

# Strengths and Limitations of Linear Programming Relaxations

THÈSE N° 7948 (2017)

PRÉSENTÉE LE 27 OCTOBRE 2017  
À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS  
LABORATOIRE DE THÉORIE DU CALCUL 2  
PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

**Abbas BAZZI**

acceptée sur proposition du jury:

Prof. R. Urbanke, président du jury  
Prof. O. N. A. Svensson, directeur de thèse  
Prof. R. Zenklusen, rapporteur  
Prof. V. Kaibel, rapporteur  
Prof. M. Kapralov, rapporteur



ÉCOLE POLYTECHNIQUE  
FÉDÉRALE DE LAUSANNE

Suisse  
2017



Suffering is temporary,  
but a degree lasts forever.  
— *a T-shirt on Amazon.com*

To Hanan, Yasmina and Kareem ...



# Acknowledgements

A PhD student is considered lucky if her/his advisor is either a great researcher, a genuinely kind and understanding person, a funny and cool friend, or a competitive gym buddy... Imagine then how a student would feel if her/his advisor were all of the above! That, unfortunately, shall remain a mystery.

On an unrelated note, Ola Svensson was my PhD advisor, and I'm in trouble; having a serious non-technical discussion with/about Ola is alien to me, let alone writing it down in prose! Luckily though, I do not even need to do that. Ola summed it up in the best possible way a while back when he rightfully told me that having him as my PhD advisor was the second luckiest thing that ever happened to me.

Moreover, I would like to thank the jury members of my thesis, Prof. Volker Kaibel, Prof. Michael Kapralov, Prof. Rico Zenklusen, and the president of the jury Prof. Rüdiger Urbanke, for their careful reading of my thesis<sup>1</sup> and the insightful discussion we had during and after the private defense. It is a rather happy anomaly that scientists of their stature signed off on my graduation, and I sincerely hope it won't come back to haunt them.

No words can express my gratitude to Chantal Schneeberger; to say the least, she is the rampart standing between the members of THL2 and utter chaos. She was always there to answer any questions I had, help fix any mess that (Ashkan and) I created, and make the complex bureaucracy at EPFL seem like a walk in the park.

I would like to also thank Ioana Ivan for the opportunity that she gave me at Google NYC, and Thomas Rothvoß for hosting me for six months at the University of Washington. Should I have trouble getting along with my future bosses, then Ioana and Thomas, along with Ola, are to be blamed; these people have definitely set the bar too high.

I would also like to blame Farah and Louay for influencing my decision to pursue a PhD, and I blame Ola for (unintentionally) making "*not quitting*" seem like a no-brainer. Other culprits include Gitanes, Marlboro, Nespresso, Starbucks, Glenfiddich, Corona and Youtube; without them, the four years would have felt even longer than the couple of decades that they seemed like. Last but not least, I condemn my family and friends for being there for me when I needed to blow off some steam, thereby ensuring that I do not get (much) crazier by the day.

Finally, academia, *it's not you... it's me!*

---

<sup>1</sup>Pierre Yaacoub is the one to blame for any mistakes in the French version of the abstract; otherwise, I modestly take any credits for that part.



# Abstract

Many of the currently best-known approximation algorithms for NP-hard optimization problems are based on Linear Programming (LP) and Semi-definite Programming (SDP) relaxations. Given its power, this class of algorithms seems to contain the most favourable candidates for outperforming the current state-of-the-art approximation guarantees for NP-hard problems, for which there still exists a gap between the inapproximability results and the approximation guarantees that we know how to achieve in polynomial time. In this thesis, we address both the power and the limitations of these relaxations, as well as the connection between the shortcomings of these relaxations and the inapproximability of the underlying problem.

In the first part, we study the limitations of LP relaxations of well-known graph problems such as the Vertex Cover problem and the Independent Set problem. We prove that any small LP relaxation for the aforementioned problems, cannot have an integrality gap strictly better than 2 and  $\omega(1)$ , respectively. Furthermore, our lower bound for the Independent Set problem also holds for any SDP relaxation. Prior to our work, it was only known that such LP relaxations cannot have an integrality gap better than 1.5 for the Vertex Cover Problem, and better than 2 for the Independent Set problem.

In the second part, we study the so-called knapsack cover inequalities that are used in the current best relaxations for numerous combinatorial optimization problems of covering type. In spite of their widespread use, these inequalities yield LP relaxations of exponential size, over which it is not known how to optimize exactly in polynomial time. We address this issue and obtain LP relaxations of quasi-polynomial size that are at least as strong as that given by the knapsack cover inequalities.

In the last part, we show a close connection between structural hardness for  $k$ -partite graphs and tight inapproximability results for scheduling problems with precedence constraints. This connection is inspired by a family of integrality gap instances of a certain LP relaxation. Assuming the hardness of an optimization problem on  $k$ -partite graphs, we obtain a hardness of  $2 - \epsilon$  for the problem of minimizing the makespan for scheduling with preemption on identical parallel machines, and a super constant inapproximability for the problem of scheduling on related parallel machines. Prior to this result, it was only known that the first problem does not admit a PTAS, and the second problem is NP-hard to approximate within a factor strictly better than 2, assuming the Unique Games Conjecture.

## Acknowledgements

---

Key words: Linear Programming, Approximation Algorithms, Integrality Gap, Graph Problems, Knapsack Problem, Scheduling Problems



# Résumé

La majorité des algorithmes d'approximation dédiés à l'optimisation des problèmes de type NP-difficile sont basés sur des relaxations de type Programmation Linéaire (LP) et Programmation Semi-Définie (SDP). Cette classe d'algorithmes semble englober le nombre le plus important de candidats qui pourraient surpasser les garanties d'approximations actuelles de pointe des problèmes de type NP-difficile, pour lesquels il existe toujours un écart entre les résultats d'inapproximabilité et les garanties d'approximations qui peuvent être résolus en temps polynomial. Nous examinons dans cette thèse la puissance et les limites de ces relaxations, ainsi que la relation entre les défaillances de ces dernières et l'inapproximabilité du problème sous-jacent.

Dans un premier temps, nous analysons les limites des relaxations de type LP relatifs à des problèmes classiques du domaine de la théorie des graphes tels que le problème du transversal minimum (Vertex Cover) et le problème du stable maximum (Independent Set). Nous démontrons que n'importe quelle relaxation de type LP est incapable de réaliser un écart d'intégralité strictement inférieur à 2 et  $\omega(1)$  pour les problèmes mentionnés respectivement ci-dessus. De plus, le minorant relatif au problème du stable maximum est également applicable à n'importe quelle relaxation de type SDP. Les études précédentes avaient démontré que ces relaxations de type LP sont incapables d'avoir un écart d'intégralité inférieur à 1.5 pour le problème du transversal minimum, et inférieur à 2 pour le problème du stable maximum.

Dans la deuxième partie, nous étudions les inégalités de couverture de Sac à Dos (knapsack cover inequalities) utilisées dans les meilleures relaxations actuelles pour de nombreux problèmes d'optimisation combinatoire de couverture (combinatorial problems of covering type). Malgré leur application répandue, ces inégalités génèrent des relaxations LP de taille exponentielle pour lesquelles les optimisations en temps polynomial demeurent inconnues. Cette thèse traite ce problème et parvient à obtenir des relaxations linéaires de taille quasi-polynomiale qui sont au moins aussi puissantes que celles obtenues par les inégalités de couverture de Sac à Dos.

Dans la dernière partie nous démontrons l'existence d'un lien étroit entre la difficulté de structure (structural hardness) pour les graphes  $k$ -partites et les résultats d'inapproximabilité étroite pour des problèmes de planification avec des relations de préséances (scheduling with precedence constraints). Cette connexion est essentiellement basée sur un groupe d'instances d'écarts d'intégralité d'une certaine relaxation de type LP. En supposant qu'un problème d'optimisation sur des graphes  $k$ -partites est NP-difficile, nous

## Acknowledgements

---

obtenons une NP-difficulté égale à  $2 - \varepsilon$  pour le problème de "makespan minimization for scheduling with preemption on identical parallel machines" et une super constante d'inapproximabilité pour le problème de "scheduling on related parallel machines". Avant les résultats de cette thèse, il était uniquement connu que le premier problème n'admet pas de PTAS, et qu'il est NP-difficile d'approximer le second avec un facteur strictement inférieur à 2, compte tenu de la conjecture des jeux uniques (Unique Games Conjecture).

Mots clefs : Programmation Linéaire, Écart d'intégralité, Problèmes des Graphes, Problème de Sac à Dos, Problèmes de Planification

# Contents

<b>Abstract (English/Français)</b>	<b>iii</b>
<b>List of figures</b>	<b>xi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Approximation Algorithms . . . . .	3
1.2 Relaxation-based Approximation Algorithms . . . . .	4
1.3 Overview of our Contributions . . . . .	8
1.4 Outline of the Thesis . . . . .	10
<b>2 Preliminaries</b>	<b>13</b>
2.1 Constraint Satisfaction Problems . . . . .	13
2.2 Formulations and Computational Models . . . . .	16
2.2.1 Polytopes and Extended Formulations . . . . .	16
2.2.2 Formulation Complexity . . . . .	17
2.2.3 Uniform vs. Non-Uniform Models . . . . .	19
<b>3 Sherali-Adams Gaps for Constraint Satisfaction Problems</b>	<b>21</b>
3.1 Introduction . . . . .	21
3.2 Sherali-Adams Hierarchy for CONSTRAINT SATISFACTION PROBLEMS. . . . .	22
3.3 Unique Games. . . . .	27
3.4 Sherali-Adams Integrality Gap for 1F-CSP. . . . .	28
3.5 Sherali-Adams Integrality Gap for $\kappa$ -NOR. . . . .	38
3.5.1 Functions Over the Domain $\mathbb{Z}_q$ . . . . .	39
3.5.2 Reduction from UNIQUE GAMES to $\kappa$ -NOR. . . . .	42
3.6 LP-Hardness of 1F-CSP and $\kappa$ -NOR . . . . .	47
<b>4 LP Hardness of Vertex Cover</b>	<b>49</b>
4.1 Introduction. . . . .	49
4.2 From CONSTRAINT SATISFACTION PROBLEMS to Graphs . . . . .	54
4.2.1 Reduction to Graphs . . . . .	55
4.2.2 Reduction to Hypergraphs . . . . .	59
4.3 LP Reduction Framework . . . . .	60
4.4 LP-Hardness for Vertex Cover and Independent Set. . . . .	62

## Contents

---

4.5	LP-Hardness for $q$ -UNIFORM-VERTEX-COVER. . . . .	65
4.6	Upper bounds. . . . .	67
4.7	SDP-Hardness for Independent Set. . . . .	69
4.8	Conclusion . . . . .	70
<b>5</b>	<b>Knapsack</b> . . . . .	<b>71</b>
5.1	Introduction . . . . .	72
5.2	Preliminaries. . . . .	77
5.2.1	Polyhedral Pairs, Extended Formulations and Slack Matrices. . .	77
5.2.2	Randomized Communication Protocols. . . . .	78
5.2.3	Weighted Threshold Functions and Karchmer-Wigderson Game. . .	79
5.3	Small LP relaxation for MIN-KNAPSACK. . . . .	79
5.3.1	Overview. . . . .	79
5.3.2	The Protocol. . . . .	82
5.3.3	MIN-KNAPSACK with Polynomial (and Integer) Demand and Sizes	87
5.4	Flow-cover inequalities. . . . .	90
5.4.1	Preliminaries. . . . .	92
5.4.2	Randomized Protocol for Canonical Feasible Solutions. . . . .	93
5.4.3	Randomized Protocol for Arbitrary Feasible Solutions. . . . .	98
5.5	Algorithmic Aspects. . . . .	100
5.6	Conclusion. . . . .	104
<b>6</b>	<b>Scheduling Problems</b> . . . . .	<b>105</b>
6.1	Introduction . . . . .	105
6.2	Preliminaries . . . . .	109
6.3	Integrality Gap For $P prec, pmtn, p_j = 1 C_{max}$ . . . . .	110
6.3.1	LP Relaxation of $P prec, pmtn, p_j = 1 C_{max}$ . . . . .	111
6.3.2	Integrality Gap of the LP . . . . .	111
6.4	Hardness of $P prec, pmtn C_{max}$ assuming the UGC . . . . .	117
6.5	Structured $k$ -Partite Problem . . . . .	123
6.6	Lower Bounds for Scheduling Problems . . . . .	124
6.6.1	$P prec, pmtn C_{max}$ . . . . .	124
6.6.2	$Q prec C_{max}$ . . . . .	130
6.7	Conclusion . . . . .	134
<b>7</b>	<b>Conclusion and Future Directions</b> . . . . .	<b>137</b>
7.1	LP-lower bounds . . . . .	137
7.2	Knapsack . . . . .	138
7.3	The Scheduling Problems and the $k$ -Partite Hypothesis . . . . .	138
<b>A</b>	<b>List of Problems</b> . . . . .	<b>139</b>
	<b>Bibliography</b> . . . . .	<b>150</b>

Curriculum Vitae

151



# List of Figures

3.1	Distribution for the 1F-CSP constraints . . . . .	33
3.2	Distribution for the $\kappa$ -NOR constraints . . . . .	43
4.1	Example of an <i>FGLSS</i> graph corresponding to a MAX CUT instance. . . . .	56
5.1	Execution of the communication protocol between Alice and Bob. . . . .	84
6.1	Example of the integrality gap instance $\mathcal{S}(4)$ with 3 machines of [SCHED-LP]112	
6.2	Illustration of the integrality gap instance $\mathcal{S}(k, d)$ . . . . .	116
6.3	Bipartite Graph $G = (V, W, E)$ satisfying the YES case of Theorem 6.7. . . . .	118
6.4	Reduction from a bipartite graph $G = (V, W, E)$ to a scheduling instance $\mathcal{S}$ . . . . .	119
6.5	The structure of the scheduling instance in the YES case of Theorem 6.7. . . . .	122
6.6	$k$ -partite Graph $G = (V_1, V_2, \dots, V_k, E_1, E_2, \dots, E_{k-1})$ satisfying the YES case of Hypothesis 6.14. . . . .	125
6.7	Reduction from a 5-partite graph $G = (V_1, V_2, V_3, V_4, V_5, E_1, E_2, E_3, E_4)$ to a scheduling instance $\mathcal{S}$ . . . . .	127
6.8	An example of a scheduling instance corresponding to the YES case of Hypothesis 6.14. . . . .	129
6.9	Structure of the soundness versus completeness of $P _{\text{prec, pmtn, } p_j = 1} _{C_{\max}}$ assuming Hypothesis 6.14. . . . .	131
6.10	The reduction from a 3-partite graph to a scheduling instance $\mathcal{S}$ of $Q _{\text{prec}} _{C_{\max}}$ . . . . .	132
6.11	Structure of the soundness versus completeness of $Q _{\text{prec}} _{C_{\max}}$ assuming Hypothesis 6.14. . . . .	135





# 1 Introduction

Perhaps the most basic goal in the field of computer science is to assess how rapidly we can solve a given problem. The notion of speed in the theoretical branch of this field is measured asymptotically with respect to the size  $n$  of the input. As time is an *expensive* resource, the running time of an efficient algorithm is preferred to be at most polynomial in  $n$ .

For many optimization problems, we are able to design efficient algorithms that always return an optimal solution. In some of these cases, we can also prove that it is impossible to find a faster algorithm. This resulted in a rich literature of efficient optimization algorithms for problems that might seem to be very hard at first sight.

Nonetheless, for many other interesting problems, this goal is still out of reach due to the unresolved status of the P vs. NP question. In other words, our current understanding of computational complexity is short of designing efficient algorithms for NP-hard combinatorial optimization problems, or ruling out such an option. To cope with this, a long line of research in this area has focused on settling for a solution that is not necessarily optimal, yet provably not very far from it. This paved the way for the area of approximation algorithms, where a polynomial time algorithm is said to be  $\alpha$ -approximate if its solution is guaranteed to be at most  $\alpha$  away from the optimal value.

It has been shown, however, that even with this kind of compromise, our task is not necessarily easier in the worst case. For instance, finding an assignment that satisfies 99% of the clauses in a 3-SAT formula, or even 88% of them, is as hard as satisfying all the clauses [57]. The study of the limits of approximation for this class of problems resulted in the rich area of the hardness of approximation, where various approaches have been used to understand these limits for approximation.

For example, the design of an  $\alpha$ -approximation algorithm for certain problems, which runs in polynomial time, would (dis)prove certain complexity assumptions, such as  $P \neq NP$  or the stronger Unique Games Conjecture (UGC). Alternatively, a parallel line

## Chapter 1. Introduction

---

of research addresses the unconditional inapproximability of these problems when the underlying algorithm belongs to a certain restricted family of approximation algorithms.

In particular, many of the currently best known approximation algorithms for NP-hard optimization problems are based on Linear Programming (LP) and Semi-definite Programming (SDP) relaxations. Given its power, this class of algorithms seems to contain the most favourable candidates for outperforming the current state-of-the-art approximation guarantees for NP-hard problems for which there still exists a gap between the inapproximability results, and the approximation guarantees that we know how to achieve.

In this thesis we will address both the power and the limitations of these relaxations. On the one hand, we will focus on the limits of approximation algorithms that arise from LP relaxations for the (generalization of the) VERTEX COVER problem, and from LP and SDP relaxations for the INDEPENDENT SET problem. On the other hand, we will prove that the strongest-to-date LP formulations for certain optimization problems that required *exponential* size, can actually be approximated by *quasi-polynomial* size LP formulations. In particular, we will prove that the exponentially many knapsack cover inequalities which have been successfully used in the best known LP for many optimization problems, can in fact be approximated with a quasi-polynomial number of inequalities. Furthermore, we study the inapproximability of scheduling problems with precedence constraints. Inspired by a family of integrality gap instances that fools a powerful LP relaxation of the problem of scheduling precedence-constrained jobs on identical parallel machines with preemption, we hypothesize that a certain  $k$ -partite graphs ordering problem is hard. Assuming this hypothesis, we are able to prove tight inapproximability results for the aforementioned scheduling problem and rule out any constant factor approximation for the problem of scheduling precedence-constrained jobs on related machines. A special case of this theorem for bipartite graphs is known to hold, assuming a variant of the UGC, and it yields tight inapproximability results for two other important scheduling problems. Thus, our hypothesis seems to capture the intrinsic hardness of scheduling problems with precedence constraints.

This chapter serves as a general introduction for the remaining chapters of this thesis and it introduces the basic notions that will be repeatedly used thereafter. In particular, we give a general definition of *approximation algorithms* in Section 1.1, and then focus in Section 1.2 on those algorithms that are based on LP *relaxations*. We also define in Section 1.2 how to assess the performance of an LP relaxation in terms of its *size* and its *integrality gap*. Following that, we then summarize the contributions of this thesis in Section 1.3, and give a general outline of the remaining chapters in Section 1.4.

## 1.1 Approximation Algorithms

Let  $\mathcal{I}$  be an instance of an NP-hard optimization problem  $L$ , and let  $OPT_L(\mathcal{I})$  denote the optimal value for this instance. Assuming  $P \neq NP$ , designing a polynomial time algorithm that returns  $OPT_L(\mathcal{I})$  for every instance  $\mathcal{I}$  of  $L$  is not possible. To cope with the intractability of such problems, we design an approximation algorithm  $ALG_L$  for  $L$ , that returns a value  $ALG_L(\mathcal{I})$  for every instance  $\mathcal{I}$  of  $L$ .  $ALG_L$  is then called an  $\alpha$ -approximation algorithm for  $L$  if  $ALG_L(\mathcal{I})$  is at most a factor of  $\alpha$  away from  $OPT_L(\mathcal{I})$  for every instance  $\mathcal{I}$  of  $L$ . Formally, an  $\alpha$ -approximation algorithm is defined as follows:

**Definition 1.1.** Let  $\alpha \geq 1$  be the approximation factor. Given a *minimization* problem  $L$ , we say that  $ALG_L$  is an  $\alpha$ -approximation algorithm for  $L$ , if for every instance  $\mathcal{I} \in L$ ,  $\frac{ALG_L(\mathcal{I})}{OPT_L(\mathcal{I})} \leq \alpha$ . Alternatively an  $\alpha$ -approximation algorithm for a *maximization*<sup>1</sup> problem  $L$  should satisfy  $\frac{OPT_L(\mathcal{I})}{ALG_L(\mathcal{I})} \leq \alpha$  for every instance  $\mathcal{I} \in L$ .

*Remark 1.2.* Hereinafter, we define the various notions in terms of minimization problems for ease of presentation. These concepts can then be generalized to maximization problems in the natural way.

Given that the VERTEX COVER problem is one of the main problems that we tackle in this thesis, we will use it as a running example in the introduction to illustrate the various concepts of interest.

Formally, given a graph  $G = (V, E)$ , we say that a subset  $S \subseteq V$  of vertices is a *vertex cover* of  $G$  if every edge  $e \in E$  is covered by  $S$ , i.e., at least one of the endpoints of  $e$  is in  $S$ . Equipped with this, the VERTEX COVER problem can be defined as follows:

**Definition 1.3.** In the VERTEX COVER problem, we are given a graph  $G = (V, E)$ , and the goal is to find a minimum cardinality<sup>2</sup> set  $S \subseteq V$  such that  $S$  is a vertex cover of  $G$ .

The (decision version of the) VERTEX COVER problem is perhaps one of the most used examples of NP-complete problems, and appeared in Karp's list of 21 NP-complete problems [66]. Thus designing a polynomial time algorithm that is guaranteed to return an optimal vertex cover for any graph  $G$ , is equivalent to proving that  $P=NP$ .

However, if we are willing to settle for a 2-approximation of the VERTEX COVER problem, then an easy algorithm such as the one presented in Algorithm 1 does the job.

It is not hard to see that Algorithm 1 runs in polynomial time, and that its output  $S$  is a vertex cover. To see that it indeed returns a 2-approximation of the vertex cover (i.e., the cardinality of the returned vertex cover  $S$  is at most twice the size of the minimum

<sup>1</sup>One can equivalently define an  $\alpha$ -approximation algorithm for a maximization problem  $L$  for  $0 < \alpha \leq 1$  by requiring  $\frac{ALG_L(\mathcal{I})}{OPT_L(\mathcal{I})} \geq \alpha$

<sup>2</sup>Alternatively, if  $G$  is vertex-weighted, then the goal is to find a minimum weight vertex cover  $S$ .

---

**Algorithm 1** Greedy 2-approximation algorithm for the VERTEX COVER problem.

---

- 1:  $S \leftarrow \emptyset$ .
  - 2:  $E_S \leftarrow \emptyset$ .
  - 3:  $E' \leftarrow E$ .
  - 4: **repeat**
  - 5:   Let  $e = (u, v)$  be an arbitrary edge of  $E'$ .
  - 6:    $S \leftarrow S \cup \{u, v\}$ .
  - 7:    $E_S \leftarrow E_S \cup \{e\}$ .
  - 8:   Remove any edge incident to either  $u$  or  $v$  from  $E'$ .
  - 9: **until**  $E' = \emptyset$
  - 10: return  $S$ .
- 

cardinality vertex cover of  $G$ ), note that the edges in  $E_S$  are vertex disjoint, hence any vertex cover (and in particular the optimal vertex cover  $S^*$ ), must contain at least one endpoint of each edge in  $E_S$ . Thus  $|S^*| \geq |E_S|$ . Moreover, the size of  $S$ , the output of Algorithm 1, is exactly twice that of  $E_S$ . Combining all of these we obtain that

$$|S| = 2|E_S| \leq 2|S^*|.$$

In the next section, we present another easy 2-approximation algorithm for the VERTEX COVER problem that is based on LP relaxations, the main topic of this thesis.

## 1.2 Relaxation-based Approximation Algorithms

One strong and consistent way for designing approximation algorithms is to solve *exactly* a *relaxed* version of the problem at hand, and then translate the result to an *approximate* solution to the *original* problem.

