Convex and online optimization : applications to scheduling and selection problems

Victor Verdugo

THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences et Lettres
PSL Research University

**Préparée dans le cadre d'une cotutelle entre
École normale supérieure et l'Universidad de Chile**

Convex and Online Optimization: Applications to Scheduling and Selection Problems

Optimisation convexe et en ligne: applications aux problèmes de planification et de sélection

Ecole doctorale n°386
École Doctorale de Sciences Mathématiques de Paris Centre

Spécialité Informatique

COMPOSITION DU JURY :

M. Jose Correa
Universidad de Chile, Directeur de thèse

M. Christoph Dürr
Sorbonne Université, Rapporteur

M. Samuel Fiorini
Université libre de Bruxelles, Rapporteur et membre du jury

M. Rida Laraki
Université Paris Dauphine, Membre du jury

Mme. Claire Mathieu
École normale supérieure et Collège de France, Directrice de thèse

Mme. Nicole Megow
Universität Bremen, Membre du jury

M. Tobias Mömke
Universität Bremen, Membre du jury

Mme. Alantha Newman
Université Grenoble Alpes, Membre du jury

M. Fernando Ordóñez
Universidad de Chile, Membre du jury

**Soutenue par Victor Verdugo
le 13 Juin 2018**

Dirigée par
Jose Correa et Claire Mathieu

ENS ÉCOLE NORMALE SUPÉRIEURE | PSL RESEARCH UNIVERSITY PARIS | UNIVERSIDAD DE CHILE

Acknowledgments

First of all, I'm really grateful to my advisors, Claire Mathieu and Jose Correa, for all your support, guidance and patience, for those inspiring conversations about research and life. I've learned a lot from you these years about how to do research, and I'm sure I will keep learning from you in the future. Thank you!

I thank Christoph Dürr and Samuel Fiorini for carefully reviewing this work, and your helpful comments. I also thank to Rida Laraki, Nicole Megow, Tobias Mömke, Alantha Newman and Fernando Ordoñez for being part of the jury.

I had the chance to share with two very inspiring group of people, at Talgo ENS and the Acgo UChile. I thank Éric Colin de Verdiére, Hang Zhou, Zenthao Li and Vincent Cohen-Addad for all those conversations about research and french. Many thanks to Frederik Mallmann-Trenn, for all the nice moments and also for being a great collaborator. I learned a lot about randomness and algorithms working together.

I thank Bastian Bahamondes, Carlos Bonet, Antoine Hochart, Ruben Hoeksma, Tim Oosterwijk, Kevin Schewior, Marc Schröder, Andreas Tönnis and Andreas Wiese for the really nice research environment, the nice conversations at lunch or coffee, and also for our research meetings at Pepperland. Thanks to my office mates Andres Cristi, Patricio Foncea, Dana Pizarro and Raimundo Saona for all the inspiring conversations and collaboration. Thanks to Fábio Botler, for the friendship and advice, for the patience to understand my basic portuguese, and for sharing your knowledge about graph theory with me. I also thank the great people at the DSI, specially to Andrea Canales, Alvaro Brunel, Carlos Casorrán, Javier Ledezma and Dana Pizarro. Thank you all!

I thank Varun Kanade for all the advice and support, for your collaboration and for sharing your insight and knowledge in theoretical computer science. I thank Monaldo Mastrolilli for hosting me in IDSIA twice, where I also worked with Adam Kurpisz, for sharing all your expertise and knowledge about convex hierarchies. I thank Alberto Marchetti-Spaccamela for hosting me in Sapienza, to work on real time scheduling. I thank Jannik Matuschke for hosting me in Tor Vergata, and also in TU Munich, I really enjoyed and learned a lot from our research meetings about flows and combinatorial problems. I thank Tobias Mömke for hosting me at Max Planck Institute, to work on the traveling salesman problem.

I thank Jose Soto for guiding me through online selection problems, for sharing all your expertise in combinatorial optimization, and also for your advice and support.

I thank José Verschae for the support over these years, since I started my master

thesis. For sharing your knowledge on approximation and online problems, and for the motivation to work and learn together about algebra and optimization. We still have a lot to learn!

I thank Mario Bravo, Roberto Cominetti and Cristóbal Guzmán for all the great conversations about math, research and teaching.

Thanks to Fernanda Melis and Linda Valdés for all your help.

My special thanks to Victor Bucarey, for being such a great friend, for all the support over these years, for our conversations about life, football, music and research; Muchas gracias tocayo! Also my special thanks to Sebastián Reyes Riffo, for all the support and friendship, for hosting me every recent time I've been in Paris, and for sharing this great experience that is living far from home.

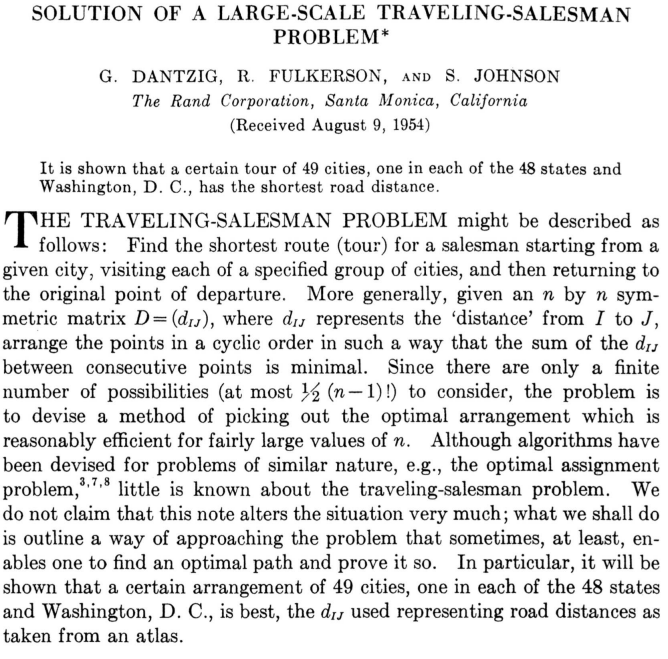
Thanks to all the amazing people and friends, Nacho, Nico, Niko, Mati, Javi, Geraldine, Joce, Felipe S., María Angélica, Daniel, Karl, Vito, Camila, Seba H., Seba Z., Alvaro, Felipe G., sorry if I forgot someone!

Finally, thanks to my parents, Victor and Marta, to my brother Bastián, my sister Estefanía and my niece Pascuala, you have been always a tremendous support. Muchas gracias por todo!

Contents

General Introduction

The difficulty of solving a combinatorial optimization problem comes in many different ways. Sometimes, the input is totally available but the number of possible solutions is so large that looking for a *good* one requires a lot of effort. In other situations the problem is the opposite: Finding a good solution is not hard but the input is only partially revealed. Over the years, techniques have been developed in different contexts that help to attack the situations above. Nevertheless, they are usually very adapted to the particular problem at hand, or the instances that are to be solved. Are there *unified* approaches to find good solutions in these many situations?



SOLUTION OF A LARGE-SCALE TRAVELING-SALESMAN PROBLEM*

G. DANTZIG, R. FULKERSON, AND S. JOHNSON
The Rand Corporation, Santa Monica, California
(Received August 9, 1954)

It is shown that a certain tour of 49 cities, one in each of the 48 states and Washington, D. C., has the shortest road distance.

THE TRAVELING-SALESMAN PROBLEM might be described as follows: Find the shortest route (tour) for a salesman starting from a given city, visiting each of a specified group of cities, and then returning to the original point of departure. More generally, given an n by n symmetric matrix $D=(d_{IJ})$, where d_{IJ} represents the 'distance' from I to J, arrange the points in a cyclic order in such a way that the sum of the d_{IJ} between consecutive points is minimal. Since there are only a finite number of possibilities (at most $\frac{1}{2}(n-1)!$) to consider, the problem is to devise a method of picking out the optimal arrangement which is reasonably efficient for fairly large values of n. Although algorithms have been devised for problems of similar nature, e.g., the optimal assignment problem,[3,7,8] little is known about the traveling-salesman problem. We do not claim that this note alters the situation very much; what we shall do is outline a way of approaching the problem that sometimes, at least, enables one to find an optimal path and prove it so. In particular, it will be shown that a certain arrangement of 49 cities, one in each of the 48 states and Washington, D. C., is best, the d_{IJ} used representing road distances as taken from an atlas.

Figure 1: Dantzig, Fulkerson and Johnson in 1954, introduced the *cutting planes* approach to find a tour passing through Washington DC and the other states. Since then, it is a unifying and successful tool for tackling large scale problems.

Input is known, problems are hard! In practical situations, usually the instances and restrictions over the search space are complicated, and a first step is to simplify reality and *model* it. For the last part, a tool that is widely used nowadays and that has become a unifying approach is *linear programming*. The goal is to find a

way in which the solutions of the combinatorial problem can be mapped to solutions satisfying some linear inequalities, and their performance is measured using a linear objective. Once a model has been found, the second step is to *solve* the linear program. Since the introduction of the *simplex method* by Dantzig in 1947 [29], obtaining a solution became practical in many contexts, and it was used intensively those years in operations research and economics, to mention a few. This development coincided with the growing of the practical necessities of finding good solutions to large problems, and with a progressive advance in the computational technologies. Today, there is an extremely rich body of methods for solving linear programs very accurately, with a reasonable computational cost, and very fast. It is now part of the toolbox in machine learning, combinatorial optimization, economics and *decision making* in general.

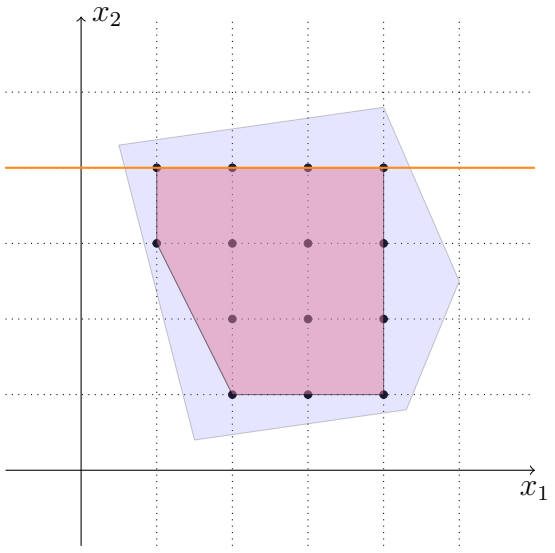


Figure 2: By introducing a Chvatal-Gomory cut, one improves the integrality gap by removing basic feasible solutions that are fractional. The constraint $x_2 \leq 4$ is valid for every integer feasible solution.

So when the question of solving a linear program is well understood, one goes back to the first step: The model. The picture becomes less clear at this point because the way a model is constructed depends heavily on the point of view adopted by *who* designed it. That is, very different models can answer the same question, but what is more important, they have direct implications over the second step: Solving. Whereas model A can be solved rapidly, model B can exhibit a poor performance in terms of accuracy and solving time. The situation turns down to be more dramatic when the search space is restricted to integer values, that is *integer programming*. In general, solving an integer program is an NP-hard problem, so the classic and very successful approach is to *relax* the search space and look for a fractional solution. How much do we lose in this step? We quantify this in the so-called *integrality gap*, that is, the ratio between the best integral solution and the best fractional solution.

At the moment of suggesting a model there are two important aspects to consider:

The size and the integrality gap. The first question can be measured easily, but the second aspect is far from being easy to estimate. The ideal model is one of reasonable size and very good integrality gap, but in most of the cases one of them should be resigned to attain the other. Many methods have been suggested the last decades to improve the integrality gap of a model and one of the most popular is the *cutting planes* approach, from the seminal work of Dantzig, Fulkerson and Johnson [28]. There, one looks for better integrality gaps by introducing new constraints that are valid for every integer solution. Chvátal in 1973 [27] showed a particular way of constructing these constraints that eventually provide a full description of the convex hull of the integer solutions. Many state-of-the-art solvers already include these methods at the moment of solving an integer program.

$$
\begin{aligned}
2x_1 + 2x_2 &\geq 1,\\
x_1^2 &= x_1,\\
x_2^2 &= x_2.
\end{aligned}
\qquad\qquad
\begin{aligned}
2x_1 + 2y_{\{1,2\}} &\geq x_1,\\
2x_2 + 2y_{\{1,2\}} &\geq x_2,\\
2x_2 - 2y_{\{1,2\}} &\geq 1 - x_1,\\
2x_1 - 2y_{\{1,2\}} &\geq 1 - x_2,\\
x_1 + x_2 - y_{\{1,2\}} &\leq 1,\\
\min\{x_1, x_2\} &\geq y_{\{1,2\}},\\
x_1, x_2, y_{\{1,2\}} &\geq 0.
\end{aligned}
$$

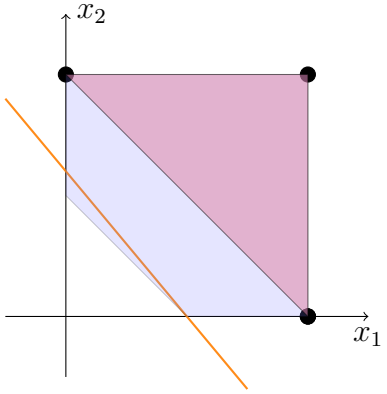


Figure 3: The set $\{(x_1, x_2) \in \{0,1\}^2 : 2x_1 + 2x_2 \geq 1\}$ can be relaxed using linear programming. This region K^0 includes fractional basic feasible solutions such as $(1/2, 0)$ and $(0, 1/2)$. By adding new variables simulating the possible products between the variables and introducing new constraints, one obtains a larger program, but is *easy* to solve. Furthermore, the projection K^1 over the original space it is strictly contained in K^0. This approach was introduced by Sherali & Adams [95]. By combining the first, third and fifth constraint at the right one can show that $2x_1 + 5x_2/3 \geq 1$ holds, which cuts the point $(0, 1/2)$.

Another way to gradually improve the integrality gap is the *lift & project* approach. The idea is to introduce many new variables in order to be able of including more complicated constraints that are valid for every integer solution. One obtains a program in a higher dimensional space, and then goes back by projecting to the variable

space of the original program. By doing this it is possible to *simulate* non-linear constraints, at the cost of incrementing the size. Eventually, by repeating this step many times it is possible to reach the convex hull of the integer solutions. The programs obtained from these methods are also based on *semidefinite programming*, which is a generalization of the linear case and can be solved efficiently. Recently these methods have attracted a lot of interest in the theoretical computer science community, since it provides a candidate for unifying many of the existing optimal algorithms. We refer to the survey of Barak and Steurer for an overview and some open problems [14].

Both of these methods are guaranteed to improve the integrality gap, but due to computational limitations it is not possible to apply them for many steps. Therefore, the key question is: How *fast* the integrality gap decreases? We try to answer this question in Part I in the particular case of a scheduling problem, which is among the most fundamental and studied combinatorial problems.

Problems are easy, input is unknown! In contrast to before, there are many combinatorial problems that are relatively simple to solve, but the difficulty relies on the availability of input data. This can be really problematic especially when decisions have to be made over time. We then say that the input is revealed *online*.

Consider the following situation. A logistics company needs to hire a data scientist to develop new technologies in their operations area. Some candidates are shortlisted to be interviewed, but due to time constraints, the time windows between them are not so short. Assuming that when an offer is made the interviewing process is finished, the question is: Who to offer? This is an *online selection problem*.

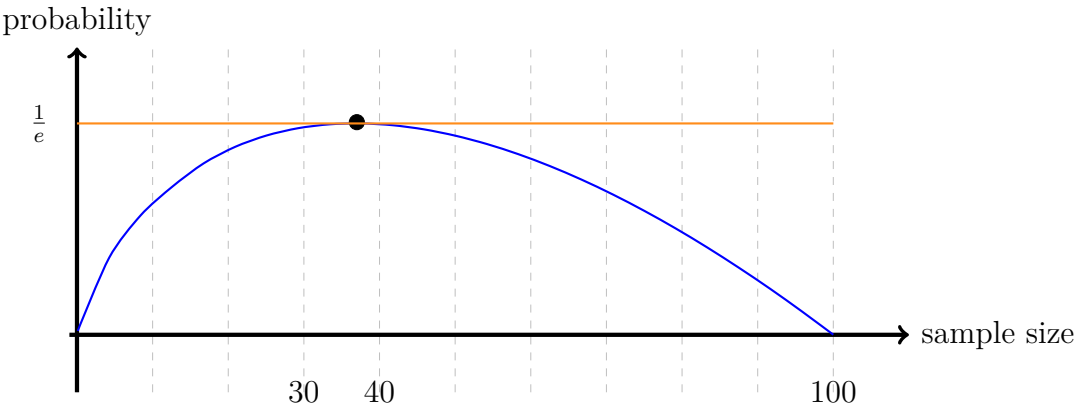


Figure 4: There are a 100 candidates. The decision maker interviews s candidates without making any offer, this is the *sampling* phase. After the sampling phase, the decision maker selects the best candidate seen so far. With probability $\approx -\frac{s}{100}\ln\left(\frac{s}{100}\right)$ the selected candidate is the best among the hundred. This probability is maximized at $s = 37 \approx 100/e$.

Sometimes at the moment of revealing an element it is also revealed a *weight* or *reward*, but more often the decision maker *ranks* the elements seen so far. The second

ingredient is the *order* in which the elements are revealed. When the order is chosen uniformly at random, the problem is widely known as the *secretary problem*. It is neither clear when the problem was stated for the first time nor who solved it. The problem appeared in the February 1960 column of *Scientific American*, but Cayley and even Kepler already thought about similar questions. It was Lindley [75] in 1961 who seems to be the first in publishing a solution in a scientific journal but posteriorly the results were extended and studied profoundly in the optimization community, optimal control, probability and in the last decade, *algorithms.*

The key algorithmic question is then how to construct *simple* and *provably good* stopping rules. Observe that under full knowledge, which is the *offline* setting, the optimal solution is just to pick the best element. If the selection constraints are more complicated, say, selecting at most three instead of only one, the offline problem is still very simple. In fact, one can see that if the selection is restricted to what is known as *matroid*, the optimal solution for the offline setting is given by the *greedy* algorithm. The Kruskal algorithm for finding minimum spanning trees is just an implementation of the greedy algorithm for a very particular matroid.

The answer to the hiring problem is the following: Interview about the 37% of the candidates without making any offer, and then make an offer to best candidate seen so far. With probability close to 0.37 the decision maker will pick the best candidate among every. The ratio between the best we could have done offline versus what we do online is called the *competitiveness* of the algorithm. Then the question is the following: Is it possible to find simple algorithms with good competitiveness for the selection problem, under combinatorial constraints, that guarantee *every* element in the optimal solution to be picked with good probability? We study this question in Part II, when the selection is restricted to matroid constraints.

Contributions of this Thesis

In the first part of this thesis we study the problem of scheduling identical machines to minimize the makespan. In the scheduling literature it is usually referred as $P\|C_{\max}$. This problem has been studied extensively from an algorithmic point of view. In particular, a *polynomial time approximation scheme*[1] exists for this problem, which is based on rounding the instance to decrease the combinatorics and then running a dynamic program. The question we try to answer is the following:

> Q_1: Is it possible to match the best approximation factors by applying *lift & project* methods over a known relaxation to reduce the integrality gap?

We show that applying these methods over the natural linear program relaxations for this problem does *not* help to reduce the integrality gap. More specifically, the

[1] For every $\varepsilon > 0$, there is an algorithm that computes a solution with cost at most $(1+\varepsilon)$opt, where opt is the cost of the optimal solution in a minimization problem. The running time of the algorithm is polynomial in the input size.

problem is modeled as an *assignment problem* for which its integrality gap is known to be 2. We answer the question negatively.

$A_1(-)$: Even if we apply *lift & project* to construct linear or semidefinite programs of exponential size, the integrality gap is not less than 1.0009.

We remark that the constant in the lower bound has not been optimized, and probably can be improved, but it is good enough for the exposition of the result. This answer, although informative, it is a bit unsatisfactory since we know this problem is *easy* from an approximation point of view. A very particular feature of the case when the machines are identical is that the relaxations we obtain are all *symmetric* respect to the action of a group: If we permute machines, we obtain other feasible schedule with the same makespan. In practice and also in theory, this is known to be harmful at the moment of optimizing. Then, the approach we consider is the following. Before applying *lift & project*, we break the machine symmetries of the ground linear program by introducing constraints. We show that in this case we are able to reduce the integrality gap, and matching the best known approximation factors.

$A_1(+)$: If we apply *lift & project* after breaking machine symmetries, we can obtain relaxations of polynomial size and with integrality gap arbitrarily close to one.

In the second part of this thesis we study the secretary problem when the selection is constrained to a matroid. Usually, the problem is studied under the existence of a weight associated to every element, and the performance of an algorithm is measured in terms of the expected weight of the algorithm selection. For many matroid families there are algorithms with constant competitiveness, but it remains as an open question whether the same can be attained for *any* matroid. We introduce a notion of competitiveness that does not assume the existence of weights, but rather assumes the ability to rank the elements seen so far at every time step. The secretary problem was originally stated in this framework, but most of the subsequent work focused on the weighted version. Our goal is to maximize the probability for which any element in the optimal solution is selected by the algorithm, that is a stronger notion of competitiveness.

Q_2: Can we design algorithms with good competitiveness for the secretary problem for this stronger notion?

We answer this question positively for many matroid families, by showing algorithms that match or beat the best known competitiveness in the weaker *weighted* notion. Furthermore, we develop a general framework for designing algorithms with constant probability competitiveness. We also design algorithms for general matroids in the the stronger notion.

A_2: There is a general framework that yields $O(1)$ probability competitiveness for many matroid families.

The first chapter of each Part I and II provide a deeper introduction and a formal exposition of the results obtained.

Publications of the Author

[1] José Correa, Patricio Foncea, Dana Pizarro, and Victor Verdugo. *From pricing to prophets, and back!* Submitted, 2017.

[2] José Soto, Abner Turkieltaub and Victor Verdugo. *Strong Algorithms for the Ordinal Matroid Secretary Problem.* In Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018.

[3] Sanjoy Baruah, Vincenzo Bonifaci, Alberto Marchetti-Spaccamela and Victor Verdugo. *A scheduling model inspired by control theory.* In Proceedings of the 25th International Conference on Real-Time Networks and Systems, RTNS 2017.

[4] Varun Kanade, Frederik Mallmann-Trenn and Victor Verdugo. *How Large Is Your Graph?* 31st International Symposium on Distributed Computing, DISC 2017.

[5] Frederik Mallmann-Trenn, Claire Mathieu and Victor Verdugo. *Skyline Computation with Noisy Comparisons.* CoRR, abs/1710.02058, 2017.

[6] José Correa, Victor Verdugo and José Verschae. *Splitting versus setup tradeoffs for scheduling to minimize weighted completion time.* Operations Research Letters, 2016.

[7] Adam Kurpisz, Monaldo Mastrolilli, Claire Mathieu, Tobias Mömke, Victor Verdugo and Andreas Wiese. *Semidefinite and Linear Programming Integrality Gaps for Scheduling Identical Machines.* Mathematical Programming. A preliminary version appeared in Integer Programming and Combinatorial Optimization, IPCO 2016.

[8] José Correa, Alberto Marchetti-Spaccamela, Jannik Matuschke, Leen Stougie, Ola Svensson, Victor Verdugo and José Verschae. *Strong LP formulations for scheduling splittable jobs on unrelated machines.* Mathematical Programming, 2015. A preliminary version appeared in Integer Programming and Combinatorial Optimization, IPCO 2014.

[9] Frans Schalekamp, René Sitters, Suzanne van der Ster, Leen Stougie, Victor Verdugo and Anke van Zuylen. *Split scheduling with uniform setup times.* Journal of Scheduling, 2015.

Part I

Convex Optimization: Applications to Scheduling

Chapter 1

Introduction

Machine scheduling is a classical family of problems in combinatorial optimization. In this paper we study the problem of scheduling a set J of n jobs on a set M of identical machines to minimize the *makespan*, i. e., the maximum completion time of a job, where each job $j \in J$ has a *processing time (or size)* p_j. A job cannot be preempted nor migrated to a different machine, and every job is released at time zero. This problem admits a *polynomial-time approximation scheme* (PTAS) [3,4,50,51,60], which is the best possible approximation result, unless $\mathsf{P} = \mathsf{NP}$, since the problem is strongly NP-hard [45]. However, there is no known algorithm based on convex relaxations that meets the results.

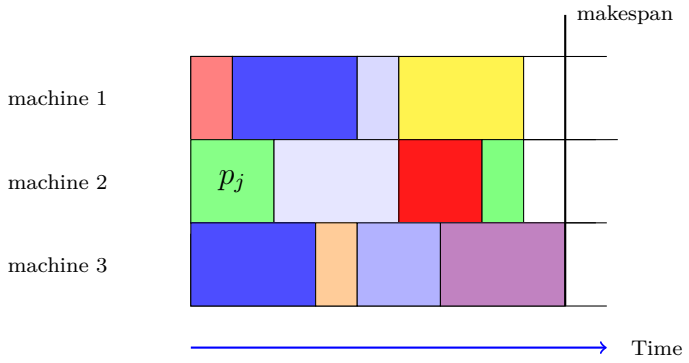


Figure 1.1: Example of a schedule on three identical machines.

A straightforward way to model the problem with a integer program is given by *assignment linear program* which has a variable x_{ij} for each combination of a machine $i \in M$ and a job $j \in J$, modeling whether job j is assigned to machine i. Every job has to be scheduled in exactly one machine. Instead of considering the makespan value as a variable that should be minimized, we guess its value T and we require that $\sum_{j \in J} x_{ij} p_j \leq T$ for each machine $i \in M$. Therefore, the relaxation of this integer

program, denoted by [assign(T)], is given by

$$\begin{aligned} \sum_{i \in M} x_{ij} &= 1 \quad \text{for all } j \in J, \\ \sum_{j \in J} x_{ij} p_j &\leq T \quad \text{for all } i \in M, \\ x_{ij} &\geq 0 \quad \text{for all } i \in M, \text{ for all } j \in J. \end{aligned}$$

1.1 The Configuration Linear Program

The assignment LP is dominated by the *configuration linear program*, or configuration LP for short, which is, to the best of our knowledge, the strongest relaxation for the problem studied in the literature [100]. Given a value $T > 0$, a *configuration* corresponds to a multiset of processing times such that its total sum does not exceed T, i. e., it is a feasible assignment for a machine when the makespan is equal to T. The multiplicity function $m(p, C)$ indicates the number of times that the processing time p appears in the multiset C. The *load* of a configuration C is just the total processing time, that is, $\sum_{p \in \{p_j : j \in J\}} m(p, C) \cdot p$. Given T, let $\mathcal{C}$ denote the set of all feasible configurations, that is, with load at most T. Observe that the definition in terms of multisets makes sense since we are working in a setting of identical machines.

For each combination of a machine $i \in M$ and a configuration $C \in \mathcal{C}$, the configuration LP has a variable y_{iC} that models whether machine i is scheduled with jobs according to configuration C. Letting n_p denote the number of jobs in J with processing time p, we can write the linear program relaxation, clp(T), given by

$$\sum_{C \in \mathcal{C}} y_{iC} = 1 \quad \text{for all } i \in M, \tag{1.1}$$

$$\sum_{i \in M} \sum_{C \in \mathcal{C}} m(p, C) y_{iC} = n_p \quad \text{for all } p \in \{p_j : j \in J\}, \tag{1.2}$$

$$y_{iC} \geq 0 \quad \text{for all } i \in M, \text{ for all } C \in \mathcal{C}. \tag{1.3}$$

We remark that in another common definition [100], a configuration is a subset, not of processing times but of jobs. We can solve that linear program to an arbitrary accuracy in polynomial time [100] and similarly our linear program above.

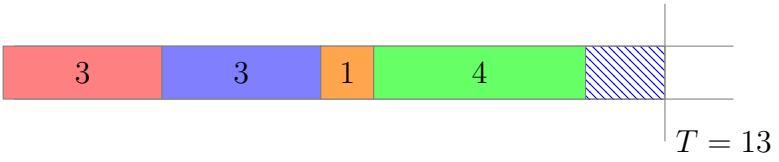


Figure 1.2: Machine scheduled according to a configuration C with makespan $T = 13$, with $m(3, C) = 2$, $m(1, C) = 1$ and $m(4, C) = 1$. The load of the configuration is $2 \cdot 3 + 1 \cdot 1 + 1 \cdot 4 = 11$.

Integrality gap. Recall the configuration LP, $\text{clp}(T)$, does not have an objective function and instead we seek to determine the smallest value T for which it is feasible. In this context, given an integer program that models the problem of scheduling identical machines and with T a fixed value of the makespan, we say that $K(T)$ is a *convex relaxation* if it is a convex set that contains every integer feasible solution. We define the *integrality gap* to be the value $\sup_{I \in \mathcal{I}} C_{\max}(I)/T^*(I)$ over all scheduling instances $\mathcal{I}$, where $C_{\max}(I)$ is the optimal makespan of the scheduling instance I and $T^*(I)$ is the minimum value T such that $K(T)$ is feasible.

With the additional constraint that $T \geq \max_{j \in J} p_j$, the assignment LP relaxation has an integrality gap of 2. This can be shown using the analysis of the list scheduling algorithm, see e. g., [102]. On the other hand, a lower bound of $2 - 1/|M|$ can be easily shown for an instance with $|M| + 1$ jobs of unit size. Naturally, the integrality gap of the configuration LP is as well upper bounded by 2, since it dominates the assignment LP. One of the first results we show is that the configuration LP has an the integrality gap which is at least 1.0009 (Chapter 2). For the more general *unrelated machines* scheduling problem, the formulation where configurations are jobs sets has an integrality gap equal to 2 [100].

1.2 Convex Hierarchies

An interesting question is whether other convex relaxations have better integrality gaps. Convex hierarchies, parameterized by a number of *levels*, *rounds* or *steps*, are systematic approaches to gradually tightening the ground relaxation, at the cost of increased running time in solving the relaxation. As a consequence, they are good candidates for designing approximation algorithms based on rounding a solution obtained from these relaxations. Given a polytope $K \subseteq [0,1]^n$, all these approaches produce a family of convex relaxations $K^1, K^2, \ldots, K^n$ satisfying that

$$K = K^0 \supseteq K^1 \supseteq K^2 \supseteq \cdots \supseteq K^{n-1} \supseteq K^n = \text{conv}(K \cap \{0,1\}^n).$$

Popular among these methods are the Sherali-Adams (SA) hierarchy [95] (Chapter 3) the Lovász-Schrijver hierarchy (LS) and its semidefinite (LS_+) counterpart [78], (Chapter 4) and the Lasserre/Sum-Of-Squares (Las) hierarchy [72, 88] (Chapter 5), which is the strongest of the three. The level r relaxations are known to be solvable in time $n^{O(r)}$, where n is the number of variables, provided some assumptions over the ground polytope. In particular, when r is constant, the complexity of solving the relaxation becomes polynomial on the program size. This is relevant when looking for an approximation algorithm based on this relaxation. For a comparison between them and their algorithmic implications we refer to [26, 73, 89]. In some settings, for example the Independent Set problem in sparse graphs [12], a hierarchy obtained from SA strenghtened with semidefinite constraints at the first level has also been considered.

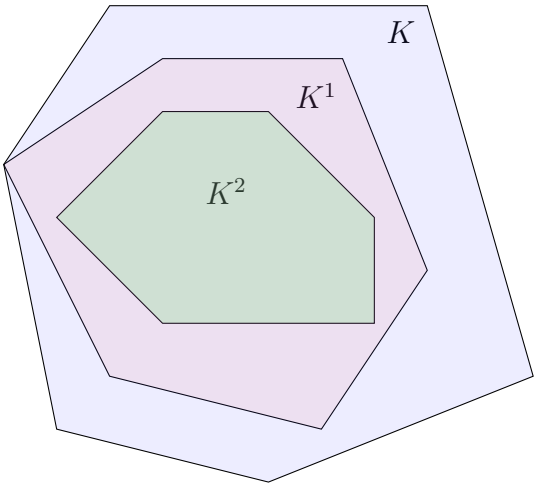


Figure 1.3: Two tightenings of a polytope using the hierarchy.

Positive results. For many problems the approximation factors of the best known algorithms match the integrality gap after performing a constant number of rounds of these hierarchies. For example, Alekhnovich, Arora and Tourlakis [2] show that one round of LS_+ yields the Goemans-Williamson [48] relaxation for Max-Cut, and the third level of LS_+ is at least as strong as the ARV relaxation for Sparsest-Cut [6]. In both cases, the base relaxation over which the hierarchy is applied corresponds to the metric defining linear program. Also for Max-Cut, Fernandez de la Vega and Mathieu [30] prove that the integrality gap of the SA hierarchy drops to $1+\varepsilon$ after $f(1/\varepsilon)$ rounds for dense graphs. For general *constraint satisfaction problems* (CSP) in its approximation version, Chan et al. [86] show that polynomial-sized linear programs are as powerful as programs arising from constant rounds of SA. In that sense, this hierarchy captures the best integrality gaps obtained by linear programming in CSP's like Max-Cut and Max-3-Sat. For the Knapsack problem, Chlamtac, Friggstad, and Georgiou [25] show that $1/\varepsilon^3$ levels of LS_+ yield an integrality gap of $1+\varepsilon$ and prove that it is possible to approximate Set-Cover using the linear relaxations of LS when the objective function is lifted into the constraints. In the scheduling context, for minimizing the makespan on two machines in the setting of unit size jobs and precedence constraints, Svensson solves the problem optimally with only one level of the linear LS hierarchy (published in [89, Section 3.1], personal communication between Svensson and the author of [89]). Furthermore, for a constant number of machines, Levey and Rothvoss give a $(1+\varepsilon)$-approximation algorithm using $(\log(n))^{\Theta(\log\log n)}$ rounds of SA hierarchy [74]. For minimizing weighted completion time on unrelated machines, one round of LS_+ leads to the current best approximation algorithm [13]. Thus, hierarchies are a strong tool for approximation algorithms.

Negative results. Nevertheless, there are known limitations on these hierarchies. Lower bounds on the integrality gap of LS_+ are known for Independent Set [37], Vertex Cover [5, 23, 46, 93], Max-3-Sat and Hypergraph Vertex Cover [2], and k-Sat [21]. For the Max-Cut problem, there are lower bounds for the SA [30] and LS_+ [93]. For the Min-

Sum scheduling problem (i. e., scheduling with job dependent cost functions on one machine) the integrality gap is unbounded even after $O(\sqrt{n})$ rounds of Lasserre [69]. In particular, that holds for the problem of minimizing the number of tardy jobs even though that problem is solvable in polynomial time, thus SDP hierarchies sometimes fail to reduce the integrality gap even on easy problems.

1.3 Lower bounds

One of the questions we try to answer in this work is the following: *Is it possible to obtain a polynomial time* $(1+\epsilon)$*-approximation algorithm based on directly applying the SA or the* LS_+ *hierarchy over the configuration LP?* This would match the best known polynomial time approximation factor known. We answer this question in the negative. We prove that even after $\Omega(n)$ rounds of SA or LS_+ to the configuration LP, where n is the number of jobs in the instance, the integrality gap of the resulting relaxation is still at least $1 + 1/1023$. Since the configuration LP dominates the assignment LP, our result also holds if we apply $\Omega(n)$ rounds of SA or LS_+ to the assignment LP.

Theorem 1. *Consider the problem of scheduling identical machines to minimize the makespan,* $P\|C_{\max}$*. For each* $n \in \mathbb{N}$ *there exists an instance with* n *jobs such that:*

(i) the configuration LP has an integrality gap of at least 1024/1023*.*

(ii) after applying $r = \Omega(n)$ *rounds of the SA hierarchy to the configuration LP the obtained relaxation has an integrality gap of at least* 1024/1023*.*

(iii) after applying $r = \Omega(n)$ *rounds of the* LS_+ *hierarchy to the configuration LP the obtained relaxation has an integrality gap of at least* 1024/1023*.*[1]

Therefore, the SA and the LS_+ hierarchies do *not* yield the best possible approximation algorithms when applied over the configuration LP. Namely, suppose there exists $r \in \mathbb{N}$ such that the r level of the SA hierarchy has an integrality gap of at most $1+\varepsilon$, with $\varepsilon < 1/1023$. In particular, that holds as well for the hard instances of Theorem 1 and $r = \Omega(n)$. We remark that for the hierarchies studied in Theorem 1, a number of rounds equal to the number of variables in $\text{clp}(T)$ suffice to bring the integrality gap down to exactly one, although this number is in general exponentially large in the input size. Nevertheless, we also prove in Chapter 3 that a number of rounds equal to number of machines suffices to reduce the integrality gap to one, when the SA hierarchy is applied over the configuration LP. This comes as a consequence of a stronger decomposition lemma, that relies on the structure of the configuration linear program.

[1] In general, the SA and LS_+ hierarchies are incomparable, so (iii) does not follow directly from (ii). The linear counterpart of Lovász-Schrijver, the LS hierarchy, is weaker than SA and therefore the result follows in that case from (ii).

We prove Theorem 1 by defining a family of instances $\{I_k\}_{k\in\mathbb{N}}$ constructed from the *Petersen graph* (see Figure 2.3). The size of instance I_k, given by the number of machines and jobs, is $\Theta(k)$ and the number of jobs is $\Theta(k)$ as well. In Chapter 2 we provide an explicit construction of the hard instances and we prove that the configuration LP is feasible for $T = 1023$ while the integral optimum has a makespan of at least 1024. In Chapter 3, we show for each instance I_k that we can define a fractional solution that is feasible for the polytope obtained by applying $\Omega(k)$ rounds of SA to the configuration LP parametrized by $T = 1023$. Finally, in Chapter 4 we prove the same for the semidefinite relaxations obtained with the LS_+ hierarchy, and we study the matrices arising in the lower bound proof.

1.4 Upper bound

The assignment and configuration LP's have in common the fact that we can permute the machines, and the solutions obtained remain feasible. The sets satisfying this property are said to be *invariant* under the action of some permutation group. In the case of these polytopes, the *symmetric group* of size equal to the number of machines is acting over the feasible solutions. The same holds for the relaxations obtained from applying SA and LS_+ over the configuration LP. This is a key fact when reducing the dimensionality of the matrices certifying the feasible solutions in Chapter 4. The results shown in Theorem 1 suggest that symmetric relaxations *do not* seem to be good candidates for obtaining small integrality gaps for the scheduling problem. Therefore, the question we study is the following: *Is it possible to obtain a polynomial sized linear or semidefinite relaxation with an integrality gap of* $(1+\varepsilon)$ *that is not invariant for the machine symmetries?* This time, we provide a positive answer. Below we state, informally, the main theorem which is proven in Chapter 5.

Theorem 2. *For every* $\varepsilon > 0$*, there exists a non-machine symmetric polynomial sized semidefinite relaxation with an integrality gap of at most* $(1+\varepsilon)$*, for the problem of scheduling to minimize the makespan.*

The theorem is based on introducing a formulation that *breaks* the symmetries in the assignment LP by adding new constraints. They enforce a very particular structure over any feasible solution of the formulation that should respect a *lexicographic* order over the machine configurations induced by any feasible integer solution. On top of the relaxation obtained from adding the aforementioned constraints, we apply the Lasserre/SoS hierarchy. In Chapter 5 we provide a full proof of the theorem, and also a direct application of it: A polynomial time approximation scheme based on solving the semidefinite relaxation.

Practice. The presence of symmetries in the formulations is known to be harmful and the literature about how to deal with them is large. Most of them focus on how to add constraints that *break* the symmetries, or how to modify the *divide and conquer* branching techniques in order to avoid unnecessary computations by approaches such

as perturbation of the numeric values defining the program and variables fixing [80, 81, 84]. Particular interest have attracted *partitioning problems* such as scheduling, packing, coloring and clustering. One way of removing the symmetries of a formulation is to consider decompositions such as *Dantzig-Wolfe*, where all the structure of a partition is hidden and the goal is to find how many times such a structure appears on a solution. In fact, the configuration LP can be seen as an intermediate step between the assignment LP and its Dantzig-Wolfe decomposition. This approach has proven to be successful in practice for many situations including the above mentioned problems in transportation, routing and coloring [16, 31, 83, 99].

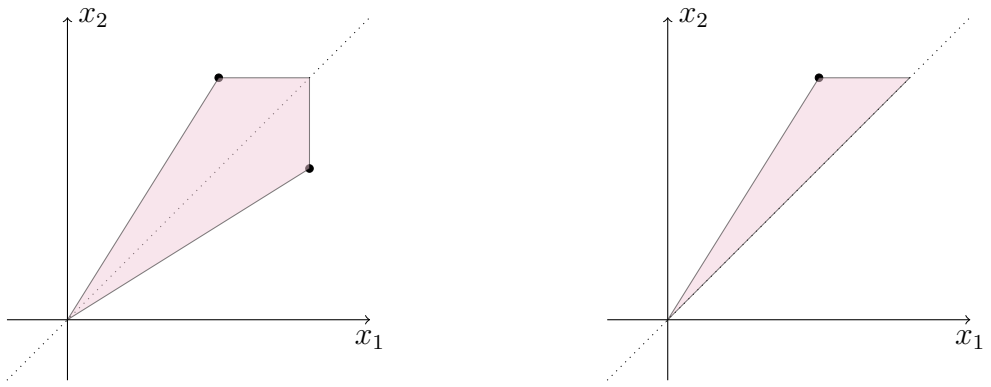


Figure 1.4: The polytope in $\mathbb{R}^2$ at the left is invariant under the action of permuting variables, that is, every time (x_1, x_2) is feasible, then (x_2, x_1) is also feasible.

Extended formulations. Similar results to Theorem 1 can be found in the context of extended formulations, that is, a convex set on a higher dimensional space that coincides with the original one when projected to the initial variables space. Remarkable are the results of Yannakakis [103] that neither the matching polytope nor the TSP polytope have symmetric linear extended formulation of subexponential size. Recently, this results were extended also for non symmetric linear extended formulations by Fiorini et al. [42] in the case of TSP and by Rothvoss [90] for matching. The same negative result follows if one consider semidefinite formulations, since Braun et al. [19] showed that any symmetric semidefinite program for matching has exponential size. They also show that every symmetric relaxation of polynomial size n^k for asymmetric TSP is not stronger than an $O(k)$ level Lasserre/SoS relaxation.

Symmetry breaking constraints. The way we obtain improvements in the gap at Theorem 2 is by adding constraints on top of the initial symmetric relaxation, which is the assignment polytope in this case, in order to obtain a program that is not invariant. The result would also follow if we consider initially the configuration LP, since one can show that the Lasserre/SoS hierarchy over the assignment LP at certain level is *stronger* than the configuration LP, provided some assumptions over the instances. The idea behind all these approaches is to remove parts of the feasible region that provide no extra information in terms of objective value and solution structure. De-

pending on the problem and the particular group action, different sets of symmetry breaking inequalities have been considered. For partitioning problems there are ways of enforcing an order over the solutions and then guaranteeing that only one solution per *orbit* is considered [43, 44, 59]. Following the same lines, in the particular case of scheduling there is a way of ensuring that exactly one representative per orbit is selected by intersecting the original partitioning polytope with the so called *orbitope* [61]. They provide a linear description of the orbitope that is of exponential size, but it can be separated efficiently. For an extensive treatment we refer to the excellent survey of Margot [82].

