , it can process s units of volume per unit of time. Naturally, the energy consumption of the processor is defined by the integral over time of $P(s(t))$, where $s(t)$ denotes the speed of the processor at time t . A schedule determines which job is processed at what speed at each time t , so that each job j is fully processed after its respective release time r_j . As a QoS objective, we consider the total (weighted) *flow time* (also known as *response time*) of a schedule, that is, the (weighted) sum over all jobs of the difference between the time their processing is completed and the time they were released.

We adopt the quintuple notation $\star\star\star\star$ from [4] to describe different variants of the problem related to this work: The first entry denotes whether the objective

*This work has been accepted for publication at STACS 2026. A full version can be found at <https://doi.org/10.48550/arXiv.2512.17663>

[†]a.antoniadis@utwente.nl University of Twente

[‡]d.f.graafsma@utwente.nl University of Twente

[§]r.p.hoeksma@utwente.nl University of Twente

[¶]m.vlasiou@utwente.nl University of Twente

is the total (weighted) *Flow* plus *Energy* or the total (weighted) flow time subject to an energy *Budget* (think, e.g., of a battery-powered device). The second entry distinguishes whether the flow is *Integral* or *Fractional*¹, the third whether the speed is *Continuous* or *Discrete*, the fourth whether jobs are *Weighted* or *Unweighted* (i.e., all weights are one), and the fifth and final entry whether the sizes are *Arbitrary* or *Unit*.

The combination of speed scaling with a flow time objective was first explored by Pruhs et al. [7] who presented a polynomial-time algorithm for the B-ICUU variant of the problem. The combined objective, which is also the focus of this work, was introduced by Albers and Fujiwara [1]. They studied the unit-size job setting and proposed a polynomial-time algorithm based on dynamic programming for FE-ICUU, which can also be extended to the energy-budget variant. With respect to the fractional flow objective, Antoniadis et al. [2] showed that FE-FDWA is solvable in polynomial time by an incremental algorithm. On the other hand, the variants \star -I \star WA are known to be NP-hard already in the fixed-speed setting [5], while Megow and Verschae [6] show the NP-hardness of B-IDWA. Finally, Barcelo et al. [4] settle the computational complexity of variants B-IDUA, B-IDWU, FE-IDUU, and FE-FCWA: The first two are NP-hard whereas the latter two are solvable in polynomial time. Four variants remained open: FE-IDUA, FE-ICUA, FE-IDWU, FE-ICWU (see also [4] and the open problem discussion in [3]). Our first main contribution is to resolve those cases by proving that all four variants are NP-hard.

Theorem 1. *FE-IDUA is NP-hard.*

Theorem 2. *FE-IDWU is NP-hard.*

The proof of Theorem 1 consists of a reduction from the budget variant of the problem B-IDUA with two available speeds, which is known to be NP-hard [4]. This reduction crucially depends on a job of comparatively large volume, something that the FE-IDWU variant does not allow for. As a consequence, the proof of Theorem 2 requires a more technically involved argument. It consists of a reduction from the subset sum problem, which is similar to the NP-hardness proof of B-IDWU from [4].

Note that while the discrete setting only allows for a fixed set of speeds, intermediate speeds can be achieved by interpolation, resulting in a piecewise linear power function. Hence, the discrete setting can be considered as a special case of the continuous setting. As a result, both NP-hardness results can be extended to the respective continuous-speed problems. Therefore, Theorems 1 and 2 resolve the complexity of the last four unresolved variants and complete the computational complexity landscape of flow-plus-energy and flow under an energy-budget problems.

We extend the quintuple notation by the suffix -C, indicating that feasible schedules must adhere to a given completion-time ordering. To be more precise:

¹In this work we only consider the, more common, integral flow – but keep the tuple entry nevertheless for reasons of consistency.

if $j \prec j'$ in the ordering, then j' cannot be completed before j is completed. Regarding this setting, we obtain the following result:

Theorem 3. *There is a polynomial-time algorithm that computes optimal schedules for all \star -ID \star -C variants.*

The polynomial-time algorithm is obtained by formulating a linear program that can be adapted to any \star -ID \star -C variant. In addition to providing efficient algorithms, Theorem 3 allows for a better understanding of the interplay between deciding on the job order and on the processor speed at each time. As a consequence, our results provide more insight into where the complexity of the problem stems from. In particular, when combined with our previous NP-hardness results, Theorem 3 directly implies that for \star -IDUA, \star -IDWU and \star -IDWA it is NP-hard to compute an optimal completion-time ordering.

References

- [1] Susanne Albers and Hiroshi Fujiwara. “Energy-efficient algorithms for flow time minimization”. In: *ACM Trans. Algorithms* 3.4 (2007), p. 49. DOI: 10.1145/1290672.1290686. URL: <https://doi.org/10.1145/1290672.1290686>.
- [2] Antonios Antoniadis et al. “Efficient Computation of Optimal Energy and Fractional Weighted Flow Trade-Off Schedules”. In: *Algorithmica* 79.2 (2017), pp. 568–597. DOI: 10.1007/S00453-016-0208-X.
- [3] Neal Barcelo. “The Complexity of Speed-Scaling”. PhD thesis. University of Pittsburgh, Sept. 2015. URL: <https://d-scholarship.pitt.edu/25280/>.
- [4] Neal Barcelo et al. “On the Complexity of Speed Scaling”. In: *Proceedings of 40th Symposium on Mathematical Foundations of Computer Science (MFCS 2015)*. 2015, pp. 75–89. DOI: 10.1007/978-3-662-48054-0_7.
- [5] Jacques Labetoulle et al. “Preemptive Scheduling of Uniform Machines Subject to Release Dates”. In: *Progress in Combinatorial Optimization*. Academic Press, 1984, pp. 245–261. ISBN: 978-0-12-566780-7. DOI: <https://doi.org/10.1016/B978-0-12-566780-7.50020-9>.
- [6] Nicole Megow and José Verschae. “Dual Techniques for Scheduling on a Machine with Varying Speed”. In: *SIAM J. Discret. Math.* 32.3 (2018), pp. 1541–1571. DOI: 10.1137/16M105589X. URL: <https://doi.org/10.1137/16M105589X>.
- [7] Kirk Pruhs, Patchrawat Uthaisombut, and Gerhard J. Woeginger. “Getting the best response for your erg”. In: *ACM Trans. Algorithms* 4.3 (2008), 38:1–38:17. DOI: 10.1145/1367064.1367078. URL: <https://doi.org/10.1145/1367064.1367078>.

Resource leveling problems with precedence constraints and convex cost functions

Péter Györgyi *

Tamás Kis (Speaker) †

1 Introduction

Resource leveling is a common challenge in project scheduling that involves balancing the demand for shared resources across multiple projects. The main objective is to keep workload levels steady, which is the key focus of this paper. Because of the problem's complexity, using polynomial-time optimization or approximation algorithms are rare, resulting in a reliance on exact methods and heuristics for solutions [1].

Resource leveling problems occur when it is advantageous to smooth out fluctuations in resource usage over time, all while ensuring the project is completed within its designated deadline. The problem has several practical applications, e.g., in construction industry [2] and in chemical engineering [3].

In this paper we study two variants of the following general resource leveling problem. Formally, there is a set of n jobs \mathcal{J} , and a common resource. Each job $j \in \mathcal{J}$ has a positive integer processing time p_j , and a resource requirement $a_j \geq 0$. There is a precedence relation on the set of jobs \mathcal{J} . Moreover, d is a common deadline of all the jobs and f is an arbitrary convex function that describes the cost of the resource usage. A schedule S specifies a set of disjoint time intervals for each job j when it is processed with total length p_j . Let S_j and C_j denote the starting time and completion time of job j , respectively. A schedule is feasible if $C_j \leq d$ for each jobs j , and the precedence constraints are respected, i.e., $C_i \leq S_j$ whenever job i precedes job j in the precedence relation. The cost of a schedule S is $\text{cost}_f(S) := \sum_{t=0}^{d-1} f(\ell_t^S)$, where $\ell_t^S = \sum_{j \in \mathcal{J}_t} a_j$, \mathcal{J}_t being the set of jobs processed in the time interval $[t, t + 1]$ in schedule S . The goal is to find a feasible schedule of minimum cost.

2 No machines, $p_j = a_j = 1$ and in-trees

In this section, we consider unit-time jobs, each one requiring one unit from a common resource. We assume that the precedence graph decomposes into a set of

*gyorgyi.peter@sztaki.hu. HUN-REN SZTAKI, Kende Str. 13-17, 1111 Budapest, Hungary.

†kis.tamas@sztaki.hu. HUN-REN SZTAKI, Kende Str. 13-17, 1111 Budapest, Hungary.

in-trees.

We emphasize that f can be any convex function, which permits to model some popular objective functions in resource leveling. For instance, using $f(\ell) = \max\{0, \ell - c\}$, we can penalize resource usage exceeding a threshold c . If $f(\ell) = (\ell - c)^2$, then the squared deviation from c is minimized, which is equivalent to minimizing $\sum_t (\ell_t^S)^2$.

The *height* of job j , denoted by $h(j)$ is the length of the path between j and the root of its in-tree in the precedence graph. Let $r = \max_j h(j)$ be the maximum job height. Let $V_i := \{j : h(j) = i\}$ for $i = 0, \dots, r$. The *scheduling graph* G has a set of nodes $N = \{0\} \cup \{d-r, \dots, d\}$ and a set of edges $E = \{(a, b) \in N \times N \mid a < b\}$. The set of jobs $\mathcal{J}_{(t,u)}$ corresponding to an edge (t, u) of G is $\bigcup_{i=x}^y V_i$, where $x = d - u$, and $y = r$ if $t = 0$, and $y = d - t - 1$ otherwise. For any edge (t, u) of G , a *profile* is a mapping $p : \{t, \dots, u - 1\} \rightarrow \mathbb{Z}_{>0}$ such that $\sum_{\tau=t}^{u-1} p(\tau) = |\mathcal{J}_{(t,u)}|$ and $p(t) \geq p(t+1) \geq \dots \geq p(u-1)$. An *ideal profile* $p^{(t,u)}$ for edge (t, u) is one with $p_t^{(t,u)} = \lceil \alpha_{(t,u)} \rceil$ and $p_{u-1}^{(t,u)} = \lfloor \alpha_{(t,u)} \rfloor$, where $\alpha_{(t,u)} = |\mathcal{J}_{(t,u)}| / (u - t)$. We consider only those edges of G where the jobs in $\mathcal{J}_{(t,u)}$ can be scheduled in the time interval $[t, u]$ such that $p_\tau^{(t,u)}$ jobs are started at τ for $\tau = t, \dots, u - 1$. These edges are termed *ideal*. To verify this condition, we can use a variant of Hu's classical algorithm [4] for $P|p_j = 1, \text{in-tree}|C_{\max}$. Note that the ideal profile minimizes the cost of scheduling the jobs in $\mathcal{J}_{(t,u)}$ in the time interval $[t, u]$.

One can prove that there exists an optimal schedule which corresponds to a least cost path P from node 0 to node d of G consisting of ideal edges only. From P , we construct the optimal schedule by concatenating partial schedules corresponding to the edges of this path.

Theorem 1 *We can get an optimal solution of the resource leveling problem on a set of in-trees with a convex cost function in $O(r^3 + r \cdot n \log n)$ time.*

3 Parallel machines, arbitrary precedence constraints

In this section we consider parallel machine scheduling problems with unit-time or preemptive jobs and arbitrary precedence constraints between the jobs, as well as shop scheduling problems with unit-time or preemptive jobs. The resource requirements of the jobs can be specified by arbitrary non-negative numbers. The cost of a schedule S is defined as in the previous section. Throughout this section we assume that $f(x) := f_\alpha(x) = x^\alpha$ for some integer $\alpha \geq 1$.

Since it is NP-hard to decide if a feasible schedule with a given makespan exists, our goal is to modify a feasible schedule by rescheduling some of the jobs to later time points such that the makespan is at most doubled, while we get a good approximation ratio for our cost function. Let $\text{cost}_{f_\alpha}^*(T)$ be the least cost of a feasible schedule with makespan at most T .

Theorem 2 *For any $\alpha \geq 1$ and feasible schedule S for $P|prec, p_j = 1, res|cost_{f_\alpha}$, we can determine a feasible schedule S^b in $O(n \log n)$ time such that $\text{cost}_{f_\alpha}(S^b) <$*

$\rho(\alpha) \cdot \text{cost}_{f_\alpha}^*(C_{\max}(S))$ and $C_{\max}(S^b) \leq 2 \cdot C_{\max}(S)$, where $\rho(\alpha)$ is a given constant defined by [5].

The next table depicts some values of $\rho(\alpha)$:

α	$\rho(\alpha)$
1	1
2	1.042
3	1.084
4	1.127
∞	2.25.

We can generalize this result to other problems as follows.

Theorem 3 Consider an arbitrary instance I of $P|prec, pmtn, res|cost_{f_\alpha}$ or that of a shop problem (flowshop, jobshop, openshop) with unit processing time tasks or preemptive tasks with integer processing times. Let S be a feasible schedule for I . For any $\alpha \geq 1$, one can find in $O(n \log n)$ time a (preemptive) feasible schedule S^b such that $\text{cost}_{f_\alpha}(S^b) \leq \rho(\alpha) \cdot \text{cost}_{f_\alpha}^*(C_{\max}(S))$ and $C_{\max}(S^b) \leq 2 \cdot C_{\max}(S)$.

4 Acknowledgment

This project has received funding from the European Union's Horizon Europe programme under grant agreement No. 101137954. The authors would like to thank the rest of the BATTwin consortium for supporting this research.

References

- [1] S. HARTMANN AND D. BIRSKORN (2022). *An updated survey of variants and extensions of the resource-constrained project scheduling problem*, European Journal of Operational Research, Vol. 297(1), pp. 1-14.
- [2] S. KRETER, J. RIECK AND J. ZIMMERMANN (2014). *The total adjustment cost problem: Applications, models, and solution algorithms*, Journal of Scheduling, 17(2), pp. 145-160.
- [3] N. MEGOW, R.H. MÖHRING AND J. SCHULZ (2011). *Decision support and optimization in shutdown and turnaround scheduling*, INFORMS Journal on Computing, 23(2), pp. 189-204.
- [4] T.C. HU (1961). *Parallel sequencing and assembly line problems*, Operations Research, Vol. 01, pp. 1-5.
- [5] A. K. CHANDRA AND C.-K. WONG (1975). *Worst-case analysis of a placement algorithm related to storage allocation*, SIAM Journal on Computing, 4(3), pp. 249-263.

A Practical 73/50 Approximation for Contiguous Monotone Moldable Job Scheduling

Klaus Jansen ^{*} Felix Ohnesorge (Speaker) [†]

1 Introduction

Motivated by many practical applications, such as high performance computing, where parallelism is exploited to speed up the execution of jobs, many extensions of the classical job scheduling problem have been studied. One of which is scheduling *moldable* jobs. This is a natural extension where jobs can be executed on a variable number of machines.

In moldable job scheduling we are given $J = \{1, \dots, n\}$ jobs and m identical machines. Each job can be assigned to $k \leq m$ machines and the processing time of any job j is dependent on the number of assigned machines. We denote the processing time of any job j as $t(j, k)$. Further, the *work* of any job j is defined as $w(j, k) = t(j, k) \cdot k$ and can be described as its area. For *monotone* moldable job scheduling we assume that for any job, the work function and time function are non-decreasing and non-increasing, respectively. Specifically, we have for each any job j and $k \leq k' \leq m$: (1) $w(j, k) \leq w(j, k')$ and (2) $t(j, k) \geq t(j, k')$.

A solution to an instance of this problem is given by a *schedule*, containing a start time s_j for each job $j \in J$, and an allotment $\alpha(j) \in \{1, \dots, m\}$ for each job. A schedule is called *feasible* if: (1) Each job starts its execution simultaneously on all its assigned machines, (2) No job may be interrupted during its execution time, and (3) Each machine executes at most one job at a time.

In the *contiguous* variant of the problem, we additionally require each job to be processed on an adjacent set of machines. In many practical applications obtaining such a schedule, where jobs are processed on adjacent machines, is beneficial as it allows for better memory locality and reduced communication overhead between machines [9].

1.1 Related Work

It has been proven that for moldable job scheduling (without monotony!) no polynomial-time approximation algorithm with a guarantee below $3/2$ exists, unless $P = NP$ [7, 8]. But there exists a pseudo-polynomial $(\frac{5}{4} + \varepsilon)$ -approximation

^{*}kj@informatik.uni-kiel.de. Kiel University

[†]foh@informatik.uni-kiel.de. Kiel University

algorithm for scheduling contiguous moldable jobs with a time complexity of $O(n \log n) \cdot m^{f(1/\varepsilon)}$, where $f(\cdot)$ is a computable function [6].

We focus on *monotone* moldable job scheduling. The best known approximation ratio is $1 + \varepsilon$, given by the Polynomial Time Approximation Scheme (PTAS) in [5]. When using $\varepsilon = \frac{1}{2}$, the running time of this PTAS is roughly $nm^{2 \cdot 10^{2.4 \cdot 10^{17}}}$. When comparing the $(\frac{3}{4} + \varepsilon)$ -approximation in [6], to current $\frac{3}{2}$ -approximation algorithms by setting $\varepsilon := \frac{1}{4}$, we get a running time of $\Omega((mn)^{16^{4^{13}}})$. While these algorithms are of theoretical interest, they have an impractically large running time. Motivated by this, we focus on *practically efficient* approximation algorithms, that are polynomial in n, m , and $\frac{1}{\varepsilon}$ (Table 1).

Ratio	Complexity	Author (Year)
$\frac{3}{2} + \varepsilon$	$O(\log(1/\varepsilon)nm)$	Mounié, Rapine, Trystram
$\frac{3}{2} + \varepsilon$	$O(\frac{n}{\varepsilon^2} \log m (\frac{\log m}{\varepsilon} + \log^3(\varepsilon m)))$	Jansen, Land (2018) [2]
$\frac{3}{2} + \varepsilon$	$O(n \log^2(\frac{1+\log(\varepsilon m)}{\varepsilon}) + \frac{n}{\varepsilon} \log(\frac{1}{\varepsilon}) \log(\varepsilon m))$	Grage, Jansen, Ohnesorge
$\frac{3}{2}$	$O(nm \log(nm))$	Wu, Zhang, Chen (2023) [
$1.4593 + \varepsilon$	$O(\log(1/\varepsilon)nm)$	This Work

Table 1: Overview of approximation algorithms for monotone moldable job scheduling polynomial in n, m and $1/\varepsilon$.

1.2 Our Contributions

Previously it was assumed that there exists no practically efficient $(\frac{3}{2} - \varepsilon)$ -approximation for the problem of monotone moldable job scheduling, for any $\varepsilon > 0$ [4]. We break this barrier by presenting an approximation algorithm with a worst-case approximation ratio of $\approx 1.4593 < 1.5$. It is worth noting that the PTAS presented in [5] cannot solve the contiguous variant of the problem, and it is unknown whether there exists a PTAS for this variant of the problem. Surprisingly, we show that our algorithm creates a contiguous schedule with a makespan of at most $1.4593 \cdot \widetilde{OPT}$, where \widetilde{OPT} is a lower bound on the optimum of the non-contiguous problem. This result also bounds the gap between the contiguous and non-contiguous variant of the problem.

Theorem 1 *Let $\varepsilon > 0$. For contiguous monotone moldable job scheduling, there exists an algorithm with an approximation ratio of $(1.4593 + \varepsilon)$ and a running time of $O(nm \log \frac{1}{\varepsilon})$.*

We achieve this result by improving the long-standing algorithm of Mounié, Rapine, and Trystram [1]. Although this algorithm has been improved many times with respect to the running time [2, 3], improving the approximation ratio seemed to be a difficult task (except for eliminating the ε factor [4]). We manage this

with the following new techniques: (1) relaxing the standard 0/1-Knapsack used in their algorithm to a Multiple-Choice Knapsack Problem, (2) packing the jobs into more containers, which results in more complex repair algorithms, and (3) analyzing a single critical case carefully with an elegant geometric argument. Our method of analyzing this critical case results in the so-called Lambert W function appearing in the approximation ratio.

A similar geometric analysis could be beneficial in other scheduling and packing problems. Furthermore, our theoretical results are supported by practical experiments demonstrating that a schedule length of at most $\frac{10}{7}OPT$ can be guaranteed in most cases. This makes this algorithm a promising candidate for practical applications.

The full paper is available under <https://arxiv.org/abs/2601.02836> and will appear in the 43rd International Symposium on Theoretical Aspects of Computer Science (STACS 2026).

References

- [1] G. MOUNIE, C. RAPINE AND D. TRYSTRAM (2007). *A 3/2-Approximation Algorithm for Scheduling Independent Monotonic Malleable Tasks*. SIAM J. Comput.
- [2] K. JANSEN AND F. LAND (2018). *Scheduling Monotone Moldable Jobs in Linear Time*. IPDPS.
- [3] K. GRAGE, K. JANSEN AND F. OHNESORGE (2023). *Improved Algorithms for Monotone Moldable Job Scheduling Using Compression and Convolution*. Euro-Par.
- [4] F. WU AND X. ZHANG AND B. CHEN (2023). *An improved approximation algorithm for scheduling monotonic moldable tasks*. Eur. J. Oper. Res.
- [5] K. JANSEN AND R. THÖLE (2010). *Approximation Algorithms for Scheduling Parallel Jobs*. SIAM J. Comput.
- [6] K. JANSEN AND M. RAU (2019). *Closing the Gap for Pseudo-Polynomial Strip Packing*. ESA.
- [7] M. DROZDOWSKI (1995). *On the complexity of multiprocessor task scheduling*. Bulletin of The Polish Academy of Sciences-technical Sciences.
- [8] B. JOHANNES (2006). *Scheduling parallel jobs to minimize the makespan*. J. Sched.
- [9] R. BLEUSE, S. HUNOLD, S. KEDAD-SIDHOUM, F. MONNA, G. MOUNIÉ AND D. TRYSTRAM (2017). *Scheduling independent moldable tasks on multi-cores with GPUs*. IEEE Trans. Parallel Distributed Syst.

Robust Gittins for Stochastic Scheduling

Benjamin Moseley ^{*} Heather Newman (speaker) [†] Kirk Pruhs [‡]
 Rudy Zhou [§]

We consider how to make stochastic scheduling more robust against mispredictions in the given job size distributions. We address the question of whether one can design and analyze a nonanticipatory scheduling policy that is robust with respect to minor errors in the reported probability distributions on the job sizes for the following classical stochastic scheduling problem:

Scheduling Problem Definition: The input consists of non-negative probability distributions \mathcal{D}_j for $j \in [n]$, where the j th job has size $P_j \sim \mathcal{D}_j$. We assume that the P_j 's are independent. The scheduler is initially only given the distributions, \mathcal{D}_j , and observes the realized processing times P_j over time as the jobs are scheduled. Our goal is to construct a *nonanticipatory* scheduling policy¹ that *preemptively* schedules all n jobs to completion. The objective is to minimize the *expected* total completion time, $\mathbb{E}[\sum_{j \in [n]} C_j]$, where C_j is the time that job j completes with respect to a given scheduling policy.

This problem is a stochastic version of the scheduling problem $1 \mid pmtn \mid \sum_j C_j$ using the standard 3-field scheduling notation. Many variations of this stochastic scheduling problem have been studied for decades. The stated problem is well-understood: a policy known as the Gittins (Index) policy is optimal (among all possible nonanticipatory policies) for all distributions [1]. Specialized to distributions with increasing hazard rates, for example, the Gittins (Index) is simply the Shortest Expected Processing Time (SEPT) policy.

Many stochastic scheduling policies are brittle/non-robust, in that even small errors in the reported distributions can result in quite poor schedules. For instance, policies may require carefully rescaled and truncated statistics based on exponential [3, 2] and p th moments [5], which are highly sensitive to errors. We will see that the Gittins policy is likewise brittle.

To address the question of whether there is a robust scheduling policy, we first need a notion of distance between distributions so that we have a measure of the error in the reported distributions. We introduce the following in [6].

^{*}moseleyb@andrew.cmu.edu. Tepper School of Business, Carnegie Mellon University.

[†]hnewman@vassar.edu. Department of Computer Science, Vassar College.

[‡]krp2@pitt.edu. Department of Computer Science, University of Pittsburgh.

[§]rudyzhou@microsoft.com. Microsoft, Supply Chain Optimization Technologies.

¹Roughly, a policy that does not anticipate information about the future, i.e., that does not, at time t , use the realizations of sizes of jobs that have not yet been completed by time t .

Definition 1. Consider two probability distributions, \mathcal{D} and \mathcal{D}' , over $\mathbb{R}_{\geq 0}$. Let $\alpha \geq 1$ be a constant. Then the pair $(\mathcal{D}, \mathcal{D}')$ is α -close if for all $x \geq 0$, we have

$$\frac{1}{\alpha} \cdot \mathbb{P}_{P \sim \mathcal{D}}(P > \alpha x) \leq \mathbb{P}_{P' \sim \mathcal{D}'}(P' > x) \leq \alpha \cdot \mathbb{P}_{P \sim \mathcal{D}}(P > x/\alpha).$$

With this definition of closeness in hand, our first contribution in [6] is showing that the Gittins policy is brittle with respect to small errors in the predicted distributions. We use a formulation of Gittins based on partitioning the jobs into subjobs called quanta. The Gittins policy runs these quanta in some pre-computed order based on a ranking function associated with the job size distributions. To formalize the result that the Gittins policy is brittle, we first need the following.

Definition 2. Let $\hat{\mathcal{I}} = \{\hat{\mathcal{D}}_j\}_{j=1}^n$ and $\mathcal{I}^* = \{\mathcal{D}_j^*\}_{j=1}^n$ be collections of non-negative job size distributions. We let $A(\mathcal{I}^*, \hat{\mathcal{I}})$ be a nonanticipatory policy (or the expected total completion time for the policy) that is given access to predicted distributions $\hat{\mathcal{I}}$ at the beginning of time and is unaware of true distributions \mathcal{I}^* . At any fixed time, the policy knows the completed job sizes and how much it has processed each incomplete job, where the job sizes are drawn from \mathcal{I}^* .

In particular, with the notation defined above, we can express the optimality of Gittins (GIPP) as follows. We let $\text{OPT}(\mathcal{I})$ be the optimal expected cost among all nonanticipatory scheduling policies with input \mathcal{I} .

Theorem 3 ([4, 7, 8]). For any instance $\mathcal{I} = \{\mathcal{D}_j\}_{j \in [n]}$, it is the case that $\text{GIPP}(\mathcal{I}, \mathcal{I}) = \text{OPT}(\mathcal{I})$.

In [6], we give the following lower bound, stating that GIPP is *not* robust to mis-specified predicted distributions, even if arbitrarily close to the true distributions.

Theorem 4. For all $\alpha > 1$, and for all $n \geq 2$, there exist true distributions $\mathcal{D}_j^* = \mathcal{D}_j^*(\alpha, n)$ which depend on α and n for all $j \in [n]$ and predicted distributions, $\hat{\mathcal{D}}_j = \hat{\mathcal{D}}_j(n)$ which depend on n for all $j \in [n]$. All such distributions are finitely supported, and every pair $(\mathcal{D}_j^*, \hat{\mathcal{D}}_j)$ is α -close. Then the instances $\mathcal{I}^* = \{\mathcal{D}_j^*\}_{j \in [n]}$ and $\hat{\mathcal{I}} = \{\hat{\mathcal{D}}_j\}_{j \in [n]}$ satisfy

$$\text{GIPP}(\mathcal{I}^*, \hat{\mathcal{I}}) = \Omega(n) \cdot \text{GIPP}(\mathcal{I}^*, \mathcal{I}^*) = \Omega(n) \cdot \text{OPT}(\mathcal{I}^*).$$

Intuitively, the reason for the brittleness of the Gittins policy is that both the design and analysis of Gittins depend on conditional probability distributions derived from the job size distributions, and for these instances the conditional probability distributions can be quite brittle with respect to small errors in the predicted job size distributions.

Our main contribution in [6] is a robust version of the Gittins policy. Our robust policy, which we call Robust Gittins (RG), is a modest modification of the Gittins policy that naturally arises from consideration of the lower bound instances in the proof of Theorem 4. Roughly, if the Gittins policy would preempt a job j after running it continuously for q time units, the Robust Gittins policy would run j for an additional $(\alpha - 1)q$ time units. We now give our robustness guarantee.

Theorem 5. Let $\alpha \geq 1$. Let $\mathcal{I}^* = \{\mathcal{D}_j^*\}_{j \in [n]}$ be a collection of true size distributions with finite support on n jobs, and $\hat{\mathcal{I}} = \{\hat{\mathcal{D}}_j\}_{j \in [n]}$ be a collection of predicted size distributions with finite support on n jobs. Further assume that every pair of distributions $(\mathcal{D}_j^*, \hat{\mathcal{D}}_j)$ is α -close. Then

$$RG(\mathcal{I}^*, \hat{\mathcal{I}}) \leq \alpha^6 \cdot GIPP(\mathcal{I}^*, \mathcal{I}^*) = \alpha^6 \cdot OPT(\mathcal{I}^*).$$

For example, in the case that $\alpha = 1 + \varepsilon$ for some small ε , Theorem 5 states that the expected total completion time for our Robust Gittins policy is roughly within a $(1 + 6\varepsilon)$ factor of the optimal expected total completion time, despite receiving the potentially erroneous predicted job size distributions as input.

As we believe our error metric is of independent interest, in [6] we also prove some desirable properties it satisfies (e.g., monotonicity, symmetry), give examples of natural distributions which are close in this metric, and relate it to well-known measures such as Wasserstein- and Lévy-distance.

References

- [1] J. C. Gittins. “Bandit processes and dynamic allocation indices”. In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 41.2 (1979), pp. 148–164.
- [2] A. Gupta, A. Kumar, V. Nagarajan, and X. Shen. “Stochastic load balancing on unrelated machines”. In: *Mathematics of Operations Research* 46.1 (2021), pp. 115–133.
- [3] J. Kleinberg, Y. Rabani, and É. Tardos. “Allocating bandwidth for bursty connections”. In: *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*. 1997, pp. 664–673.
- [4] A. G. Konheim. “A Note on Time Sharing with Preferred Customers”. In: *Probability Theory and Related Fields* 9 (1968), pp. 11–18.
- [5] M. Molinaro. “Stochastic ℓ_p load balancing and moment problems via the L-function method”. In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2019, pp. 343–354.
- [6] B. Moseley, H. Newman, K. Pruhs, and R. Zhou. “Robust Gittins for Stochastic Scheduling”. In: *ACM SIGMETRICS Performance Evaluation Review* 53.1 (2025), pp. 166–168.
- [7] K. C. Sevcik. “Scheduling for Minimum Total Loss Using Service Time Distributions”. In: *Journal of the ACM* 18.4 (1971), pp. 717–729.
- [8] G. Weiss. “On Almost Optimal Priority Rules for Preemptive Scheduling of Stochastic Jobs on Parallel Machines”. In: *Advances in Applied Probability* 27.3 (1995), pp. 821–839.

Minimizing Completion Times of Stochastic Jobs on Parallel Machines is Hard

Benjamin Moseley ^{*} Kirk Pruhs [†] Marc Uetz (speaker) [‡]
 Rudy Zhou [§]

1 Introduction

This paper considers the scheduling of stochastic jobs on parallel identical machines to minimize the expected total weighted completion time. While this is a classical problem with a significant body of research on approximation algorithms over the past two decades, constant-factor performance guarantees are currently known only under very restrictive assumptions on the input distributions, even when all job weights are identical. This algorithmic difficulty is striking given the lack of corresponding complexity results: to date, it is conceivable that the problem with unit weights could be solved optimally in polynomial time. We address this gap with hardness results that demonstrate the problem’s inherent intractability. For unit weights, we prove that deciding whether there exists a scheduling policy with expected cost at most a given threshold is $\#P$ -hard. Along the way, we also show that evaluating the expected objective value of the standard (W)SEPT greedy policy is itself $\#P$ -hard.

2 Model & Results

We consider a standard stochastic scheduling model with m *parallel identical machines* and n *non-preemptive* jobs. Each job j must be assigned to a single machine and, once started, runs to completion without interruption. The processing time p_j of job j is unknown in advance and is modeled as a nonnegative random variable X_j , independently distributed across jobs. Each job j additionally may have a weight w_j . Under a scheduling policy, let S_j and C_j denote the (random) start and completion times of job j , respectively. By definition, $\mathbb{E}[C_j] = \mathbb{E}[S_j] + \mathbb{E}[X_j]$. The objective is to find, or evaluate, a scheduling policy that minimizes the expected weighted sum of completion times, $\mathbb{E}\left[\sum_j w_j C_j\right]$. Our main result is the following.

^{*}moseleyb@andrew.cmu.edu. Carnegie Mellon University, Pittsburgh, PA, USA.

[†]kirk@cs.pitt.edu. University of Pittsburgh, Pittsburgh, PA, USA.

[‡]m.uetz@utwente.nl. University of Twente, Enschede, NL.

[§]rudvzhou1234@gmail.com. Microsoft, Washington DC, USA.

Theorem 1. *The problem to decide if there exists a scheduling policy with expected cost $\mathbb{E}[\sum_j C_j] \leq x$ for any given instance and $x > 0$, is a #P-hard problem.*

Proof idea. The proof goes by identifying two slightly different problem instances with weights w_j and two-point processing time distributions, so that evaluating the WSEPT rule (weighted shortest expected processing time first) on both instances allows us to count the the number of feasible solutions to a Knapsack constraint, a known #P-complete problem. The instances use “blocker jobs” which block all but one machine, so that a set of “knapsack jobs” has to be done on the remaining machine. The design of the instances is crafted so that each subset of items that does not fit into the knapsack, leads to a well-controlled “overflow” of knapsack jobs that allows us to accomplish the counting. This shows that evaluating WSEPT is #P-hard. The instances can be further fine-tuned to get rid of the weights w_j , proving #P-hardness to evaluate SEPT. Finally, further fine tuning of the instances allows to prove that SEPT is the *only* optimal policy, which then leads to the claimed result. \square

3 Why Care?

Optimal scheduling policies for this problem are known only in restricted special cases: When all weights are equal and processing times are exponentially distributed, the greedy SEPT policy is optimal [1, 2]. This generalizes to arbitrary distributions that are pairwise stochastically comparable [3]. With exponentially distributed jobs, SEPT is also optimal for weighted jobs whenever the SEPT order is agreeable with the weights [4].

In general, however, it is well understood that optimal policies must be *adaptive* [5, 6, 7]: The adaptivity gap between non-adaptive and adaptive policies can grow proportionally to the squared coefficient of variation of the processing time distributions [6, 7].

Moreover, a substantial literature has developed approximation algorithms for stochastic scheduling, typically based on linear programming relaxations of the optimal adaptive policy [8, 9, 10, 6, 11, 12, 13]. These approximation guarantees also scale with the largest squared coefficient of variation of the random variables X_j . For the special case of unit weights, an exception is [14], whose performance guarantee instead depends on the number of machines m . More recently, for two-point Bernoulli processing times, sublinear-in- m approximation guarantees have been obtained that do not depend on second moments [15, 16].

Despite the sophistication of these approximation results, lower bounds on the computational complexity of stochastic scheduling have remained absent for min-sum objectives. Indeed, hardness results for stochastic scheduling problems are known only in much more structured settings [17, Thm. 19.7][18], or for the makespan objective with pre-assigned jobs [19].

Hence, our result is a first contribution that testifies the inherent intractability of one of the most basic stochastic parallel machine scheduling problems.

References

- [1] G. Weiss and M. Pinedo. Scheduling tasks with exponential service times on non-identical processors to minimize various cost functions. *J. Appl. Prob.* 17 (1980), 187–202.
- [2] J.L. Bruno, P. J. Downey, and G.N. Frederickson. 1981. Sequencing tasks with exponential service times to minimize the expected flowtime or makespan. *J. ACM* 28 (1981), 100–113.
- [3] R.R. Weber, P. Varaiya, and J. Walrand. 1986. Scheduling jobs with stochastically ordered processing times on parallel machines to minimize expected flowtime. *J. Appl. Prob.* 23 (1986), 841–847.
- [4] T. Kämpke. 1987. On the optimality of static priority policies in stochastic scheduling on parallel machines. *J. Appl. Prob.* 24 (1987), 430–448.
- [5] M. Uetz. 2003. When greediness fails: Examples from stochastic scheduling. *O.R. Lett.* 31 (2003), 413–419.
- [6] M. Skutella, M. Sviridenko, and M. Uetz. 2016. Stochastic Scheduling on Unrelated Machines. *Math. O.R.* 41, 3 (2016), 851–864.
- [7] F. Eberle, F. Fischer, J. Matuschke, and N. Megow. 2019. On index policies for stochastic minsum scheduling. *O.R. Lett.* 47 (2019), 213–218.
- [8] R. H. Möhring, A. S. Schulz, and M. Uetz. 1999. Approximation in stochastic scheduling: The power of LP-based priority policies. *J. ACM* 46 (1999), 924–942.
- [9] M. Skutella and M. Uetz. 2005. Stochastic machine scheduling with precedence constraints. *SIAM J. Comput.* 34 (2005), 788–802.
- [10] N. Megow, M. Uetz, and T. Vredeveld. 2006. Models and Algorithms for Stochastic Online Scheduling. *Math. O.R.* 31, 3 (2006), 513–525.
- [11] S.R. Balseiro, D.B. Brown, and C. Chen. 2018. Static Routing in Stochastic Scheduling: Performance Guarantees and Asymptotic Optimality. *O.R.* 66 (2018), 1641–1660.
- [12] V. Gupta, B. Moseley, M. Uetz, and Q. Xie. 2020. Greed works—online algorithms for unrelated machine stochastic scheduling. *Math. O.R.* 45 (2020), 497–516.
- [13] S. Jäger. 2023. An improved greedy algorithm for stochastic online scheduling on unrelated machines. *Discr. Opt.* 47 (2023), 100753.
- [14] S. Im, B. Moseley, and K. Pruhs. Stochastic Scheduling of Heavy-tailed Jobs. In: *STACS 2015*, 474–486.
- [15] A. Gupta, B. Moseley, and R. Zhou. 2023. Minimizing Completion Times for Stochastic Jobs via Batched Free Times. In *SODA 2023*. 1905–1930.
- [16] A. Antoniadis, R. Hoeksma, K. Schewior, and M. Uetz. 2025. Stochastic scheduling with Bernoulli-type jobs through policy stratification. In: *FOCS 2025*, 2449–2472.
- [17] Chr. Papadimitriou. 1993. *Computational Complexity*. Pearson.
- [18] J.N. Hagstrom. 1988. Computational Complexity of PERT Problems. *Networks* 18 (1988), 139–147.
- [19] A. Gupta, A. Kumar, V. Nagarajan, and X. Shen. 2020. Stochastic Load Balancing on Unrelated Machines. *Math. O.R.* 46, 1 (2020), 115–133.

A further investigation of Learning-SEPT

P.J. van Mill (speaker) * André Berger † Tjark Vredeveld ‡

1 Introduction

There exists a substantial amount of literature on stochastic scheduling, in which various properties of jobs are not certain but instead randomly drawn from some probability distribution; see e.g. Pinedo [6]. Most of this work assumes that these distributions, or at least the first moments, are known. This is not always realistic: decisionmakers may face substantial uncertainty in addition to randomness. Bayesian stochastic scheduling incorporates such uncertainty into the model. In Bayesian scheduling, the parameters of the probability distributions involved are not fully known in advance but are *learned* through the process of scheduling. This substream of literature dates back to Gittins & Glazebrook [4] and has historically drawn on the study of multi-armed bandits.

We consider a Bayesian variant of a single-machine scheduling problem with the sum of completion times objective, where jobs belong to classes and each class has a probability distribution for the processing times of the jobs. These distributions are uncertain; as jobs are scheduled, the observed processing times provide information.

We study a simple scheduling policy for this problem, called Learning-SEPT. The Learning-SEPT policy has been studied before by Marbán, Rutten and Vredeveld [1], who showed an upper and lower bound on its performance, leaving a gap that grows linearly with the number of classes.

2 Model

In the problem at hand, jobs are scheduled non-preemptively on a single machine, with the objective of minimising the expected sum of completion times. Each job belongs to one of K classes, with n_i being the number of jobs in class i , and the total number of jobs being $n = \sum_{i \in [K]} n_i$. The processing time of a job is stochastic, according to the probability distribution associated with its class. The

*p.j.vanmill@maastrichtuniversity.nl. Department of quantitative economics, Maastricht University. Postal address: Secretariat QE, School of Business and Economics, Tongersestraat 53, 6211LM Maastricht, Netherlands.

†a.berger@maastrichtuniversity.nl. idem.

‡t.vredeveld@maastrichtuniversity.nl. idem.

distributions of the various classes are all from the same family but parametrised differently. The distribution parameter associated with a class is unknown, but a prior distribution for this parameter is given, modeling the *a priori* beliefs about the class. Each time after a job has completed, the beliefs about the parameter of its class are updated.

In the model of [1] and [2], the distribution family is exponential, with rate parameters θ_i . The prior distribution of θ_i is $\text{Gamma}(\alpha_i, \omega_i)$; this is a conjugate prior to the exponential distribution. Updating the beliefs after observing a job realisation yields a Gamma posterior, which becomes the prior for the next job in its class.