Similarly to the previous section, we use the VERTEX COVER problem to explain this family of approximation algorithms.

**LP Relaxation for the VERTEX COVER Problem.** Given a graph  $G = (V, E)$  over  $n = |V|$  vertices, the VERTEX COVER problem can be formulated exactly as the following optimization problem:

- Minimize:**  $|S|$   
**Such that:**  $S \subseteq V$  and  $S$  is a vertex cover of  $G$ ,

or equivalently as:

$$\text{Minimize: } \sum_{v \in V} x_v \tag{VC-ILP}$$

## 1.2. Relaxation-based Approximation Algorithms

---

$$\textbf{Subject to: } x_u + x_v \geq 1 \qquad \forall e = (u, v) \in E \qquad \text{(ILP.I)}$$

$$x \in \{0, 1\}^n, \qquad \text{(ILP.II)}$$

where we think of each binary vector  $x \in \{0, 1\}^n$  as the indicator vector of some set  $S \subseteq V$ . It is not hard to see that satisfying the constraints of type (ILP.I) for every edge  $e = (u, v) \in E$ , guarantees that  $x$  is indeed an indicator vector for an actual vertex cover  $S_x \subseteq V$  of  $G$ , where  $S_x = \{v : x_v = 1\}$  is the set indicated by the binary vector  $x$ .

We refer to (VC-ILP) as the Integer Linear Programming (ILP) formulation of the VERTEX COVER problem, as the underlying objective function and the constraints are linear, and the feasible solution  $x$  are restricted to being integral (boolean in this case).

Solving ILPs in general is an NP-hard problem itself, but dropping the integrality constraints renders the problem much easier. In particular, if we *relax* the integrality constraints (ILP.II), and allow  $x$  to take fractional values instead, we get an LP that we know how to solve efficiently in polynomial time, at the expense of allowing feasible solutions  $x \in [0, 1]^n$  that do not necessarily correspond directly to actual subsets of vertices.

By relaxing the integrality constraints (ILP.II) of (VC-ILP), we get the following well-known LP relaxation of the VERTEX COVER problem:

$$\textbf{Minimize: } \sum_{v \in V} x_v \qquad \text{(VC-LP)}$$

$$\textbf{Subject to: } x_u + x_v \geq 1 \qquad \forall e = (u, v) \in E \qquad \text{(LP.I)}$$

$$0 \leq x_v \leq 1 \qquad \forall v \in V. \qquad \text{(LP.II)}$$

Since any vertex cover of  $G$  (and hence any feasible solution for (VC-ILP)) corresponds to a feasible solution of (VC-LP), it follows that the optimal value of (VC-LP) is a lower bound on the optimal cardinality vertex cover. Formally, if we let  $OPT(G)$  and  $LP(G)$  denote the cardinality of the optimal vertex cover of  $G$ , and the optimal value of (VC-LP) respectively, then we get that

$$LP(G) \leq OPT(G), \qquad \text{(1.1)}$$

for any graph  $G = (V, E)$ .

Albeit trivial, Equation 1.1 is at the essence of most LP based approximation algorithms and, in some sense, suggests a natural way of designing them, as we will see in the following section.

**LP-Based Approximation Algorithm for the VERTEX COVER Problem.** We will restrict ourselves in most of the chapters of this thesis to algorithms arising from LP and

## Chapter 1. Introduction

---

SDP relaxations. In order to quantify the complexity of a problem in this context, we resort to studying the minimum size of a relaxation that could be useful in providing a good approximation guarantee.

To this end, we measure the quality of a relaxation by its *integrality gap* that roughly translates to how well this relaxation can approximate the original problem. The *complexity* of the relaxation (also known as the *size* of the relaxation) is measured by the number of required inequalities. Hence, the task of proving unconditional inapproximability results for this model of computation boils down to proving that any relaxation that has an integrality gap of at most  $\alpha$  must have a large size.

To begin with, it is easy to see that the size of (VC-LP) is  $(|E| + 2|V|)$  since we have  $|E|$ -many inequalities of type (LP.I) and  $2|V|$ -many inequalities of type (LP.II)<sup>3</sup>.

Moreover, given any (fractional) solution  $x' \in \mathbb{R}_+^n$  of (VC-LP) of value  $\text{val}_{LP}(x') = \sum_{v \in V} x'_v$ , we can easily translate it to an actual vertex cover of cost no more than twice of  $\text{val}_{LP}(x')$ . To see this, define the operator  $\lceil \cdot \rceil : [0, 1]^n \rightarrow \{0, 1\}^n$  to be

$$\lceil z \rceil_v = \begin{cases} 1 & \text{if } z_v \geq \frac{1}{2} \\ 0 & \text{o.w.,} \end{cases} \quad (1.2)$$

for any  $z \in [0, 1]^n$ . Since  $x'$  is a feasible solution for (VC-LP), and in particular satisfies every constraint of type (LP.I), we get that for every edge  $e = (u, v) \in E$ , at least one of  $x'_u$  or  $x'_v$  is greater than or equal to  $\frac{1}{2}$ . Thus  $\lceil x' \rceil$  also satisfies all the constraints of type (LP.I). Moreover,  $\lceil x' \rceil \in \{0, 1\}^n$  has only boolean entries by construction (hence satisfies (LP.II)), so we get that  $\lceil x' \rceil$  is also a feasible integer solution for (VC-LP). Hence by viewing  $\lceil x' \rceil$  as an indicator vector of a set  $S_{\lceil x' \rceil}$  in the natural way, we get that  $S_{\lceil x' \rceil}$  is a vertex cover and

$$|S_{\lceil x' \rceil}| = \sum_{v \in V} \lceil x' \rceil_v \leq 2 \sum_{v \in V: x'_v \geq 1/2} x'_v \leq 2 \sum_{v \in V} x'_v = 2 \text{val}_{LP}(x').$$

It follows from the above reasoning and Equation (1.1) that Algorithm 2 provides a 2-approximation for the VERTEX COVER problem:

---

**Algorithm 2** LP-bases 2-approximation algorithm for the VERTEX COVER problem.

---

- 1:  $x \leftarrow$  optimal solution of (VC-LP).
  - 2:  $z \leftarrow \lceil x \rceil$ .
  - 3: return  $S_z$ .
- 

We can actually show that Algorithm 2 is in fact tight, in the sense that we *cannot* design

<sup>3</sup>Since (VC-LP) is a minimization problem, constraint (LP.II) can be replaced by  $x_v \geq 0$ , hence after removing redundant constraints, the size of the relaxation becomes  $|V| + |E|$ .

## 1.2. Relaxation-based Approximation Algorithms

---

an approximation algorithm that returns an integral solution of cost strictly better than twice the cost of (VC-LP), even if we employ a more sophisticated rounding scheme. This is due to the fact that the approximation guarantee of an LP-based algorithm cannot be better than the *integrality gap* of the underlying LP, and the integrality gap of (VC-LP) is 2. Formally, the integrality gap is defined as follows:

**Definition 1.4.** Let  $L$  be an minimization problem, and let  $LP_L$  be an LP formulation of  $L$ . For every instance  $\mathcal{I} \in L$ , let  $OPT_L(\mathcal{I})$  be the optimal value of  $\mathcal{I}$ , and  $LP_L(\mathcal{I})$  be the optimal value of the LP formulation for the instance  $\mathcal{I}$ . Then the integrality gap of  $LP_L$  is defined as

$$\sup_{\mathcal{I} \in L} \frac{OPT_L(\mathcal{I})}{LP_L(\mathcal{I})}.$$

In order to prove that the integrality gap of an LP relaxation  $LP_L$  of a minimization problem  $L$  is at least  $\alpha$ , it would then be enough to construct an instance  $\mathcal{I}_{bad}$  of  $L$  such that  $\frac{OPT_L(\mathcal{I}_{bad})}{LP_L(\mathcal{I}_{bad})} = \alpha$ .

For instance, to see that the integrality gap of (VC-LP) is 2 as the number of vertices  $n$  tends to infinity, consider  $K_n$ , the complete graph over  $n$  vertices. It is not hard to see that any subset  $S$  of vertices such that  $|S| \leq n - 2$  would leave at least 1 edge uncovered, thus any vertex cover of  $K_n$  has size at least  $n - 1$ . However,  $x = (1/2, 1/2, \dots, 1/2) \in [0, 1]^n$  is a feasible solution for (VC-LP) with cost  $\sum_{v \in V} x_v = \frac{n}{2}$ . It then follows from Definition 1.4 that the integrality gap of (VC-LP) is at least

$$\frac{OPT_{VC}(K_n)}{VC-LP(K_n)} = \frac{n-1}{n/2} = 2 - \frac{2}{n}.$$

The importance of the integrality gap of an LP stems from the fact that most LP based approximation algorithm can be seen as the (generic) Algorithm 3, where **round**(.) is any procedure with the following guarantee:

if  $x$  is a feasible solution of the LP relaxation  $LP_L$  with cost  $LP_L(x)$ , then  $z = \mathbf{round}(x)$  is an *integral* feasible solution of  $LP_L$  such that  $LP_L(z) \leq \alpha LP_L(x)$ .

It is not hard to see that Definition 1.4 implies that the integrality gap is a lower bound on  $\alpha$  of **round**(.), since  $OPT_L(\mathcal{I}) \leq LP_L(z)$  for any integral feasible solution  $z$ , hence  $\alpha \geq \frac{OPT_L(\mathcal{I})}{LP_L(x)} = \frac{OPT_L(\mathcal{I})}{LP_L(\mathcal{I})}$ .

For the VERTEX COVER problem, the **round**(.) procedure corresponds to the simple  $\lceil \cdot \rceil$  operator defined in Equation 1.2 that has a guarantee  $\alpha \leq 2$ .

In most of the chapters of this thesis, we will mainly be interested in the minimum size of an LP relaxation required to guarantee a *good* integrality gap. For example, we prove

---

**Algorithm 3** Generic LP based  $\alpha$ -approximation algorithm of problem  $L$ .

---

- 1: **Input:** Instance  $\mathcal{I}$  of  $L$ .
  - 2: **Output:** Integral solution  $x$  of  $\mathcal{I}$ .
  - 3: Let  $LP_L$  be an LP relaxation of  $L$ .
  - 4:  $x \leftarrow$  optimal solution of  $LP_L(\mathcal{I})$ .
  - 5:  $z \leftarrow \mathbf{round}(x)$ .
  - 6: **Return**  $z$ .
- 

in Chapter 4 that even sub-exponential size LPs for the VERTEX COVER problem will still have an integrality gap of 2. On the positive side, we prove in Chapter 5 that for certain other problems whose LP relaxations were known to be of exponential size because they are strengthened by the so-called knapsack cover inequalities, we can in fact obtain a quasi-polynomial size relaxation without affecting the integrality gap by much.

### 1.3 Overview of our Contributions

As it has become apparent thus far, LP relaxations constitute the main theme of this thesis. In particular, we study the following three aspects of these relaxations:

Limitations: In this context, our task is to prove LP lower bounds for a certain combinatorial problem. Specifically, this translates to statements that read as follows:

*For a combinatorial problem  $\Pi$ , any LP relaxation that has a good integrality gap, must have a very large size.*

Power: Alternatively, here we are interested in proving LP upper bounds. Statements in this context read as follows:

*Although the known LP relaxations with a small integrality gap for problem  $\Pi$  are huge, there exists substantially smaller LP relaxations for this problem with roughly the same guarantee.*

Implications: For many optimization problems, LP-based approximation algorithms are known to be the most powerful. This suggests in some cases that (a variant of) the LP integrality gap instances might also capture the intrinsic hardness of the problem in general computation models, and not only in the LP setting. Thus our final results in this setting can be simply seen as inapproximability results and they read as follows:

*Assuming certain complexity assumptions, no polynomial time algorithm can achieve a good approximation guarantee for the combinatorial problem  $\Pi$ .*

We are now ready to present an overview on the main contributions of this thesis.



**LP Lower Bounds.** In Chapter 3, we study the LP approximability of two CONSTRAINT SATISFACTION PROBLEMS which we refer to as 1F-CSP and  $\kappa$ -NOR, and are formally defined in Definitions 2.6 and 2.7 respectively. We show that any small LP cannot have a bounded integrality gap for these two problems.

Using the LP-hardness in Chapter 3, we are able to prove the following inapproximability results in Chapter 4:

1. We prove that any LP, whose size is sub-exponential, cannot have an integrality gap strictly better than 2 for the VERTEX COVER problem. This improves upon the previous known integrality gap of 1.5 [24] for same-size LPs<sup>4</sup>.
2. For the generalization of the VERTEX COVER problem of  $q$ -uniform hypergraphs, that we denote by  $q$ -UNIFORM-VERTEX-COVER, we show that any LP whose size is at most quasi-polynomial cannot have an integrality gap strictly better than  $q$  for this problem. Despite the importance of this problem, no LP-lower bound for large LPs was known for this problem except for the 1.5 LP-hardness of [24] implied by their VERTEX COVER lower bound.
3. We prove that any LP whose size is sub-exponential cannot have a bounded integrality gap for the INDEPENDENT SET problem. This improves upon the previous known integrality gap of 2 [24] for same-size LPs. We also generalize our results to any SDP approximating the INDEPENDENT SET problem; we prove that any polynomial size SDP cannot have a bounded integrality gap for this problem.

Our results in Chapters 3 and 4 are based on a joint work with Samuel Fiorini, Sebastian Pokutta and Ola Svensson, published in FOCS 2015 [13].

**LP Upper Bounds.** Initially developed for the MIN-KNAPSACK problem, the knapsack cover inequalities are used in the current best relaxations for numerous combinatorial optimization problems of covering type. In spite of their widespread use, these inequalities yield LP relaxations of exponential size, over which it is not known how to optimize exactly in polynomial time. In Chapter 5, we address this issue and obtain LP relaxations of quasi-polynomial size that are at least as strong as those given by the knapsack cover inequalities.

For the MIN-KNAPSACK cover problem, our main result can be stated formally as follows: for any  $\varepsilon > 0$ , there is a  $(1/\varepsilon)^{O(1)} n^{O(\log n)}$ -size LP relaxation with an integrality gap of at most  $2 + \varepsilon$ , where  $n$  is the number of items. Prior to this work, there was no known relaxation of subexponential size with a constant upper bound on the integrality gap.

---

<sup>4</sup>Both our results and the results of [24] prove the LP-hardness for quasi-polynomial size LPs, as both works use the results in [31] as intermediate steps. However, the sub-exponential lower bound in both can be obtained by using the latter improvement of [31] in [71] as a blackbox.

Moreover, we show that our technique can be applied to the generalization of the knapsack cover inequalities, known as the flow-cover inequalities. In particular, we show that we can also approximate this type of inequalities by a quasi-polynomial size LP.

Our results in Chapter 5 are based on a joint work with Samuel Fiorini, Ola Svensson and Sangxia Huang, published in SODA 2017 [12].

**Scheduling with Precedence Constraints.** In Chapter 6, we show a close connection between structural hardness for  $k$ -partite graphs and tight inapproximability results for scheduling problems with precedence constraints. Assuming a natural but nontrivial generalization of the bipartite structural hardness result of [8], we obtain a hardness of  $2 - \epsilon$  for the problem of minimizing the makespan for scheduling precedence-constrained jobs with preemption on identical parallel machines. This matches the best approximation guarantee for this problem [53, 47]. Prior to our work, it was only known that this problem does not admit a PTAS. This generalization from bipartite to  $k$ -partite graphs is in fact motivated by the structure of the integrality gap instances of the LP relaxation of the scheduling problem.

Assuming this same hypothesis, we also obtain a super constant inapproximability result for the problem of scheduling precedence-constrained jobs on related parallel machines, hence making progress towards settling an open question in both lists of ten open questions by Williamson and Shmoys [105], and by Schuurman and Woeginger [97]. Prior to this result, it was only known that this problem is NP-hard to approximate within a factor of 2, assuming the UGC [101].

Our results in Chapter 6 are based on a joint work with Ashkan Norouzi-Fard, published in ESA 2015 [14].

## 1.4 Outline of the Thesis

We begin in Chapter 2 by presenting the required preliminaries of this thesis. Specifically, we include for completeness a definition of CONSTRAINT SATISFACTION PROBLEMS in Section 2.1, as this class of problems will be used heavily in Chapters 3 and 4. We also define in Section 2.2 some of the main concepts that will be used throughout the thesis such as *formulation complexity*, *extension complexity* and the different models of computations in this setting.

In Chapter 3 we prove the LP lower bounds for 1F-CSP and  $K$ -NOR, our CONSTRAINT SATISFACTION PROBLEMS of interest. Equipped with this, we are able to prove in Chapter 4 LP lower bounds for the following three problems: VERTEX COVER, INDEPENDENT SET and  $q$ -UNIFORM-VERTEX-COVER. In Chapter 5, we study the size of a good extended

formulation for the knapsack problem, and prove LP upper bounds for the aforementioned problem as well as some of the related problems. Then in Chapter 6 we study the hardness of approximation of scheduling problems with precedence constraints by noting a close connection between a certain  $k$ -partite graph ordering problem, and these scheduling problems. We conclude in Chapter 7 by presenting potential research directions that can either build upon the results of this thesis, or improve the results therein.

All the chapters of this thesis are to some extent self-contained in the sense that a reader who is interested in

**LP Lower bounds,** can read Chapters 2 and 3 to understand the LP lower bounds for CONSTRAINT SATISFACTION PROBLEMS, and Chapters 2 and 4 to understand the LP lower bounds for the (generalization of the) VERTEX COVER problem and the INDEPENDENT SET problem, assuming the results of Chapter 3.

**LP Upper bounds,** can read Chapters 2 and 5 in order.

**Inapproximability results,** can read only Chapter 6.

For completeness, we also include in Appendix A definitions of the main problems that we tackle or mention in this thesis.



## 2 Preliminaries

We present in this chapter the required notions and definitions that will frequently appear throughout the thesis. Specifically, we give a formal definition of Constraint Satisfaction Problems in Section 2.1, as this class of problems will be used heavily in Chapters 3 and 4. We also define, in Section 2.2, some of the main concepts that will be used throughout the thesis such as *formulation complexity*, *extension complexity* and the different models of computations in this setting. Before we proceed, note that throughout the thesis, we use the following notations to define sets of numbers.

**Definition 2.1.** For an integer  $n \in \mathbb{N}^+$ , let  $\lceil n \rceil$  denote the following set:

$$\lceil n \rceil = \{0, 1, \dots, n-1\}.$$

**Definition 2.2.** For an integer  $n \in \mathbb{N}^+$ , let  $\lfloor n \rfloor$  denote the following set:

$$\lfloor n \rfloor = \{1, 2, \dots, n\}.$$

### 2.1 Constraint Satisfaction Problems

Before we define the class of CONSTRAINT SATISFACTION PROBLEMS, we note that in this context, the terms *constraints* and *predicates* have been used interchangeably in the literature, however for the sake of clarity, we differentiate between the two.

**Constraints vs. Predicates.** We define a predicate  $P := P_{k,R,f} : \lceil R \rceil^k \rightarrow \{0, 1\}$  by specifying its *arity*  $k$ , its (variables) *domain*  $\lceil R \rceil$  and the truth table of its *function*  $f : \lceil R \rceil^k \rightarrow \{0, 1\}$  mapping each of the  $|R|^k$  possible inputs  $x \in \lceil R \rceil^k$  to either  $P(x) = f(x) = 0$  or  $P(x) = f(x) = 1$ . For example, the well-known boolean 3-SAT predicate defined as  $(x_1 \vee x_2 \vee x_3)$  has arity  $k = 3$ , domain  $R = 2$ , and its corresponding function

$f : \{0, 1\}^3 \rightarrow \{0, 1\}$  sets:

$$f(x) = \begin{cases} 0 & \text{if } x = (x_1 = 0, x_2 = 0, x_3 = 0) \\ 1 & \text{o.w.,} \end{cases}$$

for all  $x \in \{0, 1\}^3$ . Note that for  $z \in \lceil R \rceil^k$ , and  $1 \leq i \leq k$ , we used  $z_i$  to denote the  $i$ -th entry of  $z$ .

**Fact 2.3.** *The number of distinct predicates of arity  $k$  and domain size  $R$  is  $2^{R^k}$ .*

Given a binary<sup>1</sup> predicate  $P : \{0, 1\}^k \rightarrow \{0, 1\}$ , the *free bit complexity* of  $P$  is defined to be  $\log_2(|\{z \in \{0, 1\}^k : P(z) = 1\}|)$ . For example, the MAX CUT predicate  $x_i \oplus x_j$  has a free bit complexity of *one*, since the only *two* accepting configurations are  $(x_1 = 0, x_2 = 1)$  and  $(x_1 = 1, x_2 = 0)$ . Notwithstanding that the notion of (free) bits suggests that the predicate must be binary, we abuse the notation and say that a predicate  $P : \lceil R \rceil^k \rightarrow \{0, 1\}$  over arbitrary domain  $R \geq 2$  has *zero* free bit complexity, if the number of satisfying assignment is equal to 1.

**Fact 2.4.** *The number of distinct binary predicates of arity  $k$  and free bit complexity one is  $\binom{2^k}{2}$ .*

A constraint  $C := C_{P_{k,R,f},n,S,A} : \lceil R \rceil^n \rightarrow \{0, 1\}$  in our terminology can be defined by specifying the following parameters:

1. The underlying predicate  $P := P_{k,R,f}$ .
2. The number of variables  $n$ .
3. The *ordered* subset  $S = \{i_1, \dots, i_k\} \subset [n]$  of the variables that  $C$  is *applied* to.
4. The *literals* assignment  $A \in \lceil R \rceil^k$ .

In this terminology, we define  $C_{P_{k,R,f},n,S,A}(x)$  for  $x \in \{0, 1\}^n$  as

$$C_{P_{k,R,f},n,S,L}(x) = P(x_{i_1} \ominus_R A_1, x_{i_2} \ominus_R A_2, \dots, x_{i_k} \ominus_R A_k),$$

where  $\ominus_R$  is subtraction<sup>2</sup> modulo  $R$ . In the binary case,  $A$  dictates whether the variables indexed by  $S$  appear negated in the constraints or not. For example, the boolean 3-SAT constraint  $C = (x_3 \wedge \bar{x}_5 \wedge \bar{x}_7)$  corresponds in our case to  $P$  being the 3-SAT predicate, the set  $S = \{3, 5, 7\} \subset [n]$  with  $n \geq 7$  being the number of variables, and  $A = (0, 1, 1)$  indicating that the second and third literals, according to  $S$ , appear negated in  $C$ . When the predicate  $P_{k,R,f}$ , and the number of variables  $n$  are clear from the context, we *index* the constraint  $C$  by  $S$  and  $A$ , i.e.,  $C_{S,A}$ .

<sup>1</sup>A predicate is called binary if  $R = 2$ , i.e., if its domain is  $\{0, 1\}^k$ .

<sup>2</sup>Although the addition modulo  $R$  (denoted by  $\oplus_R$ ) is the more used convention, we prefer the  $\ominus_R$  as it makes our definition for  $\kappa$ -NOR more natural.

**Fact 2.5.** *The number of distinct constraints  $C_{P_{k,R,f},n,S,L}(x)$  over  $n$  variables for a fixed predicate  $P_{k,R,f}$  is  $2^k \cdot k!$*

For the sake of presentation, we drop the subscripts of the predicates and the constraints when they are clear from the context. We also say that a constraint  $C$  is of *type*  $P$  where  $P$  is a predicate, if  $P$  is the underlying predicate of  $C$ .