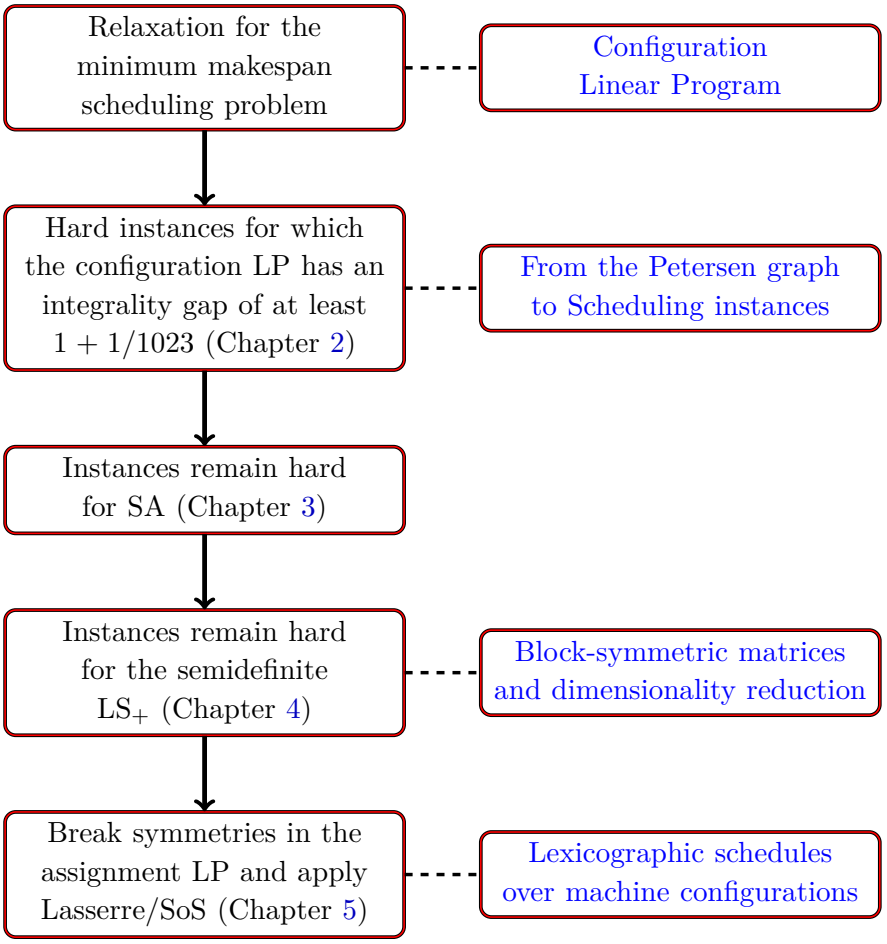


Figure 1.5: Organization of Part I.

Chapter 2

The hard instances

In this chapter we show that the configuration LP described by constraints (1.1) and (1.2) has an integrality gap of at least 1024/1023. To this end, for each $k \in \mathbb{N}$ we define an instance I_k that is inspired by the *Petersen graph* $G = (V, E)$ (see Figure 2.1) with vertex set $V = \{0, 1, \ldots, 9\}$. We first have to introduce a family of multigraphs obtained from G.

For each $k \in \mathbb{N}$, we construct a multigraph $G_k = (V, E_k)$ with vertex set equal to V and the set of edges E_k is defined as follows: For each edge $\{u, v\} \in E$ of the Petersen graph G, we have a multiset $E(\{u, v\})$ with k copies of the edge $\{u, v\}$. The set E_k is just the multiset obtained from the union of all these multisets, that is, $E_k = \cup_{\{u,v\}\in E} E(\{u, v\})$. In particular we have that $G_1 = G$, the Petersen graph.

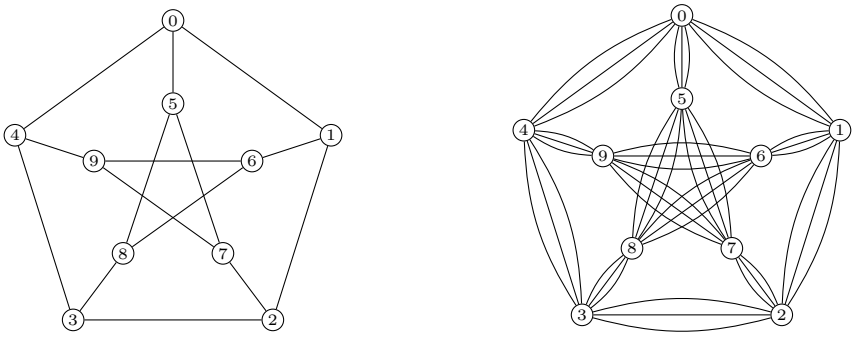


Figure 2.1: On the left, the Petersen graph G_1. On the right, the mutigraph G_3 obtained from considering 3 copies of each edge in the Petersen graph.

The scheduling instances. In the instance I_k we have a job j_e for every $e \in E_k$, which is the set of edges in the multigraph G_k. Thus I_k has $15k$ jobs. Let $e \in E_k$ be such that $e \in E(\{u, v\})$, that is, e is an edge between nodes u and v. The processing time of j_e is $p_{j_e} = 2^u + 2^v$. We define the set of machines for I_k to be $[3k] = \{1, \ldots, 3k\}$.

Observe that every subset of edges $F \subseteq E_k$ induces a multiset C_F of job-sizes as follows: For every $\{u, v\} \in E$ and $p = 2^u + 2^v$, we have $m(p, C_F) = |F \cap E(\{u, v\})|$. In words, for every possible processing time, we check how many copies of the corresponding edge in G are in F, and this number is the multiplicity of the processing time

in the multiset C_F. If the load of C_F is at most T, we say that C_F is the *configuration induced* by F. We remark that this mapping is not injective, since different subsets of edges may induce the same processing times multiplicities and therefore the same configuration.

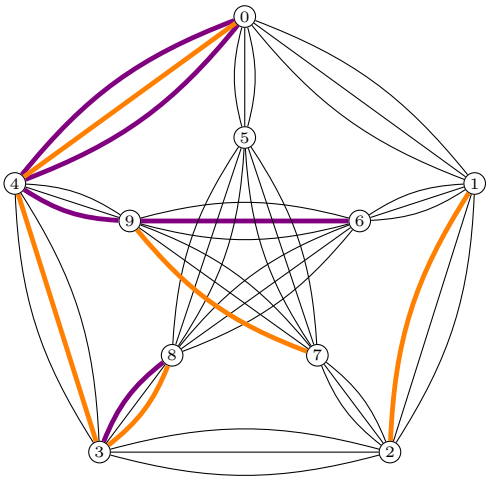


Figure 2.2: The edges in purple induce the configuration $\{2^0+2^4, 2^0+2^4, 2^4+2^9, 2^9+2^6, 2^3+2^8\}$ and the orange edges induce the configuration $\{2^0+2^4, 2^4+2^3, 2^3+2^8, 2^9+2^7, 2^2+2^1\}$. The first has a load of 1402 and the second has a load of 951.

Matching configurations. The Petersen graph G has exactly six perfect matchings $\beta_1, \ldots, \beta_6$. We have that the sum of the job sizes in a perfect matching β_ℓ is

$$\sum_{e \in \beta_\ell} p_{j_e} = \sum_{u=0}^{9} 2^u \deg_{\beta_\ell}(u) = 1023,$$

since $\deg_{\beta_\ell}(u) = 1$ for every node $u \in V$. Therefore, β_ℓ induces a configuration C_ℓ with load 1023 in the following way: $C_\ell = \{2^u + 2^v : \{u, v\} \in \beta_\ell\}$, that is, for every edge $e \in \beta_\ell$ we have in C_ℓ one copy of a job with processing time p_{j_e} defined as above. The configurations set $\mathcal{C}^\beta = \{C_1, \ldots, C_6\}$ are called *matching configurations*.

It can be checked easily that is not possible to find a partition of the edges in the Petersen graph such that every part is a perfect matching, i. e., there is no *1-factorization* of the Petersen graph. This property is a key ingredient for proving the lower bounds on the integrality gaps of the different relaxations we consider for the scheduling problem. We extend the definition of 1-factorization to multigraphs in the natural way. In the following lemma we show an infinite sequence of multigraphs in $\{G_k\}_{k \in \mathbb{N}}$ for which there is no 1-factorization.

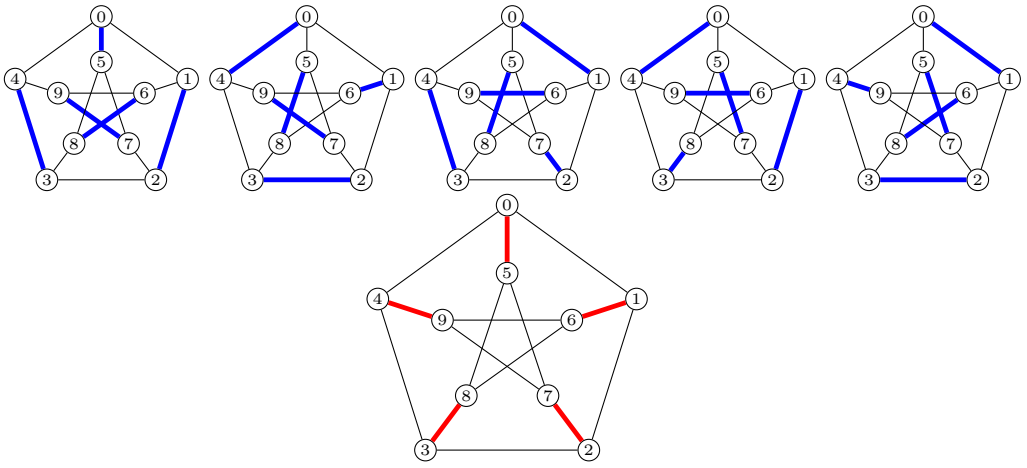


Figure 2.3: The Petersen graph and its six perfect matchings (coloured edges). Observe that the blue matchings are isomorphic.

Lemma 1. *For every odd $k \in \mathbb{N}$, there is no 1-factorization of the multigraph G_k.*

Proof. Let β_6 be the perfect matching of the Petersen graph consisting of the five edges $\{0,5\},\{1,6\},\{2,7\},\{3,8\}$ and $\{4,9\}$, called *spokes* (last matching at Figure 2.3). Suppose there exists a 1-factorization of G_k, and let ω_6 be the number of times that the perfect matching β_6 appears in the 1-factorization. In particular, the size of the 1-factorization is $3k$, since there are $15k$ edges in G_k and every perfect matching has exactly 5 edges.

Each spoke, which appears in exactly one other perfect matching in $\{\beta_1,\ldots,\beta_5\}$, must be contained in exactly k perfect matchings of the 1-factorization. Therefore, for all $j \in [5]$, the perfect matching β_j appears $k-\omega_6$ times in the 1-factorization. Thus, in total the size of the 1-factorization is $5(k-\omega_6)+\omega_6 = 5k-4\omega_6$. However, that sum equals $3k$, and so $\omega_6 = k/2$. Since k is odd and ω_6 an integer, the contradiction follows. □

2.1 Integrality gap for clp: Proof of Theorem 1(i)

In this section we prove that the integrality gap of the configuration LP is at least 1024/1023. In fact, we prove a stronger statement: For every $k \in \mathbb{N}$, the gap of the configuration LP at instance I_k is at least 1024/1023. Given I_k, we proceed in two steps: We first prove that the configuration LP with $T = 1023$ is feasible for I_k, and secondly we prove that the makespan of I_k is at least 1024.

Lemma 2. *For every $k \in \mathbb{N}$, the configuration linear program for $T = 1023$ is feasible at instance I_k.*

Proof. We define a fractional solution that is supported only by matching configurations: For every machine $i \in [3k]$ and each $\ell \in \{1,2,\ldots,6\}$ we set $y_{iC_\ell} = 1/6$. For

every machine $i \in [3k]$ and every configuration $C \in \mathcal{C} \setminus \mathcal{C}^\beta$ we set $y_{iC} = 0$. The set of machine constraints (1.1) at $\mathrm{clp}(T)$ is clearly satisfied, since for every $i \in [3k]$ we have

$$\sum_{C \in \mathcal{C}} y_{iC} = \sum_{\ell=1}^{6} y_{iC_\ell} = 6 \cdot 1/6 = 1.$$

For the set of job-size constraints (1.2), recall that for every $e = \{u, v\} \in E$ and $p = 2^u + 2^v$, in I_k we have that $n_p = k$. The Petersen graph is such that there are exactly two perfect matchings β_1^p, β_2^p containing e, and therefore $m(p, C_1^p) = m(p, C_2^p) = 1$, where C_1^p and C_2^p are the matching configurations induced by β_1^p and β_2^p, respectively. Thus, we get

$$\sum_{i=1}^{3k} \sum_{C \in \mathcal{C}} m(p, C) y_{iC} = \sum_{i=1}^{3k} (y_{iC_1^p} + y_{iC_2^p}) = 3k \cdot (1/6 + 1/6) = k,$$

and so y is feasible. □

Lemma 3. *For every odd $k \in \mathbb{N}$, the optimal makespan for I_k is at least 1024.*

Proof. We first show that the optimal makespan is at least 1023. The total size of the jobs in the instance is equal to

$$\sum_{e \in E_k} p_{j_e} = \sum_{u=0}^{9} 2^u \deg_{G_k}(u) = 1023 \cdot 3k,$$

since $\deg_{G_k}(u) = 3k$ for every $u \in \{0, \ldots, 9\}$. There are $3k$ machines, so if we had a schedule with makespan strictly less than 1023 it would imply that the total load is strictly less than $3k \cdot 1023$, which is a contradiction. In particular, the optimal integral makespan for I_k is at least 1023.

Suppose that the optimal makespan is equal to 1023. By the argument above, it implies that every machine is scheduled with a configuration of load equal to 1023. Let C be a configuration with load equal to 1023, and let $F \subseteq E_k$ such that C is the configuration induced by F.

Claim 4. *The set F is a perfect matching in G_k.*

In particular, C is a matching configuration. Since every job has to be assigned to some machine, the schedule is induced by a partition of the edges in E_k where every part is a perfect matching, i. e., a 1-factorization of G_k. However, this is not possible, since by Lemma 1 there is no 1-factorization of G_k when k is odd. Therefore, we have a contradiction and we conclude that the optimal makespan is at least 1024. □

Proof of Claim 4 in Lemma 3. The load of configuration C is

$$1023 = \sum_{u=0}^{9} 2^u \deg_F(u) = \deg_F(0) + 2 \sum_{u=1}^{9} 2^{u-1} \deg_F(u).$$

In particular, the last equality implies that $\deg_F(0)$ is odd. By induction on u it must be that for every $u \in \{0, \ldots, 9\}$, $\deg_F(u)$ is odd. Since the sum does not exceed $1023 = \sum_{u=0}^{9} 2^u$, it follows that $\deg_F(u) = 1$ for every $u \in \{0, \ldots, 9\}$ and so F is a perfect matching in G_k. □

Chapter 3

Sherali-Adams (SA) Hierarchy

Convex hierarchies provide a way of obtaining gradually stronger relaxations for an integer program. In order to design approximation algorithms based on them it is crucial to understand how fast the gap decreases. In this chapter we study relaxations obtained from the Sherali-Adams (SA) hierarchy over the configuration LP, and we prove that the hard instances shown in Chapter 2 remain hard even after a linear number of rounds of the SA hierarchy.

The SA hierarchy is based on linear programming and basically works by adding new variables and constraints to a linear program in a higher dimensional space and then projecting back to the original variables space. We introduce the relaxations obtained by applying the hierarchy over the configuration LP in a self-contained way, that is equivalent to the approach in the original work of Sherali & Adams [95]. We revisit and/or introduce the necessary properties for our purposes.

In the configuration LP, $\mathrm{clp}(T)$, the variables set is $M \times \mathcal{C}$. The level r Sherali-Adams relaxation, $\mathrm{SA}^r(\mathrm{clp}(T))$, is a polytope in $[0,1]^{\mathcal{P}_{r+1}(M\times\mathcal{C})}$ where $\mathcal{P}_{r+1}(M \times \mathcal{C}) = \{A \subseteq M \times \mathcal{C} : |A| \leq r+1\}$. It is defined by the following set of constraints:

$$\sum_{C\in\mathcal{C}} y_{H\cup\{(i,C)\}} = y_H \qquad \text{for all } i \in M, \text{ for all } H \in \mathcal{P}_r(M \times \mathcal{C}), \tag{3.1}$$

$$\sum_{i\in M}\sum_{C\in\mathcal{C}} m(p,C) y_{H\cup\{(i,C)\}} = n_p y_H \quad \text{for all } p \in \{p_j : j \in J\}, \text{ for all } H \in \mathcal{P}_r(M \times \mathcal{C}), \tag{3.2}$$

$$y_H \geq 0 \qquad \text{for all } H \in \mathcal{P}_{r+1}(M \times \mathcal{C}), \tag{3.3}$$

$$y_\emptyset = 1. \tag{3.4}$$

We denote by $\mathrm{SA}^r_{\mathrm{proj}}(\mathrm{clp}(T))$ the projection in $\mathbb{R}^{M\times\mathcal{C}}$ of the polytope above. The properties summarized in the lemma below are common to most of the hierarchies considered in the literature. For the sake of completeness, we show a self-contained proof in the context of the configuration LP.

Lemma 5. *Let $r \in \mathbb{N}$. For every scheduling instance and every $T > 0$, the following holds:*

(i) $clp(T) \cap \{0,1\}^{M\times\mathcal{C}} \subseteq SA^r_{proj}(clp(T))$.

(ii) $SA^{r+1}_{proj}(clp(T)) \subseteq SA^{r}_{proj}(clp(T))$.

(iii) *Let* $y \in SA^{r}(clp(T))$. *If* $K \in \mathcal{P}_{r+1}(M \times \mathcal{C})$ *and* $H \subseteq K$, *then* $y_K \leq y_H$.

The first property shows that SA provides relaxations of the integer solutions of the configuration LP. The second property guarantees that at every step we obtain a relaxation at least as strong than the previous one. Finally, the third property reveals some consistency on the relaxation values. In order to gain intuition, it is useful to think off the value y_K as the *probability that every variable in* K *is set to one.*

Proof of Lemma 5. Consider a scheduling instance and $T > 0$.

(i) Let $z \in \text{clp}(T) \cap \{0,1\}^{M \times \mathcal{C}}$. For every non-empty $H \in \mathcal{P}_{r+1}(M \times \mathcal{C})$ we set $y_H = \prod_{(i,C) \in H} z_{iC}$, and $y_\emptyset = 1$. Clearly, the set of constraints in (3.3) and (3.4) are satisfied. Fix a set $H \in \mathcal{P}_{r+1}(M \times \mathcal{C})$. For every $i \in M$ we have that

$$\sum_{C \in \mathcal{C}} y_{H \cup \{(i,C)\}} - y_H = y_H \left(\sum_{C \in \mathcal{C}} z_{iC} - 1 \right).$$

Since $z \in \text{clp}(T)$ it follows that $\sum_{C \in \mathcal{C}} z_{iC} - 1 = 0$ and therefore the machine constraints in (3.1) are all satisfied. Analogously, for every $p \in \{p_j : j \in J\}$,

$$\sum_{i \in M} \sum_{C \in \mathcal{C}} m(p,C) y_{H \cup \{(i,C)\}} - n_p y_H = y_H \left(\sum_{i \in M} \sum_{C \in \mathcal{C}} m(p,C) z_{iC} - n_p \right),$$

and since $z \in \text{clp}(T)$ we have $\sum_{i \in M} \sum_{C \in \mathcal{C}} m(p,C) z_{iC} - n_p = 0$. Therefore, the job-size constraints (3.2) are satisfied.

(ii) Let $y \in \text{SA}^{r+1}(\text{clp}(T))$. By definition of the relaxation at level $r+1$, y satisfies every constraint at level r and therefore the restriction of y to $\mathcal{P}_{r+1}(M \times \mathcal{C})$ is feasible for $\text{SA}^r(\text{clp}(T))$. Since every singleton in $M \times \mathcal{C}$ belongs to $\mathcal{P}_{r+1}(M \times \mathcal{C})$, the lemma follows.

(iii) It is enough to prove it when $K = H \cup \{(i,R)\}$ for some machine $i \in M$ and some configuration $R \in \mathcal{C}$. Since $y \geq 0$, the machine constraints (3.1) imply that

$$y_H = \sum_{C \in \mathcal{C}} y_{H \cup \{(i,C)\}} \geq y_{H \cup \{(i,R)\}} = y_K.$$ □

3.1 Machine Decomposition Lemma

Given a polytope, the minimum number of rounds that are necessary to guarantee the convergence to the convex hull of the integer solutions is known as *rank* of the hierarchy [7, 21, 24, 73]. In general, the rank of a polytope in $[0,1]^E$ in the Sherali-Adams hierarchy is upper bounded by $|E|$ (see e. g. [95]). In this case, $|E| = |M \times \mathcal{C}| = m|\mathcal{C}|$, which is exponentially large on the input size. Instead we get an upper bound on the rank of the configuration LP which is linear in the input size.

Theorem 3. *For every scheduling instance with m machines and every $T > 0$,*

$$SA^m_{proj}(clp(T)) = conv\left(clp(T) \cap \{0,1\}^{M\times\mathcal{C}}\right).$$

The theorem above is a direct consequence of a lemma we state next. It relies heavily on the structure of the configuration LP. It can also be proved in terms of the stronger Lasserre/SoS hierarchy by means of the *Decomposition Theorem* [62], but we show that the relaxations obtained from the Sherali-Adams hierarchy are strong enough to get the desired bound on the rank.

Lemma 6. *Let $r \in \mathbb{N}$ and $S \subseteq M$ be a subset of machines with $r \geq |S|$. Then,*

$$SA^r_{proj}(clp(T)) \subseteq conv\left(SA^{r-|S|}_{proj}(clp(T)) \cap \{0,1\}^{S\times\mathcal{C}}\right).$$

In words, the lemma says that we can decompose a solution at level r into a convex combination of solutions at level $r - |S|$ and all of them are integral at the variables for the machines in S. When the number of machines is fixed, i. e., not part of the input, we obtain the following direct corollary from Theorem 6.

Corollary 7. *Consider the problem of scheduling identical machines to minimize the makespan with a fixed number of machines equal to m. Then, the $m = O(1)$ level of the SA hierarchy over the configuration LP has an integrality gap equal to 1.*

Proof of Theorem 3. The convexity of the set $\mathrm{SA}^m_{\mathrm{proj}}(\mathrm{clp}(T))$ and Lemma 5(i) imply that $\mathrm{SA}^m_{\mathrm{proj}}(\mathrm{clp}(T)) \supseteq \mathrm{conv}\left(\mathrm{clp}(T) \cap \{0,1\}^{M\times\mathcal{C}}\right)$. The other inclusion comes from applying Lemma 6 with $S = M$ and $r = m$. □

Given $r \in \mathbb{N}$, we prove Lemma 6 by induction on the cardinality of S. Before proceeding with the proof we need to introduce a technical lemma about the relaxations obtained from Sherali-Adams over the configuration LP. Let $y \in \mathrm{SA}^r(\mathrm{clp}(T))$ and consider a single machine $i \in M$. If $y_{\{(i,C)\}} \in (0,1)$ for some configuration C, we say that y is *fractional at machine i*, otherwise we say that y is *integral at machine i*. For every $C \in \mathcal{C}$ such that $y_{\{(i,C)\}} \in (0,1)$, consider the vector defined by

$$y(i,C)_H = \frac{y_{H\cup\{(i,C)\}}}{y_{\{(i,C)\}}}$$

for every $H \in \mathcal{P}_r(M\times\mathcal{C})$. Observe that constraints (3.1) guarantee that $\sum_{C\in\mathcal{C}} y_{\{(i,C)\}} = 1$. Furthermore,

$$y = \sum_{\substack{C\in\mathcal{C}:\\ y_{\{(i,C)\}}\in(0,1)}} y_{\{(i,C)\}}\, y(i,C),$$

and therefore y is a convex combination of the vectors in $\{y(i,C) : C \in \mathcal{C}, y_{\{(i,C)\}} \in (0,1)\}$. The vector $y(i,C)$ is said to be the *conditioning* of y at (i,C). The following lemma is the key for the inductive step.

Lemma 8. *Suppose that y is fractional at machine $i \in M$. Then, for every $C \in \mathcal{C}$ such that $y_{\{(i,C)\}} \in (0,1)$, we have $y(i,C) \in SA^{r-1}(clp(T))$ and $y(i,C)$ is integral at machine i.*

Proof. Observe that $y(i,C)_{\{(i,C)\}} = y_{\{(i,C)\}\cup\{(i,C)\}}/y_{\{(i,C)\}} = 1$. The feasibility of $y(i,C)$ in $\mathrm{clp}(T)$ implies that $y(i,C)$ is integral at machine i.

Now we have to verify that $y(i,C)$ satisfies the constraint at the level $r-1$. Let $H \in \mathcal{P}_{r-1}(M \times \mathcal{C})$ and consider a machine $\ell \in M$. We have that $|H \cup \{(i,C)\}| \leq r$, so from the feasibility of y at level r we obtain that

$$\sum_{R\in\mathcal{C}} y(i,C)_{H\cup\{(\ell,R)\}} = \frac{1}{y_{\{(i,C)\}}} \sum_{R\in\mathcal{C}} y_{H\cup\{(\ell,R)\}\cup\{(i,C)\}} = \frac{y_{H\cup\{(i,C)\}}}{y_{\{(i,C)\}}} = y(i,C)_H,$$

and thus the machine constraints (3.1) are satisfied. Consider a job size $p \in \{p_j : j \in J\}$. Analogously as before, from the feasibility of y at level r we have that

$$\begin{aligned}\sum_{\ell\in M}\sum_{R\in\mathcal{C}} m(p,R)y(i,C)_{H\cup\{(\ell,R)\}} &= \frac{1}{y_{\{(i,C)\}}} \sum_{\ell\in M}\sum_{R\in\mathcal{C}} m(p,R)y_{H\cup\{(\ell,R)\}\cup\{(i,C)\}} \\ &= n_p \frac{y_{H\cup\{(i,C)\}}}{y_{\{(i,C)\}}} = n_p y(i,C)_H,\end{aligned}$$

and then the job-size constraints (3.2) are satisfied. The non-negativity of $y(i,C)$ is clear and $y(i,C)_\emptyset = y_{\emptyset\cup\{(i,C)\}}/y_{\{(i,C)\}} = 1$. That finishes the proof. □

In particular, this implies Lemma 6 when $S = \{i\}$ and since it does not depend on i, it holds true for every $S \subseteq M$ of cardinality one. Observe that if y was integral at machine i then the lemma follows by the nested property at Lemma 5(ii). Another important property of the conditioning is the following: If the vector y was integral at some machine ℓ, then after conditioning the vector obtained remains integral at machine ℓ. To check this, suppose that $y_{\{(\ell,S)\}} = 1$. It is enough to check that $y(i,C)_{\{(\ell,R)\}} = 0$ for every $R \neq S$, since the machine constraints (3.1) imply then $y(i,C)_{\{(\ell,S)\}} = 1$. By Lemma 5(iii), we have that

$$y(i,C)_{\{(\ell,R)\}} = \frac{y_{\{(\ell,R)\}\cup\{(i,C)\}}}{y_{\{(i,C)\}}} \leq \frac{y_{\{(\ell,R)\}}}{y_{\{(i,C)\}}} = 0.$$

Proof of Lemma 6. In the following assume $r \geq 2$. Let $S \subseteq M$ of cardinality at least 2 and $y \in \mathrm{SA}^r(\mathrm{clp}(T))$. Consider the set $\tilde{S} = S \setminus \{\ell\}$ for some $\ell \in S$. By induction, there exists a family of vectors $\{y^\theta\}_{\theta\in\Theta} \subseteq \mathrm{SA}^{r-|\tilde{S}|}(\mathrm{clp}(T)) = \mathrm{SA}^{r-|S|+1}(\mathrm{clp}(T))$ such that y can obtained as a convex combination of them, and for every $\theta \in \Theta$ the vector y^θ is integral at every machine in $\tilde{S}$.

In the following we prove that for every $\theta \in \Theta$, the vector y^θ restricted to $\mathcal{P}_{r-|S|+1}(M \times \mathcal{C})$ can be obtained as a convex combination of vectors in $\mathrm{SA}^{r-|S|}(\mathrm{clp}(T))$ and they are all integral at every machine in $\tilde{S} \cup \{\ell\} = S$. For that end we make use of our technical Lemma 8. As explained before, if y^θ is integral at machine ℓ then we are done. Suppose then that y^θ is fractional at machine ℓ and consider the family of

vectors $\{y^\theta(\ell, C) : C \in \mathcal{C}, y^\theta_{\{(\ell,C)\}} \in (0,1)\}$ as defined before. By Lemma 8, they are all in $\mathrm{SA}^{(r-|S|+1)-1}(\mathrm{clp}(T)) = \mathrm{SA}^{r-|S|}(\mathrm{clp}(T))$ and they are all integral at machine ℓ. Since the vector y^θ was integral at $\tilde{S}$, the vectors obtained from conditioning remain integral there and therefore they are all integral at S, which concludes the proof. □

3.2 Integrality gap for SA: Proof of Theorem 1(ii)

We show that for the family of instances $\{I_k\}_{k\in\mathbb{N}}$ defined in Section 2.1, if we apply $O(k)$ rounds of the Sherali-Adams hierachy to the configuration LP for $T = 1023$, then the resulting relaxation is feasible. Thus, after $\Omega(k)$ rounds of SA the configuration LP still has an integrality gap of at least 1024/1023 for an instance with $\Theta(k)$ jobs and machines.

We can observe that the configuration LP computes a set of edges in a complete bipartite graph with vertex sets M and $\mathcal{C}$. The edges are selected such that each node in M is incident to at most one selected edge. More specifically, consider the complete directed bipartite graph with vertex sets M and $\mathcal{C}$, which is identified with $M \times \mathcal{C}$. We say that $F \subseteq M \times \mathcal{C}$ is an *M-matching* if $|\{C \in \mathcal{C} : (i, C) \in F\}| = \deg_F(i) \leq 1$. We say that $i \in M$ is *incident* to F if $\deg_F(i) = 1$. The notation extends if the configurations set is not $\mathcal{C}$ but a subset of it.

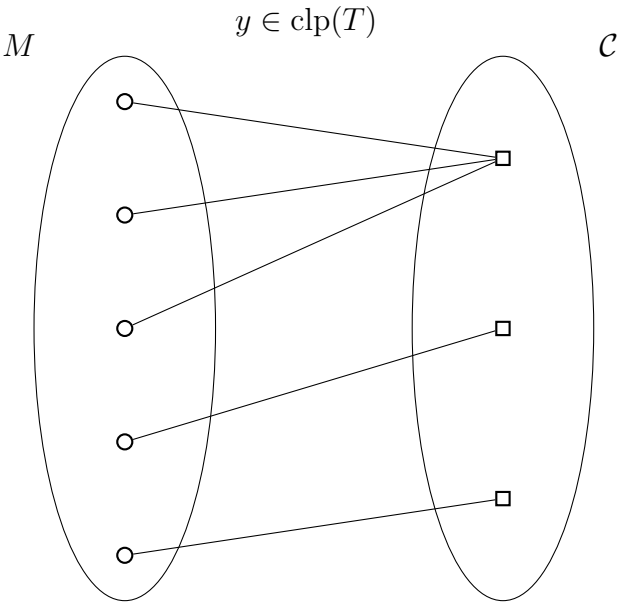


Figure 3.1: Feasible schedule given by an integral solution of the configuration LP. Example with five machines, three of them in the same configuration. The degree of every machine node is exactly one, as specified by the constraints (1.1).

In the following we consider the family of instances $\{I_k \colon k \in \mathbb{N}, k \text{ is odd}\}$ as in Section 2.1 and $T = 1023$. For any set S, let $\mathcal{P}(S)$ be the power set of S. We define

a solution to $\mathrm{SA}^r(\mathrm{clp}(T))$ for $T = 1023$. To this end, we need to provide a value y_A for each set $A \in \mathcal{P}_{r+1}(M \times \mathcal{C})$. These values are given by the following function: Let $\phi : \mathcal{P}(M \times \mathcal{C}^\beta) \to \mathbb{R}$ be such that

$$\phi(A) = \frac{1}{(3k)_{|A|}} \prod_{j \in [6]} (k/2)_{\deg_A(C_j)}$$

if A is an M-matching, and zero otherwise, where $(x)_0 = 1$ and $(x)_a = x(x-1)\cdots(x-a+1)$ if $a \geq 1$ is integer positive. This is called the *lower factorial function*. To get some understanding about how the ϕ works, it could be useful to think on a *probabilistic* interpretation that is formalized in the lemma below: Suppose we know that a set A is chosen (i.e., we *condition* on this), then the conditional probability that a pair (i, C_j) is chosen equals $(k/2 - \deg_A(C_j))/(3k - |A|)$ when $A \cup \{(i, C_j)\}$ is an M-matching.

Lemma 9. *Let $A \subseteq M \times \mathcal{C}^\beta$ be an M-matching of size at most $3k-1$. If $i \in M$ is not incident to A, then for every $j \in [6]$,*

$$\phi(A \cup \{(i, C_j)\}) = \phi(A) \frac{k/2 - \deg_A(C_j)}{3k - |A|}.$$

Proof. Given that i is not incident to A, we have $|A \cup \{(i, C_j)\}| = |A| + 1$. Furthermore, for $\ell \neq j$ we have that $\deg_{A \cup \{(i,C_j)\}}(C_\ell) = \deg_A(C_\ell)$ and $\deg_{A \cup \{(i,C_j)\}}(C_j) = \deg_A(C_j) + 1$. □

We are ready now to define our solution to $\mathrm{SA}^r(\mathrm{clp}(T))$. It is the vector $y^\phi \in \mathbb{R}^{\mathcal{P}_{r+1}(M \times \mathcal{C})}$ defined such that $y^\phi_A = \phi(A)$ if A is an M-matching in $M \times \mathcal{C}^\beta$, and zero otherwise.

Lemma 10. *For every odd k, y^ϕ is a feasible solution in $SA^r(clp(T))$ for the instance I_k when $r = \lfloor k/2 \rfloor$ and $T = 1023$.*

We note that in the proof above, the projection of y^ϕ onto the space of the configuration LP is exactly the fractional solution from Lemma 2. The proof of Theorem 1(ii) now follows directly from the lemma above.

Proof of Theorem 1(ii). Let k be such that $n = 15k + \ell$ where k is the greatest odd integer such that $15k \leq n$. The theorem follows by considering the instance I_k as defined before, $T = 1023$ and $r = \lfloor k/2 \rfloor$. □

Proof of Lemma 10. The non-negativity follows by checking that the lower factorial in the definition of ϕ remains non-negative for $r = \lfloor k/2 \rfloor$. It is also clear from the definition that $y^\phi_\emptyset = \phi(\emptyset) = 1$.

We next prove that y^ϕ satisfies the machine constraints (3.1) in $\mathrm{SA}^r(\mathrm{clp})$. If i is a machine incident to H, then all terms in the left-hand summation are 0 except for

the unique pair (i, C) that belongs to H, so the sum equals y^ϕ_H. If i is not incident to H, then by Lemma 9 we have

$$\sum_{C\in\mathcal{C}} y^\phi_{H\cup\{(i,C)\}} = \frac{\phi(H)}{3k-|H|}\sum_{j=1}^{6}(k/2-\deg_H(C_j)) = \phi(H) = y^\phi_H,$$

since $6\cdot k/2 = 3k$ and $\sum_{j=1}^{6}\deg_H(C_j) = |H|$. Finally we prove that y^ϕ satisfies the set of constraints (3.2) for every processing time. Fix p and H. Since y^ϕ is supported by six configurations, we have

$$\sum_{i\in M}\sum_{C\in\mathcal{C}} m(p,C)y^\phi_{H\cup\{(i,C)\}} = \sum_{i\in M}\sum_{j=1}^{6} m(p,C_j)\phi(H\cup\{(i,C_j)\}).$$

There are exactly two configurations $C^p_1, C^p_2 \in \mathcal{C}^\beta$ such that $m(p,C^p_1) = m(p,C^p_2) = 1$, and for the others it is zero, so

$$\sum_{j=1}^{6} m(p,C_j)\phi(H\cup\{(i,C_j)\}) = \phi(H\cup\{(i,C^p_1)\}) + \phi(H\cup\{(i,C^p_2)\}).$$

Let $\pi_M(H) = \{i\in M : \deg_H(i) = 1\}$ be the subset of machines incident to H. We split the sum over $i\in M$ into two parts, $i\in\pi_M(H)$ and $i\notin\pi_M(H)$. For the first part,

$$\sum_{i\in\pi_M(H)}(\phi(H\cup\{(i,C^p_1)\}) + \phi(H\cup\{(i,C^p_2)\})) = \phi(H)(\deg_H(C^p_1)+\deg_H(C^p_2))$$

since $\phi(H\cup\{(i,C^p_1)\})$ is either $\phi(H)$ or 0 depending on whether $(i,C^p_1)\in H$, and the same holds for C^p_2. For the second part, using Lemma 9 we have that for $\ell\in\{1,2\}$,

$$\begin{aligned}\sum_{i\notin\pi_M(H)}\phi(H\cup\{(i,C^p_\ell)\}) &= \frac{\phi(H)}{3k-|H|}\sum_{i\notin\pi_M(H)}(k/2-\deg_H(C^p_\ell))\\ &= \phi(H)(k/2-\deg_H(C^p_\ell)),\end{aligned}$$

since $|H\setminus\pi_M(H)| = 3k-|H|$. Thanks to cancellations, we get precisely what we want,

$$\begin{aligned}\sum_{i\in M}\sum_{C\in\mathcal{C}} m(p,C)y^\phi_{H\cup\{(i,C)\}} &= \sum_{\ell\in\{1,2\}}\phi(H)(\deg_H(C^p_\ell)+k/2-\deg_H(C^p_\ell))\\ &= k\cdot\phi(H) = n_p y^\phi_H.\end{aligned}$$

□

Chapter 4

Lovász-Schrijver (LS_+) Hierarchy

In contrast to the last chapter, we introduce the Lovász-Schrijver hierarchy in a general context. This is due to the simplicity behind the construction of the relaxations. In what follows, if $y = (a, x) \in \mathbb{R} \times \mathbb{R}^d$, we denote $y_\emptyset = a$ and $y_i = x_i$ for all $i \in [d]$. Consider the operator N_+ on the family of convex cones in $\mathbb{R} \times \mathbb{R}^d$ defined as follows. Let $R \subseteq \mathbb{R} \times \mathbb{R}^d$ be a convex cone. Then, $y \in N_+(R)$ if and only if there exists a symmetric matrix $Y \subseteq \mathbb{R}^{(d+1)\times(d+1)}$ such that

(i) $y = Ye_\emptyset = \operatorname{diag}(Y)$,

(ii) for all $i \in [d]$, $Ye_i, Y(e_\emptyset - e_i) \in R$,

(iii) Y is positive semidefinite,

where e_i denotes the vector with a one in the i-th entry and zero elsewhere. A matrix Y satisfying the conditions above is sometimes called in the literature the *protection matrix* of y. The following property comes directly from the definition of the operator.

Proposition 11. *Let R be a convex cone. Then, $N_+(R)$ is a convex cone and $N_+(R) \subseteq R$.*

Proof. Let $y, z \in N_+(R)$ and $\lambda, \mu \in \mathbb{R}_+$. It is enough to show that $\lambda y + \mu z \in N_+(R)$. Let Y and Z be protection matrices for y and z respectively. Then, $\lambda Y + \mu Z$ is a protection matrix for $\lambda y + \mu z$. Conditions (i) and (ii) come from the linearity of diag and the matrix product. Since the set of symmetric positive semidefinite matrices is a convex cone, the condition (iii) follows as well.

Now we prove the second property. Let $y \in N_+(R)$. In particular, $Ye_1, Y(e_\emptyset - e_1) \in R$. Since R is a convex cone, it follows that $Ye_1 + Y(e_\emptyset - e_1) = Ye_\emptyset = y$ by condition (i), and therefore $y \in R$. □

Given a polytope $P \subseteq [0,1]^d$, let $Q = \{(a, x) \in \mathbb{R}_+ \times P : x/a \in P\} \subseteq \mathbb{R}_+ \times \mathbb{R}^d$ be the *lifted convex cone* of P. The level r relaxation of the LS_+ hierarchy, $N_+^r(Q) \subseteq \mathbb{R} \times \mathbb{R}^d$, is defined recursively as follows: $N_+^0(Q) = Q$ and $N_+^r(Q) = N_+(N_+^{r-1}(Q))$. Proposition 11 guarantees the recursive definition is well defined ant it defines a nested family. Furthermore, the family is guaranteed to converge to the cone spanned by the integer vectors at Q at level d [78, Theorem 1.4].

4.1 Integrality gap of LS_+: Proof of Theorem 1(iii)

To prove the integrality gap for LS_+ we follow an inductive argument. We start from the configuration LP, $\mathrm{clp}(T)$, and we denote by Q_k the lifted convex cone of $\mathrm{clp}(T)$ for instance I_k and $T = 1023$. For simplicity, we sometimes use the notation iC for the ordered pair (i, C), as it is usually done for directed graphs.

Let A be an M-matching in $M \times \mathcal{C}^\beta$, as defined in Section 3.2. The *partial schedule* $y(A) \in \mathbb{R}^{M \times \mathcal{C}}$ is the vector such that for every $i \in M$ and $j \in \{1, 2, \ldots, 6\}$, $y(A)_{iC_j} = \phi(A \cup \{(i, C_j)\})/\phi(A)$, and zero otherwise. Below is the key Lemma that implies the main theorem. We postpone its proof to the end of the next section.

Lemma 12. *Let k be an odd integer and $r \leq \lfloor k/2 \rfloor$. Then, for every M-matching A of cardinality $\lfloor k/2 \rfloor - r$ in $M \times \mathcal{C}^\beta$, we have $y(A) \in N_+^r(Q_k)$.*

Proof of Theorem 1(iii). Let k be such that $n = 15k + \ell$ where k is the greatest odd integer such that $15k \leq n$. Consider instance I_k defined in Section 2.1, $T = 1023$ and $r = \lfloor k/2 \rfloor$. By Lemma 31, for $A = \emptyset$ we obtain $y(\emptyset) \in N_+^r(Q_k)$. □

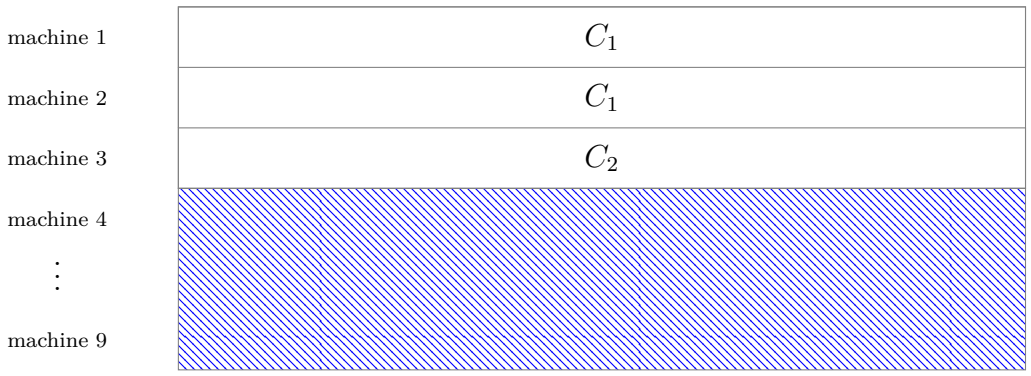


Figure 4.1: Consider the instance I_3, with nine machines. It is shown the partial schedule given by the M-matching $\{(1, C_1), (2, C_1), (3, C_2)\}$; the machines in $\{4, \ldots, 9\}$ are fractional.

In the following we provide two lemmas that describe structural properties of every partial schedule. The first of them justifies the intuition behind their name: If A is an M-matching, the corresponding partial schedule is integral for every machine incident to A. Therefore, all these machines are scheduled integrally and the whole solution can be seen as partial schedule for the instance.

Lemma 13. *Let A be an M-matching in $M \times \mathcal{C}^\beta$. Then, $y(A)$ is integral for every machine incident to A.*

Proof. If $C \notin \mathcal{C}^\beta$ then $y(A)_{iC} = 0$ by definition. If $(i, C_j) \in A$ then $y(A)_{iC_j} = \phi(A \cup \{(i, C_j)\})/\phi(A) = \phi(A)/\phi(A) = 1$. For $\ell \in [6] \setminus \{j\}$, the set $A \cup \{(i, C_\ell)\}$ is not an M-matching and thus $y(A)_{iC_\ell} = 0$. □

The second lemma below provides the base case for the induction in the proof of Lemma 31 and complements the lemma above: The partial schedules are indeed feasible solutions for the configuration LP.