For this problem Hamada and Glazebrook [2] describe an optimal policy, and even for a generalisation where each class has a weight. Their results partially overlap with those of Burnetas and Katehakis [3], who studied a similar problem with two classes, where the parameter of one class is known. Despite an optimal policy being known, Marbán et al. [1] argued that it is very computationally intensive and therefore studied simple heuristic policies.

We consider a different distribution family, namely the one-point distributions (distributions with a step function cdf). After scheduling a single job of a class, the processing times of the other jobs of the class are, almost surely, all the same as the observed processing time of that job. The prior distribution of a class is then simply a distribution for the true processing time P_i associated with the class.

3 Learning-SEPT

For the stochastic setting where the processing time distributions are known, the SEPT (shortest expected processing time) rule was proven optimal by Rothkopf [5]. In the Bayesian setting, two variants of the SEPT rule can be considered: NonAdaptive-SEPT, which uses the *a priori* beliefs, and Learning-SEPT, which at every point uses up-to-date beliefs for the expected processing times.

Marbán et al. [1] studied the approximation ratio, defined as $\frac{\mathbb{E}[\Pi]}{\mathbb{E}[\text{OPT}]}$ where Π is the objective value achieved by the policy. They showed that the approximation ratio of both NonAdaptive-SEPT and Learning-SEPT is bounded from above by K , which is tight for NonAdaptive-SEPT. In addition, they showed a lower bound of $\sqrt{K-1} + 1$ on the approximation ratio of Learning-SEPT, and conjectured based on computational experiments that this bound may be tight.

With our different distributional assumptions, we have the following as a first step towards proving more general results:

Theorem 1 *Let the processing times follow one-point distributions and let the priors satisfy $\mathbb{E}[P_i] = 1$ for all classes i ; let an adversary decide in which order the classes are considered. In this setting, the limit as $n \rightarrow \infty$ of the approximation ratio of Learning-SEPT is $O(\sqrt{K})$.*

In the analysis, we make use of comparisons to the following problem for some $N < K$. in which N points are placed in the unit interval and weights are assigned

to them, so as to maximise a certain expression involving the weighted sum of pairwise minima and the weighted sum of pairwise maxima.

$$\begin{aligned} \max \quad & \frac{\frac{1}{2} \sum_{i \in [N]} w_i^2 + \sum_{i \in [N]} \sum_{j \in [N]: j > i} w_i w_j \max\{x_i, x_j\}}{\frac{1}{2} \sum_{i \in [N]} w_i^2 + \sum_{i \in [N]} \sum_{j \in [N]: j > i} w_i w_j \min\{x_i, x_j\}} \\ \text{s.t.} \quad & \sum_{i \in [N]} w_i = 1 \\ & x_i \in [0, 1], w_i \in \mathbb{R}_+ \qquad i \in [N] \end{aligned}$$

The optimal value of this problem is $O(\sqrt{N})$.

As a second result, we generalise the upper bound of K to arbitrary processing time distributions and prior distributions.

4 Future work

At time of writing, we are working on generalising our result beyond the current assumptions, which are still quite restrictive. Other simple policies for the problem, perhaps inspired by multi-armed bandit policies, should also be studied.

References

- [1] S. MARBÁN, C. RUTTEN AND T. VREDEVELD (2011). Learning in stochastic machine scheduling. *Proceedings of the 9th WAOA*, pp. 21-34.
Extended version: C. RUTTEN (2013), *The power of simple scheduling policies* (PhD thesis, Maastricht University), chapter 6.
- [2] T. HAMADA AND K.D. GLAZEBROOK (1993). A Bayesian sequential single machine scheduling problem to minimize the expected weighted sum of flow-times of jobs with exponential processing times. *Operations Research*, 41(5), pp. 924-934.
- [3] A.N. BURNETAS AND M.N. KATEHAKIS (1993). On sequencing two types of tasks on a single processor under incomplete information. *Probability in the Engineering and Informational Sciences*, 7, pp 85-119.
- [4] J.C. GITTINS AND K.D. GLAZEBROOK (1977). On Bayesian models in stochastic scheduling. *Journal of Applied Probability*, 14(3), pp. 556-565.
- [5] M.H. ROTHKOPF (1966). Scheduling with random service times. *Management Science*, 12(9), pp. 703-713.
- [6] M. PINEDO (2004). Offline deterministic scheduling, stochastic scheduling, and online deterministic scheduling. *Handbook of scheduling*, ed. by J.Y.-T. LEUNG, Chapman & Hall / CRC, chapter 38.

A Decomposition-Based Exact Solution Approach for Lot-sizing and Scheduling Problem in Co-production Systems

Eyüp Ensar Işık (Speaker) * Z. Caner Taşkın † Semra Ağralı ‡

1 Introduction

Co-production systems allow the simultaneous production of multiple products due to the structure of the products or for efficient resource use [7]. Therefore, optimizing lot-sizing decisions and the sequence of these lots to minimize inventory and setup costs is crucial. Simultaneous consideration of these decisions corresponds to the well-known NP-hard lot-sizing and scheduling problem (LSP) in the literature [5]. To achieve effective and optimal production plans, making these decisions simultaneously is critical. By determining lot sizes, customer demand is satisfied while minimizing production-related costs. Simultaneously, sequence decisions of the production lots are made to minimize the total setup cost.

In the literature, several models and solution methods have been developed to solve the simultaneous LSP efficiently [5]. The General LSP (GLSP) uses macro and micro periods to address the impracticality of using smaller subperiods [2]. The problem is called “general” since it can be adapted to other problem variants. GLSP is well-studied in the literature. However, most of the studies work on meta or matheuristics [5]. To the best of our knowledge, there is no exact solution method for GLSP in co-production systems.

2 Problem Description and Solution Method

In this study, we consider a single-machine, single-level GLSP in co-production systems with sequence-dependent setup costs. The problem can be formalized using the GLSP model proposed by [2] with minor adaptations. This model is the basic formulation for GLSP, but there are improved versions. Most of these formulations are classified in [6]. The authors emphasize that both the network-flow reformulation of GLSP (GLSP^{NF}) and the facility-location reformulation proposed by [1] (GLSP^{CC}) provide better results.

*eeisik@yildiz.edu.tr. Department of Industrial Engineering, Boğaziçi University, 34342, Beşiktaş, İstanbul, Türkiye. Department of Industrial Engineering, Yıldız Technical University, 34349, Beşiktaş, İstanbul, Türkiye.

†caner.taskin@bogazici.edu.tr. Department of Industrial Engineering, Boğaziçi University, 34342, Beşiktaş, İstanbul, Türkiye.

‡semra.agrali@mef.edu.tr. Department of Industrial Engineering, MEF University, 34396, Sarıyer, İstanbul, Türkiye.

We decompose the problem into two. In the master problem, lot-sizing decisions are made to minimize the inventory holding cost and estimated setup cost. In the subproblem, the sequences of the lots to be produced in the master problem are determined, and it can be considered a single-machine scheduling problem with release and due dates. By decomposing the problem, we significantly decrease the number of variables in the master problem by eliminating micro-periods. Since our subproblem has binary integer variables, basic decomposition approaches cannot be used. Here, we adapt the integer L-shaped method proposed by [4]. Using this method, we can drive our optimality cuts.

2.1 Algorithmic Improvements

Since our preliminary studies show that the basic algorithm is not effective for medium and large-sized instances, we perform improvements to the algorithm. We started with cut lifting by observation on the setup costs, which have a triangular property. Based on this observation, we reformulate the L-shaped cuts and show that they are valid and stronger than the basic version of the cuts for triangular setup instances.

Another observation on integer L-shaped cuts concerns the impact of the lower bound on the estimated setup cost on its performance. To increase the performance of the cuts, we calculate better lower bounds than zero. Afterwards, we disaggregate our cuts by period and across periods to provide stronger cuts. To generate these cuts, we need an efficient way to calculate a lower bound on the setup cost for each period. We use a dynamic programming algorithm that, at each iteration, recursively computes the minimum setup cost to obtain lower bounds. We extend this algorithm to compute the minimum setup cost for the entire sequence.

3 Computational Study

To test the effectiveness of our decomposition algorithm (DA), we use benchmark instances from [3] and compare it with adapted MIP formulations. We also test the performance of the algorithmic improvements by comparing the basic DA with its improved versions. We made our computational study using the IBM CPLEX 22.1.1 (64-bit) solver on C++ with a time limit of 600 seconds. We worked on a server with an Intel Xeon Silver 4214R 2.4 GHz CPU, 64 GB RAM, running Windows Server 2019.

We start the comparison with the basic version of our DA and three MIP formulations. The results show that MIP models and DA can optimally solve small-sized instances. For larger instances, the performance of all approaches decreases. However, GLSP^{NF} has the best average solution time for medium and large-sized instances. We test the performance of our improvements in three phases. According to the results of the first phase, which focused on the impact of cut lifting and lower bounds, cut-lifting and lower-bound calculations improved the algorithm's performance. The number of cuts added to the MP is decreased, and the solution time and quality are improved for all instances. In the second phase, which also uses disaggregated cuts, solution times decrease for small instances, and optimality

gaps decrease for medium and large instances. Finally, we obtain better solution times and optimality gaps for all instances when we use the dynamic programming approach to solve our subproblem. Results also show that the best version of our algorithm outperforms the GLSP^{NF} formulation for all metrics.

4 Conclusion

We study a single-machine, single-level GLSP in co-production systems with sequence-dependent setup costs. We developed MIP formulations to solve the problem optimally. To efficiently solve this problem, we propose a decomposition-based exact solution approach. Since our subproblem has binary integer variables, we use the integer L-shaped-based cutting plane algorithm to solve the decomposed model. We also perform improvements on the algorithm by lifting the cuts, calculating better lower bounds, disaggregating the cuts, and developing a dynamic programming-based approach to solve the subproblem. We use a benchmark dataset to test the performance of the algorithm and adapted formulations. The results show that improvements significantly affect the performance of the algorithm, and the best version of our algorithm outperforms MIP formulations. Effective heuristics can be used to tackle more complex cases as a future research area. Adding stochastic demand structures to the problem is an additional approach that could be taken.

References

- [1] A. R. CLARK AND S. J. CLARK (2000). *Rolling-horizon lot-sizing when set-up times are sequence-dependent*. International Journal of Production Research, 38(10):2287–2307.
- [2] B. FLEISCHMANN AND H. MEYR (1997). *The general lotsizing and scheduling problem*. Operations-Research-Spektrum, 19(1):11–21.
- [3] B. PAMUK, Z. C. TAŞKIN, S. AĞRALI AND B. KABAKULAK (2022). *A lot-sizing problem in deliberated and controlled co-production systems*. IISE Transactions, 54(10):950–962.
- [4] G. LAPORTE AND F. V. LOUVEAUX (1993). *The integer l-shaped method for stochastic integer programs with complete recourse*. Operations Research Letters, 13(3):133–142.
- [5] K. COPIL, M. WÖRBELAUER, H. MEYR AND H. TEMPELMEIER (2017). *Simultaneous lotsizing and scheduling problems: a classification and review of models*. OR Spectrum, 39(1):1–64.
- [6] L. GUIMARAES, D. KLABJAN AND B. ALMADA-LOBO (2014). *Modeling lot-sizing and scheduling problems with sequence dependent setups*. European Journal of Operational Research, 239(3):644–662.
- [7] S. AĞRALI (2012). *A dynamic uncapacitated lot-sizing problem with co-production*. Optimization Letters, 6(6):1051–1061.

SMART-MIG: A Learning Framework for Scalable and Energy-Efficient GPU Scheduling

Tanvi Hisaria ^{*} Neel Karia [†] Clifford Stein (Speaker) [‡]
 Asser Tantawi [§] Olivier Tardieu [¶] Wenqing Yu ^{||}

1 Introduction

Data centers play a critical role in sustaining our modern world and enabling technological advances, but they currently carry a significant environmental cost. In 2022, data centers consumed 2% of global energy demand with a total of 460 terawatt hours consumed[2]. This value is expected to more than double by 2026, an increase driven largely by the recent popularity of large language models (LLMs) and other AI/ML workloads, which depend on the use of power-hungry GPUs.

Efforts to reduce data center energy consumption include the development of advanced hardware and energy-efficient processors, widespread adoption of virtualization, the use of smart cooling systems, and strategic data center design. In addition to these developments, there is an immediate opportunity to further reduce energy consumption by scheduling workloads in a more efficient manner that considers the power characteristics of the GPUs. In this work, we focus on the energy-efficient scheduling of energy-expensive AI/ML workloads.

Recently developed GPU features present particularly interesting opportunities for scheduling algorithms. Companies such as NVIDIA have developed GPU-sharing features that help improve utilization by enabling the colocation and concurrent processing of multiple workloads on the same GPU. A recently released capability is the MIG feature. MIG enables the partitioning of its memory and compute resources to run multiple jobs in various *slices*, to utilize increasingly powerful GPUs more efficiently and avoid costly under-utilization. While MIG allows sharing one GPU among workloads, little has been done to recommend how to divide the GPU especially given energy-related objectives.

There is often a trade-off between latency and energy usage when scheduling workloads [1]. The trade-off is well understood to be non-linear, as power con-

^{*}th2720@columbia.edu. Department of IEOR, Columbia University

[†]nmk2154@columbia.edu Department of IEOR, Columbia University

[‡]cliff@ieor.columbia.edu Department of IEOR, Columbia University

[§]tantawi@us.ibm.com IBM TJ Watson Research Center

[¶]tardieu@us.ibm.com IBM TJ Watson Research Center

^{||}wv2462@columbia.edu Department of IEOR, Columbia University

sumption in computing devices is typically a super-linear function of their speed. We explore this trade-off for MIG-enabled GPUs, where workloads can be processed on different-sized slices of the GPU leading to different performance and energy consumption results.

2 Results

In this work, we design an online scheduling framework that decides how to dynamically repartition a MIG-enabled GPU and allocate AI/ML workloads to its slices. We aim to achieve low energy consumption and good scheduling performance simultaneously, using a multi-parameter objective.

The first novelty of our work is its focus on energy-efficient scheduling for MIG, a relatively unexplored area that distinguishes our work from previous research on MIG scheduling problems. Previous work on MIG focused primarily on performance optimization, while our work explores scheduling with the dual objectives of energy and a quality-of-service metric. While our ultimate goal is to be able to schedule multi-GPU systems, an important step is to understand the single-GPU case. Thus, our work intentionally focuses on the single-GPU use case to establish a foundational understanding of energy-aware scheduling on MIG.

The second novelty of our work lies in the diverse AI/ML workloads we consider and the inelasticity of their scaling. Our queue consists of a mix of inference and fine-tuning or training jobs with varying elasticity; the performance of some jobs scales linearly with increased resources, while others scale sublinearly or not at all, adding significant complexity to our scheduling problem. We analyze an NVIDIA A100 40GB GPU as our single machine with an experimentally derived non-linear power curve across utilization levels. Although we focus on this setting, our methodology can extend, with appropriate modeling, to other computational environments and job mixes. This combination of a heterogeneous job mix, non-linear elasticity, and nonlinear power along with preemption creates a complex problem that we solve for an online and realistic setting. In subsequent sections, we explain our modeling, algorithmic, and experimental choices.

Our MIG scheduling problem has similarities and key differences with other scheduling problems. The job scheduling problem within a MIG partition is related to scheduling with heterogeneous servers[4], where a slice corresponds to a server. In such a class of queueing problems, multiple types of jobs are served by a set of servers with different service rates. Threshold control policies have been designed (given some simplifying assumptions) and analytical results are limited to tractable cases[3]. However, the MIG scheduling problem is more complex in many ways discussed above, after including the diverse job mix and inelasticity. The repartitioning problem is analogous to having a total service capacity that needs to be split among several servers with different capacities such that the sum of the capacities is equal to the total available capacity. To complicate matters, only a particular set of combinations are possible.

This problem setup gives rise to three subproblems: (i) choosing the parti-

tioning of the GPU from a fixed set of possible slicing profiles, (ii) deciding the prioritization of jobs from the queue, and (iii) selecting an available slice to assign to each job. The exact version of our scheduling problem is NP-hard, as it is NP-hard even without considering the energy and with a fixed configuration of MIG with more than one slice (which is implied by a result in [5]). We solve this problem in two parts: (1) scheduling jobs onto slices within a given partition and (2) deciding when to repartition and to which configuration to repartition. We summarize our results below:

- We define a multi-objective metric named ET that considers both energy consumption and average tardiness to relatively evaluate our algorithms.
- We test several algorithms and experimentally observe that the Earliest Deadline First - Slowest Slice (EDF-SS) algorithm generally outperforms the other algorithms for scheduling different queues of inelastic jobs within various MIG configurations.
- By using dynamic repartitioning with a reinforcement learning model based on the EDF-SS scheduling algorithm, we obtain considerable improvements according to ET . When evaluated based on a hybrid of real and simulated data, the RL-based dynamic repartitioning algorithm performs better than the benchmark models of twice-daily repartitioning by 26%, static partitioning by 31%, and no partitioning by 68%.

References

- [1] Ghazanfar Ali, Mert Side, Sridutt Bhalachandra, Nicholas J Wright, and Yong Chen. Performance-aware energy-efficient GPU frequency selection using DNN-based models. In *Proceedings of the 52nd International Conference on Parallel Processing*, pages 433–442, 2023.
- [2] Cohen, A. AI is pushing the world toward an energy crisis. *forbes.*, 2024. <https://www.forbes.com/sites/arielcohen/2024/05/23/ai-is-pushing-the-world-towards-an-energy-crisis/>.
- [3] Dmitry Efrosinin, Natalia Stepanova, Janos Sztrik, and Andreas Plank. Approximations in performance analysis of a controllable queueing system with heterogeneous servers. *Mathematics*, 8(10), 2020.
- [4] Jung Hyun Kim, Hyun-Soo Ahn, and Rhonda Righter. Managing queues with heterogeneous servers. *Journal of Applied Probability*, 48(2):435–452, 2011.
- [5] René Sitters. Complexity of preemptive minsum scheduling on unrelated parallel machines. *Journal of Algorithms*, 57(1):37–48, 2005.

On the Complexity of the Euclidean Capacitated Vehicle Routing Problem

Steven Miltenburg (Speaker) *

1 Introduction

The Vehicle Routing Problem (VRP) is one of the most well-studied optimization problems in operations research due to its many applications in logistics and its close relation to the well-studied traveling salesperson problem (TSP). In this paper, we study the complexity of the capacitated VRP (CVRP) with unit demand for constant capacity c in Euclidean space. In this problem we are given a location in space r_0 , called the depot, and a set of n locations in Euclidean space called requests. There is an unlimited number of vehicles available, each with a capacity c . We define a tour of a vehicle as a route that starts and ends in the depot, while visiting at most c requests in between. The goal is to visit all requests in tours while minimizing the total distance traveled. The unit demand CVRP is also known as the k -tours problem [2] (where the capacity corresponds to the number of tours covering all the points).

Several papers claim that CVRP for constant capacity $c \geq 3$ is an NP-hard problem. For example, Nie and Zhou [8] write: “The Euclidean CVRP is a generalization of the Euclidean traveling salesman problem and is known to be NP-hard for all $c \geq 3$ ”. They attribute this to Asano et al. [3], but they prove NP-hardness (even APX-hardness) only for a general metric space. Bompadre et al. [4] write: “However, Euclidean CVRP is NP-hard for any $c \geq 3$ ”, and Adamaszek et al. [1] write: “The Euclidean CVRP contains the TSP problem as a special case and it is known to be NP-hard for all $c \geq 3$ ”. For constant capacity, however, the TSP is not a special case of CVRP, and therefore does not provide an immediate proof of NP-hardness. Note that for $c = 2$, CVRP reduces to a matching problem, and hence can be solved in polynomial time. Some other papers [3, 5] seem to suggest NP-hardness of Euclidean CVRP for constant capacity $c \geq 3$, but are less explicit in their claim. A common citation for NP-hardness of CVRP is the classic article [7] by Haimovich and Rinnooy Kan. However, that paper only studies approximation algorithms and heuristics and contains no hardness results. Despite these suggestions, it appears that the complexity of Euclidean CVRP with

*s.j.g.miltenburg@vu.nl. Vrije Universiteit Amsterdam, De Boelelaan 1105, Amsterdam, 1081 HV, the Netherlands.

constant capacity $c \geq 3$ is still an open problem. We show that Euclidean CVRP is NP-hard in \mathbb{R}^3 for constant capacity $c = 3$.

2 Main Result

Our main contribution is the following theorem:

Theorem 1. *Euclidean capacitated VRP with $c = 3$ is NP-hard in \mathbb{R}^3 .*

We present an elegant proof by reduction from the Planar Exact Cover by 3-sets problem (Planar-X3C). In this problem, we are given a collection of elements and triplets, where each triplet contains exactly three elements, and we must determine whether there exists a set of triplets that covers each element exactly once. The planarity constraint ensures that the bipartite graph relating elements to triplets can be drawn without crossings.

The key insight of our reduction is to encode the combinatorial structure of the Planar-X3C instance into a geometric CVRP instance where the optimal routing solution directly corresponds to the choice of triplets in the exact cover. We construct a CVRP instance in three-dimensional Euclidean space where the depot is placed far from a plane containing all request points. This ensures that the distance to the depot dominates, making all depot-to-request distances approximately equal and forcing optimal solutions to use tours of exactly three requests. This gives us a lower bound on the objective which we use to prove the optimality of the proposed solution.

The geometric construction represents each element and each triplet from the Planar-X3C instance as specific request configurations. For each triplet, we create three collinear requests that can be efficiently covered by a single tour. For each element, we create a single request that must be visited. The crucial aspect is the connection between triplets and elements: we create paths of alternating single and double requests (where double requests are two requests at the same location) that link triplets to their associated elements.

When a triplet is selected, its three associated paths can be efficiently covered by pairing double requests with neighboring single requests, ultimately visiting each element request exactly once. Conversely, when a triplet is not selected, its three requests must be split across the three different paths, which therefore do not cover the element request. This alternate covering is equally efficient.

When there is no exact cover solution, it is impossible to cover all paths efficiently. Thus the geometry enforces that an optimal CVRP solution achieves a specific target cost if and only if the corresponding Planar-X3C instance has an exact cover.

3 Extensions and Open Problems

The reduction extends naturally to any capacity $c = 3p$ (multiples of 3) by creating p times as many requests at each location in the construction. However, extending

the proof to an arbitrary constant capacity c remains an open problem. The extension is possible if Planar Exact Cover by c -sets is NP-hard for general $c > 3$. In this case our reduction technique can be adapted by using locations with $c - 1$ requests instead of double requests and arranging triplet requests in regular c -gons.

A natural question is whether the problem remains NP-hard in \mathbb{R}^2 rather than \mathbb{R}^3 . Our reduction relies on placing the depot in a third dimension to ensure approximately equal distances to all requests, leading to a simple lower bound which proves optimality of the proposed CVRP solution. In two dimensions the problem structure changes significantly, and a similar reduction fails because the proposed CVRP solution is no longer optimal. Thus a reduction in \mathbb{R}^2 would require additional techniques. Despite these challenges, we conjecture that Euclidean CVRP remains NP-hard in \mathbb{R}^2 for all constant $c \geq 3$.

References

- [1] A. ADAMASZEK, A. CZUMAJ, AND A. LINGAS (2010). PTAS for k-tour cover problem on the plane for moderately large values of k. *International Journal of Foundations of Computer Science*, 21(06):893–904.
- [2] S. ARORA (1998). Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782.
- [3] T. ASANO, N. KATOH, H. TAMAKI, AND T. TOKUYAMA (1996). Covering points in the plane by k-tours: a polynomial approximation scheme for fixed k. Technical Report RT0162, IBM Tokyo Research Laboratory.
- [4] A. BOMPADRE, M. DROR, AND J.B. ORLIN (2006). Improved bounds for vehicle routing solutions. *Discrete Optimization*, 3(4):299–316.
- [5] M. CHARIKAR, S. KHULLER, AND B. RAGHAVACHARI (1998). Algorithms for capacitated vehicle routing. In *Proceedings of the thirtieth annual ACM Symposium on Theory of Computing*, pages 349–358.
- [6] M.E. DYER AND A.M. FRIEZE (1986). Planar 3DM is NP-complete. *Journal of Algorithms*, 7(2):174–184.
- [7] M. HAIMOVICH AND A. RINNOOY KAN (1985). Bounds and heuristics for capacitated routing problems. *Mathematics of Operations Research*, 10(4):527–542.
- [8] Z. NIE AND H. ZHOU (2023). Euclidean capacitated vehicle routing in random setting: A 1.55-approximation algorithm. arXiv preprint arXiv:2304.11281.

Capacitated Vehicle Routing with Order Restrictions: Models, Algorithms, and Limits

Steven Miltenburg* Tim Oosterwijk (Speaker)† René Sitters‡

1 Introduction

The capacitated vehicle routing problem (c -VRP), also known as the k -tour cover problem, is a fundamental problem in planning and scheduling, capturing the task of allocating limited resources (vehicles) over time and space to serve a set of requests efficiently. Vehicles of limited capacity must serve a set of requests from a central depot while minimizing total travel distance. Beyond the classical formulation, many practical applications impose *order restrictions*: certain requests must be served before others due to physical loading constraints, temporal dependencies, or logical precedence relations, as commonly arising in scheduling and constrained planning settings.

Capacitated routing problems have been studied extensively, such as in the seminal work by Haimovich and Kann [4], who introduced the iterated tour partitioning heuristic. Over the years, strong approximation schemes have been developed for Euclidean instances, including PTASs and QPTASs for various capacity regimes [1, 3, 6]. In contrast, much less is known about the effect of additional structural constraints such as precedence relations.

In this work, we study c -VRP under arbitrary precedence constraints that apply only between requests assigned to the same tour. This restriction is natural in many applications: requests handled by different vehicles need not be ordered, while within a single route, feasibility may depend on respecting a given partial order.

2 Problem Definition

We are given a metric space with distance function $d(\cdot, \cdot)$ with a set of requests located at vertices in the metric space, and a distinguished depot r . Each vehicle

*s.j.g.miltenburg@vu.nl. Department of Operations Analytics Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV, Amsterdam, the Netherlands.

†t.oosterwijk@vu.nl. Department of Operations Analytics Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV, Amsterdam, the Netherlands.

‡r.a.sitters@vu.nl. Department of Operations Analytics Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV, Amsterdam, the Netherlands.

starts and ends at the depot and can serve at most c requests. A feasible solution consists of a set of tours covering all requests.

In addition, we are given a precedence relation \preceq on the requests. For any tour, the order in which requests are visited must be in line with the restriction of \preceq to the requests assigned to that tour. No constraints are imposed between requests served by different vehicles.

The objective is to minimize the total length of all tours. We study this problem in Euclidean spaces of fixed dimension and on trees, and we consider both constant and unbounded vehicle capacity.

3 Approximation in Euclidean Spaces

We consider c -VRP in fixed-dimensional Euclidean space under arbitrary precedence constraints. For constant vehicle capacity c we obtain a particularly simple polynomial-time approximation scheme. Our approach is similar to [6].

The algorithm proceeds in two main steps. First, the instance is decomposed into a collection of bounded subinstances using a geometric partitioning around the depot. This approach is inspired by classical planar separator techniques for approximation schemes [2] and by earlier PTASs for Euclidean k -tour cover [1]. Second, each bounded instance is solved independently. If the instance is small, we apply complete enumeration; otherwise, all request locations are rounded to a grid and a trivial solution is used that sends vehicles directly to grid points.

The same framework naturally extends beyond constant vehicle capacities. By combining the geometric decomposition with a dynamic programming procedure on bounded subinstances, the approach applies to moderately growing capacities up to $c = \tilde{O}\left(\log^{1/(D+1)} n\right)$, where D is the fixed dimension of the Euclidean space.

Theorem 1 (PTAS in Euclidean Space) *There exists a PTAS for c -VRP in \mathbb{R}^D for fixed $D \geq 1$ with arbitrary precedence constraints and $c = \tilde{O}\left(\log^{1/(D+1)} n\right)$.*

N.B. Note that our bound on c is much smaller (worse) than the bound of Adamaszek et al. [1]. However, our approach does not rely on the QPTAS for c -VRP by Das and Mathieu [3] and includes arbitrary precedence constraints.

In addition, the framework naturally extends to variants with time windows, where each request must be served within a specified interval.

Theorem 2 (Time Windows) *There exists a PTAS for c -VRP with arbitrary time windows in \mathbb{R}^D for fixed $D \geq 1$ for $c = \tilde{O}\left(\log^{1/(D+1)} n\right)$ if the time windows are allowed to be violated by at most a factor $1 + \mathcal{O}(\varepsilon)$.*

4 Hardness on Trees

The situation changes significantly on tree metrics. While a PTAS exists for unordered c -VRP on trees for arbitrary capacity [5], we show that adding order

restrictions fundamentally changes the complexity of the problem.

In particular, we prove that when the precedence constraints form a total order, c -VRP on trees is APX-hard if the vehicle capacity is part of the input. We prove this by an approximation-preserving reduction from the Maximum Bounded 3-Dimensional Matching problem.

Theorem 3 (APX-Hardness on Trees) *The c -VRP on a tree is APX-hard under precedence constraints forming a total ordering, if c is part of the input.*

From the above theorem, c -VRP with arbitrary precedence constraints and c -VRP with time windows are APX-hard on a tree, since they both contain the total order as a special case. Without precedence constraints, a PTAS for c -VRP on trees, for arbitrary c , was given only quite recently by Mathieu and Zhou [5]. Our result implies that the PTAS cannot be extended with a total order constraint, assuming $P \neq NP$. For constant c and precedence constraints, determining the complexity remains an open problem for now.

5 Discussion

We show that capacitated vehicle routing with order restrictions admits efficient approximation in Euclidean spaces, even under moderately growing vehicle capacities and relaxed time windows, but becomes significantly harder on trees. Interesting open directions are to prove NP-hardness for the plane, or to prove NP-hardness or find a PTAS on trees for constant c .

References

- [1] A. Adamaszek, A. Czumaj, and A. Lingas. PTAS for k -tour cover problem on the plane for moderately large values of k . *International Journal of Foundations of Computer Science*, 21(06):893–904, 2010.
- [2] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41(1):153–180, 1994.
- [3] A. Das and C. Mathieu. A quasi-polynomial time approximation scheme for Euclidean capacitated vehicle routing. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 390–403, 2010.
- [4] M. Haimovich and A. Rinnooy Kan. Bounds and heuristics for capacitated routing problems. *Mathematics of Operations Research*, 10(4):527–542, 1985.
- [5] C. Mathieu and H. Zhou. A PTAS for capacitated vehicle routing on trees. *ACM Transactions on Algorithms*, 19(2):1–28, 2023.
- [6] R. Sitters. An $n^{O(\log \log n)}$ time approximation scheme for capacitated VRP in the Euclidean plane. *CoRR*, abs/2507.15549, 2025.

A 3.3904-Competitive Online Algorithm for List Update with Uniform Costs ^{*}

Mateusz Basiak [†] Marcin Bienkowski [†] Martin Böhm (Speaker) [†]
 Marek Chrobak [‡] Łukasz Jeż [†] Jiří Sgall [§] Agnieszka Tatarczuk [†]

1 Introduction

The List Update problem. In the online *List Update problem* [1, 4], the objective is to maintain a set of items stored in a linear list in response to a sequence of access requests. The cost of accessing a requested item is equal to its distance from the front of the list. After each request, an algorithm is allowed to rearrange the list by performing an arbitrary number of swaps of adjacent items. In the model introduced by Sleator and Tarjan in their seminal 1985 paper on competitive analysis [6], an algorithm can repeatedly swap the requested item with its preceding item at no cost. These swaps are called *free*. All other swaps are called *paid* and have cost 1 each. As in other problems involving self-organizing data structures, the goal is to construct an *online* algorithm, i.e., operating without the knowledge of future requests.

Sleator and Tarjan proved that the algorithm MOVE-TO-FRONT (MTF), which after each request moves the requested item to the front of the list, is 2-competitive. This ratio is known to be optimal if the number of items is unbounded. Their work was the culmination of previous extensive studies of list updating, including experimental results, probabilistic approaches, and earlier attempts at amortized analysis.

As shown in subsequent work, MTF is not unique — there are other strategies that achieve ratio 2, such as TIMESTAMP or algorithms based on work functions. In fact, there are infinitely many algorithms that achieve ratio 2 [4].

The uniform cost model. Following [5], we will refer to the cost model of [6] as *standard*, and we will denote it here by LUP_S. This model has been questioned in the literature for not accurately reflecting true costs in some implementations [5], with the concept of free swaps being one of the main concerns.

^{*}This submission was first presented at ESA 2025. A complete article is available on arXiv [3].

[†]Institute of Computer Science, University of Wrocław, Poland.

[‡]Department of Computer Science and Engineering, University of California, Riverside, USA.

[§]Computer Science Institute of Charles University, Prague, Czech Republic.

A natural approach to address this concern, considered in some later studies (see, e.g., [5, 4, 2]), is simply to charge cost 1 for *any* swap. We will call it here the *uniform cost model* and denote it LUP_1 . A general lower bound of 3 on the competitive ratio of deterministic algorithms for LUP_1 was given by Reingold *et al.* [5]. Changing the cost of free swaps from 0 to 1 at most doubles the cost of any algorithm, so MTF is no worse than 4-competitive for LUP_1 . Surprisingly, no algorithm is known to beat MTF, i.e., achieve ratio lower than 4.¹

Our main result. To address this open problem, we develop an on-line algorithm FULL-OR-PARTIAL-MOVE (FPM) for LUP_1 with competitive ratio $\frac{1}{8} \cdot (23 + \sqrt{17}) \approx 3.3904$, significantly improving the previous upper bound of 4.

Our algorithm FPM remains 3.3904-competitive even for the *partial cost* function, where the cost of accessing location ℓ is $\ell - 1$, instead of the *full cost* of ℓ used in the original definition of List Update [6]. Both functions have been used in the literature, depending on context and convenience. For any online algorithm, its partial-cost competitive ratio is at least as large as its full-cost ratio, although the difference typically vanishes when the list size is unbounded. We perform our analysis in terms of partial costs.

FPM remains 3.3904-competitive also in the dynamic scenario of List Update that allows operations of insertions and deletions, as in the original definition in [6].

Technical challenges and new ideas. The question whether ratio 3 can be achieved remains open. We also do not know if there is a *simple* algorithm with ratio below 4. We have considered some natural adaptations of MTF and other algorithms that are 2-competitive for LUP_S , but for all we were able to show lower bounds higher than 3 for LUP_1 .

As earlier mentioned, MTF is 4-competitive for LUP_1 . It is also easy to show that its ratio is not better than 4: repeatedly request the last item in the list. Ignoring additive constants, the algorithm pays twice the length of the list at each step, while any algorithm that just keeps the list in a fixed order, pays only half the length on the average.

The intuition why MTF performs poorly is that it moves the requested items to front “too quickly”. For the aforementioned adversarial strategy against MTF, ratio 3 can be obtained by moving the items to front only *every other time* they are requested. This algorithm, called DBIT, is a deterministic variant of algorithm BIT from [5] and a special case of algorithm COUNTER in [5], and it has been also considered in [4]. In the full version of the paper [3], we show that DBIT is not better than 4-competitive in the partial cost model, and not better than 3.302-competitive in the full cost model.

One can generalize these approaches by considering a more general class of algorithms that either leave the requested item at its current location or move it to the front. We show that such a strategy cannot achieve ratio better than

¹The authors of [4] claimed an upper bound of 3 for LUP_1 , but later discovered that their proof was not correct (personal communication with S. Kamali).

3.25, even for just three items. A naive fix would be, for example, to always move the requested item half-way towards front. This algorithm is even worse: its competitive ratio is at least 6 (both those proofs are in the full version of the paper [3]).

We also show via a computer-aided argument that, for LUP_1 , the work function algorithm's competitive ratio is larger than 3, even for lists of length 5. This is in contrast to its performance for the standard model, where it achieves optimal ratio 2.

Our algorithm FPM overcomes the difficulties mentioned above by combining a few new ideas. The first idea is a more sophisticated choice of the target location for the requested item. That is, aside from *full moves* that move the requested items to the list front, FPM sometimes performs a *partial move* to a suitably chosen target location in the list. This location roughly corresponds to the front of the list when this item was requested earlier.

The second idea is to keep track, for each pair of items, of the work function for the two-item subsequence consisting of these items. These work functions are used in two ways. First, they roughly indicate which relative order between the items in each pair is “more likely” in an optimal solution. The algorithm uses this information to decide whether to perform a full move or a partial move. Second, the simple sum of all these pair-based work functions is a lower bound on the optimal cost, which is useful in analyzing the competitive ratio of FPM.

References

- [1] S. Albers. Online list update. In *Encyclopedia of Algorithms*, pages 598–601. Springer, 2008. doi:10.1007/978-0-387-30162-4_266.
- [2] Y. Azar, S. Lewkowicz, and D. Vainstein. List update with delays or time windows. In *Proc. ICALP*, pages 15:1–15:20. Schloss Dagstuhl, 2024. doi:10.4230/LIPICS.ICALP.2024.15.
- [3] M. Basiak, M. Bienkowski, M. Böhm, M. Chrobak, Ł. Jeż, J. Sgall, and A. Tatarczuk. A 3.3904-competitive online algorithm for list update with uniform costs. *arXiv preprint*, 2025.
- [4] S. Kamali and A. López-Ortiz. A survey of algorithms and models for list update. In *Space-Efficient Data Structures, Streams, and Algorithms*, pages 251–266. Springer, 2013. doi:10.1007/978-3-642-40273-9_17.
- [5] N. Reingold, J. Westbrook, and D. D. Sleator. Randomized competitive algorithms for the list update problem. *Algorithmica*, 11(1):15–32, 1994. doi:10.1007/BF01294261.
- [6] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, 1985. doi:10.1145/2786.2793.

Don't Melt Your Cache: Low-Associativity with Heat-Sink

Michael A. Bender^{*} Alex Conway[†] Daniel DeLayo (Speaker)^{*}
 Martin Farach-Colton[‡] Jaehyun Han[§] Linfeng He[¶] Rob Johnson^{||}
 Sudarsun Kannan[¶] William Kuszmaul^{**} Donald Porter^{††}
 Evan West^{‡‡}

1 Introduction

In the *paging problem* [6], one must maintain a *cache* capable of storing up to n pages at a time. The goal is to support a sequence x_1, x_2, \dots of accesses to pages. If a page x_i is not in cache when accessed, then the access is said to be a *cache miss*, and the paging algorithm is required to put x_i in cache. This may force the algorithm to *evict* another page y from cache. The choice of which page to evict, also known as the *eviction policy*, is the algorithmic knob that distinguishes different solutions to the paging problem and determines how resources are managed.

Given the sequence x_1, x_2, \dots , the *offline optimal algorithm* OPT is any algorithm that minimizes the overall number of cache misses. An online algorithm alg is said to be α -*competitive* with OPT, using β *resource augmentation*, if the total (expected) number of cache misses that alg incurs using a cache of size n is at most α times the total number of cache misses that OPT incurs on a cache of size n/β .

The canonical solution to the online paging problem is to use *LRU eviction*, which is the resource management policy of always evicting the least recently accessed page out of those in cache. In a seminal 1985 paper, Sleator and Tarjan proved that LRU eviction

^{*}{bender, ddelayo}@cs.stonybrook.edu. Department of Computer Science, Stony Brook University, USA.

[†]me@ajhconway.com. Department of Computer Science, Cornell Tech, USA.

[‡]martin@farach-colton.com. Department of Computer Science and Engineering, New York University, USA.

[§]jaehyun@inu.ac.kr. Department of Computer Science and Engineering, Incheon National University, South Korea.

[¶]{lh818@scarletmail.rutgers.edu, sudarsun.kannan@rutgers.edu}. Department of Computer Science, Rutgers, USA.

^{||}rob@robjohnson.io. VMware Research, USA.

^{**}william.kuszmaul@gmail.com. Department of Computer Science, Carnegie Mellon University, USA.

^{††}porter@cs.unc.edu. Department of Computer Science, The University of North Carolina at Chapel Hill, USA.

^{‡‡}evan.ts.west@gmail.com. Meta Research. USA.

is a provably good policy: with resource augmentation 2, it is 2-competitive with OPT. This result is essentially optimal in the sense that any $O(1)$ -competitive policy *must* use $1 + \Omega(1)$ resource augmentation [6].

Caches with Low Associativity. LRU itself, however, it not implementable on most modern caches. This is because most caches [3, 4] are designed with an additional restriction known as ***d-way associativity*** or *d-associativity* for short, in which each page x has a set of only d positions $h_1(x), h_2(x), \dots, h_d(x) \in [n]$ where it is allowed to reside in cache. When a page x is brought into cache, the eviction algorithm must respect the associativity: the page y that we evict must be one of the d pages in positions $h_1(x), h_2(x), \dots, h_d(x)$.

This paper [1] studies the online resource management problem of ***d-associative paging***, which consists of designing a d -associative cache by both specifying the hash functions h_i and designing a page-eviction algorithm that respects the associativity.

Motivating Questions. Given the centrality of low-associativity caches, it is surprising that some basic questions about low-associativity paging have not been addressed, for example:

- For what values of d can we design a d -associative cache for which d -LRU has low competitive ratio with low resource augmentation?
- For what values of d is there a d -associative design that admits an eviction rule with a low competitive ratio and low resource augmentation?