**CONSTRAINT SATISFACTION PROBLEMS.** The class of CONSTRAINT SATISFACTION PROBLEMS (CSPs) captures a large variety of combinatorial problems, such as MAX CUT and MAX 3-SAT. In general, we are given a collection of  $k$ -arity predicates  $\mathcal{P} = \{P_1, \dots, P_\ell\}$  over the domain  $[R]$ , and a collection of constraints  $\mathcal{C} = \{C_1, \dots, C_m\}$  over  $n$  variables where each constraint  $C_i \in \mathcal{C}$  is of type  $P_j$  for some  $P_j \in \mathcal{P}$ . The goal in such problems is to find an assignment for  $x \in [R]^n$  in such a way as to maximize the total fraction of satisfied predicates.

The *value* of an assignment  $x \in [R]^n$  for a CSP instance  $\mathcal{J}$  is defined as

$$\text{Val}_{\mathcal{J}}(x) := \frac{1}{m} \sum_{i=1}^m C_i(x) = \mathbb{E}_{C \in \mathcal{C}} [C(x)],$$

and the optimal value of such instance  $\mathcal{J}$ , denoted by  $\text{OPT}(\mathcal{J})$  is

$$\text{OPT}(\mathcal{J}) = \max_{x \in [R]^n} \text{Val}_{\mathcal{J}}(x).$$

To be more precise, we define a CONSTRAINT SATISFACTION PROBLEM  $\Pi := \Pi_{\mathcal{P},n}$  by specifying the collection of allowed predicates  $\mathcal{P}$ , and the number of variables. An instance  $\mathcal{J}$  of  $\Pi$  is then specified by a collection  $\mathcal{C}$  of constraints where each constraints  $C \in \mathcal{C}$  is of type  $P$  for some  $P \in \mathcal{P}$ , and is over  $n$  variables. We abuse this notation for well-known problems where the predicate is implicit in the name of problem; for instance, we say MAX 3-SAT $_n$  to denote the CONSTRAINT SATISFACTION PROBLEM defined over  $n$  variables where the only predicate is the 3-SAT predicate.

For the rest of this chapter, we are mainly interested in the following two CONSTRAINT SATISFACTION PROBLEMS that we denote by 1F-CSP and K-NOR, respectively.

Recall that a binary predicate  $P$  is said to have one free bit if its number of accepting configurations is exactly 2. Thus we define our 1F-CSP $_{n,k}$  problem (where 1F stands for *one Free bit*) to be the CONSTRAINT SATISFACTION PROBLEM over  $n$  variables where the set of predicates  $\mathcal{P}$  is the collection of all one free bit predicates of arity  $k$ .

**Definition 2.6** (1F-CSP). For a fixed arity  $k$ , the 1F-CSP $_{n,k}$  problem is a CONSTRAINT SATISFACTION PROBLEM over a set of boolean variables  $\{x_1, \dots, x_n\}$ , where the collection

of predicates  $\mathcal{P}$  contains all the one free bit predicates of arity  $k$ . In other words, for every instance  $\mathcal{I}$  of 1F-CSP $_n$ , and every constraint  $C$  of  $\mathcal{I}$ ,  $C$  has only two accepting configurations out of the  $2^k$  possible ones.

Before we define the  $\kappa$ -NOR $_n$ , we review a well-know *zero* free bit predicate, the  $\kappa$ -NOR predicate that is the generalization of the NOR gate to  $k$  binary inputs. Namely,  $\kappa$ -NOR :  $\{0, 1\}^k \rightarrow \{0, 1\}$  is defined to be:

$$\kappa\text{-NOR}(x) = \begin{cases} 1 & \text{if } x=\mathbf{0} \\ 0 & \text{o.w..} \end{cases}$$

where  $\mathbf{0} \in \{0, 1\}^k$  is the all zero input. The  $\kappa$ -NOR can be generalized in the natural way to any additive group  $\mathbb{Z}_R$  (i.e., for any domain  $[R]$ ) with  $R > 2$  by setting  $\kappa\text{-NOR}(x) = 0$  for all  $x \in [R]^k \setminus \{\mathbf{0}\}$ , and  $\kappa\text{-NOR}(\mathbf{0}) = 1$ . Note that with this generalization,  $\kappa$ -NOR still has zero free bit complexity. Thus, a *constraint*  $C_{S,A}$  of type  $\kappa$ -NOR over  $n$  variables indexed by a subset of indices  $S = \{i_1, \dots, i_k\}$  and literals assignment  $A = (a_1, \dots, a_k)$  is then defined to be

$$C_{S,A}(x) = \kappa\text{-NOR}(x_{i_1} \ominus_R A_1, x_{i_2} \ominus_R A_2, \dots, x_{i_k} \ominus_R A_k),$$

or equivalently,

$$C_{S,A}(x) = 1 \quad \text{if and only if} \quad \bigwedge_{j=1}^k (x_{i_j} = a_j).$$

We are now ready to define the  $\kappa$ -NOR problem.

**Definition 2.7** ( $\kappa$ -NOR). The  $\kappa$ -NOR $_{n,R}$  problem is a CONSTRAINT SATISFACTION PROBLEM over a set of variables  $\{x_1, \dots, x_n\}$  and the domain  $[R]$ , where the only predicate allowed is the  $\kappa$ -NOR predicate, i.e.,  $\mathcal{P} = \{\kappa\text{-NOR}\}$ .

## 2.2 Formulations and Computational Models

Throughout most of the chapters of this thesis, we are mainly interested in the power and the limitations of LP relaxations. In this section, we present the notions that will appear frequently in this context.

### 2.2.1 Polytopes and Extended Formulations

For completeness, we provide in this section some definitions for basic geometric structures and notions that will come in handy for the remainder of this thesis. Namely, we define *polytopes* and *polyhedra* (singular *polyhedron*), and the *extended formulations* and the



extension complexity of a polytope.

**Definition 2.8.** A polyhedron  $Q \subseteq \mathbb{R}^n$  is the intersection of finitely many halfspaces, i.e.,

$$Q = \{y \in \mathbb{R}^n : Ax \leq b\},$$

where  $A \in \mathbb{R}^{m \times n}$ , and  $b \in \mathbb{R}^{m \times 1}$ .

**Definition 2.9.** A polytope  $P \subseteq \mathbb{R}^n$  is a bounded polyhedron. Equivalently, a polytope can be defined as the convex hull of finitely many points  $X \in \mathbb{R}^n$ .

For example, the spanning tree polytope of a graph  $G = (V, E)$  is defined as:

$$\text{CONV} \left( \chi_T \in \{0, 1\}^{|E|} : \chi_T \text{ is the indicator vector of a spanning tree } T \text{ of } G \right).$$

For the scope of this thesis, the polytopes that we discuss either arise from taking the convex hull of the feasible solutions of a certain combinatorial problem, or correspond to the feasible region of a bounded LP relaxation.

Let  $P \subseteq \mathbb{R}^n$  be a polytope and  $Q \subseteq \mathbb{R}^n$  be a polyhedron containing  $P$ . The complexity of the *polyhedral pair*  $(P, Q)$  can be measured by its extension complexity, which roughly measures how compactly we can represent a relaxation of  $P$  contained in  $Q$ . The formal definition is as follows.

**Definition 2.10.** Given a polyhedral pair  $(P, Q)$  where  $P \subseteq Q \subseteq \mathbb{R}^n$ , we say that a system  $E \leq x + F \leq y \leq g \leq, E^-x + F^-y = g^-$  in  $\mathbb{R}^{n+k}$  is an *extended formulation* of  $(P, Q)$  if the polyhedron  $R := \{x \in \mathbb{R}^n \mid \exists y \in \mathbb{R}^k : E \leq x + F \leq y \leq g \leq, E^-x + F^-y = g^- \}$  contains  $P$  and is contained in  $Q$ . The *size* of the extended formulation is the number of inequalities in the system. The *extension complexity* of  $(P, Q)$ , denoted by  $\text{xc}(P, Q)$ , is the minimum size of an extended formulation of  $(P, Q)$ .

Although the case  $P = Q$  is probably the most frequent, we will need polyhedral pairs in Chapter 5. Note that in this case (i.e., when  $P = Q$ ), we can further simplify Definition 2.10 by alternatively defining the extended formulation of a polytope  $P$  by the description of 1) a polyhedron  $R \in \mathbb{R}^{n+k}$  and 2) an affine map  $\pi : \mathbb{R}^{n+k} \rightarrow \mathbb{R}^n$ , such that  $\pi(R) = P$ .

### 2.2.2 Formulation Complexity

Before we define the formulation complexity of an optimization problem, we shall first give a general definition of the latter.

**Definition 2.11.** An *optimization problem*  $\Pi = (\mathcal{S}, \mathcal{I})$  consists of a (finite) set  $\mathcal{S}$  of feasible solutions and a set  $\mathcal{I}$  of instances. Each instance  $\mathcal{I} \in \mathcal{I}$  specifies an objective function

from  $\mathcal{S}$  to  $\mathbb{R}_+$ . We will denote this objective function by  $\text{Val}_{\mathcal{J}}$  for maximization problems, and  $\text{Cost}_{\mathcal{J}}$  for minimization problems. We let  $\text{OPT}(\mathcal{J}) := \max_{S \in \mathcal{S}} \text{Val}_{\mathcal{J}}(S)$  for a maximization problem and  $\text{OPT}(\mathcal{J}) := \min_{S \in \mathcal{S}} \text{Cost}_{\mathcal{J}}(S)$  for a minimization problem.

With this in mind we can give a general definition of the notion of an LP relaxation of an optimization problem  $\Pi$ . We deal with minimization problems first.

**Definition 2.12.** Let  $\rho \geq 1$ . A *factor- $\rho$  LP relaxation* (or  *$\rho$ -approximate LP relaxation*) for a minimization problem  $\Pi = (\mathcal{S}, \mathcal{J})$  is a linear system  $Ax \geq b$  with  $x \in \mathbb{R}^d$  together with the following realizations:

(i) **Feasible solutions** as vectors  $x^S \in \mathbb{R}^d$  for every  $S \in \mathcal{S}$  so that

$$Ax^S \geq b, \quad \text{for all } S \in \mathcal{S}.$$

(ii) **Objective functions** via affine functions  $f_{\mathcal{J}} : \mathbb{R}^d \rightarrow \mathbb{R}$  for every  $\mathcal{J} \in \mathcal{J}$  such that

$$f_{\mathcal{J}}(x^S) = \text{Cost}_{\mathcal{J}}(S), \quad \text{for all } S \in \mathcal{S}.$$

(iii) **Achieving approximation guarantee  $\rho$**  via requiring

$$\text{OPT}(\mathcal{J}) \leq \rho \text{LP}(\mathcal{J}), \quad \text{for all } \mathcal{J} \in \mathcal{J},$$

where  $\text{LP}(\mathcal{J}) := \min \{f_{\mathcal{J}}(x) \mid Ax \geq b\}$ .

Similarly, one can define factor- $\rho$  LP relaxations of a maximization problem for  $\rho \geq 1$ . In our context, the concept of a  $(c, s)$ -approximate LP relaxation will turn out to be most useful. Here,  $c$  is the *completeness* and  $s \leq c$  is the *soundness*. For a maximization problem, this corresponds to replacing condition (iii) above with

(iii)' **Achieving approximation guarantee  $(c, s)$**  via requiring

$$\text{OPT}(\mathcal{J}) \leq s \implies \text{LP}(\mathcal{J}) \leq c, \quad \text{for all } \mathcal{J} \in \mathcal{J}.$$

The *size* of an LP relaxation is the number of inequalities in  $Ax \geq b$ . We let  $\text{fc}_+(\Pi, \rho)$  denote the minimum size of a factor- $\rho$  LP relaxation for  $\Pi$ . In the terminology of [25], this is the  *$\rho$ -approximate LP formulation complexity* of  $\Pi$ . We define  $\text{fc}_+(\Pi, c, s)$  similarly.

**Relation between Formulation and Extension Complexity.** Let  $\Pi = (\mathcal{S}, \mathcal{J})$  be a maximization problem as in Definition 2.11, and consider a  $(c, s)$ -approximate LP relaxation

## 2.2. Formulations and Computational Models

---

of  $\Pi$  consisting of a linear system  $Ax \leq b$ ,  $x \in \mathbb{R}^d$ , and encodings  $X^{\mathcal{S}} = \{x^{\mathcal{S}} : \mathcal{S} \in \mathcal{S}\}$  and  $\{f_{\mathcal{J}} : \mathcal{J} \in \mathcal{J}\}$  of feasible solutions and objective functions respectively.

We now define the polyhedral pair  $(P, Q)$  as follows:

$$P = \text{conv}(X^{\mathcal{S}}), \quad Q = \left\{x \in \mathbb{R}^d : f_{\mathcal{J}}(x) \leq c, \forall \mathcal{J} \in \mathcal{J}\right\}.$$

Recall that the function  $f_{\mathcal{J}}$  are affine. Given that we started from a valid LP relaxation of  $\Pi$ , we get that  $P \subseteq Q$ . Moreover, if we define  $K$  to be

$$K = \{x \in \mathbb{R}^d : Ax \leq b\},$$

we get that  $P \subseteq K \subseteq Q$ . Thus, the formulation complexity of  $\Pi$  is the minimum size of an extended formulation over all possible linear encodings of  $\Pi$ .

### 2.2.3 Uniform vs. Non-Uniform Models

In order to be able to discuss the power and limitations of LP relaxations, one should first fix the computational model in use. In this context, we typically differentiate between two models of computation, namely the *uniform* and the *non-uniform* models.

*Remark 2.13.* We stress however that this should not be confused with the usual notion of uniform and non-uniform models in the context of circuits and Turing Machines (TM). There, a uniform TM for a problem  $\Pi$ , is a TM that is supposed to solve any instance of  $\Pi$ , irrespective of the size of the instance; whereas a non-uniform circuit for the same problem that is further parametrized by  $n$  is supposed to solve any instance of  $\Pi$  of size  $n$ . In particular, we have a different circuit for every input size  $n$ . That is, when designing a non-uniform model, we have the extra information of knowing the size of the instances that we are interested in solving. In contrast, both uniform and non-uniform models of computation for LP relaxations are in the same spirit of the usual notion of non-uniform models.

Specifically, both of these models assume extra information about the problem that we are interested in solving, yet they differ in the amount of information that they are given. The difference is perhaps best illustrated in the context of graph problems:

Non-uniform model: Given a graph  $G = (V, E)$ , write down a single set of constraints that is supposed to solve the graph problem on any induced subgraph of  $G$ . Here the constraints are allowed to adapt to the structure of the graph  $G$ , yet the induced subgraphs of interest only appear in the objective function of the LP.

Uniform model: Given  $n$ , write down a single set of constraints that is supposed to solve the graph problem on any graph of  $n$  vertices. Here the only information that is provided at the time of writing down the LP constraints is the (maximum)

number of vertices. The actual graph of interest is revealed only in the objective function.

*Remark 2.14.* Specifying an induced subgraph  $H = (V', E \cap (V' \times V'))$  of a graph  $G = (V, E)$  with  $V' \subseteq V$  is equivalent to providing a 0/1 weight function  $w \in \{0, 1\}^{|V|}$  on the vertices of  $G$ . Similarly, specifying a graph  $H = (V, E)$  such that  $|V| \leq n$ , is equivalent to providing a weight function  $w \in \{0, 1\}^{n \times n}$  of the edges of  $K_n$ , the complete graph over  $n$  vertices. Thus, an instance of the problem in the non-uniform model could be thought of as being specified by a 0/1 weight function on the vertices of the input graph  $G$ ; whereas an instance of the problem in the uniform model can be thought of as being specified by a 0/1 weight function on the edges of the complete graph  $K_n$ .

From a high level the difference between the two computational models is that the purpose of the non-uniform model is to write down a single LP formulation for all the weighted versions of a *single specific* instance, whereas the goal in the uniform model is to write a single LP for a *whole family* of instances. This implies that lower bounds on the LP formulation in the non-uniform model are stronger, because the constraints of the LP in this case are allowed to adapt to the instance. In both computational models, we do not bound the coefficients in the constraints or the time required to write down these constraints, but we are only interested in the number of inequalities defining the feasible region of the LP.

Although an optimization problem can be studied in any of the two models of computation, in some cases one of the models is more natural given the structure of the problem. For instance, the natural way to write down an LP relaxation for the VERTEX COVER problem as we saw in Section 1.2 is to assume that we know the graph of interest at the time we write down the constraints of the LP. This corresponds to the VERTEX COVER problem in the non-uniform model that we study in Chapter 4, denoted by VERTEX COVER( $G$ ) for a graph  $G = (V, E)$  and defined as follows:

VERTEX COVER( $G$ ): Given a graph  $G = (V, E)$ , VERTEX COVER( $G$ ) is a minimization problem where the goal is to find the minimum cardinality vertex cover in *any* induced subgraph  $H$  of  $G$ . In order to specify an instance  $\mathcal{I}$  of the problem, we specify a subset  $S \subseteq V$  of vertices that, in turn, corresponds to the induced subgraph  $H = (S, (S \times S) \cap E)$ . Thus in this case, the constraints of the LP relaxation would be written down *with* the knowledge of the input graph  $G = (V, E)$ , but the subset  $S$  of vertices defining an instance of the problem would be revealed only in the objective function.

Similarly, for an integer  $n$ , we can study the VERTEX COVER problem in the uniform model, denoted by VERTEX COVER( $n$ ). There, we are supposed to write down a single set of constraints that should solve the VERTEX COVER problem for any graph  $G = (V, E)$  such that  $|V| \leq n$ .

## 3 Sherali-Adams Gaps for Constraint Satisfaction Problems

In this chapter, we study the LP (in)approximability of two CONSTRAINT SATISFACTION PROBLEMS that we denote by 1F-CSP and  $\kappa$ -NOR. In particular, we prove that any Linear Program arising from linear number of rounds of the Sherali-Adams hierarchy will have an unbounded integrality gap for these two problems, as the arity goes to infinity. Given that Sherali-Adams Linear Programs are at least as powerful as any Linear Program of roughly the same size, we conclude that any Linear Program of comparable size for 1F-CSP and  $\kappa$ -NOR has an unbounded integrality gap.

Although CONSTRAINT SATISFACTION PROBLEMS are interesting in their own merit, our main motivation for studying the LP inapproximability of 1F-CSP and  $\kappa$ -NOR, is that we use them as building blocks to prove LP inapproximability results for other interesting combinatorial problems that do not fall under the class of CONSTRAINT SATISFACTION PROBLEMS. The relevance of these two CONSTRAINT SATISFACTION PROBLEMS will become apparent in Chapter 4, as they are crucial for proving LP-hardness for the VERTEX COVER problem, and its generalization to hypergraphs, that we denote by  $q$ -UNIFORM-VERTEX-COVER, as well as the INDEPENDENT SET problem.

### 3.1 Introduction

Many of the currently best-known approximation algorithms for CONSTRAINT SATISFACTION PROBLEMS are based on LP and SDP relaxations, and for some of them, the *natural* LP/SDP relaxation achieves the best possible approximation guarantee assuming certain complexity assumptions, such as  $P \neq NP$  or the Unique Games Conjecture (UGC). Despite the strength of these relaxations, their simplicity makes them prone to being *fooled* by simple problem instances. One can strengthen these relaxations and rule out such cases by adding appropriate constraints that overcome these families of fooling instances; however, it becomes infeasible to enumerate all the extreme cases. This led researchers to develop systematic ways to strengthen these relaxations, in order to see

whether adding only a small number of constraints and/or variables to the existing formulation would decrease the integrality gap.

This approach gave rise to many LP and SDP hierarchies, often referred to as lift-and-project methods: to name a few, Lovász-Schrijver [80] (LS) and Sherali-Adams [98] (SA) hierarchies can be applied to LPs to strengthen them, and similarly the Lassere/SOS [72, 74, 89] hierarchy can be applied to SDPs.

These hierarchies are performed on a round basis, where each round can either decrease the integrality gap or keep it unchanged<sup>1</sup>. It is known that, if the original variables of the relaxation are intended to be boolean, then performing  $n$  rounds in the boolean case decreases the integrality gap all the way down to 1. However, solving a relaxation resulting from  $r$ -rounds requires a running time that is exponential in  $r$ , hence as  $r$  approaches  $n$ , this becomes as expensive as running a brute force approach for finding an optimal solution. For our purposes, we are only interested in the Sherali-Adams hierarchy, as it turns out, that it actually captures the strongest LPs for CONSTRAINT SATISFACTION PROBLEMS [31, 71].

**Outline of the Chapter.** In this chapter, we prove that any small size LP for 1F-CSP and  $\kappa$ -NOR will have an unbounded integrality gap. As both of our problems of interest are CONSTRAINT SATISFACTION PROBLEMS, it will be enough to construct integrality gap instances that can fool  $r$ -rounds Sherali-Adams relaxation for  $r$  large enough. By virtue of [31, 71], this would rule out *any* good LP relaxation, and hence yields our result. Moreover, these SA integrality gap instances will be constructed via reductions from the integrality gap instances of the UNIQUE GAMES problem.

The remainder of this chapter will be organized as follows: we begin in Section 3.2 by giving a tailor made definition of Sherali-Adams relaxations to constraint CONSTRAINT SATISFACTION PROBLEMS. We then define the UNIQUE GAMES problem in Section 3.3 and state some of the known results that will come in handy for proving our main result. Equipped with these, we prove our main results for the 1F-CSP and  $\kappa$ -NOR problems in Sections 3.4 and 3.5, respectively.

### 3.2 Sherali-Adams Hierarchy for CONSTRAINT SATISFACTION PROBLEMS.

We define the canonical relaxation for CONSTRAINT SATISFACTION PROBLEMS as it is obtained by  $r$ -rounds of the Sherali-Adams (SA) hierarchy. We follow the notation as in e.g., [50]. For completeness we also describe in Section 3.2 why this relaxation is equivalent to the one obtained by applying the original definition of SA as a reformulation-

---

<sup>1</sup>In the language of SOS, we alternatively use *degree* instead of *round*.

### 3.2. Sherali-Adams Hierarchy for CONSTRAINT SATISFACTION PROBLEMS.

linearization technique on a binary program.

Consider any CSP defined over  $n$  variables  $x_1, \dots, x_n \in [R]$ , with a set of  $m$  constraints  $\mathcal{C} = \{C_1, \dots, C_m\}$  where the arity of each constraint is at most  $k$ . Let  $S_i = S_{C_i}$  denote the set of variables that  $C_i$  depends on. The  $r$ -rounds SA relaxation of this CSP has a variable  $X_{(S,\alpha)}$  for each  $S \subseteq [n], \alpha \in [R]^S$  with  $|S| \leq r$ . The intuition is that  $X_{(S,\alpha)}$  models the indicator variable whether the variables in  $S$  are assigned the values in  $\alpha$ . The  $r$ -rounds SA relaxation with  $r \geq k$  is now

$$\begin{aligned} \max \quad & \frac{1}{m} \sum_{i=1}^m \sum_{\alpha \in [R]^{S_i}} C_i(\alpha) \cdot X_{(S_i,\alpha)} \\ \text{s.t.} \quad & \sum_{u \in [R]} X_{(S \cup \{j\}, \alpha \circ u)} = X_{(S,\alpha)} \quad \forall S \subseteq [n] : |S| < r, \alpha \in [R]^S, j \in [n] \setminus S, \\ & X_{(S,\alpha)} \geq 0 \quad \forall S \subseteq [n] : |S| \leq r, \alpha \in [R]^S, \\ & X_{(\emptyset, \emptyset)} = 1. \end{aligned} \quad (3.1)$$

Here we used the notation  $(S \cup \{j\}, \alpha \circ u)$  to extend the assignment  $\alpha$  to assign  $u$  to the variable indexed by  $j$ . Note that the first set of constraints say that the variables should indicate a consistent assignment.

Instead of dealing with the constraints of the Sherali-Adams LP relaxation directly, it is simpler to view each solution of the Sherali-Adams LP as a consistent collection of local distributions over partial assignments.