Lemma 14. *Let A be an M-matching in $M \times \mathcal{C}^\beta$ of cardinality at most $\lfloor k/2 \rfloor$. Then, $y(A) \in clp(T)$.*

Proof. We note that $y(A)_{iC} = y^\phi_{A\cup\{(i,C)\}}/y^\phi_A$, and then the feasibility of $y(A)$ in $\mathrm{clp}(T)$ is implied by the feasibility of y^ϕ in $\mathrm{SA}^r(\mathrm{clp}(T))$, for $r = \lfloor k/2 \rfloor$. □

The protection matrices. Given A an M-matching of $M \times \mathcal{C}^\beta$ and the respective partial schedule $y(A)$, let $Y(A)$ be the real symmetric matrix with rows and columns indexed in $\{\emptyset\} \cup (M \times \mathcal{C})$, such that its principal submatrix indexed by $\{\emptyset\} \cup (M \times \mathcal{C}^\beta)$ equals

$$\begin{pmatrix} 1 & y(A)^\top \\ y(A) & Z(A) \end{pmatrix},$$

where $Z(A)_{iC_j,\ell C_h} = \phi(A \cup \{(i,C_j),(\ell,C_h)\})/\phi(A)$. All the other entries of the matrix $Y(A)$ are equal to zero. The matrix $Y(A)$ provides the *protection matrix* we need in the proof of the key Lemma. Condition (i) in the definition of the protection matrix is guaranteed by construction. To check that condition (ii) is satisfied we require the following lemma.

Lemma 15. *Let A be an M-matching in $M \times \mathcal{C}^\beta$ and i a non-incident machine to A. Then, $\sum_{j=1}^6 Y(A)e_{iC_j} = Y(A)e_\emptyset$.*

Finally, condition (iii) is the more technical condition to be checked. It requires the use of algebraic tools and a series of operations to reduce the dimensionality of the problem. We postpone the proof of Theorem 4 to Section 4.2. We then prove Lemma 15 and posteriorly we are ready to prove the key Lemma.

Theorem 4. *For every M-matching A in $M \times \mathcal{C}^\beta$ such that $|A| \le \lfloor k/2 \rfloor$, the matrix $Y(A)$ is positive semidefinite.*

Proof of Lemma 15. Let S be the index of a row of $Y(A)$. If $S \notin \{\emptyset\} \cup (M \times \mathcal{C}^\beta)$ then that row is identically zero, so the equality is satisfied. Otherwise,

$$e_S^\top \sum_{j=1}^6 Y(A)e_{iC_j} = \frac{1}{\phi(A)} \sum_{j=1}^6 \phi(A \cup \{(i,C_j)\} \cup S).$$

If $A \cup S$ is not an M-matching then $\phi(A \cup S \cup \{i, C_j\}) = 0$ for all i and $j \in [6]$, and $e_S^\top Y(A)e_\emptyset = \phi(A \cup S) = 0$, so the equality is satisfied. If $A \cup S$ is an M-matching, then

$$\begin{aligned} \frac{1}{\phi(A)} \sum_{j=1}^6 \phi(A \cup \{(i,C_j)\} \cup S) &= \frac{\phi(A \cup S)}{\phi(A)} \sum_{j=1}^6 \frac{\phi(A \cup S \cup \{(i,C_j)\})}{\phi(A \cup S)} \\ &= e_S^\top Y(A)e_\emptyset \sum_{j=1}^6 \frac{y^\phi_{A\cup S\cup\{(i,C_j)\}}}{y^\phi_{A\cup S}} = e_S^\top Y(A)e_\emptyset, \end{aligned}$$

since y^ϕ is a feasible solution for the SA hierarchy and threfore the summation above is equal to 1 due to the machine constraints (3.1). □

Proof of Lemma 31. We proceed by induction in r. For $r = 0$ this is implied by Lemma 14, so now suppose that it holds for $r = t$. Let $y(A)$ be a partial schedule of A of cardinality $\lfloor k/2 \rfloor - t - 1$. We prove that the matrix $Y(A)$ is a protection matrix for $y(A)$. It is symmetric by definition, and $Y(A)e_\emptyset = (1, \mathrm{diag}(Y(A))) = (1, y(A))$, so condition (i) is satisfied. Thanks to Theorem 4 the matrix $Y(A)$ is positive semidefinite, that is condition (iii).

It remains to check that condition (ii) is fulfilled. Let (i, C) be such that $y(A)_{iC} \in (0, 1)$. In particular, by Lemma 13 we have $(i, C) \notin A$ and $C \in \mathcal{C}^\beta$. We claim that $Y(A)e_{iC}/y(A)_{iC}$ is equal to $(1, y(A \cup \{(i, C)\}))$. If S indexes a row not in $M \times \mathcal{C}^\beta$ then the respective entry in both vectors is zero, so the equality is satisfied. Otherwise,

$$\frac{e_S^\top Y(A)e_{iC}}{y(A)_{iC}} = \frac{\phi(A \cup \{(i, C)\} \cup S)}{\phi(A \cup \{(i, C)\})} = y(A \cup \{(i, C)\})_S.$$

The cardinality of the M-matching $A \cup \{(i, C)\}$ is equal to $|A| + 1 = \lfloor k/2 \rfloor - t$, and therefore by induction we have that $Y(A)e_{iC}/y(A)_{iC} = (1, y(A \cup \{(i, C)\})) \in N_+^t(Q_k)$. Now we prove that the vectors $Y(A)(e_\emptyset - e_{iC})/(1 - y(A)_{iC})$ are feasible for $N_+^t(Q_k)$. By Lemma 15 we have that for every $\ell \in \{1, 2, \ldots, 6\}$,

$$\frac{Y(A)(e_\emptyset - e_{iC_\ell})}{1 - y(A)_{iC_\ell}} = \sum_{j \in [6] \setminus \{\ell\}} \left(\frac{y(A)_{iC_j}}{\sum_{j \in [6] \setminus \{\ell\}} y(A)_{iC_j}} \right) y(A \cup \{(i, C_j)\}),$$

and then $Y(A)(e_\emptyset - e_{iC_\ell})/(1 - y(A)_{iC_\ell})$ is a convex combination of the partial schedules $\{y(A \cup \{(i, C_j)\}) : j \in [6] \setminus \{\ell\}\} \subset N_+^t(Q_k)$, concluding the induction. □

4.2 The protection matrices are PSD

In this section we provide a full proof of Theorem 4. To prove that $Y(A)$ is a positive semidefinite matrix we perform several transformations of the original matrix that preserve the property of being positive semidefinite or not. We start with a short summary of the proof.

Proof scheme. First, we remove all those zero columns and rows. Then, $Y(A)$ is positive semidefinite if and only if its principal submatrix indexed by $\{\emptyset\} \cup (M \times \mathcal{C}^\beta)$ is positive semidefinite. We construct the matrix $\mathrm{Cov}(A)$ by taking the Schur's Complement of $Y(A)$ with respect to the entry $(\emptyset, \emptyset)$.

The resulting matrix is positive semidefinite if and only if $Y(A)$ is positive semidefinite. After removing null rows and columns in $\mathrm{Cov}(A)$ we obtain a new matrix, $\mathrm{Cov}_+(A)$, which can be written using tensor products as $I \otimes Q + (J - I) \otimes W$, with $Q, W \in \mathbb{R}^{6 \times 6}$, $W = \alpha Q$ for some $\alpha \in (-1, 0)$ and I, J being the identity and the all-ones matrix, respectively. By applying a lemma about block matrices in [49], $Y(A)$ is positive semidefinite if and only if W is positive semidefinite.

Finally, the matrix W is equal to $D_u - uu^\top$ for some $u \in \mathbb{R}^6$ and D_u is a diagonal matrix such that $\text{diag}(D_u) = u$. By applying Jensen's inequality it follows that W is positive semidefinite.

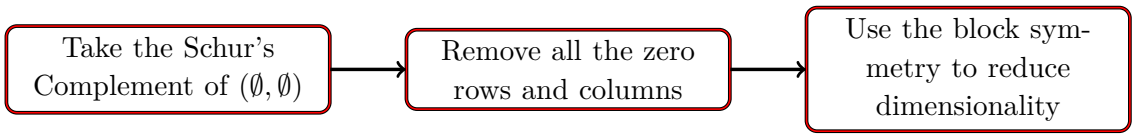


Figure 4.2: Proof scheme: The protection matrices are positive semidefinite.

We now continue with the full argument. The first basic operation we can perform over the matrix is to remove null rows and columns. In the case of our protection matrices, there are many rows and columns that are equal to zero by construction.

Lemma 16. *A symmetric matrix $X \in \mathbb{R}^{E \times E}$ is positive semidefinite if and only if the principal submatrix of non-null columns and rows is positive semidefinite.*

Proof. Let $\tilde{X} \in \mathbb{R}^{F \times F}$ be the matrix obtained by removing the null rows and columns of x. Then, $z^\top X z = \sum_{i,j \in E} X_{ij} z_i z_j = \sum_{i,j \in F} X_{ij} z_i z_j = z_F^\top \tilde{X} z_F$, where $z_F \in \mathbb{R}^F$ is the restriction of Z to the variables in F. It follows that $z^\top X z \geq 0$ if and only if $z_F^\top \tilde{X} z_F \geq 0$. □

Therefore, in order to prove that a matrix is positive semidefinite we can remove in advance those null rows and columns. In the following lemma we describe a reduction based on a congruent transformation of a matrix known as *Schur's complement*. Given the symmetric matrix

$$X = \begin{pmatrix} N & B \\ B^\top & C \end{pmatrix}$$

with N invertible, the *Schur's complement* of N in X is the matrix $S = C - B^\top N^{-1} B$.

Lemma 17. *If N is positive definite, then X is positive semidefinite if and only if S is positive semidefinite.*

A proof of this fact can be found in [54, Theorem 7.7.9 on page 496]. We apply the two previous transformations to $Y(A)$ as described in the following lemma.

Lemma 18. *Let A be an M-matching in $M \times \mathcal{C}^\beta$. Then, the matrix $Y(A)$ is positive semidefinite if and only if $Z(A) - y(A)y(A)^\top$ is positive semidefinite.*

Proof. Thanks to Lemma 16 the matrix $Y(A)$ is positive semidefinite if and only if its principal submatrix indexed by $\{\emptyset\} \cup (M \times \mathcal{C}^\beta)$ is positive semidefinite, since every other row and column are zero. The Schur's complement of the entry $Y(A)_{\emptyset,\emptyset} = 1$ corresponds to $Z(A) - y(A)y(A)^\top$. A direct application of Lemma 17 implies that $Y(A)$ is positive semidefinite if and only if $Z(A) - y(A)y(A)^\top$ is positive semidefinite. □

Further removals. We denote by $\mathrm{Cov}(A)$ the matrix $Z(A) - y(A)y(A)^\top$, the *covariance matrix* of $y(A)$. In the proof of Lemma 18 we removed first all those null rows and columns in $Y(A)$. In fact, in the new matrix $\mathrm{Cov}(A)$ there are null rows and columns that can be removed of the matrix, so we can perform this operation again. Recall that $\pi_M(A)$ is the set of machines incident to A, and let $\mathrm{Cov}_+(A)$ be the principal submatrix of $\mathrm{Cov}(A)$ obtained by removing every row and column indexed by $E(\pi_M(A)) = \{(i, C_j) : i \in \pi_M(A), j \in \{1, 2, \ldots, 6\}\}$.

Lemma 19. *Let A be an M-matching in $M \times \mathcal{C}^\beta$. Then, $Y(A)$ is positive semidefinite if and only if $Cov_+(A)$ is positive semidefinite.*

Proof. We prove that for every $(i, C_j) \in E(\pi_M(A))$, $\mathrm{Cov}(A)e_{iC_j} = 0$. By expanding,

$$e_{\ell C_k}^\top \mathrm{Cov}(A) e_{iC_j} = \frac{1}{\phi(A)} \phi(A \cup \{(\ell, C_k), (i, C_j)\}) - y(A)_{iC_j} y(A)_{\ell C_k}.$$

By Lemma 13, if i is a machine incident to A we have $y(A)_{iC_j} = 1_{\{(i,C_j) \in A\}}$, and by the definition of the function ϕ we have that it is supported over M-matchings only. Then, $\phi(A \cup \{(\ell, C_k), (i, C_j)\}) = \phi(A \cup (\ell, C_k)\})1_{\{(i,C_j)\in A\}}$, and

$$e_{\ell C_k}^\top \mathrm{Cov}(A) e_{iC_j} = \left(\frac{1}{\phi(A)} \phi(A \cup \{(\ell, C_k)\}) - y(A)_{\ell C_k} \right) 1_{\{(i,C_j)\in A\}} = 0,$$

because of the definition of the partial schedule $y(A)$. The Lemma follows by a direct application of Lemma 16. □

Note that if machine i is not incident to A, the values $\{y(A)_{iC_j} : j \in [6]\}$ do not depend on the machine, but only on A. This motivates the definition of the vector $\nu(A) \in \mathbb{R}^6$ such that

$$\nu(A)_j = \frac{k/2 - \deg_A(C_j)}{3k - |A|},$$

for every $j \in \{1, 2, \ldots, 6\}$. If $y(A)$ is a partial schedule, for every machine incident to A the entries y_{iC_j} are in $\{0, 1\}$, which is in total $3|A|$ entries. This is why $\mathrm{Cov}_+(A)$ has a very particular structure: If we order the rows and columns according to machines, a block structure appears. To properly define this we require some notation. Let $U \in \mathbb{R}^{p\times q}$ and $T \in \mathbb{R}^{r\times s}$. Then, the tensor product between U and T, $U \otimes T$, is the matrix

$$\begin{pmatrix} U_{11}T & \ldots & U_{1q}T \\ \vdots & \ddots & \vdots \\ U_{p1}T & \ldots & U_{pq}T \end{pmatrix} \in \mathbb{R}^{pr\times qs}.$$

Proposition 20. *Let A, B, C, D four matrices such that the products AC and BD are well defined. Then, the following holds:*

(i) $(A \otimes B)(C \otimes D) = AC \otimes BD$.

(ii) $(A \otimes B)^\top = A^\top \otimes B^\top$.

A proof of (i) can be found at [53, Lemma 4.2.10 on page 244] and (ii) is direct from the definition. The fractional part of $y(A)$ corresponds to $\mathbf{1} \otimes \nu(A)$, where $\mathbf{1}$ is the vector in $\mathbb{R}^{3k-|A|}$ with every entry equal to 1. Furthermore, the value $Z(A)_{iC_j,\ell C_\ell} = \gamma(A)_{j\ell}$ is independent of the machines when i, ℓ are not incident to A,

$$\gamma(A)_{j\ell} = \begin{cases} \dfrac{(k/2 - \deg_A(j))(k/2 - \deg_A(\ell))}{(3k - |A|)(3k - |A| - 1)} & \text{if } j \neq \ell, \\ \dfrac{(k/2 - \deg_A(j))(k/2 - \deg_A(j) - 1)}{(3k - |A|)(3k - |A| - 1)} & \text{if } j = \ell. \end{cases}$$

Then, we can fully express the matrix $\mathrm{Cov}_+(A)$ in terms of tensor products involving $\gamma(A)$ and $\nu(A)$

$$I \otimes D_\nu(A) + (J - I) \otimes \gamma(A) - (\mathbf{1} \otimes \nu(A))(\mathbf{1} \otimes \nu(A))^\top, \tag{4.1}$$

where I is the $3k - |A|$ dimensional identity matrix, J is the $3k - |A|$ dimensional all-ones matrix and $D_\nu(A) \in \mathbb{R}^{6\times 6}$ is such that $D_\nu(A)_{jj} = \nu(A)_j$ for $j \in \{1, \ldots, 6\}$, and $D_\nu(A)_{jh} = 0$ when $j \neq h$. The key consequence is that we can reduce the task of proving that $Y(A)$ is positive semidefinite to prove the same for matrices of constant dimension.

Block matrices. We say that a matrix $X \in \mathbb{R}^{rn\times rn}$ belongs to the (r, n)-block symmetry set, Sym_r^n, if there exist matrices $D, E \in \mathbb{R}^{r\times r}$ such that $X = I \otimes D + (J - I) \otimes E$. For example, when $n = 3$,

$$X = \begin{pmatrix} D & E & E \\ E & D & E \\ E & E & D \end{pmatrix}.$$

The following lemma characterizes the set of positive semidefinite matrices in Sym_r^n.

Lemma 21 ([49])**.** *Let $X = I \otimes D + (J - I) \otimes E \in Sym_r^n$. Then, X is positive semidefinite if and only if i) $D - E$ is positive semidefinite and ii) $D + (n - 1)E$ is positive semidefinite.*

As a consequence, we reduce the dimensionality of the problem to r. In particular, $\mathrm{Cov}_+(A)$ belongs to $\mathrm{Sym}_6^{3k-|A|}$ and therefore the task now is to check that two matrices of constant dimension are positive semidefinite.

Lemma 22. *For every M-matching A in $M \times \mathcal{C}^\beta$, the covariance matrix $Cov_+(A)$ belongs to $Sym_6^{3k-|A|}$ and it is equal to*

$$I \otimes \Delta(A) - (J - I) \otimes \frac{1}{3k - |A| - 1}\Delta(A),$$

where $\Delta(A) = D_\nu(A) - \nu(A)\nu(A)^\top$.

Proof. We check first that $\mathrm{Cov}(A)_+ \in \mathrm{Sym}_6^{3k-|A|}$. By Proposition 20, we have

$$(\mathbf{1}\otimes\nu(A))(\mathbf{1}\otimes\nu(A))^\top = (\mathbf{1}\otimes\nu(A))\left(\mathbf{1}^\top\otimes\nu(A)^\top\right) = J\otimes\nu(A)\nu(A)^\top.$$

Replacing this in the expression in 4.1 we get

$$\begin{aligned}\mathrm{Cov}_+(A) &= I\otimes D_\nu(A) + (J-I)\otimes\gamma(A) - J\otimes\nu(A)\nu(A)^\top\\ &= I\otimes\left(D_\nu(A)-\nu(A)\nu(A)^\top\right) + (J-I)\otimes\left(\gamma(A)-\nu(A)\nu(A)^\top\right)\\ &= I\otimes\Delta(A) + (J-I)\otimes\left(\gamma(A)-\nu(A)\nu(A)^\top\right),\end{aligned}$$

so it remains to check that $\gamma(A)-\nu(A)\nu(A)^\top = -1/(3k-|A|-1)\cdot\Delta(A)$. Consider two configurations $C_j \neq C_\ell$. Then, the non-diagonal entry (j,ℓ) is equal to

$$\begin{aligned}\gamma(A)_{j\ell} - \nu(A)_j\nu(A)_\ell &= \frac{(k/2-\deg_A(j))(k/2-\deg_A(\ell))}{(3k-|A|)(3k-|A|-1)} - \nu(A)_j\nu(A)_\ell\\ &= \left(\frac{3k-|A|}{3k-|A|-1}-1\right)\nu(A)_j\nu(A)_\ell = -\frac{1}{3k-|A|-1}\Delta(A)_{j\ell}.\end{aligned}$$

For a diagonal element, we have

$$\begin{aligned}\gamma(A)_{jj} - \nu(A)_j^2 &= \frac{(k/2-\deg_A(j))(k/2-\deg_A(j)-1)}{(3k-|A|)(3k-|A|-1)} - \nu(A)_j^2\\ &= \frac{3k-|A|}{3k-|A|-1}\left(\nu(A)_j^2 - \frac{1}{3k-|A|}\nu(A)_j\right) - \nu(A)_j^2\\ &= -\frac{1}{3k-|A|-1}(\nu(A)_j - \nu(A)_j^2) = -\frac{1}{3k-|A|-1}\Delta(A)_{jj}.\end{aligned}$$

□

In addition with Lemma 21 we have the ingredients to prove that $\mathrm{Cov}_+(A)$ is a positive semidefinite matrix when $|A|\leq\lfloor k/2\rfloor$.

Proof of Theorem 4. By Lemma 22, $\mathrm{Cov}_+(A)\in\mathrm{Sym}_6^{3k-|A|}$ and then we can apply Lemma 21. Therefore, the matrix $\mathrm{Cov}_+(A)$ is positive semidefinite if and only if i) $\Delta(A) - \frac{1}{3k-|A|-1}\Delta(A) = \frac{3k-|A|}{3k-|A|-1}\Delta(A)$ is positive semidefinite, and ii) $\Delta(A) - (3k-|A|-1)\cdot\frac{1}{3k-|A|-1}\Delta(A) = 0$ is positive semidefinite. The last condition is trivially satisfied, and then $\mathrm{Cov}_+(A)$ is positive semidefinite if and only if $\Delta(A)$ is positive semidefinite. Given any vector $x\in\mathbb{R}^6$, we have

$$x^\top\Delta(A)x = \sum_{j=1}^6\nu(A)_jx_j^2 - \left(\sum_{j=1}^6\nu(A)_jx_j\right)^2.$$

The vector $\nu(A)\in\mathbb{R}^6$ is non-negative if $|A|\leq\lfloor k/2\rfloor$. As it satisfies that $\|\nu(A)\|_1 = 1$, by applying Jensen's inequality with the function $\phi(w)=w^2$ the latter expression is non-negative for every $x\in\mathbb{R}^6$. Then, $\mathrm{Cov}_+(A)$ is positive semidefinite, and thanks to Lemma 19 we conclude that $Y(A)$ is positive semidefinite. □

Chapter 5

Break symmetries to approximate

A natural source of symmetry in our problem comes from the fact that the machines are identical: Given a schedule, we obtain other schedules with the same makespan by permuting the assignment over the machines. A second source of symmetry in the problem arises from permuting jobs with the same processing time.

The symmetric nature of the combinatorial problem does not directly imply that a formulation is symmetric, but most of the time the *natural* formulations reflect the symmetries as well. This is the case for the two formulations considered so far in this work, the assignment LP and the configuration LP. In what follows we introduce the necessary algebraic tools for this chapter and formalize all these notions.

5.1 Group invariant sets

Given a set of variables D, we denote by $\mathrm{GL}(\mathbb{R}^D)$ the set of invertible matrices in $\mathbb{R}^{D\times D}$. For a group G, we say that $\Phi : G \to \mathrm{GL}(\mathbb{R}^D)$ is a *matrix representation* over $\mathbb{R}^D$ if Φ is a *G-homomorphism*, that is, for every $g, h \in G$ we have $\Phi(g)\Phi(h) = \Phi(gh)$. A polytope $K \subseteq \mathbb{R}^D$ is said to be *G-invariant* if for every $g \in G$ and $x \in K$, $\Phi(g)x \in K$. We say that G induces an *action* over K, by the function $\rho : G \times K \to K$ such that $\rho(g, x) = \Phi(g)x$.

A group of particular interest is the *symmetric group* of order d, denoted by S_d. It corresponds to the set of permutations over $\{1, \ldots, d\}$ and the composition as product. This group has a matrix representation P over $\mathbb{R}^d$ given by the *permutation matrices*: For every $\ell, k \in \{1, \ldots, d\}$, let $P(\sigma)_{\ell k} = 1$ if and only if $\sigma(\ell) = k$.

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 5.1: Permutation matrix of $\sigma = (12)(34)(5)$.

It is not hard to check that for every scheduling instance and every $T > 0$, the assignment polytope is S_m-invariant, where the set of machines is $M = \{1, \dots, m\}$. In this case, the variables are indexed by the set $M \times J$ and the representation of S_m over $\mathbb{R}^{M \times J}$ is constructed as follows: Given $\sigma \in S_m$, define $\Phi(\sigma) = P(\sigma) \otimes I$, where I is the identity matrix in $\mathbb{R}^{J \times J}$. The fact that Φ is a matrix representation follows using the properties in Proposition 20: For every $\sigma, \pi \in S_m$,

$$\Phi(\sigma)\Phi(\pi) = (P(\sigma) \otimes I)(P(\pi) \otimes I) = P(\sigma)P(\pi) \otimes I^2 = P(\sigma\pi) \otimes I = \Phi(\sigma\pi).$$

Let $\sigma \in S_m$ and $x \in \text{assign}(T)$. For every $j \in J$, we have

$$\sum_{i \in M} (\Phi(\sigma)x)_{ij} = \sum_{i \in M} x_{\sigma(i)j} = \sum_{i \in M} x_{ij} = 1,$$

since σ is a permutation of M. Therefore, every job constraint remains the same when permuting machines. Now given $i \in \{1, \dots, m\}$, we have

$$\sum_{j \in J} (\Phi(\sigma)x)_{ij} p_j = \sum_{j \in J} x_{\sigma(i)j} p_j \leq T,$$

since the last corresponds to the machine constraint for $\sigma(i)$. Observe that having every right hand side in the machine constraint equal to T is relevant to check the invariance. If we had different machine availabilities then is not clear that the program remains invariant using the same representation. Then we should allow the group to act also over the coefficients in the linear functions defining the polytope.

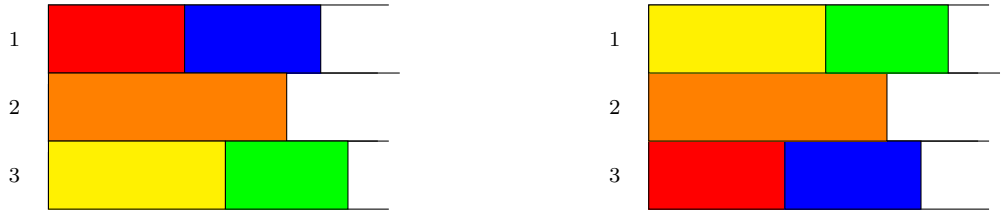


Figure 5.2: Schedule of an instance with three machines and five jobs. Effect of the permutation $\sigma = (13)$ in the solution.

We can also observe that S_m is acting over the configuration LP. The representation of the permutation group is constructed in a very similar way, as well as checking that $\text{clp}(T)$ is S_m-invariant. Scheduling identical machines belongs to a larger class of *partitioning problems*. The common feature of all these problems is to have a set of elements that are to be partitioned in subsets, each of them meeting some constraints. In this case, we look for a partition of the jobs, where every part satisfies a *packing* constraint.

5.2 Symmetry breaking inequalities

Recall that in the configuration LP we look for an assignment of machines to the set of configurations $\mathcal{C}$. For example, in the hard instances obtained from the Petersen graph

in Chapter 2 (see Figure 2.1) the number of configurations for $T = 1023$ is equal to six, so every feasible schedule should have many machines scheduled according to the same configuration. In what follows, we introduce a set of constraints that guarantee *every* integer solution in the polytope to obey a specific order on the configurations over the machines. Recall that the set of machine is the set $M = \{1, \dots, m\}$. Therefore, if we knew that a machine ℓ is scheduled according to a given configuration C, it should be that every machine smaller (larger) than ℓ is scheduled respecting the order on the configurations set. This is a way of *breaking* the machine symmetries since we are restricting the set of possible assignments.

We need some notation before introducing the symmetry breaking constraints. Suppose we have a partitioning $\mathcal{J}$ of the job set J into s parts, $\mathcal{J} = \{J_1, \dots, J_s\}$. In principle, the partition is arbitrary. An example of such a partition is given in the following way. Suppose the job sizes are ordered from largest to smallest, that is $p^1 > p^2 > \cdots > p^s$ where $s = |\{p_j : j \in J\}|$ is the number of different job sizes. Then, we could define the partition such that $J_q = \{j \in J : p_j = p^q\}$ for every $q \in [s]$, and we call it the *job-sizes partition*.

In this context, a *configuration* C is a multiset of elements in $\{1, \dots, s\}$. Recall that for every $q \in \{1, \dots, s\}$, the multiplicity of q in C, $m(q, C)$, is the number of times that p^q appears repeated in C. For example, if $s = 3$ and $C = \{1, 1, 1, 2\}$, we have $m(1, C) = 3$, $m(2, C) = 1$ and $m(3, C) = 0$. We denote by $\mathcal{C}$ the set of all configurations. Observe that it coincides with the configuration notion introduced in Chapter 1 if we consider the job-sizes partition.

We say that a configuration C is *lexicographically* larger than S, and we denote $C >_{\text{lex}} S$, if there exists $q \in [s]$ such that $m(\ell, C) = m(\ell, S)$ for all $\ell < q$ and $m(q, C) > m(q, S)$. Following the example with $s = 3$, the configuration $C = \{1, 1, 1, 2, 2\}$ is lexicographically larger than $S = \{1, 1, 1, 2, 3\}$ since $m(1, C) = m(1, S) = 3$ and $m(2, C) = 2 > m(2, S) = 1$.

Given a positive integer number B, consider the polytope $\text{assign}(B, T)$ given by

$$\text{assign}(T) \cap \bigcap_{i=1}^{m-1} \left\{ x \in \mathbb{R}^{M \times J} : \sum_{q=1}^{s} B^{s-q} \sum_{j \in J_q} \left(x_{ij} - x_{(i+1)j} \right) \geq 0 \right\}.$$

Sometimes, to avoid confusion we use the notation $\text{assign}(J, B, T)$ to emphasize that we are considering the polytope for the jobs set J. We also remark that the symmetry breaking constraints depend on the partitioning of J. Given a subset of jobs $K \subset J$ such that $\sum_{j \in K} p_j \leq T$, we denote by $\text{conf}(K)$ the configuration such that for every $q \in \{1, \dots, s\}$, $m(q, \text{conf}(K)) = |K \cap J_q|$. We then say that $\text{conf}(K)$ is the *configuration induced by* K and we denote by $\mathcal{C}$ the set of all possible configurations induced in this way. Observe that $>_{\text{lex}}$ defines a total order over $\mathcal{C}$.

In the following we show that for sufficiently large, but polynomially sized B, every integer solution in the polytope obeys the lexicographic order on configurations over the machines. More specifically, given a feasible integer solution $x \in \text{assign}(T)$, for every machine $i \in M$ let $\mathcal{C}_i(x) \in \mathcal{C}$ be the configuration defined by the number of jobs for each possible part that are scheduled in i according to x, that is, for every $q \in \{1, \dots, s\}$, $m(q, \mathcal{C}_i(x)) = \sum_{j \in J_q} x_{ij}$.

Theorem 5. *There exists $B = O(|J|^2)$ such that for every integer solution $x \in assign(B,T)$, $\mathcal{C}_i(x) \geq_{lex} \mathcal{C}_{i+1}(x)$ for every machine $i \in M \setminus \{m\}$.*

In general, the polytope above is *not* S_m-invariant, but it is a valid formulation for the problem of finding a schedule with makespan at most T. More specifically, we show that if there is a schedule with makespan at most T, then there is an integer solution in assign(B,T).

Lemma 23. *Suppose there is an integer solution in assign(T). Then, there exists $B = O(|J|^2)$ for which there is an integer solution in assign(B,T).*

We prove the theorem statements above by introducing an intermediate result connecting the lexicographic order over the configurations and the symmetry breaking constraints shown above. Given $B \in \mathbb{N}$, let $\mathcal{L}_B : \mathcal{C} \to \mathbb{R}$ be the function such that for every configuration $C \in \mathcal{C}$,

$$\mathcal{L}_B(C) = \sum_{q=1}^{s} B^{s-q} m(q, C).$$

Recall that the lexicographic order over the configurations induces a total order over $\mathcal{C}$. We show that exists a polynomially sized value of B for which $\mathcal{L}_B$ is a *strictly increasing function*, that is, if $C <_{\text{lex}} S$ then $\mathcal{L}_B(C) < \mathcal{L}_B(S)$. Using this result we then prove Theorem 5.

Lemma 24. *Let $B > 2s \max_{q\in[s]} |J_q|$. Then, $\mathcal{L}_B$ is strictly increasing.*

Proof of Theorem 5. Fix a machine $i \in M \setminus \{m\}$. Since x is an integer solution in assign(B,T), we have $\mathcal{C}_i(x), \mathcal{C}_{i+1}(x) \in \mathcal{C}$. The symmetry breaking constraints implies that

$$0 \leq \sum_{q=1}^{s} B^{s-q} \left(m(q, \mathcal{C}_i(x)) - m(q, \mathcal{C}_{i+1}(x)) \right) = \mathcal{L}_B(\mathcal{C}_i(x)) - \mathcal{L}_B(\mathcal{C}_{i+1}(x)).$$

Applying Lemma 24 for $B = 1 + 2s \max_{q\in[s]} |J_q| = O(|J|^2)$ it holds that $\mathcal{L}_B$ is strictly increasing and therefore $\mathcal{C}_i(x) \geq_{\text{lex}} \mathcal{C}_{i+1}(x)$. □

Proof of Lemma 23. Consider an integer solution $x \in$ assign(T). Since the lexicographic relation defines a total order over $\mathcal{C}$, there exists a permutation $\sigma \in S_m$ such that $\mathcal{C}_{\sigma(i)}(x) \geq_{\text{lex}} \mathcal{C}_{\sigma(i+1)}(x)$ for every $i \in M \setminus \{m\}$. Consider the new schedule $\tilde{x}$ obtained by permuting the solution according to σ, that is, $\tilde{x}_{ij} = x_{\sigma(i)j}$ for every $(i,j) \in M \times J$. Then, for every $i \in M \setminus \{m\}$ it follows that

$$\begin{aligned}\sum_{q=1}^{s} B^{s-q} \sum_{j\in J_q} \left(\tilde{x}_{ij} - \tilde{x}_{(i+1)j} \right) &= \sum_{q=1}^{s} B^{s-q} \left(m(q, \mathcal{C}_{\sigma(i)}(x)) - m(q, \mathcal{C}_{\sigma(i+1)}(x)) \right) \\ &= \mathcal{L}_B(\mathcal{C}_{\sigma(i)}(x)) - \mathcal{L}_B(\mathcal{C}_{\sigma(i+1)}(x)) \geq 0,\end{aligned}$$

where the last inequality holds by Lemma 24. We conclude that $\tilde{x} \in$ assign(B,T). □

We recall that having B of polynomial input size is relevant at the moment of solving the linear program assign(B,T). In particular, the input size is $O(|J|^2 \cdot \log(B^s)) = O(\text{poly}(|J|))$, and so in can be solved in time $O(\text{poly}(|J|))$ using, for example, the Karmakar algorithm or the Ellipsoid method [94, Chapter 15]. In what follows, we use a superscript over the configurations to indicate its lexicographic order, that is, $C^1 > C^2 > C^3 > \cdots > C^{|\mathcal{C}|}$.

Proof of Lemma 24. Consider two configurations $C, S \in \mathcal{C}$ such that $C >_{\text{lex}} S$. Let $\tilde{q}$ be the smallest in $\{1,\ldots,s\}$ integer such that the multiplicities of the configurations are different, that is, $m(\ell, C) = m(\ell, S)$ for every $\ell < \tilde{q}$. For the sake of contradiction suppose that $m(\tilde{q}, C) < m(\tilde{q}, S)$. In particular, every term up to $\max\{0, \tilde{q}-1\}$ in the summation defining $\mathcal{L}_B(C) - \mathcal{L}_B(S)$ is equal to zero. By upper bounding the summation from $\min\{s, \tilde{q}+1\}$ we obtain that

$$\begin{aligned}\sum_{q=\min\{s,\tilde{q}+1\}}^{s} B^{s-q}\left(m(q,C) - m(q,S)\right) &\leq \sum_{q=\min\{s,\tilde{q}+1\}}^{s} B^{s-q}\left(|m(q,C)| + |m(q,S)|\right)\\ &\leq \sum_{q=\min\{s,\tilde{q}+1\}}^{s} B^{s-q} \cdot 2|J_q|\\ &< \left(2s \max_{q\in[s]} |J_q|\right) B^{s-\tilde{q}-1} \leq B \cdot B^{s-\tilde{q}-1},\end{aligned}$$

and since $m(\tilde{q}, S) - m(\tilde{q}, C) \geq 1$ it follows that

$$\begin{aligned}\sum_{q=\tilde{q}}^{s} B^{s-q}\left(m(q,C) - m(q,S)\right) &< B^{s-\tilde{q}} + B^{s-\tilde{q}}\left(m(\tilde{q},C) - m(\tilde{q},S)\right)\\ &< B^{s-\tilde{q}}\left(1 + m(\tilde{q},C) - m(\tilde{q},S)\right) < 0,\end{aligned}$$

yielding to the contradiction. □

5.3 The Lasserre/SoS (Las) Hierarchy

The Lasserre/SoS hierarchy is the strongest *lift & project* operator among the ones considered in this work, that is Sherali-Adams (SA) and Lóvasz-Schrijver (LS$_+$). Before introducing the relaxations obtained from the hierarchy, we require some notation about *moment matrices*, that play a role in what follows. Then we introduce the hierarchy in a self-contained way. Given a set E be a set and $r \in \mathbb{N}$, recall that $\mathcal{P}_r(E) = \{A \subseteq E : |A| \leq r\}$. For a vector $z \in \mathbb{R}^{\mathcal{P}_{2r}(E)}$, the *$r$-moment matrix*, $M_r(z) \in \mathbb{R}^{\mathcal{P}_r(E)\times\mathcal{P}_r(E)}$, is such that for every $U, V \in \mathcal{P}_r(E)$, $M_r(z)_{U,V} = z_{U\cup V}$. Observe that this matrix is symmetric. We consider an operation $\star$ called the *shift product* between a vector $z \in \mathbb{R}^{\mathcal{P}_r(E)}$ and a vector $a \in \mathbb{R}^{\mathcal{P}_1(E)}$ such that for all $U \in \mathcal{P}_r(E)$,

$$(a \star z)_U = \sum_{e\in E} a_e x_{U\cup\{e\}} - a_\emptyset x_U,$$

recalling that $a_e = a_{\{e\}}$ for simplicity when e is a singleton. Given a scheduling instance and $T > 0$, for every $i \in M$ let $\text{pack}^i \in \mathbb{R}^{\mathcal{P}_1(M\times J)}$ be such that $\text{pack}^i_\emptyset = T$, $\text{pack}^i_{\ell j} = -p_j$ if $\ell = i$ and zero otherwise. Similarly, for every $i \in M \setminus \{m\}$ let $\text{break}^i \in \mathbb{R}^{\mathcal{P}_1(M\times J)}$ be such that

$$\text{break}^i_{\ell j} = \begin{cases} B^{s-q} & \text{if } \ell = i \text{ and } j \in J_q, \text{ for all } q \in \{1,\dots,s\}, \\ -B^{s-q} & \text{if } \ell = i+1 \text{ and } j \in J_q, \text{ for all } q \in \{1,\dots,s\}, \\ 0 & \text{otherwise.} \end{cases}$$

The level r Lasserre/SoS relaxation, $\text{Las}^r(\text{assign}(B,T))$, is an SDP in $[0,1]^{\mathcal{P}_{2r+2}(M\times J)}$,

$$\sum_{i\in M} x_{I\cup\{(i,j)\}} - x_I = 0 \text{ for all } j \in J, \text{ for all } I \in \mathcal{P}_{2r+1}(M\times J), \tag{5.1}$$

$$M_r(\text{pack}^i \star x) \succeq 0 \text{ for all } i \in M, \tag{5.2}$$

$$M_r(\text{break}^i \star x) \succeq 0 \text{ for all } i \in M \setminus \{m\}, \tag{5.3}$$

$$M_{r+1}(x) \succeq 0, \tag{5.4}$$

$$x_\emptyset = 1. \tag{5.5}$$

5.4 Balanced partitionings

Observe that $\text{Las}^0(\text{assign}(B,T))$ is at least as strong as $\text{assign}(B,T)$. In this section we study the integrality gap of the SDP relaxations obtained from the Lasserre/SoS hierarchy. More specifically, we show that after a *small* number of rounds that depends on the configurations set $\mathcal{C}$, and therefore on the partitioning $\mathcal{J}$, the integrality gap can be arbitrarily close to one. Recall that whereas the total number of jobs can be large, the number of configurations could be small and therefore we obtain better bounds. We say a partitioning $\mathcal{J}$ is α-*balanced*, with $\alpha \geq 1$, if for every $K, H \subseteq J$ such that $\text{conf}(K) = \text{conf}(H)$,

$$\sum_{j\in K} p_j \leq \alpha \sum_{j\in H} p_j.$$

Other parameter that plays a key role is the maximum number of jobs that can be scheduled in the same machine with makespan at most T, that is,

$$\lambda = \max\left\{ |K| : \sum_{j\in K} p_j \leq T \right\}.$$

For example, if we knew that $p_j \geq T/3$ for every $j \in J$, then $\lambda \leq 3$. The following is the main result of this chapter: Given a balanced partitioning of J and its induced configurations $\mathcal{C}$, there is a level of the hierarchy for which its feasibility implies the existence of an integral solution with a makespan close to T. Provided that B is chosen accordingly to Lemma 5, it gives us a schedule respecting the lexicographic order of configurations over the machines.

Theorem 6. *Consider a scheduling instance, $T > 0$ and an α-balanced partitioning of the set J. Suppose the $\lambda|\mathcal{C}| = \tau(\mathcal{C})$ level of the Lasserre/SoS hierarchy over assign(B, T) is feasible, that is,*

$$Las^{\tau(\mathcal{C})}(assign(B,T)) \neq \emptyset.$$

Then, there exists a schedule, $x^{lex} \in assign(B, \alpha T) \cap \{0,1\}^{M \times J}$, that is, with makespan at most αT. Furthermore, it can be found in time $|J|^{O(\tau(\mathcal{C}))}$.

The value $\tau(\mathcal{C}) = \lambda|\mathcal{C}|$ can be improved, but this is good enough for this exposition. If we go back to the hard instances shown in Chapter 2, we observe that for $T = 1023$ there is a constant number of configurations for all the instances $\{I_k\}_{k\in\mathbb{N}}$ if we consider the job-sizes partition, since there are only 15 different job sizes. Furthermore, since the smallest job size in the instances is 33, we have that $\lambda \leq 1023/33 = 31$. Recall that for $T = 1023$ there is no feasible schedule for the instance and the job-sizes partition is 1-balanced. Therefore, for every $k \in \mathbb{N}$, there is an $O(1)$ level Lasserre relaxation for $T = 1023$ which is equal to the empty set.

An important observation is that we assume the number of machines m to be greater than $\tau(\mathcal{C})$, since otherwise the theorem follows by a direct application of Theorem 3 in Chapter 3. The proof of this theorem follows in a slightly similar way to that of the Machine Decomposition Lemma (see Chapter 3, Lemma 6). There, the argument was based on an induction over the number of machines that are to be integral. We leave the full proof of Theorem 6 to Section 5.6, and in the following section we show how to obtain a polynomial time approximation scheme as corollary of the main theorem.

5.5 An approximation scheme for Scheduling

An idea that has been frequently exploited for designing approximation schemes in scheduling and packing problems is to split the instances into *long* jobs and *short* jobs. Then, each sub-instance is solved or approximated by using suitable techniques, and merged afterwards in order to provide a solution for the original problem. Rounding the numeric values of the instances plays a key role in this approaches since it allows to reduce the underlying combinatorics. In particular, it allows exhaustive enumeration in some cases.

We provide an approximation scheme that uses the idea of splitting the instance into long and short jobs, but we *do not* round the numeric values. In what follows, given $\varepsilon > 0$, we say that a job $j \in J$ is long if $p_j \geq \varepsilon T$, and it is short otherwise. The subset of long jobs is denoted by J_{long} and the short jobs are $J_{\text{short}} = J \setminus J_{\text{long}}$. We consider a partitioning that *simulates* the rounding by grouping jobs with a similar processing time. More specifically, for every $q \in \{1, \ldots, (1-\varepsilon)/\varepsilon^2\}$, let

$$J_q = \left\{ j \in J_{\text{long}} : \left(\frac{1}{\varepsilon} + q\right)\varepsilon^2 T > p_j \geq \left(\frac{1}{\varepsilon} + q - 1\right)\varepsilon^2 T \right\},$$

and we call this the *ε-partitioning* of the long jobs. We show next that this partitioning is arbitrarily close to being 1-balanced, and therefore we can try to schedule long jobs using the main theorem.