In this paper, we address these problems and show the surprising results that d -LRU can be a poor eviction rule (such as for $d = o(\log n / \log \log n)$) but that there exist other good eviction rules, even for $d = 2$.

1.1 Results

Part 1: The Downfall of LRU. We begin by considering the simplest low-associativity caching policy: each page x hashes to 2 independent random positions $h_1(x), h_2(x)$ in cache, and each eviction performs LRU on the two eligible positions. We call this **2-LRU**. Our first result is a lower bound: We show that 2-LRU is *not* (α, β) -competitive for any $\alpha, \beta \in O(1)$. This is surprising, because 2-LRU has been proposed as the gold standard for how to manage 2-associative caches [2, 5].

This lower bound extends to moderately large values of d : For any $d \in o(\log n / \log \log n)$, the lower bound holds.

In fact, the constraint that the hashes $h_1(x), \dots, h_d(x)$ are uniformly random is also not necessary. The lower bound continues to hold even if we allow for arbitrary dependencies between the hashes $h_1(x), \dots, h_d(x)$ and even if we allow for each individual $h_i(x)$ to be drawn from an arbitrary distribution satisfying a mild distributional assumption that we call *semi-uniformity*. What this means is that, for $d = o(\log n / \log \log n)$, almost all natural variations of d -associative LRU cannot asymptotically match the performance of fully-associative LRU.

Part 2: The Power of Randomized Choice. Our second result is a positive one. Even though LRU performs poorly, there is another policy, which we call **2-RANDOM**, that we prove is $(O(1), O(1))$ -competitive with OPT. As in 2-LRU, in 2-RANDOM each page hashes to two random positions, but rather than evicting the least recently used of the two, 2-RANDOM simply evicts a *random* one.

It may seem counter-intuitive that random eviction would outperform LRU. However, 2-RANDOM performs well because of a *heat-dissipation* phenomenon, where good decisions tend to be long-lived and poor decisions short-lived.

Part 3: HEAT-SINK LRU. Our final result is a d -associative algorithm that we call **HEAT-SINK LRU**. See Figure 1. Even with very low associativity, HEAT-SINK LRU is able to compete with *fully associative* LRU up to $1 + o(1)$ factors. More precisely, for any $\epsilon \in (0, 1)$, if we implement HEAT-SINK LRU with associativity $d = \text{poly}(\epsilon^{-1})$, then the algorithm is guaranteed to be $(1 + \epsilon, 1 + \epsilon)$ -competitive with LRU. This means that, up to low-order terms, we can get something just as good as LRU even on caches with very low associativity.

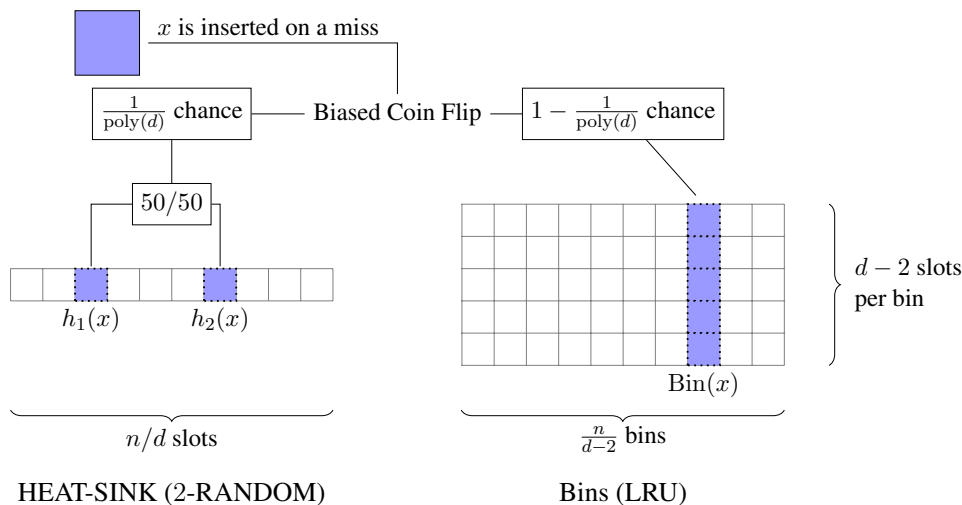


Figure 1: The design of HEAT-SINK LRU. The element x may be stored in any slot in $\text{Bin}(x)$ or in one of 2 HEAT-SINK slots. Thus, these are the only slots we need to check on an access of x . Eviction in the HEAT-SINK is governed by 2-RANDOM and eviction in the bin is governed by LRU. When x is brought into cache, we flip a biased coin to determine if x is to be placed in the HEAT-SINK or $\text{Bin}(x)$. Intuitively, the HEAT-SINK dissipates the heat of the hot (overfilled) bins.

References

- [1] Michael A. Bender, Alex Conway, Daniel DeLayo, Martin Farach-Colton, Jaehyun Han, Linfeng He, Rob Johnson, Sudarsun Kannan, William Kuszmaul, Donald Porter, and Evan West. Don't Melt Your Cache: Low-Associativity with Heat-Sink. In *Proceedings of the 37th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '25, pages 555–565, New York, NY, USA, July 2025. Association for Computing Machinery.
- [2] A. Djordjalian. Minimally-skewed-associative caches. pages 100–107. IEEE Comput. Soc, 2002.
- [3] Giovanni Gracioli, Ahmed Alhammad, Renato Mancuso, Antônio Augusto Fröhlich, and Rodolfo Pellizzoni. A survey on cache management mechanisms for real-time embedded systems. *ACM Computing Surveys*, 48:1–36, 11 2015.
- [4] Swadhash Kumar and P K Singh. An overview of modern cache memory and performance analysis of replacement policies. pages 210–214. IEEE, 3 2016.
- [5] André Seznec. A case for two-way skewed-associative caches. *ACM SIGARCH Computer Architecture News*, 21:169–178, 5 1993.
- [6] Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28:202–208, 2 1985.

Is competitive paging an artifact?

Enoch Peserico *

Michele Scquizzato (Speaker) †

1 Paging

The memory system of computing devices is organized as a hierarchy of several layers of progressively larger capacity but also higher access cost, in terms of both time and energy. The most widely studied model involves two layers: a smaller *memory* layer with a capacity of k pages (data blocks), and a larger layer of infinite capacity whose pages can only be accessed by first copying them into memory – an operation called a (*page*) *fault*. Given a sequence of pages to access in order, a *paging algorithm* must choose which page(s) to evict from memory, whenever a new page must be copied into it, so as to minimize the total number of faults.

While the simple algorithm LFD that evicts the page accessed furthest in the future is optimal [2], paging is often studied *online*, i.e. evictions can depend only on past requests. A paging algorithm is said to have an (h, k) –*competitive ratio* of (at most) ρ if, for every request sequence, it incurs in expectation with a memory of capacity k at most ρ times as many faults as an optimal prescient algorithm incurs with a memory of capacity $h \leq k$, plus a number of faults independent of the request sequence. The ratio $\frac{k}{h}$ is called *resource augmentation*. Simple deterministic online algorithms such as LRU and FIFO have an (h, k) –competitive ratio of $\frac{k}{k-h+1}$ [4, 2] and thus never fare worse than an optimal *offline* (i.e. prescient) algorithm would on a system with half the memory and twice the access cost.

2 Zero-in paging

A crucial shortcoming of the model above is the implicit assumption that every page/datum begins its existence outside of memory, and thus incurs an inevitable fault when accessed for the first time – a so-called “cold miss”. Instead, a newly computed datum begins its existence in the processor registers, at the very top of the memory hierarchy rather than at the bottom; so no additional time or energy must be spent to access it for further processing.

*enoch@dei.unipd.it. Dipartimento di Ingegneria dell’Informazione, Università degli Studi di Padova, via Gradenigo 6, 35131 Padova, Italy.

†scquizzas@math.unipd.it. Dipartimento di Matematica, Università degli Studi di Padova, via Trieste 63, 35121 Padova, Italy.

This was captured by early theoretical I/O models [3]; but the first work obtaining competitiveness results for online paging [4] considered paging as a special case of list update, with new list elements/pages being placed in the last (i.e. most expensive) list position rather than the first – implicitly assuming that new data begin their existence as far as possible from the processor registers rather than within them. And all subsequent work on online paging has followed suit.

We investigate the consequences of removing this incorrect assumption, designating instead an *arbitrary* set of pages in any given request sequence as “zero-in” and charging no cost for the initial access to any such page. Zero-in pages can represent newly computed data, as well as data inherited at the top of the memory hierarchy from a different process, including one running in parallel.

3 Offline vs. online zero-in paging

Allowing zero-in pages has essentially no impact on offline paging, and LFD remains optimal: it is easy to prove that any paging algorithm that is *zero-independent*, i.e. that for every request sequence σ makes the same eviction choices regardless of the zero-in page set Z , if optimal for $Z = \emptyset$ remains optimal for all Z . In a nutshell, if a paging algorithm is zero-independent, then for any $Z \neq \emptyset$ the number of faults it incurs is reduced by $|Z|$ compared to the case $Z = \emptyset$ (since each page in Z incurs no fault on its first access, while it would have inevitably incurred a fault if $Z = \emptyset$). Hence, if an algorithm ALG could outperform LFD for some σ with $Z = \bar{Z}$, ALG *run as if* $Z = \bar{Z}$ would be zero-independent and would continue to outperform LFD on σ even with $Z = \emptyset$ (since both algorithms would see their cost increase by $|\bar{Z}|$ when transitioning to $Z = \emptyset$).

On the other hand, with zero-in pages, no online paging algorithm can be competitive, even if randomized, even with arbitrary resource augmentation, even with request sequences that are not tailored against it but are instead representative of widely used computational techniques – *beam searches* and *genetic algorithms*. These are basically branch-and-bound techniques for search/optimization problems that, starting from a set of m seed solutions, at each step generate a new set of $M \gg m$ solutions, of which (only) the best m are retained as seeds for the next generation. An offline algorithm knowing *which* are the future seeds, with enough memory to hold $\approx 2m$ solutions can avoid all faults after the first generation. On the other hand, any online algorithm running with at most $M/2 \gg 2m$ memory and no way to predict future seeds evicts at each generation roughly half the seeds of the next – and thus incur $\approx m/2$ faults at each generation, i.e. arbitrarily more faults than the offline algorithm over sufficiently many generations.

The issue is masked if one incorrectly assumes that a new datum inevitably incurs a cold miss, because then *any* paging algorithm (even offline) incurs faults on a significant fraction of its data accesses. This is not the case in practice: considerable effort is always spent to ensure that beam searches and genetic algorithms perform essentially all accesses in memory, by adjusting the size of each generation and/or providing indirect hints to the memory system.

4 Conclusions

The negative results in the previous section appear robust against most existing refinements of competitive analysis developed to mitigate its pessimistic outlook. Only Full-Cost analysis [5], where the cost ratio between a fault and non-fault access is a finite value $\frac{1}{\epsilon}$, can achieve some success in this direction; but it still predicts that the product of the cost and capacity overheads of any online algorithms compared to the offline optimal (i.e. the product of its competitive ratio and resource augmentation) is at least of the same order as $\frac{1}{\epsilon}$.

Ultimately, good performance of paging algorithms when blindly facing adversarial request sequences may simply be a mirage born out of an incorrect assumption. After all, programmers and compilers go to great lengths to craft request sequences that are as “nice” as possible rather than adversarial, exploiting their knowledge of the task at hand and of a system’s architecture. It would not then be surprising if such help were in many cases indispensable for acceptable memory management. Whether a simple model can capture the quintessential elements of this assistance may well be the most important question this work leaves open.

One consequence of our results is in the area of Cache-Oblivious and Cache-Adaptive [1] algorithms, where the cache model is an *Ideal Cache* adopting optimal offline paging, justified with the factor-2-optimality of online paging: switching from an Ideal Cache to a cache of twice the capacity controlled by an online algorithm such as LRU at most doubles the total access cost. This is no longer true with zero-in pages, requiring careful re-evaluation of optimality claims.

On the positive side, a tight characterization of the competitive ratio attainable in the Full-Cost model ($1 + \frac{1-\epsilon}{\epsilon} \frac{h-1}{k}$) shows that classic algorithms such as LRU and FIFO are still optimal among deterministic online paging algorithms, increasing our confidence that their comparative effectiveness is not an artifact of the framework, either theoretical or experimental, used to analyse them.

References

- [1] Michael A. Bender, Roozbeh Ebrahimi, Jeremy T. Fineman, Golnaz Ghasemiasfeh, Rob Johnson, and Samuel McCauley. Cache-adaptive algorithms. In *Proceedings of SODA*, pages 958–971, 2014.
- [2] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
- [3] Jia-Wei Hong and H. T. Kung. I/O complexity: The red-blue pebble game. In *Proceedings of STOC*, pages 326–333, 1981.
- [4] Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, 1985.
- [5] Eric Torng. A unified analysis of paging and caching. *Algorithmica*, 20(2):175–200, 1998.

Edmonds++: Efficiently Coloring more Matroid Intersections

Stephen Arndt ^{*} Benjamin Moseley [†] Kirk Pruhs [‡]
 Michael Zlatin (speaker) [§]

1 Introduction

We study the following matroid optimization problem:

- The input is a collection of matroids $M_1 = (X, \mathcal{I}_1), \dots, M_k = (X, \mathcal{I}_k)$ on common ground set X . The intersection $M = \cap_{i=1}^k M_i$ is defined to be the set system $(X, \cap_{i=1}^k \mathcal{I}_i)$.
- A feasible solution is a coloring of the elements of X such that, for every color, the set of elements assigned that color is independent in M , i.e. independent in each matroid M_i for $i \in [k]$.
- The objective is to *minimize the number of colors* used. For any set system M , we use $\chi(M)$ to denote the chromatic number of M , that is the minimum number of colors necessary for the existence of a feasible coloring.

There is a significant literature within the combinatorics/mathematics community that establishes existential results related to this problem. However, most of the proofs of these results are nonconstructive, in that they are not readily adaptable to yield polynomial-time coloring algorithms. For example, we have:

Theorem 1. [3] *For k general matroids M_1, \dots, M_k , it is the case that $\chi(\cap_{i=1}^k M_i) \leq (2k - 1) \max_{i=1}^k \chi(M_i)$.*

For the intersection of two general matroids ($k = 2$), this can be improved to the following.

Theorem 2. [4] *For two general matroids M_1 and M_2 it is the case that $\chi(M_1 \cap M_2) \leq \chi(M_1) + \chi(M_2)$.*

^{*}Tepper School of Business, Carnegie Mellon University.

[†]Tepper School of Business, Carnegie Mellon University. Supported in part by Google Research Award, NSF grants CCF- 2121744 and CCF-1845146 and ONR Grant N000142212702.

[‡]Computer Science Department, University of Pittsburgh. Supported by National Science Foundation grant CCF-2209654

[§]Computer Science Department, Pomona College

The proofs of Theorems 1 and 2 both hinge on topological fixed-point arguments, deriving from an influential earlier paper by Aharoni and Berger [2] that shows that $\chi(M_1 \cap M_2) \leq 2 \max\{\chi(M_1), \chi(M_2)\}$. These theorems show that in some sense the chromatic number of the intersection of k matroids grows at most linearly with k . Further, there is a lower bound that rules out the possibility of many natural types of sublinear dependence.

Theorem 3. [3] *There are infinitely many natural numbers k , such that there are $k + 1$ partition matroids M_1, \dots, M_{k+1} , each with chromatic number k , and where the intersection of these matroids has chromatic number k^2 .*

However, the algorithmic question of producing such colorings for matroid intersection lags well behind what is known existentially. We now summarize the literature on polynomial-time algorithms for matroid intersection coloring.

- **A single matroid.** Edmonds' algorithm optimally colors the elements in polynomial time [9, 15].
- **Two partition matroids.** This case reduces to edge-coloring a bipartite graph and is solvable in polynomial time [15].
- **Two strongly-base-orderable matroids.** Davies and McDiarmid give a polynomial-time optimal algorithm for any pair of strongly-base-orderable matroids assuming oracle access to the bijective map [8]. Partition, transversal, and gammoid matroids are all strongly-base-orderable; but most notably, graphic matroids are not strongly-base-orderable [5, 15].
- **k partition matroids.** A simple greedy procedure colors the intersection using $1 + \sum_{i=1}^k (\chi(M_i) - 1)$ colors.
- **k combinatorial matroids.** If each of the matroids are one of the standard combinatorial types (graphic, laminar, transversal, partition, or a gammoid), then there is a polynomial-time algorithm that uses $1 + \sum_{i=1}^k (2\chi(M_i) - 1)$ colors [11, 6, 12], as they can be reduced to a partition matroid while increasing the chromatic number by a factor of at most two [11, 6, 12]. In contrast it is known that no such reduction is possible for some binary matroids [13, 1].
- **Hardness.** It is NP-hard to optimally color the intersection of two general matroids, or even the intersection of a graphic matroid and a partition matroid [7, 10], or to optimally color the intersection of three partition matroids [14].

1.1 Results

We work in the standard model where the matroids are accessed via a polynomial-time independence oracle. The main result of this paper is:

Theorem 4. *There is a polynomial-time algorithm that, given a general matroid $M_1 = (X, \mathcal{I}_1)$ and $k - 1$ partition matroids M_2, \dots, M_k , will produce a coloring of the intersection $M = \bigcap_{i=1}^k M_i$ using at most $1 + \sum_{i=1}^k (\chi(M_i) - 1)$ colors.*

The upper bound in Theorem 4 matches the standard upper bound when all of the matroids are partition matroids, improves by one on the existential upper

bound for two general matroids in Theorem 2, and improves by a factor of two on the existential upper bound for k general matroids from Theorem 1. We also show that the runtime of the algorithm is $O(n^3 T(M_1))$, where $n = |X|$ and $T(M_1)$ is the time for a single independence oracle call to the general matroid M_1 . Another immediate corollary of Theorem 4 is:

Corollary 1. *There is a polynomial-time algorithm that, given a general matroid $M_1 = (X, \mathcal{I}_1)$ and $k - 1$ matroids M_2, \dots, M_k that are each of the following types: graphic, laminar, transversal, partition, or a gammoid, will produce a coloring of the intersection $M = \bigcap_{i=1}^k M_i$ that has worst-case approximation ratio $2k - 1$.*

Corollary 1 follows from Theorem 4 by first noting that the chromatic number of the intersection of matroids is at least the maximum chromatic number of the individual matroids and then applying the known efficient reductions of these standard combinatorial matroids to partition matroids [11, 6, 12].

1.2 Techniques

One of the main components in the design of the algorithm that establishes Theorem 4, is a modest generalization of Edmonds' algorithm, which we call Generalized Edmonds' Algorithm. Upon closer examination of the standard proof of correctness for Edmonds' algorithm [15], one can observe that it is sufficient that P is what we will call a color-chordless path, which is roughly a path where there is no chord/shortcut between vertices that involve the same color. A shortest path, as it has no chords, is trivially a color-chordless path. The Generalized Edmonds' Algorithm augments along an arbitrary color-chordless path on each iteration, giving us greater flexibility when trying to adapt the algorithm to coloring multiple matroids.

Next, we design and analyze the algorithm that establishes Theorem 4. At a high level, our algorithm finds a color-chordless path P in the Edmonds digraph for matroid M_1 that is suffix-feasible with respect to each of the partition matroids M_2, \dots, M_k . The path P is suffix-feasible with respect to a matroid M_i if iteratively augmenting along P (as in Edmonds' algorithm) results in a sequence of recolorings where each recoloring is unfettered in M_i at the time of the recoloring. To accomplish this, the algorithm first constructs a subgraph H of the Edmonds digraph G for M_1 , such that in H every source-sink path is a color-chordless path in G . The algorithm then finds a source-sink path P within H , that is suffix-feasible with respect to each of the partition matroids M_2, \dots, M_k , using an iterative greedy algorithm.

References

- [1] Abdolazimi, D., Karlin, A.R., Klein, N., Gharan, S.O.: Matroid partition property and the secretary problem. In: 14th Innovations in Theoretical Computer Science Conference (ITCS 2023). vol. 251, pp. 2:1–2:9. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2023)

- [2] Aharoni, R., Berger, E.: The intersection of a matroid and a simplicial complex. *Transactions of the American Mathematical Society* **358**(11), 4895–4917 (2006)
- [3] Aharoni, R., Berger, E., Guo, H., Kotlar, D.: Intersections of matroids. In: To appear in 9th International Conference on Electronics, Communication and Aerospace Technology (ICECA 2025) (2025)
- [4] Berger, E., Guo, H.: Coloring the intersection of two matroids. *arXiv* (2024)
- [5] Brualdi, R.A.: Induced matroids. In: *Proceedings of the American Mathematical Society*. vol. 29, pp. 213–221 (1971)
- [6] Bérczi, K., Schwarcz, T., Yamaguchi, Y.: List coloring of two matroids through reduction to partition matroids. *SIAM Journal on Discrete Mathematics* **35**, 2192–2209 (2021)
- [7] Bérczi, K., Csáji, G., Király, T.: On the complexity of packing rainbow spanning trees. *Discrete Mathematics* **346**(113297) (2023)
- [8] Davies, J., McDiarmid, C.: Disjoint common transversals and exchange structures. *Journal of the London Mathematical Society* **14**, 55–62 (1976)
- [9] Edmonds, J.: Matroid partition. In: *Mathematics of the Decision Sciences, Part 1, Lectures in Applied Mathematics*, vol. 11, pp. 335–345. American Mathematical Society (1968), proceedings of the Fifth Summer Seminar on the Mathematics of the Decision Sciences, Stanford, California, 1967
- [10] Hörsch, F., Kaiser, T., Kriesell, M.: Rainbow bases in matroids. *SIAM Journal on Discrete Mathematics* **38**(2), 1472–1491 (2024)
- [11] Im, S., Moseley, B., Pruhs, K.: The matroid intersection cover problem. *Operations Research Letters* **49**, 17–22 (2021)
- [12] Leichter, M., Moseley, B., Pruhs, K.: An efficient reduction of a gammoid to a partition matroid. In: 29th Annual European Symposium on Algorithms (ESA 2021). vol. 204, pp. 62:1–62:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2021)
- [13] Leichter, M., Moseley, B., Pruhs, K.: On the impossibility of decomposing binary matroids. *Operations Research Letters* **50**, 623–625 (2022)
- [14] Obszarski, P., Jastrzbski, A.: Edge-coloring of 3-uniform hypergraphs. *Discrete Applied Mathematics* **217** (07 2016). <https://doi.org/10.1016/j.dam.2016.06.009>
- [15] Schrijver, A.: *Combinatorial Optimization: Polyhedra and Efficiency, Algorithms and Combinatorics*, vol. 24. Springer (2003)

Efficiently Coloring the Intersection of General Matroids

Stephen Arndt (Speaker) * Benjamin Moseley † Kirk Pruhs ‡
 Chaitanya Swamy § Michael Zlatin ¶

1 Introduction

We begin by defining the matroid intersection coloring problem.

- The input is a collection of matroids $M_1 = (X, \mathcal{I}_1), \dots, M_k = (X, \mathcal{I}_k)$ on common ground set X . The intersection $M = \cap_{i=1}^k M_i$ is defined to be the set system $(X, \cap_{i=1}^k \mathcal{I}_i)$.
- A feasible solution is a coloring of the elements of X such that, for every color, the set of elements assigned that color is independent in M , i.e. independent in each matroid M_i for $i \in [k]$.
- The objective is to *minimize the number of colors* used. For any set system M , we use $\chi(M)$ to denote the chromatic number of M , that is the minimum number of colors necessary for the existence of a feasible coloring.

There is a significant literature within the combinatorics/mathematics community that establishes existential results related in some way to matroid intersection coloring (and/or packing). However, most of the proofs of these results are nonconstructive, in that they are not readily adaptable to yield polynomial-time coloring algorithms. In particular, many of these results are established with topological fixed-point arguments that employ Sperner's lemma. We now highlight some of these existential results that are most relevant for this paper.

Theorem 1 [2] *For k general matroids M_1, \dots, M_k , it is the case that $\chi(\cap_{i=1}^k M_i) \leq (2k - 1) \max_{i=1}^k \chi(M_i)$.*

For the intersection of two general matroids ($k = 2$), this can be improved to the following.

*sarn@andrew.cmu.edu. Tepper School of Business, Carnegie Mellon University.

†moseleyb@andrew.cmu.edu. Tepper School of Business, Carnegie Mellon University.

‡kirk@cs.pitt.edu. Computer Science Department, University of Pittsburgh.

§cswamy@uwaterloo.ca. Combinatorics & Optimization Department, University of Waterloo.

¶michael.zlatin@pomona.edu. Computer Science Department. Pomona College.

Theorem 2 [4] *For two general matroids M_1 and M_2 it is the case that $\chi(M_1 \cap M_2) \leq \chi(M_1) + \chi(M_2)$.*

The proofs of Theorems 1 and 2 both hinge on topological fixed-point arguments, deriving from an influential earlier paper by Aharoni and Berger [1] that shows that $\chi(M_1 \cap M_2) \leq 2 \max\{\chi(M_1), \chi(M_2)\}$. These theorems show that in some sense the chromatic number of the intersection of k matroids grows at most linearly with k . Further, there is a lower bound that rules out the possibility of many natural types of sublinear dependence.

Theorem 3 [2] *There are infinitely many natural numbers k , such that there are $k + 1$ partition matroids M_1, \dots, M_{k+1} , each with chromatic number k , and where the intersection of these matroids has chromatic number k^2 .*

However, the algorithmic question of producing such colorings for matroid intersection lags well behind what is known existentially. We now provide a short summary of the literature on polynomial-time algorithms for matroid intersection coloring.

- **A single matroid.** Edmonds' algorithm optimally colors the elements in polynomial time [6, 10].
- **k partition matroids.** A simple greedy procedure colors the intersection using $1 + \sum_{i=1}^k (\chi(M_i) - 1)$ colors.
- **k combinatorial matroids.** If each of the matroids are one of the standard combinatorial types (graphic, laminar, transversal, partition, or a gammoid), then there is a polynomial-time algorithm that uses $1 + \sum_{i=1}^k (2\chi(M_i) - 1)$ colors [7, 5, 8], as all of these types of combinatorial matroids can be efficiently reduced to a partition matroid at a cost of increasing the chromatic number by at most a factor of two [7, 5, 8].
- **1 general matroid and $k - 1$ combinatorial matroids.** If one matroid is general and the remaining $k - 1$ matroids are partition, then there is a polynomial-time algorithm that uses $1 + \sum_{i=1}^k (\chi(M_i) - 1)$ colors [3]. The algorithm uses a novel approach to combine Edmonds' algorithm for optimally coloring a single matroid with the greedy algorithm for coloring partition matroids. By leveraging the efficient reductions of the standard combinatorial matroids to partition matroids, this can be extended to one general matroid and $k - 1$ combinatorial matroids using $1 + \sum_{i=1}^k (2\chi(M_i) - 1)$ colors.
- **k general matroids.** There are no nontrivial approximation algorithms for the full matroid intersection coloring problem, even for $k = 2$ general matroids. A standard reduction to set cover yields an $O(k \log n)$ -approximation.
- **Instance optimality hardness.** It is NP-hard to optimally color the intersection of two general matroids.

1.1 Our Results

We work in the standard model where the matroids are accessed via a polynomial-time independence oracle. The main result of this paper is:

Theorem 4 Let $M_i = (N, \mathcal{I}_i)$ for $i \in [k]$ be k matroids on common ground set N . Then there exists a polynomial-time algorithm to compute a $(k^2 - k) \max_{i=1}^k \chi(M_i)$ -coloring of $M = \cap_{i=1}^k M_i$.

Because the maximum chromatic number of any individual matroid is trivially a lower bound on the chromatic number of the intersection, this yields a $(k^2 - k)$ -approximation for the matroid intersection coloring problem. This is the first $O(1)$ -approximation algorithm for matroid intersection coloring for $k = O(1)$ general matroids. Further, for the well-studied $k = 2$ case, as a direct corollary we give the first constructive proof that $\chi(M_1 \cap M_2) \leq 2 \max(\chi(M_1), \chi(M_2))$.

Theorem 5 Let $M_1 = (N, \mathcal{I}_1)$ and $M_2 = (N, \mathcal{I}_2)$ be matroids on common ground set N . Then there exists a polynomial-time algorithm to compute a $2 \max(\chi(M_1), \chi(M_2))$ -coloring of $M_1 \cap M_2$.

The algorithm is based on an iterative refinement algorithm from [9]. We can use the iterative refinement algorithm to reduce k -matroid intersection coloring to $(k + 1)$ -matroid intersection, following the standard reduction of single-matroid coloring to two-matroid intersection. A black-box reduction yields a k^k -approximation for k -matroid intersection coloring; our stronger analysis yields a $(k^2 - k)$ -approximation.

References

- [1] Ron Aharoni and Eli Berger. The intersection of a matroid and a simplicial complex. *Transactions of the American Mathematical Society*, 358(11):4895–4917, 2006.
- [2] Ron Aharoni, Eli Berger, He Guo, and Dani Kotlar. Intersections of matroids. In *To appear in 9th International Conference on Electronics, Communication and Aerospace Technology (ICECA 2025)*, 2025.
- [3] Stephen Arndt, Benjamin Moseley, Kirk Pruhs, and Michael Zlatin. Efficiently coloring the intersection of a general matroid and partition matroids. In *Proceedings of the International Conference on Integer Programming and Combinatorial Optimization (IPCO 2026)*, 2026. To appear. Available at <https://arxiv.org/abs/2508.19473>.
- [4] Eli Berger and He Guo. Coloring the intersection of two matroids. *arXiv*, 2024.
- [5] Kristof Bérczi, Tamás Schwarcz, and Yutaro Yamaguchi. List coloring of two matroids through reduction to partition matroids. *SIAM Journal on Discrete Mathematics*, 35:2192–2209, 2021.

- [6] Jack Edmonds. Matroid partition. In *Mathematics of the Decision Sciences, Part 1*, volume 11 of *Lectures in Applied Mathematics*, pages 335–345. American Mathematical Society, 1968. Proceedings of the Fifth Summer Seminar on the Mathematics of the Decision Sciences, Stanford, California, 1967.
- [7] Sungjin Im, Benjamin Moseley, and Kirk Pruhs. The matroid intersection cover problem. *Operations Research Letters*, 49:17–22, 2021.
- [8] Marilena Leichter, Benjamin Moseley, and Kirk Pruhs. An efficient reduction of a gammoid to a partition matroid. In *29th Annual European Symposium on Algorithms (ESA 2021)*, volume 204, pages 62:1–62:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.
- [9] André Linhares, Neil K. Olver, Chaitanya Swamy, and Rico Zenklusen. Approximate multi-matroid intersection via iterative refinement. *Mathematical Programming*, 183:397–418, 2020.
- [10] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer, 2003.

Approximating Matroid Basis Testing for Partition Matroids using Budget-In-Expectation

Lisa Hellerstein ^{*} Benedikt M. Plank [†] Kevin Schewior (Speaker) [‡]

1 Introduction

Stochastic Boolean Function Evaluation (SBFE) problems are a class of information-acquisition problems in stochastic environments. In this work, we focus on the fundamental unit-cost version. Here, one is given a (compactly represented) function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and Boolean random variables x_1, \dots, x_n , where for all $i \in [n]$ the variable x_i is 1 (we also say element i is *active*) independently with known probability $p_i \in (0, 1)$. The function $f(x_1, \dots, x_n)$ must be evaluated by testing variables sequentially for their values. In other words, testing must continue until a certificate for the value of the function has been found.

The problem is to (adaptively) determine the order in which to perform the tests so as to minimize the expected number of tests performed. An algorithm for such a problem is not required to output the explicit testing strategy (decision tree); it is merely required to output, in any situation, the next test to perform. For a recent survey of results in the area, see [7].

In this work, we introduce the following framework closely related to several directions considered in the literature [3, 4, 6]. Let $\mathcal{M} = (E, \mathcal{I})$, be a matroid, where $E = \{1, \dots, n\}$ is the ground set and \mathcal{I} is the independence system. We define $f^{\mathcal{M}}$ to be a function that is 1 if and only if there is a basis of \mathcal{M} that consists only of active elements (variables). We refer to the SBFE problem for this function, with the goal of minimizing the expected *number* of queries (equivalently, assuming unit testing costs), as the *Matroid Basis Testing* problem (MBT).

The case when \mathcal{M} is a uniform matroid (i.e., $E' \in \mathcal{I}$ iff $|E'| \leq k$ for some given k) is identical to the SBFE problem for k -of- n functions. The case when \mathcal{M} is a graphical matroid (i.e., $E' \in \mathcal{I}$ iff (V, E') is an acyclic subgraph of a given graph $G = (V, E)$) is the global-connectivity version of the aforementioned s - t connectivity function.

^{*}lisa.hellerstein@nyu.edu. Department of Computer Science and Engineering, New York University, United States.

[†]b.plank@mail.de. Berlin, Germany.

[‡]k.schewior@uni-koeln.de. Department of Mathematics and Computer Science, University of Cologne, Germany, and Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark.

While some of our results concern more general problems, the bulk of this work focuses on MBT when \mathcal{M} is a partition matroid. In this case, the matroid \mathcal{M} is represented by a partition $\mathcal{P} = \{P_1, \dots, P_d\}$ of E as well as upper bounds k_1, \dots, k_d , such that $E' \in \mathcal{I}$ iff $|E' \cap P_i| \leq k_i$ for all $i \in \{1, \dots, d\}$. Our main result is the following.

Theorem 1 *There is a polynomial-time $O(1)$ -approximation algorithm for MBT on partition matroids.*

The MBT problem on partition matroids is equivalent to the SBFE problem for Boolean functions $f = g_1 \wedge g_2 \wedge \dots \wedge g_d$, where the g_i are defined on disjoint sets of variables, and each g_i is a k_i -of- n_i function, for some $1 \leq k_i \leq n_i$. For example, $f(x_1, x_2, x_3, x_4, x_5) = g_1(x_1, x_2) \wedge g_2(x_3, x_4, x_5)$ where $g_1(x_1, x_2)$ is the 1-of-2 function (Boolean OR), and $g_2(x_3, x_4, x_5)$ is the 2-of-3 function (majority). Because the k_i -of- n_i functions are linear threshold functions, and the g_i are defined on disjoint sets of variables, the MBT problem on partition matroids can be shown to be a special case of Explainable Stochastic d -Halfspace Evaluation. Ghuge et al. presented an approximation algorithm for that more general problem, with an approximation factor of $O(d^2 \log d)$ [2].

A special case of MBT on partition matroids is where each $k_i = 1$, equivalently, where f is representable by a *read-once CNF formula*. Along with k -of- n functions, functions represented by read-once CNF (dually, DNF) formulas are among the very few Boolean function classes for which the SBFE problem is known to be exactly solvable by a polynomial-time algorithm [1]. By Boolean duality, the MBT problem on partition matroids is effectively equivalent to the above SBFE problem for the disjunction, rather than conjunction, of the g_i 's. Similarly, the SBFE problem for read-once DNF formulas is effectively equivalent to the SBFE problem for CNF formulas.

Although partition matroids are arguably the next step up from uniform matroids, and seem to be simple, the MBT problem on partition matroids poses a significant new challenge. In terms of the SBFE formulation above, we know how to optimally evaluate each of the g_i on its own because each is a k_i -of- n_i function. But because f is the conjunction of the g_i , we need to decide how and when to switch between testing variables in each of the g_i . A natural approach is to perform the tests in a depth-first order, where once we begin testing the variables of one g_i , we complete the evaluation of that g_i before performing tests on the variables in another g_i . In fact, a depth-first approach is optimal for the special case of read-once CNF functions. However, when the g_i are arbitrary k_i -of- n_i functions, the approximation ratio of a depth-first approach is not bounded by any function of d (the number of g_i). On the other hand, the round-robin approach that Ghuge et al. employ to get their $O(d^2 \log d)$ -approximation for Explainable Stochastic d -Halfspace Evaluation inevitably loses a factor of d . How then, should one decide when to switch from performing tests on variables in one g_i to performing tests in another g_i ? Our approach to answering this question is based on a reduction to a novel type of maximization problem, with a constraint on the *expected* cost. We refer to this type of problem as a “budget-in-expectation problem”.

Our particular budget-in-expectation problem, which we call MP0, is as follows. Given a single g_i , the probabilities p_i , and a budget B , find a strategy that maximizes the probability of obtaining a certificate that $f(x) = 0$, with the constraint that the *expected* cost of the strategy is at most B . Here, the term “in expectation” is essential and causes an arbitrarily large factor (“gap”) in the attainable probability relative to a hard, per-realization constraint on cost. This is obvious for $B < 1$, but, more importantly, it also holds for arbitrarily large budgets. Algorithmically, with a constraint on expected cost, the challenge is to figure out when test results are promising enough to justify performing further tests, and when one should just abort. Interestingly, for MP0, conducting the tests in the same (adaptive) order as the optimal strategy for the SBFE problem on k -of- n functions does not lead to a constant-factor approximation, even if one aborts this strategy in an optimal way. In this work, we show how to still achieve a constant-factor approximation for MP0. For a more complete version of our paper, see [5].

References

- [1] BOROS, E., AND ÜNLÜYURT, T. Sequential testing of series-parallel systems of small depth. In *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*, M. Laguna and J. L. G. Velarde, Eds. Springer, 2000, pp. 39–73.
- [2] GHUGE, R., GUPTA, A., AND NAGARAJAN, V. Non-adaptive stochastic score classification and explainable halfspace evaluation. In *International Conference on Integer Programming and Combinatorial Optimization (IPCO)* (2022), pp. 277–290.
- [3] GRAMMEL, N., HELLERSTEIN, L., KLETENIK, D., AND LIU, N. Algorithms for the unit-cost stochastic score classification problem. *Algorithmica* 84, 10 (2022), 3054–3074.
- [4] GUO, M., LI, J., NEUMANN, A., NEUMANN, F., AND NGUYEN, H. X. Limited query graph connectivity test. In *AAAI Conference on Artificial Intelligence (AAAI)* (2024), pp. 20718–20725.
- [5] HELLERSTEIN, L., PLANK, B. M., AND SCHEWIOR, K. Approximating Matroid Basis Testing for Partition Matroids using Budget-In-Expectation. In *Symposium on Discrete Algorithms (SODA)* (2026), pp. 2588–2616.
- [6] NIELSEN, M. A., ROHWEDDER, L., AND SCHEWIOR, K. Non-adaptive evaluation of k -of- n functions: Tight gap and a unit-cost PTAS. In *International Conference on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)* (2025), pp. 26:1–26:18.
- [7] ÜNLÜYURT, T. Sequential testing problem: A follow-up review. *Discrete Applied Mathematics* 377 (2025), 356–369.

Randomized online bidding with predictions

Mathis Degryse (speaker)* Imrane Zaakour† Christoph Dürr*
 Spyros Angelopoulos‡

1 Introduction

In the online bidding problem, a player repeatedly submits bids until one is greater than or equal to some hidden threshold u . The cost of the player is the sum of those bids. The *competitive ratio at u* is defined as this cost divided by u , and represents the price for the player of not knowing u . Formally, a strategy is defined by its sequence of bids $(x_i)_{i \in \mathbb{Z}}$, which we can assume to be bi-infinite and strictly increasing. The competitive ratio at u is thus

$$\frac{\sum_{i \leq j} x_i}{u},$$

where j is the index satisfying $x_{j-1} < u \leq x_j$. A *randomized* bidding sequence is a distribution over deterministic sequences, and its competitive ratio for a fixed threshold u is defined as its expected cost divided by u . The competitive ratio of a strategy is the worst-case competitive ratio among all thresholds u .

The problem has been introduced in [CK06; Chr+06] in order to illustrate the power of doubling strategies, which is a central online algorithmic technique. It has also been studied implicitly in other contexts, e.g., in the linear search problem [BN70], which gave the first proof that the best deterministic strategy, namely $(2^i)_{i \in \mathbb{Z}}$, has competitive ratio equal to 4. The randomized version was studied in [Chr+06], where a tight e competitive ratio is achieved by the strategy $(e^{i+\lambda})_{i \in \mathbb{Z}}$ for uniform random $\lambda \in [0, 1]$.

Online bidding has also been studied in a *learning-augmented setting* in which the player is provided with a prediction u_0 on the threshold. In this context, the competitive ratio of a given bidding sequence at threshold equal to u_0 is called its *consistency*, whereas its competitive ratio at worst-case thresholds is called its *robustness*. The objective is to find the best-possible consistency-robustness tradeoffs, i.e., the best consistency subject to a robustness requirement. For deterministic strategies, strategies with optimal consistency-robustness tradeoffs were given in [Ang+20]. The randomized version of the learning-augmented problem

*Sorbonne Université, CNRS, LIP6, Paris, France

†Centrale Supélec, Palaiseau, France, and ILLS, Montreal, Canada

‡CNRS and ILLS, Montreal, Canada

was studied in [AS25], which gave an upper bound on the consistency by analyzing a specific class of algorithms, and a lower bound based on applying Yao’s principle to linear combinations of consistency and robustness.