Suppose that for every set  $S \subseteq [n]$  with  $|S| \leq r$ , we are given a local distribution  $\mathcal{D}(S)$  over  $[R]^S$ . We say that these distributions are *consistent* if for all  $S' \subseteq S \subseteq [n]$  with  $|S'| \leq r$ , the marginal distribution induced on  $[R]^{S'}$  by  $\mathcal{D}(S)$  coincides with that of  $\mathcal{D}(S')$ .

The equivalence between SA solutions and consistent collections of local distributions basically follows from the definition of (3.1) and is also used in [32] and [31]. More specifically, we have

**Lemma 3.1** (Lemma 1 in [50]). *If  $\{\mathcal{D}(S)\}_{S \subseteq [n]: |S| \leq r}$  is a consistent collection of local distributions then*

$$X_{(S,\alpha)} = \mathbb{P}_{\mathcal{D}(S)}[\alpha]$$

*is a feasible solution to (3.1).*

Moreover, we have the other direction.

**Lemma 3.2.** *Consider a feasible solution  $(X_{(S,\alpha)})_{S \subseteq [n]: |S| \leq r, \alpha \in [R]^S}$  to (3.1). For each  $S \subseteq [n]$*

### Chapter 3. Sherali-Adams Gaps for Constraint Satisfaction Problems

---

with  $|S| \leq r$ , define

$$\mathbb{P}_{\mathcal{D}(S)}[\alpha] = X_{(S,\alpha)} \quad \text{for each } \alpha \in [R]^S.$$

Then  $(\mathcal{D}(S))_{S \subseteq [n]: |S| \leq r}$  forms a consistent collection of local distributions.

*Proof.* Note that, for each  $S \subseteq [n]$  with  $|S| \leq r$ ,  $\mathcal{D}(S)$  is indeed a distribution because by the equality constraints of (3.1)

$$\sum_{\alpha \in [R]^S} \mathbb{P}_{\mathcal{D}(S)}[\alpha] = \sum_{\alpha \in [R]^S} X_{(S,\alpha)} = \sum_{\alpha' \in [R]^{S'}} X_{(S',\alpha')} = X_{(\emptyset,\emptyset)} = 1,$$

where  $S' \subseteq S$  is arbitrary; and moreover  $\mathbb{P}_{\mathcal{D}(S)}[\alpha] = X_{(S,\alpha)} \geq 0$ . Similarly we have, again by the equality constraints of (3.1), that for each  $S' \subseteq S$  and  $\alpha' \in [R]^{S'}$

$$\mathbb{P}_{\mathcal{D}(S')}[\alpha'] = X_{(S',\alpha')} = \sum_{\alpha'' \in [R]^{S \setminus S'}} X_{(S,\alpha' \circ \alpha'')} = \sum_{\alpha'' \in [R]^{S \setminus S'}} \mathbb{P}_{\mathcal{D}(S)}[\alpha' \circ \alpha''],$$

so the local distributions are consistent.  $\square$

When a SA solution  $(X_{(S,\alpha)})$  is viewed as consistent collection  $\{\mathcal{D}(S)\}$  of local distributions, the value of the SA solution can be computed as

$$\frac{1}{m} \sum_{i=1}^m \sum_{\alpha \in [R]^{S_i}} C_i(\alpha) \cdot X_{(S_i,\alpha)} = \mathbb{E}_{C \in \mathcal{C}} \left[ \mathbb{P}_{\alpha \sim \mathcal{D}(S_C)}[\alpha \text{ satisfies } C] \right],$$

where  $S_C$  is the support of constraint  $C$ , and the expectation is taken over the uniform distribution on the set of constraints  $\mathcal{C}$ .

**Sherali-Adams as Local Distributions.** For completeness, we give the general definition of the  $r$ -rounds SA tightening of a given LP, and then we show that for CSPs the obtained relaxation is equivalent to (3.1).

Consider the following Binary Linear Program for  $c \in \mathbb{R}^n$ ,  $A \in \mathbb{R}^{m \times n}$  and  $b \in \mathbb{R}^{m \times 1}$ :

$$\begin{aligned} \max \quad & \sum_{i=1}^n c_i x_i \\ \text{s.t.} \quad & Ax \leq b \\ & x \in \{0, 1\}^n. \end{aligned}$$

By replacing the integrality constraint with  $0 \leq x \leq 1$ , we get an LP relaxation.

Sherali and Adams [99] proposed a systematic way for tightening such relaxations, by



### 3.2. Sherali-Adams Hierarchy for CONSTRAINT SATISFACTION PROBLEMS.

reformulating them in a higher dimensional space. Formally speaking, the  $r$ -rounds SA relaxation is obtained by multiplying each base inequality  $\sum_{j=1}^n A_{ij}x_j \leq b_i$  and  $0 \leq x_j \leq 1$  by  $\prod_{s \in S} x_s \prod_{t \in T} (1 - x_t)$  for all disjoint  $S, T \subseteq [n]$  such that  $|S \cup T| < r$ . This gives the following set of polynomial inequalities for each such pair  $S$  and  $T$ :

$$\left( \sum_{j \in [n]} A_{ij}x_j \right) \prod_{s \in S} x_s \prod_{t \in T} (1 - x_t) \leq b_i \prod_{s \in S} x_s \prod_{t \in T} (1 - x_t) \quad \forall i \in [m],$$

$$0 \leq x_j \prod_{s \in S} x_s \prod_{t \in T} (1 - x_t) \leq 1 \quad \forall j \in [n].$$

These constraints are then linearized by first expanding (using  $x_i^2 = x_i$ , and thus  $x_i(1 - x_i) = 0$ ), and then replacing each monomial  $\prod_{i \in H} x_i$  by a new variable  $y_H$ , where  $H \subseteq [n]$  is a set of size at most  $r$ . Naturally, we set  $y_\emptyset := 1$ . This gives us the following linear program, referred to as the  $r$ -rounds SA relaxation:

$$\begin{aligned} \max \quad & \sum_{i=1}^n c_i y_{\{i\}} \\ \text{s.t.} \quad & \sum_{H \subseteq T} (-1)^{|H|} \left( \sum_{j \in [n]} A_{ij} y_{H \cup \{j\}} \right) \leq b_i \sum_{H \subseteq T} (-1)^{|H|} y_{H \cup S} \quad \forall i \in [m], S, T, \\ & 0 \leq \sum_{H \subseteq T} (-1)^{|H|} y_{H \cup S \cup \{j\}} \leq 1 \quad \forall j \in [n], \forall S, T, \\ & y_\emptyset = 1 \end{aligned}$$

where in the first two constraint we take  $S, T \subseteq [n]$  with  $S \cap T = \emptyset$  and  $|S \cup T| < r$ .

One could go back to the original space by letting  $x_i = y_{\{i\}}$  and projecting onto the  $x$ , however we will refrain from doing that, in order to be able to write objective functions that are not linear but degree- $k$  polynomials, as is natural in the context of CSPs of arity  $k$ . Since we need to do  $k$  rounds of SA before even being able to write the objective function as a linear function, it makes more sense to work in higher dimensional space.

For CONSTRAINT SATISFACTION PROBLEMS, the canonical  $r$ -rounds SA relaxation is defined as follows. Consider any CSP defined over  $n$  variables  $x_1, \dots, x_n \in [R]$ , with  $m$  constraints  $\mathcal{C} = \{C_1, \dots, C_m\}$  where the arity of each constraint is at most  $k$ . For each  $j \in [n]$  and  $u \in [R]$ , we introduce a binary variable  $x(j, u)$ , meant to be the indicator of  $x_j = u$ . Using these variables, the set of feasible assignments can naturally be formulated as

$$\begin{aligned} \sum_{u \in [R]} x(j, u) &= 1 \quad \forall j \in [n], \\ x(j, u) &\in \{0, 1\} \quad \forall j \in [n], u \in [R]. \end{aligned}$$

### Chapter 3. Sherali-Adams Gaps for Constraint Satisfaction Problems

If we relax the integrality constraints by, for each  $j \in [n]$ ,  $u \in [R]$ , replacing  $x(j, u) \in \{0, 1\}$  by  $x(j, u) \geq 0$  (we omit the upper bounds of the form  $x(j, u) \leq 1$  as they are already implied by the other constraints) then we obtain the following constraints for the  $r$ -rounds SA relaxation :

$$\sum_{H \subseteq T} (-1)^{|H|} \sum_{u \in [R]} y_{H \cup S \cup \{(j, u)\}} = \sum_{H \subseteq T} (-1)^{|H|} y_{H \cup S} \quad \forall j \in [n], S, T,$$

$$\sum_{H \subseteq T} (-1)^{|H|} y_{H \cup S \cup \{(j, u)\}} \geq 0 \quad \forall (j, u) \in [n] \times [R], S, T,$$

where we take  $S, T \subseteq [n] \times [R]$  with  $S \cap T = \emptyset$  and  $|S \cup T| < r$ .

To simplify the above description, we observe that we only need the constraints for which  $T = \emptyset$ .

**Claim 3.3.** *All the above constraints are implied by the subset of constraints for which  $T = \emptyset$ .*

*Proof.* The equality constraints are easy to verify since  $\sum_{u \in [R]} y_{S \cup \{(j, u)\}} = y_S$  for all  $S \subseteq [n] \times [R]$  with  $|S| < r$  implies

$$\sum_{S \subseteq H \subseteq S \cup T} (-1)^{|H \cap T|} \sum_{u \in [R]} y_{H \cup \{(j, u)\}} = \sum_{S \subseteq H \subseteq S \cup T} (-1)^{|S \cap T|} y_H.$$

Now consider the inequalities. If we let  $T = \{(j_1, u_1), (j_2, u_2), \dots, (j_\ell, u_\ell)\}$  then by the above equalities

$$\begin{aligned} & \sum_{H \subseteq T} (-1)^{|H|} y_{H \cup S \cup \{(j, u)\}} \\ &= \sum_{H \subseteq T \setminus \{(j_1, u_1)\}} (-1)^{|H|} y_{H \cup S \cup \{(j, u)\}} - \sum_{H \subseteq T \setminus \{(j_1, u_1)\}} (-1)^{|H|} y_{H \cup S \cup \{(j, u), (j_1, u_1)\}} \\ &= \sum_{u'_1 \in [R]: u'_1 \neq u_1} \sum_{H \subseteq T \setminus \{(j_1, u_1)\}} (-1)^{|H|} y_{H \cup S \cup \{(j, u), (j_1, u'_1)\}} \\ & \quad \vdots \\ &= \sum_{u'_t \in [R]: u'_t \neq u_t} \dots \sum_{u'_1 \in [R]: u'_1 \neq u_1} y_{S \cup \{(j, u), (j_1, u'_1), \dots, (j_t, u'_t)\}}. \end{aligned}$$

Hence, we have also that all the inequalities hold if they hold for those with  $T = \emptyset$  and  $S$  such that  $|S| < r$ .  $\square$

By the above claim, the constraints of the canonical  $r$ -rounds SA relaxation of the CSP

can be simplified to:

$$\begin{aligned} \sum_{u \in [R]} y_{S \cup \{(j,u)\}} &= y_S & \forall j \in [n], S \subseteq [n] \times [R] : |S| < r, \\ y_{S \cup \{(j,u)\}} &\geq 0 & \forall (j,u) \in [n] \times [R], S \subseteq [n] \times [R] : |S| < r. \end{aligned}$$

To see that this is equivalent to (3.1) observe first that  $y_S = 0$  if  $\{(j,u'), (j,u'')\} \subseteq S$ . Indeed, by the partition constraint, we have

$$\sum_{u \in R} y_{\{(j,u'), (j,u'')\} \cup \{(j,u)\}} = y_{\{(j,u'), (j,u'')\}},$$

which implies the constraint  $2y_{\{(j,u'), (j,u'')\}} \leq y_{\{(j,u'), (j,u'')\}}$ . This in turn (together with the non-negativity) implies that  $y_{\{(j,u'), (j,u'')\}} = 0$ . Therefore, by again using the partition constraint, we have  $y_S = 0$  whenever  $\{(j,u'), (j,u'')\} \subseteq S$  and hence we can discard variables of this type. We now obtain the formulation (3.1) by using variables of type  $X_{(\{j_1, \dots, j_t\}, \{u_1, \dots, u_t\})}$  instead of  $y_{\{(j_1, u_1), (j_2, u_2), \dots, (j_t, u_t)\}}$ . The objective function can be linearized, provided that the number of rounds is at least the arity of the CSP, that is  $r \geq k$ , so that variables for sets of cardinality  $k$  are available.

The SA integrality gap instances for the CONSTRAINT SATISFACTION PROBLEMS of interest will be constructed via a reduction from UNIQUE GAMES. Namely, starting from SA integrality gap instances for UNIQUE GAMES, we present a polynomial time construction of SA integrality gap instances for both 1F-CSP and  $k$ -NOR. Before we proceed with our main result, we briefly define the UNIQUE GAMES problem, and state known results about this problem that we use in the remainder of this chapter.

### 3.3 Unique Games.

The UNIQUE GAMES problem is defined as follows:

**Definition 3.4.** A UNIQUE GAMES instance  $\mathcal{U} = (G, [R], \Pi)$  is defined by a graph  $G = (V, E)$  over a vertex set  $V$  and edge set  $E$ , where every edge  $uv \in E$  is associated with a bijection map  $\pi_{u,v} \in \Pi$  such that  $\pi_{u,v} : [R] \mapsto [R]$  (we set  $\pi_{v,u} := \pi_{u,v}^{-1}$ ). Here,  $[R]$  is known as the label set. The goal is to find a labeling  $\Lambda : V \mapsto [R]$  that maximizes the number of satisfied edges, where an edge  $uv$  is satisfied by  $\Lambda$  if  $\pi_{u,v}(\Lambda(u)) = \Lambda(v)$ .

The following very influential conjecture, known as the UNIQUE GAMES conjecture, is due to Khot [67].

**Conjecture 3.5.** For any  $\zeta, \delta > 0$ , there exists a sufficiently large constant  $R = R(\zeta, \delta)$  such that the following promise problem is NP-hard. Given a UNIQUE GAMES instance  $\mathcal{U} = (G, [R], \Pi)$ , distinguish between the following two cases:

### Chapter 3. Sherali-Adams Gaps for Constraint Satisfaction Problems

---

1. *Completeness:* There exists a labeling  $\Lambda$  that satisfies at least  $(1 - \zeta)$ -fraction of the edges.
2. *Soundness:* No labeling satisfies more than  $\delta$ -fraction of the edges.

We remark that the above conjecture has several equivalent formulations via fairly standard transformations. In particular, one can assume that the graph  $G$  is bipartite and regular [68].

The starting point of our reduction are the following Sherali-Adams integrality gap instances for the UNIQUE GAMES problem. Note that UNIQUE GAMES are CONSTRAINT SATISFACTION PROBLEMS and hence here and in the following, we are concerned with the standard application of the Sherali-Adams hierarchy to CSPs.

**Theorem 3.6** ([32]). *Fix a label size  $R = 2^\ell$ , a real  $\delta \in (0, 1)$  and let  $\Delta := 2\lceil C(R/\delta)^2 \rceil$  (for a sufficiently large constant  $C$ ). Then for every positive  $\varepsilon$  there exists  $\kappa > 0$  depending on  $\varepsilon$  and  $\Delta$  such that for every sufficiently large  $n$  there exists an instance of UNIQUE GAMES on a  $\Delta$ -regular  $n$ -vertex graph  $G = (V, E)$  so that:*

1. *The value of the optimal solution is at most  $\frac{1}{R} \cdot (1 + \delta)$ .*
2. *There exists a solution to the LP relaxation obtained after  $r = n^\kappa$  rounds of the Sherali-Adams relaxation of value  $1 - \varepsilon$ .*

### 3.4 Sherali-Adams Integrality Gap for 1F-CSP.

In this section we establish Sherali-Adams integrality gaps for 1F-CSP and by virtue of [31, 71] this extends to general LPs. The proof uses the idea of [32] to perform a reduction between problems that preserves the Sherali-Adams integrality gap.

Specifically, we show that the reduction by Bansal and Khot [9] from the UNIQUE GAMES problem to 1F-CSP also provides a large Sherali-Adams integrality gap for 1F-CSP, assuming that we start with a Sherali-Adams integrality gap instance of UNIQUE GAMES. As large Sherali-Adams integrality gap instances of UNIQUE GAMES were given in [32], this implies the aforementioned integrality gap of 1F-CSP.

We first describe the reduction from UNIQUE GAMES to 1F-CSP that follows the construction in [9]. We then show that it also preserves the Sherali-Adams integrality gap.

**The Test  $F_{\varepsilon,t}$ .** Before we proceed with our reduction from UNIQUE GAMES to 1F-CSP, we stress the fact that our reduction is essentially the same as the one free bit test  $F_{\varepsilon,t}$  in [9], but casted in the language of CONSTRAINT SATISFACTION PROBLEMS. For

### 3.4. Sherali-Adams Integrality Gap for 1F-CSP.

completeness, we give below a brief overview of the intuition behind the test. For a more thorough discussion, see the paper by Bansal and Khot [9] in which they design this test.

The notion of *tests* generally arises in the context of *Probabilistically Checkable Proofs* (PCP), where a *verifier* is typically given an instance  $\mathcal{I}$  of a *hard* problem  $\mathfrak{J}$ , and a *proof*  $\pi$  that this instance satisfies some properties  $P$ . A test for  $\mathfrak{J}$  is then a randomized procedure in which the verifier is only allowed to read a small number of bits from  $\pi$ , and is supposed to:

1. (*Completeness:* ) accept a correct proof claiming that an instance  $\mathcal{I} \in \mathfrak{J}$  satisfies  $P$ , if  $\mathcal{I}$  indeed (almost) satisfies  $P$ .
2. (*Soundness:* ) reject any proof claiming that an instance  $\mathcal{I} \in \mathfrak{J}$  satisfies  $P$ , if  $\mathcal{I}$  is in fact far from satisfying  $P$ .

For our purposes, the hard problem that we start from is the UNIQUE GAMES problem, and the property that we need to check is if a given UNIQUE GAMES instance  $\mathcal{U} = (G, [R], \Pi)$  over a bipartite graph  $G = (V, W, E)$  has a labeling<sup>2</sup>  $\Lambda : W \mapsto [R]$  that satisfies *almost*<sup>3</sup> all the edges  $e \in E$  of  $\mathcal{U}$ . The proof in this case would then be an encoding of the labels assigned to the vertices in  $W$  according to the (almost) satisfying assignment  $\Lambda$ . For the sake of the presentation, we assume in this high level description that we want to check whether  $\mathcal{U}$  has a labeling that satisfies *all* the edges.

One of the encoding schemes that has been successfully used in the context of PCP is the so-called *long code* encoding. A long code encoding  $\mathcal{L}(\ell) \in \{0, 1\}^{2^R}$  of a label  $\ell \in [R]$ , is a  $2^R$  bits string corresponding to the truth table of the dictatorship function  $f_\ell : \{0, 1\}^R \mapsto \{0, 1\}$ , where  $f_\ell(x) = x_\ell$  for every  $x \in \{0, 1\}^R$ . Thus the long code encoding of a label is a binary vector of size  $2^R$  in which half of the bits are 1, and in total, the length<sup>4</sup> of the proof is  $|W| \cdot 2^R$ . Equivalently, each vertex  $w \in W$  has a corresponding function  $f^{(w)} : \{0, 1\}^R \mapsto \{0, 1\}$  whose truth table is written down in the proof, and ideally this function  $f^{(w)}$  is a dictatorship function of some label  $l_w \in [R]$ .

From a high level, the test  $F_{\epsilon, t}$  of [9] is inspired by the following procedure. The verifier picks a vertex  $v \in V$  uniformly at random, and a sequence of  $t$  neighbors  $w_1, \dots, w_t$  of  $v$  randomly and independently from the neighborhood of  $v$ , where  $t$  is a constant,

<sup>2</sup>Note that since the constraints in the UNIQUE GAMES problem are bijections, a labeling  $\Lambda : W \mapsto [R]$  can be greedily extended to a labeling  $\Lambda' : W \cup V \mapsto [R]$  that maximizes the number of satisfied edges over all possible labelings that are consistent with  $\Lambda$  on  $W$ .

<sup>3</sup>Note that if there exists a labeling that satisfies *all* the edges, then the problem is easy and such a labeling can be found in polynomial time.

<sup>4</sup>For technical reasons, the length of long code encoding of a label  $\ell \in [R]$  in the written proof is in fact  $2^{R-1}$ , and hence the overall length of the proof is  $|W| \cdot 2^{R-1}$ . That is, the long code is forced to satisfy the *folding* property that guarantees that  $f(x) = (1 - f(\bar{x}))$  for all  $x \in \{0, 1\}^R$ , a property that the dictatorship function naturally satisfies.

and queries the (encoding of the) labels of  $\{w_1, \dots, w_t\}$  from the proof. It then accepts if the labels were convincing, i.e., the labels assigned to  $\{w_1, \dots, w_t\}$  satisfy the edges  $v w_1, \dots, v w_t$  simultaneously under the permutations  $\pi_{v, w_1}, \dots, \pi_{v, w_t}$ . Thus ideally, we would want the verifier to accept if and only if

$$\pi_{v, w_1}^{-1}(\Lambda(w_1)) = \dots = \pi_{v, w_t}^{-1}(\Lambda(w_t)) = \Lambda(v).$$

The test as it is described above can be naively implemented in such a way that the verifier reads  $t \cdot 2^R$  bits from the proof (i.e,  $t$  chunks of  $2^R$  bits each, where each chunk corresponds to the truth table of a function  $f : \{0, 1\}^R \mapsto \{0, 1\}$ ), accepts satisfiable UNIQUE GAMES instances, and rejects any instance that is far from being satisfiable with high probability. In particular, the verifier should first verify that for every  $i = 1, \dots, t$ , the  $2^R$  bits corresponding to the vertex  $w_i$  *actually* correspond to a long code encoding of some label  $\ell_{w_i} \in [R]$ , and then verify that these labels simultaneously satisfy the  $t$  edges  $v w_1, \dots, v w_t$ .

Viewing the read bits of the proof as truth tables of functions  $f^{(w_1)}, \dots, f^{(w_t)}$ , the above test translates to checking whether:

1. *Step 1:* For every  $i = 1, \dots, t$ ,  $f^{(w_i)}$  is a dictatorship function.
2. *Step 2:* For every  $x \in \{0, 1\}^R$ ,

$$f^{(w_1)} \circ \pi_{v, w_1}(x) = \dots = f^{(w_t)} \circ \pi_{v, w_t}(x),$$

where for an input  $x \in \{0, 1\}^R$ , a function  $f : \{0, 1\}^R \mapsto \{0, 1\}$ , and a bijection  $\pi : [R] \mapsto [R]$ , we define  $\pi(x)$  and the function  $f \circ \pi$  as  $\pi(x) = (x_{\pi(1)}, \dots, x_{\pi(R)})$ , and  $f \circ \pi(x) = f(\pi(x))$ , respectively.

Note that if for some  $i = 1, \dots, t$ ,  $f^{(w_i)}$  is indeed a dictatorship function of some label  $\ell_{w_i} \in [R]$ , then  $f^{(w_i)} \circ \pi_{v, w_i}$  is also a dictatorship function of the image  $\pi_{v, w_i}^{-1}(\ell_{w_i}) \in [R]$  of the label  $\ell_{w_i}$ . Moreover, if the proof indeed encodes a labeling  $\Lambda$  that satisfies all the edges in  $\mathcal{U}$ , we get in this case that  $\pi_{v, w_1}^{-1}(\ell_{w_1}) = \dots = \pi_{v, w_t}^{-1}(\ell_{w_t}) = \Lambda(v)$ . Thus, checking Steps 1 and 2 of the test is equivalent to checking whether the functions  $f^{(w_1)} \circ \pi_{v, w_1}, \dots, f^{(w_t)} \circ \pi_{v, w_t}$  are dictatorship functions of the *same* label  $\Lambda(v) \in [R]$ .