Lemma 25. *For every $\varepsilon > 0$, the ε-partitioning is $(1+\varepsilon)$-balanced.*

Proof. Consider $K, H \subseteq J$ such that $\mathrm{conf}(K) = \mathrm{conf}(H) = C$ for the ε-partitioning $\mathcal{J}$. In particular, for every $q \in \{1, \ldots, |\mathcal{J}|\}$ we have that $|K \cap J_q| = m(q, C) = |H \cap J_q|$, and so there exists a bijection $\varphi_q : K \cap J_q \to H \cap J_q$. Furthermore, for any pair of jobs $j, \ell \in J_q$ it holds

$$\frac{p_j}{p_\ell} \leq \frac{\left(\frac{1}{\varepsilon} + q\right)\varepsilon^2 T}{\left(\frac{1}{\varepsilon} + q - 1\right)\varepsilon^2 T} \leq \frac{\frac{1}{\varepsilon} + 1}{\frac{1}{\varepsilon}} = 1 + \varepsilon,$$

since the function $(\frac{1}{\varepsilon} + q)/(\frac{1}{\varepsilon} + q - 1)$ is strictly decreasing in $[1, +\infty)$. Therefore, for every $q \in \{1, \ldots, |\mathcal{J}|\}$ it holds that $\sum_{j \in K \cap J_q} p_j \leq (1+\varepsilon) \sum_{j \in K \cap J_q} p_{\varphi_q(j)} = (1+\varepsilon) \sum_{j \in H \cap J_q} p_j$, and we conclude that

$$\sum_{j \in K} p_j = \sum_{q=1}^{|\mathcal{J}|} \sum_{j \in K \cap J_q} p_j \leq (1+\varepsilon) \sum_{q=1}^{|\mathcal{J}|} \sum_{j \in H \cap J_q} p_j = (1+\varepsilon) \sum_{j \in H} p_j. \qquad \square$$

Given $\varepsilon > 0$, consider $\mathrm{assign}(B, T)$ obtained from the partitioning above and B chosen according to Lemma 24. Then, using a binary search like procedure we look for the smallest T such that the level $\tau(\mathcal{C})$ relaxation of the Lasserre/SoS hierarchy is feasible for the long jobs. Invoking Theorem 6 we then obtain a schedule for the long jobs. The short jobs are scheduled then using the classic list scheduling algorithm [102].

Algorithm 1

Input: A scheduling instance and $T \geq 1/m \sum_{j \in J} p_j$.
Output: A schedule with makespan at most $(1+\varepsilon)T$ if there exists a schedule for J_{long} with makespan at most T; **infeasible** otherwise.

For all $(i, j) \in M \times J$, initialize $x_{ij} \leftarrow 0$.
Consider the $(\varepsilon/2)$-partitioning of J_{long} and $B = 1 + 4\frac{2-\varepsilon}{\varepsilon^2}|J_{\mathrm{long}}|$.
if $\mathrm{Las}^{\tau(\mathcal{C})}(\mathrm{assign}(J_{\mathrm{long}}, B, T)) \neq \emptyset$ **then**
    construct the Lex-schedule x^{lex} of J_{long}; for all $(i, j) \in M \times J_{\mathrm{long}}$, $x_{ij} \leftarrow x^{\mathrm{lex}}_{ij}$.
    while $J_{\mathrm{short}} \neq \emptyset$ **do**
        Pick $k \in J_{\mathrm{short}}$, and let $i \in M$ such that $i \in \mathrm{argmin}\{\ell \in M : \sum_{j \in J} p_j x_{\ell j}\}$,
        update $x_{ik} \leftarrow 1$ and $J_{\mathrm{short}} \leftarrow J_{\mathrm{short}} \setminus \{k\}$.
    Return x.
else return **infeasible**.

Theorem 7. *If $Las^{\tau(\mathcal{C})}(assign(J_{long}, B, T)) \neq \emptyset$, then Algorithm 2 returns a schedule with makespan at most $(1+\varepsilon)T$.*

Proof. Thanks to Theorem 6, if $\text{Las}^{\tau(\mathcal{C})}(\text{assign}(J_{\text{long}}, B, T)) \neq \emptyset$, then x^{lex} is an integral schedule for J_{long} with makespan at most T. In this case, jobs in J_{short} are assigned according to the list scheduling algorithm, for which we include the analysis only for completeness. Let $k \in J_{\text{short}}$ and let $i \in M$ such that $x_{ik} = 1$. Since $T > 1/m \sum_{j \in J \setminus \{k\}} p_j$, it follows that $T > \sum_{j \in J} p_j x_{ij}$ since this value is minimized at $i \in M$. Therefore, the load of machine i after scheduling job k is upper bounded by

$$p_k + \sum_{j \in J : x_{ij} = 1} p_j < \frac{\varepsilon}{2} T + \left(1 + \frac{\varepsilon}{2}\right) T = (1 + \varepsilon) T. \qquad \square$$

The approximation scheme works as follows. We perform a binary search procedure to find the smallest integer value of T such that $\text{Las}^{\tau(\mathcal{C})}(\text{assign}(B, T)) \neq \emptyset$. To do this it is enough to consider the lower bound $1/m \sum_{j \in J} p_j$ on the optimal makespan, and the upper bound $\lceil 1/m \sum_{j \in J} p_j \rceil + \max_{j \in J} p_j$. For such value of T, thanks to Theorem 7 we obtain a schedule with makespan at most $(1 + \varepsilon)T$. In particular, $\text{Las}^{\tau(\mathcal{C})}(\text{assign}(J_{\text{long}}, B, C_{\max})) \neq \emptyset$ and therefore the makespan of this schedule is at most $(1 + \varepsilon)T \leq (1 + \varepsilon) C_{\max}$.

Regarding the running time, observe that the smallest job size in J_{long} is $(\varepsilon/2)T$, so we have $\lambda \leq 2/\varepsilon$. The size of $\mathcal{C}$ can also be upper bounded by $(1 + 2/\varepsilon)^{4/\varepsilon^2}$, since not more than $2/\varepsilon$ jobs can be allocated to single machine and the size of the partition is less than $4/\varepsilon^2$. Since the running time of the algorithm is dominated by finding a feasible solution at level $\tau(\mathcal{C}) < \frac{2}{\varepsilon}(1 + \frac{2}{\varepsilon})^{4\varepsilon^{-2}}$ of the Lasserre/SoS relaxation, the algorithm runs in polynomial time.

5.6 Proof of Theorem 6

Preliminaries. Before proceeding with the proof we need to introduce some properties that are satisfied by the relaxations obtained from the Lasserre/SoS hierarchy. Similarly to the SA and LS_+ hierarchies, we also have convergence in this case to the integer hull. In the following, we show a structural lemma that shows this property as a corollary, but more crucially, a property that in general is not satisfied by weaker hierarchies, known as the Decomposition Theorem [62]. We make an exposition that is self-contained and we include the proof of these results in the context of the assignment polytope. The following properties are very simple and they will be useful throughout the section.

Proposition 26. *Let $x \in Las^r(assign(B, T))$. Then, the following holds:*

i) For every $I \in \mathcal{P}_{r+1}(M \times J)$, $x_I \in [0, 1]$.

ii) If $L \in \mathcal{P}_{r+1}(M \times J)$ and $I \subseteq L$, then $x_L \leq x_I$.

Proof. To check i) it is enough to consider the principal submatrix of $M_{r+1}(x)$ given by the entries $\emptyset$ and I. Since it is positive semidefinite as well, the determinant is non-negative, that is $x_I(1 - x_I) \geq 0$ and then $x_I \in [0, 1]$. To check ii) it is enough

to consider the principal submatrix of $M_{r+1}(x)$ given by the entries I and L. This is positive semidefinite, and so its determinant is non-negative. Since $x_{I\cup L} = x_L$ it holds that $x_L(x_I - x_L) \geq 0$, thus $x_L \leq x_I$. □

Given $x \in \mathrm{Las}^r(\mathrm{assign}(B,T))$, the *extension* of x, $\mathrm{ext}(x)$, is the vector with entries in $\mathcal{P}(M \times J)$ such that $\mathrm{ext}(x)_I = 0$ if $|I| > 2r+2$ and $\mathrm{ext}(x)_I = x_I$ otherwise. Given a machine $i \in M$ and a subset of jobs $K \subseteq J$, let $x(i,K)$ be a vector with entries in $\mathcal{P}_{2(r-|K|)+2}(M \times J)$ such that

$$x(i,K)_I = \sum_{H \subseteq J : K \subseteq H} (-1)^{|H \setminus K|} \mathrm{ext}(x)_{I \cup (\{i\} \times H)},$$

for every $I \in \mathcal{P}_{2(r-|K|)+2}(M \times J)$. The following property is usually called the *Möbius inversion* in the moment matrices literature, or *inclusion-exclusion* in the probability context.

Lemma 27. *Let $x \in Las^r(assign(B,T))$ and $i \in M$. Then, $x = \sum_{K \subseteq J} x(i,K)$.*

Proof. Let $I \in \mathcal{P}_{2r+2}(M \times J)$. By substituting, parameterizing in $H \setminus K$ and reordering the summation it follows that

$$\begin{aligned}
\sum_{K \subseteq J} x(i,K)_I &= \sum_{K \subseteq J} \sum_{H \subseteq J : K \subseteq H} (-1)^{|H \setminus K|} \mathrm{ext}(x)_{I \cup (\{i\} \times H)} \\
&= \sum_{H \subseteq J} \sum_{R \subseteq J : R \subseteq H} (-1)^{|R|} \mathrm{ext}(x)_{I \cup (\{i\} \times H)} \\
&= \sum_{H \subseteq J} \mathrm{ext}(x)_{I \cup (\{i\} \times H)} \sum_{R \subseteq J : R \subseteq H} (-1)^{|R|}.
\end{aligned}$$

Observe that if $H \neq \emptyset$, then the last term of the summation is equal to zero. Therefore, the summation above is equal to $\mathrm{ext}(x)_{I \cup (\{i\} \times \emptyset)} = x_I$. □

An important observation is that the equality above says that $x_\emptyset = \sum_{K \subseteq J} x(i,K)_\emptyset$ and $x_\emptyset = 1$, therefore

$$x = \sum_{K \subseteq J} x(i,K) = \sum_{K \subseteq J : x(i,K)_\emptyset > 0} x(i,K)_\emptyset \cdot \left(\frac{x(i,K)}{x(i,K)_\emptyset} \right),$$

so x is a convex combination of the vectors $\{x_{\mathcal{C}}(i,K) : K \subseteq J \text{ and } x(i,K)_\emptyset > 0\}$ where $x_{\mathcal{C}}(i,K) = x(i,K)/x(i,K)_\emptyset$. An important observation is that the vector $x_{\mathcal{C}}(i,K)$ is integral for machine i: If $j \in K$, it follows that

$$\begin{aligned}
x(i,K)_{ij} &= \sum_{H \subseteq J : K \subseteq H} (-1)^{|H \setminus K|} \mathrm{ext}(x)_{\{(i,j)\} \cup (\{i\} \times H)} \\
&= \sum_{H \subseteq J : K \subseteq H} (-1)^{|H \setminus K|} \mathrm{ext}(x)_{\emptyset \cup (\{i\} \times H)} = x(i,K)_\emptyset,
\end{aligned}$$

and therefore $x_{\mathcal{C}}(i,K)_{ij} = 1$. Now, given $j \notin K$, we split the summation between those sets that contain $K \cup \{j\}$ and those that do not. Therefore,

$$\begin{aligned} x(i,K)_{ij} &= \sum_{H \subseteq J : K \subseteq H} (-1)^{|H \setminus K|} \mathrm{ext}(x)_{\{(i,j)\} \cup (\{i\} \times H)} \\ &= \sum_{H \subseteq J \setminus \{j\} : K \subseteq H} ((-1)^{|H \setminus K|} + (-1)^{|H \setminus K|+1}) \mathrm{ext}(x)_{\{i\} \times (H \cup \{j\})} = 0. \end{aligned}$$

In the following we state the Decomposition Theorem [62] adapted to our purposes, which is a structural property of the Lasserre/SoS hierarchy that makes it stronger than others. We include its proof for the sake of completeness, although it follows the same lines than [62] and it could be skipped.

Theorem 8. *Let $x \in Las^r(assign(B,T))$ for some $r \geq \lambda$. Then, for every machine $i \in M$ and $K \subseteq J$ such that $x(i,K)_\emptyset > 0$, we have $x_{\mathcal{C}}(i,K) \in Las^{r-\lambda}(assign(B,T))$. Furthermore, $x_{\mathcal{C}}(i,K)_{ij} = 1$ for every $j \in K$ and zero otherwise.*

Proof of Theorem 8. It remains to show that $x_{\mathcal{C}}(i,K) \in \mathrm{Las}^{r-\lambda}(\mathrm{assign}(B,T))$. In fact, we show a stronger result: For every $i \in M$ and $K \subseteq J$, if $x(i,K)_\emptyset > 0$ then $x_{\mathcal{C}}(i,K) \in \mathrm{Las}^{r-|K|}(\mathrm{assign}(B,T))$. In particular, the constraints (5.2) at level $r = 0$ imply that K induces a configuration, $\mathcal{C}_i(x_{\mathcal{C}}(i,K)) \in \mathcal{C}$, and therefore $|K| \leq \lambda$.

The constraint (5.5) is clearly satisfied. We now check that constraint (5.4) is satisfied, that is, $M_{r-|K|+1}(x_{\mathcal{C}}(i,K)) \succeq 0$. Since $M_{r+1}(x) \succeq 0$, there exist a family of vectors $\{\alpha_I : I \in \mathcal{P}_{r+1}(M \times J)\}$ such that for every $U, V \in \mathcal{P}_{r+1}(M \times J)$, $x_{U \cup V} = \langle \alpha_U, \alpha_V \rangle$. Consider the vector $\alpha^{\mathcal{C}}(i,K)$ with entries in $\mathcal{P}_{r-|K|+1}(M \times J)$ such that

$$\alpha^{\mathcal{C}}(i,K)_I = \frac{1}{\sqrt{x(i,K)_\emptyset}} \sum_{H \subseteq J : K \subseteq H} (-1)^{|H \setminus K|} \alpha_{I \cup (\{i\} \times H)}.$$

We show that for every $U, V \in \mathcal{P}_{r-|K|+1}$, $x_{\mathcal{C}}(i,K)_{U \cup V} = \langle \alpha^{\mathcal{C}}(i,K)_U, \alpha^{\mathcal{C}}(i,K)_V \rangle$ and that implies constraint (5.4) is satisfied.

$$\begin{aligned} & x(i,K)_\emptyset \cdot \langle \alpha^{\mathcal{C}}(i,K)_U, \alpha^{\mathcal{C}}(i,K)_V \rangle \\ &= \sum_{H \subseteq J : K \subseteq H} \sum_{G \subseteq J : K \subseteq G} (-1)^{|H \setminus K| + |G \setminus K|} \langle \alpha_{U \cup (\{i\} \times H)}, \alpha_{V \cup (\{i\} \times G)} \rangle \\ &= \sum_{H \subseteq J : K \subseteq H} (-1)^{|H \setminus K|} \sum_{G \subseteq J : K \subseteq G} (-1)^{|G \setminus K|} x_{U \cup V \cup (\{i\} \times H) \cup (\{i\} \times G)}. \end{aligned}$$

Observe that $|U| \leq r - |K| + 1$ and $|V| \leq r - |K| + 1$, and therefore every term in the summation above is well defined. Consider $H \neq K$ and and element $h \in H \setminus K$. Then, the second summation is equal to

$$\sum_{G \subseteq J : K \cup \{h\} \subseteq G} ((-1)^{|G \setminus K|} + (-1)^{|G \setminus K|+1}) x_{U \cup V \cup (\{i\} \times H) \cup G} = 0,$$

and therefore in the summation above it only remains the term for $H = K$, that is,

$$\begin{aligned} & \sum_{G : K \subseteq G} (-1)^{|G \setminus K|} x_{U \cup V \cup (\{i\} \times K) \cup (\{i\} \times G)} \\ &= \sum_{G : K \subseteq G} (-1)^{|G \setminus K|} x_{U \cup V \cup (\{i\} \times G)} = x(i,K)_{U \cup V}, \end{aligned}$$

since $G \cup K = G$. To check the constraints (5.1), we have that given $I \in \mathcal{P}_{2r+1}(M \times J)$,

$$\begin{aligned}\sum_{\ell \in M} x(i,K)_{I \cup \{(\ell,j)\}} &= \sum_{\ell \in M} \sum_{H \subseteq J : K \subseteq H} (-1)^{|H \setminus K|} \mathrm{ext}(x)_{I \cup \{(\ell,j)\} \cup (\{i\} \times H)} \\ &= \sum_{H \subseteq J : K \subseteq H} (-1)^{|H \setminus K|} \sum_{\ell \in M} \mathrm{ext}(x)_{I \cup \{(\ell,j)\} \cup (\{i\} \times H)} \\ &= \sum_{H \subseteq J : K \subseteq H} (-1)^{|H \setminus K|} \mathrm{ext}(x)_{I \cup (\{i\} \times H)} = x(i,K)_I,\end{aligned}$$

since $\mathrm{ext}(x)$ satisfies the constraint (5.5) for I as well. Finally, it remains to check constraints (5.2) and (5.3). Let ρ be any vector in $\{\mathrm{pack}^i, \mathrm{break}^i : i \in M \setminus \{m\}\} \cup \{\mathrm{pack}^m\}$. Since $M_r(\rho \star x) \succeq 0$, there exists a family of vectors $\{\beta_I : I \in \mathcal{P}_r(M \times J)\}$ such that for every $U, V \in \mathcal{P}_r(M \times J)$, $(\rho \star x)_{U \cup V} = \langle \beta_U, \beta_V \rangle$. Let $\beta^{\mathcal{C}}(i,K)$ be the vector with entries in $\mathcal{P}_{r-|K|}(M \times J)$ such that

$$\beta^{\mathcal{C}}(i,K)_I = \frac{1}{\sqrt{x(i,K)_\emptyset}} \sum_{H \subseteq J : K \subseteq H} (-1)^{|H \setminus K|} \beta_{I \cup (\{i\} \times H)}.$$

We show next that for every $U, V \in \mathcal{P}_{r-|K|}$, $(\rho \star x_{\mathcal{C}}(i,K))_{U \cup V} = \langle \beta^{\mathcal{C}}(i,K)_U, \beta^{\mathcal{C}}(i,K)_V \rangle$ and that implies constraint (5.4) is satisfied. First, observe that

$$\begin{aligned}&x(i,K)_\emptyset \cdot \langle \beta^{\mathcal{C}}(i,K)_U, \beta^{\mathcal{C}}(i,K)_V \rangle \\ &= \sum_{H : K \subseteq H} (-1)^{|H \setminus K|} \sum_{G : K \subseteq G} (-1)^{|G \setminus K|} (\rho \star x)_{U \cup V \cup (\{i\} \times H) \cup (\{i\} \times G)}.\end{aligned}$$

We have that $|U| \leq r - |K|$ and $|V| \leq r - |K|$, and therefore every term in the summation above is well defined. As we showed before, in the summation above it only remains the term for $H = K$. Therefore,

$$\begin{aligned}\sum_{G : K \subseteq G} (-1)^{|G \setminus K|} (\rho \star x)_{U \cup V \cup (\{i\} \times (K \cup G))} &= \sum_{G : K \subseteq G} (-1)^{|G \setminus K|} (\rho \star x)_{U \cup V \cup (\{i\} \times G)} \\ &= (\rho \star x(i,K))_{U \cup V},\end{aligned}$$

and we conclude using that $\rho \star x(i,K) = x(i,K)_\emptyset \cdot (\rho \star x_{\mathcal{C}}(i,K))$. □

Other relevant property is that if the vector x was integral at some machine, then it remains integral for a vector in the convex combination of the decomposition.

Proposition 28. *Suppose that $x \in Las^r(assign(B,T))$ with $r \geq \lambda$, and it is integral at machine h. Let $K \subseteq J$ with $x_{\mathcal{C}}(i,K)_\emptyset > 0$. Then, $x_{\mathcal{C}}(i,K)_{hj} = x_{hj} \in \{0,1\}$ for all $j \in J$.*

Proof. Let $j \in J$ such that $x_{hj} = 0$. Observe every term in the summation defining $x_{\mathcal{C}}(i,K)_{hj}$ contains $\{(h,j)\}$, and then by Proposition 26 they are all zero. It follows that $x_{\mathcal{C}}(i,K)_{hj} = 0$. On the other hand, if $x_{hj} = 1$, since $\sum_{\ell \in M} x_{\ell j} = 1$ it follows that $x^{\ell j} =$ for every $\ell \neq h$. By using the argument above and since $x_{\mathcal{C}}(i,K) \in \mathrm{assign}(B,T)$ by Theorem 8, it follows that $x_{\mathcal{C}}(i,K)_{\ell j} = 0$ for every $\ell \neq h$ and therefore $x_{\mathcal{C}}(i,K)_{hj} = 1 - \sum_{\ell \in M} x_{\mathcal{C}}(i,K)_{\ell j} = 1$. □

Constructing the schedule. The proof of Theorem 6 follows in a constructive way. We iteratively call the Decomposition Theorem 8 and obtain, in that way, many machines that are integral. If we start from a high enough level of the hierarchy, we get at the end a solution that is feasible for assign(B,T), and therefore, the configurations of the integral machines have to obey the lexicographic order. We then show how to extend the schedule to the rest of the machines using those configurations and prove this schedule to be feasible.

The algorithm consist of two phases. In Phase 1, we use the solution obtained from high enough level of the hierarchy to find the last machine which is fractionally scheduled according to configuration C^1 using the Decomposition Theorem, and pick such a vector obtained from the convex combination. We then proceed by finding the last machine scheduled fractionally according to C^2, and so on. We end up with a solution that is integral for all these machines, and it respects the lexicographic order over the configurations. In Phase 2, we construct the schedule for the rest of the machines and jobs that have not been assigned yet. We call the schedule obtained in this way the *lexicographic schedule*, x^{lex}. Below we provide the pseudocode of the construction, and then we prove it outputs a feasible schedule.

Algorithm 2

Input: A solution $x \in \text{Las}^{\tau(\mathcal{C})}(\text{assign}(B,T))$.
Output: A schedule x^{lex} with makespan at most T.

For all $(i,j) \in M \times J$, initialize $x^{\text{lex}}_{ij} \leftarrow 0$, $\ell \leftarrow 0$ and $y^0 \leftarrow x$.
▷ Phase 1: Inducing integrality of certain machines.
for $\ell = 1$ to $|\mathcal{C}|$ **do**
  Let $M^\ell = \{i \in M : \text{exists } K \subseteq J \text{ with } y^\ell(i,K)_\emptyset > 0 \text{ and } \text{conf}(K) = C^\ell\}$.
  if $M^\ell \neq \emptyset$ **then**
    let $i^\ell = \max M^\ell$ and $K^\ell \subseteq J$ such that $\text{conf}(K^\ell) = C^\ell$ and $y^\ell(i^\ell, K^\ell)_\emptyset > 0$,
    for every $j \in K^\ell$ let $x^{\text{lex}}_{i^\ell j} \leftarrow 1$,
    $y^{\ell+1} \leftarrow y^\ell_{\mathcal{C}}(i^\ell, K^\ell)$.
  else $i^\ell = \infty$
Reset $\ell \leftarrow 1$, $i^0 = 0$.
▷ Phase 2: Extending the above solution greedily.
for $\ell = 1$ to $|\mathcal{C}|$ **do**
  if $i^\ell < \infty$ **then**
    for $\max\{i^q : 0 \le q < \ell, i^q < \infty\} < i < i^\ell$ **do**
      let $K^i \subseteq J \setminus \{j \in J : \text{exists } i \in M \text{ with } x^{\text{lex}}_{ij} = 1\}$ where $\text{conf}(K^i) = C^\ell$;
      for every $j \in K^i$, let $x^{\text{lex}}_{ij} \leftarrow 1$.
Return x^{lex}.

Lemma 29. *Let $x \in Las^r(assign(B,T))$ for $r \ge \lambda$, and $i,h \in M$ with $i < h$. Suppose that x is integral for machine h. Then, for every I such that $x(i,I)_\emptyset > 0$, we have $conf(I) \ge_{lex} \mathcal{C}_h(x)$.*

Proof. Let $I \subseteq J$ with $x_{\mathcal{C}}(i,I)_\emptyset > 0$. By Theorem 8 we have that $x_{\mathcal{C}}(i,I) \in$

assign(B,T), $x_{\mathcal{C}}(i,I)_{ij} \in \{0,1\}$ for all $j \in J$ and $\text{conf}(I) = \mathcal{C}_i(x_{\mathcal{C}}(i,I))$. In particular, from the symmetry breaking constraints it follows that $\mathcal{L}_B(\mathcal{C}_i(x_{\mathcal{C}}(i,I))) \geq \mathcal{L}_B(\mathcal{C}_h(x))$, since by Proposition 28 the vector $x_{\mathcal{C}}(i,I)$ remains integral at h and in the same configuration than x. The function $\mathcal{L}_B$ is strictly increasing, and therefore $\text{conf}(I) = \mathcal{C}_i(x_{\mathcal{C}}(i,I)) \geq_{\text{lex}} \mathcal{C}_h(x)$. □

Observe that it is guaranteed by the algorithm that machine m is integral at the end of Phase 1. In the following, we prove that at the end of every iteration of Phase 1, the machines between other two that have been induced to be integral by using the decomposition are all fractionally scheduled according to the same configuration. In particular, it guarantees that the greedy approach of Phase 2 works.

Lemma 30. *Let $\ell \in \{1,\dots,|\mathcal{C}|\}$. For every $i \in M$ such that $i_\ell = \max\{i^q : 0 \leq q < \ell, i^q < \infty\} < i \leq i^\ell$ and every $q \in \{1,\dots,s\}$, $\sum_{j\in J_q} y^\ell_{ij} = m(q, C^\ell)$.*

Proof. Let $i \in M$ be a machine such that $i_\ell < i \leq i^\ell$. By Lemma 29, for every $K \subseteq J$ such that $y^\ell(i,K)_\emptyset > 0$ it holds that $\text{conf}(K) \geq_{\text{lex}} C^\ell$, but we show next that they are all equalities. Suppose there exists $K \subseteq J$ such that $C^t = \text{conf}(K) >_{\text{lex}} C^\ell$ for some $t < \ell$. That would imply that $i \leq i^t < \infty$, contradicting that $i_\ell < i$. By Lemma 27, for every $q \in \{1,\dots,s\}$ it holds that

$$\begin{aligned}\sum_{j\in J_q} y^\ell_{ij} &= \sum_{j\in J_q} \sum_{K\subseteq J: y^\ell(i,K)_\emptyset > 0} y^\ell(i,K)_\emptyset \cdot y_{\mathcal{C}}(i,K)_{ij} \\ &= \sum_{K\subseteq J: y^\ell(i,K)_\emptyset > 0} y^\ell(i,K)_\emptyset \sum_{j\in J_q} y_{\mathcal{C}}(i,K)_{ij} \\ &= \sum_{K\subseteq J: y^\ell(i,K)_\emptyset > 0} y^\ell(i,K)_\emptyset \cdot m(q,C^\ell) = y^\ell_\emptyset \cdot m(q,C^\ell) = m(q,C^\ell).\end{aligned}$$ □

Proof of Theorem 6. For every $\ell \in \{1,\dots,|\mathcal{C}|\}$, let $N^\ell = \{i \in M : i_\ell < i < i^\ell\}$. Observe that some of these sets could be equal to the $\emptyset$. Thanks to Lemma 30, at the end of Phase 1 we obtain a solution $y^{|\mathcal{C}|}$ satisfying every constraint of assign(B,T), it is integral for every machine in $M \setminus \cup_{\ell=1}^{|\mathcal{C}|} N^\ell$ and every machine in N^ℓ is fractionally scheduled according to configuration C^ℓ, for all $\ell \in \{1,\dots,|\mathcal{C}|\}$. We now see how we construct a schedule with makespan at most $\alpha \cdot T$ for the jobs that have not been scheduled in Phase 1, and using only the machines in $\tilde{M} = \cup_{\ell=1}^{|\mathcal{C}|} N^\ell$.

Left $\tilde{J}$ be all the jobs that have not been scheduled in Phase 1. Consider the bipartite graph with nodes given by the jobs $\tilde{J}$ on one side, and the other side are the set of nodes $\mathcal{R} = \tilde{M} \times \{1,\dots,s\}$. There is an edge between $j \in \tilde{J}$ and $(i,q) \in \mathcal{R}$ if $y^{|\mathcal{C}|}_{ij} > 0$ and $j \in J_q$. We consider a transportation problem where the offer of every node in $\tilde{J}$ is exactly 1, and the demand of a node (i,q) is equal to $m(q,C^\ell)$ if $i \in N^\ell$. By construction the total offer equals to total demand and $y^{|\mathcal{C}|}$ is a fractional solution to this problem. Therefore, since this is a feasible transportation problem, the integrality of the flow formulation implies that exists an integral solution and a way of implementing Phase 2.

Given $\ell \in \{1, \ldots, |\mathcal{C}|\}$ with $i^\ell < \infty$, in Phase 2 we have that every machine $i \in N^\ell$ is such that $\text{conf}(K^i) = C^\ell = \text{conf}(K)$, where K^i are the jobs scheduled to i in x^{lex}, and K the jobs scheduled to i^ℓ. The partitioning of the jobs is α-balanced, so we have that

$$\sum_{j \in J} p_j x_{ij}^{\text{lex}} = \sum_{j \in K^i} p_j \leq \alpha \sum_{j \in K} p_j = \alpha \sum_{j \in J} p_j x_{i^\ell j}^{\text{lex}} \leq \alpha T,$$

since the load at machine i^ℓ is guaranteed to be at most T thanks to Phase 1. That concludes the proof. □

Part II

Online Optimization: Selection Problems

Chapter 6

Introduction

In the classical secretary problem (see, e.g., [41] for a survey) an employer wants to select exactly one out of n secretaries arriving in random order. After each arrival, the employer learns the relative merits of the new candidate (i.e., he can compare the candidate with previous ones but no numerical quantity is revealed), and must reject or accept immediately. Lindley [75] and Dynkin [35] show that the strategy of sampling $1/e \approx 0.37$ fraction of the candidates and then selecting the first record has a probability of at least $1/e \approx 0.37$ of selecting the best candidate and that no algorithm can beat this constant. During the last decade, generalizations of this problem have attracted the attention of researchers, specially due to applications in online auctions and online mechanism design.

Arguably one the most natural extension is the *generalized secretary problem* by Babaioff et al. [11]. In their setting, a set R of candidates of known size is presented to an algorithm on uniform random order. On arrival, each element r reveals its hidden weight $w(r)$ and the algorithm must irrevocably decide whether to select it or not while preserving the independence of the set of selected elements on a given independence system[1] $(R, \mathcal{I})$. The objective is to maximize the total weight of the selected independent set. Mainly motivated by applications and by the richness of their structural properties, Babaioff et al. focuses on the case in which $(R, \mathcal{I})$ is a *matroid*, [2], where this problem takes the name of *Matroid Secretary Problem* (MSP). In the following, an α *utility-competitive* algorithm is one that returns an independent set whose expected weight is at least $1/\alpha$ times the weight of an optimum independent set.

Babaioff et al. posts the well-known, and still open, *Matroid Secretary Conjecture* which in its weak form states that there must be a constant competitive algorithm for the MSP on any matroid, and on its strong form, claims that the constant is e. They also provide a *Threshold Price Algorithm* (TPA) achieving an $O(\log \rho)$ utility-competitive ratio on matroids of rank ρ. Better algorithms have improved this ratio

[1] An independence system is a pair $(R, \mathcal{I})$ where R is finite, and $\mathcal{I}$ is a nonempty family of subsets of R that is closed under inclusion. The sets in $\mathcal{I}$ are called independent sets.

[2] A matroid is an independence system $(R, \mathcal{I})$ satisfying the next augmentation property: whenever $X, Y \in \mathcal{I}$, and $|X| < |Y|$, there must be an element $r \in Y \setminus X$ such that $X + r \in \mathcal{I}$.

to $O(\sqrt{\log \rho})$ by Chakraborty and Lachish [22] and later to the current best ratio of $O(\log \log \rho)$ by Lachish [70] and independently by Feldman et al. [39].

The generalized secretary problem has also been studied on non-matroidal systems such as knapsack [10], online matchings on graphs and hypergraphs [32, 64, 68] and LP packings [65]. Babaioff et al. [11] show that for general independence systems every algorithm on n elements must be $\Omega(\log n / \log \log n)$ utility-competitive and Rubinstein [91] provides an $O(\log n \log \rho)$ competitive algorithm. Many works consider alternative objective functions, such as minimum sum of ranks [1], time discounted variants [9], convex costs under knapsack type restrictions [15] and submodular objectives [17, 38, 40]. Another rich line of research focuses on relaxing the random order condition and the power of the adversary setting the weights. There are $O(1)$ competitive algorithms for the MSP on the random order and assignment model [96], on the adversarial order random assignment model [87, 96] and on the free order model [56]. Non-uniform arrival orders have also been considered [63]. Related to this line of research is the design of order-oblivious algorithms, which can sample a constant fraction of the elements but afterwards, the arrival order may be arbitrary; and results related to prophet inequality settings on which the weight of each element is a random variable with known distribution but the arrival order is adversarial, see [8, 34, 67, 92].

6.1 Ordinal MSP versus Utility MSP

A common requirement for the mentioned algorithms for the MSP on general matroids is that they require to use numerical weights: they all treat elements with similar weights more or less equivalently. In contrast, in the definition of the classical secretary problem, the decisions made by an algorithm only rely on the ordinal preferences between candidates. This is a very natural condition as for many applications it is difficult to determine (for instance, via an interview) a numerical weight representing each candidate, but it is often simple to compare any two of them. For this reason we restrict our study to the *ordinal MSP* in which the decision maker can only compare seen elements. This is, it can check which element is higher in an underlying hidden total order $\succ$. In particular, the algorithm cannot really compute the *weight* of a set.

However, matroids have the following extremely nice feature: it is possible to find a maximum weight independent set using only ordinal information by greedily constructing an independent set using the order $\succ$. The set OPT found this way is the unique lexicographically[3] maximum independent of the matroid. So it is quite natural to ask what can be done without numerical weights in the secretary setting. In order to measure the performance of an algorithm we introduce three notions of competitiveness for the ordinal MSP. We say that an algorithm is α *ordinal-competitive* if for every nonnegative weight function compatible with the total order, the expected weight of the output independent set ALG is at least $1/\alpha$ the weight of OPT. It is α *intersection-competitive* if the expected fraction of elements in OPT that is in-

[3] A set $A = \{a_1, \ldots, a_k\} \subseteq R$ is lexicographically larger than a set $B = \{b_1, \ldots, b_k\} \subseteq R$ of the same size, with respect to the order $\succ$ on $A \cup B$ if $a_i \succ b_i$ for the first i on which a_i and b_i differ.

cluded into ALG is at least $1/\alpha$, and it is α *probability-competitive* if for every element $r \in \text{OPT}$, the probability that r is included into ALG is at least $1/\alpha$. It is not hard to see that any α ordinal-competitive algorithm is also α competitive in the classical (utility) sense and also that any α probability-competitive algorithm is also α competitive in every other setting. All the mentioned algorithms [11, 22, 39, 70] for the standard or *utility MSP* do not work in the ordinal model, and so the existence of an α-utility competitive algorithm does not imply the existence of an α probability/intersection/ordinal competitive algorithm. To the authors' knowledge the only known algorithms for ordinal MSP on general matroids (apart from the trivial $O(\rho)$ probability-competitive that selects the top independent singleton using a classical secretary algorithm) are Bateni et al.'s $O(\log^2 \rho)$ ordinal-competitive algorithm [17] (which also works for submodular objective functions) and Soto's variant of TPA [96] which is $O(\log \rho)$ ordinal-competitive.

Hoefer and Kodric [52] study the ordinal secretary problem obtaining constant ordinal-competitive algorithms for bipartite matching, general packing LP and independent sets with bounded local independence number. They also show that Feldman and Zenklusen's reduction [40] from submodular to linear MSP works in the ordinal model.

MSP on specific matroids. An extensive amount of work has been done on the last decade on the MSP on specific matroid classes including unitary [18, 20, 35, 47, 75], uniform [10, 66], transversal [11, 32, 68], graphic [9, 11, 68], cographic [96], regular and MFMC [33], k-column sparse [96] and laminar matroids [55, 56, 79]. Even though the algorithms are stated for the utility MSP, many of the proofs work directly on either the ordinal MSP under ordinal or even probability competitiveness. We include in Table 6.1 a summary of all known results together with the improved bounds obtained in this paper.

6.2 Our results and techniques

We first formalize our new competitiveness notions for ordinal MSP and study their interrelations. We say that a performance notion is stronger than another if any algorithm that is α competitive for the former is also α competitive for the latter. Under this definition, we show that probability is stronger than ordinal and intersection, and that ordinal is stronger than utility.

We first focus on developing a powerful technique to define strong algorithms for the MSP on many classes of matroids. Informally (see the exact definition in Chapter 7), we say that an algorithm has forbidden set of size k if it samples without selecting $s \sim \text{Bin}(n, p)$ elements and the following condition holds. Suppose that an element r^* of OPT arrives in a position $t > s$ and let R_t be the set of elements that arrived on or before time t. For each time step i between $s+1$ and $t-1$ there is a random set $\mathcal{F}_i$ of at most k forbidden elements such that if for every i, the element arriving at time i is not forbidden ($i \notin \mathcal{F}_i$) then r^* is sure to be selected. The following is the key lemma

Matroid Class	Previous guarantees (u,o,p competitive)	New algorithms		
		p-approx	ref.	forb.
Transversal	o: 16 [32], 8 [68], e [64]	e	Alg. 3	1
μ exch. gammoids (type of hypermatching)	u: $O(\mu^2)$ [68], $e\mu$ [64]	$\mu^{\mu/(\mu-1)}$	Alg. 4	μ
Matching matroids	-	4	Alg. 5	2
μ exch. matroid packings	-	$\mu^{\mu/(\mu-1)}$	Alg. 5	μ
Graphic	o: 16 [11], $3e$ [9], $2e$ [68]	4	Alg, 6	2
Hypergraphic	-	4	Alg, 6	2
k-sparse matroids	o: ke [96]	$k^{k/(k-1)}$	Alg. 7	k
k-framed matroids	-	$k^{k/(k-1)}$	Alg. 7	k
Semiplanar gammoids	-	$4^{4/3}$	Alg. 8	4
Laminar	o: 177.77 [55] o: $3\sqrt{3}e \approx 14.12$ [56] p: 9.6 [79]	$3\sqrt{3} \approx 5.196$	Alg. 9	3
Uniform $U(n, \rho)$	p: e [10] o: $1 + O(\sqrt{1/\rho})$	$1 + O(\sqrt{\log \rho/\rho})$	Alg. 10	-
Cographic	p: $3e$ [96]	-	-	-
Regular, MFMC	o: $9e$ [33]	-	-	-

Table 6.1: State-of-the-art competitive ratios for all known matroid classes, including our results.

that shows why algorithms with small forbidden sets are useful for the MSP.

Lemma 31 (Key Lemma)**.** *By setting the right sampling probability $p = p(k)$, every algorithm with forbidden sets of size k is $\alpha(k)$ probability-competitive, where*

$$(p(k),\ \alpha(k)) = \begin{cases} (1/e,\ e) & \text{if } k = 1, \\ (k^{-\frac{1}{k-1}},\ k^{\frac{k}{k-1}}) & \text{if } k \geq 2. \end{cases}$$

Obtaining algorithms with small forbidden sets is simple for many matroid classes. In fact, it is easy to see that the variant of the standard classical secretary algorithm which samples $s \sim \text{Bin}(n, p)$ elements and then selects the first element better than the best sampled element, has forbidden sets of size 1. Suppose that the maximum element r^* arrives at time $t > s$ and denote by x the second best element among those that have arrived up to time t. If x does not arrive at any time between $s + 1$ and $t - 1$ then for sure, x will be used as threshold and thus r^* will be selected. In the above notation, all forbidden sets $\mathcal{F}_i$ are equal to the singleton $\{x\}$. Using the key lemma, by setting $p = 1/e$, this algorithm is e competitive.

We provide new algorithms that beat the state of the art guarantees for transversal, graphic, k-sparse and laminar matroids. We also provide new algorithms for other classes of matroids such as matching matroids, certain matroidal graph packings (which generalize matching matroids), hypergraphic matroids, k-framed matroids (which generalize k-sparse matroids), semiplanar gammoids and low exchangeability

gammoids. As an interesting side result, we revisit Kleinberg's $1 + O(\rho^{-1/2})$ ordinal-competitive algorithm. We show that its probability-competitiveness is bounded away from 1 and propose an algorithm that achieves a probability-competitive ratio of $1 + O(\sqrt{\log \rho/\rho})$. Our new results for specific classes of matroids are summarized in Table 6.1, which, for completeness includes all matroid classes ever studied on the MSP, even those for which we couldn't improve the state of the art. In the references within the table u, o and p stand for utility, ordinal and probability competitiveness respectively.

Kesselheim et al.'s algorithm for online bipartite matching [64] uses a technique very similar to that in the previous lemma to compute the expected weight contribution of the k-th arriving vertex, which is enough to prove asymptotically $e + O(1/n)$ utility-competitiveness for online bipartite matchings. In fact, for transversal matroids, this yields to an $e + O(1/n)$ ordinal-competitive algorithm. But their analysis does not imply any guarantee on the probability notion. In the next section we show by using a different analysis, that their algorithm has forbidden sets of size 1 and so it is actually e probability-competitive (always, not just asymptotically).

We obtain as well results for the ordinal MSP on general matroids. For the weak intersection notion one can show that exists a $\ln(2/e)$ intersection-competitive algorithm for the MSP using a greedy variant. More interestingly, we show that in the ordinal notion we can match the factor attained for the weaker utility variant. For the stronger probability notion we do not attain the same competitiveness ratio.

Theorem 9. *There exist an $O(\log\log\rho)$ ordinal-competitive algorithm and an $O(\log\rho)$ probability competitive algorithm for the MSP.*

Even though Theorem 17 is attained by a very simple algorithm, we note that standard ideas such as thresholding do not work for the intersection notion since elements outside OPT that are higher than actual elements from OPT do not contribute to the objective. The algorithms mentioned in Theorem 9 are based on the recent $O(\log\log\rho)$ utility-competitive algorithm by Feldman et al. [39]. Their algorithm samples a fraction of the elements so as to classify most of the non-sampled ones into $h = O(\log\rho)$ weight classes consisting off elements whose weights are off by at most a factor of 2. They implement a clever random strategy to group consecutive weight classes together into buckets, each one containing roughly the same random number of weight classes. On each bucket they define a single random matroid with the property that if one picks an independent set from each one of these matroids, their union is independent in the original one. Their algorithm then selects a greedy independent set on each bucket matroid and outputs their union. Due to the random bucketing, the expected number of elements selected on each weight class is at least $\Omega(1/\log h)$ times the number of elements that OPT selects from the same class. This means that the global algorithm is actually $O(\log h) = O(\log\log\rho)$ utility-competitive.

In the ordinal setting we cannot implement this idea in the same way, since basically we do not have any weights. However, we can still partition the elements of the matroid into *ordered layers*. The idea is to select a collection of thresholds obtained from the optimum of a sample, and use them as separators to induce the layers. This

motivates the definition of the Layered-MSP (see Section 10.1). For both ordinal and probability competitiveness we provide a reduction from the Layered-MSP to the ordinal MSP. For the ordinal notion, we select as thresholds a geometrically decreasing (in value order) subset of the sample optimum, so we can partition the matroid into $h = O(\log \rho)$ layers. For the probability notion we use all the elements of the sample optimum as thresholds. The crucial result in this part is that our reduction allows to go from any $g(h)$ competitive algorithm for the Layered-MSP to a $g(O(1 + \log h))$ ordinal-competitive algorithm, and to a $g(O(h))$ probability-competitive algorithm. In particular, by applying Feldman et al.'s algorithm, interpreting these layers as weight classes, we get an $O(\log \log \rho)$ ordinal-competitive algorithm and an $O(\log \rho)$ probability-competitive algorithm for the original matroid.