2 Our contributions

In this work, we first consider a general form of randomized bidding sequences, which can be described by a non-decreasing function $b : \mathbb{R} \rightarrow \mathbb{R}$, and study their robustness-consistency tradeoffs. In the first part, we assume that b is *linear* within each unit interval of the form $[i, i+1)$ for $i \in \mathbb{Z}$. Such a function defines the bidding sequence $(e^{b(i+\lambda)})_{i \in \mathbb{Z}}$ for a uniformly chosen random $\lambda \in [0, 1]$. In particular, we distinguish between *uniform linear* sequences, for which $b(x+1) - b(x)$ is constant, and *continuous linear* sequences, for which b is continuous.

For uniform linear sequences, which have been studied in [AS25], we provide monotone expressions of the consistency and the robustness in terms of a single so-called *randomization* parameter ℓ , and show that for robustness at least $9/2$, the sequences are deterministic ($\ell = 0$). Similarly, for continuous linear sequences, we provide a closed formula of the tradeoffs. These studies indicate that the optimal bidding sequence needs to be neither uniform nor continuous.

In the second part of the work, we study linear programming (LP) formulations of the problem, both from the point of view of upper and lower bounds on the consistency-robustness Pareto curve of optimal sequences. The first LP discretizes the randomness in the sequence, and provides an upper bound on the tradeoff. The second one discretizes the set of possible bids and its dual provides a lower bound on the consistency-robustness tradeoff, see Figure 1. Using the latter LP, in particular, we show that any randomized R -robust sequence has consistency at least $1 + 1/2R + 1/2R^2 + 5/8R^3$. We match this bound by providing a randomized sequence inspired by the solutions to the upper bound LP, which has a consistency of $1 + 1/2R + o(1/R)$ for R greater than some constant. Finally, we discuss notions of *smoothness* that can be proven in connection to our algorithms.

References

- [BN70] Anatole Beck and Donald J Newman. “Yet more on the linear search problem”. In: *Israel journal of mathematics* 8.4 (1970), pp. 419–429.
- [Chr+06] Marek Chrobak, Claire Kenyon, John Noga, and Neal E. Young. “Oblivious Medians Via Online Bidding”. In: *LATIN 2006: Theoretical Informatics, 7th Latin American Symposium, Valdivia, Chile, March 20-24, 2006, Proceedings*. Vol. 3887. Lecture Notes in Computer Science. Springer, 2006, pp. 311–322. DOI: 10.1007/11682462\31.
- [CK06] Marek Chrobak and Claire Kenyon-Mathieu. “SIGACT news online algorithms column 10: Competitiveness via doubling”. In: *ACM SIGACT News* 37.4 (2006), pp. 115–126.

- [Ang+20] Spyros Angelopoulos, Christoph Dürr, Shendan Jin, Shahin Kamali, and Marc Renault. “Online Computation with Untrusted Advice”. In: *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. 2020, pp. 52–1.
- [AS25] Spyros Angelopoulos and Bertrand Simon. “Learning-Augmented Online Bidding in Stochastic Settings”. In: *NeurIPS*. The Thirty-Ninth Annual Conference on Neural Information Processing Systems. 2025.

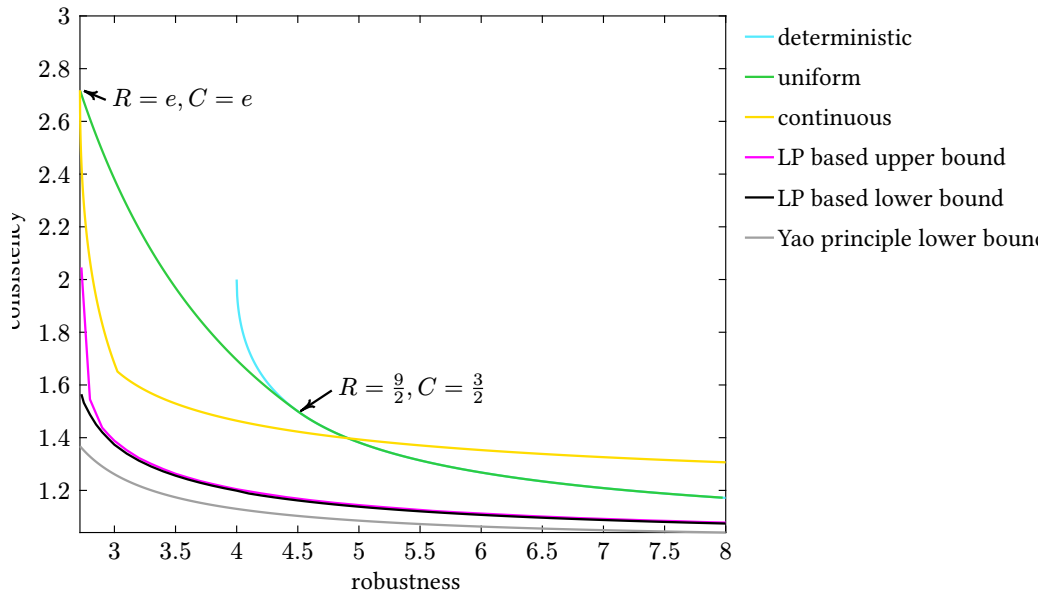


Figure 1: Pareto optimal bidding sequences for several classes of bidding sequences.

Stable Matching with Predictions: Robustness and Efficiency under Pruned Preferences

Samuel McCauley ^{*} Benjamin Moseley (Speaker) [†] Helia Niaparast [‡]
Shikha Singh [§]

1 Introduction

Centralized matching markets such as the National Resident Matching Program (NRMP) are among the most visible successes of market design. Each year, tens of thousands of residents and hospitals submit ranked preference lists, and a centralized algorithm assigns residents to hospitals. In 2024, the NRMP included over 50,000 applicants competing for roughly 40,000 residency positions [4].

The theoretical foundation underlying these two-sided markets is the *stable matching problem* [1]. In the classical problem, the agents on the two sides—the residents and hospitals—submit a ranked list of preferences over the opposite side. The goal of the problem is to find a *stable* matching: an outcome where no pair of agents would both prefer each other over their assigned match (that is, there are no “blocking pairs”). This guarantee of stability is the primary reason for the success and longevity of matching markets like the NRMP [5].

While the classic stable-matching problem assumes that participants submit complete preference lists over all potential partners, this is infeasible in practice. Hospitals must decide whom to interview given limited time and resources. Consequently, they rely on past experience to form expectations about where they will match, adopting a data-driven approach to interviewing.

For example, an elite hospital may interview highly competitive candidates until it is confident it has reached its eventual match. Meanwhile, a mid-tier hospital may skip candidates who are unlikely to accept, focusing instead on a contiguous *window* of mid-ranked applicants who are similar to those who matched there in the past. This behavior reflects a natural adaptation to capacity constraints: each hospital implicitly *predicts* the approximate rank of its eventual match and restricts attention to residents near that prediction.

^{*}srm2@williams.edu. Williams College

[†]moseleyb@andrew.cmu.edu. Tepper School of Business, Carnegie Mellon University.

[‡]hniapara@andrew.cmu.edu. Tepper School of Business, Carnegie Mellon University.

[§]ss32@williams.edu. Williams College.

This predictive truncation highlights a difference between how such markets are studied in theory vs how they operate in practice. In this paper, we study two fundamental questions that can help bridge this gap between market design theory and practice:

1. **Market Dynamics (Descriptive):** Does this heuristic behavior threaten the stability of the market? If agents only propose to a “predicted” set of candidates, do we still reach a stable matching?
2. **Algorithmic Efficiency (Prescriptive):** Can we exploit these predictions to design faster algorithms? Specifically, can we bypass the quadratic runtime of standard stable matching algorithms by leveraging predictions of where hospitals will match?

1.1 Our Model: Algorithms with Predictions

We study the stable matching problem with pruned preference lists in the *algorithms with predictions* framework [3, 2]. This new framework, also called *learning-augmented algorithms*, is motivated by the premise that real-world applications repeatedly solve a problem on similar instances that share a common underlying structure. Algorithms designed in this framework leverage predictions about these input instances to optimize efficiency on future computations. This model has been extremely successful in improving algorithmic performance; see Section ??.

In this model, the algorithm is given with a prediction about the input instance and its performance is measured as a function of the prediction quality (as well as the input size). An ideal algorithm in this framework is (a) *consistent*: matches the best-possible performance under perfect predictions, (b) *robust*: is never worse than the state-of-the-art solution even under arbitrarily bad predictions, and (c) *smooth*: it interpolates smoothly between these extremes.

1.2 Stable Matching with Predictions

We study the stable matching problem in the algorithms-with-prediction model. In particular, we assume that each hospital h_j receives a prediction about the resident it is likely to match with. We represent this prediction as the rank ρ_j of this predicted match on h_j 's preference list. Each hospital can then prune their complete preference list using ρ_j and submit these pruned lists to the stable matching algorithm.

We study two variants of the classic deferred-acceptance algorithm. In the first variant, each hospital also receives a prediction error bound η_j . The hospital only considers residents whose ranks lie within the window $[\rho_j - \eta_j, \rho_j + \eta_j]$. In the second variant, the prediction error is unknown a priori and the hospitals prune their lists using ρ_j alone.

We show that our algorithms correctly output the stable matching when run on the pruned lists and satisfy the standard metrics of the framework (consistency, smoothness and robustness). In particular, we show that the performance of the algorithm scales with respect to the size of the pruned list (rather than the full input instance), and it is never worse than the $O(n^2)$ performance of the standard stable matching algorithm even when the predictions are inaccurate.

1.3 Our Contributions

Our analysis reveals that the robustness of stability depends crucially on which side of the market proposes.

- (1) **Robustness of the Resident-Proposing Mechanism.** We show that if there exists a stable matching in which each hospital h_j is matched to a resident whose rank lies in the interval $[\rho_j - \eta_j, \rho_j + \eta_j]$, then running the *resident-proposing* deferred acceptance algorithm on the pruned instance produces a stable matching. This provides a new theoretical justification for the NRMP’s design: it remains stable under bounded prediction errors and natural strategic pruning by hospitals.
- (2) **Instability of the Hospital-Proposing Mechanism.** In contrast, if the *hospital-proposing* algorithm is run on the same pruned instance, the resulting matching can be stable on the restricted graph yet *unstable on the full market*. This highlights a distinct vulnerability: hospital-proposing mechanisms can be destabilized by data-driven truncation. This is particularly relevant given the history of the NRMP, which used a hospital-proposing algorithm in its early years [5].
- (3) **Algorithmic Efficiency and Learning.** When predictions are accurate as in (1), our algorithm runs in $O(\sum_j \eta_j)$ time, a major improvement over the $O(n^2)$ complexity of standard deferred acceptance. We further design an adaptive algorithm that learns the effective η_j values on the fly when only the ranks ρ_j are known, achieving $O(n \max_j \eta_j)$ runtime under a natural oracle assumption for blocking-pair detection. We complement this with a lower bound showing that extra information—in our algorithms, either the given η_j , or the oracle assumption for blocking pair detection—is necessary to achieve nontrivial bounds. Specially, we show that verifying stability using the predicted ranks ρ_j alone requires $\Omega(n^2)$ time.
- (4) **A Robust Doubling Algorithm via Truncation.** We analyze the case where hospitals rank their top candidates but truncate from the tail, keeping residents in $[1, \rho_j]$. It is known that for a single hospital, a perfect matching in the truncated instance implies stability on the full instance. We extend this connection to the case of simultaneous truncation by all hospitals and, more importantly, embed this analysis within the *Algorithms with Predictions* framework. We show that if the pruned instance yields a perfect matching, the outcome is stable on the full instance. Further, any stable matching that has no edge removed in truncation process, is stable on the truncated instance. We leverage this observation to design a **robust doubling algorithm** that adaptively expands the truncation window, guaranteeing stability on the full instance while providing strong efficiency gains under accurate predictions. This captures the realistic scenario where hospitals rank all highly desired candidates up to a specific threshold.

Empirical Validation. We validate our theory through simulations under three different regimes of generating preference lists. Using min- and max-rank predictions, we find that even moderately accurate predictions allow us to recover the stable matching while drastically reducing the number of proposals.

References

- [1] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American mathematical monthly*, 69(1):9–15, 1962.
- [2] Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *Journal of the ACM (JACM)*, 68(4):1–25, 2021.
- [3] Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. *Communications of the ACM*, 65(7):33–35, 2022.
- [4] National Resident Matching Program. Results and data: 2024 main residency match. <https://www.nrmp.org/match-data/2024/06/results-and-data-2024-main-residency-match/>, 2024. Accessed: 2025-12-1.
- [5] Alvin E Roth. The evolution of the labor market for medical interns and residents: a case study in game theory. *Journal of political Economy*, 92(6):991–1016, 1984.

A Subexponential Time Algorithm for Makespan Scheduling of Unit Jobs with Precedence Constraints

Jesper Nederlof* Céline M. F. Swennenhuis†

Karol Węgrzycki (Speaker)‡

1 Introduction

Scheduling of precedence-constrained jobs on identical machines is a central challenge in the algorithmic study of scheduling problems. In this problem, we have n jobs, each one of unit length, along with m identical parallel machines on which we can process the jobs. Additionally, the input contains a set of *precedence constraints* of jobs; a precedence constraint $v \prec w$ states that job v must be completed before job w can be started. The goal is to schedule the jobs non-preemptively in order to minimize the *makespan*, which is the time when the last job is completed. In the 3-field notation of Graham et al. [4] this problem is denoted as $Pm | \text{prec}, p_j = 1 | C_{\max}$. The problem received extensive interest in the research community [1, 2, 5, 6], particularly because it is one of the most fundamental computational scheduling problems with precedence constraints.

Nevertheless, the exact complexity of the problem is still very far from being understood. It is known to be NP-hard when the number of machines is part of the input [7] and polynomially solvable for $m = 2$ machines [1, 2]. In the 1970s, Lenstra and Rinnooy Kan [5] asked to determine whether problem is in P or NP-hard even for $m = 3$.

Open Question 1 ([3, 5, 7]). *Determine the running time complexity of $P3 | \text{prec}, p_j = 1 | C_{\max}$.*

As of today, Open Question 1 remains one of the most notorious open questions in the field. In fact, settling the complexity of $P3 | \text{prec}, p_j = 1 | C_{\max}$ machines is essentially the last remaining open question from the textbook of Garey and Johnson [3].

An obvious goal is to improve upon the exponential time algorithm and solve the problem significantly faster than $2^{\mathcal{O}(n)}$, even when $m = 3$. We present the first exact subexponential time algorithm for the problem.

*Utrecht University, The Netherlands, j.nederlof@uu.nl.

†Eindhoven University of Technology, The Netherlands, c.m.f.swennenhuis@tue.nl.

‡Max Planck Institute for Informatics, Saarbrücken, Germany, kwegrzyc@mpi-inf.mpg.de.

Theorem 1.1. $Pm | \text{prec}, p_j = 1 | C_{\max}$ admits an $(1 + \frac{n}{m})^{\mathcal{O}(\sqrt{nm})}$ time algorithm.

Note that for $m = \mathcal{O}(1)$, this algorithm runs in $2^{\mathcal{O}(\sqrt{n} \log n)}$ time and is subexponential when $m = o(n)$.

References

- [1] Edward G. Coffman and Ronald L. Graham. Optimal scheduling for two-processor systems. *Acta informatica*, 1(3):200–213, 1972.
- [2] Harold N. Gabow. An almost-linear algorithm for two-processor scheduling. *J. Assoc. Comput. Mach.*, 29(3):766–780, 1982.
- [3] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- [4] Ronald L. Graham, Eugene L. Lawler, Jan Karel Lenstra, and Alexander H.G. Rinnooy Kan. Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. *Annals of Discrete Mathematics*, 5(2):287–326, 1979.
- [5] Jan Karel Lenstra and Alexander H.G. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Operations Research*, 26(1):22–35, 1978.
- [6] Ravi Sethi. Scheduling graphs on two processors. *SIAM J. Comput.*, 5(1):73–82, 1976.
- [7] Jeffrey D. Ullman. NP-complete scheduling problems. *Journal of Computer and System sciences*, 10(3):384–393, 1975.

A tight double exponentially lower bound for high multiplicity bin packing

Klaus Jansen (Speaker) * Felix Ohnesorge † Lis Pirotonn ‡

1 Introduction

The BIN PACKING problem is a classic optimization problem with many applications.

Definition 1 (Bin Packing) *Given are $d \in \mathbb{Z}_{>0}$ item types of sizes $s = (s_1, \dots, s_d) \in (0, B]^d$ and item multiplicities $a = (a_1, \dots, a_d) \in \mathbb{Z}_{>0}^d$. The BIN PACKING problem asks to find the minimum number of bins of size B to pack all items.*

The BIN PACKING problem is also known as the (1-dimensional) CUTTING STOCK problem and its study goes back to the classical paper by Gilmore and Gomory [3]. While strongly NP-hard in general, a major research direction has focused on parameterized algorithms for the *high-multiplicity* setting, where d is assumed to be a small parameter. A breakthrough result in this area came in 2014 from Goemans and Rothvoss [4] who proved that it is polynomial for constant d . This answered an open question posed by McCormick, Smallwood and Spieksma [10] as well as by Eisenbrand and Shmonin [2].

To prove the result for BIN PACKING, Goemans and Rothvoss [4] study the more general CONE AND POLYTOPE INTERSECTION problem, defined as follows: Given two polytopes $\mathcal{P}, \mathcal{Q} \subseteq \mathbb{R}^d$: Is there a point in \mathcal{Q} that can be expressed as a non-negative integer combination of integer points in \mathcal{P} ? They gave an algorithm for this feasibility problem with time complexity $|\mathcal{P}|^{2^{O(d)}} \cdot |\mathcal{Q}|^{O(1)}$, where $|\mathcal{R}|$ denotes the encoding length of a polytope \mathcal{R} . Additionally, they showed how to reduce each BIN PACKING instance from a CONE AND POLYTOPE INTERSECTION instance. In this reduction, $\mathcal{P} = \left\{ \binom{x}{1} \in \mathbb{R}_{\geq 0}^{d+1} \mid s^T x \leq B \right\}$ (the knapsack polytope) contains all possible *configurations* (i.e. multisets of items that fit into a single bin), and $\mathcal{Q} = \{a\} \times [0, k]$ is constructed to encode the target item vector a and the number of bins k . This yields via binary search over k an algorithm for BIN PACKING with runtime $|I|^{2^{O(d)}}$, where $|I|$ denotes the encoding length of the instance. For $d = O(1)$ the encoding length $|I| = O(\log(\Delta))$, where Δ is the maximum over all multiplicities in a and sizes in s and B . Goemans and Rothvoss [4] wrote in their paper:

A natural open problem that arises from this work is whether the double exponential running time is necessary.

*kj@informatik.uni-kiel.de. Kiel University, Germany.

†foh@informatik.uni-kiel.de. Kiel University, Germany.

‡lpi@informatik.uni-kiel.de. Kiel University, Germany.

2 Related Work

Recent work has highlighted the inherent complexity related to this parameterization. Kowalik, Lassota, Majewski, Pilipczuk, and Sokolowski [9] proved an ETH-tight lower bound for the POINT IN CONE problem (where the second polytope \mathcal{Q} in the CONE AND POLYTOPE INTERSECTION instance consists of just one point q), showing that under the Exponential Time Hypothesis (ETH) a doubly exponential dependency on d is unavoidable for a general polytope \mathcal{P} with an exponential of inequalities.

Definition 2 (ETH, [5]) *The ETH states that 3-SAT cannot be solved in subexponential time, i.e. there exists a $\delta > 0$ such that 3-SAT can not be solved in time $2^{\delta n}$ for n variables.*

As proved in [6], this implies that there is no algorithm for 3-SAT with running time $2^{o(n+m)}$, where m denotes the number of clauses in the formula; see also Theorem 14.4 in [1].

Furthermore, the structure of solutions to the BIN PACKING problem with d item sizes is known to be complex. Eisenbrand and Shmonin [2] proved via an elegant combinatorial argument that there is always an optimum solution for BIN PACKING with a support (the number of distinct configurations needed) bounded by 2^d . Recently, Jansen, Piroton, and Tutas [7] showed that the support of any optimum solution in a BIN PACKING instances can be exponential in d . This structural hardness provides further evidence that a doubly exponential runtime may be optimal.

3 Our Contribution

We answer the open problem above by confirming that the algorithm by Goemans and Rothvoss [4] is optimal for BIN PACKING, assuming the ETH.

Theorem 3 *There is no algorithm solving high-multiplicity BIN PACKING with d distinct item sizes in time $|I|^{2^{o(d)}}$, unless the ETH fails.*

To achieve this result, we introduce a novel reduction from 3-SAT. A key component of our reduction is an efficiently encoding a 3-SAT instance with n variables into an Integer Linear Program (ILP) formulation where the number of variables and equalities is only *logarithmic* in n . This ILP is then transformed into a family of BIN PACKING instances $\mathcal{I}^{\text{BP}}(\hat{\chi})$ with $d = O(\log(n))$ distinct item sizes. Here $\hat{\chi}$ is a vector encoding some, in polynomial time computable, extra information of the 3-SAT instance. This compact encoding allows us to translate the $2^{o(n)}$ lower bound for 3-SAT into the desired $|I|^{2^{o(d)}}$ lower bound for BIN PACKING.

The inspiration for our reduction technique stems from Kaibel and Weltge [8]. In their work, they study lower bounds on the sizes of ILPs without additional variables and restate an observation from Schrijver [11] that any language in NP can be expressed with a compact ILP. Our reduction leverages this: We compactly encode algorithmic ideas in a low-dimensional ILP, while preserving the problems complexity.

We firmly believe that this reduction technique, particularly the flexible encoding via an ILP, is of independent interest. It demonstrates a powerful pattern for establishing lower bounds that can likely be adapted to prove similar results for other problems. such

as for high multiplicity ILPs with few constraints, m -dimensional knapsack, multiple knapsack, and scheduling problems as well as high multiplicity block structured n -fold and 2-stage ILPs.

References

- [1] M. CYGAN, F.V. FOMIN, L. KOWALIK, D. LOKSHANOV, D. MARX, M. PILIPCZUK, M. PILIPCZUK, AND S. SAURABH, *Parameterized Algorithms*, Springer (2015).
- [2] F. EISENBRAND AND G. SHMONIN. *Carathéodory bounds for integer cones*, Operations Research Letters, 34 (2006), 564–568.
- [3] P.C. GILMORE AND R.E. GOMORY. *A Linear Programming Approach to the Cutting-Stock Problem*. Operations research, 9 (1961), 849–859.
- [4] M. GOEMANS AND T. ROTHVOSS. *Polynomiality for Bin Packing with a Constant Number of Item Types*. Journal of the ACM 67 (2020), 38:1–38:21.
- [5] R. IMPAGLIAZZO, R. PATURI AND F. ZANE. *Which Problems Have Strongly Exponential Complexity?*. Symposium on Foundations of Computer Science, (FOCS 1998), 653–663.
- [6] R. IMPAGLIAZZO, R. PATURI AND F. ZANE. *Which Problems Have Strongly Exponential Complexity?*, Journal of Computer and System Sciences, 63 (2001), 512–530.
- [7] K. JANSEN, L. PIROTTON, AND M. TUTAS. *The Support of Bin Packing is Exponential*. European Symposium on Algorithms (ESA 2025), 48:1–48:16.
- [8] V. KAIBEL AND S. WELTGE, *Lower Bounds on the sizes of Integer Programs without Additional Variables*, Mathematical Programming 154 (2015), 407–425.
- [9] L. KOWALIK, A. LASSOTA, K. MAJEWSKI, M. PILIPCZUK, AND M. SOKOLOWSKI, *Detecting Points in Integer Cones of Polytopes is Double-Exponentially Hard*, Symposium on Simplicity in Algorithms, SOSA 2024, 279–285.
- [10] S.T. MCCORMICK, S.R. SMALLWOOD, AND F.C.R. SPIEKSMAN, *A Polynomial Algorithm for Multiprocessor Scheduling with Two Job Lengths*. Mathematics of Operations Research 26 (2001), 31–49.
- [11] A. SCHRIJVER. *Theory of Linear and Integer Programming*. Wiley, 1999.

To delay or not to delay - Online Span Minimization

Samir Khuller†(*Speaker*)* Mohzhengfu Liu † Xueyan Tang ‡

1 Introduction

Motivated by the need to reduce energy utilization, we study the problem called “span minimization”, which is closely related to the well-known BusyTime problem [5, 3, 7, 6]. In the most general form in the BusyTime problem, we are given access to virtual machines with batch capacity B and a collection of n jobs, each with a release time, deadline and processing time (denoted by r_i, d_i, p_i respectively). The goal is to partition n jobs into bundles S_1, S_2, \dots, S_k with the following property. Each bundle contains a collection of jobs that can be feasibly scheduled such that no more than B jobs are running simultaneously. The cost of this bundle is simply the duration for which the machine is on (from the earliest start time of any job in the bundle to the latest completion time of any job in the bundle). We can call this $c(S_i) = e(S_i) - s(S_i)$ where $e(S_i)$ is the ending time of the set of jobs S_i and $s(S_i)$ is the start time of a set of jobs S_i . The total cost is the sum of all bundles’ costs ($\sum_i c(S_i)$). Note that we can create an unbounded number of bundles, since each is simply a rented virtual machine (VM) with some number of parallel processors.

Since the problem is NP-hard [9] (even on the rather special case of interval (or rigid jobs) where $d_i = p_i + r_i$), significant research was done on approximation algorithms for it. The earliest algorithms for the problem only focused on the special case of interval jobs [1, 8] and gave a 2 approximation. Subsequently, Khandekar et al [6] developed a 4 approximation for the general problem, but this works in two steps. *In the first step* solve the problem of *minimizing span*, or assuming unbounded batch capacity ($B = \infty$) to fix the starting times of the jobs. In other words, imagine that you have an infinite capacity machine - how would you schedule jobs respecting starting and ending times to minimize the cost of the schedule (the duration for which the machine is on). This is what we refer to as the “span” of the set of jobs. We use this to place the jobs optimally by setting the start times, this reduces to the special case of BusyTime problem for Interval Jobs (since there is no flexibility any longer as to when the jobs start), and even this problem is NP-hard, but multiple algorithms are known that give a 2 approximation for it as mentioned earlier.

*Supported by NSF and Adobe Research Gift

†Northwestern University

‡Nanyang Technological University

However, overall this approach yields a 4 approximation for the BusyTime problem as the optimum solution might increase by a factor of 2 when we reduce to an interval instance (as shown by Chang et al [3]). Chang et al [3] develop a 3 approximation via a different approach, although in the first step we still need to solve the problem of minimizing span for unbounded batch capacity.

The only known method for minimizing span is based on dynamic programming and is extremely inefficient (with a very high polynomial run time)[6]. This prohibitive running time, is the main bottleneck in all good approximation algorithms for BusyTime Scheduling. Although the algorithms guarantee solutions within a factor of 3 of the optimal cost, as mentioned the issue is the algorithms are simply not practical due to the first phase.

On the other hand, a very simple *online* (deterministic) algorithm to minimize the span (Algorithm DOUBLER) gives a 5 approximation [7]. In recent work, Liu (one of the authors) showed that no deterministic online algorithm can get a bound better than 4. While, the question remains, how to close the gap between 4 and 5 for deterministic online algorithms - we turn our attention to how randomization could help get past the deterministic lower bound on the competitive ratio. We first consider unit length jobs, to gain some understanding of this problem.

Understanding the case when all the jobs are the unit length is surprisingly nontrivial in the online setting. The key application would be for a large number of tasks arriving in an online manner, and we have a powerful parallel machine that can schedule all the tasks in parallel. Many scheduling papers have focused on the case of unit jobs as with arbitrary release times and deadlines these problems can be non-trivial. We first show that for this case, there is a tight bound of 2 on the competitive ratio - both the upper and lower bound is exactly 2. In fact Algorithm DOUBLER achieves this bound. Whenever a job needs to be scheduled (otherwise it will miss its deadline), we schedule it (call it a flag job) and plan to run the machine for twice the length of the job. Any jobs (waiting, or released in the immediate future) that can be run for free in this time window are run for free. Then we show something interesting: the upper bound of 2 can be improved via a simple oblivious randomized algorithm called Stretch- γ .

The main issue is that if a job is released (with a very late deadline) just before the flag job ends, our algorithm will always schedule it - this is why we pay the penalty of a factor of 2. Repeat this many times, and all jobs with a late deadline could be scheduled by the optimal solution at the end. The improved randomized algorithm picks γ by a carefully chosen distribution, and rather than doubling the time the machine is on, when a flag job needs to be scheduled, we simply stretch the runtime window by a factor of γ . Once γ is chosen, it is used for the entire algorithm¹.

For *batch scheduling* as well, several problems for unit length jobs have [2] been studied with the goal of minimizing the “on” time for a single batching machine. An efficient offline algorithm called “Lazy Activation” is described for the case of scheduling unit jobs with release times and deadlines. Recently, Davies et al [4]

¹The deterministic case is when $\gamma = 1$.

also consider the case of developing polynomial time algorithms for the tradeoff between energy utilization and average waiting time.

Our main results are as follows:

- Algorithm Doubler is an optimal 2 competitive algorithm.
- Algorithm Random Stretch- γ gives an upper bound of 1.44.
- We prove that no randomized algorithm can give an upper bound better than 1.36 for unit length jobs, with an oblivious adversary.

References

- [1] M. Alicherry and R. Bhatia. Line system design and a generalized coloring problem. In *European Symposium on Algorithms*, pages 19–30. Springer, 2003.
- [2] J. Chang, H. N. Gabow, and S. Khuller. A model for minimizing active processor time. *Algorithmica*, 70(3):368–405, 2014.
- [3] J. Chang, S. Khuller, and K. Mukherjee. LP rounding and combinatorial algorithms for minimizing active and busy time. In *Proceedings of the 26th ACM symposium on Parallelism in algorithms and architectures*, pages 118–127, 2014.
- [4] S. Davies, S. Khuller, and S. Zhang. Balancing flow time and energy consumption. In *Proceedings of the 34th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 369–380, 2022.
- [5] M. Flammini, G. Monaco, L. Moscardelli, H. Shachnai, M. Shalom, T. Tamir, and S. Zaks. Minimizing total busy time in parallel scheduling with application to optical networks. *Theor. Comp. Science*, 411(40-42):3553–3562, 2010.
- [6] R. Khandekar, B. Schieber, H. Shachnai, and T. Tamir. Minimizing busy time in multiple machine real-time scheduling. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2010.
- [7] F. Koehler and S. Khuller. Busy time scheduling on a bounded number of machines. In *Workshop on Algorithms and Data Structures*, pages 521–532. Springer, 2017.
- [8] V. Kumar and A. Rudra. Approximation algorithms for wavelength assignment. In *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science: 25th International Conference, Hyderabad, India, December 15-18, 2005. Proceedings 25*, pages 152–163. Springer, 2005.
- [9] P. Winkler and L. Zhang. Wavelength assignment and generalized interval graph coloring. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 830–831, 2003.

Online Makespan Scheduling under Two Scenarios

Ekin Ergen *

1 Introduction

One of the most well-known online scheduling problems is partitioning jobs indexed $j \in [n] := \{1, \dots, n\}$ with processing times p_j that arrive online onto two machines J_1, J_2 so that the *makespan*, i.e., the maximum load $\max_{i \in \{1,2\}} \sum_{j \in J_i} p_j$ is minimized. For this problem, it is known that assigning each job to the least loaded machine upon its arrival is a $3/2$ -competitive algorithm, and this ratio cannot be beaten by any other deterministic algorithm [3].

We extend this problem by two *scenarios*, i.e., subsets $S_1, S_2 \subseteq [n]$ of jobs, and the goal is to minimize the makespan restricted to any scenario. To be more precise, we are incrementally informed of p_1, \dots, p_j as well as $S_k \cap [j]$ for $k \in \{1, 2\}$, one index j at a time, and the task is to find a partitioning $[n] = J_1 \dot{\cup} J_2$ of the jobs that minimizes the makespan $\max_{i \in \{1,2\}} \max_{k \in \{1,2\}} p(J_i \cap S_k)$, where we adopt the set notation $p(S) := \sum_{s \in S} p_s$ for $S \subseteq [n]$.

Although this problem has only been introduced recently [2] and has not been studied widely since then, some related problems are already known. For example, this problem can be seen as a special case of online two dimensional load balancing [1] where the two dimensional loads (x, y) are assumed to fulfill $x = 0$ or $y = 0$ or $x = y$. For vector balancing, it is proven that the greedy algorithm is $8/3$ competitive. However, this does not have any implications on the special case with two machines, because in this setting, any algorithm is trivially 2-competitive. Therefore, bounds for two dimensional vector scheduling on two machines remain wide open, which motivates considering special cases.

2 The Algorithm

Our main result is a $5/3$ -competitive algorithm (Algorithm 1). Our general idea is to fix a simple rule for assigning *double-scenario jobs*, i.e., jobs in $S_1 \cap S_2$, and partition the *single-scenario jobs* $j \in S_1 \triangle S_2$ so that the machines stay reasonably balanced even after upcoming hypothetical double-scenario jobs.

Rule 2.1. *If $j \in S_1 \cap S_2$, then assign the job j to a machine i such that the makespan of the schedule was previously attained by the other machine $3 - i$. In*

*ergen@math.tu-berlin.de. Technical University of Berlin, Germany

case of a tie, select the machine that received the job $j - 1$ ($j = 1$ is assigned to the first machine if applicable).

Throughout our analysis, we only use two types of lower bounds for the offline optimum: Processing times p_j of jobs j , and the average loads of scenarios S_k . Hence, while searching for the worst-case sequence of the upcoming double-scenario jobs, we may replace the actual definition of competitiveness with a proxy one:

Definition 2.2. We define the proxy competitive ratio of a schedule as

$$\rho = \frac{\max_{i,k \in \{1,2\}} p(S_k \cap J_i)}{\max \left\{ \max_{k \in \{1,2\}} \left\{ \frac{p(S_k)}{2} \right\}, \max_{j \in [n]} \arg \max_{j \in [n]} \{C_j\} \{p_j\} \right\}},$$

where C_j denotes the completion time of the job j .

Definition 2.3. Consider a schedule where, without loss of generality, the makespan is attained at $S_1 \cap J_1$. The anticipation α of this schedule is given by the ratio

$$\alpha := \begin{cases} \frac{p(J_1 \cap S_1)}{\max\{p(J_1 \cap S_2), p(J_2 \cap S_1) + p(J_1 \cap S_1) - p(J_2 \cap S_2)\}} & \text{if } p(J_2 \cap S_2) > p(J_2 \cap S_1), \\ 0 & \text{else.} \end{cases}$$

By convention, we assume $\frac{0}{0} = 1$.

In general, anticipation measures the minimum discrepancy of a machine with respect to different scenarios after machine 2 is loaded with double-scenario jobs just enough so that both machines $i \in \{1, 2\}$ have the same maximal load $\max_{k \in \{1,2\}} p(J_i \cap S_k)$. In our algorithm, one of the goals is to keep $\alpha \leq 2$ to prepare for upcoming double-scenario jobs.

Invariant 2.4. The schedule satisfies the following assertions:

- (i) Its proxy competitive ratio is bounded by $\rho \leq 5/3$.
- (ii) The schedule has anticipation bounded by $\alpha \leq 2$.
- (iii) Let k, i be given so that $J_i \cap S_k$ admits the makespan in the schedule. If the schedule is dominated by the i -th machine, i.e., $p(J_i \cap S_{k'}) > p(J_{3-i} \cap S_{k''})$ for all $k', k'' \in \{1, 2\}$, then we have $p(J_i \cap S_k) \leq 2p(J_i \cap S_{3-k})$.

Algorithm 1 outputs the index $i \in \{1, 2\}$ with $j \in J_i$, given a schedule of the first $j - 1$ jobs where Invariant 2.4 holds. The correctness of the algorithm follows by showing that if Line 9 is executed, the invariant is still maintained.

In spite of the restrictions that arise from both the fixed double-scenario rule and how we evaluate lower bounds, we find out that our algorithm in the previous subsection is not far from being best possible.

Theorem 2.5. There is no deterministic algorithm with a competitive ratio strictly smaller than $\frac{9+\sqrt{17}}{8} \approx 1.640$.

It can also be shown through a simple array of instances that, no algorithm that satisfies Rule 2.1 can beat the bound of $5/3$.

Algorithm 1 The assignment of a single job j given a $\frac{5}{3}$ -competitive schedule that satisfies Invariant 2.4.

```

1: Rename machines and scenarios so that  $(J_1 \cap S_1)$  is maximal
2: if  $j \in S_1 \setminus S_2$  then
3:   return 2
4: end if
5: if  $j \in S_2 \setminus S_1$  then
6:   if assigning  $j$  to the first machine satisfies Invariant 2.4 then
7:     return 1
8:   else
9:     return 2
10:  end if
11: end if
12: if  $j \in S_1 \cap S_2$  then
13:   return  $\operatorname{argmin}_{i=1,2} \max_{k=1,2} \{p(J_i \cap S_k)\}$ 
14: end if

```

3 Outlook

In light of our results, we may raise several open questions. For example, how do our algorithmic techniques extend to makespan scheduling with an arbitrary number of machines? Do they yield bounds that improve on the vector scheduling results? Another interesting direction seems to be applying the ideas that we have gathered to the more general online vector scheduling problem, for which no nontrivial algorithm on two machines seems to be known.

References

- [1] Ilan Cohen, Sungjin Im, and Debmalya Panigrahi. Online Two-Dimensional Load Balancing. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 34:1–34:21, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- [2] Ekin Ergen. Online Makespan Scheduling Under Scenarios. In *33rd Annual European Symposium on Algorithms (ESA 2025)*, volume 351 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 27:1–27:16, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- [3] R. L. Graham. Bounds for certain multiprocessing anomalies. *The Bell System Technical Journal*, 45(9):1563–1581, 1966. doi:10.1002/j.1538-7305.1966.tb01709.x.

Scheduling Jobs to Maximize Fractional Value

Kunal Agrawal* Sanjoy Baruah* Gregory Kehne* Jubayer Nirjhor†
 Kei Rockwell (Speaker)† Nicole Wein†

The scheduling problem considered here arose in the context of the NASA-funded ADAPT [1] project at Washington University, that addresses the timely observation of transient astrophysical phenomena such as gamma-ray bursts (GRBs) from outer space. The computational pipeline for GRB localization must execute entirely onboard a spacecraft upon a computing platform of very limited computing capacity; it is generally not possible to guarantee to always fully execute the entire workload. The problem of scheduling the GRB localization computational pipeline on board the spacecraft has been formalized as follows.

Definition 1 (Scheduling to maximize fractional value). *A job is characterized by a 3-tuple (r_i, w_i, d_i) , denoting the release time, the computational requirement, and deadline respectively. In some schedule for a particular collection of jobs, let $c_i \leq w_i$ denote the amount of computation received by job i by its deadline. We say the fractional value of job i in this schedule is c_i/w_i , and the fractional value of the schedule is the minimum fractional value of any individual job in the collection being scheduled. The scheduling objective is to maximize the fractional value of the schedule.*

Due to the dynamic, variable, and uncertain nature of GRBs, the collection of jobs comprising the computational workload on board the spacecraft is not a priori known but tends to arrive online. We seek to develop uniprocessor online algorithms for solving the problem of scheduling to maximize fractional value. We define the *competitive ratio* of such an online algorithm to be the minimum ratio, over all collections of jobs, of the fractional value of the schedule generated by the algorithm on that collection of jobs to the fractional value of the schedule generated by an optimal clairvoyant algorithm on the same collection of jobs. We show both an algorithm and an impossibility result:

Theorem 2. *No online algorithm has a competitive ratio larger than $\left(\frac{\sqrt{5}-1}{2}\right) \approx 0.618$ (i.e., the inverse of the golden ratio).*

Theorem 3. *There exists an algorithm with competitive ratio $\frac{1}{2}$.*

Our algorithm is a variant of the classical *earliest deadline first* strategy.

Earliest Deadline First (EDF): For the fundamental scheduling problem of jobs with release times and deadlines on a single preemptive processor, it is well-known that earliest deadline first (EDF) is an optimal strategy. Therefore, let's consider whether

*{kunal, baruah, kehne}@wustl.edu. Washington University in St. Louis

†{nirjhor, krockw, nswein}@umich.edu. University of Michigan

(some variant of) this strategy also works for the problem of maximizing fractional value. It is easy to see that the standard EDF algorithm has no bounded competitive ratio since, for many instances, it might complete a large fraction of some jobs at the cost of not even starting other jobs. Therefore, we propose a *fractional EDF* scheduler. In order to define this scheduler, we first define the following term.

Definition 4 (Optimal fractional value $v^o(t)$ at time t). *At any time t , we say that the optimal fractional value $v^o(t)$ of an instance is the fractional value that an optimal scheduler can provide to this instance for all the jobs that have arrived by time t .*

Note that this is simply the solution of the offline problem of maximizing fractional value of a given set of jobs (those that have arrived by time t) and can be solved easily in polynomial time as follows: For each intervals $I = [t_s, t_f]$ where both t_s, t_f belong to the set of release time/deadlines of jobs, we can calculate the workload of interval I and divide it by the length of I to calculate the maximum fractional value achievable within that interval. Taking the minimum over all such intervals gives us v^o .

Now we can define the fractional EDF scheduler.