The test  $F_{\epsilon, t}$  of Bansal and Khot [9] does this *t-wise dictatorship test* while reading a smaller number of bits and only one so-called free bit. In particular, their test implies that it is basically enough to only read  $t \cdot 2^{\epsilon R + 1}$  many bits from the proof, while maintaining a near perfect completeness and an arbitrarily low soundness. This is done using the so-called *sub-cube test*  $F_\epsilon$  which is based on the following observation:

1. For  $x \in \{0, 1\}^R$ , and a subset  $S \subset [R]$  such that  $|S| = \epsilon R$ , define the sub-cubes  $C_{x, S}$

### 3.4. Sherali-Adams Integrality Gap for 1F-CSP.

and  $C_{\bar{x},S}$  by fixing all the coordinates outside  $S$  according to  $x$  ( $\bar{x}$  respectively), and allowing the coordinate inside  $S$  to take any value (See Equation 3.2 for an exact definition of the sub-cube).

2. Now if a function  $f : \{0,1\}^R \mapsto \{0,1\}$  is indeed a dictatorship function of some index  $\ell \in [R]$  such that  $\ell \notin S$ , then  $f$  is identical on the sub-cube  $C_{x,S}$  and it is identical on the sub-cube  $C_{\bar{x},S}$ . In particular, we should get  $f(z) = z_\ell = x_\ell$  for all  $z \in C_{x,S}$ , and  $f(z) = z_\ell = \bar{x}_\ell$  for all  $z \in C_{\bar{x},S}$ .

Note that a sub-cube  $C_{x,S}$  for  $x \in \{0,1\}^R$  and  $S \subset [R]$  such that  $|S| = \varepsilon R$ , contains  $2^{\varepsilon R}$ -many points, thus in total the above test  $F_\varepsilon$  queries  $2^{\varepsilon R+1}$  many bits from the truth table to check (with high confidence) whether a *single* function is a dictator of some coordinate  $\ell \in [R]$ .

The test  $F_{\varepsilon,t}$  that we describe below can be thought of as running  $F_\varepsilon$   $t$ -many times in parallel on the functions  $f^{(w_1)} \circ \pi_{v,w_1}, \dots, f^{(w_t)} \circ \pi_{v,w_t}$ , and accepting if all these function are dictatorship functions of the same coordinate.

Moreover, another property of the test  $F_{\varepsilon,t}$  that is perhaps the most important to us is the following: once the verifier decides which  $t \cdot 2^{\varepsilon R+1}$  bits to read from the proof, there are exactly *two* assignments for these bits that can make him accept (out of the  $2^{t \cdot 2^{\varepsilon R+1}}$  possible ones). Thus, if we think of the test that the verifier performs on these  $t \cdot 2^{\varepsilon R+1}$  bits as a binary predicate  $P : \{0,1\}^{t \cdot 2^{\varepsilon R+1}} \rightarrow \{0,1\}$ , then this predicate  $P$  has a one free bit complexity in the language of Section 2.1.

To achieve this, the test  $F_{\varepsilon,t}$  does the following (for a completeness parameter  $\varepsilon > 0$  and large enough integer  $t$ ):

1. Pick a vertex  $v \in V$  uniformly at random, and pick  $t$  vertices  $w_1, \dots, w_t$  randomly and independently from the neighborhood of  $v$ . Let  $f^{(w_1)}, \dots, f^{(w_t)} : \{0,1\}^R \rightarrow \{0,1\}$  be the functions corresponding to the vertices  $w_1, \dots, w_t$  whose truth tables are written down in the proof.
2. Pick  $x \in \{0,1\}^R$  at random.
3. Pick random indices  $i_1, \dots, i_m$  from  $[R]$  where  $m = \varepsilon R$ , and let  $S = \{i_1, \dots, i_m\}$  be the set of those indices.
4. Define the sub-cubes:

$$\begin{aligned} C_{x,S} &= \{z \in \{0,1\}^R : z_j = x_j \ \forall j \notin S\} \\ C_{\bar{x},S} &= \{z \in \{0,1\}^R : z_j = \bar{x}_j \ \forall j \notin S\}. \end{aligned} \tag{3.2}$$

5. Accept if and only if for some  $b \in \{0, 1\}$ , we have that for every  $i = 1, \dots, t$ ,

$$\begin{aligned} f^{(w_i)}(\pi_{u,w_i}(z)) &= b & \forall z \in C_{x,S} \\ f^{(w_i)}(\pi_{u,w_i}(z)) &= b \oplus 1 & \forall z \in C_{\bar{x},S}. \end{aligned}$$

Note that if the starting UNIQUE GAMES instance indeed had a satisfying assignment  $\Lambda : W \cup V \mapsto [R]$ , and the functions  $f^{(w_1)}, \dots, f^{(w_t)}$  were actually the corresponding dictatorship functions to their labels according to  $\Lambda$ , then we get that for every  $i = 1, \dots, t$  and for every  $z \in \{0, 1\}^R$ , we have

$$f^{(w_i)}(\pi_{u,w_i}(z)) = f_{\Lambda(w_i)}(\pi_{u,w_i}(z)) = f_{\pi_{u,w_i}^{-1}(\Lambda(w_i))}(z) = z_{\pi_{u,w_i}^{-1}(\Lambda(w_i))} = z_{\Lambda(v)}.$$

This says that the test  $F_{\varepsilon,T}$  will always accept satisfiable UNIQUE GAMES instances, unless  $\Lambda(v)$  is in the randomly chosen set  $S$  of indices in Step 3, which happens with probability  $\varepsilon$ . Moreover, Bansal and Khot showed that it is in fact robust enough to reject with high probability the instances that are far from being satisfiable (See Lemma 3.7).

We are now ready to present our reduction from UNIQUE GAMES to 1F-CSP, which basically relies on casting the test  $F_{\varepsilon,t}$  as a CONSTRAINT SATISFACTION PROBLEM.

**Reduction.** Let  $\mathcal{U} = (G, [R], \Pi)$  be a UNIQUE GAMES instance over a regular bipartite graph  $G = (V, W, E)$ . Given instance  $\mathcal{U}$ , we construct an instance  $\mathcal{S}$  of 1F-CSP. The reduction has two parameters: the completeness parameter  $\varepsilon > 0$  and an integer  $t$ , where  $\varepsilon$  is chosen such that  $\varepsilon R$  is an integer (taking  $\varepsilon = 2^{-q}$  for some integer  $q \geq 0$  guarantees this) and  $t$  is sufficiently large depending on  $\varepsilon$  and the desired soundness  $\eta$  (see Lemma 3.7).

The resulting 1F-CSP instance  $\mathcal{S}$  will be defined over  $2^R|W|$  variables and  $c|V|$  constraints, where  $c := c(R, \varepsilon, t, \Delta)$  is a function of the degree  $\Delta$  of the UNIQUE GAMES instance, and the constants  $R, t$  and  $\varepsilon$ .<sup>5</sup> For our purposes, the UNIQUE GAMES integrality gap instance that we start from has constant degree  $\Delta$ , and hence  $c$  is a constant.

Inspired by the previously described test  $F_{\varepsilon,t}$ , the variables of the 1F-CSP instance  $\mathcal{S}$  correspond to the  $2^R$  bits (of the long code) encoding the labels of each vertex of the UNIQUE GAMES instance we start from, and the constraints correspond to all possible tests that the verifier might perform according to the random choice of  $v$ , the random neighbors  $w_1, \dots, w_t$  and the random subset of bits read by the verifier. Instead of actually enumerating all possible constraints, we give a distribution of constraints which is the same as the distribution over the test predicates of  $F_{\varepsilon,t}$ .

We refer to the variables of  $\mathcal{S}$  as follows: it has a binary variable  $\langle w, x \rangle$  for each  $w \in W$

---

<sup>5</sup>More precisely  $c(R, \varepsilon, t, \Delta)$  is exponential in the constants  $R, t$  and  $\varepsilon$ , and polynomial in  $\Delta$



### 3.4. Sherali-Adams Integrality Gap for 1F-CSP.

and  $x \in \{0, 1\}^R$ .<sup>6</sup> For further reference, we let  $\text{Var}(\mathcal{F})$  denote the set of variables of  $\mathcal{F}$ . The constraints of  $\mathcal{F}$  are picked according to the distribution in Figure 3.1.

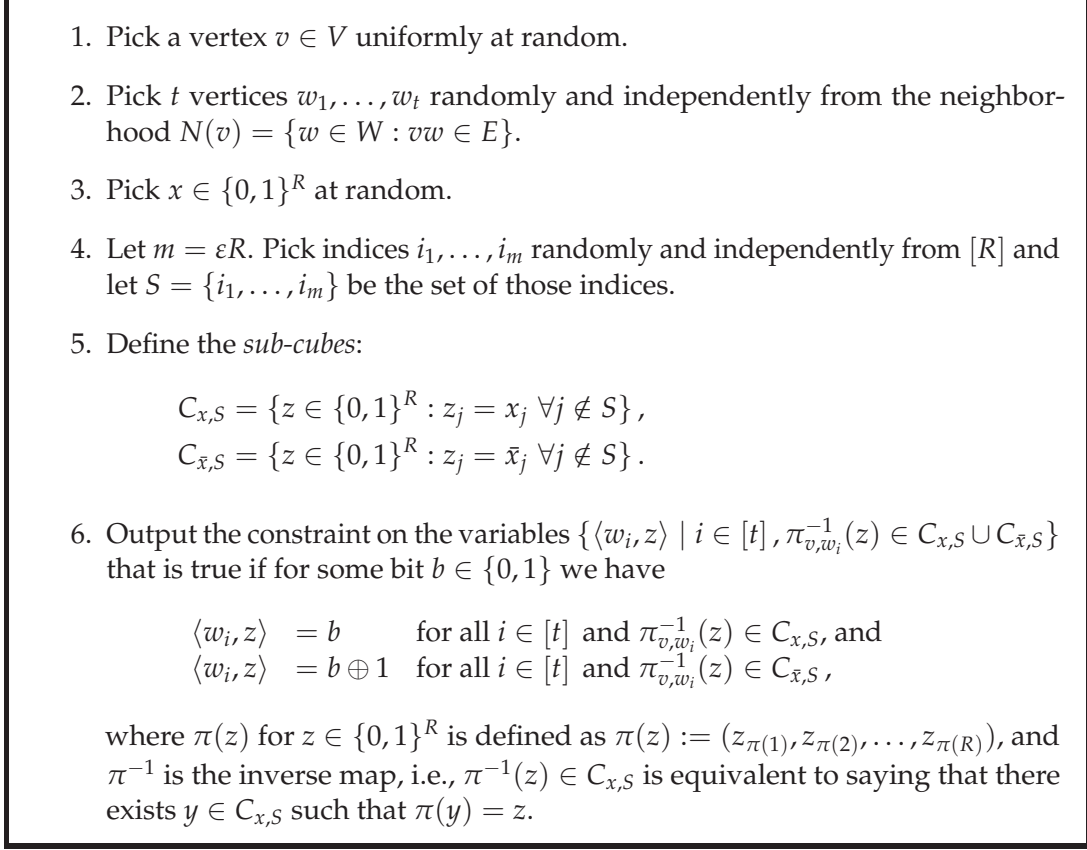


Figure 3.1 – Distribution for the 1F-CSP constraints

It is crucial to observe that our distribution over the constraints exploits the locality of a UNIQUE GAMES solution. To see this, assume we performed the first two steps of Figure 3.1 and have thus far fixed a vertex  $v \in V$  and  $t$  neighbors  $w_1, \dots, w_t$ , and let  $\mathcal{C}_{v,w_1,\dots,w_t}$  denote the set of all possible constraints resulting from steps 3-4 (i.e., for all possible  $x \in \{0, 1\}^R$  and  $S \subseteq [R]$  of size  $\varepsilon R$ ). We will argue that if there exists a local assignment of labels for  $\{v, w_1, \dots, w_t\}$  that satisfies the edges  $vw_1, \dots, vw_t$ , then we can derive a local assignment for the variables  $\{\langle w, x \rangle : w \in \{w_1, \dots, w_t\} \text{ and } x \in \{0, 1\}^R\}$  that satisfies at least  $1 - \varepsilon$  fraction of the constraints in  $\mathcal{C}_{v,w_1,\dots,w_t}$ . This essentially follows from the completeness analysis of [9], and is formalized in Claim 3.9. This allows us to convert a *good* Sherali-Adams solution of the starting UNIQUE GAMES  $\mathcal{U}$ , to a *good* Sherali-Adams solution of the resulting 1F-CSP Instance  $\mathcal{F}$ . Moreover, in order to show that  $\mathcal{F}$  is a Sherali-Adams integrality gap instance for the 1F-CSP problem, we need to

<sup>6</sup> $\langle w, x \rangle$  should be interpreted as the long-code for  $\Lambda(w)$  evaluated at  $x \in \{0, 1\}^R$ .

### Chapter 3. Sherali-Adams Gaps for Constraint Satisfaction Problems

---

show that  $\text{OPT}(\mathcal{F})$  is *small*. This follows from the soundness analysis of [9], where it was shown that:

**Lemma 3.7** (soundness). *For any  $\varepsilon, \eta > 0$  there exists an integer  $t$  so that  $\text{OPT}(\mathcal{F}) \leq \eta$  if  $\text{OPT}(\mathcal{U}) \leq \delta$  where  $\delta > 0$  is a constant that only depends on  $\varepsilon, \eta$  and  $t$ .*

The above says that if we start with a UNIQUE GAMES instance  $\mathcal{U}$  with a small optimum then we also get a 1F-CSP instance  $\mathcal{F}$  of small optimum (assuming that the parameters of the reduction are set correctly). In [9], Bansal and Khot also proved the following completeness: if  $\text{OPT}(\mathcal{U}) \geq 1 - \zeta$ , then  $\text{OPT}(\mathcal{F}) \geq 1 - \zeta t - \varepsilon$ . However, we need the stronger statement: if  $\mathcal{U}$  has a Sherali-Adams solution of large value, then so does  $\mathcal{F}$ . The following lemma states this more formally, showing that we can transform a SA solution to the UNIQUE GAMES instance  $\mathcal{U}$  into a SA solution to the 1F-CSP instance  $\mathcal{F}$  of roughly the same value.

**Lemma 3.8.** *Let  $\{\mu(S) \mid S \subseteq V \cup W, |S| \leq r\}$  be a consistent collection of local distributions defining a solution to the  $r$ -rounds Sherali-Adams relaxation of the regular bipartite UNIQUE GAMES instance  $\mathcal{U}$ . Then we can define a consistent collection of local distributions  $\{\sigma(S) \mid S \subseteq \text{Var}(\mathcal{F}), |S| \leq r\}$  defining a solution to the  $r$ -rounds Sherali-Adams relaxation of the 1F-CSP instance  $\mathcal{F}$  so that*

$$\mathbb{E}_{C \in \mathcal{C}} \left[ \mathbb{P}_{\alpha \sim \sigma(S_C)} [\alpha \text{ satisfies } C] \right] \geq (1 - \varepsilon) \left( 1 - t \cdot \mathbb{E}_{vw \in E} \left[ \mathbb{P}_{(\Lambda(v), \Lambda(w)) \sim \mu(\{v, w\})} [\Lambda(v) \neq \pi_{w,v}(\Lambda(w))] \right] \right),$$

where  $t$  and  $\varepsilon$  are the parameters of the reduction, and  $\sigma(S_C)$  is the distribution over the set of variables in the support  $S_C$  of constraint  $C$ .

We remark that the above lemma says that we can transform a SA solution to the UNIQUE GAMES instance  $\mathcal{U}$  of value close to 1, into a SA solution to the 1F-CSP instance  $\mathcal{F}$  of value also close to 1.

*Proof of Lemma 3.8.* Let  $\{\mu(S) \mid S \subseteq V \cup W, |S| \leq r\}$  be a solution to the  $r$ -rounds SA relaxation of the UNIQUE GAMES instance  $\mathcal{U}$ , and recall that  $\mathcal{F}$  is the 1F-CSP instance obtained from applying the reduction. We will now use the collection of consistent local distributions of the UNIQUE GAMES instance, to construct another collection of consistent local distributions for the variables in  $\text{Var}(\mathcal{F})$ .

For every set  $S \subseteq \text{Var}(\mathcal{F})$  such that  $|S| \leq r$ , let  $T_S \subseteq W$  be the subset of vertices in the UNIQUE GAMES instance defined as follows:

$$T_S = \{w \in W : \langle w, x \rangle \in S\}. \quad (3.3)$$

We construct  $\sigma(S)$  from  $\mu(T_S)$  in the following manner. Given a labeling  $\Lambda_{T_S}$  for the vertices in  $T_S$  drawn from  $\mu(T_S)$ , define an assignment  $\alpha_S$  for the variables in  $S$  as follows: for a variable  $\langle w, x \rangle \in S$ , let  $\ell = \Lambda_{T_S}(w)$  be the label of  $w$  according to  $\Lambda_{T_S}$ . Then the new

### 3.4. Sherali-Adams Integrality Gap for 1F-CSP.

assignment  $\alpha_S$  sets  $\alpha_S(\langle w, x \rangle) := x_\ell$ .<sup>7</sup> The aforementioned procedure defines a family  $\{\sigma(S)\}_{S \subseteq \text{Var}(\mathcal{J}), |S| \leq r}$  of local distributions for the variables of the 1F-CSP instance  $\mathcal{J}$ .

To check that these local distributions are consistent, take any  $S' \subseteq S \subseteq \text{Var}(\mathcal{J})$  with  $|S| \leq r$ , and denote by  $T_{S'} \subseteq T_S$  their corresponding set of vertices as in (3.3). We know that  $\mu(T_S)$  and  $\mu(T_{S'})$  agree on  $T_{S'}$  since the distributions  $\{\mu(S)\}$  defines a feasible Sherali-Adams solution for  $\mathcal{U}$ , and hence by our construction, the local distributions  $\sigma(S)$  and  $\sigma(S')$  agree on  $S'$ . Combining all of these together, we get that  $\{\sigma(S) \mid S \subseteq \text{Var}(\mathcal{J}), |S| \leq r\}$  defines a feasible solution for the  $r$ -round Sherali-Adams relaxation of the 1F-CSP instance  $\mathcal{J}$ .

It remains to bound the value of this feasible solution, i.e.,

$$\mathbb{E}_{C \in \mathcal{C}} \left[ \mathbb{P}_{\alpha \sim \sigma(S_C)} [\alpha \text{ satisfies } C] \right]. \quad (3.4)$$

In what follows, we denote by  $\psi(\cdot)$  the operator mapping a labeling of the vertices in  $T_S$  to an assignment for the variables in  $S$ , i.e.,  $\psi(\Lambda_{T_S}) = \alpha_S$ .

First note that a constraint  $C \in \mathcal{C}$  of the 1F-CSP instance  $\mathcal{J}$  is defined by the choice of the vertex  $v \in V$ , the sequence of  $t$  neighbors  $\mathcal{W}_v = (w_1, \dots, w_t)$ , the random  $x \in \{0, 1\}^R$ , and the random set  $S \subseteq [R]$  of size  $\epsilon R$ . We refer to such a constraint  $C$  as  $C(v, \mathcal{W}_v, x, S)$ . Thus we can rewrite (3.4) as

$$\mathbb{E}_{v, w_1, \dots, w_t} \left[ \mathbb{P}_{\Lambda \sim \mu(\{v, w_1, \dots, w_t\}), x, S} [\psi(\Lambda) \text{ satisfies } C(v, \mathcal{W}_v, x, S)] \right]. \quad (3.5)$$

Recall that the assignment  $\psi(\Lambda)$  for the variables  $\{\langle w, z \rangle : w \in \mathcal{W}_v \text{ and } z \in \{0, 1\}^R\}$  is derived from the labeling of the vertices in  $\mathcal{W}_v$  according to  $\Lambda$ . It was shown in [9] that if  $\Lambda$  satisfies the edges  $vw_1, \dots, vw_t$  simultaneously, then  $\psi(\Lambda)$  satisfies  $C(v, \mathcal{W}_v, x, S)$  with *high probability*. This is formalized in the following claim.

**Claim 3.9.** *If  $\Lambda$  satisfies  $vw_1, \dots, vw_t$  simultaneously, then  $\psi(\Lambda)$  satisfies  $C(v, \mathcal{W}_v, x, S)$  with probability at least  $1 - \epsilon$ . Moreover, if we additionally have that  $\Lambda(v) \notin S$ , then  $\psi(\Lambda)$  always satisfies  $C(v, \mathcal{W}_v, x, S)$ .*

It now follows from Claim 3.9 that for the assignment  $\psi(\Lambda)$  to satisfy the constraint  $C(v, \mathcal{W}_v, x, S)$ , it is sufficient that the following two conditions hold *simultaneously*:

1. the labeling  $\Lambda$  satisfies the edges  $vw_1, \dots, vw_t$ ;
2. the label of  $v$  according to  $\Lambda$  lies outside the set  $S$ .

<sup>7</sup>Because  $\langle w, x \rangle$  is supposed to be the dictator function of the  $\ell$ th coordinate evaluated at  $x$ , this is only the correct way to set the bit  $\langle w, x \rangle$ .

### Chapter 3. Sherali-Adams Gaps for Constraint Satisfaction Problems

Equipped with this, we can use conditioning to lower-bound the probability inside the expectation in (3.5) by a product of two probabilities, where the first is

$$\mathbb{P}_{\Lambda \sim \mu(\{v, w_1, \dots, w_t\}), x, S} [\psi(\Lambda) \text{ satisfies } C(v, \mathcal{W}_v, x, S) \mid \Lambda \text{ satisfies } vw_1, \dots, vw_t], \quad (3.6)$$

and the second is

$$\mathbb{P}_{\Lambda \sim \mu(\{v, w_1, \dots, w_t\})} [\Lambda \text{ satisfies } vw_1, \dots, vw_t].$$

Thus using Claim 3.9, we get

$$\begin{aligned} & \mathbb{E}_{C \in \mathcal{C}} \left[ \mathbb{P}_{\alpha \sim \sigma(S_C)} [\alpha \text{ satisfies } C] \right] \\ & \geq (1 - \varepsilon) \cdot \mathbb{E}_{v, w_1, \dots, w_t} \left[ \mathbb{P}_{\Lambda \sim \mu(\{v, w_1, \dots, w_t\})} [\Lambda \text{ satisfies } vw_1, \dots, vw_t] \right] \\ & \geq (1 - \varepsilon) \left( 1 - \sum_{i=1}^t \mathbb{E}_{v, w_1, \dots, w_t} \left[ \mathbb{P}_{\Lambda \sim \mu(\{v, w_1, \dots, w_t\})} [\Lambda \text{ does not satisfy } vw_i] \right] \right) \end{aligned} \quad (3.7)$$

$$= (1 - \varepsilon) \left( 1 - \sum_{i=1}^t \mathbb{E}_{v, w_1, \dots, w_t} \left[ \mathbb{P}_{\Lambda \sim \mu(\{v, w_i\})} [\Lambda \text{ does not satisfy } vw_i] \right] \right) \quad (3.8)$$

$$= (1 - \varepsilon) \cdot \left( 1 - t \cdot \mathbb{E}_{v, w} \left[ \mathbb{P}_{\Lambda \sim \mu(\{v, w\})} [\Lambda \text{ does not satisfy } vw] \right] \right), \quad (3.9)$$

where (3.7) follows from the union bound, and (3.8) is due to the fact that the local distributions of the UNIQUE GAMES labeling are consistent, and hence agree on  $\{v, w_i\}$ . Note that the only difference between what we have proved thus far and the statement of the lemma, is that the expectation in (3.9) is taken over a random vertex  $v$  and a random vertex  $w \in N(v)$ , and not random edges. However, our UNIQUE GAMES instance we start from is regular, so picking a vertex  $v$  at random and then a random neighbor  $w \in N(v)$ , is equivalent to picking an edge at random from  $E$ . This concludes the proof.  $\square$

Combining Theorem 3.6 with Lemmata 3.7 and 3.8, we get the following Corollary.