6.3 Organization

In Section 6.4 we recall some definitions, fix some notation and formally describe the performance guarantees for the ordinal MSP, studying their relations. In Chapter 7 we prove our key lemma for algorithms with small forbidden sets. We then in Chapter 8 show how devise simple algorithms for all the matroid classes mentioned using the forbidden sets technique. The competitiveness ratios are summarized in Table 6.1. In Chapter 9 we show an asymptotically 1 probability-competitive algorithm for uniform matroids. Finally, in Chapter 10 we describe our new algorithms for general matroids by proving Theorem 9 and we study the relations between the different competitiveness notions.

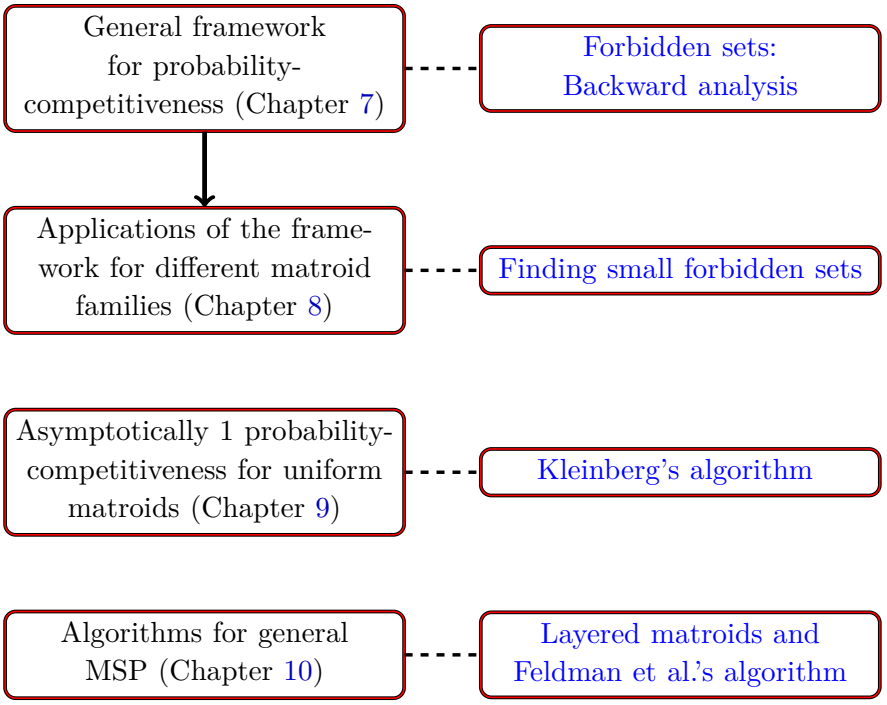


Figure 6.1: Organization of Part II.

6.4 Preliminaries

Recall that given a ground set R, a *matroid* $(R, \mathcal{I})$ is an ordered pair where $\mathcal{I} \subseteq \mathcal{P}(R)$ is non-empty and it satisfies the following two conditions:

(M1) if $I \in \mathcal{I}$ and $I' \subseteq I$, then $I' \in \mathcal{I}$,

(M2) if I_1 and I_2 are in $\mathcal{I}$ and $|I_1| < |I_2|$, then there is an element $e \in I_2 \setminus I_1$ such that $I_1 \cup \{e\} \in \mathcal{I}$.

Every family satisfying condition (M1) is called an *independence system*. The condition (M2) is known as the *augmentation axiom*. A set in $\mathcal{I}$ that is a maximal independent set for the inclusion is called a *base*. To make notation lighter, we use $+$ and $-$ for the union and difference of a set with a single element respectively. That is, $Q + r - e = (Q \cup \{r\}) \setminus \{e\}$. By the augmentation axiom is clear that every base has the same cardinality and this number is called the *rank* of a matroid. There is an stronger statement than (M2) which is called the *strong basis exchange*: For every pair of bases $X, Y \in \mathcal{I}$ and for all $x \in X \setminus Y$, there exist $y \in Y \setminus X$ such that $X - x + y \in \mathcal{I}$ and $Y - y + x \in \mathcal{I}$. That is, there exists a way of swaping x with an element y that preserves the independence of both sets.

Example 1. Given a set R and $k \in \mathbb{N}$, such that $|R| \geq k$, consider $\mathcal{I} = \{F \subseteq R : |R| \leq k\}$, that is, every subset of cardinality at most k. This is called the *k-uniform matroid* of R and its bases are the subsets of R with cardinality exactly equal to k.

Example 2. Given a graph $G = (V, E)$, consider $\mathcal{I}$ to be the set of all subsets of edges $X \subseteq E$ such that (V, X) is a forest. The pair $(E, \mathcal{I})$ is called the *graphic matroid* of G, and its bases are the maximal forests in G. In particular, when G is connected, the bases are the subsets of edges forming a spanning tree of G.

Example 3. Given a matrix $A \in \mathbb{R}^{L \times R}$, consider $\mathcal{I}$ to be the set of subsets $X \subseteq R$ such that the columns indexed by X are linearly independent in $\mathbb{R}^L$. The pair $(R, \mathcal{I})$ is called the *linear matroid* of A. In particular, every graphic matroid can be *represented* by considering the linear matroid obtained from the adjacency matrix of G. We will go back later in this part to this notion of matroid representation.

The matroids we consider are *ordered*, i. e., there is a. strict total order $\succ$ over the ground set $R = \{r^1, \ldots, r^n\}$ and we denote it by $\mathcal{M} = (R, \mathcal{I}, \succ)$. We call $\succ$ the *value order* and say that r^1 is the highest valued element, r^2 is the second one, and so on, then $r^1 \succ r^2 \succ \cdots \succ r^n$. The elements of $\mathcal{I}$ are ordered according to the *lexicographic order* induced by $\succ$.

There are many combinatorial problems that can be modeled as an ordered matroid, and the following is a classic example. Given a graph G, suppose we have a total order over the edges and we look for the lexicographically maximum spanning tree. This problem can be solved by running the *greedy Kruskal's algorithm*. Usually the

problem is posed considering weights over the edges instead of an order. In that case, one looks for a spanning tree of maximum weight, which is not necessarily unique. We say that a nonnegative weight function $w\colon E \to \mathbb{R}_+$ is compatible with the value order if the following holds: For every $i, j \in [n]$, if $r^i \succ r^j$ then $w(r^i) \geq w(r^j)$.

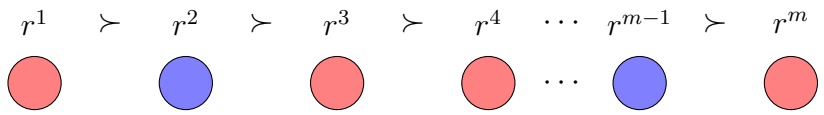


Figure 6.2: The optimal lexicographic base (red elements) can be found by the greedy algorithm feeded with elements from highest to smallest in the order. An element is selected as long as it preserves the independence.

In terms of matroids, the problem above is to find the lexicographically optimum base of the graphic matroid of G, and naturally, the same question can be formulated for a general matroid. The greedy algorithm solves this problem: From the highest to smallest element, select an element and add it to the current solution if this union is an independent set. By matroid properties, for every subset $Q \subseteq R$, there is a unique lexicographically optimum base $\mathrm{OPT}(Q)$ obtained by applying the greedy algorithm in the order $\succ$, over the set Q. Note that for every compatible weight function the set $\mathrm{OPT} = \mathrm{OPT}(E)$ is a maximum weight independent set. We reserve the use of superscripts $k \in \mathbb{N}$ on a set Q to denote the subset of the highest $\min\{k, |Q|\}$ valued elements of Q. In particular R^k and OPT^k denote the set of the top k elements of the matroid and of OPT respectively. We also reserve n and ρ to denote the number of elements of R and its rank respectively.

The definition of rank extends for subsets of R, that is, the rank of $Q \subseteq R$ is the cardinality of its bases, $\rho(Q) = \max\{|I| : I \in \mathcal{I},\ I \subseteq Q\}$, and the *span* of Q is $\mathrm{span}(Q) = \{r \in R : \rho(Q + r) = \rho(Q)\}$. In matroids, OPT has the property of *improving* any subset in the following sense.

Lemma 32. *Let $Q \subseteq R$. Then,* $\mathrm{OPT} \cap Q \subseteq \mathrm{OPT}(Q)$.

Proof of Lemma 32. Let $r = r^k$ be an element of $\mathrm{OPT} \cap Q$. Since r^k is selected by the Greedy algorithm we have that $r \notin \mathrm{span}(R^{k-1})$. But then $r \notin \mathrm{span}(R^{k-1} \cap Q)$ and so it is also selected by the Greedy algorithm applied only on the set Q. Therefore, $r \in \mathrm{OPT}(Q)$. □

In the (utility/ordinal) MSP, the elements of a (nonnegatively weighted/ordered matroid) are presented in *uniform random order* to an online algorithm that does not know a priori the (weights/value order) of unrevealed elements. At any moment, the algorithm can (view the weight of/compare in the total order) any pair of revealed element. When a new element r is presented, the algorithm must decide whether to add r to the solution and this decision is permanent. The algorithm must guarantee

that the set of selected elements is at all times independent[4] in the matroid. The objective of the algorithm is to return a set ALG as close as OPT as possible according to certain competitiveness metric.

Recall that the algorithm considered for the MSP have access to the elements in an online and uniformly at random fashion. We denote by r_1 the first element arriving, r_2 the second, and so on. In general, $R_t = \{r_1, r_2, \dots, r_t\}$ is the set of elements arriving up to time t.

6.5 Measures of competitiveness: Ordinal MSP

Recall that an algorithm for the utility MSP returning a set ALG is $\alpha \geq 1$ utility-competitive if

$$\mathbb{E}[w(\text{ALG})] \geq w(\text{OPT})/\alpha. \tag{6.1}$$

We introduce three measures of competitiveness for the ordinal MSP. Recall that the algorithm only learns ordinal information about the elements but it cannot access numerical weights. In this sense, they are closer to the classical secretary problem than the utility variant (for a discussion about this aspect in the original secretary problem, see [41]).

Since the weight function remains completely hidden for the algorithm, the first measure of competitiveness we consider is the following. An algorithm is α *ordinal-competitive* if *for every* weight function w compatible with the value order, condition (6.1) holds. An equivalent characterization of competitiveness is obtained by the following lemma.

Lemma 33. *An algorithm is $\alpha \geq 1$ ordinal-competitive if and only if for every $k \in [n]$,*

$$\mathbb{E}|\text{ALG} \cap R^k| \geq |\text{OPT} \cap R^k|/\alpha. \tag{6.2}$$

Proof. Consider an α ordinal-competitive algorithm returning a set ALG and let $k \in [n]$. Define the weight function $w(r) = 1$ if $r \in R^k$ and zero otherwise, which is compatible with the value order. Then, $\mathbb{E}[|\text{ALG} \cap R^k|] = \mathbb{E}[w(\text{ALG})] \geq \frac{1}{\alpha} w(\text{OPT}) = \frac{1}{\alpha}|\text{OPT} \cap R^k|$. Now suppose that (6.2) holds. Then, for any compatible function w, and defining $w(r^{n+1}) := 0$,

$$\begin{aligned}\mathbb{E}[w(\text{ALG})] &= \sum_{k=1}^{n} (w(r^k) - w(r^{k+1})) \cdot \mathbb{E}[|\text{ALG} \cap R^k|] \\ &\geq \sum_{k=1}^{n} (w(r^k) - w(r^{k+1})) \cdot \frac{1}{\alpha}|\text{OPT} \cap R^k| = \frac{1}{\alpha} w(\text{OPT}).\end{aligned}$$ □

[4] For particular classes of matroids, we may assume that $\mathcal{M}$ is known beforehand by the algorithm, or alternatively that it is discovered by the algorithm via an independence oracle that allows it to test any subset of revealed elements. In any case, it is a standard assumption that the algorithm at least know the number of elements n in the matroid.

In the second measure we consider, we want to make sure that every element of the optimum is part of the output with large probability. We say that an algorithm is $\alpha \geq 1$ *probability-competitive* if for every $e \in \mathrm{OPT}$,

$$\Pr(e \in \mathrm{ALG}) \geq 1/\alpha. \tag{6.3}$$

Finally, in the third measure we want to maximize the number of elements in the optimum that the algorithm outputs. We say an algorithm is $\alpha \geq 1$ *intersection-competitive* if

$$\mathbb{E}[|\mathrm{OPT} \cap \mathrm{ALG}|] \geq |\mathrm{OPT}|/\alpha. \tag{6.4}$$

Relation between variants. The ordinal and probability measures are in fact stronger than the standard utility notion. That is, any α ordinal/probability-competitive algorithm yields to an α utility-competitive algorithm. Furthermore, the probability is the strongest of them all.

Lemma 34. *If an algorithm is α ordinal-competitive then it is α utility-competitive. If an algorithm is α probability-competitive then it is also α ordinal, utility and intersection-competitive.*

Proof of Lemma 34. If an algorithm is α ordinal-competitive then by definition it is α utility-competitive. Now consider an α probability-competitive algorithm returning a set ALG. For any $k \in [n]$, we have $\mathbb{E}[|\mathrm{ALG} \cap R^k|] \geq \mathbb{E}[|\mathrm{ALG} \cap \mathrm{OPT} \cap R^k|] = \sum_{r \in \mathrm{OPT} \cap R^k} \Pr(r \in \mathrm{ALG}) \geq \frac{|\mathrm{OPT} \cap R^k|}{\alpha}$. which means that the algorithm is α ordinal-competitive, and therefore, it is also α utility-competitive. To see that the algorithm is also α intersection-competitive we note that

$$\mathbb{E}[|\mathrm{ALG} \cap \mathrm{OPT}|] = \sum_{e \in \mathrm{OPT}} \Pr(e \in \mathrm{ALG}) \geq \frac{|\mathrm{OPT}|}{\alpha}. \qquad \square$$

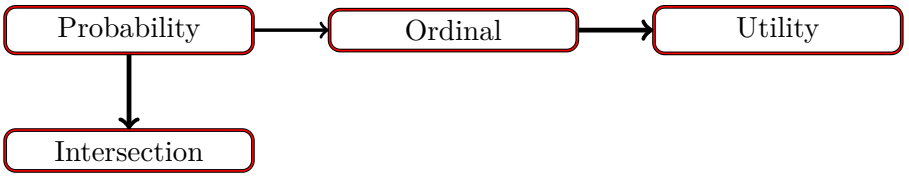


Figure 6.3: The probability notion is the strongest. The intersection notion is incomparable to the ordinal one.

We provide in Chapter 10 a comparison between the different competitiveness measures. An algorithm for the utility variant, which recall is able to use the elements' weights, may not be adapted to the ordinal MSP, since in this model it can not use any weight information. In other words, the existence of an α utility-competitive for a matroid (or matroid class) does not imply the existence of an α-competitive algorithm for any of the other three measures. It is worth noting that the intersection-competitive

measure is incomparable with the other measures. It is not hard to find fixed families of instances for which a given algorithm is almost 1 intersection-competitive but has unbounded utility/ordinal/probability-competitiveness. There are also examples achieving almost 1 ordinal-competitiveness but unbounded intersection-competitive ratio.

Chapter 7

Protect to be competitive

In this chapter we introduce and describe a technique we call *forbidden sets*, that allows us to analyze algorithms for the ordinal MSP. Thanks to this technique we devise algorithms for many matroid classes previously studied in the context of MSP and to other matroids that have not been studied in this context. Furthermore, our results improve upon the best known competitive ratios for almost all studied matroid classes, and we study them in details in Chapter 8. Recall that $|R| = n$, and for $t \in [n]$, r_t is the random element arriving at time t, and R_t is the random set of elements arriving at or before time t.

Definition 35. An algorithm has *forbidden sets of size* k if it has the following properties.

1. **(Correctness)** The algorithm returns an independent set ALG.
2. **(Sampling property)** It chooses a sample size s at random from $\mathrm{Bin}(n, p)$ for some fixed *sampling probability* p, and it does not accept any element from the first s arriving ones.
3. **(k-forbidden property)** For every triple (X, Y, r^*) with $Y \subseteq R$, $r^* \in \mathrm{OPT}(Y)$ and $X \subseteq Y - r^*$, one can define a set $\mathcal{F}(X, Y, r^*) \subseteq X$ of at most k *forbidden elements* of X such that the following condition holds. Let $t \geq s + 1$ be a fixed time. If $r_t \in \mathrm{OPT}(R_t)$ and for every $j \in \{s + 1, \ldots, t - 1\}$, $r_j \notin \mathcal{F}(R_j, R_t, r_t)$ then r_t is selected by the algorithm.

To better understand the k-forbidden property suppose that a fixed element $r^* \in \mathrm{OPT}$ arrives at step $t \geq s + 1$, that is $r_t = r^*$. Note that the set R_{t-1} of elements arriving before r_t is a random subset of size $t - 1$ of $R - r_t$, and no matter the choice of R_{t-1}, r^* is always part of $\mathrm{OPT}(R_t) = \mathrm{OPT}(R_{t-1} + r_t)$. Inductively, for $j = t - 1$ down to $j = 1$, once R_j is specified, r_j is a uniform random element of R_j, and R_{j-1} is defined as $R_j - r_j$. Moreover, this choice is *independent* of the previous random experiments (i.e., the choices of $\{r_{j+1}, \ldots, r_{t-1}, R_{t-1}\}$). The k-forbidden property (3.) says that if for every $j \in \{s + 1, \ldots, t - 1\}$ element r_j is not a forbidden element in $\mathcal{F}(R_j, R_t, r_t)$ then r^* is guaranteed to be selected by the algorithm. Designing algorithms with small forbidden sets is the key to achieve constant competitiveness as our key Lemma (that we restate below) shows.

Lemma 36 (Key Lemma)**.** *By setting the right sampling probability $p = p(k)$, every algorithm with forbidden sets of size k is $\alpha(k)$ probability-competitive, where*

$$(p(k),\ \alpha(k)) = \begin{cases} (1/e,\ e) & \text{if } k = 1, \\ (k^{-\frac{1}{k-1}},\ k^{\frac{k}{k-1}}) & \text{if } k \geq 2. \end{cases}$$

Proof. Fix an element r^* from OPT and condition on the realization of $s \sim \mathrm{Bin}(n, p)$, on the time t on which $r^* = r_t$ arrives and on the set $Y = R_t \ni r_t$ of the first t elements arriving. Abbreviate $\mathbb{P}_t(\cdot) = \Pr(\cdot | r_t = r^*, R_t = Y, s)$. By Lemma 32, $r^* \in \mathrm{OPT}(R_t)$ and by the k-forbidden property,

$$\begin{aligned} \mathbb{P}_t(r^* \in \mathrm{ALG}) &\geq \mathbb{P}_t\left(\text{For all } j \in \{s+1, \ldots, t-1\},\ r_j \in R_j \setminus \mathcal{F}(R_j, Y, r^*)\right) \\ &= \prod_{j=s+1}^{t-1} \Pr(r_j \in R_j \setminus \mathcal{F}(R_j, Y, r^*)) \geq \prod_{j=s+1}^{t-1} \left(\frac{j-k}{j}\right)_+, \end{aligned}$$

where $x_+ = \max\{0, x\}$. The equality above holds because of the independence of the random experiments defining iteratively r_{t-1}, r_{t-2}, down to r_{s+1} as mentioned before the statement of this lemma. By removing the initial conditioning we get

$$\Pr(r^* \in \mathrm{ALG}) \geq \mathbb{E}_{s \sim \mathrm{Bin}(n,p)} \frac{1}{n} \sum_{t=s+1}^{n} \prod_{j=s+1}^{t-1} \left(1 - \frac{k}{j}\right)_+. \tag{7.1}$$

To compute the right hand side we use the following auxiliary process. Suppose that n people participate in a game. Each player x arrives at a time $\tau(x)$ chosen uniformly at random from the interval $[0, 1]$. Each person arriving after time p selects a subset of k *partners* from the set of people arriving before them, without knowing their actual arrival times (if less than k people have arrived before her, then all of them are chosen as partners). A player wins if she arrives after time p and every one of her partners arrived before time p. Since the arrival times are equally distributed and the event that two people arrive at the same time has zero probability the arrival order is uniform among all possible permutations. Furthermore, the number of people arriving before time p distributes as $\mathrm{Bin}(n, p)$. Using these facts, the probability that a given person x wins is exactly the right hand side of (7.1). But we can also compute this probability using its arrival time $\tau(x)$ as

$$\int_p^1 \Pr(\text{ all partners of } x \text{ arrived before time } p \mid \tau(x) = \tau)\ d\tau \geq \int_p^1 (p/\tau)^k d\tau,$$

which holds since the arrival time of each partner of x is a uniform random variable in $[0, \tau]$, and conditioned on τ, each partner arrives before time p with probability p/τ. Since x may have less than k partners, we don't necessarily have equality. We conclude that for every $r^* \in \mathrm{OPT}$,

$$\Pr(r^* \in \mathrm{ALG}) \geq \int_p^1 (p/\tau)^k d\tau = \begin{cases} -p \ln(p), & \text{if } k = 1, \\ \dfrac{p - p^k}{k-1}, & \text{if } k \geq 2. \end{cases}$$

By optimizing the value of p as a function of k, we obtain that the probability of $r^* \in \mathrm{ALG}$ is $1/\alpha(k)$, with $\alpha(k)$ as in the statement of the lemma and the probability achieving it is $p = p(k)$. □

The idea of analyzing an algorithm as a series of stochastically independent experiments which defines the reverse arrival sequence appears very early in the history of the secretary problem. One can prove that the algorithm for the classical secretary problem, that samples $s \sim \mathrm{Bin}(n, p(1))$ elements and then selects the first element better than all the sampled ones, is e probability-competitive. In fact, it has forbidden sets of size 1. In our notation, for each (X, Y, r^*) with r^* the maximum element of Y and $X \subseteq Y - r^*$, define the forbidden set $\mathcal{F}(X, Y, r^*)$ as the singleton $\mathrm{OPT}(X)$. The 1-forbidden condition states that if the element r_t arriving at time t is a record, that is, $r_t \in \mathrm{OPT}(R_t)$, and if for every time $j \in \{s+1, \ldots, t-1\}$, $r_j \notin \mathrm{OPT}(R_j)$ (i.e., the j-th arriving element is not a record), then r_t will be chosen by the algorithm. Since all forbidden sets have size at most 1, setting the sampling probability to be $p(1) = 1/e$ guarantees the probability-competitive ratio of $\alpha(1) = e$.

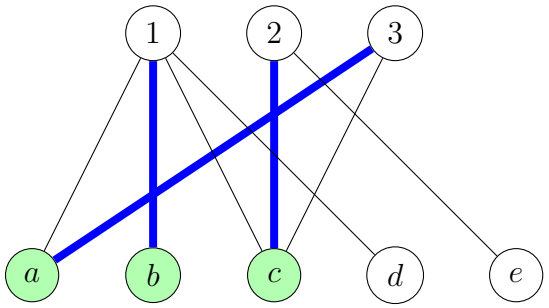


Figure 7.1: Given a bipartite graph, in the transversal matroid a set of nodes on the bottom side is independent if there exists a matching covering them. The thick edges $\{\{a, 3\}, \{b, 1\}, \{c, 2\}\}$ form a matching providing a witness for $\{a, b, c\}$.

Independence witness. It turns out that for many matroid classes there is a combinatorial object that acts as certificate for testing independence. That is, a set X is independent on a given matroid if and only if each element $r \in X$ can be mapped to an object (e.g., an edge covering r, a path ending in r, a subgraph covering r, etc.) $\tilde{r}$ such that the set $\tilde{X} = \{\tilde{r} \colon r \in X\}$ satisfies a combinatorial property (e.g., a matching covering X, a collection of edge/node disjoint paths connecting X with some source, a collection of disjoint subgraphs covering X). We call $\tilde{X}$ a *witness* for the independence of X. A set X may have multiple witnesses, but we will always assume that there is a *canonical witness*, $\mathrm{witness}(X)$, that can be computed by the algorithm and furthermore, the choice of the witness cannot depend on the set of elements seen so far nor on its arrival order. The fact that the witnesses do not depend on the arrival order makes them amenable to the analysis by the reverse arrival sequence analysis above, which in turn, will help us to devise algorithms with constant-size forbidden sets.

Chapter 8

Matroids with small forbidden sets

In this chapter we show how to apply the forbidden sets technique to devise algorithms for different matroid families. The goal is to find algorithms having *small* forbidden sets. In particular, if an algorithm has constant size forbidden sets then this immediately implies that it is $O(1)$ probability-competitive. We show this in many cases, such as *transversal, graphic and laminar* matroids. Other matroid families are considered, and we obtain probability-competitive algorithms for all of them, beating the best known ratios even for the weaker utility variant.

8.1 Transversal matroids and Gammoids

Let $G = (L \cup R, F)$ be a bipartite graph with independent set L and R, where the elements of R are called the *terminals* of G. The transversal matroid $\mathcal{T}[G]$ associated to G is the matroid with ground set R whose independent sets are those $X \subseteq R$ that can be covered by a matching in G. We call G the *transversal presentation* of $\mathcal{T}[G]$.

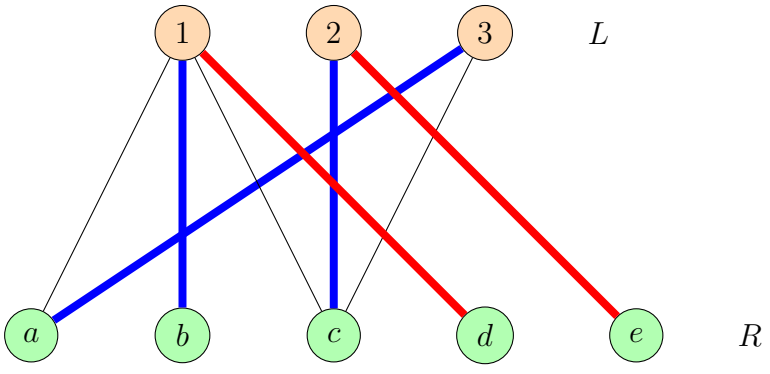


Figure 8.1: Presentation of a transversal matroid with terminals $R = \{a, b, c, d, e\}$. The sets $\{a, b, c\}$ and $\{a, d, e\}$ are bases.

Let $G = (V, E)$ be a digraph and two subsets $S, R \subseteq V$ called *sources* and *terminals* respectively, not necessarily disjoint. The *gammoid*, $\Gamma(G, S, R)$, is the matroid over

R, where $X \subseteq R$ is independent if X is *linked* to S, that is, if there are node-disjoint directed paths from S and ending on each element of X. We say that (G, S, R) is the gammoid presentation of the matroid. Note that transversal matroids can be seen as gammoids by declaring all the non-terminals as sources and directing the arcs in the transversal presentation from sources to terminals.

In the *Transversal MSP*, a transversal presentation G for an unknown ordered matroid $\mathcal{M} = \mathcal{T}[G]$ is either revealed at the beginning of the process, or it is revealed online in the following way. Initially, the algorithm only knows the number of terminals. The terminals in R arrive in random order and whenever $r \in R$ arrives, it reveals its ordinal value information and the set of its neighbors in G, which is a subset of L. In the *gammoid MSP*, a gammoid presentation (G, S, R) for an unknown gammoid is either revealed at the beginning or it is revealed online as elements from R arrive: When a terminal $r \in R$ arrives, all possible S-r paths are revealed. At that time, the algorithm has access to the subgraph $G_t \subseteq G$ only containing the arcs belonging to every possible S-R_t path and can test whether a vertex is in S or not.

Exchangeability parameter for gammoids. Let us define a parameter to control the competitiveness of our algorithm for the gammoid MSP. Let X be an independent set and let Q be a path linking a terminal $r \in R \setminus X$ outside X to S. The *exchangeability* μ of the presentation (G, S, R) is the maximum number of paths that Q intersects in a collection of S-X paths. The intuition behind is the following: in order to include Q into the collection of paths while keeping disjointness we have to remove or exchange at least μ paths from the collection. For instance, if we define the *diameter* d of the gammoid presentation as the maximum number of nodes in any source-terminal path, then μ is at most d. If furthermore the terminals are sinks, that is out-degree 0, then μ is at most $d - 1$, since paths ending at different terminals cannot intersect on a terminal.

Remark 37. *This is the case for transversal matroids: their diameter in the gammoid presentation is* 2 *and their exchangeability is 1. For our results, we assume the algorithm also knows an upper bound μ for the exchangeability parameter, and in this case we call the problem μ-gammoid MSP or bounded exchangeability gammoid MSP.*

Most of the known algorithms for transversal MSP [11, 32, 68] work with ordinal information. The best algorithm so far, by Kesselheim et al. [64] achieves an asymptotically optimal utility-competitive ratio of $e + O(1/n)$ for the more general (non-matroidal) *vertex-at-a-time bipartite online matching problem*, in which edges incident to the same arriving vertex may have different weights and the objective is to select a matching of maximum total weight. For the specific case of transversal matroid this algorithm can be implemented in the ordinal model, meaning that is $e + O(1/n)$ ordinal-competitive. Interestingly, for the broader bipartite matchings case, Kesselheim's algorithm does not work in the ordinal model, but the previous algorithm by Korula and Pál [68] does, achieving 8 ordinal-competitiveness. A recent

result by Hoefer and Kodric [52] improves this factor to $2e$ ordinal-competiive for bipartite matchings.

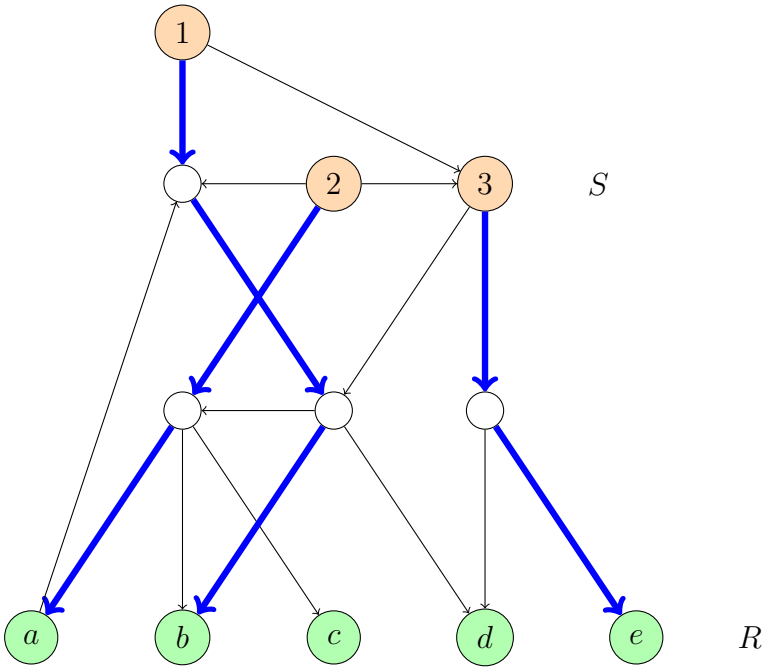


Figure 8.2: Presentation of a gammoid with terminals $R = \{a, b, c, d, e\}$ and sources $S = \{1, 2, 3\}$. The sets $\{a, b, e\}$ and $\{a, b, d\}$ are bases. The diameter of this gammoid presentation is 4, then the exchangeability μ is at most 3.

The μ-gammoid MSP problem is a special case of the (non-matroidal) *hypergraph vertex-at-a-time matching* (HVM) with edges of size at most $\mu + 1$ studied by Korula and Pál [68] and later by Kesselheim et al.'s [64] online hypermatching problem (see the discussion in those papers for precise definitions). They achieve $O(\mu^2)$ and $e\mu$ utility-competitiveness respectively. Below we propose an algorithm for the μ-gammoid MSP that has forbidden sets of size μ. Provided we know μ upfront we get an $\alpha(\mu)$ probability-competitive algorithm. Note that for $\mu \geq 2$, $\alpha(\mu) = \mu^{1+1/(\mu-1)} < e\mu$, so our guarantee strictly improves on that of previous algorithms for HVM and hypermatching, on the special case of μ-gammoids.

Our algorithms. We use the convention that for every vertex v covered by some matching M, $M(v)$ denotes the vertex matched with v in M. Furthermore, for every independent set $X \subseteq R$, we select canonically a witness matching $M_X := \text{witness}(X)$ that covers X. In the case of gammoids, for any set $\mathcal{P}$ of node-disjoint paths linking some set X to S, and for every $v \in X$, $\mathcal{P}(v)$ denotes the unique path in $\mathcal{P}$ linking v to S. We also say that $\mathcal{P}$ covers a vertex u if u is in the union of the vertices of all paths in $\mathcal{P}$. Furthermore, for every independent set $X \subseteq R$, we canonically select a fixed collection of node-disjoint S-X directed paths $\mathcal{P}_X := \text{witness}(X)$ linking X to

S, and we assume that this choice does not depend on the entire graph but only on the minimum subgraph containing all arcs in every S-X path. We also recall that on step i, $R_i = \{r_1, \ldots, r_i\}$ denotes the set of revealed terminals and G_i denotes the subgraph of the presentation currently revealed.

Algorithm 3 for transversal matroids.

Input: Presentation of a transversal matroid $\mathcal{T}[G]$ whose terminals arrive in random order.
▷M and ALG are the currently chosen matching and terminal vertices respectively.

ALG $\leftarrow \emptyset$, $s \leftarrow \mathrm{Bin}(n, p)$, $M \leftarrow \emptyset$
for $i = s + 1$ to n **do**
  if $r_i \in \mathrm{OPT}(R_i)$ and $\ell_i := M_{\mathrm{OPT}(R_i)}(r_i)$ is not covered by M **then**
    ALG $\leftarrow$ ALG $+ r_i$, $M \leftarrow M \cup \{\ell_i r_i\}$
Return ALG.

Algorithm 4 for μ-bounded gammoids.

Input: Presentation of a gammoid $\Gamma := \Gamma(G, S, R)$ whose terminals arrive in random order.
▷$\mathcal{P}$ and ALG are the currently chosen collection of node-disjoint paths and terminals selected respectively.

ALG $\leftarrow \emptyset$, $s \leftarrow \mathrm{Bin}(n, p)$, $\mathcal{P} \leftarrow \emptyset$.
for $i = s + 1$ to n **do**
  if $r_i \in \mathrm{OPT}(R_i)$ and no vertex in the path $\mathcal{P}_{\mathrm{OPT}(R_i)}(r_i)$ is covered by $\mathcal{P}$ **then**
    ALG $\leftarrow$ ALG $+ r_i$, $\mathcal{P} \leftarrow \mathcal{P} \cup \{\mathcal{P}_{\mathrm{OPT}(R_i)}(r_i)\}$
Return ALG.

The algorithms above can compute $\mathrm{OPT}(R_i)$ without knowing the terminal weights, by just using the greedy algorithm. This requires that one is able to check independence algorithmically in each case. Indeed, for the transversal MSP algorithm, a set $X \subseteq R_i$ is independent if and only if the maximum cardinality matching on $G_i[N_G(X) \cup X]$ has size $|X|$. In the case of gammoids, one can check if $X \subseteq R_i$ is independent, by a standard reduction to a flow problem on G_i.

Theorem 10. *Algorithm 4 has forbidden sets of size equal to the exchangeability μ of the gammoid presentation. If μ is known, we can set $p = p(\mu)$ to get an $\alpha(\mu) = \mu^{1+1/(\mu-1)}$ probability-competitive algorithm.*

Proof. By construction, the set $\mathcal{P}$ contains node-disjoint paths covering ALG at every time step, hence the algorithm is correct. The sampling condition is also satisfied by design. Let $r^* \in \mathrm{OPT}(Y)$ where Y is a fixed set of terminals of size $t \geq s + 1$, and suppose that $R_t = Y$ and $r_t = r^*$. Note that r_t is selected by the algorithm if all vertices in the path $\mathcal{P}_{\mathrm{OPT}(R_t)}(r_t)$ are not covered by the collection $\mathcal{P}$ prior to that iteration. In other words, by defining the forbidden sets to be

$$\mathcal{F}(X, Y, r^*) = \{v \in \mathrm{OPT}(X) \colon \mathcal{P}_{\mathrm{OPT}(X)}(v) \text{ intersects } \mathcal{P}_{\mathrm{OPT}(Y)}(r^*)\},$$

the element r_t is selected if $r_j \notin \mathcal{F}(R_j, R_t, r_t)$ for all $j \in \{s+1, \ldots, t-1\}$. By definition, each forbidden set has size at most μ. □

Algorithm 3 is essentially Algorithm 4 applied over the gammoid presentation of the transversal matroid $\mathcal{T}[G]$. Together with Remark 37, it follows the result for the transversal MSP.

Theorem 11. *Algorithm 3 has forbidden sets of size 1, and therefore, by choosing $p = 1/e$, it is an (optimal) e probability-competitive for transversal matroids.*

We remark that every constant utility-competitive algorithm so far known for transversal MSP requires to learn a bipartite presentation online. It is an open problem to find constant competitive algorithms for transversal matroids that only access the matroid via an independence oracle.

8.2 Matroidal Graph Packings

Let $\mathcal{H}$ be a finite set of graphs. An *$\mathcal{H}$-packing* of a host graph $G = (V, E)$ is a collection $\mathcal{Q} = \{H_i\}_{i=1\ldots k}$ of node-disjoint subgraphs of G such that each $H \in \mathcal{Q}$ is isomorphic to some graph in $\mathcal{H}$. A vertex in G is said to be covered by $\mathcal{Q}$ if it belongs to some graph of $\mathcal{Q}$. Let $R \subseteq V$ be a set of vertices called terminals and consider the independence system $\mathcal{M}(R, G, \mathcal{H})$ over R whose independent sets are all $X \subseteq R$ for which there is an $\mathcal{H}$-packing covering X in G. We say that $\mathcal{H}$ is *matroidal* if $\mathcal{M}(V(G), G, \mathcal{H})$ defines a matroid for every graph G. Note that in this case, if $\mathcal{M}(R, G, \mathcal{H})$ is the restriction of $\mathcal{M}(V(G), G, \mathcal{H})$ to a subset R, then it is also a matroid. We call every such $\mathcal{M}(R, G, \mathcal{H})$ a graph packing matroid.

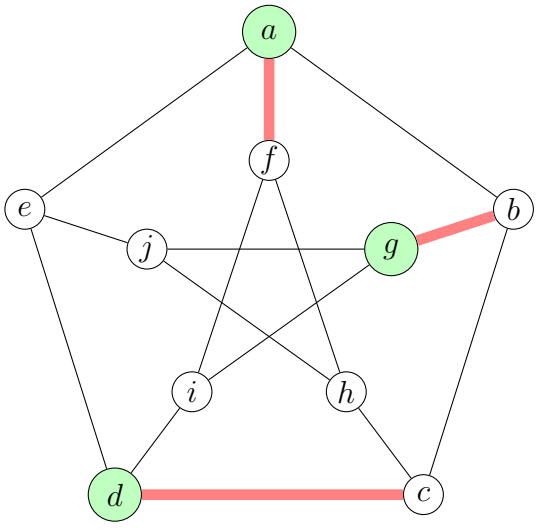


Figure 8.3: The Petersen graph (V, E). In the matching matroid $\mathcal{M}(V, G, \{K_2\})$, the set $\{a, g, d\}$ is independent since it is covered by the matching $\{\{a, g\}, \{g, b\}, \{c, d\}\}$.

Matroidal families. For an extensive treatment see Loebl and Poljak [76], Janata [58] and the references therein. In the following examples, K_n denotes the complete graph on n vertices and S_n denotes the star with n legs. If $\mathcal{H} = \{K_1\}$ then $\mathcal{M}(V, G, \mathcal{H})$ is the free matroid over V where all sets are independent. If $\mathcal{H} = \{K_2\}$ then $\mathcal{M}(V, G, \mathcal{H})$ is the *matching matroid* of G, whose independent sets are all vertex sets that can be covered by a matching. If for some k the family $\mathcal{H} = \{S_1, S_2, \ldots, S_k\}$ is a sequential set of stars, then $\mathcal{M}(V, G, \mathcal{H})$ is matroidal. It is, in fact, the matroid union of many matching matroids. The family $\mathcal{M}(V, G, \mathcal{H})$ is also matroidal if $\mathcal{H} = \{K_2, H\}$ where H is either a factor-critical graph or a 1-propeller.[1] For most known matroidal classes there are polynomial time algorithms available to check independence. This is the case for all classes above. Observe that transversal matroids are instances of matching matroids restricted to one side of the bipartition of the host graph.

As we did for gammoids, we also define an exchangeability parameter μ to control the competitiveness of our algorithm. Consider an $\mathcal{H}$-packing $\mathcal{Q}$ of G, and a subgraph $H \subseteq G$ from the class $\mathcal{H}$ covering a terminal $r \in R$ that is not covered by $\mathcal{Q}$. The exchangeability μ of $\mathcal{M}(R, G, \mathcal{H})$ is the maximum number of terminals over all such $\mathcal{Q}$ and H that would become uncovered if we removed all graphs from $\mathcal{Q}$ that intersect H, namely,

$$\mu := \max \left\{ \sum_{H' \in \mathcal{Q}:\, V(H) \cap V(H') \neq \emptyset} |V(H') \cap R| : H \text{ covers a terminal not covered by } \mathcal{Q} \right\}.$$

This parameter may be complicated to compute but there is a simple upper bound: let h be the maximum number of vertices of a graph from $\mathcal{H}$. Then the worst possible situation occurs when $\mathcal{Q}$ contains only graphs of size h, and H is also a graph of size h intersecting every graph in $\mathcal{Q}$ on exactly one vertex (different from r). In this case, all graphs from $\mathcal{Q}$ must be removed, so the number of newly uncovered vertices becomes $h \cdot (h-1)$. This means that $\mu \leq h(h-1)$.

In the *$\mathcal{H}$-Packing MSP*, the algorithm receives a collection $\mathcal{H}$ of graphs. A host graph $G = (V, E)$ with terminals $R \subseteq V$ is either revealed at the beginning or it is revealed online as elements from R arrive: when a terminal $r \in R$ arrives, all possible edges that belong to a graph $H \subseteq G$ with $r \in V(H)$ with $H \in \mathcal{H}$ (via isomorphism) are revealed. More precisely, let R_t denote the set of terminals revealed up to time t. At that time the algorithm has access to the subgraph $G_t \subseteq G$ induced by all vertices belonging to every possible subgraph $H \subseteq G$, with $H \in \mathcal{H}$ that intersects R_t. The algorithm can also test whether a vertex is a terminal or not. We also assume that an upper bound μ for the exchangeability parameter is available, and in this case we call the problem *bounded $\mathcal{H}$-packing MSP*. Analogously to previous sections, for every independent X we select a canonical packing $\mathcal{Q}_X := \text{witness}(X)$ that does not depend on the arrival order.

In the description of the algorithm we use the convention that for every $\mathcal{H}$-packing $\mathcal{Q}$ of a set X, and for every $v \in V$ covered by $\mathcal{Q}$, $\mathcal{Q}(v)$ denotes the unique graph in $\mathcal{Q}$

[1] A factor-critical graph is one such that $H - x$ admits a perfect matching for all $x \in V(H)$. A 1-propeller is a graph having a leaf r, and a vertex c such that for every $x \in V(H) - c$, $H - x$ admits a perfect matching.

covering v. We also recall that on step i, $R_i = \{r_1, \ldots, r_i\}$ denotes the set of revealed terminals and G_i denotes the subgraph of the presentation currently revealed.

Algorithm 5 for μ bounded $\mathcal{H}$-packing matroids.

Input: A matroidal family $\mathcal{H}$ and a host graph $G = (V, E)$ whose terminals $R \subseteq V$ arrive in random order.

▷ $\mathcal{Q}$ and ALG are the currently chosen $\mathcal{H}$-packing and terminals selected respectively.

ALG $\leftarrow \emptyset$, $s \leftarrow \mathrm{Bin}(n, p)$, $\mathcal{Q} \leftarrow \emptyset$.
for $i = s + 1$ to n **do**
  if $r_i \in \mathrm{OPT}(R_i)$ and r_i is already covered by $\mathcal{Q}$. **then**
    ALG $\leftarrow$ ALG $+ r_i$.
  else if $r_i \in \mathrm{OPT}(R_i)$ and $\mathcal{Q} \cup \{\mathcal{Q}_{\mathrm{OPT}(R_i)}(r_i)\}$ is an $\mathcal{H}$-packing. **then**
    ALG $\leftarrow$ ALG $+ r_i$, $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{\mathcal{Q}_{\mathrm{OPT}(R_i)}(r_i)\}$.
Return ALG.