Definition 5 (c -fractional EDF). *At any time t and for any fraction $c \leq 1$, c -fractional EDF tries to complete $c \times v^o(t)$ fraction of each job's work in an EDF manner. In particular, when a job is released, the scheduler updates $v^o(t)$, which is its (current) estimate of the optimal schedule's fractional value. For every job i , it calculates $\hat{w}_i = c \times v^o(t) \times w_i$ — this is its work-obligation for job i . It only considers jobs which have not yet completed \hat{w}_i work so far, and executes the earliest deadline job from this set.*

Note that the operation of this scheduler strongly depends on c , the competitive ratio it is trying to achieve. It is easy to see that a c -fractional EDF scheduler can not achieve a competitive ratio larger than c . Therefore, our goal is to compute the largest value of c such that c -fractional EDF is guaranteed to be able to achieve a competitive ratio of c . We show that this optimal value is $1/2$ via matching lower and upper bounds.

The impossibility result shows that for any $c > 1/2$, there is a collection of jobs where fractional-EDF cannot complete a $c \times v^o$ fraction of every job, where v^o is the fractional value of the optimal schedule:

Theorem 6. *A fractional-EDF scheduler (for any c) cannot have a competitive ratio larger than $\frac{1}{2}$.*

Matching this bound, we show that competitive ratio $\frac{1}{2}$ is achievable:

Theorem 7. *$\frac{1}{2}$ -fractional EDF has a competitive ratio of $\frac{1}{2}$.*

References

- [1] Y Htet, M Sudvarg, J Buhler, R Chamberlain, and J Buckley. Localization of gamma-ray bursts in a balloon-borne telescope. In *Proceedings of the SC '23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*, SC-W '23, page 395–398, New York, NY, USA, 2023. ACM Press.

Tight Analysis of Proportional Fairness for Minimizing Weighted Flow Time in Monotone Polytope Scheduling

Sven Jäger* Alexander Lindermayr* Bart Zondervan (Speaker)†

1 Introduction

We study the *Polytope Scheduling Problem* (PSP), a unifying framework that captures a wide range of classical scheduling models, including unrelated machine scheduling, generalized broadcast scheduling, and multidimensional scheduling [3, 4]. An instance consists of a set J of n jobs. Each job $j \in J$ is specified by a nonnegative weight w_j , a processing requirement p_j , and a release time r_j . At every time t , a scheduler \mathcal{A} assigns rates $\{y_j^{\mathcal{A}}(t)\}_{j \in J}$ to the jobs currently present in the system. The resulting rate vector must lie in a fixed packing polytope

$$\mathcal{P} = \{y \in \mathbb{R}_{\geq 0}^n \mid By \leq \mathbf{1}\},$$

where $B = (b_{dj}) \in \mathbb{Q}_{\geq 0}^{D \times n}$ encodes the resource constraints and $\mathbf{1}$ denotes the all-ones vector. This formulation allows fractional, time-varying allocations subject to linear capacity constraints and provides a convenient abstraction for diverse scheduling environments.

The completion time $C_j^{\mathcal{A}}$ of job j is the earliest time t' such that $\int_{r_j}^{t'} y_j^{\mathcal{A}}(t) dt \geq p_j$ and the corresponding flow time is $F_j^{\mathcal{A}} = C_j^{\mathcal{A}} - r_j$. Our objective is to minimize the sum of weighted flow times $\sum_{j \in J} w_j F_j^{\mathcal{A}}$, which is called the weighted flow time. We consider the online, non-clairvoyant setting: when a job j is released at time r_j , the scheduler learns its weight w_j and the associated column b_j of B , but not its processing requirement p_j , this is revealed only upon completion of j . Throughout, we allow preemptive schedules, i.e., jobs may be interrupted and resumed at a later time.

We adopt the *Proportional Fairness* (PF) algorithm as the rate-allocation rule. The PF rates $y^{\text{PF}}(A_t)$ are obtained as an optimal solution to the Eisenberg–Gale convex program [2] over the set of active jobs $A_t = \{j \in J \mid r_j \leq t < C_j^{\text{PF}}\}$ at

*{jaeger,lindermayr}@math.tu-berlin.de. Institut für Mathematik, Technische Universität Berlin, Germany.

†bart.zondervan@uni-bremen.de. Faculty of Mathematics and Computer Science, University of Bremen, Germany.

time t ,

$$\begin{aligned} & \text{maximize} && \sum_{j \in A_t} w_j \log(y_j) \\ & \text{subject to} && \sum_{j \in A_t} b_{dj} y_j \leq 1 && \forall d \in [D], \\ & && y_j \geq 0 && \forall j \in A_t. \end{aligned}$$

Proportional Fairness generalizes the classical Round Robin algorithm.

Im, Kulkarni and Munagala [3] showed that for weighted flow time, there exist PSP instances for which no deterministic non-clairvoyant algorithm is constant competitive, even in the speed augmentation model where an algorithm processes jobs at a factor of $o(\sqrt{\log n})$ faster compared to the optimal solution. Therefore, we restrict our attention to *PF-monotone* PSP instances. Formally, a PSP instance is PF-monotone if for any two job sets $J' \subseteq J$ and any job $j \in J'$ it holds that $y_j^{\text{PF}}(J') \geq y_j^{\text{PF}}(J)$. Intuitively, monotonicity means that removing jobs from the system cannot decrease PF's rate of any remaining job. This property holds for several important scheduling models, most notably related machine scheduling, scheduling with restricted assignment, or matroid scheduling. Im et al. [3] used a potential function and amortized local competitiveness to show that PF is $O(1/\varepsilon^2)$ -competitive for PF-monotone PSP with a speed augmentation of $e + \varepsilon \approx 2.718 + \varepsilon$. While PF is known to be $O(1/\varepsilon)$ -competitive given $2 + \varepsilon$ speed augmentation on identical parallel machines [1], nothing better was known for any of the special cases mentioned above. Edmonds [1] proved that even on a single machine when given speed augmentation at most $2 + \varepsilon$, no better competitive ratio is possible.

2 Our contribution

In this work, we improve both the competitive ratio and the speed augmentation factor for PF-monotone PSP.

Theorem 1. *For any $\varepsilon > 0$, Proportional Fairness is $O(1/\varepsilon)$ -competitive for minimizing weighted flow time for the PF-monotone Polytope Schedule Problem with a speed augmentation of $2 + \varepsilon$.*

This closes the gap to the above-mentioned information-theoretic lower bound for the general PF-monotone PSP. Our analysis uses dual fitting based on the fractional weighted flow time, which serves as a natural lower bound on the weighted flow time. We compare the PF schedule against an optimal offline schedule through

the dual of the fractional flow LP. This dual can be written as follows.

$$\begin{aligned}
\max \quad & \sum_{j \in J} \alpha_j - \sum_{d=1}^D \sum_{t \geq 0} \beta_{dt} \\
\text{s.t.} \quad & \alpha_j - w_j(t - r_j) \leq p_j \sum_{d=1}^D b_{dj} \beta_{dt} \quad \forall j \in J, t \geq r_j, \\
& \alpha_j, \beta_{dt} \geq 0 \quad \forall d \in [D], j \in J, t \geq 0.
\end{aligned}$$

We consider the following dual solution. We set $\bar{\alpha}_j := w_j F_j^{\text{PF}}(J^{\leq j})$, where $J^{\leq j}$ are the jobs that are released no later than job j , and $F_j^{\text{PF}}(J^{\leq j})$ is the flow time of job j in the PF schedule of $J^{\leq j}$. The dual variables $\bar{\beta}_{dt}$ are defined to be proportional to the time-dependent Lagrange multipliers $\eta_d(t)$ arising from the KKT conditions of the Eisenberg–Gale convex program. With this construction, the main hurdle in the dual fitting argument becomes to establish the inequality

$$\sum_{j \in J} w_j F_j^{\text{PF}}(J) \leq 2 \sum_{j \in J} w_j F_j^{\text{PF}}(J^{\leq j}).$$

We prove this bound by a careful incremental analysis of the PF objective: we quantify how the weighted flow time changes when a single job is appended to the system, and show that this increase is bounded by at most twice the weighted flow time contribution of that job in its own prefix schedule. This completes the comparison between the PF schedule and the optimal solution and yields the desired scalability guarantee.

References

- [1] JEFF EDMONDS (2000). *Scheduling in the dark*. Theoretical Computer Science, 235(1), 109-141.
- [2] EDMUND EISENBERG AND DAVID GALE (1959). *Consensus of subjective probabilities: The pari-mutuel method*. The Annals of Mathematical Statistics, 30(1), 165-168.
- [3] SUNGJIN IM, JANARDHAN KULKARNI AND KAMESH MUNAGALA (2018). *Competitive Algorithms from Competitive Equilibria: Non-Clairvoyant Scheduling under Polyhedral Constraints*. Journal of the ACM (JACM), 65(1), 1-33.
- [4] SVEN JÄGER, ALEXANDER LINDERMAYR AND NICOLE MEGOW (2025). *The Power of Proportional Fairness for Non-Clairvoyant Scheduling under Polyhedral Constraints*. Proceedings of the 2025 Annual ACM-SIAM, SODA 2025, New Orleans, LA, USA, January 12-15, 2025.

Fair Incomplete Round-Robin Tournaments

Frits Spieksma *

Sten Wessel (Speaker) *

1 Introduction

Round-robin tournaments are a popular way of organizing competitions and hence they have been intensively studied before, see, e.g., [3, 1]. We are focussing on single round-robin tournaments, where every pair of teams (or players) meet each other exactly once. However, often it is not possible that teams play all other teams, as this would require too many matches to be played. A variant of round-robin tournaments has been introduced that can remedy this, the incomplete round robin (iRR), where it is still the case that every team plays the same number of matches, however, not against *all* other teams. If we let n denote the number of teams, and k the number of matches played by each team, with $k < n - 1$, it follows that there are in total $nk/2$ matches. We call the k opponents that a team faces the *opponent set* of a team. Notice that it is not the case that each team faces the same set of opponents.

Incomplete round-robin competitions are used increasingly more often. There is the much studied case of the League phase in the UEFA Champions League (football, see <https://www.uefa.com/uefachampionsleague/>). The iRR was introduced in the season 2024–2025, with $n = 36$ teams and $k = 8$ matches played by each team. Another example is the Volleyball Nations League (see <https://en.volleyballworld.com/volleyball/competitions/volleyball-nations-league>). Since season 2024–2025, the format is an iRR with $n = 18$, $k = 12$. Devriesere et al. [2] describe work for the Belgian Youth Hockey league.

A clear issue that needs to be investigated when organizing an iRR is the following. When not all teams face the same set of opponents, and assuming there is variance in the strength of the teams, the question arises: is the iRR fair? Indeed, when not all teams have the same strength, is the final ranking a true representation of the strength of the teams? This work investigates this question.

We assume there is a given strength of each team, e.g., an Elo rating. In order to compare the strength of various opponent sets, we use the average strength of the teams in the opponent set. We consider the following objective to express the fairness of a draw: the difference between the largest average strength of an

*{f.c.r.spieksma,s.wessel}@tue.nl. Department of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands.

opponent set, and the smallest average strength of an opponent set. We call this the *bandwidth*. To maximize fairness, we aim to minimize the bandwidth. Of course, it is true that opponent sets with (almost) the same average strength might still be perceived differently: one opponent set may consist of teams with almost equal strength, while another opponent set with the same average strength may consist of teams whose strength vary quite a bit. However, in this work we focus on minimizing the bandwidth as our primary fairness measure.

Our contributions are as follows:

- We provide an integer program minimizing the bandwidth.
- We provide a way to find a collection of opponent sets with maximum bandwidth.
- We computationally analyze both real-world and generated instances where we minimize and maximize the bandwidth.
- We theoretically analyze the computational complexity of the problem of finding a fair iRR tournament where the bandwidth is zero, i.e., where every opponent set is of equal strength.

In the remainder, we highlight our results on the latter case.

2 Fair-iRR with Zero Bandwidth

Given are n teams that each play k matches. We assume that each team has a strength, and that the strength of an opponent set equals the average strength of the teams in the opponent set. A *draw* is an assignment from every team to an opponent set of size k , such that if team j is in the opponent set of team i , then team i is in the opponent set of team j . The decision question, which we call *Fair-iRR*, now becomes: given n teams with their strengths s_i , $i \in \{1, \dots, n\}$, does there exist a draw such that the strengths of all opponent sets are equal?

Motivated by practice, we consider three properties that jointly determine a class of iRRs: (i) the number of pots p , (ii) the number of matches per team k , and (iii) the number of countries c . Ad (i), pots are often used in practice to establish a rough partition of the teams based on strength. We will assume that n is a multiple of p . Ad (ii), we will assume that k is a multiple of p , and that each team plays k/p matches against teams from each pot. Ad (iii), teams from the same country (or national league) are not allowed to play against each other. Notice that if $c = n$ (where n is the number of teams), all matches are allowed. We highlight our results for the classes of instances with $k = 2$ and $k = 3$.

In the case of two opponents, i.e., $k = 2$, the opponent sets are the pairs $\{i, j\}$ such that the sum of strengths $s_i + s_j = \frac{2 \sum s_i}{n}$. Let $\bar{s} = \sum s_i / n$ denote the average strength. Consider now the *opponent graph* $G = (V, E)$ where the vertices $V = [n]$ are the teams and the edges $E = \{\{i, j\} : \frac{1}{2}(s_i + s_j) = \bar{s}\}$ are the opponent sets.

The edges E can be partitioned in the sets $E_d := \{\{i, j\} : s_i = \bar{s} - d, s_j = \bar{s} + d\}$, for all $d \geq 0$.

We first observe that if a Fair-iRR draw exists, then the opponent sets used in the draw form a perfect 2-matching in G . Furthermore, we show that the opponent graph must have the following structure: a disjoint union of a clique and complete bipartite graphs, as formalized in the following lemma.

Lemma 1 *If a schedule exists, then the opponent graph G is isomorphic to the graph $K_{k_0} + \sum_{d \in D} K_{k_d, k_d}$ where $D = \{d \in \mathbb{R}_{>0} : E_d \neq \emptyset\}$, $k_0 \geq 0$, $k_0 \neq 1$, $k_d \geq 1$ for all $d \in D$.*

From this, we characterize the existence of a solution for Fair-iRR when $k = 2$ in the following theorem. Note that it can be checked in polynomial time whether this condition holds for any given instance.

Theorem 2 *There exists a Fair-iRR draw precisely when the opponent graph is of the structure in Lemma 1 and:*

- $k_0 = 0$ and $|\{k_d : d \in D, k_d \text{ is odd}\}|$ is even, or
- $k_0 = 2, 3$ and $|\{k_d : d \in D, k_d \text{ is odd}\}|$ is odd, or
- $k_0 = 4$ and $|\{k_d : d \in D, k_d \text{ is odd}\}|$ is even, or
- $k_0 \geq 5$.

In the case of three opponents, i.e., $k = 3$, we show that the Fair-iRR problem is hard for the following classes.

Theorem 3 *Fair-iRR is NP-complete, even if $p = k = 3$ and $2 \leq c \leq n$.*

Theorem 4 *Fair-iRR is NP-complete, even if $p = 1$, $k = 3$, and $c = n$.*

We prove both theorems using a reduction from the Numerical 3-Dimensional Matching problem.

References

- [1] K. DEVRIESERE, L. CSATÓ, AND D. GOOSSENS (2025). *Tournament design: A review from an operational research perspective*. European Journal of Operational Research 324, 1–21.
- [2] K. DEVRIESERE AND D. GOOSSENS (2025). *Redesigning Belgian Youth Field Hockey Competitions Using an Incomplete Round-Robin Tournament*. INFORMS Journal on Applied Analytics 55, 457–468.
- [3] R. RASMUSSEN AND M. TRICK (2008). *Round robin scheduling: A survey*. European Journal of Operational Research 188, 617–636.

The Price of Diversity of the Traveling Salesman Problem

Frits Spieksma (Speaker) * Mark de Berg †

Andrés López Martínez ‡

1 Introduction

We explore how imposing *diversity* among solutions in a discrete optimization problem affects their individual costs. Simply put, diversity comes at a price. In this text, we aim to provide a formal definition of this phenomenon and introduce the *price of diversity* (PoD) as a measure to quantify this trade-off.

There are many scenarios where generating diverse yet high-quality solutions is desirable. Consider a decision maker who lacks complete information about all aspects influencing the cost or feasibility of solutions to a given problem and needs flexibility to adapt to different cost scenarios. By maintaining a diverse set of solutions, a decision maker can defer selection until the missing cost details become available, thereby minimizing the risk of committing to a suboptimal choice. Similarly, in time-sensitive settings where last-minute details can emerge, having a diverse set of solutions increases the chance that at least one feasible option is readily available.

Typically, requiring a diverse set of solutions comes at a price. In this paper, we study this phenomenon for the Traveling Salesman Problem (TSP): given an edge-weighted graph, find a shortest tour visiting all its vertices. Suppose that instead of computing a single optimal tour, we must produce k tours subject to a diversity requirement, while minimizing their bottleneck cost, i.e., the length of a longest tour. To measure the trade-off between solution quality and the desired number of solutions, we analyze the ratio between the length of a longest tour in the best set of k diverse tours and the length of an optimal single tour. Taking the worst-case value of this ratio over all instances provides a measure of how much the demand for diversity impacts the cost of solutions. We call this measure the Price of Diversity (PoD), see Section 2 for a precise definition.

*f.c.r.spieksma@tue.nl. Department of Mathematics and Computer Science, Eindhoven University of Technology, the Netherlands.

†Department of Mathematics and Computer Science, Eindhoven University of Technology, the Netherlands.

‡Department of Mathematics and Computer Science, Eindhoven University of Technology, the Netherlands.

Related Work. The generation of multiple solutions has been studied in many areas: constraint programming, mixed integer optimization, genetic algorithms, social choice, and computational economics. For references, we refer to the extended paper corresponding to this abstract, see de Berg et al. [1]. For problems where the output is a set of edges, such as matching, spanning tree, or minimum s-t cut, edge-disjointness serves as a natural diversity criterion (see Lopez Martínez [4], Fomin et al. [2]). In the context of the TSP, a related problem is the peripatetic k-TSP introduced by Krarup [3], where the objective is to minimize the sum of the lengths of k edge-disjoint tours.

We now provide a formal statement of the two problems we investigate.

Disjoint-Path TSP₂. *Given a weighted graph $G = (V, E)$ and two specified vertices $s, t \in V$, find a pair (H_1, H_2) of edge-disjoint Hamiltonian (s, t) -paths that minimizes $\text{cost}(H_1, H_2) := \max(c(H_1), c(H_2))$.*

Disjoint-TSP₂. *Given a weighted graph $G = (V, E)$, find a pair (T_1, T_2) of edge-disjoint tours (that is, Hamiltonian cycles) that minimizes $\text{cost}(T_1, T_2) := \max(c(T_1), c(T_2))$.*

2 The Price of Diversity

The PoD is formalized as follows. Let Π be a minimization problem (like the TSP) where each instance I has an associated set of feasible solutions $\Gamma(I)$. Each solution $s \in \Gamma(I)$ has a cost $c(s)$, and the objective is to find an optimal solution; that is, a feasible solution with minimum cost, $\text{OPT}(I)$. Consider now the following diverse variant of Π , called simply Π_k , which, given an instance I of problem Π and a fixed integer $k > 0$, is tasked with finding a minimum-cost set of k feasible solutions to I that satisfies a specified diversity requirement. More formally, let $\mathcal{S}(I) \subseteq \Gamma(I)^k$ represent the collection of all k -sized sets of feasible solutions for instance I that satisfy the diversity requirement, and let $\text{cost}(S)$ denote the cost of a set $S \subseteq \Gamma(I)$ of feasible solutions. Further, let $\mathcal{I}(\Pi)$ denote the set of all instances of problem Π . We define the Price of Diversity of problem Π_k , abbreviated with $\text{PoD}(\Pi_k)$, as

$$\text{PoD}(\Pi_k) = \sup_{I \in \mathcal{I}(\Pi)} \left\{ \frac{\min_{S \in \mathcal{S}(I)} \text{cost}(S)}{\text{OPT}(I)} \right\}. \quad (1)$$

In other words, $\text{PoD}(\Pi_k)$ represents the supremum, taken over all instances of problem Π , on the ratio between the minimum cost among diverse sets of feasible solutions and the cost of a single optimal solution.

Note that we have assumed that Π is a minimization problem, but this is without loss of generality as a similar measure to (1) can be defined for maximization problems in a straightforward manner. We remark that also the term ‘‘Price of Diversity’’ has appeared before, especially in computational social choice, but usually refers to the cost of enforcing diversity within a single solution; in contrast, we analyze the PoD in the setting where a collection of diverse solutions is required. To our knowledge, this is the first formal treatment of this perspective.

3 Results

We show the following.

Theorem 1 $PoD(\text{Disjoint} - \text{PathTSP}_2) = 3$.

Theorem 2 $PoD(\text{Disjoint} - \text{TSP}_2) = 2$.

We also establish the PoD for both problems when the instance is restricted to special classes of graphs.

References

- [1] DE BERG, M., A. LÓPEZ MARTÍNEZ, F.C.R. SPIEKSMÁ (2026). *Disjoint Tours and the Price of Diversity*. In the Proceedings of the 20th International Conference and Workshops on Algorithms and Computation (WALCOM 2026).
- [2] FOMIN, F.V., GOLOVACH, P.A., JAFFKE, L., PHILIP, G., SAGUNOV, D. (2024). *Diverse Pairs of Matchings*. *Algorithmica* **86**, 2026–2040.
- [3] KRARUP, J (1974). *The peripatetic salesman and some related unsolved problems*. In: *Combinatorial Programming: Methods and Applications: Proceedings of the NATO Advanced Study Institute*, pp. 173–178.
- [4] LÓPEZ MARTÍNEZ (2025). *Diversity of Solutions in Combinatorial Optimization*. PhD thesis Eindhoven University of Technology.

Online Firefighting on Cactus Graphs

Max Hugén* Bob Krekelberg (Speaker)† Alison Hsiang-Hsuan Liu‡

The *firefighting game* is a fundamental problem in theoretical computer science, modeling the containment of a spreading process under limited defensive resources. The firefighting game is known to be a computationally hard problem. Even for restricted graph classes such as bipartite graphs [5] and even for trees of maximum degree 3 [3], the problem is shown to be NP-hard.

On the positive side, several approximation algorithms are known for the firefighting game. For example, the greedy algorithm is shown to be a 2-approximation on trees [4], and also, there also exists a polynomial time approximation scheme for the firefighting game on trees [1].

While the offline version has been extensively studied, much less is known about the *competitive complexity* of the online variant, where the underlying graph is known in advance but the number of available firefighters in each round is revealed online. In this work, we study how *graph structure* governs the power of the adversary in the online firefighting game.

Problem definition. Formally, in the *online firefighting game*, the instance is $\mathcal{I} = (G, r, (f_i)_{i \geq 1})$, where $G = (V, E)$ is a graph with the *fire source* or *root* r , and $f_i \geq 0$ is the number of available firefighters in round i . The game proceeds in rounds. In each round i , the online algorithm learns the value of f_i and *protects* unburned vertices by placing the firefighters at them. Then, the fire spreads to unprotected neighbor vertices of all burning vertices, and it ends this round.

On trees, the problem is known to be 2-competitive [2], a result that relies on a strong structural alignment between the online algorithm and the optimal solution throughout the process. We show that this alignment breaks down as soon as cycles are present. In particular, the algorithm and the optimal solution may break a cycle differently, or even break different cycles, and therefore operate on fundamentally different residual graphs. As a result, the adversary gains additional leverage by steering the process along residual graph structures that no longer admit a direct comparison between the algorithm and the optimal solution.

Our results. Our main result shows that the presence of a single cycle already increases the competitive complexity of the firefighting game dramatically. We first show that even on a *tadpole graph* (a cycle with a tail), no deterministic online algorithm can be competitive.

*Department of Information and Computing Sciences, Utrecht University, The Netherlands

†b.h.a.f.krekelberg@uu.nl. Department of Information and Computing Sciences, Utrecht University, The Netherlands

‡h.h.liu@uu.nl. Department of Information and Computing Sciences, Utrecht University, The Netherlands

Theorem 1 *No deterministic online algorithm can achieve $o(\sqrt{n})$ -competitiveness for the online firefighting problem on tadpole graphs.*

We complement this lower bound with matching upper bounds by designing an $O(\sqrt{n})$ -competitive online algorithm for 1-almost trees, that is, graphs obtained from a tree by adding at most one edge.

Theorem 2 *There is a $(5\sqrt{n} + 2)$ -competitive algorithm for the online firefighting game on 1-almost-trees, which is asymptotically optimal.*

We further show that adding more cycles does not further increase the power of the adversary, as long as these cycles do not share edges. Specifically, we show extend our algorithmic framework to *cactus graphs*, where every edge is contained in at most one cycle, and prove that the competitive complexity remains $\Theta(\sqrt{n})$ for cactus graphs.

Theorem 3 *There is a $(20\sqrt{n} + 1)$ -competitive algorithm for the online firefighting game on cactus graphs, which is asymptotically optimal.*

These results together yield a tight characterization of the adversarial power induced by cycles with non-overlapping edges.

Finally, considering that cactus graphs have treewidth of 2, we study a variant in which firefighters are released in pairs, that is, an even number of firefighters becomes available in each round. Surprisingly, the competitive complexity is significantly reduced for this setting.

Theorem 4 *There is a 3-competitive algorithm for the online firefighting game on cactus graphs with even firefighters in each round.*

Main ideas of the algorithms. Our algorithms for 1-almost trees and for cactus graphs works iteratively in each round i with f_i firefighters. In each iteration, if there are enough firefighters to protect the heaviest component adjacent to the fire source, the algorithm makes the greedy choice. Moreover, if there are not enough available firefighters (in 1-almost tree or cactus graphs case, there is exactly one firefighter) to protect the heaviest component but the greedy choice guarantees a gain of at least square root of the size of the heaviest component (in this case it must be a cycle), the algorithm still follows the greedy choice.

The critical case occurs otherwise when only one firefighter is available, and the greedy choice does not guarantee a sufficiently large immediate gain. In this situation, we show that it is always possible that the algorithm can maximize the timing the fire reaches a large enough portion of vertices by protecting a vertex on a cycle and break the cycle.

For 1-almost trees, the decision of how to break a cycle is easier, as there is at most one cycle in the graph. However, in cactus graphs, there can be multiple cycles all containing the fire source at the same time. A difficulty arises because if the algorithm repeatedly breaks cycles due to the lack of immediate gain, it may fail to secure many vertices that could otherwise be saved. To address this, we introduce a *cool-down timer* mechanism: once a cycle is broken, the algorithm enters a cool-down period during which it must follow the greedy choice whenever new firefighters become available. The rule for selecting a vertex to break a cycle is thus adapted to this mechanism.

Main ideas of the analysis. The main technical challenge lies in the analysis, as the algorithm and the optimal solution may break different cycles and therefore operate on different residual graphs. As a result, unlike the tree case, it is no longer possible to rely on a direct correspondence between saved vertices in the two solutions. To address this, our algorithm makes sure that either it saves a large enough portion of the vertices that the optimal solution can save in the same round, or it delays the timing of the fire reaching a large portion of vertices by breaking a carefully selected cycle at a meticulously selected position. For the analysis, we conceptually match the vertices where the algorithm places firefighters with the vertices protected by the optimal solution at the corresponding rounds. We then partition the vertices protected by the optimal solution so that their total contribution can be charged to the vertices protected by the algorithm, with each algorithmic vertex being charged only a constant number of times.

References

- [1] D. Adjiashvili, A. Baggio, and R. Zenklusen. Firefighting on trees beyond integrality gaps. *ACM Trans. Algorithms*, 15(2):20:1–20:33, 2019.
- [2] P. Coupechoux, M. Demange, D. Ellison, and B. Jouve. Firefighting on trees. *Theor. Comput. Sci.*, 794:69–84, 2019.
- [3] S. Finbow, A. D. King, G. MacGillivray, and R. Rizzi. The firefighter problem for graphs of maximum degree three. *Discret. Math.*, 307(16):2094–2105, 2007.
- [4] B. Hartnell. Firefighting on trees: how bad is the greedy algorithm? *Congressus Numerantium*, 145:187–192, 2000.
- [5] G. MacGillivray and P. Wang. On the firefighter problem. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 47:83–96, 2003.

Scheduling Out-Trees Online to Optimize Maximum Flow

Kunal Agrawal ^{*} Benjamin Moseley [†] Heather Newman [‡]
Kirk Pruhs (Speaker) [§]

We consider scheduling online a collection of multithreaded programs/jobs, where each job is modeled by a digraph in which the directed edges specify precedence constraints between unit work nodes/subjobs, and where the objective is to minimize the maximum flow time.

The task of the online scheduler is akin to the task faced by a Tetris player in that geometric forms have to be packed as they arrive online without knowledge of the forms that will arrive in the future. However, the task faced by the online scheduler is even more daunting. The rules for the ways that a geometric form can be feasibly packed in a schedule are much more complicated than the rules of packing in Tetris. The parallelizability of the programs at a particular time is the number of ready subjobs: therefore, this parallelizability depends significantly on which subjobs the scheduler processed in the past. Therefore, the scheduler's past decisions change the "future" shape of the piece.

Roughly speaking, if the parallelizability of jobs is unvarying over the span of the job (which roughly in Tetris terms means that the jobs are rectangular shaped), then it is known that FIFO is the "right" scheduling policy for the maximum flow objective, where "right" is interpreted to mean $O(1)$ -competitive. The genesis of this research is a conjecture in [1] that FIFO is still the "right" algorithm if the parallelizability of a job can vary over the span of the job (so in Tetris terms, even if the jobs are irregularly shaped).

We obtained the following results in [2].

Lower Bound for FIFO: Our first main contribution that in fact FIFO is not the "right" algorithm, even if the digraphs/jobs are out-trees. In particular, we show that for out-trees the competitive ratio of FIFO with respect to maximum flow is $\Omega(\log m)$, where m is the number of processors. Intuitively the mistake that FIFO makes is to not take into account nodes' abilities to spawn parallelizable work.

^{*}Washington University in St. Louis

[†]Carnegie Mellon University

[‡]Vassar College

[§]University of Pittsburgh

New Algorithm for Out-Trees: Our second main contribution is a clairvoyant algorithm \mathcal{A} out-trees that is $O(1)$ -competitive for maximum flow. In this setting clairvoyant means that the scheduling algorithm learns the shape of a tree when it is released. This scheduling algorithm is based on the idea of controlling the shape of our tetris pieces in order to create pieces that can be fit together well. The first step in the design of our algorithm is to show that a simple greedy algorithm we call Longest Path First (LPF), which always prioritizes the subjobs/nodes with the highest height in the out-tree, is optimal for a single job. The next step is to observe that the LPF schedule $\text{LPF}[m/\alpha]$ on m/α processors (where α is a constant that we will eventually pick in the analysis), has a nice shape. We do not have control over the shape of $\text{LPF}[m/\alpha]$ in the first OPT time units, which we call the head of the schedule, but the shape of the portion of $\text{LPF}[m/\alpha]$ after time OPT , which we call the tail of the schedule, is essentially a rectangle with width m/α processors and length at most $(\alpha - 1)\text{OPT}$ units of time. See Figure 1. Our algorithm \mathcal{A} schedules the head of each job using m/α processors when the job arrives, and uses FIFO to prioritize jobs when scheduling the tails of the jobs, and uses the shape of the LPF schedule of each job for intra-job scheduling. Leveraging that the tails have a rectangular shape, we are able to adapt the type of analysis technique that is used for jobs with unvarying parallelizability to show that \mathcal{A} is $O(1)$ -competitive.

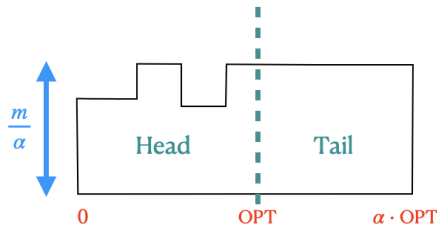


Figure 1: Generic LPF Schedule.

My main motivation for giving this talk is to highlight some of the many natural open questions in this area that are worthy of being addressed, including:

- Is FIFO $O(\log m)$ -competitive general digraphs, or maybe on series-parallel digraphs?
- Is FIFO asymptotically optimally competitive among nonclairvoyant algorithms? A nonclairvoyant run-time scheduler only learns of a subjob j in a DAG G_i when all of the predecessors of j are completed. If so, this could potentially salvage the intuition of the authors of [1] that FIFO being is the “right” algorithm.
- Is there an $O(1)$ -competitive nonclairvoyant algorithm for out-trees? It is not clear if or how a nonclairvoyant algorithm can “shape” an out-tree as our clairvoyant algorithm did. We conjecture that the worst-case instances are

those with the property that for every out-tree T and for every node $v \in T$ it is the case that at most one of v 's children in T is not a leaf. My current view is this might be the most interesting open problem on this list.

- Is there an $O(1)$ -competitive clairvoyant algorithm for series-parallel DAGs ? Series-parallel digraphs are probably the best abstract model for fork-join parallelism.
- Is there an $O(1)$ -competitive nonclairvoyant algorithm for series-parallel DAGs?
- Is there an $O(1)$ -competitive clairvoyant algorithm for general DAGS?
- Is there an $O(1)$ -competitive nonclairvoyant algorithm for general DAGS?
- Similar open problems exist for the other standard models of parallelizability, the speed-up curves model. For example, it is open whether there is a clairvoyant algorithm that is $O(1)$ -competitive for maximum flow. There is an $O(1)$ -competitive algorithm if the speed-up curve for each job does not vary over the span of that job [3].

References

- [1] Kunal Agrawal, Jing Li, Kefu Lu, and Benjamin Moseley. Scheduling parallelizable jobs online to minimize the maximum flow time. *ACM Symposium on Parallelism in Algorithms and Architectures*, pages 195–205, 2016.
- [2] Kunal Agrawal, Benjamin Moseley, Heather Newman, and Kirk Pruhs. Scheduling out-trees online to optimize maximum flow. *ACM Symposium on Parallelism in Algorithms and Architectures*, pages 77–88, 2024.
- [3] Benjamin Moseley, Ruilong Zhang, and Shanjiawen Zhao. Online scheduling of parallelizable jobs in the directed acyclic graphs and speed-up curves models *Theoretical Computer Science*, volume 938(C), pages 24–38.

Improved Approximation Algorithms for Three-Dimensional Bin Packing

Debajyoti Kar ^{*} Arindam Khan [†] Malin Rau (Speaker) [‡]

1 Introduction

We consider three classical 3D packing problems. In all of these problems, the input is a collection of (rectangular) cuboids (items), each specified by their height, width, and depth. In the 3D Bin Packing problem (3D-BP), the goal is to output a packing of all the items using the minimum number of bins, where each bin is a unit cube. In the 3D Strip Packing problem (3D-SP), we are given a three-dimensional strip having a 1×1 square base and unbounded height, and we have to pack all items minimizing the height of the strip. Finally, in the Minimum Volume Bounding Box problem (3D-MVBB), we seek to obtain a cuboidal box of minimum volume that can accommodate all input items. In all these problems, the items cannot be rotated about any axis, and they must be packed non-overlappingly. Further, we assume that all items and bins/boxes are axis-aligned. We summarize our results in Table 1. While multidimensional packing problems may not appear to be scheduling problems at first glance, geometric feasibility constraints arise in several scheduling settings. Such settings can be modeled as packing problems (see, e.g., [7]).

Problem	Absolute Approximation Ratio		Asymptotic Approximation Ratio	
	Previous Best	Our Result	Previous Best	Our Result
3D-BP	13 [9]	6 (WR: 5)	$T_\infty^2 + \varepsilon < 2.86$ [3]	$3T_\infty/2 + \varepsilon < 2.54$
3D-SP	$46/7 \approx 6.58$ [9]	6	$3/2 + \varepsilon$ [5]	–
3D-MVBB	$46/7 + \varepsilon$ [9, 1]	$3 + \varepsilon$	$46/7 + \varepsilon$ [9, 1]	$1 + \varepsilon$

Table 1: Summary of results. WR denotes the case when 90° rotation around any axis is allowed.

^{*}debajyotikar@iisc.ac.in. Department of Computer Science and Automation, Indian Institute of Science, Bengaluru, India

[†]arindamkhan@iisc.ac.in. Department of Computer Science and Automation, Indian Institute of Science, Bengaluru, India

[‡]malin.rau@chalmers.se. Department of Mathematical Sciences, Chalmers University of Technology and University of Gothenburg, SE-412 96 Göteborg Sweden

2 General approach

First, we discuss our results on the absolute approximation algorithms. We show that any packing into k bins can be transformed into $6k$ *structured* bins. For a constant k , such a structured packing can be found efficiently via a variant of the Generalized Assignment Problem. If k is not constant, we instead apply an asymptotic approximation algorithm to obtain an improved absolute guarantee. A key ingredient is the asymptotic $(3/2 + \varepsilon)$ -approximation for 3D-SP by Jansen and Prädél [5], which produces a packing of height at most $(3/2 + \varepsilon)\text{OPT}_{3\text{D-SP}} + \varepsilon + O_\varepsilon(h_{\max})$.¹ For a sufficiently small constant μ , if $h_{\max} \leq \mu$ and a packing into k bins exists, this yields a packing into $\lfloor 3k/2 \rfloor + 1$ bins. We partition the items into four classes: L (large items with all dimensions exceeding μ) and I_w, I_d, I_h (items with width, depth, or height below μ , respectively), assigning each item arbitrarily to one class if multiple are feasible. Large items can be packed into k bins by brute-force enumeration in polynomial time (for constant k, μ). Each remaining class fits into $\lfloor 3k/2 \rfloor + 1$ bins, giving in total $3(\lfloor 3k/2 \rfloor + 1) + k \leq 7k$ bins.

To improve further, we pack *large* items together with some items from one of the three classes. First, we observe that one of these classes has a volume less than $k/3$. W.l.o.g. let us assume it to be I_h . Now, first, we use a volume-based argument and use an algorithm from [9] to show that we can pack all items in I_h whose width or depth is less than $1/2$. The remaining items in I_h have both width and depth exceeding $1/2$. Next, we show that we can guess the packing of *large* items and almost all items in I_h , except a set of items with small volume. However, with a refined and technical analysis, we finally show that even these remaining items can be packed in the free regions of the six bins. For 3D-BP, this yields an improvement over the previous bound of 13 [9].

Theorem 1 *There exists a polynomial-time 6-approximation algorithm for 3D-BP.*

This directly implies an absolute $(6 + \varepsilon)$ -approximation for 3D-SP – guess the optimal Strip Packing height within a $(1 + \varepsilon)$ -factor, then use appropriate scaling to apply the above 3D-BP result, and finally stack the obtained six bins. With a more careful analysis, we can show there is some extra empty space in the strip, and the resulting height is strictly below 6.

Theorem 2 *There exists a small absolute constant $\rho > 0$, such that for any $\varepsilon > 0$, there is a polynomial-time $(6 - \rho + \varepsilon)$ -approximation algorithm for 3D-SP.*

Another implication of our result is a $(6 + \varepsilon)$ -approximation for the 3D-MVBB problem, using the connection between 3D-SP and 3D-MVBB [1]. However, we then use the power of resource augmentation in 2D-BP to obtain an APTAS for 3D-SP when we are allowed to use resource augmentation. With additional technical adaptations, we obtain a $(3 + \varepsilon)$ -approximation for 3D-MVBB.

Furthermore, surprisingly, unlike 3D-BP and 3D-SP, we show that 3D-MVBB admits an APTAS – settling the asymptotic approximability for the problem.

¹ $O_\varepsilon(f(n))$ indicates that the hidden constant may depend on ε .

Theorem 3 For any $\varepsilon > 0$, there exists a polynomial-time $(3 + \varepsilon)$ -approximation algorithm and an asymptotic polynomial-time approximation scheme for 3D-MVBB.

Finally, we turn our attention to the asymptotic approximability of 3D-BP. We exploit connections between 3D-SP and 3D-BP. Let $\text{OPT}_{3\text{D-SP}}(I)$, $\text{OPT}_{3\text{D-BP}}(I)$ denote the optimum objective value for Strip Packing and Bin Packing, for input I , respectively. Then $\text{OPT}_{3\text{D-SP}}(I) \leq \text{OPT}_{3\text{D-BP}}(I)$, as the bins can be stacked to provide a feasible solution for Strip Packing. This immediately yields a $(3 + \varepsilon)$ -approximation algorithm: First, we obtain a packing in height $(\frac{3}{2} + \varepsilon)\text{OPT}_{3\text{D-SP}}(I) + O_\varepsilon(1)$ using [5]. We then cut the strip into unit-cube bins by cutting it at integral heights $i \in \mathbb{N}$. Items fully contained in $[i, i + 1)$ are packed in the $(2i + 1)$ -th bin. Remaining items that are cut by the x - y axis-aligned plane at height i are packed in the $(2i)$ -th bin. This gives us a packing into $(3 + \varepsilon)\text{OPT}_{3\text{D-SP}}(I) + O_\varepsilon(1)$ bins.

To improve beyond T_∞^2 , our approach will be to find a packing such that the items that are cut do not have large heights. Towards this, we use *harmonic rounding* [8], where the function f_k rounds up $\alpha \in (1/k, 1]$ to nearest larger number of the form $1/q$ where $q \in \mathbb{Z}$. Thus, for $\alpha_i \in (1/(q + 1), 1/q]$, $f_k(\alpha_i) := 1/q$, for $q \in [k - 1]$. Otherwise, $f_k(\alpha_i) := \alpha_i$. It is well-known [2] that, for any sequence $\alpha_1, \alpha_2, \dots, \alpha_n$, with $\alpha_i \in (0, 1]$ and $\sum_{i=1}^n \alpha_i \leq 1$, for a small enough ε , we have $\sum_{i=1}^n f_{1/\varepsilon}(\alpha_i) \leq T_\infty + \varepsilon \approx 1.691$.