**Corollary 3.10.** *For every  $\varepsilon, \eta > 0$ , there exist an arity  $k$  and a real  $\kappa > 0$  depending on  $\varepsilon$  and  $\eta$  such that for every sufficiently large  $n$  there exists an instance of 1F-CSP of arity  $k$  over  $n$  variables, so that*

1. *The value of the optimal solution is at most  $\eta$ .*
2. *There exists a solution to the LP relaxation obtained after  $r = n^\kappa$  rounds of the Sherali-Adams relaxation of value at least  $1 - \varepsilon$ .*

*Proof.* Let  $\mathcal{U} = (G, [R], \Pi)$  be a  $\Delta$ -regular UNIQUE GAMES instance of Theorem 3.6

### 3.4. Sherali-Adams Integrality Gap for 1F-CSP.

that is  $\delta/4$ -satisfied with an  $n_G^{2k}$ -rounds Sherali-Adams solution of value  $1 - \zeta$ , where  $n_G = \Theta(n/R)$  is the number of vertices in  $G$  (if needed, we pad the instance by adding a few more dummy variables to get exactly  $n$  variables). Note that  $G = (V, E)$  is not necessarily bipartite, and our starting instance of the reduction is bipartite. To circumvent this obstacle, we construct a new bipartite UNIQUE GAMES instance  $\mathcal{U}'$  from  $\mathcal{U}$  that is  $\delta$ -satisfied with a Sherali-Adams solution of the same value, i.e.,  $1 - \zeta$ . We will later use this new instance to construct our 1F-CSP instance over  $n$  variables that satisfies the properties in the statement of the corollary.

In what follows we think of  $\delta, \zeta$  and  $R$  as functions of  $\epsilon$  and  $\eta$ , and hence fixing the latter two parameters enables us to fix the constant  $t$  of Lemma 3.7, and the constant degree  $\Delta$  of Theorem 3.6. The aforementioned parameters are then sufficient to provide us with the constant arity  $k$  of the 1F-CSP instance, along with the number of its corresponding variables and constraints, that is linear in  $n_G$ .

We now construct the new UNIQUE GAMES instance  $\mathcal{U}'$  over a graph  $G' = (V_1, V_2, E')$  and the label set  $[R]$  from  $\mathcal{U}$  in the following manner:

- Each vertex  $v \in V$  in the original graph is represented by two vertices  $v_1, v_2$ , such that  $v_1 \in V_1$  and  $v_2 \in V_2$ .
- Each edge  $e = uv \in E$  is represented by two edges  $e_1 = u_1v_2$  and  $e_2 = u_2v_1$  in  $E'$ . The bijection maps  $\pi_{u_1, v_2}$  and  $\pi_{u_2, v_1}$  are the same as  $\pi_{u, v}$ .

Note that  $G'$  is bipartite by construction, and since  $G$  is  $\Delta$ -regular, we get that  $G'$  is also  $\Delta$ -regular.

We claim that no labeling  $\Lambda' : V_1 \cup V_2 \mapsto [R]$  can satisfy more than  $\delta$  fraction of the edges in  $\mathcal{U}'$ . Indeed, assume towards contradiction that there exists a labeling  $\Lambda' : V_1 \cup V_2 \mapsto [R]$  that satisfies at least  $\delta$  fraction of the edges. We will derive a labeling  $\Lambda : V \mapsto [R]$  that satisfies at least  $\delta/4$  fraction of the edges in  $\mathcal{U}$  as follows:

For every vertex  $v \in V$ , let  $v_1 \in V_1$  and  $v_2 \in V_2$  be its representative vertices in  $G'$ . Define  $\Lambda(v)$  to be either  $\Lambda'(v_1)$  or  $\Lambda'(v_2)$  with equal probability.

Assume that at least one edge of  $e_1 = u_1v_2$  and  $e_2 = u_2v_1$  is satisfied by  $\Lambda'$ , then the edge  $e = uv \in E$  is satisfied with probability at least  $1/8$ , and hence the expected fraction of satisfied edges in  $\mathcal{U}$  by  $\Lambda$  is at least  $\delta/4$ .

Moreover, we can extend the  $r$ -rounds Sherali-Adams solution of  $\mathcal{U}$   $\{\mathcal{D}(S)\}_{S \subseteq V; |S| \leq r}$ , to a  $r$ -rounds Sherali-Adams solution  $\{\mathcal{D}'(S)\}_{S \subseteq V_1 \cup V_2; |S| \leq r}$  for  $\mathcal{U}'$  with the same value. This can be done as follows: For every set  $S = S_1 \cup S_2 \subseteq V_1 \cup V_2$  of size at most  $r$ , let  $S_{\mathcal{U}} \subseteq V$  be the set of their corresponding vertices in  $G$  and define the local distribution

### Chapter 3. Sherali-Adams Gaps for Constraint Satisfaction Problems

$\mathcal{D}'(S)$  by mimicking the local distribution  $\mathcal{D}(S_{\mathcal{U}})$ , repeating labels if the same vertex  $v \in S_{\mathcal{U}}$  has its two copies  $v_1$  and  $v_2$  in  $S$ .

Now let  $\mathcal{J}$  be the 1F-CSP instance over  $n$  variables obtained by our reduction from the UNIQUE GAMES instance  $\mathcal{U}'$ , where  $n = 2^R n_G$ . Since  $\text{OPT}(\mathcal{U}') \leq \delta$ , we get from Lemma 3.7 that  $\text{OPT}(\mathcal{J}) \leq \eta$ . Similarly, we know from Lemma 3.8 that using an  $n_G^{2\kappa}$ -rounds Sherali-Adams solution for  $\mathcal{U}'$ , we can define an  $n^\kappa$ -rounds Sherali-Adams solution of  $\mathcal{J}$  of roughly the same value, where we used the fact that  $R$  is a constant and hence  $(2^{-2R\kappa} n^{2\kappa}) > n^\kappa$  for sufficiently large values of  $n$ . This concludes the proof.  $\square$

Provided that Claim 3.9 holds, we get that the 1F-CSP problem fools the Sherali-Adams relaxation even after  $n^\kappa$  many rounds for some constant  $1 > \kappa > 0$ . Thus we conclude this section by proving Claim 3.9.

*Proof of Claim 3.9.* Assume that  $\Lambda$  satisfies  $vw_1, \dots, vw_t$  simultaneously, i.e.,

$$\pi_{v,w_1}(\Lambda(w_1)) = \dots = \pi_{v,w_t}(\Lambda(w_t)) = \Lambda(v), \quad (3.10)$$

and let  $C_{x,S}$  and  $C_{\bar{x},S}$  be the sub-cubes as in Figure 3.1. According to the new assignment, every variable  $\langle w_i, z \rangle$  in the support of  $C(v, \mathcal{W}_v, x, S)$  takes the value  $z_{\Lambda(w_i)}$ . Assume w.l.o.g. that  $\langle w_i, z \rangle$  is such that  $\pi_{v,w_i}^{-1}(z) \in C_{x,S}$ , and let  $y \in C_{x,S}$  satisfies  $\pi_{v,w_i}(y) = z$ . Then we get

$$z_{\Lambda(w_i)} = \pi_{v,w_i}(y)_{\Lambda(w_i)} = y_{\pi_{v,w_i}(\Lambda(w_i))} = y_{\Lambda(v)}, \quad (3.11)$$

where the last equality follows from (3.10). We know from the construction of the sub-cube  $C_{x,S}$  that for all  $j \notin S$  and for all  $y \in C_{x,S}$ , we have  $y_j = x_j$ . It then follows that if  $\Lambda(v) \notin S$ , equation 3.11 yields that

$$z_{\Lambda(w_i)} = y_{\Lambda(v)} = x_{\Lambda(v)}, \quad \forall \langle w_i, z \rangle \text{ s.t. } \pi_{v,w_i}^{-1}(z) \in C_{x,S}.$$

Similarly, for the variables  $\langle w_i, z \rangle$  with  $\pi_{v,w_i}^{-1}(z) \in C_{\bar{x},S}$ , we get that

$$z_{\Lambda(w_i)} = y_{\Lambda(v)} = \bar{x}_{\Lambda(v)}, \quad \forall \langle w_i, z \rangle \text{ s.t. } \pi_{v,w_i}^{-1}(z) \in C_{\bar{x},S}.$$

Thus far we proved that if  $\Lambda$  satisfies  $vw_1, \dots, vw_t$  simultaneously and  $\Lambda(v) \notin S$ , then  $\psi(\Lambda)$  satisfies  $C(v, \mathcal{W}_v, x, S)$ . But we know by construction that  $|S| = \varepsilon R$ , and hence  $\Lambda(v) \notin S$  with probability at least  $1 - \varepsilon$ .  $\square$

### 3.5 Sherali-Adams Integrality Gap for $\kappa$ -NOR.

This section will be dedicated to proving the following theorem.

**Theorem 3.11.** *For any  $\varepsilon > 0$  and integer  $q \geq 2$ , there exist a scalar  $\kappa > 0$  and an integer  $k$  so*

### 3.5. Sherali-Adams Integrality Gap for $\kappa$ -NOR.

that for every sufficiently large  $n$  there exists a  $\kappa$ -NOR instance  $\mathcal{F}$  of arity  $k$  over  $n$  variables satisfying

1.  $\text{OPT}(\mathcal{F}) \leq \varepsilon$ ;
2. There is a solution to the  $n^\kappa$ -round Sherali-Adams relaxation of value at least  $1 - 1/q - \varepsilon$ .

The above theorem states that the  $\kappa$ -NOR problem can fool the Sherali-Adams relaxation even after  $n^\kappa$  many rounds. Before we proceed, we discuss functions of the form  $f : \mathbb{Z}_q^R \rightarrow \{0, 1\}$ . These functions will play a crucial role in the analysis.

#### 3.5.1 Functions Over the Domain $\mathbb{Z}_q$ .

In order to construct Sherali-Adams integrality gaps for the  $\kappa$ -NOR problem, we also reduce from the UNIQUE GAMES problem. The analysis of this reduction relies heavily on known properties regarding functions of the form  $f : \mathbb{Z}_q^R \rightarrow \{0, 1\}$ , where  $\mathbb{Z}_q$  is to be thought of as the domain of the new CSP, and  $R$  as the label set size of the UNIQUE GAMES instance. More precisely, we exploit the drastic difference in the behavior of functions depending on whether they have *influential* coordinates or not. To quantify these differences, we first need the following definitions.

**Definition 3.12.** For a function  $f : \mathbb{Z}_q^R \rightarrow \{0, 1\}$ , and an index  $i \in [R]$ , the influence of the  $i$ -th coordinate is given by

$$\text{Inf}_i(f) = \mathbb{E} [\text{Var} [f(x) | x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_R]] ,$$

where  $x_1, \dots, x_R$  are uniformly and independently distributed, and  $\text{Var}[\cdot]$  denotes the variance.

An alternative definition for the influence requires defining the Fourier expansion of a function  $f$  of the form  $f : \mathbb{Z}_q^R \rightarrow \{0, 1\}$ . To do this, let  $\phi_0, \phi_1, \dots, \phi_{q-1} : \mathbb{Z}_q \rightarrow \mathbb{R}$  with  $\phi_0 \equiv 1$  be such that for all  $i, j \in [q]$ , we have

$$\mathbb{E}_{y \in \mathbb{Z}_q} [\phi_i(y)\phi_j(y)] = \begin{cases} 0 & \text{if } i \neq j, \\ 1 & \text{if } i = j, \end{cases}$$

where the expectation is taken over the uniform distribution, and define the functions  $\phi_\alpha : \mathbb{Z}_q^R \rightarrow \mathbb{R}$  for every  $\alpha \in \mathbb{Z}_q^R$  to be

$$\phi_\alpha(x) := \prod_{i=1}^R \phi_{\alpha_i}(x_i) ,$$

### Chapter 3. Sherali-Adams Gaps for Constraint Satisfaction Problems

---

for any  $x \in \mathbb{Z}_q^R$ ; here we identify  $\lceil q \rceil$  with  $\mathbb{Z}_q$ . We take these functions for defining our Fourier basis. Note that this coincides with the boolean case, where for  $b \in \{0, 1\}$  we have  $\phi_0 \equiv 1$ , and  $\phi_1(b) = (-1)^b$  (or the identity function in the  $\{-1, 1\}$  domain). For a more elaborate discussion on the Fourier expansion in generalized domains, we refer the interested reader to Chapter 8 in [87].

Having fixed the functions  $\phi_0, \phi_1, \dots, \phi_{q-1}$ , every function  $f : \mathbb{Z}_q^R \rightarrow \{0, 1\}$  can be uniquely expressed as

$$f(x) = \sum_{\alpha \in \mathbb{Z}_q^R} \hat{f}_\alpha \phi_\alpha(x),$$

with  $\hat{f}_\alpha \in \mathbb{R}$ . Equipped with this, we can relate the influence of a variable  $i \in [R]$  with respect to a function  $f : \mathbb{Z}_q^R \rightarrow \{0, 1\}$ , to the Fourier coefficients of  $f$  as follows:

$$\text{Inf}_i(f) = \sum_{\alpha: \alpha_i \neq 0} \hat{f}_\alpha^2.$$

In our analysis we will however be interested in *degree- $d$*  influences, denoted  $\text{Inf}_i^d(f)$  and defined as

$$\text{Inf}_i^d(f) = \sum_{\alpha: \alpha_i \neq 0, \|\alpha\|_0 \leq d} \hat{f}_\alpha^2,$$

where  $\|\alpha\|_0$  in this context is the support of  $\alpha$ , i.e., the number of indices  $j \in [R]$  such that  $\alpha_j \neq 0$ .

*Observation 3.13* (see, e.g., Proposition 3.8 in [83]). For a function  $f : \mathbb{Z}_q^R \rightarrow \{0, 1\}$ , the sum of all degree- $d$  influences is at most  $d$ .

We will also need a generalization of the notion of sub-cubes defined in Figure 3.1 in order to state the "It Ain't Over Till It's Over" Theorem [83], a main ingredient of the analysis of the reduction. In fact we only state and use a special case of it, as it appears in [102].

**Definition 3.14.** Fix  $\varepsilon > 0$ . For  $x \in \mathbb{Z}_q^R$  and  $S_\varepsilon \subseteq [R]$  such that  $|S_\varepsilon| = \varepsilon R$ , the sub-cube  $C_{x, S_\varepsilon}$  is defined as follows:

$$C_{x, S_\varepsilon} := \left\{ z \in \mathbb{Z}_q^R : z_j = x_j \ \forall j \notin S_\varepsilon \right\} \subseteq \mathbb{Z}_q^R.$$

For a sub-cube  $C_{x, S_\varepsilon} \subseteq \mathbb{Z}_q^R$  and a function  $f : \mathbb{Z}_q^R \rightarrow \{0, 1\}$ , let  $f|_{C_{x, S_\varepsilon}}$  denote the restriction of  $f$  on the sub-cube  $C_{x, S_\varepsilon}$ . In this notation,  $f|_{C_{x, S_\varepsilon}} \equiv 0$  means that  $f(z) = 0$  for all  $z \in C_{x, S_\varepsilon}$ , i.e.,  $f$  is identical to 0 on the sub-cube  $C_{x, S_\varepsilon}$ .

**Theorem 3.15** (*Special case of the It Ain't Over Till It's Over Theorem*). For every  $\varepsilon, \eta > 0$ , and integer  $q$ , there exist  $\vartheta > 0$  and integers  $t, d$  such that any collection of functions  $f_1, \dots, f_t :$



$\mathbb{Z}_q^R \rightarrow \{0, 1\}$  that satisfies

$$\forall j: \mathbb{E} [f_j] \geq \eta \quad \text{and} \quad \forall i \in [R], \forall 1 \leq \ell_1 \neq \ell_2 \leq t: \min \left\{ \text{Inf}_i^d(f_{\ell_1}), \text{Inf}_i^d(f_{\ell_2}) \right\} \leq \vartheta,$$

has the property

$$\mathbb{P}_{x, S_\varepsilon} \left[ \bigwedge_{j=1}^t \left( f_j|_{C_{x, S_\varepsilon}} \equiv 0 \right) \right] \leq \eta/2,$$

where the probability over  $x \in \mathbb{Z}_q^R$  and  $S_\varepsilon \subseteq [R]$  with  $|S_\varepsilon| = \varepsilon R$  is taken independently uniformly at random.

Essentially what this theorem says is that if a collection of  $t$  fairly balanced functions are all identical to zero on the same random sub-cube with non-negligible probability, then at least two of these functions must share a common influential coordinate. In fact all the functions that we use throughout this section satisfy a strong balance property, that we refer to as *folding*.<sup>8</sup>

**Folded Functions.** We say that a function  $f: \mathbb{Z}_q^R \rightarrow \{0, 1\}$  is *folded* if every line of the form  $\{x \in \mathbb{Z}_q^R \mid x = a + \lambda \mathbf{1}, \lambda \in \mathbb{Z}_q\}$  contains a unique point  $\gamma \in \mathbb{Z}_q^R$  with  $f(\gamma) = 0$ , where  $\mathbf{1} \in \mathbb{Z}_q^R$  is the all-one vector and  $a \in \mathbb{Z}_q^R$  is any point.

*Remark 3.16.* For any folded function  $f: \mathbb{Z}_q^R \rightarrow \{0, 1\}$ , we have that  $\mathbb{E}_x [f(x)] = 1 - 1/q$ .

We shall also extend the notion of *dictatorship* functions restricted to the folded setting. In this setting, the  $\ell$ -th coordinate dictator function  $f_\ell: \mathbb{Z}_q^R \rightarrow \{0, 1\}$  for some  $\ell \in [R]$  is defined as

$$f_\ell(x) = \begin{cases} 1 & \text{if } x_\ell \neq 0 \\ 0 & \text{if } x_\ell = 0. \end{cases}$$

Notice that  $f_\ell$  is folded because it is zero exactly on the coordinate hyperplane  $\{x \in \mathbb{Z}_q^R \mid x_\ell = 0\}$ .

**Truth Table Model.** In order to guarantee the folding property of a function  $f: \mathbb{Z}_q^R \rightarrow \{0, 1\}$  in the truth table model, we adopt the following convention:

1. The truth table  $Y_f$  has  $q^{R-1}$  entries in  $\mathbb{Z}_q$ , one for each  $x \in \mathbb{Z}_q^R$  such that  $x_1 = 0$ .

<sup>8</sup>We abuse the notion of folding here, and we stress that this should not be confused with the usual notion of folding in the literature, although it coincides with standard folding for the boolean case.

## Chapter 3. Sherali-Adams Gaps for Constraint Satisfaction Problems

---

2. For each  $x \in \mathbb{Z}_q^R$  with  $x_1 = 0$ , the corresponding entry  $Y_f(x)$  contains the unique  $\lambda \in \mathbb{Z}_q$  such that  $f(x + \lambda \mathbf{1}) = 0$ .

We can however use  $Y_f$  to query  $f(x)$  for any  $x \in \mathbb{Z}_q^R$  as follows: we have  $f(x) = 0$  whenever  $Y_f(x - x_1 \mathbf{1}) = Y_f(0, x_2 - x_1, \dots, x_R - x_1) = x_1$  and  $f(x) = 1$  otherwise.

We can now readily extend the notion of the *long code* encoding to match our definition of dictatorship functions.

**Definition 3.17.** The *long code encoding* of an index  $\ell \in [R]$  is simply  $Y_{f_\ell}$ , the truth table of the *folded* dictatorship function of the  $\ell$ -th coordinate. The long code  $Y_{f_\ell} \in \mathbb{Z}_q^{q^{R-1}}$  is indexed by all  $x \in \mathbb{Z}_q^R$  such that  $x_1 = 0$ .

### 3.5.2 Reduction from UNIQUE GAMES to K-NOR.

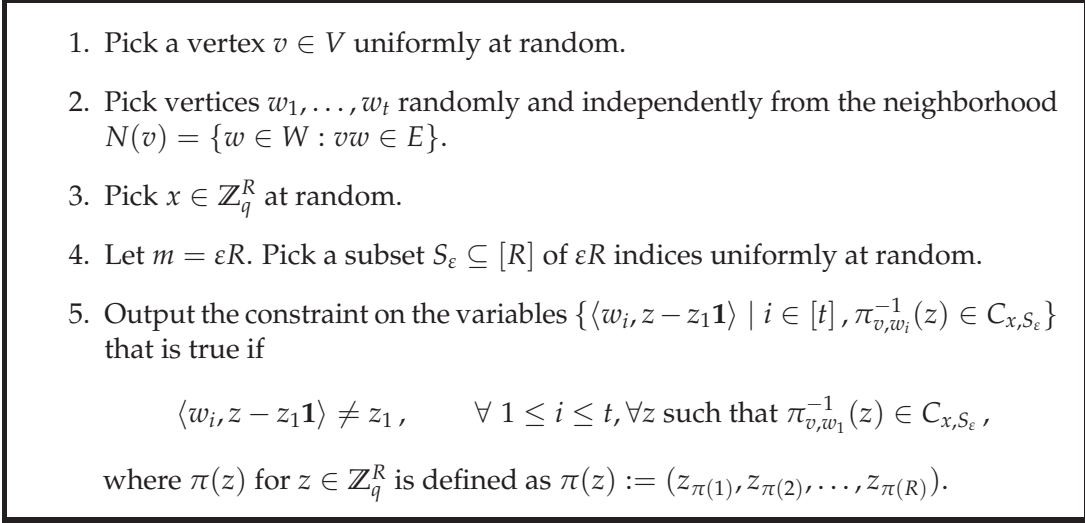
We first describe the reduction from UNIQUE GAMES to K-NOR that is similar in many aspects to the reduction in Section 3.4. We then show that it also preserves the Sherali-Adams integrality gap.

**Reduction.** Let  $\mathcal{U} = (G, [R], \Pi)$  be a UNIQUE GAMES instance over a regular bipartite graph  $G = (V, W, E)$ . Given  $\mathcal{U}$ , we construct an instance  $\mathcal{J}$  of K-NOR. The reduction has three parameters: an integer  $q \geq 2$ , a real  $\varepsilon > 0$  determining the completeness, and an integer  $t$ , where  $\varepsilon$  is chosen such that  $\varepsilon R$  is an integer and  $t$  is sufficiently large depending on  $\varepsilon, q$ , and the desired soundness  $\eta$  (see Lemma 3.18).

The resulting K-NOR instance  $\mathcal{J}$  will be defined over  $|W|q^{R-1}$  variables and  $c|V|$  constraints, where  $c := c(R, \varepsilon, t, \Delta, q)$  is a function of the degree  $\Delta$  of the UNIQUE GAMES instance, and the constants  $R, t, q$  and  $\varepsilon$ . For our purposes, the UNIQUE GAMES integrality gap instance that we start from, has constant degree  $\Delta$ , and hence  $c$  is a constant.

We refer to the variables of  $\mathcal{J}$  as follows: it has a variable  $\langle w, z \rangle \in \mathbb{Z}_q$  for each  $w \in W$  and  $z \in \mathbb{Z}_q^R$  such that  $z_1 = 0$ . For further reference, we let  $\text{Var}(\mathcal{J})$  denote the set of variables of  $\mathcal{J}$  (not to be confused with  $\mathbf{Var}[\cdot]$  denoting the variance). The constraints of  $\mathcal{J}$  are picked according the distribution in Figure 3.2. One can see that a constraint  $C := C(v, \mathcal{W}_v, x, S_\varepsilon)$  is then defined by a random vertex  $v$  (Line 1),  $t$  random neighbors  $\mathcal{W}_v = \{w_1, \dots, w_t\}$  (Line 2), a random  $x \in \mathbb{Z}_q^R$  (Line 3) and a random subset  $S_\varepsilon \subseteq [R]$  (Line 4). We remark that here, in contrast to Section 3.4, we have picked a random subset  $S_\varepsilon$  always of cardinality  $\varepsilon R$ . This in order to apply Theorem 3.15 as stated although one can see that the two versions are quantitatively equivalent.