The algorithm can compute $\mathrm{OPT}(R_i)$ without knowing the terminal weights, by applying the greedy algorithm for $\mathcal{M}(R, G, \mathcal{H})$.

Theorem 12. *Algorithm 5 has forbidden sets of size equal to the exchangeability μ of the $\mathcal{H}$-presentation. If μ is known beforehand, we can set $p = p(\mu)$ to obtain an $\alpha(\mu) = \mu^{1+1/(\mu-1)}$ probability-competitive algorithm for μ-bounded graph packing matroids.*

Proof. Correctness and the sampling condition for forbidden sets are satisfied by design. Now let $r^* \in \mathrm{OPT}(Y)$ where Y is a fixed set of terminals of size $t \geq s + 1$, and suppose that $R_t = Y$ and $r_t = r^*$. Terminal r_t is selected by the algorithm if either $\mathcal{Q}$ already covers it on arrival, or if all vertices in the graph $\mathcal{Q}_{\mathrm{OPT}(R_t)}(r_t)$ were not covered by graphs in $\mathcal{Q}$ prior to that iteration. In any case by defining as forbidden sets

$$\mathcal{F}(X, Y, r^*) = \{v \in \mathrm{OPT}(X) \colon V(\mathcal{Q}_{\mathrm{OPT}(X)}(v)) \text{ intersects } V(\mathcal{Q}_{\mathrm{OPT}(Y)}(r^*)) \setminus \{r^*\}\},$$

we have that r_t is selected if $r_j \notin \mathcal{F}(R_j, R_t, r_t)$ for all $j \in \{s + 1, \ldots, t - 1\}$. By definition, each forbidden set has size at most μ. □

We remark that the competitiveness achievable by an algorithm heavily depends on the way the matroid is presented. For instance, consider a matching matroid $\mathcal{M}$ with host graph $G = (V, E)$. Note that the exchangeability of this matroid is $\mu \leq 2(2 - 1) = 2$. If the graph is presented online then we can use the previous algorithm to obtain an $\alpha(2) = 4$ probability-competitive algorithm. If on the other hand the graph is presented upfront, then we can use the fact that every matching matroid is transversal [36] to construct a transversal presentation of $\mathcal{M}$ and then apply our e probability-competitive algorithm using that presentation.

8.3 Graphic and Hypergraphic Matroids

The *graphic* matroid $\mathcal{M}[G] = (R, \mathcal{I})$ associated to a graph $G = (V, R)$ is the one whose independent sets are all the subsets of edges $X \subseteq R$ such that (V, X) is a forest. The *hypergraphic* matroid $\mathcal{M}[G]$ associated to a hypergraph $G = (V, R)$, whose edges may be incident to any number of vertices, is the matroid over R whose independent sets $X \subseteq R$ are those, for which one can canonically choose for every $r \in X$ an edge denoted by $\text{edge}(r, X) = u(r)v(r)$ in $K_V = (V, \binom{V}{2})$ with both endpoints in r in such a way that all $\text{edge}(r, X)$, for $r \in X$ are different, and the collection $\text{edge}(X) = \{\text{edge}(r, X) \colon r \in X\}$ is a forest [77]. If all the edges of the hypergraph have size 1 or 2 we are back in the graphic case.

One can check that the hypergraphic matroid $\mathcal{M}[G] = (R, \mathcal{I})$ is the matroid induced from the graphic matroid $\mathcal{M}[K_V]$ via the bipartite graph $(R \cup \binom{V}{2}, \tilde{E})$ with $ef \in \tilde{E}$ if $f \subseteq e$. In other words, X is independent in the hypergraphic matroid $\mathcal{M}[G]$ if $\text{edge}(X)$ is independent in the graphic matroid $\mathcal{M}[K_V]$. Moreover, if G is already a graph, $\text{edge}(X) = X$ and $\text{edge}(r, X) = r$ for all $r \in X$.

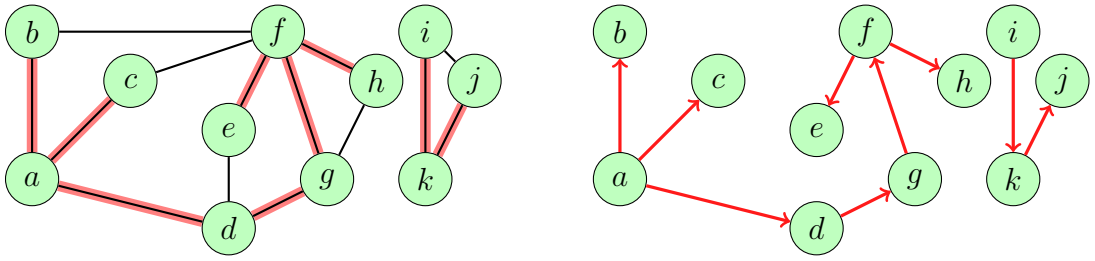


Figure 8.4: Canonical orientation (at the right) of the forest at the left.

In the *graphic MSP*/*hypergraphic MSP* we assume that the underlying graph or hypergraph G of a matroid $\mathcal{M}[G]$ is either revealed at the beginning or revealed online in the natural way: we learn edges as they arrive. Let $X \subseteq R$ be an independent set. By orienting each connected component of the forest $\text{edge}(X)$ from an arbitrary root, we obtain a *canonical orientation* $\text{arc}(X)$ of $\text{edge}(X)$ (for convenience, we denote by $\text{arc}(e, X)$ the oriented version of $\text{edge}(e, X)$) with indegree $\deg^-_{\text{arc}(X)}(v) \leq 1$ for every vertex v. The converse is *almost* true in the following sense. If A is a set of arcs (maybe including loops) such that $\deg^-_A(v) \leq 1$ for every vertex then the underlying graph is not necessarily a forest in K_V, but a pseudoforest: every connected component contains at most 1 cycle, which is directed. In fact, the edge sets of pseudoforest of a given graph J are exactly the independent set of the so called *bicircular matroid* of J. This matroid is transversal with presentation H, where $V(H) = V(J) \cup E(J)$ and $ve \in E(H)$ if and only if e is incident to v. This is the starting point for our algorithm for graphic matroids.

The algorithm. The plan is to only consider edges that belong to the current optimum. Furthermore, if we select an edge, then we orient it and include it into an arc set A with the property that each vertex has maximum in-degree 1. Instead of using a

random orientation (as in the algorithms by Korula and Pál [68] or Soto [96]), at every step we use the canonical orientation of the current optimum forest. In order to avoid closing a cycle we also impose that an arc (u, v) can not be added to A if $\deg_A^-(u) = 1$ or $\deg_A^-(v) = 1$. The same algorithm works on hypergraphic matroids if we replace each independent set X on the hypergraphic matroid by its associated forest edge(X). We also recall that on step i, $R_i = \{r_1, \ldots, r_i\}$ denotes the set of revealed edges. The algorithm is fully described below.

Algorithm 6 for graphic or hypergraphic matroids.

Input: A hypergraphic matroid $\mathcal{M}[G]$ with underlying hypergraph $G = (V, R)$, whose edges arrive in random order.

▷ ALG and A are the currently selected independent set and the orientation of its associated forest.

1: ALG $\leftarrow \emptyset$, $s \leftarrow \text{Bin}(n, p)$, $A \leftarrow \emptyset$.
2: **for** $i = s + 1$ to n **do**
3: **if** $r_i \in \text{OPT}(R_i)$ **then**
4: Let $a_i = (u_i, v_i)$ be the canonical orientation of edge$(r_i, \text{OPT}(R_i))$.
5: **if** $\deg_A^-(u_i) = 0 = \deg_A^-(v_i)$ **then**
6: ALG $\leftarrow$ ALG $+ r_i$, $A \leftarrow A + a_i$
7: Return ALG.

Theorem 13. *Algorithm 6 has forbidden sets of size 2. By setting $p = p(2) = 1/2$, we get an $\alpha(2) = 4$ probability-competitive algorithm for both graphic and hypergraphic matroids.*

Proof. We first prove that the edge set $\hat{A}$ obtained from A by removing its orientation is acyclic. Suppose by contradiction that at the end of some step i, A contains for the first time a set C such that its unoriented version $\hat{C}$ is an undirected cycle. Since this is the first time a cycle appears, r_i must be selected and $a_i = (u_i, v_i)$ must be contained in C. Since a_i is included in A, we know that after its inclusion, $\deg_A^-(v_i) = \deg_C^-(v_i) = 1$ (before its inclusion, the indegree of v_i was 0) and that $\deg_A^-(u_i) = \deg_C^-(u_i) = 0$. But since $\hat{C}$ is a cycle the outdegree of u_i is $\deg_C^+(u_i) = 2 - \deg_C^-(u_i) = 2$. But then, there must be another vertex x in C with indegree 2. This cannot happen because at every moment the indegree of each vertex is at most 1.

The proof above guarantees correctness of the algorithm: for the graphic case, we have $\hat{A} = \text{ALG}$ and for the hypergraphic case, each edge r_i of ALG is mapped to edge$(r_i, \text{OPT}(R_i)) \in \hat{A}$ which form a forest. In both cases we conclude ALG is independent.

Since the sampling condition is satisfied by design, we only need to prove the 2-forbidden condition. Let $r^* \in \text{OPT}(Y)$ where Y is an arbitrary set of $t \geq s + 1$ edges, and suppose that $R_t = Y$ and $r_t = r^*$. The algorithm would then define an arc $a_t = (u_t, v_t)$ and will proceed to add r_t to ALG provided that no arc $a_j = (u_j, v_j)$ considered before has head v_j equal to u_t or v_t. In other words, by defining

$$\mathcal{F}(X, Y, r^*) = \left\{ f \in \text{OPT}(X) : \begin{array}{c} \text{arc}(f, \text{OPT}(X)) \text{ is not oriented} \\ \text{towards any endpoint of edge}(r^*, \text{OPT}(Y)) \end{array} \right\}$$

then r_t is selected if $r_j \notin \mathcal{F}(R_j, R_t, r_t)$ for all $j \in \{s+1, \ldots, t-1\}$. Moreover, since each arc set $\text{arc}(\text{OPT}(X)) = \{\text{arc}(f, \text{OPT}(X)) \colon f \in \text{OPT}(X)\}$ has maximum indegree 1, there are at most 2 arcs in $\text{arc}(\text{OPT}(X))$ oriented towards an endpoint of $\text{edge}(r^*, \text{OPT}(Y))$. So each forbidden set has size at most 2. □

8.4 Column Sparse Representable Matroids

An interesting way to generalize graphic matroids is via their matrix representation. We say that a matroid $\mathcal{M}$ is *represented* by a $m \times n$ matrix M with coefficients in a field $\mathbb{F}$ if we can bijectively map the elements of its ground set to the columns of M in such a way that the independent sets of $\mathcal{M}$ are in correspondence with the sets of columns that are linearly independent in $\mathbb{F}^m$. Graphic matroids are representable by their adjacency matrix interpreted in $GF(2)$. In fact they can be represented in any field. Matroids that have this property are called *regular*. Note that each graphic matroid is representable by a very sparse matroid: each column has only 2 non-zero elements.

Following [96] we say that a matroid is k *column sparse representable* if it admits a representation whose columns have at most k nonzero entries each. These matroids include many known classes such as graphic matroids ($k = 2$), rigidity matroids [101] on dimension d ($k = 2d$) and more generally matroids arising from rigidity theory from d-uniform hypergraphs. These matroids are called (k, ℓ)-sparse matroids (where $0 \leq \ell \leq kd - 1$), and they are, using our notation, kd column sparse [97]. Interesting cases include generic bar-joint framework rigidity in the plane [71], which are characterized by $(2, 3)$-sparse matroids in dimension $d = 2$ (they are 4 column sparse) and generic body-bar framework rigidity in $\mathbb{R}^d$ which are characterized by $(\binom{d+1}{2}, \binom{d+1}{2})$-sparse matroids [98] (they are $d\binom{d+1}{2}$ column sparse).

	{a,b}	{a,c}	{a,d}	{b,c}	{b,d}	{c,d}
a	1	1	1	0	0	0
b	1	0	0	1	1	0
c	0	1	0	1	0	1
d	0	0	1	0	1	1

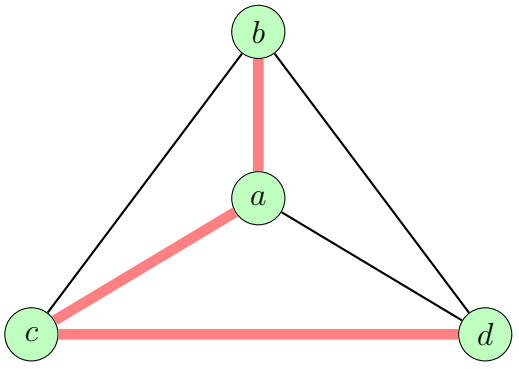


Figure 8.5: Representation of the graphic matroid in the graph K_4. The columns representing the tree $\{\{a, b\}, \{a, c\}, \{c, d\}\}$ are linearly independent.

Let $\mathcal{M}$ be a matroid with k sparse representation M and let X be an independent set of columns. It is easy to show (see e.g., [96]) that we can select one nonzero coordinate from each column in X such that no two selected entries lie on the same

row. In other words, each independent set in $\mathcal{M}$ is also independent in the transversal matroid whose bipartite representations has as color classes the rows and columns of M and where a column i is connected to a row j if entry M_{ij} is nonzero. Even though the converse is not true, we can use the intuition obtained from graphic matroids to extend the algorithm to k column sparse representable matroids with only few changes. Instead of doing that, we are going to further generalize this class of matroids in a different direction.

Multiframed matroids. A matroid $\mathcal{M}$ is called a *frame* matroid [104] if it can be extended to a second matroid $\mathcal{M}'$ (i.e. $\mathcal{M}$ is a restriction of $\mathcal{M}'$) which possesses a *frame* B, that is, a base such that every element of $\mathcal{M}$ is spanned by at most 2 elements of B. For instance, take a graphic matroid $\mathcal{M}[G]$ and consider the graph

$$H = \Big(V(G) + v_0, E(G) \cup \{v_0 v \colon v \in V(G)\}\Big)$$

where v_0 is a new vertex. Then, $\mathcal{M}[H]$ is an extension of $\mathcal{M}[G]$ containing the star centered at v_0 as basis $B := \delta_H(v_0) = \{v_0 v \colon v \in V(G)\}$. Since every edge uv in G is spanned by the set $\{v_0 u, v_0 v\}$ of (at most) 2 elements, we conclude that B is a frame for $\mathcal{M}[G]$. We define a new class of matroids, called *multiframed* or *k-framed* matroids, in a natural way. A matroid $\mathcal{M} = (R, \mathcal{I})$ is a k-framed matroid if it admits an extension $\mathcal{M}^B = (R', \mathcal{I})$ having a k-frame B, i.e., a base such that each element of $\mathcal{M}$ is spanned by at most k elements of B. Without loss of generality we assume that $B \cap R = \emptyset$ (by adding parallel elements) and $R' = B \cup R$. In other words, $\mathcal{M}^B$ is obtained by adjoining the k-frame B to the original matroid $\mathcal{M}$. Observe that if $\mathcal{M}$ is represented by a k column sparse matrix M, then the matrix $[I|M]$ obtained by adjoining an identity (in the field $\mathbb{F}$) represents an extension $\mathcal{M}^B$ of $\mathcal{M}$ where the columns of I form a base B such that each column in M is spanned by at most k elements from B (exactly those elements associated to the k nonzero rows of the column). This means that k-framed matroids generalize k column sparse matroids. This generalization is strict since there are frame matroids that are non-representable.

We define the *k-framed MSP* as the variant of the MSP in which the k-framed matroid $\mathcal{M}$, and its extension $\mathcal{M}^B$ is either fully known beforehand or we simply have access to B and an independence oracle for $\mathcal{M}^B$ (in the case of k column sparse matroid it is enough to receive the columns of the representation in an online fashion).

We need some notation. For every r in the ground set R, we define the set $C(B, r) = \{y \in B \colon B + r - y \text{ is independent}\}$ which is also the minimal subset of B spanning r. It is easy to see that $C(B, r) + r$ is the unique circuit in $B + r$, often called the *fundamental circuit* of r with respect to the base B in $\mathcal{M}^B$. Define also for each $y \in B$, the set $K(B, y) = \{r \in R \colon B + r - y \text{ is independent}\}$. It is easy to see that $K(B, y) + y$ is the unique cocircuit inside $R + y$ in the matroid $\mathcal{M}^B$, often called the *fundamental cocircuit* of y with respect to the base B. Observe that by definition, $r \in K(B, y)$ if and only if $y \in C(B, r)$. Furthermore, by definition of k-framed matroids, $C(B, r)$ has at most k elements. Before presenting the algorithm we need the following result.

Lemma 38. *Let X be an independent set of a k-framed matroid $\mathcal{M}$ with k-frame B. There is a (canonical) injection $\pi_X \colon X \to B$ such that $B + x - \pi_X(x)$ is independent for all $x \in X$.*

Proof. Extend X to a base X' of $\mathcal{M}'$. By the strong basis exchange axiom there is a bijection $\pi \colon X' \to B$ such that $B + x - \pi(x)$ is a base for all $x \in X'$. The restriction of π to X yields the desired injection. □

Algorithm 7 for k-framed matroids.

Input: A k-frame matroid $\mathcal{M}$, with independence oracle access to $\mathcal{M}^B$ and to B. The elements of $\mathcal{M}$ arrive in random order.

▷ ALG is the set currently selected and B' is the set of elements of the frame B that have been marked.

1: ALG $\leftarrow \emptyset$, $s \leftarrow \mathrm{Bin}(n, p)$, $B' \leftarrow \emptyset$.
2: **for** $i = s + 1$ to n **do**
3: **if** $r_i \in \mathrm{OPT}(R_i)$ and $C(B, r_i) \cap B' = \emptyset$ **then**
4: ALG $\leftarrow$ ALG $+ r_i$, $B' \leftarrow B' + \pi_{OPT(R_i)}(r_i)$
5: Return ALG.

Theorem 14. *Algorithm 7 has forbidden sets of size k. By setting $p = p(k)$, we get an $\alpha(k)$ probability-competitive algorithm for k-framed matroids.*

Proof. Suppose that the algorithm is not correct, and let $Z = \{r_{i(1)}, r_{i(2)}, \ldots, r_{i(\ell)}\}$ be a circuit in ALG with $s + 1 \le i(1) \le i(2) \le \cdots \le i(\ell)$. When $r_{i(1)}$ arrived, $y := \pi_{\mathrm{OPT}(R_{i(1)})}(r_{i(1)}) \in C(B, r_{i(1)})$ was marked. Furthermore, our algorithm guarantees that for every element $r_i \in$ ALG with $i > i(1)$, $y \notin C(B, r_i)$. In particular, $y \in C(B, r_{i(j)})$ if and only if $j = 1$, or equivalently, $K(B, y) \cap Z = \{r_{i(1)}\}$. But this implies that the circuit Z and the cocircuit $K := K(B, y) + y$ intersect only in one element, which cannot happen in a matroid. Therefore, the algorithm is correct.

Since the sampling condition is satisfied by design, we only need to prove the k-forbidden condition. Let $r^* \in \mathrm{OPT}(Y)$ where Y is an arbitrary set of $t \ge s + 1$ elements, and suppose that $R_t = Y$ and $r_t = r^*$. The algorithm accepts r^* if and only if no element of $C(B, r_t)$ was marked before. Let $y \in C(B, r_t)$ be an arbitrary element. A sufficient condition for y not being marked at step j is that $\pi_{\mathrm{OPT}(R_j)}(r_j) \neq y$. Therefore, if we define the forbidden sets

$$\mathcal{F}(X, Y, r^*) = \{f \in \mathrm{OPT}(X) \colon \pi_{\mathrm{OPT}(X)}(f) \in C(B, r^*)\}$$

it is clear that r_t is selected if $r_j \notin \mathcal{F}(R_j, R_t, r^*)$ for all $j \in \{s+1, \ldots, t-1\}$. Moreover, since $\pi_{\mathrm{OPT}(X)}$ is injective we conclude that $|\mathcal{F}(X, Y, r^*)| \le |C(B, r^*)| \le k$. □

8.5 Laminar Matroids and Semiplanar Gammoids

In this section we define special classes of gammoids amenable for our techniques. An *arc-capacitated gammoid* (ACG) $\mathcal{M}[N]$ is defined by a directed network $N =$

$(G, \mathbf{o}, R, c)$ where $G = (V, E)$ is a digraph, $\mathbf{o} \in V$ is a single source, $R \subseteq V \setminus \{\mathbf{o}\}$ is a collection of terminals and $c\colon E \to \mathbb{Z}^+$ is a strictly positive integer capacity function on the arcs of G. The ground set of $\mathcal{M}[N]$ is R and every set $J \subseteq R$ is independent if and only if there exist an $\mathbf{o}$-R flow on N satisfying the capacity constraints on the arcs and where each $j \in J$ receives one unit of flow. Without loss of generality we assume that every terminal is reachable from $\mathbf{o}$ via a directed path. It is easy to see that ACGs are equivalent to gammoids without loops, but they are more closely related to transportation or distribution applications.

A *semiplanar gammoid* is an ACG whose digraph G admits a *semiplanar drawing*, which is a planar drawing where all terminals are on the x-axis, the source node is on the positive y-axis, and the rest of the graph (arcs and nodes) are strictly above the x-axis, in a way that they do not touch the infinite ray starting upwards from the source node. It is easy to see that G admits a semiplanar drawing if and only if it is planar and all the terminals and the source are contained in the same face (which could be a cycle, a tree or more generally, a pseudoforest).

Laminar Matroids. An important example of semiplanar gammoids are *laminar matroids*. A collection $\mathcal{L}$ of non-empty subsets of a finite set of terminals R is called laminar if for every $L, H \in \mathcal{L}$, $L \cap H \in \{L, H, \emptyset\}$. The laminar matroid $\mathcal{M}[R, \mathcal{L}, c]$, where $\mathcal{L}$ is a laminar family over R and $c\colon \mathcal{L} \to \mathbb{Z}^+$ is a positive integer capacity function on the sets of $\mathcal{L}$, is the matroid with ground set R and whose independent sets are those $X \subseteq R$ for which the number of elements that X contains in any set of the laminar family does not exceed its capacity, i.e. $|X \cap L| \leq c(L)$. Observe that, since c is positive, every singleton is independent. Furthermore, without loss of generality we assume that $R \in \mathcal{L}$ (with $c(R)$ equal to the rank of the matroid) and that all singletons are in $\mathcal{L}$ (with $c(\{r\}) = 1$ for each $r \in R$). Adding those sets does not destroy laminarity.

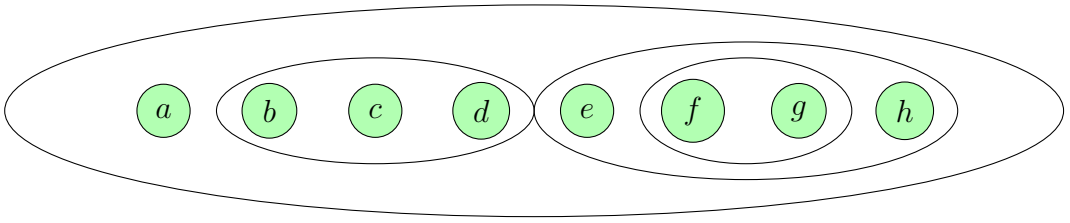


Figure 8.6: Laminar family.

The Hasse diagram of the containment order of $\mathcal{L}$ forms a tree T' where every set in $\mathcal{L}$ is a child of the smallest set in $\mathcal{L}$ that strictly contains it. Note that the leaves of T' are in bijection with the terminals, and that the root (unique maximum) v_R represents R. Consider the directed graph G obtained by adding an auxiliary vertex $\mathbf{o}$ to T' connected to v_R and orienting the tree away from $\mathbf{o}$. If we put $\mathbf{o}$ on the upper semiaxis, draw G with all arcs pointing downwards, and assign capacities to each arc equal to the capacity of the laminar set associated to the arc's head, then we obtain a semiplanar representation of $\mathcal{M}[R, \mathcal{L}, c]$. Furthermore, the terminals in R appear in

the x-axis in the left-to-right order induced by the drawing of the tree.

Semiplanar drawings and neighbours. In what follows we fix a semiplanar drawing G of a semiplanar gammoid $\mathcal{M}[N]$. We identify R with the set $[n]$ by labeling the terminals from left to right as they appear in the x-axis. For convenience in the rest of the presentation, we add to G two auxiliary nodes on the x-axis, 0 and $n+1$, where 0 is located to the left of 1, and $n+1$ is located to the right of n, together with the arcs $\mathbf{o}0$ and $\mathbf{o}(n+1)$. Observe that by our assumptions on the drawing, it is possible to add those arcs without destroying semiplanarity.

For any set $J \subseteq [n]$ and any $y \in J \cup \{0, n+1\}$, we denote by $\mathrm{Left}_J(y)$ the closest element to y in $J \cup \{0\}$ that is located strictly to its left (understanding $\mathrm{Left}_J(0) = 0$). Similarly, we denote by $\mathrm{Right}_J(y)$ the first element in $J \cup \{n+1\}$ located to the right of y (understanding $\mathrm{Right}_J(n+1) = n+1$). For $y \in [n] \setminus J$, we call $\mathrm{Left}_{J+y}(y)$ and $\mathrm{Right}_{J+y}(y)$ its left and right neighbors in J (note that they maybe equal to 0 or $n+1$ and thus are not necessarily elements of J).

Ancestors, tree-order and representatives. In the future we will map each terminal outside an independent set J to one or more of its neighbors in J. For the case in which G is a tree (i.e., for laminar matroids) we want to consistently assign each element to just one of them. We do this as follows. For every pair of nodes x, y in G, let xGy be the unique undirected x-y path in G. We say that x is an *ancestor* of y (and y is a *descendant* of x) if $\mathbf{o}Gy$ contains $\mathbf{o}Gx$; in that case we denote $x \sqsupseteq y$ (and $y \sqsubseteq x$). Note that $(V, \sqsubseteq)$ is a partial order, and in fact, it is a join-semilattice where $x \vee y$ is the lowest common ancestor of x and y. Note that by our drawing choice, for any $i \in [j, j'] \subseteq [0, n+1]$ we have $i \sqsubseteq j \vee j'$. In particular, if $[j, j'] \subseteq [k, k'] \subseteq [0, n+1]$ then $j \vee j' \sqsubseteq k \vee k'$.

For every nonempty set $J \subseteq [n]$, and for every terminal $y \in [n]$, we define its *representantive* $\pi_J(y)$ in J such that

$$\pi_J(y) = \begin{cases} y & \text{if } y \in J, \\ \mathrm{Left}_{J+y}(y) & \text{if } y \in [n] \setminus J \text{ and } y \vee \mathrm{Left}_{J+y}(y) \sqsubset y \vee \mathrm{Right}_{J+y}(y), \\ \mathrm{Right}_{J+y}(y) & \text{if } y \in [n] \setminus J \text{ and } y \vee \mathrm{Left}_{J+y}(y) \sqsupseteq y \vee \mathrm{Right}_{J+y}(y). \end{cases}$$

This element is well defined since for all $y \in [n] \setminus J$, both $y \vee \mathrm{Left}_{J+y}(y)$ and $y \vee \mathrm{Right}_{J+y}(y)$ belong to $\mathbf{o}Gy$ and so one is an ancestor of the other. Observe that $\pi_J(y)$ is never equal to 0 (because then $\mathbf{o} = 0 \vee y \sqsubset \mathrm{Right}_{J+y}(y) \vee y$ which is a contradiction as $\mathbf{o}$ is the root), nor $n+1$ (because then $\mathbf{o} = (n+1) \vee y \sqsubseteq \mathrm{Left}_{J+y}(y) \vee y \sqsubseteq v_R \sqsubset \mathbf{o}$). In particular, $\pi_J(y) \in \{\mathrm{Left}_{J+y}(y), y, \mathrm{Right}_{J+y}(y)\} \cap J$. A graphical way to understand the definition of the representative of an element y outside J is the following: let j and j' be the left and right neighbors of y in J respectively and call $\mathbf{o}Gj$ and $\mathbf{o}Gj'$ the left and right paths respectively. The representative of y is its left neighbor (respectively, its right neighbor) if and only if by starting from y and walking up on G against its orientation, the first path hit is the left path (respectively the right path). In case of a tie, the representative is the right neighbor (see Figure 8.7).

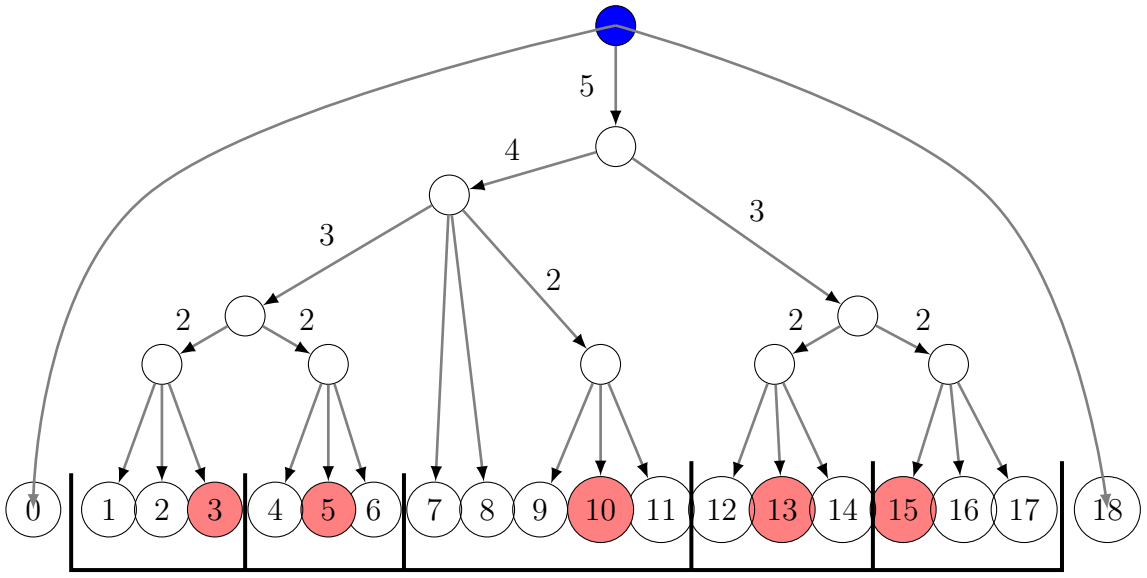


Figure 8.7: Semiplanar drawing of a laminar matroid with ground set [17]. Nonunit arc capacities are shown. Below we show the partition $(\pi_J(j))_{j\in J}$ induced by the independent set $J = \{3, 5, 10, 13, 15\}$.

We claim also that for every $j \in J$, the set of elements *to its right* having j as representative is an interval of the form $[j, k]$ with $k < \text{Right}_J(j)$. Indeed, this is true if $\text{Right}_J(j) = n + 1$. Suppose now that $j' := \text{Right}_J(j) \leq n$, and that the claim does not hold. Then, there must be two consecutive terminals $i, i + 1 \in [j, j']$ with $\pi_J(i + 1) = j$ and $\pi_J(i) = j'$. But then, we get the following contradiction:

$$(i + 1) \vee j' \sqsubseteq i \vee j' \sqsubseteq i \vee j \sqsubseteq (i + 1) \vee j \sqsubset (i + 1) \vee j'$$

where the first inequality holds since $[i+1, j'] \subseteq [i, j']$, the second, by the definition of $\pi_J(i)$, the third since $[j, i] \subseteq [j, i + 1]$ and the fourth by definition of $\pi_J(i + 1)$. Since every element in (j, j') either has j or j' as representative, we conclude, in fact, that the entire set $\pi_J^{-1}(j)$ of elements with j as representative is an interval of terminals enclosing j but strictly contained in $[\text{Left}_J(j), \text{Right}_J(j)]$. In other words $(\pi_J(j))_{j\in J}$ is a partition of $[n]$ into $|J|$ intervals.

The algorithms. In what follows we describe our algorithms for semiplanar gammoids and laminar matroids. Apart from some special case occurring when the sample is empty, our algorithms are extremely simple. Each one computes the optimal solution $\text{OPT}(R_s)$ of the sample and leave their elements *unmarked*. When a terminal r_i that is part of the current optimum arrives, it checks if the (closest neighbors / representative) of r_i in $\text{OPT}(R_s)$ (are / is) unmarked. If so, it marks (them / it) and selects r_i as part of the solution.

Algorithm 8 for semiplanar gammoids.

Input: An semiplanar gammoid with a fixed semiplanar drawing

ALG $\leftarrow \emptyset$, $s \leftarrow \text{Bin}(n, p)$,
if $s = 0$ **then**
  ALG $\leftarrow \{r_1\}$
else $B \leftarrow \emptyset$.
  for $i = s + 1$ to n **do**
    if $r_i \in \text{OPT}(R_i)$, $\text{Left}_{\text{OPT}(R_s)+r_i}(r_i) \notin B$ and $\text{Right}_{\text{OPT}(R_s)+r_i}(r_i) \notin B$ **then**
      ALG $\leftarrow$ ALG $+ r_i$, and
      $B \leftarrow B \cup \{\text{Left}_{\text{OPT}(R_s)+r_i}(r_i), \text{Right}_{\text{OPT}(R_s)+r_i}(r_i)\}$
Return ALG.

Algorithm 9 for laminar matroids.

Input: A laminar matroid with a fixed semiplanar drawing

ALG $\leftarrow \emptyset$, $s \leftarrow \text{Bin}(n, p)$,
if $s = 0$ **then**
  ALG $\leftarrow \{r_1\}$
else $B \leftarrow \emptyset$.
  for $i = s + 1$ to n **do**
    if $r_i \in \text{OPT}(R_i)$ and $\pi_{\text{OPT}(R_s)}(r_i) \notin B$ **then**
      ALG $\leftarrow$ ALG $+ r_i$, and
      $B \leftarrow B \cup \{\pi_{\text{OPT}(R_s)}(r_i)\}$.
Return ALG.

Theorem 15. *Algorithm 8 has forbidden sets of size* 4. *By setting* $p = p(4) = \sqrt[3]{1/4}$ *we get an* $\alpha(4) = 4^{4/3} \approx 6.3496$ *probability-competitive algorithm for semiplanar ACGs.*

The previous result applied to laminar matroids already beat the best guarantees known for that class (9.6 [79] and $3\sqrt{3}e$ [56, 57]). But we can do better for laminar matroids since for those, there is a unique path from the source to each terminal.

Theorem 16. *Algorithm 9 has forbidden sets of size* 3. *By setting* $p = p(3) = \sqrt{1/3}$ *we get an* $\alpha(3) = 3\sqrt{3}$ *probability-competitive algorithm for laminar matroids.*

We observe that this algorithm is very similar to the $3\sqrt{3}e$ competitive algorithm of Jaillet et al. [56, 57]. Note that the partition of the terminals given by $\pi_{\text{OPT}(R_s)}$ induces a unitary partition matroid $\mathcal{P}'$. After the sample, Algorithm 9 simply selects on each part, the first arriving element that is part of the current optimum. It can be shown that the partition matroid $\mathcal{P}'$ we define is the same as the one defined in [57, Section 3.2]. The main algorithmic difference is that in [57], the authors use the algorithm for the classic secretary problem to select one terminal on each part that has constant probability of being the largest element. Instead, we select the first

arriving element on each part that is part of the optimum at the time of arrival. This small change makes the competitive ratio of our algorithm e times smaller than theirs, but the analysis is more involved. To prove Theorems 15 and 16 we need to develop some extra tools.

For every nonempty independent set $J \subseteq [n]$ in the semiplanar gammoid $\mathcal{M}[N]$ we chose an arbitrary but fixed integral $\mathbf{o}$-J flow f_J satisfying the arc capacity constraints and saturating J. Define the unit capacity network $N^1(J) = (G^1(J), \mathbf{o}, R, 1)$, where $G^1(J)$ is obtained from G by splitting each arc uv with $f_J(uv) \geq 2$ into $f_J(uv)$ parallel arcs (we keep all arcs uv with $f_J(uv) \in \{0, 1\}$, we also keep the arcs $\mathbf{o}0$ and $\mathbf{o}(n+1)$). We do this in such a way that $G^1(J)$ is still semiplanar. The matroid $\mathcal{M}[N^1(J)]$ is the unit capacity semiplanar gammoid associated to J.

Observe that every path in $G^1(J)$ corresponds canonically to an unsplitted path in G going through the same nodes. Furthermore, every arc-disjoint collection $\mathcal{P}$ of $\mathbf{o}$-R paths in $G^1(J)$ can be regarded, by unsplitting parallel arcs, as an $\mathbf{o}$-R flow f on G such that each arc $e \in E(G)$, satisfies $f(e) \leq f_J(e) \leq c(e)$. Therefore, we have the following important lemma.

Lemma 39. *Any independent set in $\mathcal{M}[N^1(J)]$ is also independent in the original matroid $\mathcal{M}[N]$.*

We say that two paths P and Q in the drawing of a graph *cross* if P enters Q on the left, shares zero or more arcs with Q and then exits Q on the right, or viceversa. A collection of paths is mutually noncrossing if no pair of them cross.[2] The $\mathbf{o}$-J flow f_J in the network N can be mapped to an $\mathbf{o}$-J flow f_J^1 in $N^1(J)$ in a natural way. By applying flow-decomposition on f_J^1 we get a collection $\{P^k\}_{k \in J}$ of arc-disjoint paths. Define also P^0 and P^{n+1} as the paths consisting of a single arc $\mathbf{o}0$ and $\mathbf{o}(n+1)$ respectively. By a standard planar uncrossing argument, we can assume that all paths in $\{P^k\}_{k \in J \cup \{0, n+1\}}$ are mutually noncrossing (but they can still share internal nodes, in fact all paths P^k with $k \in J$ contain $\mathbf{o}$ and v_R, see Figure 8.8). For $j \in J + 0$, call $j' = \mathrm{Right}_{J+0}(j)$ and define the collection of arcs $A^j \subseteq E(G^1(J))$ as those that are drawn in the closed planar region $\mathcal{R}_j$ bounded by P^j, $P^{j'}$ and the x-axis. Since paths $\{P^k\}_{k \in J \cup \{0, n+1\}}$ form a topological star whose tips are in the x-axis we conclude that $\{\mathcal{R}_k\}_{k \in J+0}$ is a division of the region bounded by P^0, P^{n+1} and the x-axis, and thus $\{A^k\}_{k \in J+0}$ is a covering of all arcs in $G^1(J)$. In fact, if $1 \leq j' \neq n+1$, then every arc of $P^{j'}$ belongs to pieces A^j and $A^{j'}$, while each arc in $G^1(J) \setminus \{P^k\}_{k \in J}$ belongs to a single piece of the covering. Furthermore, the only terminals contained in region $\mathcal{R}_j$ are those in $[j, j']$.

Lemma 40. *Let $y \in [n] \setminus J$, and $j = \mathrm{Left}_{J+y}(y)$, $j' = \mathrm{Right}_{J+y} y$ be its neighbors in J. Then there is a directed $\mathbf{o}$-y path $D_J(y)$ in $G^1(J)$ whose arc set is completely*

[2] An alternative way to understand this definition is to imagine that in the drawing nodes and arcs have positive area (they are circles and thick lines). A drawing of a path is any continuous non-self-intersecting curve drawn *inside* the union of all circles and thick lines associated to the nodes and arcs of P visiting them in the correct order. A collection of paths are mutually noncrossing if we can find drawings such that no pair of them intersects. Note that two noncrossing paths can still share arcs and nodes.

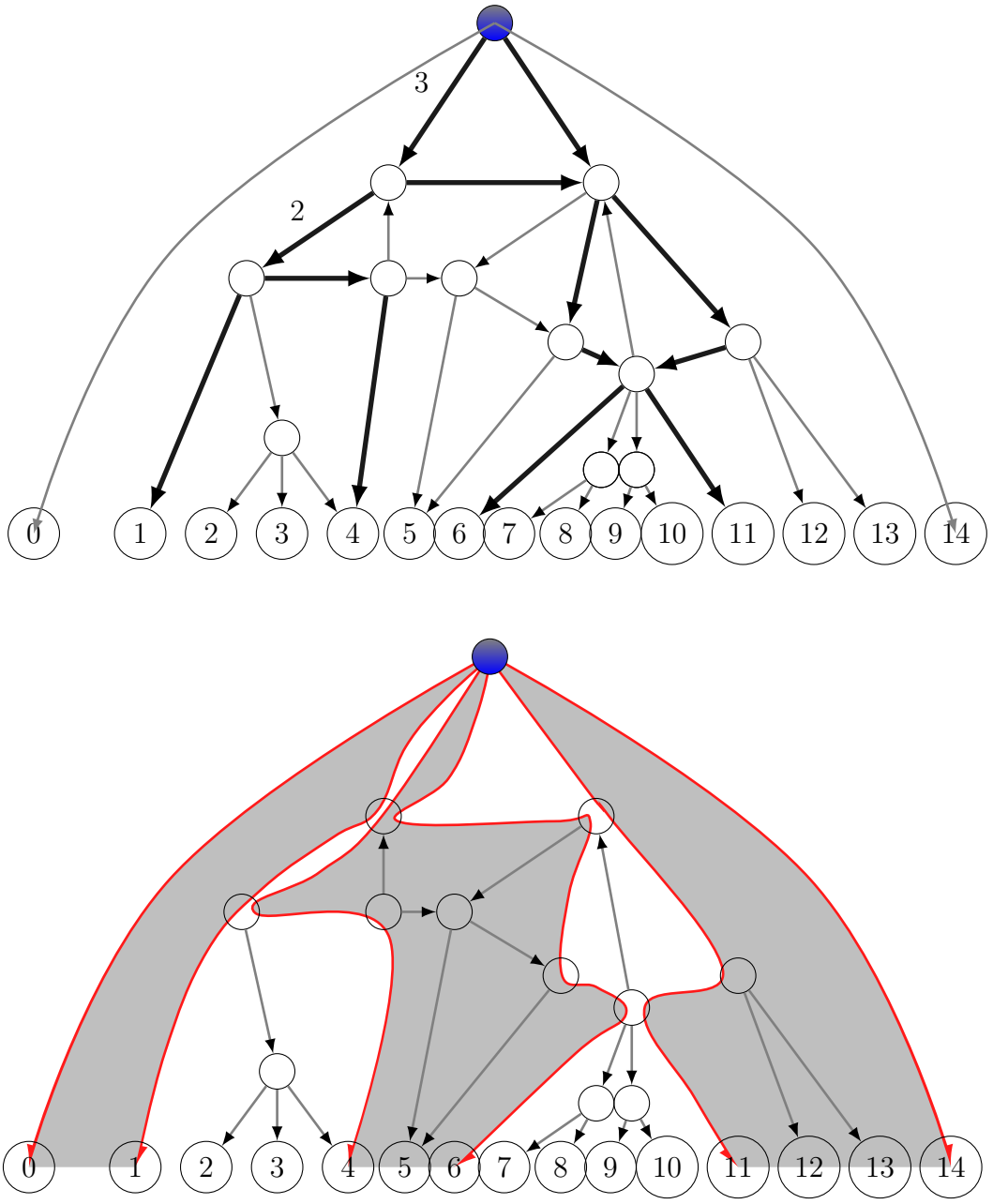


Figure 8.8: On the left, a flow f_J associated to the independent set $J = \{1, 4, 6, 11\}$ on a semiplanar gammoid $\mathcal{M}[N]$ with ground set $R = [13]$ (0 and 14 are auxiliary nodes). On the right, $\mathcal{M}[N^1(J)]$ and the partition of the drawing into regions $\mathcal{R}_0$, $\mathcal{R}_1$, $\mathcal{R}_4$, $\mathcal{R}_6$ and $\mathcal{R}_{11}$ induced by the paths $\{P^j\}_{j \in J \cup \{0,14\}}$. The regions $\mathcal{R}_0$, $\mathcal{R}_4$ and $\mathcal{R}_{11}$ are shaded.

contained in A^j. Furthermore, $D_J(y)$ can be chosen so that it arcs-intersects at most one path $\bar{P}$ in $(P^k)_{k\in J}$. In the semiplanar case, $\bar{P}$ must be one of P^j and $P^{j'}$, and in the laminar case, $\bar{P} = P^{\pi_J(\ell)}$.