We first round the item heights in I using $f_{1/\varepsilon}$ to obtain a new set of items I^∞ and obtain a 3D Strip Packing of them using the algorithm by [6]. Let $\text{OPT}_{3\text{D-BP}}^{T_\infty}(I^\infty)$ be the optimal number of $1 \times 1 \times T_\infty$ -sized bins needed to pack all items in I^∞ . Then, it is easy to see that $\text{OPT}_{3\text{D-SP}}(I^\infty) \leq T_\infty \text{OPT}_{3\text{D-BP}}^{T_\infty}(I^\infty)$.² Then we have $\frac{3}{2}\text{OPT}_{3\text{D-SP}}(I^\infty) \leq \frac{3T_\infty}{2}\text{OPT}_{3\text{D-BP}}^{T_\infty}(I^\infty) \leq \frac{3T_\infty}{2}\text{OPT}_{3\text{D-BP}}(I)$. The last inequality follows from harmonic rounding.

We must ensure that the *tall* items in I^∞ packed into a strip of height $\frac{3}{2}\text{OPT}_{3\text{D-SP}}(I^\infty)$ are not intersected by cutting planes at integral heights; we call this the *tall-not-sliced* property. A related idea was used by Bansal et al. [2] to obtain an alternative $(T_\infty + \varepsilon)$ -approximation for 2D-BP, though the three-dimensional setting is substantially more involved. Our approach exploits structural properties of the packing from [5]. We first show that the strip can be partitioned into $O_\varepsilon(1)$ cuboids such that items within each cuboid are *similar*. We then show that almost all tall items in I^∞ with a common height $1/q$ (for some $q \in [k - 1]$) can be packed at heights that are multiples of $1/q$, incurring only a small additive loss. This guarantees that these items are not cut by integral-height planes. Items with large width and depth are assigned to containers via a linear program, while the remaining items—except for a small total volume—are packed using variants of the Next-Fit-Decreasing-Height (NFDH) algorithm [4]. Finally, the remaining items are placed in the leftover free regions and an additional $O_\varepsilon(1)$ bins, yielding an improved approximation guarantee for 3D-BP after nearly two decades.

Theorem 4 For any $\varepsilon > 0$, there exists a polynomial-time algorithm for 3D-BP with an asymptotic approximation ratio $(3T_\infty/2 + \varepsilon) \approx 2.54$.

²For simplicity, we ignore the $O_\varepsilon(1)$.

References

- [1] Helmut Alt and Nadja Scharf. Approximating smallest containers for packing three-dimensional convex objects. *International Journal of Computational Geometry & Applications*, 28(2):111–128, 2018.
- [2] Nikhil Bansal, Xin Han, Kazuo Iwama, Maxim Sviridenko, and Guochuan Zhang. A harmonic algorithm for the 3d strip packing problem. *SIAM Journal on Computing*, 42(2):579–592, 2013.
- [3] Alberto Caprara. Packing d-dimensional bins in d stages. *Mathematics of Operations Research*, 33(1):203–215, 2008.
- [4] Edward G Coffman, Jr, Michael R Garey, David S Johnson, and Robert Endre Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9(4):808–826, 1980.
- [5] Klaus Jansen and Lars Prädell. A new asymptotic approximation algorithm for 3-dimensional strip packing. In *SOFSEM*, pages 327–338, 2014.
- [6] Klaus Jansen and Roberto Solis-Oba. An asymptotic approximation algorithm for 3 d-strip packing. In *SODA*, pages 143–152, 2006.
- [7] František Koblasa, Miroslav Vavroušek, and František Manlig. Three dimensional bin packing problem in batch scheduling. In *Proceedings of the 34th International Conference on Mathematical Methods in Economics (MME 2016)*, 2016.
- [8] C. C. Lee and D. T. Lee. A simple on-line bin-packing algorithm. *Journal of the ACM*, 32(3):562–572, 1985.
- [9] Keqin Li and Kam-Hoi Cheng. On three-dimensional packing. *SIAM Journal on Computing*, 19(5):847–867, 1990.

A Tight 2-Approximation for Demand Bin Packing

Debajyoti Kar ^{*} Arindam Khan [†] Malin Rau [‡]
 Ann-Brith Strömberg [§] Albert Vesterlund (Speaker) [¶]

1 Introduction

We consider the DEMAND BIN PACKING problem (DBP). In this problem, we are given a finite set \mathcal{I} of items, where each item $i \in \mathcal{I}$ has a *width* $w(i) \in (0, 1]$ and a *height* $h(i) \in (0, 1]$. These items have to be packed into 2-dimensional bins that are identical rectangles of size $[0, 1] \times [0, 1]$. A *packing* of the items in \mathcal{I} into bins is a mapping $\sigma : \mathcal{I} \rightarrow \mathbb{N}_+ \times [0, 1], i \mapsto (b_\sigma(i), x_\sigma(i))$, where $b_\sigma(i)$ denotes the index of the bin that contains item i and $x_\sigma(i)$ is the horizontal start position of i in that bin. A packing σ is *feasible* if for all $i \in \mathcal{I}$, $x_\sigma(i) + w(i) \leq 1$, and for every bin b and every $x \in [0, 1)$, $h(\{i \in \mathcal{I} \mid b_\sigma(i) = b, x_\sigma(i) \leq x < x_\sigma(i) + w(i)\}) \leq 1$, where $h(\mathcal{I}') = \sum_{i \in \mathcal{I}'} h(i)$ for any (sub)set $\mathcal{I}' \subseteq \mathcal{I}$, i.e., for any x , the total height of the items passing through x is at most 1. The objective of the DBP problem is to find a feasible packing $\sigma(\mathcal{I})$ of items \mathcal{I} that minimizes the number of bins used, i.e., $\max_{i \in \mathcal{I}} b_\sigma(i)$. This problem generalizes the classical *Bin Packing* problem, which is obtained when all items have identical unit heights.

As a practical example, consider the problem of scheduling jobs on computer clusters. Each job has an energy demand and a processing time, while each cluster has a maximum energy capacity and maximum time. Then, the problem of minimizing the number of clusters needed to process all jobs is an example of the DBP problem.

1.1 Our main result and technical contribution

We resolve both open questions posed by Albers, Gálvez, and Özdemir [1]. We design a $(1 + \ln(\frac{3}{2}) + \epsilon)$ -asymptotic approximation algorithm for Demand Bin Packing, matching the best-known asymptotic guarantee for Geometric Bin Packing.

^{*}debajyotikar@iisc.ac.in. Indian Institute of Science

[†]arindamkhan@iisc.ac.in. Indian Institute of Science

[‡]malin.rau@chalmers.se. Department of Mathematical Sciences, Chalmers University of Technology & University of Gothenburg

[§]anstr@chalmers.se. Department of Mathematical Sciences, Chalmers University of Technology & University of Gothenburg.

[¶]albertv@chalmers.se. Department of Mathematical Sciences, Chalmers University of Technology & University of Gothenburg.

Theorem 1 *For any $\epsilon > 0$, there exists a polynomial-time algorithm for DEMAND BIN PACKING with asymptotic approximation ratio $(1 + \ln(\frac{3}{2}) + \epsilon)$.*

In addition, we present a polynomial-time absolute 2-approximation algorithm for the problem, settling its absolute approximability.

Theorem 2 *There exists a polynomial-time 2-approximation algorithm for Demand Bin Packing.*

The key component of proving Theorem 1 is a structural transformation showing that any optimal packing σ into m bins, can be transformed into a new packing σ' where at least one side of each sufficiently large item is rounded, while adding at most $\frac{1}{2}m + \lceil O(\epsilon)m \rceil + 3$ bins. This is done by partitioning \mathcal{I}_b into sets \mathcal{I}'_b , \mathcal{I}''_b and $\mathcal{I}_{\text{rem},b}$ such that the total area of $\mathcal{I}_{\text{rem},b}$ is at most $O(\epsilon)$, \mathcal{I}'_b is packed into a bin either with an empty narrow horizontal strip, vertical strip, or into a bin with a specific structure such that the items can be rounded, and \mathcal{I}''_b is packed into one of three types of *half-filled bins*. Compared to Geometric Bin Packing, a challenge arises when attempting to clear a horizontal strip of height ϵ . The selection of potential items for this removal can be tricky, as in DBP items are not assigned a vertical position. Therefore, we introduce two item-selection procedures that sweep the bin from left to right to identify a set of removable items, to fulfil the above mentioned structures for \mathcal{I}'_b . We then separate any instance of a bin into cases, depending on the placements of certain wide items, certain tall items and some other case-specific items of interest. The goal is to repack enough items so that either (i) a strip of width (or height) ϵ becomes free of items in the original bin, or (ii) the repackaged items are selected such that the remaining items can be rounded and organized using linear grouping. In both cases, the repackaged items should be able to be repackaged into either half-filled bins, or have sufficiently small area. Then, we show this restructuring requires at most $\frac{1}{2} + O(\epsilon)$ extra space per bin, while allowing any item to have at least one rounded side. This yields the structural repacking of σ into σ' using at most $\frac{3}{2}m + \lceil O(\epsilon)m \rceil + 3$ bins. Using this structural repacking, we introduce the additional structures *chains*, *stacks*, and *profiles*, for differently sized items. By introducing these structures, we further narrow down the possible configurations of bins, in order to obtain an asymptotic $(\frac{3}{2} + \epsilon)$ -approximation. Further, by presenting a previously unknown result for the demand knapsack problem with constant profit-to-area ratio, we show that a configuration LP of the DBP admits an efficient approximate solution. This enables the proof for an APTAS for the configuration LP, and we present an algorithm which, following the *Round-and-Approx* framework [2], yield the final piece for proving Theorem 1.

For proving Theorem 2, we provide two key lemmas: the first shows that, for a constant K , any optimal packing using at most K bins has a packing into $\text{OPT} + 2$ bins which is computable in polynomial time; the second shows that, when the optimal packing uses a single bin, a packing using two bins can be computed in polynomial time. If K is large enough, the asymptotic approximation

algorithm can be used. Observe that if $K \geq 2$, the first lemma trivially yields a 2-approximation. To prove the first lemma, we utilize the fact that for a constant K , a structured packing can be achieved by discarding at most $O(\varepsilon)$ area from the bins. In the case $K = 1$, we make a connection between the demand packing and the geometric packing, showing that the items can be partitioned such that each partition can be packed geometrically into two separate bins, with each partition fitting within the profile of the items in the original bin. The second lemma is proved by utilizing the result of the first lemma, combined with an elaborate case analysis of the optimal packing for $\text{OPT} = 1$.

References

- [1] Albers, S., Gálvez, W. & Özdemir, Ö. On the 2D Demand Bin Packing Problem: Hardness and Approximation Algorithms. *Procedia Computer Science*. **273** pp. 301-308 (2025)
- [2] Bansal, N., Caprara, A. & Sviridenko, M. A new approximation method for set covering problems, with applications to multidimensional bin packing. *SIAM Journal On Computing*. **39**, 1256-1278 (2010)

Graph Scheduling with Group Completion Times

Lars Rohwedder*

Leander Schnaars (Speaker)[†]

1 Introduction

In Graph Scheduling with group completion times, the input consists of a multi-graph $G = (V, E)$ and jobs $J_1, \dots, J_B \subseteq E$ with weights $\omega_1, \dots, \omega_B \in \mathbb{R}_{\geq 0}$. In other words, every job consists of some set of edges, where these sets do not necessarily have to be disjoint. In each discrete time step, we are allowed to schedule a matching on the graph, that is to say a set of edges not sharing any vertex. A job is completed if all of its edges have been scheduled. The goal is to minimize the weighted sum of completion times. Group completion times can model many common objectives. If for example there is a single job containing every edge, then the problem is equivalent to makespan minimization, while if every edge belongs to a unique job, this represents min sum edge coloring.

Graph Scheduling has been studied mainly in the context of Coflow Scheduling [3, 2, 1, 6, 4, 8] and Data Migration [7, 5]. In Coflow Scheduling, the underlying graph is required to be bipartite and the groups are formed by a disjoint partition of the edge set. This models a data exchange task commonly found in applications such as distributed computing, where between computation stages the different servers have to exchange intermediate results. For several years, the best known guarantees were multiple different 4-approximations [2, 10, 1, 4], while there is a factor $(2 - \epsilon)$ -approximation hardness assuming $\mathbb{P} \neq \text{NP}$ [9]. Recently, the factor 4 has been improved to 3.415 in [8].

In Data Migration the graph is unrestricted, but each job consists of exactly the edges adjacent to some vertex, which implies that every edge belongs to exactly two jobs. As the name implies, this problem models a general data migration setting in a network, where every vertex represents a server and each edge is a data transmission requirement. A server finishes its data transmission task if all adjacent edges have been served. The best known approximation ratio for this setting is $(1 + \phi) \approx 2.618$ [7].

*rohwedder@imada.sdu.dk. University of Southern Denmark, Odense, Denmark. Supported by Dutch Research Council (NWO) project “The Twilight Zone of Efficiency: Optimality of Quasi-Polynomial Time Algorithms” [grant number OCEN.W.21.268]

[†]leander.schnaars@tum.de. Technical University of Munich, Munich, Germany. Supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - GRK 2201/2 - Projektnummer 277991500

As remarked by Fukunaga [4], some of the 4-approximation algorithms for Coflow Scheduling [2, 10] also work for general graphs, hence the best so far known approximation ratio for Graph Scheduling is 4. In [8], Rohwedder and Schnaars introduce iterative rounding techniques that achieve an essentially tight $(2 + \epsilon)$ -approximation for Coflow Scheduling in an asymptotic regime where $\text{OPT} \gg \sum_{j \in [B]} \omega_j$. Our main result extends this to general Graph Scheduling:

Theorem 1 *Given a Graph Scheduling instance G with cost vector $\omega \in \mathbb{R}_{\geq 0}^B$ and parameters $\tau \in \mathbb{N}_{\geq 2}$ and $\epsilon > 0$, in polynomial time we can find a solution S such that:*

$$\omega(S) \leq (2 + \epsilon) \cdot \text{OPT} + \mathcal{O}\left(\frac{1}{\epsilon}\right) \cdot \sum_{j \in [B]} \omega_j$$

For instances with $\text{OPT} \gg \sum_{j \in [B]} \omega_j$, this gives a $(2 + \epsilon)$ -approximation, which is essentially tight due to the $(2 - \epsilon)$ lower bound. Combining the result for Graph Scheduling with a refined analysis of the previously best known local ratio algorithm for Data Migration allows us to obtain an improvement over the 2.618-approximation for Data Migration:

Theorem 2 *There is a 2.617-approximation algorithm for Data Migration with unit sized edges.*

While the main focus is the extension of the techniques from Coflow Scheduling to general graphs and other related settings, from Theorem 1 combined with known results for Graph Scheduling, an improved approximation algorithm for the general case also follows.

Theorem 3 *There is a polynomial time $\frac{1104}{281} < 3.93$ -approximation algorithm for Graph Scheduling.*

2 Technical Ideas

The approach is a generalization of the algorithms [8] for Coflow Scheduling, which consist of two core steps: In the first step, a deadline is determined for each job. These deadlines are chosen in a way to both fulfill a cost approximation guarantee while also guaranteeing feasibility of an allocation linear program. In the second step, through an iterative rounding procedure an integral point is obtained from the feasible LP. This integral solution does not directly correspond to a feasible schedule, as edges are only assigned to so called blocks and not to individual time slots. To obtain a valid schedule, a time slot assignment is computed using König's Theorem for bipartite graphs.

The algorithm for Graph Scheduling follows the same structure, but due to the general multigraph setting both the LP as well as the allocation algorithm have to be significantly modified. The first step of determining the deadlines works essentially unmodified. The second step however requires larger technical

modifications, as we employ results from graph edge coloring theory for which we have to structurally strengthen the LP by adding exponentially many odd-set inequalities. This added structure improves the obtained guarantees, but requires additional care when rounding, as iterated rounding is highly sensitive to the number of constraints.

The main reason for the required additional structure is due to a negative result from graph edge coloring theory by Shannon [11]. A valid k -edge coloring is equivalent to a partition of the edges into k -matchings, which corresponds to a valid schedule using k time slots. While König's theorem guarantees that any bipartite graph can be edge colored using $\Delta(G)$ colors, for general multigraphs up to $\frac{3}{2}\Delta(G)$ many colors are required, which means that a naive extension of the Coflow Scheduling techniques yields an up to $\frac{3}{2}$ loss of approximation ratio. The extended structure of the LP essentially guarantees that this worst case cannot occur by enforcing a bound on certain graph theoretic parameters which are closely linked to edge coloring.

References

- [1] Saksham Agarwal, Shijin Rajakrishnan, Akshay Narayan, Rachit Agarwal, David Shmoys, and Amin Vahdat. "Sincronia: near-optimal network design for coflows". In: *Proceedings of SIGCOMM*. ACM, 2018, pp. 16–29. ISBN: 978-1-4503-5567-4. DOI: 10.1145/3230543.3230569.
- [2] Saba Ahmadi, Samir Khuller, Manish Purohit, and Sheng Yang. "On Scheduling Coflows". In: *Proceedings of IPCO*. Vol. 10328. Series Title: Lecture Notes in Computer Science. Springer International Publishing, 2017, pp. 13–24. ISBN: 978-3-319-59250-3. DOI: 10.1007/978-3-319-59250-3_2.
- [3] Mosharaf Chowdhury, Yuan Zhong, and Ion Stoica. "Efficient coflow scheduling with Varys". In: *Proceedings of SIGCOMM*. ACM, 2014, pp. 443–454. ISBN: 978-1-4503-2836-4. DOI: 10.1145/2619239.2626315.
- [4] Takuro Fukunaga. "Integrality Gap of Time-Indexed Linear Programming Relaxation for Coflow Scheduling". In: *Proceedings of APPROX/RANDOM*. Vol. 245. LIPIcs. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 36:1–36:13. ISBN: 978-3-95977-249-5. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2022.36.
- [5] Rajiv Gandhi, Magnús M Halldórsson, Guy Kortsarz, and Hadas Shachnai. "Corrigendum: Improved Results for Data Migration and Open Shop Scheduling". In: *ACM Trans. Algorithms* 9.4 (2013), pp. 1–7. DOI: 10.1145/2500123.
- [6] Samir Khuller, Jingling Li, Pascal Sturmfels, Kevin Sun, and Prayaag Venkat. "Select and permute: An improved online framework for scheduling to minimize weighted completion time". In: *Theoretical Computer Science*

- 795 (2019). Publisher: Elsevier BV, pp. 420–431. DOI: 10.1016/j.tcs.2019.07.026.
- [7] Julián Mestre. “Adaptive Local Ratio”. In: *SIAM J. Comput.* 39.7 (2010), pp. 3038–3057. DOI: 10.1137/080731712.
- [8] Lars Rohwedder and Leander Schnaars. “3.415-Approximation for Coflow Scheduling via Iterated Rounding”. In: *Proceedings of ICALP*. Vol. 334. LIPIcs. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2025, 128:1–128:19. ISBN: 978-3-95977-372-0. DOI: 10.4230/LIPIcs.ICALP.2025.128.
- [9] Sushant Sachdeva and Rishi Saket. “Optimal Inapproximability for Scheduling Problems via Structural Hardness for Hypergraph Vertex Cover”. In: *2013 IEEE Conference on Computational Complexity*. IEEE, 2013, pp. 219–229. DOI: 10.1109/coc.2013.30.
- [10] Mehrnoosh Shafiee and Javad Ghaderi. “An improved bound for minimizing the total weighted completion time of coflows in datacenters”. In: *IEEE ACM Trans. Netw.* 26.4 (2018), pp. 1674–1687. DOI: 10.1109/TNET.2018.2845852.
- [11] Claude Shannon. “A theorem on coloring the lines of a network”. In: *J. Math. Phys.* 28.1-4 (1949), pp. 148–152.

Scheduling Framework for Edge-to-Cloud Task Offloading

Thomas Erlebach ^{*} Natalia V. Shakhlevich (Speaker) [†]
Akiyoshi Shioura, [‡] Jie Xu [§]

1 Introduction

Edge-to-Cloud offloading plays an important role in modern distributed computing systems. Users expect fast processing of their tasks either on edge resources located nearby, or on cloud resources located elsewhere. Processing on the edge typically does not incur a substantial data transfer delay due to the edge proximity, but the capacity and computation speed of edge machines might be limited. If an alternative mode is selected, the job is offloaded from the edge to the cloud for processing by more powerful resources; this, however, incurs communication delays, especially if large volumes of data needed for job processing are to be transferred.

The problems of efficient management of edge/cloud resources attract significant attention of researchers, mostly in the distributed computing community. The recent review paper [1] quotes more than 20 surveys on edge-to-cloud offloading, published only in 2019-2023. Mainstream research mostly focuses on computing aspects of edge-to-cloud offloading. Our study is complementary: we explore mathematical properties of relevant problems and design optimization algorithms with provable performance guarantees.

2 Problem definition

We are given a set of jobs $J = \{1, \dots, n\}$, the sets of edge and cloud machines \mathcal{A} and \mathcal{B} , and transfer link T for offloading jobs and associated data to the cloud. Each job is available at time 0 and needs to be either allocated to the edge or

^{*}thomas.erlebach@durham.ac.uk. Department of Computer Science, Durham University, Stockton Road, Durham, DH1 3LE, UK.

[†]n.shakhlevich@leeds.ac.uk. School of Computer Science, University of Leeds, Leeds, LS2 9JT, UK.

[‡]shioura.a.aa@m.titech.ac.jp. Department of Industrial Engineering and Economics, Institute of Science Tokyo, Ookayama 2-12-1, Meguro-ku, Tokyo, 1528550, Japan.

[§]j.xu@leeds.ac.uk. School of Computer Science, University of Leeds, Leeds, LS2 9JT, UK.

offloaded to the cloud. If job j is allocated to the edge, then the transfer time from the client to the edge is negligible. Job processing time on any machine from \mathcal{A} is a_j . If job j is offloaded to the cloud, then it first uses the link T during time τ_j , followed by processing in the cloud by one of the machines from \mathcal{B} during time b_j . Assuming that a response is small, the time for sending it back to the edge and then to the client is ignored. Preemption of all operations, a_j , τ_j and b_j , is not allowed.

In order to produce a schedule, the following decisions should be made:

- (i) partitioning of the jobs into two sets for processing on the edge or cloud,
- (ii) sequencing of the edge jobs on the edge machines and of the cloud jobs on the cloud machines.

Additionally, if the transfer is limited to one job at a time, then there is one more scheduling decision:

- (iii) sequencing of the jobs on the transfer link.

For a given schedule, the completion time C_j of a job j is the time when a job completes its processing on a machine it is allocated to. The goal is to minimize the makespan $C_{\max} = \max\{C_j | j = 1, \dots, n\}$ or the total completion time $\sum C_j = \sum_{j=1}^n C_j$. Note that for the jobs available simultaneously, minimizing $\sum C_j$ can be interpreted as maximizing the throughput.

3 Results

For each objective, C_{\max} and $\sum C_j$, there are 17 non-trivial versions of the edge-to-cloud offloading problem, depending on a combination of three parameters: the number of edge machines, the number of cloud machines and ability of link T to transfer one job at a time or multiple jobs at a time. Most of the variants of the problem are NP-hard. We identify 8 groups of similar problems, and for each group we formulate a common methodology for getting an exact or approximate solution.

Our theoretical results may serve as the foundation for designing heuristic algorithms tuned to more complicated scenarios arising in modern edge-cloud systems.

References

- [1] GKONIS, P., GIANNOPOULOS, A., TRAKADAS, P., MASIP-BRUIN, X., AND D'ANDRIA, F. (2023). A survey on IoT-Edge-Cloud continuum systems: Status, challenges, use cases, and open issues. *Future Internet*, 15, 383.

Approximating optimal broadcast of files in a hose-model network*

Thomas Erlebach (Speaker) [†] Naveen Garg [‡] Sukriti Gupta [§]
Amitabh Trehan [¶]

1 Introduction

We consider the problem of file sharing between peers in a network. The file to be shared initially resides with a *root* peer and has to be *broadcast*, i.e., sent to all the remaining n peers. The peers are all connected to a *core network* with links of bounded upload and download capacities. The core network is suitably provisioned so that any traffic matrix between the peers that meets the upload and download capacity constraints on each peer can be supported; this is commonly referred to as the *hose model* [1], a flexible model of bandwidth reservation where only the maximum ingress and egress bandwidth for each endpoint needs to be specified. The rate of transfer between two peers is only constrained by the upload and download capacities of the sending and receiving peer, respectively. For instance, if peer i has an upload capacity of 10Mbps and peers j, k have download capacities of 6Mbps then peer i can transfer data to peers j, k at a rate of 5Mbps or to peer j at 6Mbps and peer k at 4Mbps or vice-versa. Our objective is to schedule the transfers between peers efficiently. While there has been a lot of previous work on broadcasting in networks, most of it focuses on the structure of the communication network [4, 5], which in our model plays no role.

When broadcasting large files, a chunk is the smallest unit of the file that must be completely available to a peer before it can start sharing any part of it with another peer. We focus on the problem of distributing one chunk residing at the root peer to the remaining peers in the network. Our model permits a chunk to be split and for a peer to send parts of the chunk to multiple peers concurrently. We consider two variants of this model. In the *non-migratory setting*, a peer must receive the entire chunk from only one peer (its parent peer) while in the *migratory*

*The talk is based on the extended abstract [2].

[†]thomas.erlebach@durham.ac.uk. Department of Computer Science, Durham University, UK.

[‡]naveen@cse.iitd.ac.in. Department of Computer Science and Engineering, Indian Institute of Technology Delhi, India.

[§]csz258469@iitd.ac.in. Department of Computer Science and Engineering, Indian Institute of Technology Delhi, India.

[¶]amitabh.trehan@durham.ac.uk. Department of Computer Science, Durham University, UK.

setting, a peer could receive parts of the chunk from different peers. Our objective in both settings is to schedule these transfers so that the time by which all peers receive the chunk (the *makespan*) is minimised.

If upload capacities were uniform then it would be natural to distribute the chunk in order of decreasing download capacities, with the peer having the largest download capacity receiving it first. Similarly, if the download capacities were uniform it would make sense to distribute the chunk in order of decreasing upload capacity and first send it to the peer with the largest upload capacity. When upload and download capacities are identical (the symmetric setting) both of these ideas yield the same order for distributing the chunk to the peers. However, when upload and download capacities are different (for a peer and across peers) there seems to be no natural preference order for distributing the chunk. For the non-migratory setting, minimizing makespan is NP-hard [3], but for the migratory setting, we do not know if the problem is NP-hard.

2 Our Results

We address broadcast in the hose model in its full generality, i.e., the asymmetric heterogeneous setting where the upload and download capacities of each peer can be different. Previous work has only considered the symmetric setting when the upload and download capacity of each peer are identical [3] or the setting when download capacities are uniform [6]. The problem of finding a constant approximation for the asymmetric setting has been open, and even the setting of uniform upload capacities had not been considered earlier.

Additionally, we introduce the novel migratory setting that allows a peer to download from multiple peers subject to its download capacity constraint. This is a natural relaxation and does not lead to loss of data identity, as we still process the chunk as a whole. The constraint that a peer starts uploading only after receiving the complete chunk is important and separates our model from the fluid limit model.

Our first result is a novel completion time integer program for the problem of minimising makespan in the migratory setting. Formulating this integer program requires proving that a certain non-trivial property is necessary and sufficient for characterising valid broadcast schedules. Relaxing the integrality constraints gives a linear program, and we exploit the structure of the optimum solution of this LP to obtain a solution with makespan at most $e^{1/e}\text{OPT} + P$. Here, OPT is the makespan of an optimal schedule and P is the time required for the slowest peer to download the chunk, and while it is definitely less than OPT , it could be significantly smaller.

For the non-migratory setting, we do not know how to formulate a suitable integer program, but we circumvent this difficulty by restricting attention to so-called *slotted schedules*. We formulate an LP by restricting attention to downloads that fit into ‘slots’ and establish that this restriction does not lead to a significant increase in makespan. Our algorithm for rounding the optimum solution to the

linear program proceeds through three stages, each of which requires careful reasoning about the structure of the solution. This yields a solution of makespan at most $3OPT' + P$ where OPT' is the minimum makespan of any slotted schedule. Since we can show that $OPT' \leq 6OPT$, we obtain a solution of makespan at most $18OPT + P$.

We also consider 2 special cases in the non-migratory setting. When all download capacities are identical, we extend the results by Liu [6] to obtain a 2-approximation algorithm. When all upload capacities are identical, we modify the download capacities of some peers and add some dummy peers to obtain a modified instance with makespan at most $2OPT + P$. We then find an optimum schedule for this modified instance and convert it into a schedule for the original instance with makespan at most $2OPT + 2P$.

In all the results mentioned, we have assumed that the root peer has an upload capacity that is at least as large as the largest download capacity. We show in [2] how our algorithms can be adapted to handle the setting where this condition is not met.

References

- [1] Nick G. Duffield, Pawan Goyal, Albert G. Greenberg, Partho Pratim Mishra, K. K. Ramakrishnan, and Jacobus E. van der Merwe. A flexible model for resource management in virtual private networks. In *ACM SIGCOMM 1999*, pages 95–108. ACM, 1999. doi:10.1145/316188.316209.
- [2] Thomas Erlebach, Naveen Garg, Sukriti Gupta, and Amitabh Trehan. Approximating optimal broadcast of files in a hose-model network. In *FSTTCS 2025*, volume 360 of *LIPICs*, pages 30:1–30:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPICs.FSTTCS.2025.30.
- [3] Kai-Simon Goetzmann, Tobias Harks, Max Klimm, and Konstantin Miller. Optimal file distribution in peer-to-peer networks. In *ISAAC 2011*, volume 7074 of *LNCS*, pages 210–219. Springer, 2011. doi:10.1007/978-3-642-25591-5_23.
- [4] Sandra Mitchell Hedetniemi, Stephen T. Hedetniemi, and Arthur L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18(4):319–349, 1988. doi:10.1002/net.3230180406.
- [5] Juraj Hromkovič, Ralf Klasing, Burkhard Monien, and Regine Peine. Dissemination of information in interconnection networks (broadcasting & gossiping). In Ding-Zhu Du and D. Frank Hsu, editors, *Combinatorial Network Theory*, page 125–212. Springer US, Boston, MA, 1996. doi:10.1007/978-1-4757-2491-2_5.
- [6] Pangfeng Liu. Broadcast scheduling optimization for heterogeneous cluster systems. *Journal of Algorithms*, 42(1):135–152, 2002. doi:10.1006/jagm.2001.1204.

Online and offline algorithms for weighted makespan minimization

Aflatoun Amouzandeh* Klaus Jansen† Lis Piroton‡
 Rob van Stee (Speaker)§ Corinna Wambsganz¶

1 Introduction

Makespan minimization is a fundamental and extensively studied problem in scheduling theory, with a significant body of research devoted to it. A sequence of jobs must be scheduled non-preemptively on m machines. Each job j is characterized by a processing time p_j and a non-negative arrival time r_j , where $1 \leq j \leq n$. In this paper, we assume that jobs arrive over time, i.e., each job becomes available for processing only at its release time. The goal is to minimize the *makespan*, defined as the maximum completion time of any job in the schedule. The problem is obviously NP-hard as it generalizes the makespan problem where all jobs arrive at time 0. Regarding the online setting, the best competitive ratio currently known, for general m , is equal to 1.5 and attained by the LPT (Longest Processing Time First) algorithm [4].

A more general objective is the *weighted makespan*, denoted as $WC_{\max} = \max_j \{w_j C_j\}$, which measures the maximum *weighted completion time* of all jobs. Each job j is assigned a positive weight w_j , reflecting its relative importance. The classical makespan minimization problem is a special case in which all weights are equal. This objective function was first considered only fairly recently and has received some attention [2, 6, 9, 10]. Even on a single machine, it is nontrivial to optimize the weighted makespan non-preemptively if jobs arrive over time. In the standard three-field notation introduced by Graham et al. [5], this problem is denoted as $1 \mid r_j \mid WC_{\max}$. If all jobs arrive at time 0 or preemptions are allowed, then the algorithm Largest Weight (LW) which always runs the job that has largest weight is 1-competitive.

In the *offline* setting, all job parameters (processing times, release times, and weights) are known in advance, allowing the algorithm to make better decisions. In

*Aflatoun.Amouzandeh@uni-siegen.de. Department of Mathematics, University of Siegen, Germany.

†kj@informatik.uni-kiel.de University of Kiel, Germany.

‡lpi@informatik.uni-kiel.de University of Kiel, Germany.

§rob.vansteer@uni-siegen.de. Department of Mathematics, University of Siegen, Germany.

¶lpi@informatik.uni-kiel.de University of Kiel, Germany.

contrast, the *online setting* assumes that jobs arrive over time and decisions must be made without knowledge of the future input.

Lu et al. [8] considered the offline single-machine problem with job rejection but without release dates. They showed that this problem is NP-hard and proposed a fully polynomial-time approximation scheme (FPTAS). More recently, Sun [10] proposed a $(2 - \frac{1}{m})$ -approximation algorithm for the weighted makespan scheduling problem on m identical machines, again in the setting without release dates ($P||WC_{\max}$). The algorithm runs List Scheduling on the jobs after sorting them by decreasing weight. Sun also introduced a randomized efficient polynomial-time approximation scheme (EPTAS) for this problem, as well as a randomized FPTAS for the case when m is fixed.

One important extension to the classical scheduling framework is the use of *restarts*, which forms a central focus of this paper. In this model, a running job can be interrupted when a more urgent job arrives (e.g., a smaller or heavier job in the case of weighted jobs). However, any work done on the interrupted job is lost, and it must be restarted from the beginning. This model, also known as *preemption with restarts*, differs from the more commonly studied preemption with resume model, where interrupted jobs can be resumed from the point of interruption.

In this paper, we consider the problem of online scheduling with restarts for minimizing the weighted makespan WC_{\max} . Specifically, we study the problem $1|r_j, \text{online, restart}|WC_{\max}$ and establish a new lower bound of 1.4656 on the competitive ratio for deterministic online algorithms in the general setting with restarts. Remarkably, this is the same value that Liang et al. [7] achieved for jobs of unit size and a single allowed restart. However, the problems are different.

Our main result is a deterministic online algorithm that achieves a competitive ratio better than 1.3098 for the case where all jobs have identical processing times.

The idea of our algorithm is as follows. In principle, we prefer to run the heaviest job. This would mean running LW with interruptions. However, if a significant part of the currently running job c has already been completed, it can be better to let it finish first, rather than interrupt it for an incoming job j . This depends on what the completion time of j will be when it completes immediately following c versus what its completion time will be if it starts immediately, and also on the relative weights of c and j . If c started at time t and j arrives at time t' , then if c is allowed to complete j will complete not before time $t + 2$. We use the following very simple rule: we will interrupt c if j is heavier than c and $t + 2$ is more than R times the completion time of j if it starts immediately, which is $t' + 1$ (assuming no further interruptions occur).

We also prove a lower bound of 1.2344 for this case. Finally, we show that the offline version of this problem is strongly NP-hard for general processing times and we provide a PTAS. The PTAS is based on Afrati et al. [1]. Some of our exposition is adapted from Chekuri and Khanna [3]; we fix some minor inaccuracies and fill in some missing details of the dynamic program in our paper.

References

- [1] Foto N. Afrati, Evripidis Bampis, Chandra Chekuri, David R. Karger, Claire Kenyon, Sanjeev Khanna, Ioannis Milis, Maurice Queyranne, Martin Skutella, Clifford Stein, and Maxim Sviridenko. Approximation schemes for minimizing average weighted completion time with release dates. In *FOCS*, pages 32–44. IEEE Computer Society, 1999.
- [2] Xing Chai, Lingfa Lu, Wenhua Li, and Liqi Zhang. Best-possible online algorithms for single machine scheduling to minimize the maximum weighted completion time. *Asia-Pacific Journal of Operational Research*, 35(06):1850048, 2018.
- [3] Chandra Chekuri and Sanjeev Khanna. Approximation algorithms for minimizing averageweight completion time. In Joseph Y.-T. Leung, editor, *Handbook of Scheduling - Algorithms, Models, and Performance Analysis*. Chapman and Hall/CRC, 2004.
- [4] B. Chen and A. Vestjens. Scheduling on identical machines: How good is LPT in an on-line setting? *Operations Research Letters*, 21(4):165–19, 1997.
- [5] Ronald Lewis Graham, Eugene Leighton Lawler, Jan Karel Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of discrete mathematics*, volume 5, pages 287–326. Elsevier, 1979.
- [6] Wenjie Li. A best possible online algorithm for the parallel-machine scheduling to minimize the maximum weighted completion time. *Asia-Pacific Journal of Operational Research*, 32(04):1550030, 2015.
- [7] Xiaoxiao Liang, Lingfa Lu, Xueke Sun, Xue Yu, and Lili Zuo. Online scheduling on a single machine with one restart for all jobs to minimize the weighted makespan. *AIMS Mathematics*, 9(1):2518–2529, 2024.
- [8] Lingfa Lu, Liqi Zhang, and Jinwen Ou. Single machine scheduling with rejection to minimize the weighted makespan. In *Algorithmic Aspects in Information and Management: 15th International Conference, AAIM 2021, Virtual Event, December 20–22, 2021, Proceedings 15*, pages 96–110. Springer, 2021.
- [9] Feng Qi and Yuan Jinjiang. Np-hardness of a multicriteria scheduling on two families of jobs. *Operations Research Transactions*, pages 121–126, 2007.
- [10] Ruiqing Sun. Randomized approximation schemes for minimizing the weighted makespan on identical parallel machines. *Journal of Combinatorial Optimization*, 47(3):34, 2024.

Online Weighted Flow Time with Equal-Size Jobs

Alexander Lindermayr *

Morten Weber (Speaker) †

1 Introduction

We study the following fundamental online scheduling problem: a set of jobs $J = \{1, 2, \dots, n\}$ arrives online over time and must be scheduled on a single machine; pre-emption is allowed, that is, jobs can be interrupted and resumed later. Jobs have individual integral release times r_j , processing times p_j , and weights w_j . The goal is to construct a schedule online that minimizes the sum of weighted flow times, where the flow time F_j of job j is the time between its release time and its completion time C_j , that is, $F_j := C_j - r_j$. We study this problem in the special case where all jobs $j \in J$ have equal processing time $p_j = p$. Even in this restricted regime, the online difficulty remains: arrivals are unknown and the weights alone are the challenge. The goal is to design algorithms and to prove bounds on the competitive ratio, which give a quality guarantee. An algorithm is called c -competitive for a constant c if the objective value (weighted flow time) is at most c times the optimal offline objective value for every feasible instance; the competitive ratio is the smallest such c .

If all weights are equal, then the classic Shortest Remaining Processing Time (SRPT) algorithm is optimal [7]. For the general case, the best-known result is a $O(\min\{\log W, \log P, \log D\})$ -competitive algorithm by Azar and Touitou [2]. Earlier, Chekuri, Khanna, and Zhu [6] presented a $O(\log^2 P)$ -competitive algorithm. The parameters W , P , and D thereby denote the ratios between the largest and smallest job weights, processing times, and densities which is $d_j = w_j/p_j$. For equal processing times, the mentioned algorithms provide bounds of 12 [2] and 136 [6]. Furthermore, Bansal and Chan [3] showed that no deterministic online algorithm can be $O(1)$ -competitive in the general case: It exists a lower bound of $\Omega(\min\{\sqrt{\frac{\log W}{\log \log W}}, \sqrt{\frac{\log \log P}{\log \log \log P}}\})$ in the asymptotic regime. Hence, this result cannot be applied for equal-size jobs with $P = 1$.

2 Our Results

We give an improved upper bound for the equal-size regime.

Theorem 1 *There exists a 4-competitive online algorithm for minimizing weighted flow time with equal-size jobs on a single machine. In particular, it is 2-competitive when all weights are powers of 2.*

*alexander.lindermayr@tu-berlin.de. Institut für Mathematik, Technische Universität Berlin, Germany.

†moweber@uni-bremen.de. Faculty of Mathematics and Computer Science, University of Bremen, Germany.

To achieve this result, we use the same algorithm of Azar and Touitou [2] and simplify it for the special case of equal-size jobs. Our analysis uses the common *volume cover* framework, which was also used in [2], but also in other results in online flow time scheduling [1, 4, 7]. We crystallize its main ingredients that are required for the equal-size case and achieve better constants by exploiting the equal job size in a more careful inductive analysis compared to [2]. We also show that our bound on the algorithm's competitive ratio is tight.

Next, we present the first lower bound on the competitive ratio of any deterministic algorithm for minimizing weighted flow time with equal-size jobs.