Note that if we think of the variables  $\langle w, z \rangle$  for a fixed  $w \in W$  as the truth table of some function  $f_w : \mathbb{Z}_q^R \rightarrow \{0, 1\}$ , then  $f_w$  is *forced* to satisfy the folding property.


 Figure 3.2 – Distribution for the  $\kappa$ -NOR constraints

We claim that if the starting UNIQUE GAMES instance  $\mathcal{U}$  was a Sherali-Adams integrality gap instance, then  $\mathcal{F}$  is also an integrality gap instance for the  $\kappa$ -NOR problem. Similar to Section 3.4, we prove this in two steps; we first show that if  $\text{OPT}(\mathcal{U})$  is *small*, then so is  $\text{OPT}(\mathcal{F})$ . Formally speaking, the following holds:

**Lemma 3.18.** *For every alphabet size  $q \geq 2$  and  $\varepsilon, \eta > 0$  such that  $\eta \leq 1/q$  there exists an integer  $t$  so that  $\text{OPT}(\mathcal{F}) \leq \eta$  if  $\text{OPT}(\mathcal{U}) \leq \delta$  where  $\delta > 0$  is a constant that only depends on  $\varepsilon, \eta$ , and  $q$ .*

*Proof.* Suppose towards contradiction that  $\text{OPT}(\mathcal{F}) > \eta$ . As noted earlier, for a fixed  $w \in W$  we can think of the variables  $\langle w, z \rangle \in \text{Var}(\mathcal{F})$  as the truth table of a folded function  $f_w : \mathbb{Z}_q^R \rightarrow \{0, 1\}$ , where  $Y_{f_w}(z) := \langle w, z \rangle$ . This is possible since the variables  $\langle w, z \rangle \in \text{Var}(\mathcal{F})$  are restricted to  $z \in \mathbb{Z}_q^R$  with  $z_1 = 0$ . Given this alternative point of view, define for every vertex  $w \in W$ , a set of *candidate labels*  $L[w]$  as follows:

$$L[w] = \{i \in [R] : \text{Inf}_i^d(f_w) \geq \vartheta\},$$

where  $d$  and  $\vartheta$  are selected depending on  $\varepsilon, \eta$ , and  $q$  according to Theorem 3.15. Note that  $|L[w]| \leq d/\vartheta$  by Observation 3.13.

For every vertex  $v \in V$ , and every  $\mathcal{W}_v = \{w_1, \dots, w_t\} \subseteq N(v)$ , consider the following set of constraints:

$$\mathcal{C}_{v, \mathcal{W}_v} := \left\{ C_{v, \mathcal{W}_v, x, S} : x \in \mathbb{Z}_q^R, S \subseteq [R] \text{ such that } |S| = \varepsilon R \right\}.$$

A standard counting argument then shows that if  $\text{OPT}(\mathcal{F}) > \eta$ , then at least an  $\eta/2$

### Chapter 3. Sherali-Adams Gaps for Constraint Satisfaction Problems

fraction of the tuples  $(v, w_1, \dots, w_t)$  have more than  $\eta/2$  fraction of the constraints inside  $\mathcal{C}_{v, \mathcal{W}_v}$  satisfied. We refer to such tuples as *good*. Adopting the language of folded functions instead of variables, the aforementioned statement can be casted as

$$\mathbb{P}_{x \in \mathbb{Z}_q^R, S_\varepsilon \subseteq [R]} \left[ \bigwedge_{i=1}^t \left( f_{w_i}^{x|_{\pi_{v, w_i}(C_{x, S_\varepsilon})}} \equiv 1 \right) \right] > \eta/2, \quad \text{if the tuple } (v, w_1, \dots, w_t) \text{ is good,}$$

where for a permutation  $\pi$  we use the notation

$$f|_{\pi(C_{x, S_\varepsilon})} \equiv 1 \quad \iff \quad f(z) = 1 \quad \forall z \text{ such that } \pi^{-1}(z) \in C_{x, S_\varepsilon}.$$

From Remark 3.16, we get that  $\mathbb{E}[f_{w_i}] = 1 - 1/q \leq 1 - \eta$ , and hence invoking Theorem 3.15 on the functions  $\bar{f}_{w_1}, \dots, \bar{f}_{w_t}$ , where  $\bar{f}_{w_i}(x) := 1 - f_{w_i}(\pi_{v, w_i}(x))$ , yields that for every good tuple, there exists  $\ell_1 \neq \ell_2 \in [t]$  such that  $\bar{f}_{w_{\ell_1}}$  and  $\bar{f}_{w_{\ell_2}}$  share a common influential coordinate. Note that this is equivalent to saying that there exists  $j_1 \in L[w_{\ell_1}]$ ,  $j_2 \in L[w_{\ell_2}]$  such that  $\pi_{v, w_{\ell_1}}(j_1) = \pi_{v, w_{\ell_2}}(j_2)$ .

We now claim that if  $\text{OPT}(\mathcal{F}) > \eta$ , then we can construct a labeling  $\Lambda : V \cup W \rightarrow [R]$  that satisfies at least  $\frac{\eta \vartheta^2}{2d^2 t^2}$  of edges, which contradicts the fact that  $\text{OPT}(\mathcal{U}) \leq \delta$  for a small enough value of  $\delta > 0$ . Towards this end, consider the following randomized labeling procedure:

1. For every  $w \in W$ , let  $\Lambda(w)$  be a random label from the set  $L[w]$ , or an arbitrary label if  $L[w] = \emptyset$ .
2. For every  $v \in V$ , pick a random neighbor  $w \in N(v)$  and set  $\Lambda(v) = \pi_{v, w}(\Lambda(w))$ .

We can readily calculate the fraction of edges in  $\mathcal{U}$  that are satisfied by  $\Lambda$ . This follows from putting the following observations together:

1. If we pick a random tuple  $(v, w_1, \dots, w_t)$ , it is *good* with probability at least  $\eta/2$ .
2. If  $(v, w_1, \dots, w_t)$  is *good*, and we pick  $w', w''$  at random from  $\{w_1, \dots, w_t\}$ , then with probability at least  $1/t^2$  the functions  $f_{w'}$  and  $f_{w''}$  share a common influential coordinates.
3. If  $(v, w_1, \dots, w_t)$  is *good*, and the functions  $f_{w'}$  and  $f_{w''}$  share a common influential coordinates, then picking a random label to  $w'$  and  $w''$  from  $L[w']$  and  $L[w'']$  respectively, will satisfy  $\pi_{v, w'}(\Lambda(w')) = \pi_{v, w''}(\Lambda(w''))$  with probability at least  $1/(d^2/\vartheta^2)$ .

Hence the expected number of edges satisfied by  $\Lambda$  in this case is

$$\mathbb{P}_{vw \in E} [\Lambda(v) = \pi_{v, w}(\Lambda(w))] \geq \frac{\eta \vartheta^2}{2d^2 t^2}.$$

□

We now show that given an  $r$ -rounds Sherali-Adams solution of *high* value for  $\mathcal{U}$ , we can also come up with an  $r$ -rounds Sherali-Adams solution for  $\mathcal{F}$  of high value as well. The proof goes along the same lines of that of Lemma 3.8, and hence we will only highlight the differences.

**Lemma 3.19.** *Let  $\{\mu(S) \mid S \subseteq V \cup W, |S| \leq r\}$  be a consistent collection of local distributions defining a solution to the  $r$ -rounds Sherali-Adams relaxation of the regular bipartite UNIQUE GAMES instance  $\mathcal{U}$ . Then we can define a consistent collection of local distributions  $\{\sigma(S) \mid S \subseteq \text{Var}(\mathcal{F}), |S| \leq r\}$  defining a solution to the  $r$ -rounds Sherali-Adams relaxation of the  $\kappa$ -NOR instance  $\mathcal{F}$  so that*

$$\begin{aligned} & \mathbb{E}_{C \in \mathcal{C}} \left[ \mathbb{P}_{\alpha \sim \sigma(S_C)} [\alpha \text{ satisfies } C] \right] \\ & \geq (1 - \varepsilon) \left(1 - \frac{1}{q}\right) \left(1 - t \cdot \mathbb{E}_{vw \in E} \left[ \mathbb{P}_{(\Lambda(v), \Lambda(w)) \sim \mu(\{v, w\})} [\Lambda(v) \neq \pi_{w, v}(\Lambda(w))] \right] \right), \end{aligned}$$

where  $t, \varepsilon$  are the parameters of the reduction, and  $\sigma(S_C)$  is the distribution over the set of variables in the support  $S_C$  of constraint  $C$ .

*Proof.* Let  $\{\mu(S) \mid S \subseteq V \cup W, |S| \leq r\}$  be a solution to the  $r$ -rounds SA relaxation of the UNIQUE GAMES instance  $\mathcal{U}$ , and recall that  $\mathcal{F}$  is the  $\kappa$ -NOR instance we get by applying the reduction. We will now use the collection of consistent local distributions of the UNIQUE GAMES instance, to construct another collection of consistent local distributions for the variables in  $\text{Var}(\mathcal{F})$ .

For every set  $S \subseteq \text{Var}(\mathcal{F})$  such that  $|S| \leq r$ , let  $T_S \subseteq W$  be the subset of vertices in the UNIQUE GAMES instance defined as follows:

$$T_S := \{w \in W : \langle w, x \rangle \in S\}. \quad (3.12)$$

We will now construct  $\sigma(S)$  from  $\mu(T_S)$  in the following manner. Given a labeling  $\Lambda_{T_S}$  for the vertices in  $T_S$  drawn from  $\mu(T_S)$ , define an assignment  $\alpha_S$  for the variables in  $S$  as follows: for a variable  $\langle w, x \rangle \in S$ , let  $\ell = \Lambda_{T_S}(w)$  be the label of  $w$  according to  $\Lambda_{T_S}$ . Then the new assignment  $\alpha_S$  sets  $\alpha_S(\langle w, x \rangle) := Y_{f_\ell}(x)$ , where  $Y_{f_\ell}$  is the long code encoding of  $\ell$  as in Definition 3.17. The aforementioned procedure defines a family  $\{\sigma(S)\}_{S \subseteq \text{Var}(\mathcal{F}), |S| \leq r}$  of local distributions for the variables of the  $\kappa$ -NOR instance  $\mathcal{F}$ . The same argument as in the proof of Lemma 3.8 yields that  $\{\sigma(S) \mid S \subseteq \text{Var}(\mathcal{F}), |S| \leq r\}$  defines a feasible solution for the  $r$ -round Sherali-Adams relaxation of the  $\kappa$ -NOR instance  $\mathcal{F}$ .

### Chapter 3. Sherali-Adams Gaps for Constraint Satisfaction Problems

It remains to bound the value of this feasible solution, i.e.,

$$\begin{aligned} & \mathbb{E}_{C \in \mathcal{C}} \left[ \mathbb{P}_{\alpha \sim \sigma(S_C)} [\alpha \text{ satisfies } C] \right] \\ &= \mathbb{E}_{v, w_1, \dots, w_t} \left[ \mathbb{P}_{\Lambda \sim \mu(\{v, w_1, \dots, w_t\}), x, S} [\psi(\Lambda) \text{ satisfies } C(v, \mathcal{W}_v, x, S)] \right]. \end{aligned} \quad (3.13)$$

where  $\psi(\cdot)$  the operator mapping a labeling of the vertices in  $T_S$  to an assignment for the variables in  $S$ , i.e.,  $\psi(\Lambda_{T_S}) = \alpha_S$ . The following claim, which is in some sense the equivalent of Claim 3.9 in the  $\kappa$ -NOR language, along with the same remaining steps of the proof of Lemma 3.8 will yield the proof.

**Claim 3.20.** *If  $\Lambda$  satisfies  $vw_1, \dots, vw_t$  simultaneously, then  $\psi(\Lambda)$  satisfies  $C(v, \mathcal{W}_v, x, S)$  with probability at least  $(1 - \varepsilon)(1 - \frac{1}{q})$ . Moreover, if we additionally have that  $\Lambda(v) \notin S$  and  $x_{\Lambda(v)} \neq 0$ , then  $\psi(\Lambda)$  always satisfies  $C(v, \mathcal{W}_v, x, S)$ .*

Equipped with this, we can use conditioning to lower-bound the probability inside the expectation in (3.13) by a product of two probabilities, where the first is

$$\mathbb{P}_{\Lambda \sim \mu(\{v, w_1, \dots, w_t\}), x, S} [\psi(\Lambda) \text{ satisfies } C(v, \mathcal{W}_v, x, S) | \Lambda \text{ satisfies } vw_1, \dots, vw_t], \quad (3.14)$$

and the second is

$$\mathbb{P}_{\Lambda \sim \mu(\{v, w_1, \dots, w_t\})} [\Lambda \text{ satisfies } vw_1, \dots, vw_t].$$

Thus using Claim 3.20, we get

$$\begin{aligned} & \mathbb{E}_{C \in \mathcal{C}} \left[ \mathbb{P}_{\alpha \sim \sigma(S_C)} [\alpha \text{ satisfies } C] \right] \\ & \geq (1 - \varepsilon) \left(1 - \frac{1}{q}\right) \cdot \mathbb{E}_{v, w_1, \dots, w_t} \left[ \mathbb{P}_{\Lambda \sim \mu(\{v, w_1, \dots, w_t\})} [\Lambda \text{ satisfies } vw_1, \dots, vw_t] \right] \\ & \geq (1 - \varepsilon) \left(1 - \frac{1}{q}\right) \cdot \left(1 - t \cdot \mathbb{E}_{v, w} \left[ \mathbb{P}_{\Lambda \sim \mu(\{v, w\})} [\Lambda \text{ does not satisfy } vw] \right] \right). \end{aligned}$$

□

The proof of Corollary 3.10 adjusted to the  $\kappa$ -NOR problem now yields Theorem 3.11. It remains to prove Claim 3.20:

*Proof.* Claim 3.20: Assume that  $\Lambda$  satisfies  $vw_1, \dots, vw_t$  simultaneously, i.e.,

$$\pi_{v, w_1}(\Lambda(w_1)) = \dots = \pi_{v, w_t}(\Lambda(w_t)) = \Lambda(v), \quad (3.15)$$

and let  $C_{x, S_\varepsilon}$  be the sub-cube as in Figure 3.2. Recall that a constraint  $C(v, \mathcal{W}_v, x, S_\varepsilon)$  looks

as follows:

$$\langle w_i, z - z_1 \mathbf{1} \rangle \neq z_1, \quad \forall 1 \leq i \leq t, \forall z \text{ such that } \pi_{v, w_1}^{-1}(z) \in C_{x, S_\varepsilon}. \quad (3.16)$$

We now adopt the functions point of view, i.e., for a  $w \in W$ , the variables  $\langle w, z \rangle$  for  $z \in [q]^R$  are the entries of the truth table of a function  $f_w$ , and according to the new assignment  $\Lambda$ ,  $f_w$  is the *folded* dictatorship function of the label of  $\Lambda(w)$ .

So if we let  $f := f_{w_i}$  for some  $1 \leq i \leq t$ , and  $z := \langle w_i, z \rangle$ , we get that

$$\langle w_i, z - z_1 \mathbf{1} \rangle \neq z_1 \quad \iff \quad f(z) \neq 0,$$

and by our definition of the dictatorship function, the latter is zero iff  $z_{\Lambda(w_i)} = 0$ . But

$$z_{\Lambda(w_i)} = \pi_{v, w_i}(\mathbf{y})_{\Lambda(w_i)} = \mathbf{y}_{\pi_{v, w_i}(\Lambda(w_i))} = \mathbf{y}_{\Lambda(v)}, \quad (3.17)$$

where the last equality follows from (3.15). We know from the construction of the sub-cube  $C_{x, S_\varepsilon}$  that for all  $j \notin S_\varepsilon$  and for all  $\mathbf{y} \in C_{x, S_\varepsilon}$ , we have  $y_j = x_j$ . It then follows that if  $\Lambda(v) \notin S_\varepsilon$ , equation 3.17 yields that

$$z_{\Lambda(w_i)} = \mathbf{y}_{\Lambda(v)} = x_{\Lambda(v)} \quad \forall \langle w_i, z \rangle \text{ s.t. } \pi_{v, w_i}^{-1}(z) \in C_{x, S_\varepsilon}.$$

Moreover, given that  $x$  is chosen uniformly at random from  $[q]^R$ , we get that for any  $i \in [R]$ ,  $\mathbb{P}_{x \in [q]^R} [x_i = 0] = \frac{1}{q}$ .

Thus far we proved that if  $\Lambda$  satisfies  $v w_1, \dots, v w_t$  simultaneously and  $\Lambda(v) \notin S$ , then  $\psi(\Lambda)$  satisfies  $C(v, \mathcal{W}_v, x, S)$  with probability  $1 - \frac{1}{q}$ . But we know by construction that  $|S| = \varepsilon R$ , and hence  $\Lambda(v) \notin S$  with probability at least  $1 - \varepsilon$ .  $\square$

### 3.6 LP-Hardness of 1F-CSP and $\kappa$ -NOR

We have thus far constructed 1F-CSP (and  $\kappa$ -NOR) instances that are able to fool relaxations resulting from performing a large number of rounds of the SA hierarchy (formally stated in Corollary 3.10 and Theorem 3.11). In the language of LP  $(c, s)$ -approximability, our results can be restated as follows:

**Theorem 3.21.** *For every  $\varepsilon > 0$  (and  $q \geq 2$ ), and for infinitely many  $n$ , no  $\Omega_\varepsilon(n)$ -rounds Sherali-Adams relaxation can achieve a  $(1 - \varepsilon, \varepsilon)$ -approximation (and  $(1 - \frac{1}{q} - \varepsilon, \varepsilon)$ ) for the 1F-CSP problem (and respectively for the  $\kappa$ -NOR problem) for instances on  $n$  variables.*

As far as CONSTRAINT SATISFACTION PROBLEMS problems are concerned, LPs arising from the SA hierarchy are in some sense at least as powerful as any other LP of the same size. In particular, Theorem 3.22 was first proved in [31] for general alphabet CSPs; the lower bound was later improved in [71], as stated in Theorem 3.23, however it currently

### Chapter 3. Sherali-Adams Gaps for Constraint Satisfaction Problems

---

only applies directly to Binary CSPs:

**Theorem 3.22** (Theorem 3.2 in [31]). *Consider a function  $f : \mathbb{N} \rightarrow \mathbb{N}$ . Suppose that the  $f(n)$ -round Sherali-Adams relaxation cannot achieve a  $(c, s)$ -approximation for a CONSTRAINT SATISFACTION PROBLEM  $\Pi_n$  over  $n$  variables. Then for sufficiently large  $n$ , no LP relaxation of size at most  $n^{f(n)^2}$  can achieve a  $(c, s)$ -approximation for  $\Pi_N$ , where  $N \leq n^{10f(n)}$ .*

**Theorem 3.23** (Theorem 1.2 in [71]). *There exist constants  $0 < h < H$  such that the following holds. Consider a function  $f : \mathbb{N} \mapsto \mathbb{N}$ . Suppose that the  $f(n)$ -round Sherali Adams relaxation for a binary CONSTRAINT SATISFACTION PROBLEM cannot achieve a  $(c, s)$ -approximation on instances on  $n$  variables. Then no LP of size at most  $n^{hf(n)}$  can achieve a  $(c, s)$ -approximation for the CONSTRAINT SATISFACTION PROBLEM on  $n^H$  variables.*

Combining Theorems 3.21, 3.22 and 3.23 yields the main result of this chapter stated in the following corollaries:

**Corollary 3.24.** *For some universal constant  $H \geq 1$ , and for every  $\varepsilon > 0$ , there exists constants  $c_1(\varepsilon)$ ,  $k = k(\varepsilon)$  such that no LP relaxation of size less than  $2^{c_1(\varepsilon)n^{1/H}}$  achieves a  $(1 - \varepsilon, \varepsilon)$  for the 1F-CSP $_{n,k}$  problem.*

**Corollary 3.25.** *For every  $\varepsilon > 0$ , and alphabet size  $q \geq 2$ , there exists a constant arity  $k = k(\varepsilon)$  such that no LP relaxation of size less than  $n^{o\left(\frac{\log n}{\log \log n}\right)}$  achieves a  $(1 - 1/q - \varepsilon, \varepsilon)$ -approximation for the  $\kappa$ -NOR $_{n,q}$  problem.*



## 4 LP Hardness of Vertex Cover

We proved in Chapter 3 that any good LP relaxation for 1F-CSP and  $\kappa$ -NOR must have a large size. As mentioned earlier, the choice of these two problems is motivated by their implications on other combinatorial problems, such as the Vertex Cover problem and its complement, the Independent Set problem.

The Vertex Cover problem is one of the most important and intensively studied combinatorial optimization problems. Khot and Regev [68] prove that the problem is NP-hard to approximate within a factor  $2 - \epsilon$ , assuming the Unique Games Conjecture. This is tight because the problem has an easy 2-approximation algorithm. Without resorting to the Unique Games Conjecture, the best inapproximability result for the problem is due to Dinur and Safra [40]: Vertex Cover is NP-hard to approximate within a factor 1.3606.

We prove the following unconditional result about the linear programming relaxations of the problem: every LP relaxation that approximates Vertex Cover within a factor  $2 - \epsilon$  has super-exponentially many inequalities. As a direct consequence of our methods, we also establish that LP relaxations (as well as SDP relaxations) that approximate the Independent Set problem within any constant factor must have super-polynomial size.

### 4.1 Introduction.

In this chapter, we prove tight inapproximability results for VERTEX COVER with respect to linear programming relaxations of polynomial size. Recall that the VERTEX COVER problem is the following classic problem: given a graph  $G = (V, E)$  together with vertex costs  $c_v \geq 0, v \in V$ , find a minimum cost set of vertices  $U \subseteq V$  such that every edge has at least one endpoint in  $U$ . Such a set of vertices meeting every edge is called a *vertex cover*.

We saw in Chapter 1 that the natural LP relaxation for the unweighted version of the VERTEX COVER problem has an integrality gap of 2. Similarly, the following weighted

LP relaxation

$$\begin{aligned}
 \min \quad & \sum_{v \in V} c_v x_v \\
 \text{s.t.} \quad & x_u + x_v \geq 1 \quad \forall uv \in E \\
 & 0 \leq x_v \leq 1 \quad \forall v \in V
 \end{aligned} \tag{4.1}$$

approximates VERTEX COVER within a factor 2. (See e.g., Hochbaum [58] and the references therein.) This means that for every cost vector there exists a vertex cover whose cost is at most 2 times the optimum value of the LP. In fact, the (global) *integrality gap* of this LP relaxation, the worst-case ratio over all graphs and all cost vectors between the minimum cost of an integer solution and the minimum cost of a fractional solution, equals 2.

One way to make the LP relaxation (4.1) stronger is by adding valid inequalities. Here, a *valid inequality* is a linear inequality  $\sum_{v \in V} a_v x_v \geq \beta$  that is satisfied by every integral solution. Adding all possible valid inequalities to (4.1) would clearly decrease the integrality gap all the way from 2 to 1, and thus provide a perfect LP formulation. However, this would also yield an LP that we would not be able to write down or solve efficiently, unless  $P = NP$ . Hence, it is necessary to restrict to more tangible families of valid inequalities.