Proof. Note that $y \in [j, j']$ is in the planar region $\mathcal{R}_j$. Since y is reachable from $\mathbf{o}$ in G, there is a path Q from $\mathbf{o}$ to y in $G^1(J)$. Let v be the last vertex in Q contained in the vertices of P^j and $P^{j'}$ (v maybe in one or both paths), say v is in $\bar{P} \in \{P^j, P^{j'}\}$. By concatenating the initial piece of $\bar{P}$ from $\mathbf{o}$ to v and the final piece of Q from v to y we get a path $D_J(y)$ that is completely contained in A^j and that intersects the arcs of at most one path in $\{P_j, P_{j'}\}$.

Consider the same situation in the laminar case. All arcs in P^j, $P^{j'}$ and Q are splitted versions of arcs in $\mathbf{o}Gj$, $\mathbf{o}Gj'$ and $\mathbf{o}Gy$ respectively. The vertex v defined above satisfies $v = y \vee j \vee j'$. If $y \vee j \sqsubset y \vee j'$ then $v = y \vee j$, $\pi_J(y) = j$ and we can construct $D_J(y)$ avoiding all arcs in $P^{j'}$ by selecting $\bar{P} = P^j$ in the previous argument. Analogously, if $y \vee j' \sqsubseteq y \vee j$ then $v = y \vee j'$, $\pi_J(y) = j'$, and we can choose $D_J(\ell)$ to avoid P^j by setting $\bar{P} = P^{j'}$. □

Lemma 41. *Let $J \subseteq [n]$ be a nonempty independent set in $\mathcal{M}[N]$ and $I \subseteq [n] \setminus J$. Consider the following properties.*

(P1) For every $x, y \in I$ with $x \neq y$,

$$\{\mathrm{Left}_{J+x}(x), \mathrm{Right}_{J+x}(x)\} \cap \{\mathrm{Left}_{J+y}(y), \mathrm{Right}_{J+y}(y)\} = \emptyset.$$

(P2) $\mathcal{M}[N]$ is laminar and for every $x, y \in I$, $\pi_J(x) \neq \pi_J(y)$.

If either (P1) or (P2) holds then I is independent in $\mathcal{M}[N^1(J)]$

Proof. Suppose that $(P1)$ is satisfied. Then for every pair of distinct $x, y \in I$, the paths $D_J(x)$ and $D_J(y)$ constructed in Lemma 40 must lie on non-consecutive (and hence disjoint) regions in $\{\mathcal{R}_k\}_{k\in J+0}$. Therefore the paths $\{D_J(z)\}_{z\in I}$ are mutually arc-disjoint from which we conclude that I is an independent set in $\mathcal{M}[N^1(J)]$.

Suppose now that $(P2)$ is satisfied. Let $x < y$ be distinct terminals in I. We claim that the paths $D_J(x)$ and $D_J(y)$ are arc disjoint. If x and y belong to different regions then the only possibility for them to share an arc is that x is in $\mathcal{R}_j$, $y \in \mathcal{R}_{j'}$ with $j' = \mathrm{Right}_J(j)$, and both share arcs in $P^{j'}$. But then, by Lemma 40, $\pi_J(x) = \pi_J(y) = j'$ which is a contradiction.

If x and y are in the same region $\mathcal{R}_j$, then $x, y \in [j, j']$ with $j = \mathrm{Left}_{J+x}(x) = \mathrm{Left}_{J+y}(y)$ and $j' = \mathrm{Right}_{J+x}(x) = \mathrm{Right}_{J+y}(y) \in J + (n+1)$. Since the function π_J partitions $[n]$ into intervals and $x < y$, we must have $\pi_J(x) = j$, $\pi_J(y) = j'$. Suppose now that $D_J(x)$ and $D_J(y)$ had an arc a in common and let w be its head. Since $D_J(x)$ and $D_J(y)$ are split versions of $\mathbf{o}Gx$ and $\mathbf{o}Gy$ we get that $w \sqsupseteq x \vee y$. Since w and $x \vee j$ are both in $\mathbf{o}Gx$, one is an ancestor of the other. Note that w cannot be an ancestor of $x \vee j$ since above the latter $D_J(x)$ coincides with P^j which is arc-disjoint from $D_J(y)$ by Lemma 40 (and a is a common arc). It follows that $x \vee j \sqsupset w \sqsupseteq x \vee y \sqsupseteq y$, and thus, $x \vee j \sqsupseteq y \vee j$. But since $\pi_J(y) = j'$ we have $y \vee j \sqsupseteq y \vee j'$ and we conclude that

$x \vee j \sqsupseteq y \vee j' \sqsupseteq j'$. From the last expression we get $x \vee j \sqsupseteq x \vee j'$ which contradicts the fact that $\pi_J(x) = j$.

We have thus shown that all paths $(D_J(z))_{z \in I}$ are arc-disjoint, thence I is independent in $\mathcal{M}[N^1(J)]$. □

Now we are ready to define the forbidden sets of sizes at most 4 and 3 respectively for Algorithms 8 and 9. For (X, r^*) with $r^* \in [n]$, $X \subseteq [n] \setminus \{r^*\}$ define the set $\mathcal{F}_4(X, r^*)$ given by

$$\{\text{Left}_X(\text{Left}_{X+r^*}(r^*)), \text{Left}_{X+r^*}(r^*), \text{Right}_{X+r^*}(r^*), \text{Right}_X(\text{Right}_{X+r^*}(r^*))\},$$

and $I_4(X, r^*) = [\text{Left}_X(\text{Left}_{X+r^*}(r^*)), \text{Right}_X(\text{Right}_{X+r^*}(r^*))]$. The set $\mathcal{F}_4(X, r^*)$ contains the 2 closest terminals to the left of r^* in X and the 2 closest terminals to its right. $I_4(X, r^*)$ is the enclosing interval containing $\mathcal{F}_4(X, r^*)$. Similarly, for the case of laminar matroids, define

$$\begin{aligned} \mathcal{F}_3(X, r^*) &= \{\text{Left}_X(\pi_X(r^*)), \pi_X(r^*), \text{Right}_X(\pi_X(r^*))\} \\ I_3(X, r^*) &= [\text{Left}_X(\pi_X(r^*)), \text{Right}_X(\pi_X(r^*))]. \end{aligned}$$

where $\mathcal{F}_3(X, r^*)$ consists of the representative of r^* in X and its two neighbors, and $I_3(X, r^*)$ is its enclosing interval. We need one last technical lemma to analize the algorithms performance.

Lemma 42. *Let X and r^* as above, and let $x \in X$.*

(a) If $x \in \mathcal{F}_4(X, r^*)$ then $I_4(X, r^*) \subseteq I_4(X - x, r^*)$.

(b) If $x \notin \mathcal{F}_4(X, r^*)$ then $\mathcal{F}_4(X, r^*) = \mathcal{F}_4(X - x, r^*)$ and $I_4(X, r^*) = I_4(X - x, r^*)$.

If the matroid is laminar, the following properties also hold

(c) If $x \in \mathcal{F}_3(X, r^*)$ then $I_3(X, r^*) \subseteq I_3(X - x, r^*)$.

(d) If $x \notin \mathcal{F}_3(X, r^*)$ then $\mathcal{F}_3(X, r^*) = \mathcal{F}_3(X - x, r^*)$ and $I_3(X, r^*) = I_3(X - x, r^*)$.

Proof of Lemma 42.
Part (a): Let $\mathcal{F}_4(X, r^*) = \{a, b, c, d\}$ from left to right (note that near the borders we could have $a = b$ or $c = d$), in particular $a \leq b \leq r^* \leq c \leq d$. Denote by $a^- = \text{Left}_X(a)$ and $d^+ = \text{Right}_X(d)$.

(i) If $x \in \{a, b\}$ then $\mathcal{F}_4(X - x, r^*) = (\mathcal{F}_4(X, r^*) - x) + a^-$ and so $I_4(X, r^*) = [a, d] \subseteq [a^-, d] = I_4(X - x, r^*)$.

(ii) If $x \in \{c, d\}$ then $\mathcal{F}_4(X - x, r^*) = (\mathcal{F}_4(X, r^*) - x) + d^+$ and so $I_4(X, r^*) = [a, d] \subseteq [a, d^+] = I_4(X - x, r^*)$.

Part (b): Direct.

For parts (c) and (d) let $\mathcal{F}_3(X, r^*) = \{a, b, c\}$ from left to right (in particular $b = \pi_X(r^*)$, note that we could be in the cases $a = b = 0$ or $b = c = n + 1$). Denote by $a^- = \text{Left}_X(a)$ and $c^+ = \text{Right}_X(c)$.

Part (c): We have many possibilities to analyze.

(i) If $x = b$, then the closest neighbors of r^* in $X - x$ are a and c. In particular $\pi_{X-x}(r^*)$ is either a or c. In both cases, $I_3(X, r^*) = [a, c] \subseteq I_3(X - x, r^*)$.

(ii) If $x = a \neq b$ and $r^* \in [b, c]$, then b and c are still the closest neighbors of r^* in $X - x$, and in particular, the representative in $X - x$ is the same as that in X, i.e., $\pi_{X-x}(r^*) = b$. Note that since we removed a, $\text{Left}_{X-x}(b) = a^-$ and so, $I_3(X, r^*) = [a, c] \subseteq [a^-, c] = I_3(X - x, r^*)$

(iii) The case $x = c \neq b$ and $r^* \in [a, b]$ is analogous to the previous one

(iv) If $x = a \neq b$, and $r^* \in [a, b]$, then the closest neighbors of r^* in $X - x$ are a^- and b. We have that $r^* \vee b \sqsubseteq r^* \vee a \sqsubseteq r^* \vee a^-$ where the first inequality holds since $\pi_X(r^*) = b$ and the second one since $[a, r] \subseteq [a^-, r^*]$. We conclude that $\pi_{X-x}(r^*) = b$. In particular $I_3(X, r^*) = [a, c] \subseteq [a^-, c] = I_3(X - x, r^*)$.

(v) If $x = c \neq b$ and $r^* \in [b, c]$ then the closest neighbors of r^* in $X - x$ are b and c^+. We have that $r^* \vee b \sqsubset r^* \vee c \sqsubseteq r^* \vee c^+$ where the first inequality holds since $\pi_X(r^*) = b$ and the second holds since $[r^*, c] \subseteq [r^*, c^+]$. We conclude that $\pi_{X-x}(r^*) = b$. In particular $I_3(X, r^*) = [a, c] \subseteq [a, c^+] = I_3(X - x, r^*)$.

Part (d): Since $x \notin \mathcal{F}_3(X, r^*)$ the neighbors of r^* in $X - x$ are the same as those in X, it follows that $\pi_{X-x}(r^*) = \pi_X(r^*) = b$, $\text{Left}_{X-x}(b) = a$, $\text{Right}_{X-x}(b) = c$. Therefore $\mathcal{F}_3(X - x, r^*) = \mathcal{F}_3(X, r^*)$ and $I_3(X - x, r^*) = I_3(X, r^*)$. □

Now we are ready to prove the guarantees for Algorithms 8 and 9.

Proofs of Theorems 15 and 16. For both algorithms, if $s = 0$ then $|\text{ALG}| = 1$ and so it is independent. In the following assume that $s \geq 1$. Line 6 in the algorithms guarantees that the set ALG satisfies the conditions of Lemma 41, hence ALG is independent in $\mathcal{M}[N^1(J)]$, and by Lemma 39, ALG is independent in the original matroid. This proves correctness. Since the sampling condition holds by construction, we only need to check the forbidden property. For the rest of the proof, define $\mathcal{F}(X, r^*) = \mathcal{F}_i(X, r^*)$, $I(X, r^*) = I_i(X, r^*)$ where $i = 4$ on the semiplanar case, and $i = 3$ for the laminar case. We will show that the sets $\mathcal{F}(X, Y, r^*) := \mathcal{F}(X, r^*) \cap [n]$ are forbidden sets of size at most 4 for Algorithm 8 and of size 3 for Algorithm 9.

Let $r^* \in \text{OPT}(Y)$ where Y is an arbitrary set of $t \geq s + 1$ elements, and suppose that $R_t = Y$ and $r_t = r^*$. Assume now that the condition

$$\text{for every } i \in \{s + 1, \ldots, t - 1\}, r_i \notin \mathcal{F}(\text{OPT}(R_i), r_t), \qquad (\star)$$

holds. We have to show that $r_t = r^*$ is chosen by the algorithm.

Claim. The intervals $I(\mathrm{OPT}(R_i), r^*)$ are non-decreasing in i, namely, for all $i \leq t-2$,

$$I(\mathrm{OPT}(R_i), r^*) \subseteq I(\mathrm{OPT}(R_{i+1}), r^*).$$

Indeed, let $i \leq t-2$. If $\mathrm{OPT}(R_i) = \mathrm{OPT}(R_{i+1})$ then the claim is trivial, so assume otherwise. In particular, we have $r_{i+1} \in \mathrm{OPT}(R_{i+1}) \setminus \mathrm{OPT}(R_i)$. To simplify notation, let $A = \mathrm{OPT}(R_{i+1})$ and $A' = A - r_{i+1}$. By condition $(\star)$, $r_{i+1} \notin \mathcal{F}(A, r^*)$, and then by Lemma 42 using $X = A$,

$$\mathcal{F}(A, r^*) = \mathcal{F}(A', r^*) \text{ and } I(A, r^*) = I(A', r^*). \tag{8.1}$$

By the matroid exchange axiom we have two cases: either $\mathrm{OPT}(R_i) = A'$ or $\mathrm{OPT}(R_i) = A' + \tilde{r}$ for some $\tilde{r} \neq r_{i+1}$. In the first case we have by (8.1) that $I(\mathrm{OPT}(R_i), r^*) = I(A', r^*) = I(A, r^*) = I(\mathrm{OPT}(R_{i+1}), r^*)$, which ends the proof, and so we focus on the the second case. If $\tilde{r} \in \mathcal{F}(A' + \tilde{r}, r)$, then by (8.1) and Lemma 42 applied to $X = A' + \tilde{r}$, we have $I(\mathrm{OPT}(R_i), r^*) = I(A' + \tilde{r}, r^*) \subseteq I(A', r^*) = I(A, r^*) = I(\mathrm{OPT}(R_{i+1}), r^*)$. On the other hand, if $\tilde{r} \notin \mathcal{F}(A' + \tilde{r}, r)$, then, again by Lemma 42 and and (8.1) we have $I(\mathrm{OPT}(R_i), r^*) = I(A' + \tilde{r}, r^*) = I(A', r^*) = I(A, r^*) = I(\mathrm{OPT}(R_{i+1}), r^*)$. This concludes the proof of the claim.

We show how to finish the proof using the claim. Suppose that r_i is selected by the algorithm for some $i \in \{s+1, \ldots, t-1\}$. By $(\star)$ and the claim we deduce that $r_i \notin I(\mathrm{OPT}(R_i), r^*) \supseteq I(\mathrm{OPT}(R_s), r^*)$. In particular r_i is *far away* from r^*:

(a) In the semiplanar case, there are at least 2 terminals of the set $\mathrm{OPT}(R_s)$ between r^* and r_i. In particular, $\{\mathrm{Left}_{\mathrm{OPT}(R_s)+r_i}(r_i), \mathrm{Right}_{\mathrm{OPT}(R_s)+r_i}(r_i)\}$ and $\{\mathrm{Left}_{\mathrm{OPT}(R_s)+r^*}(r^*), \mathrm{Right}_{\mathrm{OPT}(R_s)+r^*}(r^*)\}$ do not intersect and so, at iteration i, neither $\mathrm{Left}_{\mathrm{OPT}(R_s)+r^*}(r^*)$ nor $\mathrm{Right}_{\mathrm{OPT}(R_s)+r^*}(r^*)$ are added into B.

(b) In the laminar case, there is at least one terminal of the set $\mathrm{OPT}(R_s)$ between $\pi_{\mathrm{OPT}(R_s)}(r^*)$ and r_i. In particular, $\pi_{\mathrm{OPT}(R_s)}(r_i) \neq \pi_{\mathrm{OPT}(R_s)}(r^*)$, and so, at iteration i, $\pi_{\mathrm{OPT}(R_s)}(r^*)$ is not added into B.

Since the statements above are satisfied for every $i \leq t-1$, we conclude that at time t, $r_t = r^*$ satisfies the conditions in line 6 of the algorithms, and so it is selected. This concludes the proof that Algorithms 8 and 9 have forbidden sets of sizes 4 and 3 respectively. □

Chapter 9

Algorithm for Uniform Matroids

We devise a variant of Kleinberg's algorithm [66] for uniform matroids whose probability competitiveness tends to 1 as the rank ρ goes to infinity. Kleinberg's algorithm is better described when both the rank $\rho = 2^k$ and the number of elements $n = 2^N$ are powers of 2. It was originally presented in a recursive manner, but it is illustrative to describe it without using recursion.

Kleinberg's algorithm. For every $i \in \mathbb{N}$, let I_i be the interval of the first $n/2^i$ elements. Then the intervals $J_k := I_k, J_{k-1} := I_{k-1} \setminus I_k, \ldots, J_1 := I_1 \setminus I_2$ and $J_0 = I_0 \setminus I_1$ partition the ground set R. The algorithm treats each interval in $\{J_0, J_1, \ldots, J_k\}$ separately. For $0 \leq i \leq k-1$, it selects at most $b_i = \rho/2^{i+1}$ elements from J_i and at most one element from J_k, so that in total at most ρ elements are selected. The way the algorithm selects an element or not is determined by a threshold for each interval. Namely, it selects the first arriving element from J_k and for $i \in \{0, 1, \ldots, k-1\}$, the algorithm uses as threshold in the interval J_i the $(\rho/2^i)$-th highest element seen strictly before the interval, i.e., the $(\rho/2^i)$-th element of I_{i+1}. It selects every element better than this threshold until the budget b_i of the interval is depleted, ignoring all the elements arriving later in this interval.

The probability-ratio of Kleinberg's algorithm is at least 4/3. Kleinberg [66] shows that the previous algorithm is guaranteed to obtain an expected fraction $1-O(\sqrt{1/\rho})$ of the optimum weight, which in our notation means to be $1/(1-O(\sqrt{1/\rho})) = 1+O(\sqrt{1/\rho})$ utility-competitive. This guarantee also holds for the ordinal notion since the algorithm does not need weights. However, as we show next, its probability competitiveness is bounded away from 1. Since J_0 and I_1 have the same cardinality, with probability tending to $1/4$ as ρ goes to infinity, we simultaneously have that $|I_1 \cap R^{\rho-1}| < |J_0 \cap R^{\rho-1}|$ and $r^\rho \in J_0$. Given that, the threshold for J_0, which is the $\rho/2$-th element of I_1, will be attained by an element of $R^{\rho-1}$ which is strictly better than r^ρ. Thus, r^ρ will not be selected. Therefore, the algorithm selects r^ρ (which is in OPT) with probability at most $1 - 1/4 = 3/4$.

An asymptotically 1 probability-competitive algorithm. The next algorithm is a simple modification of Kleinberg's, which we write in a continuous setting. Every element is associated with a uniform random variable with support $[0, 1)$. It is useful for this part to imagine $[0, 1)$ as a time interval, and identify the realization of the uniform random variable as the *arrival time* of the element. For each $j \in \mathbb{N}$, let I_j be the interval $[0, 2^{-j})$ and $J_j = [2^{-j-1}, 2^{-j})$ to be its second half. The sequence $\{J_j\}_{j\in\mathbb{N}}$ partitions the interval $[0, 1)$. For convenience, let $K_j = [2^{-j-1}, 2^{-j-1}(2 - 4\varepsilon_j))$ be the left $(1 - 4\varepsilon_j)$ fraction of the interval J_j, for some parameter ε_j, depending on ρ, to be chosen later.

Algorithm 10 for uniform matroids.

Input: A uniform matroid $U(n, \rho)$ with ground set R and rank ρ.

1: Get a sample of size n, independently and uniformly at random from $[0, 1)$. Sort it from smallest to largest as $t_1 < t_2 < \cdots < t_n$. Thus t_i is interpreted as the arrival time of r_i.
2: ALG $\leftarrow \emptyset$
3: **for** $i = 1$ to n **do**
4: Compute the index j such that $t_i \in J_j = [2^{-j-1}, 2^{-j})$.
5: Compute the threshold f_j equal to the $\lceil (\frac{1}{2})^{j+1}(1 + \varepsilon_j)\rho \rceil$-th highest element in R_i with arrival time in $I_{j+1} = [0, 2^{-j-1})$.
6: **if** less than $(\frac{1}{2})^{j+1}\rho$ elements with arrival in J_j have been selected and $r_i \succ f_j$ **then**
7: ALG $\leftarrow$ ALG $+ r_i$
8: Return ALG.

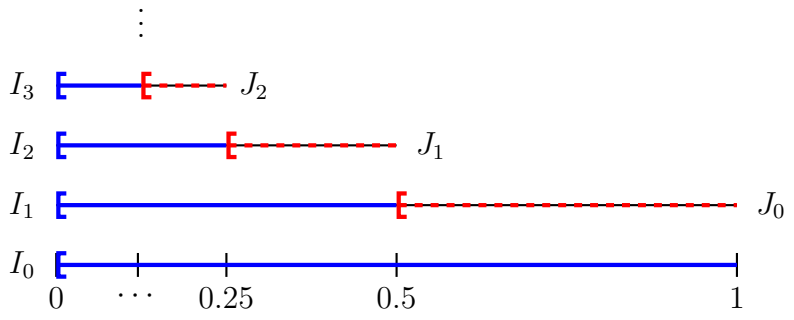


Figure 9.1: Definition of the sets I_j and J_j

Theorem 43. *Algorithm 10 is* $1 + O(\sqrt{\log \rho / \rho})$ *probability-competitive.*

By going from utility to probability we incur in a factor of $\sqrt{\log \rho}$ on the error probability. It remains open whether we can achieve Kleinberg's utility-competitiveness of $1 + O(\sqrt{1/\rho})$ for the stronger probability notion.

Proof of Theorem 43. Since the algorithm selects at most $(1/2)^{j+1}\rho$ elements from interval J_j, in total at most $\sum_{j\in\mathbb{N}}(1/2)^{j+1}\rho = \rho$ elements are included into ALG. Therefore, the algorithm is correct. Since the matroid is uniform of rank ρ, the optimal base is $\text{OPT} = \{r^1, r^2, \ldots, r^\rho\}$. Let $r^* = r^k \in \text{OPT}$, for some $k \in \{1, \ldots, \rho\}$. In what follows, let j be an integer such that $0 \leq j \leq \lfloor \frac{1}{2}\log(\rho/96)\rfloor := j^*$ and let $\varepsilon_j = \sqrt{12 \cdot 2^j \ln \rho/\rho}$. We first find a lower bound on the probability that r^* is selected, given that its arrival time is in J_j.

For each $r^i \in R - r^*$, let $X_i^{(j)}$ and $Y_i^{(j)}$ be the indicator variables that the arrival time of r^i is in I_{j+1} and K_j respectively. Let $\sigma(f_j)$ be the global ranking of the computed threshold f_j, that is, $r^{\sigma(f_j)} = f_j$. Consider the following three events,

$$\sum_{i\in[\rho+1]\setminus\{k\}} X_i^{(j)} < \frac{(1+\varepsilon_j)}{2^{j+1}}\rho, \quad \sum_{i\in\left[\left\lceil\frac{1+\varepsilon_j}{1-\varepsilon_j}\rho\right\rceil+1\right]\setminus\{k\}} X_i^{(j)} \geq \frac{1+\varepsilon_j}{2^{j+1}}\rho, \quad \sum_{i\in\left[\left\lceil\frac{1+\varepsilon_j}{1-\varepsilon_j}\rho\right\rceil+1\right]\setminus\{k\}} Y_i^{(j)} < \frac{1}{2^{j+1}}\rho,$$

that we call U_1, U_2 and U_3 respectively. Consider a fourth event, U_4, that is $t^k \in K_j$ where t^k is the arrival time of r^k. We claim that provided $t^k \in J_j$, the intersection of these four events guarantees that r^* is selected by the algorithm. Conditional on the event that $t^k \in J_j$, event $U_1 \cap U_2$ implies that $\rho + 1 < \sigma(f_j) \leq \lceil(1+\varepsilon_j)\rho/(1-\varepsilon_j)\rceil + 1$ and $r^* \succ f_j$. Event $U_2 \cap U_3$ implies that the number of elements strictly higher than the threshold f_j and arriving on interval K_j is strictly less than $(\frac{1}{2})^{j+1}\rho$. In particular, this implies that the algorithm has not selected enough elements in the interval J_j. Therefore, conditional on the event that $t^k \in J_j$, the event $U_1 \cap U_2 \cap U_3 \cap U_4$ implies that $r^* \in \text{ALG}$. Calling $\overline{U_i}$ to the negation of event U_i, by union bound and noting that events from 1 to 3 are independent of the arrival time of $r^* = r^k$, we have $\Pr(r^k \notin \text{ALG} \,|\, t^k \in J_j) \leq \sum_{i=1}^4 \Pr(\overline{U_i} \,|\, t^k \in J_j) = \Pr(\overline{U_1}) + \Pr(\overline{U_2}) + \Pr(\overline{U_3}) + 4\varepsilon_j$.

The random variables $\{X_i^{(j)}\}_{i\in[n]\setminus\{k\}}$ are independently and identically Bernoulli distributed of parameter equal to the length of I_{j+1}, that is 2^{-j-1}. Similarly, the random variables $\{Y_i^{(j)}\}_{i\in[n]\setminus\{k\}}$ are independently and identically Bernoulli distributed of parameter equal to the length of K_j that is $(1-4\varepsilon_j)2^{-j-1}$. In the following we upper bound the probabilities in the sum above by using the Chernoff bound [85, p. 64-66, Theorems 4.4 and 4.5], then

$$\Pr(\overline{U_1}) \leq \exp\left(-\frac{\varepsilon_j^2}{3}2^{-j-1}\rho\right) = \exp\left(-\frac{12\cdot 2^j \ln\rho}{3\rho}2^{-j-1}\rho\right) = \exp\left(-2\ln\rho\right) \leq \frac{1}{\rho}.$$

Let μ_X and μ_Y be the expected sums of the random variables $X_i^{(j)}$, and respectively $Y_i^{(j)}$, for $i \in \left[\left\lceil\frac{1+\varepsilon_j}{1-\varepsilon_j}\rho\right\rceil + 1\right] \setminus \{k\}$. We have $\mu_X = \left\lceil\frac{1+\varepsilon_j}{1-\varepsilon_j}\rho\right\rceil \left(\frac{1}{2}\right)^{j+1} \geq \frac{1+\varepsilon_j}{1-\varepsilon_j} \cdot \frac{\rho}{2^{j+1}}$. The choice of j guarantees that $1/\rho < \epsilon_j < 1/8$, and therefore

$$\begin{aligned}\mu_Y = \left\lceil\frac{1+\varepsilon_j}{1-\varepsilon_j}\rho\right\rceil \frac{1-4\varepsilon_j}{2^{j+1}} &\leq \left(\frac{1+\varepsilon_j}{1-\varepsilon_j} + \varepsilon_j\right)\rho \cdot \frac{1-4\varepsilon_j}{2^{j+1}} \\ &\leq \frac{(1-4\varepsilon_j)(1+2\varepsilon_j)}{1-\varepsilon_j}\frac{\rho}{2^{j+1}} \leq \frac{1}{1+\varepsilon_j}\cdot\frac{\rho}{2^{j+1}},\end{aligned}$$

where in the last inequality we used that $\varepsilon_j < 1/8$. By Chernoff bound on events $\overline{U_2}$ and $\overline{U_3}$, we obtain

$$\begin{aligned}\Pr(\overline{U_2}) \le \Pr\left(\sum_i X_i^{(j)} < \mu_X(1-\varepsilon_j)\right) &\le \exp\Big(-\frac{\varepsilon_j^2}{2}\mu_X\Big) \\ &\le \exp\Big(-\frac{12\cdot 2^j \ln\rho}{2\rho}\cdot\frac{1+\varepsilon_j}{1-\varepsilon_j}\cdot\frac{\rho}{2^{j+1}}\Big) \le \frac{1}{\rho},\end{aligned}$$

$$\begin{aligned}\Pr(\overline{U_3}) \le \Pr\left(\sum_i Y_i^{(j)} \ge \mu_Y(1+\varepsilon_j)\right) &\le \exp\Big(-\frac{\varepsilon_j^2}{3}\mu_Y\Big) \\ &\le \exp\Big(-\frac{12\cdot 2^j \ln\rho}{3\rho}\cdot\frac{1+\varepsilon_j}{1-\varepsilon_j}\cdot\frac{(1-4\varepsilon_j)\rho}{2^{j+1}}\Big) \le \frac{1}{\rho}.\end{aligned}$$

Putting all together, it follows that

$$\begin{aligned}\Pr(r^k \notin \mathrm{ALG}) &= \sum_{j>j^*} \Pr(r^k \notin \mathrm{ALG} | t^k \in J_j)\frac{1}{2^{j+1}} + \sum_{j=0}^{j^*} \Pr(r^k \notin \mathrm{ALG} | t^k \in J_j)\frac{1}{2^{j+1}} \\ &\le \sum_{j>j^*}\frac{1}{2^{j+1}} + \sum_{j\ge 0}\left(\frac{3}{\rho}+4\varepsilon_j\right)\frac{1}{2^{j+1}} = \frac{1}{2^{j^*+1}} + \left(\frac{3}{\rho} + 4\sqrt{\frac{12\ln\rho}{\rho}}\sum_{j\ge 0}\frac{2^{j/2}}{2^{j+1}}\right),\end{aligned}$$

which is $O\left(\sqrt{\frac{1}{\rho}} + \frac{1}{\rho} + \sqrt{\frac{\log\rho}{\rho}}\right) = O\left(\sqrt{\frac{\log\rho}{\rho}}\right)$. Then, the algorithm is $(1-O(\sqrt{\frac{\log\rho}{\rho}}))^{-1} = (1+O(\sqrt{\log\rho/\rho}))$ probability-competitive. □

Chapter 10

Algorithms for general matroids

In this section we provide algorithms for general matroids in the different competitiveness notions we consider in this work. We first show an $O(1)$-intersection competitive algorithm and then we proceed with ordinal/probability notions. Our algorithm for the intersection notion works as follows. We first sample s elements, where s is chosen appropriately later. After that, we select an element as long as it is part of the optimum of the subset of elements seen so far, and it preserves the independence of the current solution. Recall that we denote by $R_i = \{r_1, r_2, \ldots, r_i\}$ the first i elements seen by the algorithm.

Algorithm 11 Improving Greedy

Input: Matroid $\mathcal{M}(R, I)$ in random order $r_1, r_2, \ldots, r_n$, and a fixed integer $s \in \{1, \ldots n\}$.

1: ALG $\leftarrow \emptyset$
2: **for** $i = s + 1$ to n **do**
3: **if** $r_i \in \text{OPT}(R_i)$ and $\text{ALG} + r_i \in \mathcal{I}$ **then**
4: ALG $\leftarrow$ ALG $+ r_i$.
5: Return ALG.

Theorem 17. *Algorithm 11 is* $\ln(e/2)$ *intersection-competitive for the general MSP.*

The idea of only considering elements that belong to the current optimum is not new. Ma et al. [79] consider an algorithm that after sampling a fraction of the elements, selects an element as long as it belongs to the current offline optimum, and they prove this to be 9.6 utility-competitive (and the analysis holds for probability as well) for laminar matroids. The same algorithm was suggested by Babaioff et al. [11], and they showed how this algorithm fails to be even constant utility-competitive. In contrast, we show this algorithm to be $O(1)$ intersection-competitive.

Lemma 44. *Let* $B = \{r_i : r_i \in \text{OPT}(R_i),\ i \in \{s+1, \ldots, n\}\}$. *Then* $\mathbb{E}[|B|] = (H_n - H_s)\rho$, *where* H_j *denotes the* j*-th harmonic number.*

Proof. For every $i \in [n]$ if we condition the set R_i to be some fixed set $F \in \binom{R}{i}$, we have

$$\Pr(r_i \in \mathrm{OPT}(R_i) | R_i = F) = \Pr(r_i \in \mathrm{OPT}(F)) \leq \frac{\rho}{i}.$$

Therefore, by linearity of expectation we conclude that

$$\mathbb{E}[|B|] = \sum_{i=s+1}^{n} \Pr(r_i \in \mathrm{OPT}(R_i)) \leq \sum_{i=s+1}^{n} \frac{\rho}{i} = (H_n - H_s)\rho. \qquad \square$$

Proof of Theorem 17. We study the competitiveness of the algorithm when the sample size is s, and then we optimize over this value to conclude the theorem. For any random ordering, $|\mathrm{ALG}| = \rho(R \setminus R_s) \geq \rho(\mathrm{OPT} \setminus R_s) = |\mathrm{OPT} \setminus R_s|$. Therefore,

$$\begin{aligned}\mathbb{E}[|\mathrm{ALG} \cap \mathrm{OPT}|] &\geq \mathbb{E}[|\mathrm{ALG}|] + \mathbb{E}[|\mathrm{OPT} \setminus R_s|] - \mathbb{E}[|\mathrm{ALG} \cup (\mathrm{OPT} \setminus R_s)|] \\ &\geq 2\mathbb{E}[|\mathrm{OPT} \setminus R_s|] - \mathbb{E}[|\mathrm{ALG} \cup (\mathrm{OPT} \setminus R_s)|].\end{aligned}$$

Furthermore, $\mathrm{ALG} \cup (\mathrm{OPT} \setminus R_s) \subseteq B$, where B is the set defined in Lemma 44. Since the elements arrive in uniform random order, for $r \in \mathrm{OPT}$ we have that $\Pr(r \notin R_s) = (n-s)/n = 1 - s/n$. Therefore, the right hand side of the previous inequality is at least

$$\begin{aligned}2\mathbb{E}[|\mathrm{OPT} \setminus R_s|] - \mathbb{E}[|B|] &= 2\left(1 - \frac{s}{n}\right)\rho - (H_n - H_s)\rho \\ &\geq \left(2 - \frac{2s}{n} - \int_s^n \frac{1}{x} dx\right)\rho = \left(2 - \frac{2s}{n} + \ln(s/n)\right)\rho.\end{aligned}$$

This quantity is maximized in $s = n/2$. So, by assuming n even (which can be done by adding an extra dummy element if n is odd), and setting the algorithm for $s = n/2$, we obtain

$$\mathbb{E}[|\mathrm{ALG} \cap \mathrm{OPT}|] \geq \rho(1 - \ln(2)) = |\mathrm{OPT}|/\ln(e/2).$$

The same holds if we modify the algorithm by choosing a random sample size of expected value $n/2$, that is, $s \sim \mathrm{Bin}(n, 1/2)$. $\square$

10.1 Ordinal/Probability: Proof of Theorem 9

We introduce a variant of the MSP that helps us to leverage existing algorithms for the utility version of the MSP, in order to get competitive algorithms for the ordinal and probability variants. We need a concept similar to the *aided sample-based MSP* introduced by Feldman et al . [39].

In the *Layered-MSP* the input is a tuple $(\mathcal{M}, F, C, \succ)$ where $\mathcal{M} = (R, \mathcal{I}, \succ)$ is a totally ordered matroid, $C = \{c_1, c_2, \ldots, c_k\}$ is a finite set with $C \cap R = \emptyset$, $\succ$ is a total order over $R \cup C$ with $c_1 \succ c_2 \succ \cdots \succ c_k$, and F is a random subset of R in which every element is present with probability $1/2$. The set C defines a partition of R in the following way. We call $C_0 = \{r \in R : r \succ c_1\}$ the *highest layer* and

$C_k = \{r \in R : c_k \succ r\}$ the *lowest layer*. For $j \in \{1, \ldots, k-1\}$, the *j-th layer* is the set $C_j = \{r \in R : c_j \succ r \succ c_{j+1}\}$. By construction, the layers $\{C_0, C_1, \ldots, C_k\}$ induced by C form a partition of R. In the following we call a tuple $(\mathcal{M}, F, C, \succ)$ a *layered matroid*.

An algorithm for the Layered-MSP first sees F but is unable to select any of its elements. The rest of the elements of $\mathcal{M}$ arrive in uniform random order. At time step t, the algorithm can get the full value order in R_t by using only ordinal information, it can use an independence oracle to test any subset of R_t and it can also check membership of any element to each layer induced by C. We say that an algorithm for the Layered-MSP is α-competitive if it returns an independent set $\text{ALG} \in \mathcal{I}$, and for each $j \in \{0, 1, \ldots, |C|\}$, $\mathbb{E}[|\text{ALG} \cap C_j|] \geq \frac{1}{\alpha}|\text{OPT} \cap C_j|$, where the expectation is taken both over the distribution of F and the internal algorithm randomness.

Theorem 45 (Feldman et al., [39, Corollary 4.1])**.** *There exists an $8\lceil\log(|C|+1)+1\rceil$-competitive algorithm for the Layered-MSP.*

For the sake of completeness we include the mentioned algorithm for Layered-MSP of Feldman et al. We remark that the algorithm was introduced in the context of the utility version. Nevertheless, the result above follows in the absence of weights.

Algorithm 12 (Feldman et al. [39]) for Layered-MSP

Input: A layered matroid $(\mathcal{M}, F, C, \succ)$

Let τ be a uniformly at random number from $\{0, 1, \ldots, \lceil\log(|C|+1)\rceil\}$.
Let Δ be a uniformly at random number from $\{0, 1, \ldots, 2^\tau - 1\}$.
Let $\mathbf{B} = \{B_1, B_2, \ldots, B_{\lceil(\Delta+|C|)/2^\tau\rceil}\}$ where $B_i = \bigcup_{j=\max\{0,2^\tau(i-1)-\Delta+1\}}^{\min\{|C|,2^\tau i-\Delta\}} C_j$.
With probability $1/2$, set $H = \text{odd}(|C|)$ or $\text{even}(|C|)$ otherwise.
For each $i \in H$ let $T_i \leftarrow \emptyset$.
for each element in $r \in R \setminus F$ **do**
    Let i be such that $r \in B_i$.
    if $i \in H$ and $r \in N_i$ and $T_i + r \in \mathcal{I}_i$ **then**
        $T_i \leftarrow T_i + r$.
Return $\text{ALG} = \cup_{i \in H} T_i$.

About the algorithm of Feldman et al. We denote by $\text{odd}(k)$ and $\text{even}(k)$ the odd and even numbers, respectively, in the set $\{0, 1, \ldots, k\}$. The set $\mathcal{I}_i$ is the independent sets family of a matroid $M_i = (N_i, \mathcal{I}_i)$ defined as follows. Let $B_{\geq i} = \cup_{j \in \{i, \ldots, \lceil(\Delta+|C|)/2^\tau\rceil\}} B_j$. The matroid M_1 is obtained from M by contracting [1] $F \cap B_{\geq 2}$ and then restricting[2] to B_1. For $i > 1$, M_i is obtained from M by contracting $F \cap B_{\geq i+1}$ and then restricting it to $B_i \cap \text{span}(F \cap B_{\geq i-1})$.

[1] The *contraction* of $\mathcal{M} = (R, \mathcal{I})$ by Q, $\mathcal{M}/Q$, has ground set $R - Q$ and a set I is independent if $\rho(I \cup Q) - \rho(Q) = |I|$.

[2] The *restriction* of $\mathcal{M} = (R, \mathcal{I})$ to Q, $\mathcal{M}|_Q$, has ground set Q and a set I is independent if $I \in \mathcal{I}$ and $I \subseteq Q$.

From Layered-MSP to ordinal MSP. The main result of this section corresponds to the lemma below, which is a reduction between the ordinal MSP and the version introduced in this section, the Layered MSP. In particular, it allows us to design ordinal/probability-competitive algorithms as long as we have a competitive algorithm for the Layered MSP. Theorem 9 follows directly using this lemma, and the rest of this section is devoted to prove the lemma.

Lemma 46. *Suppose there exists a $g(|C|)$-competitive algorithm for the Layered-MSP, where g is a non-decreasing function. Then,*

(i) there exists an $O(g(1+\log\rho))$ ordinal-competitive algorithm for the MSP, and

(ii) there exists an $O(g(1+\rho))$ probability-competitive algorithm for the MSP.

Proof of Theorem 9. We observe that $g(x) = 8\lceil\log(x+1)+1\rceil$ is non-decreasing, so we can apply Lemma 46 using Algorithm 12 and Theorem 45. □

In the following, let $\mathcal{A}^{\text{layer}}$ be a $g(|C|)$-competitive algorithm for the layered-MSP. Our algorithm for Lemma 46 (i), depicted as Algorithm 13, first gets a sample from R with expected size $n/2$, and constructs a partition C using the optimum of the sample. By sampling a set F over the remaining elements it feeds $\mathcal{A}^{\text{layer}}$ with a layered matroid.

Algorithm 13 $O(g(1+\log\rho))$ ordinal-competitive algorithm

Input: Matroid $\mathcal{M}(R,\mathcal{I},\succ)$ in random order $r_1, r_2, \ldots, r_n$.

1: Let $s \sim \text{Bin}(n, 1/2)$ and compute $\text{OPT}(R_s) = \{s(1), \ldots, s(\ell)\}$, where $s(1) \succ s(2) \succ \cdots \succ s(\ell)$.
2: Let $C = \{s(1), s(2), s(4), \ldots, s(2^{k-1})\}$, where $k = \lfloor\log\ell\rfloor + 1$.
3: Let $t \sim \text{Bin}(n-s, 1/2)$ and let $F = \{r_{s+1}, \ldots, r_{s+t}\}$ be the next t elements from $R \setminus R_s$.
4: Return $\text{ALG} = \mathcal{A}^{\text{layer}}(\mathcal{M}|_{R\setminus R_s}, F, C, \succ)$.

Proof of Lemma 46 (i). Let $\text{OPT} = \{f(1), \ldots, f(\rho)\}$ be such that $f(1) \succ f(2) \succ \cdots \succ f(\rho)$. Consider the function $T : \mathbb{N} \to \mathbb{N}$ given by $T(0) = 0$, $T(i) = |\{r \in R\colon r \succ f(i)\}|$ if $i \in [1,\rho]$ and $T(i) = n$ if $i > \rho$. In particular, for $i \in [1,\rho]$, $T(i)$ is the number of elements in $\mathcal{M}$ that are at least as high as the i-th element of OPT, so $f(i) = r^{T(i)}$. Observe that for all k such that $T(i) \le k < T(i+1)$ we have $|\text{OPT} \cap R^k| = i$. In the following we study the expected number of elements from R^k that the algorithm selects, so we can conclude using Lemma 33. If $T(0) \le k < T(1)$ it holds $\mathbb{E}[|\text{ALG} \cap R^k|] = 0 = |\text{OPT} \cap R^k|$, so this case is done.