Theorem 2 *For minimizing weighted flow time on a single machine with equal-size jobs, every deterministic online algorithm has a competitive ratio of at least $\frac{1+\sqrt{37}}{6} > 1.1804$. Moreover, the competitive ratio is at least $7/6 > 1.66$ even if all weights are integer powers of 2.*

A common technique for proving lower bounds for (weighted) flow time is to argue that, at some time, an online algorithm has accumulated significantly more *unfinished weight* than the optimum; this motivates *local competitiveness*. An algorithm is locally c -competitive if at any time t the total weight of unfinished jobs in its schedule is at most c times the corresponding unfinished weight in an optimal schedule; the *local competitive ratio* is the smallest such c . Given a time when the algorithm is locally “bad”, one can often amplify the gap by releasing many *very short*, high-weight jobs at that time, forcing both schedules to focus on them; repeating this yields strong lower bounds on the local competitive ratio. Since weighted flow time can be expressed as the time integral of unfinished weight, local competitiveness implies (global) competitiveness. Conversely, lower bounds on the local competitive ratio can often be turned into lower bounds on the competitive ratio via a similar short-job streaming argument [5].

For equal-size jobs, this short-job streaming amplification is not available, which makes lower bounds much more challenging. Moreover, it is not clear whether the local and (global) competitive ratios remain equivalent in this regime. We prove a strong lower bound on the local competitive ratio of any deterministic algorithm.

Theorem 3 *For minimizing weighted flow time on a single machine with equal-size jobs, every deterministic online algorithm has a local competitive ratio of at least 2 even if all weights are integer powers of 2.*

Our bounds in Theorem 1 also hold for the local competitive ratio of our algorithm. Hence, Theorem 3 shows that our algorithm is a best-possible online algorithm for equal-size jobs in terms of local competitive ratio for the power-of-2 weights case.

3 Conclusion

We initiate the study of online weighted flow time minimization for equal-size jobs. Our results give an essentially tight picture for the local competitive ratio in this special regime.

Several natural questions remain open, e.g., if the analysis for local and global competitive ratios can be separated for equal-size jobs. Another question is whether the

remaining gap between our deterministic upper and lower bounds on the (global) competitive ratio can be closed. It is also possible to consider different priority rules. An intuitive idea is to use the job with maximal weight-to-remaining-processing-time-ratio. This is done in the *Weighted Shortest Remaining Processing Time* (WSRPT), but it also has a competitive ratio of at least $3/2$ for equal processing times.

Acknowledgments. The authors would like to thank Nicole Megow for insightful discussions about this research and valuable comments on the writeup.

References

- [1] Yossi Azar, Stefano Leonardi, and Noam Touitou. Flow time scheduling with uncertain processing time. In *STOC*, pages 1070–1080. ACM, 2021.
- [2] Yossi Azar and Noam Touitou. Improved online algorithm for weighted flow time. In *FOCS*, pages 427–437. IEEE Computer Society, 2018.
- [3] Nikhil Bansal and Ho-Leung Chan. Weighted flow time does not admit $o(1)$ -competitive algorithms. In *SODA*, pages 1238–1244. SIAM, 2009.
- [4] Nikhil Bansal and Kedar Dhamdhere. Minimizing weighted flow time. *ACM Trans. Algorithms*, 3(4):39, 2007.
- [5] Luca Becchetti, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Kirk Pruhs. Online weighted flow time and deadline scheduling. *J. Discrete Algorithms*, 4(3):339–352, 2006.
- [6] Chandra Chekuri, Sanjeev Khanna, and An Zhu. Algorithms for minimizing weighted flow time. In *STOC*, pages 84–93. ACM, 2001.
- [7] Linus Schrage. Letter to the editor - A proof of the optimality of the shortest remaining processing time discipline. *Oper. Res.*, 16(3):687–690, 1968.

Scheduling with Testing: Competitive Algorithms for Minimizing the Total Weighted Completion Time in the Adversarial Model

Felix Buld (Speaker) * Andreas S. Schulz *

1 Introduction

We examine an adversarial scheduling with testing problem, which extends the model introduced in [1, 5] to job-dependent weights. We are given n jobs to be scheduled on a single machine or on identical parallel machines. Each job j is equipped with a known weight $w_j > 0$ and admits two possible processing options. One is to process the job untested in a safe mode, requiring u_j time. The other one is to test it, taking $t_j > 0$ time, and reveal its actual processing time $p_j \in [0, u_j]$, which is initially unknown. The job can be processed later, taking p_j time. Thus, it is allowed to interrupt between testing and processing a job in this mode. The objective is to minimize the total weighted completion time, $\sum_j w_j C_j$. A machine can test or process at most one job at a time, i.e., testing and processing require the same capacity. The aim is to design deterministic and randomized semi-online algorithms whose performances are competitive with that of optimal algorithms using full information.

The unweighted version of this problem, aiming to minimize the total completion time, was introduced with unit testing times [5]. Subsequent work extended the model to arbitrary testing times [1, 8] and to identical parallel machines [7, 9]. In other related variants, obligatory testing [3] was examined, as were processing time oracles [4], in which the processing time of a job is p_j , whether or not it is tested. Two of these papers [3, 9] were presented at MAPSP 2024.

In this research [2], we establish the first theoretical results in this model to minimize the total *weighted* completion time, on a single machine, as well as on parallel machines. The major contributions lie in extending various deterministic and randomized algorithms to the weighted setting, and in matching, or even improving, the known performance guarantees in the unweighted case.

*felix.buld@tum.de, andreas.s.schulz@tum.de.

Technical University of Munich, Arcisstr. 21, 80333 Munich, Germany.

2 Results and Techniques

In this talk, we will present results from [2] and additional results derived subsequently. In Table 1, the currently best-known competitive ratios for minimizing $\sum_j C_j$ established by [7, 8, 9] are depicted. We extend the algorithms by [8, 9] to incorporate job-dependent weights, replicate three of the best-known bounds in the weighted setting, and propose a new randomized algorithm that enhances the previous performance guarantee for parallel-machine scheduling.

Objective	1 / P	Deterministic Algorithms	Randomized Algorithms	Reference
$\sum C_j$	1	2.32	2.16	[8]
	P	3.24 (2.78)	2.84	[7] ([9])
$\sum w_j C_j$	1	2.32	2.16	This Research [2]
	P	2.78	2.52	

Table 1: Overview of currently best competitive ratios for various models. The abbreviations 1 and P denote single and parallel machine environments, as usual.

The algorithms for this model, established and analyzed in literature, typically work in three steps: a testing decision for each job made in advance, a priority rule for selecting the next operation to be scheduled, and a list-scheduling framework to embed this in a parallel-machine setting. For the first step, threshold-based strategies are commonly used. A job is tested if and only if its ratio u_j/t_j is at least a fixed threshold α chosen in advance for all jobs. The *Golden Ratio*, $\varphi := (1+\sqrt{5})/2 \approx 1.62$, is frequently used as a threshold, being the optimal choice for single-job instances [1, 5].

2.1 Priority Rules for Single-Machine Scheduling

For the second step, we keep a priority list containing currently available operations. These are sorted by priority values, and in each iteration, we schedule the operation with the smallest priority value, proportionally weighted, next. The priority values we propose are $\beta t_j/w_j$ for the testing part, u_j/w_j if a job is processed without testing, and $(t_j+p_j)/w_j$ or alternatively p_j/w_j for the processing of a job j that was tested, where β is a delay factor to incorporate the uncertainty of unknown processing times to a certain extent. These naturally extend the priority rules used by [8] and [1] to job-dependent weights. The single-machine algorithms that consist of the threshold-based testing decision above, together with the outlined unweighted priority rules, were introduced as (α, β) -PCP algorithm [8] and (α, β) -SORT algorithm [1].

Theorem 1 *The competitive ratio of the WEIGHTED (α, β) -PCP algorithm with parameters $\alpha = \varphi$ and $\beta = (\varphi + \sqrt{\varphi(\varphi+4)})/2$ is at most $\beta \leq 2.32$. A randomized version thereof has an expected competitive ratio of at most 2.16.*

The analysis used in [2, 8] bounds the sum of mutual contributions of two jobs j and k against the ones in an optimum by a suitable case distinction. The parameters α and β are optimized afterwards. For the WEIGHTED (α, β) -SORT, this analysis framework yields slightly weaker upper bounds. Both algorithms cannot be better than 2 for any combination of $\alpha \geq 1$ and $\beta \geq 0$, even in the unit testing time case [1, 2].

2.2 From Single to Identical Parallel Machines

The parallel-machine algorithms considered in [2] combine the previous ideas for the single-machine case with list scheduling. One keeps the same order of operations as on a single machine, and schedules the next available operation once a machine idles, while respecting that a tested processing operation is only available after the completion of a test of the same job, similar to [7, 9] for minimizing the total completion time. The bounds are derived by comparing the performances of the proposed algorithms against the classical lower bound of [6] for $P \parallel \sum w_j C_j$.

Theorem 2 *The WEIGHTED (α, β) -PCP algorithm with parameters as in Theorem 1 combined with list scheduling has a competitive ratio of at most $2.78 - 0.45/m$. A randomized version of it has an expected competitive ratio of at most $2.52 - 0.35/m$.*

References

- [1] S. ALBERS AND A. ECKL (2021). *Explorable uncertainty in scheduling with non-uniform testing times*. WAOA 2020. LNCS, vol. 12806, pp. 127–142.
- [2] F. BULD AND A.S. SCHULZ (2025). *Scheduling with testing: Competitive algorithms for minimizing the total weighted completion time in the adversarial model*. IJTCS-FAW 2025. LNCS, vol. 15828, pp. 64–77.
- [3] K. DOGEAS, T. ERLEBACH AND Y.-C. LIANG (2024). *Scheduling with obligatory tests*. ESA 2024. LIPIcs, vol. 308, pp. 48:1–48:14.
- [4] F. DUFOSSÉ, C. DÜRR, N. NADAL, D. TRYSTRAM AND Ó.C. VÁSQUEZ (2022). *Scheduling with a processing time oracle*. Applied Mathematical Modelling, vol. 104, pp. 701–720.
- [5] C. DÜRR, T. ERLEBACH, N. MEGOW AND J. MEISSNER (2020). *An adversarial model for scheduling with testing*. Algorithmica, vol. 82, no. 12, pp. 3630–3675.
- [6] W.L. EASTMAN, S. EVEN, I.M. ISAACS (1964). *Bounds for the optimal scheduling of n jobs on m processors*. Management Science, vol. 11, no. 2, pp. 268–279.

- [7] M. GONG, Z.Z. CHEN AND K. HAYASHI (2024). *Approximation algorithms for multiprocessor scheduling with testing to minimize the total job completion time*. *Algorithmica*, vol. 86, no. 5, pp. 1400–1427.
- [8] A.H.H. LIU, F.H. LIU, P.W. WONG AND X.O. ZHANG (2023). *The power of amortization on scheduling with explorable uncertainty*. WAOA 2023. LNCS, vol. 14297, pp. 90–103.
- [9] A.H.H. LIU, F.H. LIU, P.W. WONG AND X.O. ZHANG (2024). *Amortization helps for scheduling with explorable uncertainty, even on parallel machines*. Proceedings of MAPSP 2024, pp. 259–262.

Interval Scheduling Games

Vipin Ravindran Vijayalakshmi* Marc Schröder † Tami Tamir (Speaker)‡

1 Introduction and Model

Scheduling problems and game-theory are a fruitful and well studied combination. The machine scheduling problem that we consider here is motivated by the beam selection problem on a base station. In 5G base stations, antennas use beamforming to focus signals toward specific users. Since a base station can only activate a single beam each time, the system must strategically allocate time intervals to different beams, to maximize throughput. We model this as a generalization of the classic interval scheduling problem: at any given time, the machine is set to a specific “color” (representing a beam direction), allowing all jobs of that color to be processed concurrently. A job is successfully completed only if the machine maintains its required color throughout the job’s entire processing interval.

In the game-theoretic variant of this model, each player controls jobs of a specific color, and strategically select their processing intervals to compete for machine access.

Formally, an interval scheduling game is given by a tuple $\langle \mathcal{J}, \mathcal{C}, T, (p_j)_{j \in \mathcal{J}}, (w_j)_{j \in \mathcal{J}} \rangle$, containing a set of jobs $\mathcal{J} = \{1, \dots, n\}$, a set of colors \mathcal{C} with $|\mathcal{C}| = c$ and a time interval $[0, T)$, where every job $j \in \mathcal{J}$ has a color $c_j \in \mathcal{C}$, a length $0 \leq p_j \leq T$ and a weight $w_j \geq 0$.

For $i \in \mathcal{C}$, denote by S_i the jobs having color i . We assume that all the jobs of S_i are controlled by one player. A strategy, σ_i , of player i assigns a processing interval $[s_j, f_j) \subseteq [0, T)$ with $f_j - s_j = p_j$ to each job j with $c_j = i$. A profile $\sigma = (\sigma_i)_{i \in \mathcal{C}}$ assigns a strategy to each player.

Given a strategy profile σ , the machine faces the following scheduling problem. We say that job j is *covered* in a given schedule if the machine is configured to process color c_j during $[s_j, f_j)$. Note that if the machine is configured to a specific color, the machine can service an unlimited number of jobs from that color simultaneously. Let $\mathcal{S} \subseteq \mathcal{J}$ be the set of covered jobs. The goal of the machine is to maximize $\sum_{j \in \mathcal{S}} w_j$. Formally, the output for the machine scheduling problem is a configuration for the machine, i.e., a partition of the interval $[0, T)$ to intervals $\{[0, t_1), \dots, [t_{m-1}, t_m = T)\}$ and a mapping $\gamma : \{1, \dots, m\} \rightarrow \mathcal{C}$, such that for every $1 \leq k \leq m$, the machine is configured to process jobs of color $\gamma(k)$ during $[t_{k-1}, t_k)$.

Given strategy profile σ , the machine solves its optimization problem. Thus, determining for each job whether it is covered or not. The utility of each player $i \in \mathcal{C}$, denoted by $u_i(\sigma)$, is given by the sum of weights of covered jobs j with $c_j = i$.

*vipin.rv@oms.rwth-aachen.de. Viavi Solutions, Berlin, Germany

†m.schroeder@maastrichtuniversity.nl. School of Business and Economics, Maastricht University, Netherlands.

‡tami@runi.ac.il. School of Computer Science. Reichman University. Israel.

A strategy profile σ is called a *Nash equilibrium* (NE) if for all $i \in \mathcal{C}$, $u_i(\sigma_i, \sigma_{-i}) \geq u_i(\sigma'_i, \sigma_{-i})$ for all σ'_i .

We consider three problems arising in this model: first, the underlying scheduling problem of the machine, determining the jobs to be covered. Second, optimizing job assignments to maximize the total weight of covered jobs; and third, evaluating the game's equilibrium properties. This includes investigating the existence and computation of pure Nash equilibria, the convergence of best-response dynamics, and the resulting equilibrium inefficiency, often referred to as the Price of Anarchy.

The machine scheduling problem we study is a generalization of the classical interval scheduling problem [2, 4, 5]. There are many game-theoretic models of job scheduling problems, in particular games in which each job selfishly chooses a machine or a processing interval so as to minimize a certain objective [6, 3]. Other related games consider *selfish packing* [1, 8]. We note that in the above packing games, a profile does not specify the location of a packed item in the knapsack, unlike our game, in which the specific interval in which a job is processed within the interval $[0, T)$ plays a crucial role.

Example: Consider the following game with $T = 4$ and $c = 2$. Player 1 controls the set S_1 , which includes two jobs, where $p_1 = 4$ and $p_2 = 1$, both having weight 2. Player 2 controls the set S_2 , which includes a single job of length $p_3 = 1$ and weight 3.

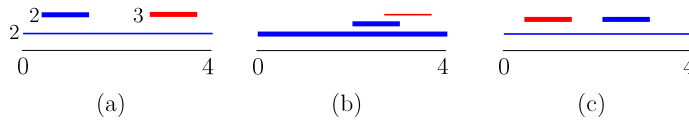


Figure 1: A game G with $c = 2$ that has no NE. Intervals are labeled by their weights. Bold intervals are covered by the machine.

If the two unit jobs are placed in disjoint intervals (See Fig. 1(a)) then the machine will process them. The utilities of the players are $(2, 3)$. This schedule is the social optimum, however, it is not a NE. If the two unit jobs are placed in overlapping intervals (Fig. 1(b)) then the machine will cover only jobs of S_1 . The utilities of the players are $(4, 0)$. Therefore, the game has no NE: Player 2 will place its unit job such that it does not overlap with the unit-job of player 1, and player 1 will place its unit-job such that it overlaps with job 3. Since $T = 4$, player 2 always has a valid move (Fig. 1(c)).

2 Our Results

We first provide an optimal algorithm for the machine. Specifically, given a strategy profile of the players, we show that the machine scheduling problem can be solved in polynomial time using dynamic programming. Then, we show that computing the socially optimal solution for the players can be done in polynomial time if the number of colors is constant, but is NP-hard for an arbitrary number of colors.

We next analyze Nash equilibria of the interval scheduling game. As demonstrated in the example above, in general, even with two players, a (pure) Nash equilibrium need not exist. However, for the game-theoretic variant of the classic interval scheduling problem, where every player controls a single job, we prove that a (pure) Nash equilibrium always exists and the price of stability is 1. For this class of games, we present an algorithm for

computing an NE, and we provide tight analysis of the inefficiency of Nash equilibria. We show that for $n \leq 5$, the price of anarchy (PoA) is at most 2, and for $n \geq 6$, the PoA is at most $(n - 1)/2$. Moreover, these bounds are tight. For the restricted class of interval scheduling instances with processing time proportional to the weight, we provide a significantly lower upper bound of 3 to the PoA.

For general games, where there may be multiple jobs of the same color, we prove that even with only two players, computing the best-response of a player, as well as deciding whether an instance has a Nash equilibrium, is NP-hard. We present an upper bound of c for the PoA, and show that even the best NE can have poor quality, by proving that the price of stability of this class is at least $\frac{c}{2} - \epsilon'$ for any $\epsilon' > 0$.

A second class of interval scheduling games with a guaranteed Nash equilibrium are games with unit processing times. For this class, the price of anarchy is significantly lower. Specifically, we provide a tight bound of $\min\{3 - 2/c, 3 - 2/\lfloor T \rfloor\}$.

Lastly, we define a natural extension of the game, in which jobs may be associated with different release times and due dates, and show that even instances with unit processing time might lack equilibria.

For the full version of our work, we refer to [7].

References

- [1] V. Bilò. On the packing of selfish items. In *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*, 2006.
- [2] K. I. Bouzina and H. Emmons. Interval scheduling on identical machines. *Journal of Global Optimization*, 9:379–393, 1996.
- [3] R. Cole, J. Correa, V. Gkatzelis, V. Mirrokni, and N. Olver. Decentralized utilitarian mechanisms for scheduling games. *Games and Economic Behavior*, 92:306–326, 2015.
- [4] S. Irani and K. R. Pruhs. Algorithmic problems in power management. *SIGACT News*, 36(2):63–76, 2005.
- [5] A. W. J. Kolen, J. K. Lenstra, C. H. Papadimitriou, and F. Spieksma. Interval scheduling: A survey. *Naval Research Logistics (NRL)*, 54(5):530–543, 2007.
- [6] A. Herzel, M. Hopf, and C. Thielen. Multistage interval scheduling games. *Journal of Scheduling*, 22:359–377, 2019.
- [7] V. R. Vijayalakshmi, M. Schröder, and T. Tamir. Interval scheduling games. <https://arxiv.org/abs/2601.15148>, 2026.
- [8] C. Wang and G. Zhang. A best cost-sharing rule for selfish bin packing. <https://arxiv.org/abs/2204.09202>, 2022.

Randomized Rounding over Dynamic Programs

Étienne Bamas ^{*} Shi Li [†] Lars Rohwedder (Speaker) [‡]

1 Introduction

Dynamic Programming (DP) is a fundamental algorithmic principle that applies to problems that can be broken into smaller subproblems recursively, such that only a bounded number of *different* subproblems can occur throughout the entire recursion tree. Then one can solve the subproblems iteratively, which is the core idea of dynamic programming. Many textbook problems are solved in this way. Another important approach in algorithm design is the concept of *Randomized Rounding*, which is widely used to transform a fractional solution to a linear programming (LP) relaxation into an actual solution. In this paper, we combine these two concepts into one framework. With this framework at hand, a great number of notorious problems in approximation algorithms can be solved by writing recurrences, which are sometimes as easy as textbook dynamic programs.

We call the type of DP problems we consider *Additive-DPs*, which we define in the following. Since we will later apply randomized rounding, we linearize our problems, namely, we assume that all solutions are encoded as non-negative integer vectors $x \in \mathbb{Z}_{\geq 0}^d$ and we want to minimize (or maximize) some linear objective $c^\top x$ for some given $c \in \mathbb{R}^d$. There is a bounded family of subproblems \mathcal{I} , including the root problem I° itself. Between the subproblems, there is a partial order \prec .

Each $I \in \mathcal{I}$ is associated with some set of feasible solutions $S(I) \subseteq \mathbb{Z}_{\geq 0}^d$. There is a set of subproblems $\mathcal{I}^{\text{base}} \subseteq \mathcal{I}$ that we call base problems. For every base problem $I \in \mathcal{I}^{\text{base}}$, we have either $S(I) = \{x^{(I)}\}$ for some given vector $x^{(I)} \in \mathbb{Z}_{\geq 0}^d$ or $S(I) = \emptyset$. The latter indicates infeasibility. For every non-base problem $I \in \mathcal{I} \setminus \mathcal{I}^{\text{base}}$ the solution set $S(I)$ is not given explicitly, but instead defined recursively as follows. We can branch on some decision $C \in \{1, 2, \dots, k_I\}$ for some $k_I \in \mathbb{Z}_{> 0}$. Based on I and C we can decompose I into a fixed part $x^{(I,C)} \in \mathbb{Z}_{\geq 0}^d$ and a recursive part corresponding to $\ell(I, C) \geq 1$ subproblems $\Lambda_1(I, C), \dots, \Lambda_{\ell(I,C)}(I, C) \in \mathcal{I}$, where $\Lambda_i(I, C) \prec I$ for all $i \in [\ell(I, C)]$. The solution set for I is

$$S(I) := \bigcup_{C=1}^{k_I} \{x^{(I,C)}\} \oplus S(\Lambda_1(I, C)) \oplus \dots \oplus S(\Lambda_{\ell(I,C)}(I, C)) ,$$

^{*}etienne.bamas@epfl.ch. Mathematics Institute, EPFL, Switzerland.

[†]shili@nju.edu.cn. Department of Computer Science, Nanjing University, China.

[‡]rohwedder@sdu.dk. IMADA. SDU. Denmark.

where \oplus is the sumset operator, that is, $A \oplus B = \{a + b : a \in A, b \in B\}$.

A standard dynamic program to find $x \in S(I^\circ)$ minimizing $c^\top x$ would work as follows: Starting with the base cases and then following the order of \prec it would compute for each subproblem $I \in \mathcal{I}$ a solution $z(I) \in S(I)$ minimizing $c^\top z(I)$. Here, solutions to non-base cases are derived using the recurrence above and the fact that we can assume that the decomposed solutions are minimizers of their respective subproblems.

A solution may directly or indirectly contain several solutions to the same subproblem. In a dynamic programming algorithm as above, one would always obtain consistent solutions to these same subproblems. We emphasize that in our semantics, we do not require this to be the case. If for example $S(I)$ contains $S(I') \oplus S(I')$ then clearly $2x \in S(I)$ for all $x \in S(I')$, but we also have $x + x' \in S(I)$ for $x, x' \in S(I')$ with $x \neq x'$. This is motivated by the fact that we will later add more constraints on top of the Additive-DP problem, in which case it is not clear anymore that the optimum solution should always take the same solution for many occurrences of the same subproblem.

Consider now that, in addition to the recursive solution structure, we need to satisfy highly non-decomposable constraints. Namely, assume that we have packing constraints $Ax \leq (1, \dots, 1)^\top$ for a given matrix $A \in [0, 1]^{m \times d}$. We assume there is an optimal solution that satisfies these constraints. We are looking for a feasible solution to the Additive-DP problem, which is an α -approximation with respect to the extra packing constraints. More precisely, we want a solution x that satisfies $Ax \leq (\alpha, \dots, \alpha)^\top$.

The main restriction of our model is that solutions are computed using one large branching and the combination of subproblems is via vector addition. In general dynamic programs, one could imagine also more complicated circuits that compute new solutions from previous ones. Still, many textbook examples of dynamic programming, such as the Knapsack problem, Shortest Path, or Longest Common Subsequence, can be cast in the framework above. For instance, Shortest Path can be phrased as Additive-DP as follows.

Without loss of generality, we assume the input graph $G = (V, E)$ is a DAG. A general graph can be converted to a DAG by creating n layers of vertices, without changing the problem. Our goal is to find the shortest path from s to t in G , w.r.t. the edge costs $c \in \mathbb{R}_{\geq 0}^E$. We have one subproblem $I(v)$ for each vertex $v \in V$, with $I^\circ := I(s)$ being the root problem. The solutions are vectors in $\{0, 1\}^E$, with $S(I(v)) := \{\mathbf{1}_p : p \text{ is a path from } v \text{ to } t\}$ for each $v \in V$, where $\mathbf{1}_p$ is the indicator vector for the edges in p . The relation \prec is defined by the topological order of the vertices in G . The solutions can be defined recursively as follows. For the subproblem $I(t)$, we have $S(I(t)) = \{\mathbf{0}\}$. For every vertex $v \neq t$, we have

$$S(I(v)) = \bigcup_{(v,u) \in \delta^+(v)} S(I(u)) \oplus \{\mathbf{1}_{(v,u)}\}$$

Finding the shortest s - t path is equivalent to finding $x \in S(I(s))$ minimizing $c^\top x$. On top of this, we can define packing constraints on the s - t path we choose.

Inspired by techniques from network design problems, we present reductions that simplify the structure of the dynamic programs. We then construct a non-trivial LP relaxation for optimizing over the simplified DP structure while maintaining additional packing constraints. Before describing our result formally, we need to introduce the notion of a solution size, which will affect the running time and which we denote by $\text{size}_I(x)$ for every $I \in \mathcal{I}$ and $x \in S(I)$. Intuitively, we can define a recursion tree for $x \in S(I)$ showing how x is obtained, and $\text{size}_I(x)$ is the minimum possible number of vertices over any such tree. For a base problem I , we have $\text{size}_I(x^{(I)}) = 1$. For any non-base problem I and $x \in S(I)$, $\text{size}_I(x)$ is defined as the minimum of $1 + \text{size}_{\Lambda_1(I,C)}(z^{(1)}) + \text{size}_{\Lambda_2(I,C)}(z^{(2)}) + \dots + \text{size}_{\Lambda_{\ell(I,C)}(I,C)}(z^{(\ell(I,C))})$, over all $C \in [k_I]$, $z^{(1)} \in S(\Lambda_1(I,C))$, $z^{(2)} \in S(\Lambda_2(I,C))$, \dots , $z^{(\ell(I,C))} \in S(\Lambda_{\ell(I,C)}(I,C))$ satisfying $x^{(I,C)} + z^{(1)} + z^{(2)} + \dots + z^{(\ell(I,C))} = x$.

Theorem 1 *Suppose we are given an Additive-DP instance. Let $\Phi := |\mathcal{I}_{\text{base}}| + \sum_{I \in \mathcal{I} \setminus \mathcal{I}_{\text{base}}, C \in [k_I]} (\ell(I,C) + 1)$. Suppose we are given a promised upper bound Δ on $\text{size}_{I^\circ}(x^*)$ for the optimum solution $x^* \in S(I^\circ)$.*

Let $\varepsilon > 0$. For some $\alpha = O\left(\frac{\Delta^\varepsilon}{\varepsilon^2} \cdot \log m\right)$, we can in time $(\Delta\Phi)^{O(1/\varepsilon)} \text{poly}(\Phi, \Delta, m, d)$ find a solution $x \in S(I)$ with $Ax \leq \alpha \cdot \mathbf{1}$ and $c^\top x \leq \text{opt}$, where opt is the optimum cost of the Additive-DP instance. When $c = \mathbf{0}$ and thus our goal is to find any $x \in S(I^\circ)$ satisfying the packing constraints, we can make $\alpha = O\left(\frac{\Delta^\varepsilon}{\varepsilon} \cdot \log m\right)$.

In many applications, all the parameters are polynomially bounded in the input size n . Then our result implies a polylogarithmic approximation in quasi-polynomial time by setting $\varepsilon = \log \log(n) / \log(n)$, or a $O(n^\varepsilon)$ -approximation in $n^{O(1/\varepsilon)}$ -time by setting ε as a constant. While our main theorem is on packing constraints and DP structure, it can be applied indirectly to many other structures as well, including certain covering problems, matching structures and more.

Non-Clairvoyant Scheduling with Progress Bars*

Ziyad Benomar[†] Romain Cosson[‡] Alexander Lindermayr (speaker)[§]
 Jens Schöter[¶]

1 Introduction

Progress bars are a ubiquitous form of *partial information*: software downloads display a percentage, machine-learning training shows an epoch counter, and manufacturing dashboards report milestone completions. Such signals are helpful, but they are often *coarse* (only a few milestones) and sometimes *misaligned* (e.g., the “first 90%” appears fast while the last steps drag on).

We ask a basic algorithmic question: *how much can progress bars help online scheduling?* Formally, we study preemptive single-machine scheduling with unknown processing times, and we aim to minimize the sum of job completion times $\sum_{j=1}^n C_j$. Without any feedback, the classical non-clairvoyant model yields an optimal competitive ratio of 2 (achieved by Round-Robin) [MPT94]. In contrast, with perfect size information, an optimal solution is achievable by ordering jobs by their processing times [Smi56]. Progress bars naturally interpolate between these extremes by providing *dynamic* information that improves as a job runs.

We focus on two settings and state only the main formal guarantees: (i) an *adversarial* model where the progress signal may be inaccurate, and (ii) a *stochastic* model where progress events are generated at random while a job is processed. (Additional results for multi-level signals and extensions are omitted here.)

2 Model

Each job j has an unknown processing time $p_j > 0$ and can be preempted at any time. After receiving $e \in [0, p_j]$ units of processing, job j displays a progress value

$$X_j(e) = \varphi_j\left(\frac{e}{p_j}\right) \in [0, 1],$$

*The conference version of this abstract appeared in the proceedings of the Thirty-Ninth Annual Conference on Neural Information Processing Systems, NeurIPS 2025 [Ben+25].

[†]CREST, ENSAE, Ecole Polytechnique, Fairplay joint team, Palaiseau, France.

[‡]Courant Institute of Mathematical Sciences, New York University, United States.

[§]Technische Universität Berlin, Germany.

[¶]Centrum Wiskunde & Informatica (CWI), The Netherlands.

where the progress-bar function $\varphi_j : [0, 1] \rightarrow [0, 1]$ is nondecreasing with $\varphi_j(0) = 0$ and $\varphi_j(1) = 1$. The displayed value $X_j(e)$ can be interpreted as feedback about the unknown ratio e/p_j .

We mostly discuss *step-function* progress bars of granularity g : the display takes values from a fixed sequence $0 = \alpha^{(0)} < \alpha^{(1)} < \dots < \alpha^{(g)} < \alpha^{(g+1)} = 1$. For intuition, the simplest case is $g = 1$: there is a single announced intermediate level $\alpha \in (0, 1]$. Job j emits a signal when its *true* processed fraction reaches some $\beta_j \in [0, 1]$ (possibly $\beta_j \neq \alpha$).

We compare an online algorithm ALG to the clairvoyant optimum OPT using competitive analysis. When the announced signal is accurate (truthful case), we also consider *consistency* with respect to OPT, and in the worst case we consider *robustness* guarantees.

3 Results

3.1 Adversarial progress bars

We first consider the granularity-1 model above (one possible signal per job). A natural baseline is to run Round-Robin until some job emits its signal, and then give that job *preferential execution* (run it alone until completion), before returning to Round-Robin.

Theorem 1 (Truthful signals) *Assume each job emits its signal exactly after being processed for a fraction α of its processing time, i.e., $\beta_j = \alpha$ for all jobs. Running Round-Robin and granting preferential execution upon a signal achieves competitive ratio $1 + \alpha$, which is optimal even for randomized algorithms.*

When signals can be inaccurate ($\beta_j \neq \alpha$), blindly following them can be non-robust. Our main single-signal algorithm interpolates between exploration (via SETF, which equalizes elapsed processing) and exploitation after a signal, controlled by a parameter $\rho \in (0, 1]$.

Theorem 2 (Untrusted signals: consistency, robustness, smoothness)

Let $\rho \in (0, 1]$. There exists a deterministic single-signal algorithm that is $(1 + \alpha)$ -consistent and $(1 + \frac{1}{\rho\alpha})$ -robust. Moreover, if $\rho \in (0, 1)$, then for signal fractions $(\beta_i)_{i=1}^n$ it satisfies the smooth bound

$$\text{ALG} \leq (1 + \alpha)\text{OPT} + \frac{2n}{\rho(1 - \rho)\alpha^2} \sum_{i=1}^n |\beta_i - \alpha| p_i .$$

3.2 Stochastic progress bars

We also study a more optimistic model where progress information is generated stochastically. Each job has granularity g and emits *progress events* according to a Poisson process of rate g while it is being processed. Equivalently, the number

of displayed milestones reached during an interval is random but proportional to the processing spent. This creates a close connection to (mortal) stochastic bandits [LS20].

A simple *Repeated Explore-Then-Commit* strategy runs Round-Robin on the currently unfinished jobs until some job reaches a displayed threshold, then commits to that job until completion, and repeats. With a tuned threshold $\kappa = \Theta(g^{2/3})$, we obtain an optimal asymptotic rate in g .

Theorem 3 (Upper bound) *Let $g \geq 12$ and set $\kappa = \lceil (g/2)^{2/3} \rceil + 1$. Repeated Explore-Then-Commit with threshold $\kappa/(g + 1)$ has expected competitive ratio at most $1 + (12/g)^{1/3}$.*

Theorem 4 (Lower bound) *The expected competitive ratio of any scheduling algorithm with a Poisson stochastic progress bar of granularity $g \in \mathbb{N}$ is at least $1 + \frac{1}{36}g^{-1/3}$.*

Even coarse or imperfect progress feedback can provably improve non-clairvoyant scheduling: in adversarial settings we obtain tight interpolation and strong consistency/robustness guarantees, and in stochastic settings the ratio approaches 1 as the granularity g grows.

References

- [Ben+25] Ziyad Benomar, Romain Cosson, Alexander Lindermayr, and Jens Schlöter. “Non-Clairvoyant Scheduling with Progress Bars”. In: *The Thirty-Ninth Annual Conference on Neural Information Processing Systems, NeurIPS 2025*. 2025.
- [LS20] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.
- [MPT94] Rajeev Motwani, Steven J. Phillips, and Eric Torng. “Non-Clairvoyant Scheduling”. In: *Theor. Comput. Sci.* 130.1 (1994), pp. 17–47.
- [Smi56] Wayne E Smith. “Various optimizers for single-stage production”. In: *Naval Research Logistics Quarterly* 3.1-2 (1956), pp. 59–66.

A Little Clairvoyance Is All You Need*

Anupam Gupta[†] Haim Kaplan[‡] Alexander Lindermayr[§]
 Jens Schöter (speaker)[¶] Sorrachai Yingchareonthawornchai^{||}

1 Introduction

We study the following fundamental online scheduling problem: Jobs arrive online over time without prior knowledge, and an algorithm can at any time process a job on a single machine, but cannot change its past decisions. Preemption is allowed, that is, an algorithm can stop processing a job and resume it at a later point in time. The goal is to minimize the sum of flow times, where the *flow time* of a job is the time between its release date and completion time.

In the *clairvoyant* setting, where the processing time of a job becomes known at its arrival, a classic result of Schrage [Sch68] proves that always processing the job with the least amount of remaining work is optimal (Shortest-Remaining-Processing-Time rule, SRPT). In particular, SRPT is a 1-competitive online algorithm; we say that an algorithm is α -competitive if its schedule has a total flow time of at most α times the total flow time of an offline optimum solution. The *competitive ratio* of an algorithm is the smallest α such that it is α -competitive.

However, practically relevant in many applications, such as scheduling tasks in computing systems, is the *non-clairvoyant* scenario, where an algorithm has no knowledge about a job's processing time until it completes. In contrast to the well manageable clairvoyant variant, Motwani, Phillips, and Torng [MPT94] showed strong lower bounds of $\Omega(n^{1/3})$ and $\Omega(\log n)$ on the competitive ratio of every deterministic and randomized non-clairvoyant algorithm, respectively, where n is the total number of jobs.

Motivated by these strong lower bounds, non-clairvoyant algorithms have been studied in learning-augmented settings [ALT21; ALT22; Gup+26; Bec+04]: Each job j arrives with a prediction \hat{p}_j on the actual unknown processing time p_j . Recently, Gupta et al. [Gup+26] gave a best-possible $O(\mu)$ -competitive algorithm,

*The conference version [Gup+25] of this abstract will appear in the proceedings of the 66th Annual Symposium on Foundations of Computer Science (FOCS 2025).

[†]New York University

[‡]Tel Aviv University

[§]Technische Universität Berlin

[¶]Centrum Wiskunde & Informatica (CWI), Amsterdam

^{||}Institute for Theoretical Studies, ETH Zürich

where $\mu = \mu_1 \cdot \mu_2$ and $\mu_1 = \max_j \hat{p}_j/p_j$ and $\mu_2 = \max_j p_j/\hat{p}_j$ are the maximum overestimation and underestimation factors.

However, a drawback of the learning-augmented model is that the prediction must be available upon job arrival, when job and scheduler have not yet interacted. Therefore, it is unclear how such predictions could be obtained in many real-world applications. Instead, we study the ϵ -clairvoyant model, which was first proposed by Yingchareonthawornchai and Torng [YT17] and considers a relaxation of non-clairvoyant scheduling without a-priori estimates of job sizes:

ϵ -Clairvoyant Model. Fix some constant $\epsilon \in [0, 1]$. When a job j arrives, we know nothing about its processing time p_j . However, we start to learn the job size as it is being processed: the value p_j is revealed when an ϵ -fraction of the job j remains to be processed.

This model naturally interpolates between clairvoyant and non-clairvoyant settings: 0-clairvoyance corresponds to the non-clairvoyant model of [MPT94], and 1-clairvoyance is the classical (“fully” clairvoyant) model.

2 Our Results

We give an algorithm that achieves a constant competitive ratio for very constant ϵ , proving that *a little clairvoyance is enough* to achieve a constant competitive ratio.

Theorem 1 *For any constant $\epsilon \in (0, 1]$, there is a deterministic ϵ -clairvoyant algorithm that is $\lceil \frac{1}{\epsilon} \rceil$ -competitive for the objective of minimizing the total flow time on a single machine.*

The algorithm is particularly simple and natural, and can be seen as following the paradigm of “optimism in the face of uncertainty”. For each job j , the algorithm maintains an optimistic estimate of its actual size. If the job length p_j has not yet been revealed, the estimate is $\hat{p}_j(t) = e_j(t)/(1 - \epsilon)$, where $e_j(t)$ is the amount of processing j has received until time t , also called the *elapsed time*. Once the length of job j is revealed, we define $\hat{p}_j(t) = p_j$. Our algorithm runs SRPT based on these optimistic estimates, and at any time t processes the job j that minimizes $\hat{p}_j(t) - e_j(t)$; i.e., any job that minimizes the lower bound on its remaining processing time. Note that this algorithm interpolates between two well-known algorithms: If $\epsilon = 1$, then the algorithm is identical to SRPT. On the other hand, if $\epsilon \rightarrow 0$, then the algorithm behaves like *SETF (Shortest Elapsed Time First)* [KP00; Che+04] and executes the jobs with the smallest elapsed times, leading to a round-robin or “water filling” algorithm.

We prove that the competitive ratio of our algorithm is best-possible for deterministic algorithms and asymptotically best-possible for randomized algorithms.

Theorem 2 For all constant $\epsilon \in (0, 1]$, every deterministic ϵ -clairvoyant algorithm has a competitive ratio of at least $\lceil \frac{1}{\epsilon} \rceil$ for the objective of minimizing the total flow time on a single machine. Every randomized ϵ -clairvoyant algorithm has a competitive ratio of $\Omega(\frac{1}{\epsilon})$.

Finally, we consider the special case of *sum of completion times minimization*, where all jobs arrive at time 0, and prove that our algorithm also achieves the best-possible competitive ratio for this special case.

Theorem 3 For all $\epsilon \in [0, 1]$, our algorithm is $(2 - \epsilon)$ -competitive for sum of completion time minimization on a single machine. Moreover, every randomized algorithm in this case has a competitive ratio of at least $2 - \epsilon$.