Let $k \in \mathbb{N}$ be such that $T(1) \le k < T(8)$. Let f be the highest non-loop element in the value order with $f(1) \succ f$. With probability at least $1/4$, it holds that $f(1) \in R \setminus R_s$ and $f \in R_s$. In this case, $f \in \text{OPT}(R_s)$, since $f(1) = s(1)$, and the only non-loop element of C_0 is $f(1)$. Thus, $C_0 = \{f(1)\} = \text{OPT}(R \setminus R_s) \cap C_0$. Therefore,

$$\Pr(C_0 = \{f(1)\}) \ge \Pr(f(1) \in R \setminus R_s, f \in R_s) \ge 1/4.$$

Since g is non-decreasing and $|C| \leq 1+\log \ell \leq 1+\log \rho$, Algorithm $\mathcal{A}^{\text{layer}}$ is $g(1+\log \rho)$-competitive. Furthermore, the event $C_0 = \{f(1)\}$ depends only on steps 1 and 2 of the algorithm before executing $\mathcal{A}^{\text{layer}}$. It follows that

$$\begin{aligned}\Pr(f(1) \in \text{ALG}) &\geq \frac{1}{4}\Pr(f(1) \in \text{ALG} | C_0 = \{f(1)\}) = \frac{1}{4}\mathbb{E}[|\text{ALG} \cap C_0| \mid C_0 = \{f(1)\}] \\ &\geq \frac{1}{4g(1+\log \rho)}\mathbb{E}[|\text{OPT}(R \setminus R_s) \cap C_0| \mid C_0 = \{f(1)\}] = \frac{1}{4g(1+\log \rho)}.\end{aligned}$$

Since $|\text{OPT} \cap R^k| \leq 8$, we conclude in this case that

$$\mathbb{E}[|\text{ALG} \cap R^k|] \geq \Pr(f(1) \in \text{ALG}) \geq \frac{1}{4g(1+\log \rho)} \geq \frac{1}{32g(1+\log \rho)}|\text{OPT} \cap R^k|.$$

Let $j \geq 3$ and k be such that $T(2^j) \leq k < T(2^{j+1})$. We denote $q = 2^{j-3}$. Let A_j be the event where $|\{f(1), \ldots, f(2q)\} \cap R \setminus R_s| \geq q$ and B_j is the event where $|\{f(2q+1), \ldots, f(6q)\} \cap R_s| \geq 2q$. We have that $\Pr(A_j \cap B_j) \geq 1/4$, since in our algorithm the probability for an element to be sampled equals the probability of not being sampled. Observe that any subset of elements strictly better than $f(t)$ has rank at most $t-1$, and therefore $f(2q) \succeq s(2q)$. If B_j holds, then $s(2^{j-2}) = s(2q) \succeq f(6q)$. Since $f(6q) \succeq f(8q) = f(2^j)$, it follows that

$$\bigcup_{i=0}^{j-3} C_i \subseteq \{r \in R \setminus R_s : r \succeq f(2^j)\} = R^{T(2^j)} \cap R \setminus R_s.$$

This implies that

$$\mathbb{E}[|\text{ALG} \cap R^k|] \geq \frac{1}{4}\mathbb{E}\Big[|\text{ALG} \cap R^{T(2^j)}| \,\Big|\, A_j \cap B_j\Big] \geq \frac{1}{4}\mathbb{E}\Big[\Big|\text{ALG} \cap \bigcup_{i=0}^{j-3} C_i\Big| \,\Big|\, A_j \cap B_j\Big].$$

Furthermore, if A_j holds, then

$$\begin{aligned}\Big|\text{OPT}(R \setminus R_s) \cap \bigcup_{i=0}^{j-3} C_i\Big| &= |\{f \in \text{OPT} \cap R \setminus R_s \colon f \succeq s(2q)\}| \\ &\geq |\{f \in \text{OPT} \cap R \setminus R_s \colon f \succeq f(2q)\}| \geq q.\end{aligned}$$

Since g is non-decreasing and $|C| \leq 1+\log \ell \leq 1+\log \rho$, Algorithm $\mathcal{A}^{\text{layer}}$ is $g(1+\log \rho)$-competitive. Since events A_j and B_j depend only on the sampling at line 1 of the algorithm (before executing $\mathcal{A}^{\text{layer}}$) and by using linearity of the expectation and the observation above we have that

$$\begin{aligned}\frac{1}{4}\mathbb{E}\Big[\Big|\text{ALG} \cap \bigcup_{i=0}^{j-3} C_i\Big| \,\Big|\, A_j \cap B_j\Big] &\geq \frac{1}{4g(1+\log \rho)}\mathbb{E}\Big[\Big|\text{OPT}(R \setminus R_s) \cap \bigcup_{i=0}^{j-3} C_i\Big| \,\Big|\, A_j \cap B_j\Big] \\ &\geq \frac{1}{4g(1+\log \rho)}q \geq \frac{1}{64g(1+\log \rho)}|\text{OPT} \cap R^k|,\end{aligned}$$

where the last inequality holds since $|\text{OPT} \cap R^k| \leq 2^{j+1} = 16q$. By using Lemma 33 we conclude that the algorithm is $O(g(1+\log \rho))$ ordinal-competitive. □

Algorithm 14 $O(g(1+\rho))$ probability-competitive algorithm

Input: Matroid $\mathcal{M}(R, \mathcal{I}, \succ)$ in random order $r_1, r_2, \ldots, r_n$.

1: Let $s \sim \text{Bin}(n, 1/2)$ and compute $\text{OPT}(R_s) = \{s(1), \ldots, s(\ell)\}$, where $s(1) \succ s(2) \succ \cdots \succ s(\ell)$.
2: Let $t \sim \text{Bin}(n-s, 1/2)$ and let $F = \{r_{s+1}, \ldots, r_{s+t}\}$ be the next t elements from $R \setminus R_s$.
3: Return $\text{ALG} = \mathcal{A}^{\text{layer}}(\mathcal{M}|_{R_s^+}, F \cap R_s^+, \text{OPT}(R_s), \succ)$, where $R_s^+ = \{r \in R \setminus R_s : r \in \text{OPT}(R_s + r)\}$.

To prove Lemma 46 (ii), consider Algorithm 14 depicted above. Before the analysis, it will be useful to consider the next process. Let $(X_t)_{t\in\mathbb{N}}$ be a sequence of Bernoulli independent random variables such that $\Pr(X_t = 0) = \Pr(X_t = 1) = 1/2$ for every $t \in \mathbb{N}$. We create two sets $V, W \subseteq R$ iteratively using the following procedure.

Algorithm 15 Coupling procedure

Input: Matroid $\mathcal{M}(R, \mathcal{I}, \succ)$.

Initialize $V \leftarrow \emptyset$, $W \leftarrow \emptyset$ and $\theta \leftarrow 0$.
for $i = 1$ to n **do**
    if $r^i \in \text{OPT}(V + r^i)$ **then**
        $\theta \leftarrow \theta + 1$ and $Y_\theta = X_i$.
        if $Y_\theta = 0$ **then**
            $V \leftarrow V + r^i$
        else $W \leftarrow W + r^i$.

The value θ represents a counter on the elements that improve over the current set V. When the counter is updated, we say that the element considered on that iteration *is assigned coin* Y_θ.

Lemma 47. *(V, W) has the same distribution as $(\text{OPT}(R_s), R_s^+)$ in Algorithm 14.*

Proof. Let $(V_i, W_i)_{i=1}^n$ be the states of the coupling process at the end of each iteration. Let $Z = \{r^i : X_i = 0\}$. Observe that Z and the sample R_s of Algorithm 14 have the same distribution. Since the coupling procedure checks from highest to lowest element, it follows that $V_i = \text{OPT}(Z \cap R^i)$ for every $i \in \{1, \ldots, n\}$, and therefore $V_n = \text{OPT}(Z)$. To conclude the lemma it suffices to check that $W_n = \{r \in R \setminus Z : r \in \text{OPT}(Z + r)\}$. In fact, it can be proven by induction that

$$W_i = \{r \in R^i \setminus Z : r \in \text{OPT}(V_i + r)\}$$

for every $i \in \{1, \ldots, n\}$, and the lemma follows, since $R^n = R$ and $V_n = \text{OPT}(Z)$. □

Proof of Lemma 46 (ii). Thanks to Lemma 47 we assume that $(\text{OPT}(R_s), R_s^+)$ is generated by the process described in Algorithm 15. In what follows, fix $f(j) = r^{T(j)} \in \text{OPT}$. We know that at step $T(j)$ of the coupling procedure, $f(j) \in \text{OPT}(R_s + f(j))$ no matter the trajectory of $(X_i)_{i\in\mathbb{N}}$. Let θ be such that $r^{T(j)}$ is assigned coin Y_θ,

that is, $Y_\theta = X_{T(j)}$. Then, with probability at least $1/8$, the event $\mathcal{E}$ defined as $Y_{\theta-1} = Y_{\theta+1} = 0$ and $Y_\theta = 1$, holds. If $\mathcal{E}$ happens, let $s(h)$ be the element of $\mathrm{OPT}(R_s)$ who was assigned coin $Y_{\theta-1}$. In particular, the element $s(h+1)$ is assigned coin $Y_{\theta+1}$. Thus,

$$C_h = \{r \in R_s^+ \colon s(h) \succ r \succ s(h+1)\} = \{f(j)\} = \mathrm{OPT}(R_s^+) \cap C_h.$$

Therefore by using that the occurrence of $\mathcal{E}$ is decided before executing $\mathcal{A}^{\text{layer}}$ which is $g(1+\ell) \leq g(1+\rho)$ competitive, we get that

$$\begin{aligned}\Pr(f(j) \in \mathrm{ALG}) &\geq \frac{1}{8}\Pr(f(j) \in \mathrm{ALG}|\mathcal{E}) \\ &= \frac{1}{8}\mathbb{E}[|\mathrm{ALG} \cap C_h||\mathcal{E}] \geq \frac{1}{8g(1+\rho)}\mathbb{E}[|\mathrm{OPT}(R_s^+) \cap C_h||\mathcal{E}] = \frac{1}{8g(1+\rho)},\end{aligned}$$

and we conclude that Algorithm 14 is $O(g(1+\rho))$ probability-competitive. □

10.2 Comparison between ordinal measures

In this section we discuss about some incomparability results for the competitiveness measures previously introduced. We show that an algorithm that is utility-competitive is not necessarily competitive for the rest of the measures. In particular, we provide an instance where the $O(\log \rho)$ utility-competitive algorithm by Babaioff, Immorlica and Kleinberg [11] has a poor competitiveness for the other three. Regarding the notions of the ordinal MSP, we show that the intersection and the ordinal measures are incomparable. More specifically, we show the existence of an algorithm and an instance where it is arbitrarily close to 1 intersection-competitive, but have an unbounded ordinal/probability-competitiveness. And on the other hand, we also show the existence of an algorithm and an instance where it is arbitrarily close to 1 ordinal-competitive, but has an unbounded intersection/probability-competitiveness. Recall that probability is the strongest in the sense that it implies competitiveness for all the other measures considered (see Lemma 34).

In the utility variant, the weight $w(r)$ of an element is revealed to the algorithm when arrived. Suppose for simplicity that the rank ρ of the matroid is known[3] and that the weights the algorithm sees are all different. In the above mentioned algorithm, called by the authors the *Threshold Price Algorithm* (TPA), it is taken a sample R_s of $s \sim \mathrm{Bin}(n, 1/2)$ elements[4] and it records the top weight w^* of a non-loop[5] element seen in R_s. It chooses uniformly at random a number τ in the set $\{0, 1, 2, \ldots, \lceil \log_2 \rho \rceil\}$, and then it selects greedily any non-sampled element whose weight is at least $T = w^*/2^\tau$.

Theorem 48 (Babaioff et al. [11])**.** *The algorithm TPA is $O(\log \rho)$ utility-competitive.*

[3]This assumption can be removed by using standard arguments; we can set ρ equal to twice the rank of the sampled part.

[4]The original analysis uses half of n, but the analysis gets simpler if one uses $\mathrm{Bin}(n, 1/2)$ since one can assume that each element is in the sample with probability $1/2$ independent of the rest.

[5]A *loop* in a matroid is an element that belongs to no basis.

We show in this section that TPA is $\Omega(\rho)$-competitive in the intersection, ordinal and probability measures. We first revisit the proof by [11] for the $O(\log \rho)$ utility-competitiveness of TPA.

Proof of Theorem 48. Let $\mathrm{OPT} = \{f(1), \ldots, f(\rho)\}$ be the optimal base such that $w_1 > w_2 > \cdots > w_\rho$, where $w_i = w(f(i))$ for each $i \in \{1, \ldots, \rho\}$. Suppose that $f(1) \notin R_s$. Then, if the second non-loop element of the matroid is sampled and if $\tau = 0$, the element $f(1)$ will be selected in the second phase of TPA. Hence $\Pr(f(1) \in \mathrm{ALG}) \geq 1/4 \cdot 1/(\lceil \log \rho \rceil + 1) = \Omega(1/\log \rho)$.

Let $B = \{i \in \{2, \ldots, \rho\} : w_i \geq w_1/\rho\}$, and let $\mathcal{E}_i$ be the event where $f(1) \in R_s$ and $w_i/2 < T \leq w_i$. For each $i \in B$, we have $\log(w_1/w_i) < \log \rho$ and therefore $\Pr(\mathcal{E}_i) = 1/2 \cdot \Pr(\tau = \lceil \log(w_1/w_i) \rceil) = \Omega(1/\log \rho)$. The random order assumption implies that in expectation $(i-1)/2 \geq i/4$ elements in $\{f(2), \ldots, f(i)\}$ are non-sampled, hence the expected rank of the matroid restricted to non-sampled elements of weight at least $w_i = w(f(i))$ is $\Omega(i)$. Therefore, given $i \in B$ and conditioned on $\mathcal{E}_i$, the algorithm selects $\Omega(i)$ elements of value at least $w_i/2$. It follows that for each $i \in B$,

$$\mathbb{E}[|\mathrm{ALG} \cap \{r \colon w(r) \geq w_i/2\}|] \geq \Pr(\mathcal{E}_i)\mathbb{E}[|\mathrm{ALG} \cap \{r \colon w(r) \geq w_i/2\}||\mathcal{E}_i] = \Omega\left(\frac{i}{\log \rho}\right).$$

In addition, observe that the elements in $\mathrm{OPT} \setminus \{f(i) : i \in B \cup \{1\}\}$ have total weight less than $w_1(\rho - |B| - 1)/\rho < w_1$, and therefore $2\sum_{i \in B \cup \{1\}} w_i \geq w(\mathrm{OPT})$. Putting all together, we have that the expected weight of the output is

$$\begin{aligned}
\mathbb{E}[w(\mathrm{ALG})] &\geq \frac{1}{2}\sum_{i=1}^{\rho}(w_i - w_{i+1})\mathbb{E}[|\mathrm{ALG} \cap \{r \colon w(r) \geq w_i/2\}|] \\
&= \Omega\left(\frac{1}{\log \rho}\right) \sum_{i \in B \cup \{1\}} (w_i - w_{i+1}) \cdot i \\
&= \Omega\left(\frac{1}{\log \rho}\right) \sum_{i \in B \cup \{1\}} w_i = \Omega\left(\frac{1}{\log \rho}\right) w(\mathrm{OPT}).
\end{aligned}$$

□

We study in the following the competitiveness of TPA for the ordinal measures. Since probability is the strongest (Lemma 34), it is enough to check that TPA is $\Omega(\rho)$ ordinal and intersection competitive. Let $R = \{r^1, \ldots, r^n\}$ with $n = 2\rho^3$. Consider the laminar family $\{R^{n/2}, R\}$, where $R^{n/2} = \{r^1, \ldots, r^{n/2}\}$, and take the laminar matroid $(R, \mathcal{I})$ where a set $I \in \mathcal{I}$ is independent if $|I \cap R^{n/2}| \leq 1$ and $|I| \leq \rho$. In particular, the matroid rank is ρ. Given $\varepsilon > 0$, the weights are given by $w(r^i) = 8 - \varepsilon i$ for $i \in \{1, \ldots n/2\}$, and $w(r^i) = 7 - \varepsilon i$ for $i \in \{n/2 + 1, \ldots, n\}$.

Observe that the optimal basis is given by $\mathrm{OPT} = \{r^1, r^{n/2+1}, \ldots, r^{n/2+\rho-1}\}$. If we run TPA on this instance, with probability $p = 1 - 2^{-n/2}$ the top weight w^* in the sample is from $R^{n/2}$, and thus, $7 < w^* < 8$. Let $\mathcal{E}$ be this event. No matter the value of T, the algorithm always select at most ρ elements in the non-sampled part, and

therefore

$$\begin{aligned}\mathbb{E}[|\mathrm{ALG}\cap\mathrm{OPT}|] &= (1-2^{-n/2})\mathbb{E}[|\mathrm{ALG}\cap\mathrm{OPT}||\mathcal{E}] + 2^{-n/2}\mathbb{E}[|\mathrm{ALG}\cap\mathrm{OPT}||\overline{\mathcal{E}}]\\ &\leq \mathbb{E}[|\mathrm{ALG}\cap\mathrm{OPT}||\mathcal{E}] + \rho\cdot 2^{-\rho^3} \leq \mathbb{E}[|\mathrm{ALG}\cap\mathrm{OPT}||\mathcal{E}] + 1.\end{aligned}$$

Conditional on $\mathcal{E}$, we have that either $T > 7$ or $T < 4$. In the first case, TPA will select at most one element, which would come from $R^{n/2}$, and so $|\mathrm{ALG}\cap\mathrm{OPT}| \leq 1$. Otherwise, if $T < 4$, the algorithm will select at most one element from $R^{n/2}$ and at most the first ρ non-sampled elements from $R\setminus R^{n/2}$. The expected number of elements in $\{r^{n/2+1},\ldots,r^{n/2+\rho-1}\}$ appearing in $\{r_{s+1},\ldots,r_{s+\rho}\}$, that is the first ρ elements appearing after the sample, is $(\rho-1)\rho/n < 1$. It follows that $\mathbb{E}[|\mathrm{ALG}\cap\mathrm{OPT}||\mathcal{E}]$ is upper bounded by 2, and therefore $\mathbb{E}[|\mathrm{ALG}\cap\mathrm{OPT}|] \leq 3$. The discussion above implies as well that $\mathbb{E}[|\mathrm{ALG}\cap R^{n/2+\rho-1}|] \leq 3$, and then TPA is $\Omega(\rho)$ intersection and ordinal competitive. The conclusion for the ordinal case follows by using the characterization for ordinal competitiveness in Lemma 33.

Ordinal and intersection notions. We show in the following that there is no competitiveness dominance between the intersection and the ordinal notions. Let m and M be two positive integers. Consider $R = \{r^1, r^2, \ldots, r^{(M+1)m}\}$ and let $\mathcal{L}$ be the partition of R given by

$$\mathcal{L} = \bigcup_{i=1}^{m}\{\{r^i\}\} \cup \bigcup_{j=1}^{M}\{\{r^{jm+1},\ldots,r^{(j+1)m}\}\}.$$

In particular, we consider the partition matroid $\mathcal{M}_{m,M} = (R,\mathcal{I})$ where $I \in \mathcal{I}$ if $|I\cap L| \leq 1$ for every $L \in \mathcal{L}$. The matroid rank is $m+M$. The algorithm we choose is greedy: we initialize $\mathrm{ALG} \leftarrow \emptyset$, and when an element r arrives it is selected if $\mathrm{ALG}+r \in \mathcal{I}$.

Proposition 49. *Suppose $M \geq m^2$. Then, the greedy algorithm over instance $\mathcal{M}_{m,M}$ is $(1+1/m)$ ordinal-competitive and $\Omega(m)$ intersection-competitive.*

In fact, the ordinal competitiveness holds no matter what the value of M is. We adjust the value of M in order to get a poor competitiveness for the intersection notion.

Proof of Proposition 49. The optimal base of $\mathcal{M}_{m,M}$ is the highest element of each part in $\mathcal{L}$, that is,

$$\mathrm{OPT} = \{r^1, r^2, \ldots, r^m\} \cup \{r^{jm+1} : j \in \{1,\ldots,M\}\}.$$

In particular, we have

$$|\mathrm{OPT}\cap R^k| = \begin{cases} k & \text{if } k \in \{1,\ldots,m\},\\ m+j & \text{if } k \in \{jm+1,\ldots,(j+1)m\} \text{ and } j \in \{1,\ldots,M\}.\end{cases}$$

Observe that an element $r \in Q$, with $Q \in \mathcal{L}$, is selected by the algorithm if and only if r is the first element of Q that arrives. Therefore, $\Pr(r^i \in \mathrm{ALG}) = 1$ if

$i \in \{1, \ldots, m\}$ and $\Pr(r^i \in \text{ALG}) = 1/m$ if $i \in \{m+1, \ldots, (m+1)m\}$. It follows that $\mathbb{E}[|\text{ALG} \cap R^k|] = k$ if $k \in \{1, \ldots, m\}$. If $k > m$, then

$$\mathbb{E}[|\text{ALG} \cap R^k|] = \sum_{i=1}^{m} \Pr(r^i \in \text{ALG}) + \sum_{i=m+1}^{k} \Pr(r^i \in \text{ALG}) = m + \frac{1}{m}(k-m).$$

Thus, when $k \in \{1, \ldots, m\}$, we have $|\text{OPT} \cap R^k| / \mathbb{E}[|\text{ALG} \cap R^k|] = 1$. Suppose $k = jm + r$ with $j \geq 1$ and $0 \leq r \leq m$. Then,

$$\begin{aligned} \frac{|\text{OPT} \cap R^k|}{\mathbb{E}[|\text{ALG} \cap R^k|]} &= \frac{m+j}{m + \frac{1}{m}(k-m)} \\ &= 1 + \frac{m-r}{m^2 + mj + r - m} \leq 1 + \frac{1}{m+j-1} \leq \frac{1}{m}, \end{aligned}$$

and thus the greedy algorithm is $(1+1/m)$ ordinal-competitive for $\mathcal{M}_{m,M}$. In the first inequality we used that $\phi(x) = (m-x)/(m^2 + mj + x - m)$ is a decreasing function in the interval $[0, m]$. Observe that the competitiveness result holds no matter the value of M. It remains to study the intersection competitiveness. By the observations above, we have that

$$\frac{|\text{OPT}|}{\mathbb{E}[|\text{ALG} \cap \text{OPT}|]} = \frac{m+M}{m + \frac{1}{m} \cdot M} \geq \frac{m+m^2}{m+m} \geq \frac{m}{2},$$

and so the algorithm is at least $\Omega(m)$ intersection-competitive. □

Although not mentioned explicitly in the proof, since $\Pr(r^i \in \text{ALG}) = 1/m$ for $i > m$ it follows that the algorithm is m probability competitive for $\mathcal{M}_{m,M}$, and for every M. In the following we construct an instance for which the intersection competitiveness is close to 1, but the ordinal competitiveness is poor. Let m be a positive integer and $R = \{r^1, \ldots, r^{2m-1}\}$. Consider the partition $\mathcal{L}$ given by

$$\mathcal{L} = \left\{ \{r^1, \ldots, r^m\}, \{r^{m+1}\}, \{r^{m+2}\}, \ldots, \{r^{2m-1}\} \right\}.$$

Let $\mathcal{N}_m = (R, \mathcal{I})$ be the partition matroid where $I \in \mathcal{I}$ if $|I \cap L| \leq 1$ for every $L \in \mathcal{L}$. The matroid rank is m.

Proposition 50. *The greedy algorithm over instance $\mathcal{N}_m$ is $(1 + 1/m)$ intersection-competitive and $\Omega(m)$ ordinal-competitive.*

Proof. An element $r \in Q$, with $Q \in \mathcal{L}$, is selected by the algorithm if and only if r is the first element of Q that arrives. Therefore, $\Pr(r^i \in \text{ALG}) = 1/m$ if $i \in \{1, \ldots, m\}$ and $\Pr(r^i \in \text{ALG}) = 1$ if $i \in \{m+1, \ldots, 2m-1\}$. Since $\text{OPT} = \{r^1, r^{m+1}, \ldots, r^{2m-1}\}$,

$$\frac{|\text{OPT} \cap R^1|}{\mathbb{E}[|\text{ALG} \cap R^1|]} = \frac{1}{\Pr(r^1 \in \text{ALG})} = m,$$

hence the algorithm is $\Omega(m)$ ordinal-competitive. Finally, we have

$$\frac{|\text{OPT}|}{\mathbb{E}|\text{ALG} \cap \text{OPT}|} = \frac{m}{1/m + m - 1} \leq 1 + \frac{1}{m},$$

and so the greedy algorithm is $(1 + 1/m)$ intersection-competitive over $\mathcal{N}_m$. □

Concluding Remarks

In this thesis we studied applications to scheduling and online selection problem through general frameworks. For scheduling, we considered relaxations obtained from performing *lift & project* steps. We proved that if we apply these techniques directly over the assignment formulations known for the problem, then they fail to reduce the integrality gap even when the size of the relaxations is exponentially large on the input size. All the relaxations above mentioned are symmetric with respect to machine permutations, so we go back one step in our approach, and before applying *lift & project* we break the machine symmetries. It turns out that in this way we get arbitrarily close to one integrality gaps, and the relaxations are of polynomial size. We summarize below the main message of this part.

> *Lift & project* methods are powerful tools for obtaining relaxations with good integrality gaps, but they have be applied to the *right model*. Symmetries are harmful, so, we suggest an operator that *break symmetires, lift & project*.

We believe that understanding the interaction between the action of a group over a formulation and its effect at the moment of using hierarchies should be understood better. In particular, it opens the way to combine techniques that look for symmetries in a formulation (they are not always evidently present as in the scheduling problem), break them and then strengthening the model.

In the second part of the thesis we considered the matroid secretary problem under a strong notion of competitiveness. We provide a general tool that allows to obtain $O(1)$ competitiveness for many matroid families. Furthermore, we obtained probability competitiveness for general matroids, and in a weaker notion we matched the best known for the classic weighted secretary problem.

> We develop a framework to analize algorithms by studying the size of the forbidden sets and the competitiveness attained is a function of this parameter. The smaller the size of the forbidden sets, the better the competitiveness.

In particular, the existence of an algorithm with forbidden sets of size one for *every* matroid would prove the strong MSP conjecture. It is an interesting question whether this approach can be applied to obtain good competitive ratios in non-matroidal settings. In our analysis we require an optimality condition that is necessary for matroids, but we do not know if it is sufficient.

Bibliography

[1] M. Ajtai, N. Megiddo, and O. Waarts. Improved Algorithms and Analysis for Secretary Problems and Generalizations. *SIAM Journal on Discrete Mathematics*, 14(1):1–27, 2001.

[2] M. Alekhnovich, S. Arora, and I. Tourlakis. Towards strong nonapproximability results in the Lovász-Schrijver hierarchy. In *STOC*, pages 294–303, 2005.

[3] N. Alon, Y. Azar, G. J. Woeginger, and T. Yadid. Approximation schemes for scheduling. In *SODA*, pages 493–500, 1997.

[4] N. Alon, Y. Azar, G. J. Woeginger, and T. Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1(1):55–66, 1998.

[5] S. Arora, B. Bollobás, L. Lovász, and I. Tourlakis. Proving integrality gaps without knowing the linear program. *Theory of Computing*, 2:19–51, 2006.

[6] S. Arora, S. Rao, and U. Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM*, 56(2):5, 2009.

[7] A. Atserias and E. Maneva. Sherali–adams relaxations and indistinguishability in counting logics. *SIAM Journal on Computing*, (1):112–137, 2013.

[8] P. D. Azar, R. Kleinberg, and S. M. Weinberg. Prophet inequalities with limited information. In *Proc. of SODA 2014*, pages 1358–1377, 2014.

[9] M. Babaioff, M. Dinitz, A. Gupta, N. Immorlica, and K. Talwar. Secretary Problems: Weights and Discounts. In *Proc. of SODA 2009*, pages 1245–1254, 2009.

[10] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg. A knapsack secretary problem with applications. In *Proc. of APPROX-RANDOM 2007*, volume 4627 of *LNCS*, pages 16–28, 2007.

[11] M. Babaioff, N. Immorlica, and R. Kleinberg. Matroids, secretary problems, and online mechanisms. In *Proc. of SODA 2007*, pages 434–443, 2007.

[12] N. Bansal. Approximating independent sets in sparse graphs. In *SODA*, pages 1–8, 2015.

[13] N. Bansal, A. Srinivasan, and O. Svensson. Lift-and-round to improve weighted completion time on unrelated machines. *CoRR*, abs/1511.07826, 2015.

[14] B. Barak and D. Steurer. Sum-of-squares proofs and the quest toward optimal algorithms. *arXiv preprint arXiv:1404.5236*, 2014.

[15] S. Barman, S. Umboh, S. Chawla, and D. L. Malec. Secretary problems with convex costs. In *Proc. of ICALP 2012*, volume 7391 of *LNCS*, pages 75–87, 2012.

[16] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998.

[17] M. Bateni, M. Hajiaghayi, and M. Zadimoghaddam. Submodular secretary problem and extensions. *ACM Transactions on Algorithms*, 9(4):1–23, 2013.

[18] M. Beckmann. Dynamic programming and the secretary problem. *Computers & Mathematics with Applications*, 19(11):25–28, 1990.

[19] G. Braun, J. Brown-Cohen, A. Huq, S. Pokutta, P. Raghavendra, A. Roy, B. Weitz, and D. Zink. The matching problem has no small symmetric sdp. *Mathematical Programming*, 165(2):643–662, 2017.

[20] F. T. Bruss. Sum the odds to one and stop. *The Annals of Probability*, 28(3):1384–1391, 2000.

[21] J. Buresh-Oppenheim, N. Galesi, S. Hoory, A. Magen, and T. Pitassi. Rank bounds and integrality gaps for cutting planes procedures. *Theory of Computing*, 2:65–90, 2006.

[22] S. Chakraborty and O. Lachish. Improved Competitive Ratio for the Matroid Secretary Problem. In *Proc. of SODA 2012*, pages 1702–1712, 2012.

[23] M. Charikar. On semidefinite programming relaxations for graph coloring and vertex cover. In *SODA*, pages 616–620, 2002.

[24] K. H. Cheung. Computation of the lasserre ranks of some polytopes. *Mathematics of Operations Research*, (1):88–94, 2007.

[25] E. Chlamtac, Z. Friggstad, and K. Georgiou. Lift-and-project methods for set cover and knapsack. In *Algorithms and Data Structures*, pages 256–267. 2013.

[26] E. Chlamtac and M. Tulsiani. Convex relaxations and integrality gaps. In *Handbook on semidefinite, conic and polynomial optimization*, pages 139–169. Springer, 2012.

[27] V. Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete mathematics*, pages 305–337, 1973.

[28] G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4):393–410, 1954.

[29] G. B. Dantzig. *Origins of the simplex method*. ACM, 1990.

[30] W. F. de la Vega and C. Mathieu. Linear programming relaxations of maxcut. In *SODA*, pages 53–61, 2007.

[31] M. Desrochers and F. Soumis. A column generation approach to the urban transit crew scheduling problem. *Transportation Science*, 23(1):1–13, 1989.

[32] N. B. Dimitrov and C. G. Plaxton. Competitive Weighted Matching in Transversal Matroids. *Algorithmica*, 62(1-2):333–348, 2012.

[33] M. Dinitz and G. Kortsarz. Matroid Secretary for Regular and Decomposable Matroids. *SIAM Journal on Computing*, 43(5):1807–1830, 2014.

[34] P. Dütting and R. Kleinberg. Polymatroid prophet inequalities. In *Proc. of ESA 2015*, volume 9294 of *LNCS*, pages 437–449, 2015.

[35] E. B. Dynkin. The optimum choice of the instant for stopping a Markov process. *Soviet Math. Dokl*, 4:627–629, 1963.

[36] J. Edmonds and D. Fulkerson. Transversals and matroid partition. *Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics*, 69B(3):147–153, 1965.

[37] U. Feige and R. Krauthgamer. The probable value of the Lovász–Schrijver relaxations for maximum independent set. *SIAM Journal on Computing*, 32:345–370, 2003.

[38] M. Feldman, J. Naor, and R. Schwartz. Improved competitive ratios for submodular secretary problems (extended abstract). In *APPROX-RANDOM*, volume 6845 of *LNCS*, pages 218–229, 2011.

[39] M. Feldman, O. Svensson, and R. Zenklusen. A Simple O (log log(rank)) - Competitive Algorithm for the Matroid Secretary Problem. In *Proc. of SODA 2015*, pages 1189–1201, 2015.

[40] M. Feldman and R. Zenklusen. The submodular secretary problem goes linear. In *Proc. of FOCS 2015*, pages 486–505, 2015.

[41] T. S. Ferguson. Who Solved the Secretary Problem? *Statistical Science*, 4(3):282–289, 1989.

[42] S. Fiorini, S. Massar, S. Pokutta, H. R. Tiwary, and R. De Wolf. Linear vs. semidefinite extended formulations: exponential separation and strong lower bounds. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 95–106. ACM, 2012.

[43] P. Flener, A. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, J. Pearson, and T. Walsh. Symmetry in matrix models. In *Proceedings of SymCon*, volume 1. Citeseer, 2001.

[44] P. Flener, A. M. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, J. Pearson, and T. Walsh. Breaking row and column symmetries in matrix models. In *International Conference on Principles and Practice of Constraint Programming*, pages 462–477. Springer, 2002.

[45] M. Garey and D. Johnson. "Strong" NP-completeness results: motivation, examples, and implications. *Journal of the ACM*, 25:499–508, 1978.

[46] K. Georgiou, A. Magen, T. Pitassi, and I. Tourlakis. Integrality gaps of 2-o(1) for vertex cover SDPs in the Lovász–Schrijver hierarchy. *SIAM Journal on Computing*, 39:3553–3570, 2010.

[47] J. P. Gilbert and F. Mosteller. Recognizing the Maximum of a Sequence. *Journal of the American Statistical Association*, 61(313):35, 1966.

[48] M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

[49] N. Gvozdenovic and M. Laurent. The operator ψ for the chromatic number of a graph. *SIAM Journal on Optimization*, 19:572–591, 2008.

[50] D. Hochbaum. *Approximation algorithms for NP-hard problems*. PWS Publishing Co., 1996.

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

[52] M. Hoefer and B. Kodric. Combinatorial Secretary Problems with Ordinal Information. In I. Chatzigiannakis, P. Indyk, F. Kuhn, and A. Muscholl, editors, *Proc. of ICALP 2017*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 133:1–133:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[53] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1994.

[54] R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge university press, 2012.

[55] S. Im and Y. Wang. Secretary Problems: Laminar Matroid and Interval Scheduling. In *Proc. of SODA 2011*, pages 1265–1274, 2011.

[56] P. Jaillet, J. A. Soto, and R. Zenklusen. Advances on Matroid Secretary Problems: Free Order Model and Laminar Case. In *Proc. of IPCO 2013*, volume 7801 of *LNCS*, pages 254–265, 2013.

[57] P. Jaillet, J. A. Soto, and R. Zenklusen. Advances on Matroid Secretary Problems: Free Order Model and Laminar Case (Full version). http://arxiv.org/abs/1207.1333v2, 2014.

[58] M. Janata. Matroids induced by packing subgraphs. *SIAM J. Discrete Math.*, 18(3):525–541, 2005.

[59] R. Jans. Solving lot-sizing problems on parallel identical machines using symmetry-breaking constraints. *INFORMS Journal on Computing*, 21(1):123–136, 2009.

[60] K. Jansen. An eptas for scheduling jobs on uniform processors: using an milp relaxation with a constant number of integral variables. *SIAM Journal on Discrete Mathematics*, 24(2):457–485, 2010.

[61] V. Kaibel and M. Pfetsch. Packing and partitioning orbitopes. *Mathematical Programming*, 114(1):1–36, 2008.

[62] A. Karlin, C. Mathieu, and C. Nguyen. Integrality gaps of linear and semidefinite programming relaxations for knapsack. In *IPCO*, pages 301–314. 2011.

[63] T. Kesselheim, R. Kleinberg, and R. Niazadeh. Secretary Problems with Non-Uniform Arrival Order. In *Proc. of STOC 2015*, pages 879–888, 2015.

[64] T. Kesselheim, K. Radke, A. Tönnis, and B. Vöcking. An Optimal Online Algorithm for Weighted Bipartite Matching and Extensions to Combinatorial Auctions. In *Proc. of ESA 2013*, volume 8125 of *LNCS*, pages 589–600, 2013.

[65] T. Kesselheim, K. Radke, A. Tönnis, and B. Vöcking. Primal beats dual on online packing LPs in the random-order model. In *Proc. of STOC 2014*, pages 303–312, 2014.

[66] R. Kleinberg. A Multiple-Choice Secretary Algorithm with Applications to Online Auctions. In *Proc. of SODA 2005*, pages 630–631, 2005.

[67] R. Kleinberg and S. M. Weinberg. Matroid prophet inequalities. In *Proc. of STOC 2012*, pages 123–136, 2012.

[68] N. Korula and M. Pál. Algorithms for Secretary Problems on Graphs and Hypergraphs. In *Proc. of ICALP 2009*, volume 5556 of *LNCS*, pages 508–520, 2009.

[69] A. Kurpisz, S. Leppänen, and M. Mastrolilli. A Lasserre lower bound for the min-sum single machine scheduling problem. In *Algorithms–ESA 2015*, pages 853–864. 2015.

[70] O. Lachish. O(log log Rank) Competitive Ratio for the Matroid Secretary Problem. In *Proc. of FOCS*, pages 326–335, 2014.

[71] G. Laman. On graphs and rigidity of plane skeletal structures. *Journal of Engineering Mathematics*, 4(4):331–340, 1970.

[72] J. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11:796–817, 2001.

[73] M. Laurent. A comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre relaxations for 0-1 programming. *Mathematics of Operations Research*, 28:470–496, 2003.

[74] E. Levey and T. Rothvoss. A Lasserre-based $(1+\varepsilon)$-approximation for $\mathrm{P}m|p_j = 1, \mathrm{prec}|\mathrm{C}_{\max}$. *CoRR*, abs/1509.07808, 2015.

[75] D. V. Lindley. Dynamic Programming and Decision Theory. *Applied Statistics*, 10(1):39–51, 1961.

[76] M. Loebl and S. Poljak. On matroids induced by packing subgraphs. *Journal of Combinatorial Theory, Series B*, 44(3):338–354, 1988.

[77] M. Loréa. Hypergraphes et matroïdes. *Cahiers Centre Etud. Rech. Oper*, 17:289–291, 1975.

[78] L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization*, 1:166–190, 1991.

[79] T. Ma, B. Tang, and Y. Wang. The Simulated Greedy Algorithm for Several Submodular Matroid Secretary Problems. *Theory of Computing Systems*, 58(4):681–706, 2016.

[80] F. Margot. Exploiting orbits in symmetric ilp. *Mathematical Programming*, 98(1-3):3–21, 2003.

[81] F. Margot. Symmetric ilp: Coloring and small integers. *Discrete Optimization*, 4(1):40–62, 2007.

[82] F. Margot. Symmetry in integer linear programming. In *50 Years of Integer Programming 1958-2008*, pages 647–686. Springer, 2010.

[83] A. Mehrotra and M. A. Trick. A column generation approach for graph coloring. *informs Journal on Computing*, 8(4):344–354, 1996.

[84] I. Méndez-Díaz and P. Zabala. A branch-and-cut algorithm for graph coloring. *Discrete Applied Mathematics*, 154(5):826–847, 2006.

[85] M. Mitzenmacher and E. Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[86] S. C. On, J. Lee, P. Raghavendra, and D. Steurer. Approximate constraint satisfaction requires large lp relaxations. In *FOCS*, pages 350–359. IEEE, 2013.

[87] S. Oveis Gharan and J. Vondrák. On Variants of the Matroid Secretary Problem. *Algorithmica*, 67(4):472–497, 2013.

[88] P. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical programming*, 96:293–320, 2003.

[89] T. Rothvoß. The lasserre hierarchy in approximation algorithms. *Lecture notes for the MAPSP*, 2013.

[90] T. Rothvoß. The matching polytope has exponential extension complexity. *Journal of the ACM (JACM)*, 64(6):41, 2017.

[91] A. Rubinstein. Beyond matroids: secretary problem and prophet inequality with general constraints. In *Proc. of STOC 2016*, pages 324–332, 2016.

[92] A. Rubinstein and S. Singla. Combinatorial prophet inequalities. In *Proc. of SODA 2017*, pages 1671–1687. SIAM, 2017.

[93] G. Schoenebeck, L. Trevisan, and M. Tulsiani. Tight integrality gaps for Lovász-Schrijver LP relaxations of vertex cover and max cut. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 302–310, 2007.

[94] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[95] H. Sherali and W. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3:411–430, 1990.

[96] J. A. Soto. Matroid Secretary Problem in the Random-Assignment Model. *SIAM Journal on Computing*, 42(1):178–211, 2013.

[97] I. Streinu and L. Theran. Natural realizations of sparsity matroids. *Ars Mathematica Contemporanea*, 4(1):141–151, 2011.

[98] T. S. Tay. Rigidity of multi-graphs. I. Linking rigid bodies in n-space. *Journal of Combinatorial Theory, Series B*, 36(1):95–112, 1984.

[99] P. H. Vance, C. Barnhart, E. L. Johnson, and G. L. Nemhauser. Solving binary cutting stock problems by column generation and branch-and-bound. *Computational optimization and applications*, 3(2):111–130, 1994.

[100] J. Verschae and A. Wiese. On the configuration-LP for scheduling on unrelated machines. *Journal of Scheduling*, 17:371–383, 2014.

[101] W. Whiteley. Some matroids from discrete applied geometry. *Contemporary Mathematics*, 197:171–311, 1996.

[102] D. Williamson and D. Shmoys. *The design of approximation algorithms*. Cambridge University Press, 2011.

[103] M. Yannakakis. Expressing combinatorial optimization problems by linear programs. *Journal of Computer and System Sciences*, 43(3):441–466, 1991.

[104] T. Zaslavsky. Frame Matroids and Biased Graphs. *European Journal of Combinatorics*, 15(3):303–307, 1994.

Résumé

L'optimisation convexe a été un outil puissant pour concevoir des algorithmes. Dans la pratique est largement utilisé dans des domaines tels que la recherche opérationnelle et l'apprentissage automatique, mais aussi dans de nombreux problèmes combinatoires fondamentaux, ils cèdent aux algorithmes d'approximations les mieux connues fournissant des garanties inconditionnelles sur la qualité de la solution. Dans la première partie de ce travail, nous étudions l'effet de la construction de relaxations convexes sur un problème d'emballage, basé sur l'application de méthodes de levage et de projet. Nous montrons une faiblesse de ces relaxations lorsqu'elles sont obtenues à partir des formulations naturelles de ce problème, en montrant l'impossibilité de réduire l'écart même lorsque ces relaxations sont très importantes. Nous fournissons un moyen de combiner des procédures de rupture de symétrie et des méthodes de levage et de projet pour obtenir des écarts arbitraires.

Dans la deuxième partie de cette thèse, nous étudions les problèmes de sélection en ligne, c'est-à-dire que les éléments arrivent dans le temps et nous devons en sélectionner certains irrévocablement pour répondre à certaines contraintes combinatoires, mais aussi pour maximiser la qualité de la sélection. Habituellement, cette qualité est mesurée en termes de poids, mais nous considérons une variante plus forte dans laquelle les poids ne sont pas nécessairement connus en raison de la disponibilité de l'information. Au lieu de cela, tant que nous pouvons classer les éléments, nous pouvons fournir un cadre général pour obtenir des algorithmes d'approximation avec de bons ratios compétitifs dans de nombreux contextes.

Mots Clés

Programmation linéaire, Programmation semi-définie, Ordonnancement, Selection en ligne, Algorithmes d’approximation.

Abstract

Convex optimization has been a powerful tool for designing algorithms. In practice is a widely used in areas such as operations research and machine learning, but also in many fundamental combinatorial problems they yield to the best know approximations algorithms providing unconditional guarantees over the solution quality. In the first part of this work we study the effect of constructing convex relaxations to a packing problem, based on applying lift & project methods. We exhibit a weakness of this relaxations when they are obtained from the natural formulations of this problem, by showing the impossibility of reducing the gap even when this relaxations are very large. We provide a way of combining symmetry breaking procedures and lift & project methods to obtain arbitrarily good gaps.

In the second part of this thesis we study online selection problems, that is, elements arrive over time and we have to select some of them, irrevocably, in order to meet some combinatorial constraints, but also trying to maximize the quality of the selection. Usually this quality in measured in terms of weight, but we consider a stronger variant in which weights are not necessarily known because of information availability. Instead, as long as we can rank the elements, we can provide a general framework to obtain approximation algorithms with good competitive ratios in many contexts.

Keywords

Linear Programming, Semidefinite programming, Scheduling, Online Selection, Approximation algorithms.