References

- [ALT21] Yossi Azar, Stefano Leonardi, and Noam Touitou. “Flow time scheduling with uncertain processing time”. In: *STOC*. ACM, 2021, pp. 1070–1080.
- [ALT22] Yossi Azar, Stefano Leonardi, and Noam Touitou. “Distortion-Oblivious Algorithms for Minimizing Flow Time”. In: *SODA*. SIAM, 2022, pp. 252–274.
- [Bec+04] Luca Becchetti, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Kirk Pruhs. “Semi-clairvoyant scheduling”. In: *Theor. Comput. Sci.* 324.2-3 (2004), pp. 325–335.
- [Che+04] Chandra Chekuri, Ashish Goel, Sanjeev Khanna, and Amit Kumar. “Multi-processor scheduling to minimize flow time with epsilon resource augmentation”. In: *STOC*. ACM, 2004, pp. 363–372.
- [Gup+25] Anupam Gupta, Haim Kaplan, Alexander Lindermayr, Jens Schlöter, and Sorrachai Yingchareonthawornchai. “A Little Clairvoyance Is All You Need”. In: *CoRR* abs/2508.17759 (2025).
- [Gup+26] Anupam Gupta, Amit Kumar, Debmalya Panigrahi, and Zhaozi Wang. “An Optimal Online Algorithm for Robust Flow Time Scheduling”. In: *SODA*. SIAM, 2026, pp. 1214–1238.
- [KP00] Bala Kalyanasundaram and Kirk Pruhs. “Speed is as powerful as clairvoyance”. In: *J. ACM* 47.4 (2000), pp. 617–643.
- [MPT94] Rajeev Motwani, Steven J. Phillips, and Eric Torng. “Non-Clairvoyant Scheduling”. In: *Theor. Comput. Sci.* 130.1 (1994), pp. 17–47.
- [Sch68] Linus Schrage. “Letter to the Editor - A Proof of the Optimality of the Shortest Remaining Processing Time Discipline”. In: *Oper. Res.* 16.3 (1968), pp. 687–690.
- [YT17] Sorrachai Yingchareonthawornchai and Eric Torng. “Delayed-Clairvoyant Scheduling”. In: *MAPSP*. 2017, pp. 198–201.

What to Predict for Efficient Scheduling on Multiple Machines?

Evripidis Bampis* Bruno Escoffier* Dimitris Fotakis[†]
 Giorgos Mitropoulos (Speaker)* Michalis Xeferis[‡]

1 Introduction

Modern breakthroughs in machine learning have motivated researchers to incorporate *predictions* of unknown quality into algorithm design for optimization problems, giving rise to the rapidly growing field of *algorithms with predictions*. While most work has focused on online optimization, there has been increasing interest in applying such algorithms to overcome the computational challenges of NP-hard problems in offline settings. The use of various predictions in this context has led to advances on well-known problems, such as CLUSTERING [7, 8], MAXCUT [1, 5, 6] and MAXIMUM INDEPENDENT SET [4].

In a recent work, Bampis et al. [2] adapted the approach of Braverman and Mossel [3] for noisy sorting without resampling to tackle permutation optimization problems. In [3], the authors showed that access to an oracle for independent pairwise comparisons that is even slightly biased towards the truth allows one to recover, with high probability, an ordering in which each element is at most $\mathcal{O}(\log n)$ positions away from its true position. Using this model, Bampis et al. [2] proved that permutation optimization problems satisfying certain structural properties can be solved with high probability in polynomial time. One such problem is $1 \mid \text{prec} \mid \sum C_j$, making this the first use of predictions to obtain an exact solution for an NP-hard scheduling problem (and NP-hard problems in general) in the offline setting, to our knowledge.

A natural continuation of their work is to develop prediction models for scheduling problems involving multiple machines, whether the number of machines is a constant or part of the input. Such settings make scheduling problems more challenging, especially in the presence of constraints (e.g. precedence, release dates),

*evripidis.bampis@lip6.fr, bruno.escoffier@lip6.fr, georgios.mitropoulos@lip6.fr. Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

[†]fotakis@cs.ntua.gr National Technical University of Athens, Greece and Archimedes Research Unit, Athena RC, Greece. Research partially supported by project MIS 5154714 of the National Recovery and Resilience Plan Greece 2.0 funded by the European Union under the NextGenerationEU Program.

[‡]xeferis@uni-bremen.de. University of Bremen, Germany

since they often require solving two interdependent tasks simultaneously: distributing jobs across machines and ordering them on each machine.

2 Our Contribution

In this work, we utilize predictions to address makespan minimization scheduling problems of n jobs to multiple machines. Specifically, given an encoding of a *predicted* solution, we seek to recover the optimal one. Naturally, we expect both the solution encoding and the prediction quality to significantly influence our ability to do so. We examine three distinct approaches to encoding a solution.

The first is a *partition-based encoding*, where we are given a *predicted* assignment of jobs to machines. For predictions based on this encoding, we prove that if at most k jobs are assigned to the wrong machine, then recovering the optimal solution for $P2 || C_{max}$ cannot be achieved in time $n^{o(k)}$ under ETH. This essentially rules out any realistic prediction model, since the problem may be solvable in polynomial time iff only a constant number of jobs are incorrectly assigned, regardless of the input size.

The remaining two encodings are based on job orderings, which we represent as a permutation. This decision is motivated by the observation that such a prediction can be derived from an oracle performing appropriate independent one-bit queries between jobs. Indeed, a direct consequence of [3] is that if these queries are even slightly biased towards the truth, then with high probability one can recover a *predicted* ordering, in which each job is displaced from its position in the optimal permutation by at most $\mathcal{O}(\log n)$ positions.

First, we examine a *greedy-based encoding*, in which the optimal ordering is one such that assigning each job to the currently least-loaded machine yields an optimal solution. For this encoding, we prove that even if such an ordering is given with each job being at most one position away from its optimal position, $P2 || C_{max}$ remains NP-complete. Again, this rules out any realistic prediction model.

Second, we examine an *order-based encoding*, in which the optimal ordering is a one-dimensional array obtained by concatenating the optimal job schedules of the m machines in sequential order (i.e. the jobs of machine 1, followed by those of machine 2, etc.). Although the $m - 1$ cut positions are, in principle, required to correctly extract each machine's schedule from the optimal ordering, we omit them in order to keep the encoding minimal, since they can be recovered in polynomial time using a simple dynamic programming (DP) algorithm. This encoding appears to be the only one that retains sufficient information to recover the optimal solution in the presence of a non-constant number of errors. Intuitively, this is because it can exploit the locality implied by the fact that each job is at most $k = \mathcal{O}(\log n)$ positions away from its position in the optimal ordering. We assume that we are given a *predicted* ordering for this encoding and use it to recover the optimal solution for each of the following problems.

For $P || C_{max}$, we develop a DP algorithm that takes as input the *predicted* job ordering and the number of machines m and determines the optimal cut positions

to minimize the makespan. It does so by iteratively finding the optimal cut positions when $m' \in \{1, \dots, m\}$ machines are available, while optimally distributing the k jobs on each side of the cut among the machines.

We additionally examine two scheduling variants involving constraints. The first variant assumes that each job has a release date, known a priori. Since makespan minimization in this setting can be solved using our previous algorithm, we instead focus on the more challenging $P|r_j|\sum C_j$, for which we generalize our DP approach. The second variant assumes the existence of precedence constraints. For constant number of machines ($Pm|\text{prec}|C_{max}$), we obtain a pseudopolynomial-time algorithm by first guessing the assignment of jobs to machines and then using a DP table to sequentially determine the position of each job on its assigned machine.

Nevertheless, we show that, although it is powerful, this model is not a silver bullet for all scheduling problems. Specifically, we prove that it cannot assist in efficiently solving the $P|\text{prec}|C_{max}$ problem when the number of machines is part of the input, unless $P = NP$, via a reduction from 3-SAT. This reduction holds even when the jobs in the given *predicted* ordering are at most two positions away from their position in the optimal ordering.

As a concluding remark, we note that the aforementioned algorithms extend to problems beyond parallel identical machines scheduling. In particular, scheduling on unrelated parallel machines with even multidimensional objectives falls within the scope of this work and marks a promising direction. Furthermore, the algorithm for $P||C_{max}$ can be adapted to solve all problems that receive a *predicted* ordering for the *order-based encoding* as input and seek to find a partition into m subsets S_1, \dots, S_m that minimizes $\max_{i \in [m]} f_i(S_i)$ (or maximizes $\min_{i \in [m]} f_i(S_i)$), where each f_i depends only on the subset index $i \in \{1, \dots, m\}$ and can be computed efficiently. Similarly, the algorithms for $Pm|\text{prec}|C_{max}$ with constant number of machines and $P|r_j|\sum C_j$ can be modified to address analogous problems.

References

- [1] Evripidis Bampis, Bruno Escoffier, and Michalis Xefferis. “Parsimonious Learning-Augmented Approximations for Dense Instances of NP-hard Problems”. In: *ICML*. OpenReview.net, 2024.
- [2] Evripidis Bampis et al. “Polynomial Time Learning Augmented Algorithms for NP-hard Permutation Problems”. In: *ICML*. Vol. 267. Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025.
- [3] Mark Braverman and Elchanan Mossel. “Noisy sorting without resampling”. In: *SODA*. SIAM, 2008, pp. 268–276.
- [4] Vladimir Braverman et al. “Learning-Augmented Maximum Independent Set”. In: *APPROX/RANDOM*. Vol. 317. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, 24:1–24:18.

- [5] Vincent Cohen-Addad et al. “Learning-Augmented Approximation Algorithms for Maximum Cut and Related Problems”. In: *NeurIPS*. 2024.
- [6] Yin hao Dong, Pan Peng, and Ali Vakilian. “Learning-Augmented Streaming Algorithms for Approximating MAX-CUT”. In: *ITCS*. Vol. 325. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025, 44:1–44:24.
- [7] Jon C. Ergun et al. “Learning-Augmented k-means Clustering”. In: *ICLR*. OpenReview.net, 2022.
- [8] Buddhima Gamlath et al. “Approximate Cluster Recovery from Noisy Labels”. In: *COLT*. Vol. 178. Proceedings of Machine Learning Research. PMLR, 2022, pp. 1463–1509.

Strategyproof Scheduling with Predictions*

Eric Balkanski (Speaker)[†] Vasilis Gkatzelis[‡] Xizhi Tan[§]

1 Introduction

In their seminal paper which initiated the field of algorithmic mechanism design, Nisan and Ronen [1999] focused on makespan minimization with strategic agents: a set of m jobs needs to be scheduled on a set of n machines aiming to minimize the makespan, and each machine is owned by an agent who requires a monetary compensation in exchange for processing the jobs assigned to them. The goal is to design *strategyproof* mechanisms that carefully choose the outcome to ensure that no agent has an incentive to misreport their costs. Nisan and Ronen [1999] proposed a strategyproof mechanism for this problem that is an n -approximation of the optimal makespan. Even though this guarantee is much less appealing than the 2-approximation that one can achieve in polynomial time if the costs are publicly known [Lenstra et al., 1990], they went on to make the bold conjecture that this is the best possible worst-case approximation that any deterministic strategyproof mechanism can achieve (polynomial time or not). Christodoulou et al. [2023] recently validated the conjecture by proving a lower bound of n for all deterministic mechanisms. This result paints a very pessimistic picture about designing practical strategyproof mechanisms for this problem. However, it heavily depends on the, rather unrealistic assumption, that the mechanism has no information regarding the costs of the agents.

Facing analogous worst-case impossibility results that are overly pessimistic and rather uninformative, a recent surge of work has focused on the “learning-augmented framework”, aiming to achieve more refined and informative bounds, while retaining the benefits of worst-case analysis (see [Mitzenmacher and Vassilvitskii, 2021] for a survey). This framework assumes that the designer is provided with some machine-learned predictions that can be used to design more practical algorithms that achieve near optimal performance when the predictions are correct (this is called the *consistency* guarantee). However, rather than assuming that the predictions are always accurate and forfeiting the benefits of worst-case analysis altogether, this framework seeks to also provide strong guarantees, even if

*This paper was published at ITCS 2023.

[†]eb3224@columbia.edu. Department of Industrial Engineering and Operations Research, Columbia University

[‡]gkatz@drexel.edu. Computer Science Department, Drexel University

[§]xizhi@stanford.edu. Computer Science Department, Stanford University

the predictions are arbitrarily inaccurate (this is called the *robustness* guarantee). More formally, the consistency of an algorithm is the worst-case approximation guarantee that it achieves assuming that the prediction it was provided with is accurate, while its robustness is the worst-case approximation guarantee without any assumptions regarding the prediction accuracy. This framework was recently extended to settings involving strategic agents [Agrawal et al., 2023, Gkatzelis et al., 2022, Xu and Lu, 2022], motivating the design of new mechanisms leveraging predictions to overcome the incentive issues that arise.

Given predictions regarding the machines’ processing times, a naive solution would be to assume the predictions are accurate and output a schedule with small makespan according to the predicted processing times alone (i.e., without eliciting the agents’ true processing times). This is a strategyproof mechanism that would achieve a consistency of $O(1)$, but its robustness would be unbounded (e.g., even if a single job’s processing time is mispredicted, it could end up being assigned to a machine where its actual processing time is arbitrarily high). On the other hand, the mechanism of Nisan and Ronen [1999] would achieve a robustness of $O(n)$, but its consistency would also be no better than $O(n)$, since its output disregards the predictions. In an attempt to provide a middle ground between these two extremes, very recent work by Xu and Lu [2022] proposed a strategyproof mechanism that can guarantee a consistency of $O(1)$ with bounded robustness, but the robustness guarantee of $O(n^3)$ that it achieves is much worse than the $O(n)$ robustness achieved in Nisan and Ronen [1999]. Our main result in this paper is a strategyproof that combines the “best of both worlds”: a consistency of $O(1)$ with a robustness of $O(n)$ that matches the lower bound of Christodoulou et al. [2023].

Our results. We assume that the mechanism is provided with predictions $\hat{p}(i, j)$ regarding the amount of time that each machine i would require in order to fully process each job j ; crucially, these predictions may be highly inaccurate.

Our main result is a deterministic polynomial-time strategyproof mechanism that guarantees a consistency of $(4 + 2\gamma)$ and a robustness of $(1 + \frac{1}{\gamma})n$ for any choice of $\gamma \in (0, \frac{n}{2} - 1)$, which is a parameter that the designer can determine to receive their desired trade-off between consistency and robustness. For example, setting $\gamma = 1$ yields a mechanism with a small constant consistency of 6 (i.e., a 6-approximation of the optimal makespan when the predictions are accurate) while maintaining a robustness of $2n$ (i.e., a worst-case approximation guarantee that asymptotically matches the best approximation possible, irrespective of how inaccurate the predictions may be).

Our main mechanism, SCALEDGREEDY, uses the predicted processing times to pre-compute a schedule that (approximately)¹ minimizes the makespan, assuming these predictions are correct. The mechanism then uses this schedule as a guide and adds a “bias” in favor of scheduling jobs on the machines that they were assigned

¹Note that even if the processing times are known in advance, no known polynomial time algorithm can achieve an approximation factor better than 2, and it is NP-hard to achieve a factor better than 1.5 [Lenstra et al., 1990].

to in this schedule. After introducing this bias, the mechanism follows a simple greedy procedure, so the main technical novelty is the subtle way in which this bias towards the predictions is introduced, while ensuring that we asymptotically match the best known robustness when the predictions are arbitrarily inaccurate. Roughly speaking, the mechanism carefully chooses a subset of jobs and proceeds to assign each of them to the same machine as in the pre-computed schedule, unless the processing times reported by the machines for that job differ significantly from the predicted ones.

Recent follow-up work. Recently, Christodoulou et al. [2024] and Cole et al. [2025] have studied the same problem but with parsimonious predictions, where the goal is to achieve desirable consistency-robustness tradeoffs without having to predict all mn processing times.

References

- P. Agrawal, E. Balkanski, V. Gkatzelis, T. Ou, and X. Tan. Learning-augmented mechanism design: Leveraging predictions for facility location. *Mathematics of Operations Research*, 0(0):0–0, 2023. doi: 10.1287/moor.2022.0225.
- G. Christodoulou, E. Koutsoupias, and A. Kovács. A proof of the nisan-ronen conjecture. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 672–685, 2023.
- G. Christodoulou, A. Sgouritsa, and I. Vlachos. Mechanism design augmented with output advice. *Advances in Neural Information Processing Systems*, 37:41934–41953, 2024.
- R. Cole, A. Gupta, and P. Jangir. Parsimonious predictions for strategyproof scheduling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.
- V. Gkatzelis, K. Kollias, A. Sgouritsa, and X. Tan. Improved price of anarchy via predictions. In D. M. Pennock, I. Segal, and S. Seuken, editors, *EC '22: The 23rd ACM Conference on Economics and Computation, Boulder, CO, USA, July 11 - 15, 2022*, pages 529–557. ACM, 2022.
- J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Program.*, 46:259–271, 1990.
- M. Mitzenmacher and S. Vassilvitskii. *Algorithms with Predictions*, page 646–662. Cambridge University Press, 2021. doi: 10.1017/9781108637435.037.
- N. Nisan and A. Ronen. Algorithmic mechanism design (extended abstract). In J. S. Vitter, L. L. Larmore, and F. T. Leighton, editors, *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA*, pages 129–140. ACM, 1999.
- C. Xu and P. Lu. Mechanism design with predictions. In L. D. Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 571–577. ijcai.org, 2022. doi: 10.24963/ijcai.2022/81. URL <https://doi.org/10.24963/ijcai.2022/81>.

Speed Predictions for Online Energy-Efficient Scheduling

Eric Balkanski

Jingwei Li (Speaker)

Clifford Stein

Cherlin Zhu*

1 Introduction

Massive data centers that power modern artificial intelligence engines are placing significant pressure on power grids and sustainability goals. When scheduling computing jobs, the energy consumption is typically controlled by the speed at which machines are run. In *speed scaling* problems, the operating system/algorithm designer controls the machine speed, and the power usage is typically modeled as the speed to some power $\alpha > 1$. The total energy consumption is power integrated over time, and the goal is to scale the speed at which jobs run in order to minimize total energy consumption while satisfying performance requirements such as deadlines.

We study *online* speed scaling with deadlines, where jobs arrive over time and future jobs are unknown. Classical online algorithms achieve constant competitive ratios, but treating future jobs as entirely unknown can be overly pessimistic. This has motivated the *learning-augmented / algorithms with predictions* framework, where an online algorithm is provided with predictions about the future in order to improve performance while remaining robust to poor predictions.

A crucial modeling choice in learning-augmented algorithms is *what should be predicted*. Prior work on speed scaling with predictions typically predicts (some or all of) the *job parameters* (the input). This extended abstract argues for a different viewpoint: in speed scaling, it is often more natural to predict the *machine speed* (the output) rather than the entire job sequence. We propose prediction models where the advice is a *speed profile* suggesting how fast the machine should run over time.

Speed predictions have three main advantages. *Succinct predictions*: a job consists of multiple parameters (release time, deadline, work, and sometimes additional attributes), whereas speed predictions only specify the machine speed. *Smoothness without error metrics*: competitive guarantees for predictions are often stated as a function of a chosen prediction-error measure, but such measures can be complex and incomparable across models; by predicting the output, one can instead define guarantees relative to the *quality of the predicted output* itself. *Dynamic*

*eb3224@columbia.edu, jl6639@columbia.edu, cliff@ieor.columbia.edu, cz2740@columbia.edu. Department of Industrial Engineering and Operations Research at Columbia University.

predictions: speed predictions can be provided at every time t , allowing them to incorporate data observed at runtime rather than being fixed at time 0.

Our contributions are (i) new speed prediction models for online speed scaling, together with a smoothness notion that does not require a prediction error metric and allows infeasible predictions, and (ii) learning-augmented algorithms with strong consistency/robustness/smoothness guarantees.

2 Model and prediction setups

Online speed scaling with deadlines. The input is a set J of jobs. Each job $j \in J$ has a release date r_j , a deadline $d_j > r_j$, and a work requirement w_j . Jobs are revealed online by release time. At each time t , an algorithm chooses a machine speed $s(t)$ and processes available work. The energy cost of a schedule is

$$\text{cost}(s) := \int_0^\infty s(t)^\alpha dt,$$

where $\alpha > 1$ is a fixed constant. A schedule is feasible if every job completes by its deadline. Let $\text{OPT}(J)$ denote the unique optimal offline speed function for instance J .

Speed prediction models. We consider three prediction models:

- **Dynamic machine speed predictions:** at every time t , the algorithm receives a prediction $\hat{s}(t)$ for the machine speed at time t .
- **Offline machine speed predictions:** the full predicted speed function $\hat{s}(\cdot)$ is revealed at time 0.
- **Dynamic job speed predictions:** upon release of job j at time r_j , the algorithm receives a prediction $\hat{\ell}_j$ for the (constant) speed at which job j should run in an optimal schedule; these induce a predicted machine-speed profile by running jobs in EDF order at speeds $\hat{\ell}_j$.

Consistency and robustness. As is standard, *consistency* is the competitive ratio when predictions are accurate, while *robustness* is the competitive ratio under arbitrary predictions. An algorithm is γ -consistent if, when the prediction corresponds to $\text{OPT}(J)$, its cost is at most $\gamma \cdot \text{cost}(\text{OPT}(J))$. It is β -robust if for all instances and all predictions its cost is at most $\beta \cdot \text{cost}(\text{OPT}(J))$.

Smoothness via output quality and feasibility projection. For output predictions, it is natural to compare to a benchmark that simply follows the prediction. However, in speed scaling a predicted speed profile can be *infeasible* (too slow to finish all work by deadlines). To obtain meaningful guarantees even for infeasible predictions, we use a *feasibility projection*.

Definition 1 (Feasibility projection) A procedure $\text{MAKEFEAS}(J, \hat{s})$ is a feasibility projection if it returns a feasible schedule; equals \hat{s} whenever \hat{s} is feasible; and is online (its speed at time t depends only on information available at t).

Definition 2 (Smoothness) An algorithm is ζ -smooth (w.r.t. MAKEFEAS) if for all instances J and predictions \hat{s} ,

$$\frac{\text{cost}(\text{ALG}(J, \hat{s}))}{\text{cost}(\text{MAKEFEAS}(J, \hat{s}))} \leq \zeta.$$

This notion does not require defining a prediction-error metric, and it provides strong guarantees whenever the predicted output itself has low cost.

3 Algorithms and guarantees

3.1 Dynamic machine speed predictions: $(1 + \varepsilon)$ -consistency and $O(1)$ -robustness

In the dynamic machine speed prediction model, we give an algorithm that is, for any constant $\varepsilon > 0$, $(1 + \varepsilon)$ -consistent and $O(1)$ -robust. At a high level, the algorithm partitions jobs into two sets: jobs in J_{pred} are processed according to the predicted speed profile, while jobs in J_{on} are processed by a standard robust online algorithm (e.g. Optimal Available). The key challenge is to maintain robustness even though blindly following the prediction can be arbitrarily bad. To do so, the algorithm maintains a *forecast schedule* (the cheapest schedule consistent with predictions so far) and a *robust forecast schedule* whose cost is provably controlled; when the forecast becomes too aggressive, it *reassigns* work from the predicted set to the robust set so feasibility is preserved without blowing up the objective.

Theorem 3 (Dynamic machine speed predictions) Fix $\alpha > 1$. For any $\varepsilon \in (0, 1)$, there exists a parameter choice (confidence level) such that the above algorithm is $(1 + \varepsilon)$ -consistent and $O(1)$ -robust, with constants depending only on α , ε , and the competitive ratio of the chosen baseline online algorithm.

These guarantees match the qualitative guarantees of prior learning-augmented results for speed scaling, while requiring *less informative* predictions than predicting the entire input sequence.

3.2 Offline speed and dynamic job speed predictions: $(1 + \varepsilon)$ -smoothness and $O(1)$ -robustness

For offline machine speed predictions and dynamic job speed predictions, we provide an algorithm that achieves the stronger guarantee of $(1 + \varepsilon)$ -smoothness while maintaining $O(1)$ -robustness.

The algorithm again partitions jobs into J_{on} and J_{pred} . Jobs in J_{on} are handled by a robust online algorithm. Jobs in J_{pred} are handled by a feasibility projection

MAKEFEAS that follows the prediction whenever safe, and otherwise minimally increases speed to ensure upcoming deadlines are met. For offline machine speed predictions, the feasibility projection can be described succinctly: at time t , look at the earliest deadline among available unfinished jobs; if the predicted remaining work until that deadline is insufficient, run at the maximum of the predicted speed and the minimum speed needed to finish those jobs by the deadline; otherwise follow the prediction.

Theorem 4 (Smoothness and robustness) *Fix $\alpha > 1$. For any $\varepsilon > 0$, in the offline machine speed prediction model, the above algorithm is $(1 + \varepsilon)$ -smooth with respect to the feasibility projection MAKEFEAS and $O(1)$ -robust. Analogous guarantees hold in the dynamic job speed prediction model with an appropriate feasibility projection.*

This shows that, under output predictions, one can obtain guarantees that smoothly track the quality of the predicted output itself, without committing to any particular notion of prediction error.

References

- [1] F. YAO, A. DEMERS, AND S. SHENKER (1995). *A scheduling model for reduced CPU energy*. In *Proceedings of IEEE FOCS*, pp. 374–382.
- [2] N. BANSAL, T. KIMBREL, AND K. PRUHS (2007). *Speed scaling to manage energy and temperature*. *Journal of the ACM*, 54(1):1–39.
- [3] E. BAMAS, A. MAGGIORI, L. ROHWEDDER, AND O. SVENSSON (2020). *Learning augmented energy minimization via speed scaling*. In *NeurIPS*, pp. 15350–15359.
- [4] E. BALKANSKI, N. PERIVIER, C. STEIN, AND H.-T. WEI (2023). *Energy-efficient scheduling with predictions*. In *NeurIPS*, pp. 79012–79023.
- [5] A. ANTONIADIS, P. JABBARZADE, AND G. SHAHKARAMI (2022). *A novel prediction setup for online speed-scaling*. In *Scandinavian Symposium and Workshops on Algorithm Theory*.
- [6] N. CHRISTIANSON, J. SHEN, AND A. WIERMAN (2023). *Optimal robustness-consistency tradeoffs for learning-augmented metrical task systems*. In *AIS-TATS*, pp. 9377–9399.
- [7] E. CHO, S. MYERS, AND J. LESKOVEC (2011). *Friendship and mobility: user movement in location-based social networks*. In *KDD*, pp. 1082–1090.
- [8] T. LYKOURIS AND S. VASSILVITSKII (2021). *Competitive caching with machine learned advice*. *Journal of the ACM*, 68(4):1–25.

Revisiting Johnson's rule for minimizing makespan in the two-machine flow shop scheduling problem

Federico Della Croce ^{*} Quentin Schau (Speaker) [†]

1 Introduction

We consider the classical two-machine flow shop scheduling problem with the objective of minimizing the makespan, commonly denoted by $F2 \parallel C_{\max}$ [2]. A set $J = \{1, \dots, n\}$ of jobs must be processed on two machines M_1 and M_2 in this order. Job j requires a processing time $p_{1,j}$ on M_1 and $p_{2,j}$ on M_2 . All jobs are available at time 0, each machine can process at most one job at a time, and preemption is not allowed. It is well known that there always exists an optimal schedule that is a permutation schedule and can be obtained by Johnson's rule [3]. This rule first schedules the set A of jobs such that $p_{1,j} \leq p_{2,j}$ in nondecreasing order of $p_{1,j}$, and then the set B of jobs such that $p_{1,j} > p_{2,j}$ in nonincreasing order of $p_{2,j}$.

Due to the required sorting operations, Johnson's rule runs in $O(n \log n)$ time and $O(n)$ space. We investigate general conditions under which the problem can be solved optimally in linear time. More specifically, we prove that a simple procedure running in $O(n)$ time allows one to determine whether a full sort can be avoided and, correspondingly, whether an optimal solution can be computed in linear time. To the best of our knowledge, this is the first work showing that Johnson's rule can be implemented in linear time with high probability under standard distributions of processing times.

2 Revisiting Johnson's rule for problem $F2 \parallel C_{\max}$

The problem $F2 \parallel C_{\max}$ satisfies the reversibility property stated below. Consider a pair of two-machine flow shops where $p_{i,j}^{(1)}$ and $p_{i,j}^{(2)}$ denote the processing time of job j on machine i in the first and second flow shop, respectively.

Property 1 ([4, 5]) *If $p_{1,j}^{(1)} = p_{2,j}^{(2)}$ and $p_{2,j}^{(1)} = p_{1,j}^{(2)}$ for all jobs j , then sequence j_1, \dots, j_n in the first flow shop results in the same makespan as sequence j_n, \dots, j_1 in the second flow shop.*

Assume, without loss of generality, that the $n = n_A + n_B$ jobs are indexed according to Johnson's sequence, where jobs $1, \dots, n_A$ belong to set A and jobs $n_A + 1, \dots, n$ belong

^{*}federico.dellacroce@polito.it. DIGEP, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy. CNR, IEIIT, Corso Duca degli Abruzzi 24, 10129 Torino, Italy.

[†]quentin.schau@polito.it. DIGEP, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy. Laboratoire d'Informatique Fondamentale et Appliquée, Université de Tours, 64 rue Jean Portalis, 37200 Tours, France.

to set B . Let p^{\max} denote the largest processing time among all operations. Moreover, let p_{1A}^{\max} (resp., p_{2B}^{\max}) denote the largest processing time in subset A on machine M_1 (resp., in subset B on machine M_2). Let $P_1 = \sum_{i=1}^n p_{1,i}$ and $P_2 = \sum_{i=1}^n p_{2,i}$ be the total processing times on machines M_1 and M_2 , respectively. Similarly, let $P_{1A} = \sum_{i=1}^{n_A} p_{1,i}$ and $P_{2A} = \sum_{i=1}^{n_A} p_{2,i}$ be the total processing times of the jobs in subset A on machines M_1 and M_2 , and let $P_{1B} = \sum_{i=n_A+1}^n p_{1,i}$, and $P_{2B} = \sum_{i=n_A+1}^n p_{2,i}$ be the corresponding totals for subset B . Finally, for each job i , let $C_{1,i}$ (resp., $C_{2,i}$) denote its completion time on machine M_1 (resp., on machine M_2). The following properties (proof omitted) provide sufficient conditions for finding an optimal solution to $F2 \parallel C_{\max}$ without requiring a full sort of the job set.

Property 2 *If $P_{1A} \leq P_{2A} - p_{1A}^{\max}$, let k_A denote the smallest index such that $\sum_{i \in A: p_{1,i} \leq p_{1,k_A}} p_{1,i} \leq \sum_{i \in A: p_{1,i} \leq p_{1,k_A}} p_{2,i} - p_{1A}^{\max}$ and $p_{1,k_A} < p_{1,k_A+1}$. Then, for set A , an optimal schedule is obtained by sequencing jobs $1, \dots, k_A$ as in Johnson's order, followed by jobs $k_A + 1, \dots, n_A$ in any order.*

Property 3 *If $P_{2B} \leq P_{1B} - p_{2B}^{\max}$, let k_B denote the largest index such that $\sum_{i \in B: p_{2,i} \leq p_{2,k_B}} p_{2,i} \leq \sum_{i \in B: p_{2,i} \leq p_{2,k_B}} p_{1,i} - p_{2B}^{\max}$ and $p_{2,k_B} > p_{2,k_B-1}$. Then, for set B , an optimal schedule is obtained by sequencing jobs $n_A + 1, \dots, k_B - 1$ in any order, followed by jobs k_B, \dots, n as in Johnson's order.*

Combining Properties 2 and 3, we can decide in linear time whether a sufficient condition for solving $F2 \parallel C_{\max}$ in overall linear time holds. Indeed, partitioning the jobs into subsets A and B takes linear time. Finding k_A in A and k_B in B can be done in linear time via selection algorithms, e.g., the median-finding technique of [1]. If $k_A \log k_A \leq n$ and $(n - k_B + 1) \log(n - k_B + 1) \leq n$, then sorting the reduced sets $1, \dots, k_A$ and k_B, \dots, n also requires overall linear time. Computational testing on instances with uniform distribution indicates that linear time complexity consistently holds in practice (detailed computational results will be presented at the Conference).

3 Probabilistic analysis

In this section, p^{\max} denotes the upper bound of the uniform distribution (i.e., the maximum possible processing time), consistently with the experimental setting. Here, we present a probabilistic analysis for instances with n jobs, where processing times are independently and uniformly distributed over the set $\{1, \dots, p^{\max}\}$. We focus on parameter k_A and subset A . By the reversibility property (Property 1), the same analysis applies symmetrically to parameter k_B and subset B . Our goal is to estimate the probability that the value of k_A is sufficiently small to guarantee linear-time complexity, and to analyze how this probability behaves as n grows. We assume that $p^{\max} = \Theta(n)$, which is a standard setting in flow shop benchmark instances.

3.1 Step 1: Lower bound on the size of set A

We fix a parameter α with $0 < \alpha < 1$. For each job i , we consider the following restrictive condition: $C_1(i) := (p_{1,i} \leq \alpha p^{\max}) \wedge (p_{2,i} > \alpha p^{\max})$. Clearly, condition $C_1(i)$ implies that job i belongs to set A , although the converse does not necessarily hold.

Lemma 1 (Probability of condition C_1) For any job i , the probability that condition $C_1(i)$ holds is $\Pr[C_1(i)] = \alpha(1 - \alpha)$.

Let X be the random variable counting the number of jobs satisfying condition C_1 . Then X follows a binomial distribution with parameters n and $p = \alpha(1 - \alpha)$. Since we are interested in the probability that at least k_A jobs satisfy condition C_1 , we define $P_1^* := \Pr(X \geq k_A) = \sum_{k=k_A}^n \binom{n}{k} [\alpha(1 - \alpha)]^k [1 - \alpha(1 - \alpha)]^{n-k}$.

3.2 Step 2: Lower bound on the sum of processing time differences

From Step 1, it follows that with probability lower bounded by P_1^* there exist at least k_A jobs $i \in A$ such that $p_{1,i} \leq \alpha p^{\max}$. Consider the first k_A jobs in set A according to Johnson's sequence. The inequality $p_{1,i} \leq \alpha p^{\max}$ necessarily holds for all these jobs. For each such job i , let $\delta_i := p_{2,i} - p_{1,i}$. Conditioned on a fixed value of $p_{1,i}$, the random variable δ_i is uniformly distributed over $\{0, \dots, p^{\max} - p_{1,i}\}$. Let $p' := (1 - \alpha)p^{\max}$. We further restrict the support of each δ_i to the smaller set $\{0, \dots, p'\}$. Correspondingly, the variables $\delta_1, \dots, \delta_{k_A}$ are independent and uniformly distributed over $\{0, \dots, p'\}$.

Let $P(j, \gamma)$ denote the probability that $\sum_{i=1}^j \delta_i = \gamma$, i.e., the ratio between the number of realizations yielding exactly γ and $(p' + 1)^j$, where $(p' + 1)^j$ represents the total number of possible realizations. The maximum attainable value of γ is jp' . We initialize the recursion with $P(0, 0) = 1$, $P(0, \gamma) = 0 \forall \gamma > 0$ and $P(j, \gamma) = 0$ if $\gamma < 0$. Our recursion relation is then $P(j, \gamma) = \frac{\sum_{i=0}^{p'} P(j-1, \gamma-i)}{p'+1}$, $j = 1, \dots, k_A$.

Finally, the probability that $\sum_{i=1}^{k_A} \delta_i \geq p^{\max}$ holds is given by $P_2^* = 1 - \sum_{\gamma=0}^{p^{\max}-1} P(k_A, \gamma)$.

Correspondingly, the product $P^* = P_1^* \cdot P_2^*$ provides a lower bound on the probability that only the first k_A jobs need to be sorted in subset A , subject to the complexity constraint $k_A \log k_A \leq p_{\max} = n$. In particular, we obtain $P^* > 1 - 10^{-3}$ for $n = 100$ and $P^* > 1 - 10^{-9}$ for $n = 200$.

References

- [1] M. Blum, R.W. Floyd, V.R. Pratt, R.L. Rivest, and R.E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7(4), 448–461, 1972.
- [2] 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.
- [3] S. M. Johnson. Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1(1):61–68, 1954.
- [4] E. J. Muth. The Reversibility Property of a Production Line. *Management Science*, 25, 152–158, 1979.
- [5] M. L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*, 5th ed. Springer, New York, 2016.

Revisit the Scheduling Problem with Calibrations

Lin Chen ^{*} Yixiong Gao (Speaker) [†] Minming Li [‡] Guohui Lin [§]
Kai Wang [¶]

1 Introduction

Scheduling with calibrations models scenarios in which machines must be calibrated before processing jobs and remain operational only for a fixed duration after each calibration. In 2013, Bender et al. [1] proposed a theoretical framework for scheduling with calibrations. They considered unit-time jobs with release times and deadlines and studied the objective of minimizing the number of calibrations. For the single-machine case, they obtained an optimal polynomial-time algorithm, while for multiple machines they established a 2-approximation algorithm.

Subsequent research has investigated extensions including non-unit processing times and resource augmentation [2], multiple calibration types and hardness results [8, 7], alternative objectives such as weighted throughput [4] and flow-time minimization [3], batch calibration models and related cost formulations [5, 6], and online variants [9]. More recently, calibration-constrained scheduling has also been studied for multi-interval jobs [10]. Nevertheless, even for unit-time jobs with release times and deadlines, the computational complexity of the general multi-machine case remains unresolved.

In this paper, we consider the original multi-machine setting proposed by Bender et al. [1]. There are m identical parallel machines. Each job j has a release time r_j and a deadline d_j , and must be processed within its time window $[r_j, d_j]$. A calibration can be performed instantaneously at any time and enables processing on that machine for a subsequent interval of length T ; outside calibrated intervals, machines cannot process jobs.

^{*}chenlin198662@gmail.com. Department of Computer Science, Zhejiang University, Hangzhou, Zhejiang, China.

[†]yixiong.gao@my.cityu.edu.hk. Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong SAR, China.

[‡]minming.li@cityu.edu.hk. Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong SAR, China.

[§]guohui@ualberta.ca Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada.

[¶]kai.wang@my.cityu.edu.hk Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong SAR, China.

We study two optimization problems under this model: (i) calibration minimization, which schedules all jobs while minimizing the total number of calibrations; and (ii) weighted throughput maximization, which selects and schedules a subset of jobs maximizing the total weight of completed jobs given a calibration budget.

We develop a unified algorithmic framework for both objectives. When either the number of machines m or the calibration length T is constant, we obtain polynomial-time exact algorithms. For the general case, we design a polynomial-time approximation scheme. The computational complexity of the general multi-machine case remains open.

2 Results

We first consider the calibration minimization problem. A key structural observation is that, once the schedule of calibrations is fixed, the schedule of jobs can be obtained by applying the classical *Earliest-Deadline-First* (EDF) scheduling algorithm, in which for any time slot, the job of the earliest deadline has the highest priority to be considered to schedule whenever a machine is available (i.e., calibrated).

Thus, the challenge lies in determining the placement of calibration intervals. Our algorithms are based on dynamic programming over calibration placements.

Theorem 1 *If the number of machines m or the calibration length T is constant, then the calibration minimization problem can be solved in polynomial time.*

The dynamic program maintains the relative positions of the most recent calibration intervals on each machine.

For the general case, we obtain the following approximation result.

Theorem 2 *For any $\varepsilon > 0$, there exists a polynomial-time $(1 + \varepsilon)$ -approximation algorithm for the calibration minimization problem.*

The PTAS is built on a resource-augmentation analysis: we first consider a relaxed variant in which we allow additional εm machines. Under this augmentation, the structure of near-optimal solutions becomes sufficiently regular to permit discretizing the time horizon and a controlled enumeration of calibration placements. We then remove the augmentation via a careful loss-compensation argument, showing that the solution can be transformed into a feasible one for the original instance with only a $(1 + \varepsilon)$ -fold increase in the number of calibrations.

The weighted throughput maximization problem can be addressed within the same structural framework. The dynamic programming approach is extended by incorporating the number of calibrations used into the state description and optimizing the accumulated weight subject to the calibration budget. The PTAS can be adapted using analogous rounding and discretization techniques.

References

- [1] M.A. BENDER, D.P. BUNDE, V.J. LEUNG, S. MCCAULEY, AND C.A. PHILLIPS (2013). Efficient Scheduling to Minimize Calibrations. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '13, pp. 280–287.
- [2] J.T. FINEMAN AND B. SHERIDAN (2015). *Scheduling non-unit jobs to minimize calibrations*. In *Proceedings of the 27th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '15, pp. 161–170.
- [3] V. CHAU, M. LI, S. MCCAULEY, AND K. WANG (2017). *Minimizing total weighted flow time with calibrations*. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '17, pp. 67–76.
- [4] V. CHAU, S. FENG, M. LI, Y. WANG, G. ZHANG, AND Y. ZHANG (2019). *Weighted throughput maximization with calibrations*. In *Algorithms and Data Structures: 16th International Symposium*, WADS '19, pp. 311–324.
- [5] V. CHAU, M. LI, E.Y. WANG, R. ZHANG, AND Y. ZHAO (2020). *Minimizing the cost of batch calibrations*. *Theoretical Computer Science*, 828:55–64.
- [6] K. WANG (2020). *Calibration scheduling with time slot cost*. *Theoretical Computer Science*, 821:1–14.
- [7] H. CHEN, V. CHAU, L. CHEN, AND G. ZHANG (2020). *Scheduling many types of calibrations*. In *International Conference on Algorithmic Applications in Management*, pp. 286–297.
- [8] E. ANGEL, E. BAMPIS, V. CHAU, AND V. ZISSIMOPOULOS (2021). *Calibration scheduling with arbitrary lengths and activation length*. *Journal of Scheduling*, 24(5):459–467.
- [9] Z. CHEN AND J. ZHANG (2022). *Online scheduling of time-critical tasks to minimize the number of calibrations*. *Theoretical Computer Science*, 914:1–13.
- [10] V. CHAU, C. DAMERIUS, P. KLING, M. LI, F. SCHNEIDER, AND R. ZHANG (2025). *Scheduling with calibrations for multi-interval jobs*. *INFORMS Journal on Computing*. doi:10.1287/ijoc.2023.0430.