For instance, if  $C \subseteq V$  is the vertex set of an odd cycle in  $G$ , then  $\sum_{v \in C} x_v \geq \frac{|C|+1}{2}$  is a valid inequality for vertex covers, known as an *odd cycle inequality*. However, the integrality gap remains 2 after adding all such inequalities to (4.1). More classes of inequalities are known beyond the odd cycle inequalities. However, we do not know any such class of valid inequalities that would decrease the integrality gap strictly below 2.

There has also been much success in ruling out concrete polynomial-size linear programming formulations arising from, e.g., the addition of a polynomial number of inequalities with sparse support or those arising from hierarchies, where new valid inequalities are generated in a systematic way. For instance, what about adding all valid inequalities supported on at most  $o(n)$  vertices (where  $n$  denotes the number of vertices of  $G$ ), or all those obtained by performing a few rounds of the Lovász-Schrijver (LS) lift-and-project procedure [80]? In their influential paper Arora, Bollobás and Lovász [2] (the journal version [3] is joint work with Tzourakis) prove that none of these broad classes of valid inequalities are sufficient to decrease the integrality gap to  $2 - \epsilon$  for any  $\epsilon > 0$ .

The paper of Arora *et al.* was followed by many papers that derive stronger and stronger tradeoffs between number of rounds and integrality gap for VERTEX COVER and many other problems in various hierarchies, see the related work section below. The focus of this chapter is to prove lower bounds in a more general model. Specifically, our goal is to understand the strength of *any* polynomial-size linear programming relaxation of

VERTEX COVER independently of any hierarchy and irrespectively of any complexity-theoretic assumption such as  $P \neq NP$ .

We will rule out *all possible* polynomial-size LP relaxations obtained from adding an *arbitrary* set of valid inequalities of polynomial size. By “all possible LP relaxations”, we mean that the variables of the LP can be chosen arbitrarily. They do not have to be the vertex-variables of (4.1).

### Contribution.

In the terminology of Section 2.2, we consider the general non-uniform model of LP relaxations as in [31], see also [25]. Given an  $n$ -vertex graph  $G = (V, E)$ , a system of linear inequalities  $Ax \geq b$  in  $\mathbb{R}^d$ , where  $d \in \mathbb{N}$  is arbitrary, defines an *LP relaxation* of VERTEX COVER (on  $G$ ) if the following conditions hold:

**Feasibility:** For every vertex cover  $U \subseteq V$ , we have a feasible vector  $x^U \in \mathbb{R}^d$  satisfying  $Ax^U \geq b$ .

**Linear objective:** For every vertex-costs  $c \in \mathbb{R}_+^V$ , we have an affine function (degree-1 polynomial)  $f_c : \mathbb{R}^d \rightarrow \mathbb{R}$ .

**Consistency:** For all vertex covers  $U \subseteq V$  and vertex-costs  $c \in \mathbb{R}_+^V$ , the condition  $f_c(x^U) = \sum_{v \in U} c_v$  holds.

For every vertex-costs  $c \in \mathbb{R}_+^V$ , the LP  $\min\{f_c(x) \mid Ax \geq b\}$  provides a guess on the minimum cost of a vertex cover. This guess is always a lower bound on the optimum.

We allow arbitrary computations for writing down the LP and do not bound the size of the coefficients. We only care about the following two parameters and their relationship: the *size* of the LP relaxation, defined as the number of inequalities in  $Ax \geq b$ , and the (graph-specific) *integrality gap*, which is the worst-case ratio over all vertex-costs between the true optimum and the guess provided by the LP, for this particular graph  $G$  and LP relaxation.

While formally equivalent to the polyhedral-pair approach in extended formulations [22] (see also [90]), the formalization from above naturally models affine linear functions that we need for reductions and it does not require an LP relaxation to start from. We refer the interested reader to the surveys [37, 62] for an introduction to extended formulations; see also Section 4.3 for more details.

In this chapter, we prove the following result about LP relaxations of the VERTEX COVER problem and, as a byproduct, the INDEPENDENT SET problem.<sup>1</sup>

<sup>1</sup>Recall that an *independent set* (stable set) in graph  $G = (V, E)$  is a set of vertices  $I \subseteq V$  such that no edge has both endpoints in  $I$ . INDEPENDENT SET is the corresponding maximization problem: given a graph

**Theorem 4.1.** *For every sufficiently large  $n$ , there exists an  $n$ -vertex graph  $G = G(n)$  such that: (i) Every size- $2^{n^{o(1)}}$  LP relaxation of VERTEX COVER on  $G$  has integrality gap  $2 - o(1)$ ; (ii) Every size- $2^{n^{o(1)}}$  LP relaxation of INDEPENDENT SET on  $G$  has integrality gap  $\omega(1)$ .*

This solves an open problem that was posed both by Singh [100] and Chan, Lee, Raghavendra and Steurer [31]. In fact, Singh conjectured that every compact (that is, polynomial size), *symmetric* extended formulation for VERTEX COVER has an integrality gap of at least  $2 - \epsilon$ . We prove that his conjecture holds, even if asymmetric extended formulations are allowed.<sup>2</sup>

Our result for the INDEPENDENT SET problem is even stronger than Theorem 4.1, as we are also able to rule out any polynomial size SDP with a constant integrality gap for this problem. Furthermore, combining our proof strategy with more complex techniques we can prove a result similar to Theorem 4.1 for  $q$ -UNIFORM-VERTEX-COVER (that is, vertex cover in  $q$ -uniform *hypergraphs*), for any fixed  $q \geq 2$ . For that problem, every size  $n^{o(\log n / \log \log n)}$  LP relaxation has integrality gap  $q - o(1)$ . This generalizes our result on (graph) VERTEX COVER.

In the general model of LP relaxations outlined above, the LPs are designed with the knowledge of the graph  $G = (V, E)$ ; As we saw in Section 2.2.3, this is a *non-uniform* model as the LP can depend on the graph. It captures the natural LP relaxations for VERTEX COVER and INDEPENDENT SET whose constraints depend on the graph structure. This is in contrast to previous lower bound results ([22, 4, 23]) on the LP formulation complexity of INDEPENDENT SET, which are of a *uniform* nature: In these works, the formulation of the LP relaxation was agnostic to the input graph and only allowed to depend on the number of vertices of the graph. In general non-uniform models are stronger (hence also the lower bounds for it) and interestingly, this allows for stronger LP relaxations for INDEPENDENT SET than NP-hardness would predict. This phenomenon is related to the approximability of problems with preprocessing. In Section 4.6, we observe that a result of Feige and Jozeph [44] implies that there exists a size- $O(n)$  LP formulation for approximating INDEPENDENT SET within a multiplicative factor of  $O(\sqrt{n})$ .

### Related Work.

Most of the work on extended formulations is ultimately rooted in Yannakakis's famous paper [107] in which he proved that every symmetric extended formulation of the matching polytope and (hence) TSP polytope of the  $n$ -vertex complete graph has size  $2^{\Omega(n)}$ . Yannakakis's work was motivated by approaches to proving  $P = NP$  by providing

---

together with a weight for each vertex, find a maximum weight independent set.

<sup>2</sup>Note that in some cases imposing symmetry is a severe restriction, see Kaibel, Pashkovich and Theis [63].

small (symmetric) LPs for the TSP, which he ruled out.

The paper of Arora *et al.* [2, 3] revived Yannakakis’s ideas in the context of hardness of approximation and provided lower bounds for VERTEX COVER in LS. It marked the starting point for a whole series of papers on approximations via hierarchies. Shortly after Arora *et al.* proved that performing  $O(\log n)$  rounds of LS does not decrease the integrality gap below 2, Schoenebeck, Trevisan and Turlakis [96] proved that this also holds for  $o(n)$  rounds of LS. A similar result holds for the stronger Sherali-Adams (SA) hierarchy [99]: Charikar, Makarychev and Makarychev [32] showed that  $\Omega(n^\delta)$  rounds of SA are necessary to decrease the integrality gap beyond  $2 - \varepsilon$  for some  $\delta = \delta(\varepsilon) > 0$ .

Beyond linear programming hierarchies, there are also semidefinite programming (SDP) hierarchies, e.g., Lovász-Schrijver (LS+) [80] and Sum-of-Squares/Lasserre [89, 72, 73]. Georgiou, Magen, Pitassi and Turlakis [49] proved that  $O(\sqrt{\log n / \log \log n})$  rounds of LS+ does not approximate VERTEX COVER within a factor better than 2. In this chapter, we focus mostly on the LP case.

Other papers in the “hierarchies” line of work include [39, 48, 95, 70, 92, 103, 65, 18, 17].

Although hierarchies are powerful tools, they have their limitations. For instance,  $o(n)$  rounds of SA does not give an approximation of KNAPSACK with a factor better than 2 [65]. However, for every  $\varepsilon > 0$ , there exists a size- $n^{1/\varepsilon+O(1)}$  LP relaxation that approximates KNAPSACK within a factor of  $1 + \varepsilon$  [19].

Besides the study of hierarchy-based approaches, there is a distinct line of work that, inspired directly by Yannakakis’s paper, seeks to study the power of general (linear) extended formulations, independently of any hierarchy, see e.g., [93, 46, 22, 17, 4, 23, 94]. Limitations of semidefinite extended formulations were also studied recently, see [27, 76].

The lines of work on hierarchies and (general) extended formulations in the case of CONSTRAINT SATISFACTION PROBLEMS (CSPs) were merged in the work of Chan *et al.* [31], and later strengthened by Kothari *et al.* [71]. This is crucial for proving our results for 1F-CSP and  $\kappa$ -NOR in Chapter 3. Recall that the main result of [31, 71] states that for Max-CSPs, SA is the best possible among all LP relaxations in the sense that if there exists a size- $n^r$  LP relaxation approximating a given Max-CSP within factor  $\alpha$  then performing  $2r$  rounds of SA would also provide a factor- $\alpha$  approximation. They obtained several strong LP inapproximability results for Max-CSPs such as MAX CUT and MAX 3-SAT. This result was recently strengthened in a breakthrough by Lee, Raghavendra, and Steurer [76], who obtained analogous results showing (informally) that the Sum-of-Squares/Lasserre hierarchy is the best possible among all SDP relaxations for Max-CSPs.

Braun, Pokutta and Zink [25] developed a framework for proving lower bounds on the size of LP relaxations via reductions. Using [31] and FGLSS graphs [43], they obtained a  $n^{\Omega(\log n / \log \log n)}$  size lower bound for approximating VERTEX COVER within

a factor of  $1.5 - \epsilon$  and INDEPENDENT SET within a factor of  $2 - \epsilon$ . We improve these inapproximability factors to a tight  $2 - \epsilon$  and any constant, respectively.

### Outline.

The framework in Braun *et al.* [25] formalizes sufficient properties of reductions for preserving inapproximability with respect to extended formulations / LP relaxations; this reduction mechanism does not capture all known reductions as it relates the objective functions in a linear way and maps instances and solutions independently. Using this framework, they gave a reduction from MAX CUT to VERTEX COVER thus yielding the aforementioned result.

A natural approach for strengthening the hardness factor is to reduce from UNIQUE GAMES instead of MAX CUT (since VERTEX COVER is known to be UNIQUE GAMES-hard to approximate within a factor  $2 - \epsilon$ ). However, one obstacle is that, in known reductions from UNIQUE GAMES, the optimal value of the obtained VERTEX COVER instance is not *linearly* related to the value of the UNIQUE GAMES instance. This makes these reductions unsuitable for the framework in [25] (see Definition 4.4).

We overcome this obstacle by designing a two-step reduction, where the first step was already presented in Chapter 3. To recap, in the first step, we interpreted the “one free bit” PCP test of Bansal and Khot [9] as a reduction from a UNIQUE GAMES instance to a one free bit CSP (1F-CSP). We then used the family of SA integrality gap instances for the UNIQUE GAMES problem constructed by Charikar *et al.* [32], to construct a similar family for this CSP. This, together with the main result of Chan *et al.* [31] and Kothari *et al.* [71] applied to this particular CSP, implied that no subexponential size LP relaxation can provide a constant factor approximation for 1F-CSP. This chapter deals with the second step of this reduction. Specifically, we will perform in Section 4.4 a reduction from 1F-CSP to VERTEX COVER, in the framework of Braun *et al.* [25], which yields our main result.

Later, following a slightly different and more challenging route we prove tight hardness of approximation for LP relaxations of  $q$ -UNIFORM-VERTEX-COVER for every  $q \geq 2$ . This is done in Section 4.5.

## 4.2 From CONSTRAINT SATISFACTION PROBLEMS to Graphs

We present in Sections 4.4 and 4.5 the reductions that yield the promised LP-hardness results. However, we give in this section a clean intuition behind these reductions, without the overhead introduced by the reduction framework.

### 4.2.1 Reduction to Graphs

Being one of most well-known NP-complete problems, the inapproximability of the VERTEX COVER problem has attracted a long line of research in various computational model such as  $P \neq NP$  [57, 40], the Unique Games Conjecture [69, 9], extended formulations [25], etc.. Most of these inapproximability reductions start from a family of hard CONSTRAINT SATISFACTION PROBLEM instances, and use what is now known as the FGLSS<sup>3</sup> graph [43] reduction.

Namely, given a binary CONSTRAINT SATISFACTION PROBLEM instance  $\mathcal{F}$  over  $n$  variables  $x_1, \dots, x_n$ , and a collection of  $m$  constraints  $\mathcal{C} = \{C_{S_1, A_1}, \dots, C_{S_m, A_m}\}$ <sup>4</sup>, the corresponding FGLSS graph  $G_{\mathcal{F}} = (V, E)$  is constructed as follows:

**Vertex set:** For every constraint  $C_{S,A} \in \mathcal{C}$ , let  $C_{S,A}^{-1}(1)$  be the set of satisfying partial assignments, i.e.,

$$C_{S,A}^{-1}(1) = \{\alpha \in \{0,1\}^{|S|} : C_{S,A}(x_\alpha) = 1\},$$

where  $x_\alpha \in \{0,1\}^n$  is any  $n$ -dimensional bit vector whose projection on  $S$  is  $\alpha$ . In particular, if  $f$  is the free bit complexity of the underlying predicate of the constraint  $C_{S,A}$ , then  $|C_{S,A}^{-1}(1)| = 2^f$ . In total, our vertex set is then

$$V = \bigcup_{C_{S,A} \in \mathcal{C}} V_{S,A}, \quad \text{where} \quad V_{S,A} = \{v_{S,A,\alpha} : \alpha \in C_{S,A}^{-1}(1)\}.$$

**Edge set:** We have an edge between every pair of vertices corresponding to conflicting assignments. Formally, we have an edge  $e \in E$  between two vertices  $v_{S,A,\alpha}, v_{S',A',\alpha'}$  if there exists some index  $i \in S \cap S'$  such that  $\alpha(i) \neq \alpha'(i)$ , where  $\alpha(i)$  is the assignment of the variable  $x_i$  according to the partial assignment  $\alpha$ .

We illustrate this FGLSS reduction in Figure 4.1 by using a MAX CUT instance as the starting CONSTRAINT SATISFACTION PROBLEM.

The main motivation behind this reduction is that there is a *clean* mapping between the maximum number of simultaneously satisfied constraints of  $\mathcal{F}$ , and the maximum size independent set of  $G_{\mathcal{F}}$ . In particular, one can easily prove the following well-known lemma.

**Lemma 4.2.** *Fix some integer  $f \in \mathbb{N}^+$ . Let  $\mathcal{F}$  be a CONSTRAINT SATISFACTION PROBLEM instance over  $n$  variables and a collection of constraints  $\mathcal{C}$ , such that for every  $C \in \mathcal{C}$ ,*

<sup>3</sup>Feige, Goldwasser, Lovász, Safra and Szegedy came up with this reduction in [43], and hence the name FGLSS.

<sup>4</sup>Recall that a constraint  $C_{S,A}$  is indexed by the set  $S \subseteq [n]$  of variables indices, and the literals set  $A \in \{0,1\}^{|S|}$ . See Section 2.1.

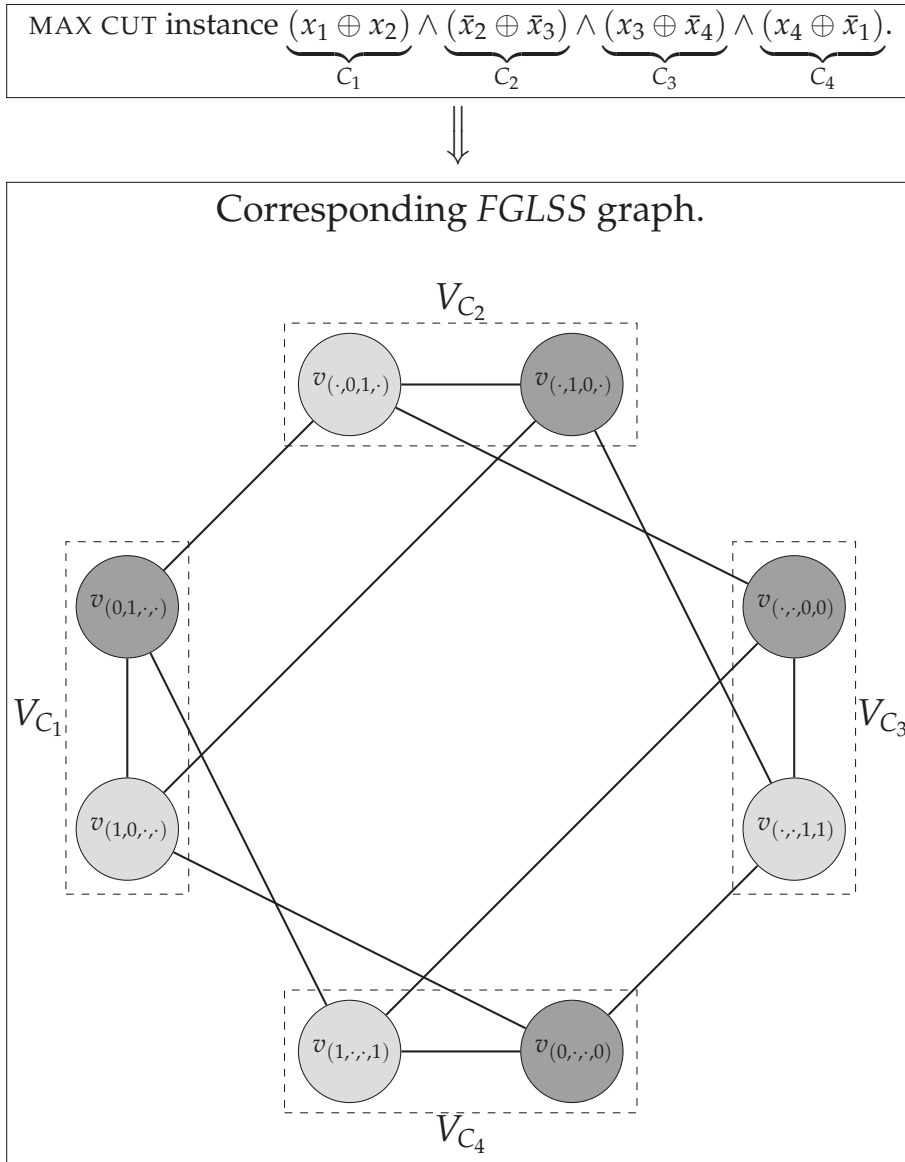


Figure 4.1 – Example of an *FGLSS* graph corresponding to a MAX CUT instance. A MAX CUT constraint  $(a \oplus b)$  has two satisfying assignment, namely  $(a = 1, b = 0)$  and  $(a = 0, b = 1)$ . Hence each set  $V_C$  has 2 vertices corresponding to those partial satisfying assignments. We have an edge between every two vertices corresponding to conflicting partial assignments. For example, we have an edge between  $v_{(.,1,0,.)}$  in  $V_{C_2}$  and  $v_{(.,.,1,1)}$  in  $V_{C_3}$  since they give a different assignment for the variable  $x_3$ . Note that this means that we have an edge between every two vertices of a set  $V_C$ , since their corresponding partial assignments assign different values for the same variables. The dark-grey colored vertices form an independent set since they all agree with the global assignment  $(x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 0)$  that satisfies the starting MAX CUT instance.



## 4.2. From CONSTRAINT SATISFACTION PROBLEMS to Graphs

$|C^{-1}(1)| = f$ . Moreover, let  $G_{\mathcal{F}} = (V, E)$  be its corresponding FGLSS graph. Then the following holds:

1. (Completeness: ) Assume there exists  $x \in \{0,1\}^n$  such that  $\mathcal{F}(x) \geq c \in [0,1]$ . Then there exists an independent set  $I \subseteq V$ , such that  $\frac{|I|}{|V|} \geq \frac{c}{f}$ .
2. (Soundness: ) Assume that for every  $x \in \{0,1\}^n$ ,  $\mathcal{F}(x) < s \in [0,1]$ . Then for every independent set  $I \subseteq V$ ,  $\frac{|I|}{|V|} < \frac{s}{f}$ .

*Proof.* We start by proving the completeness of the reduction. Towards this end, let  $x \in \{0,1\}^n$  be an assignment of the variables such that for some  $c \in [0,1]$ , we have

$$\mathcal{F}(x) = \frac{1}{|\mathcal{C}|} \sum_{C_{S,A} \in \mathcal{C}} C_{S,A}(x) \geq c,$$

and define the subset  $I \subseteq V$  of vertices as follows:

$$S = \{v_{S,A,\alpha} : \alpha \text{ agrees with } x \text{ on } S\}.$$

We say that  $\alpha$  agrees with  $x$  on  $S$ , if  $\alpha$  is the projection of  $x$  on the variables in  $S$ .

By construction of the vertex set of  $G_{\mathcal{F}}$ , we know that for every  $C_{S,A} \in \mathcal{C}$ ,  $V_{S,A}$  contains vertices corresponding to all the partial satisfying assignments of  $C_{S,A}$ . In particular, for every constraint  $C_{S,A}$  such  $C_{S,A}(x) = 1$ , the partial assignment  $x_S \in \{0,1\}^{|S|}$  corresponding to the projection of  $x$  on  $S$ , has a corresponding vertex  $v_{S,A,x_S}$  in  $V_{S,A}$ . By the definition of  $I$ , it follows that for such satisfied constraints  $C_{S,A}$ ,  $I \cap V_{S,A} = \{v_{S,A,x_S}\}$ . Since  $|C^{-1}(1)| = f$  for every  $C \in \mathcal{C}$ , we get that that

$$\frac{|I|}{|V|} = \frac{|\{C \in \mathcal{C} : C(x) = 1\}|}{|V|} = \frac{|\{C \in \mathcal{C} : C(x) = 1\}|}{f|\mathcal{C}|} \leq \frac{c|\mathcal{C}|}{f|\mathcal{C}|} = \frac{c}{f}.$$

To see that  $I$  is indeed a independent set, it is enough to see that since all the partial assignments in  $I$  agree with a global assignment  $x$ , then no conflict can occur between them, and hence no edge is completely contained inside  $I$ .

To prove the soundness, fix an instance  $\mathcal{F}$  such that for every  $x \in \{0,1\}^n$ ,  $\mathcal{F}(x) < s \in [0,1]$ , and assume towards contradiction that there still exists some independent set  $I \subseteq V$  such that  $\frac{|I|}{|V|} \geq \frac{s}{f}$ . We will show that we can derive from  $I$ , an assignment  $x_I \in \{0,1\}^n$  such that  $\mathcal{F}(x_I) \geq s$ , yielding a contradiction.

Since  $I$  is independent set, we know that no edge is completely contained in  $I$ , and hence no conflict occurs between any of the partial assignments corresponding to vertices in  $I$ . Thus there exists a global assignment  $x \in \{0,1\}^n$  such that  $x$  agrees with  $\alpha$  on  $S$  for every  $v_{S,A,\alpha} \in I$ . This implies that for every  $C_{S,A} \in \mathcal{C}$  such that  $|V_{S,A} \cap I| \geq 1$ ,  $C_{S,A}(x) = 1$ .

## Chapter 4. LP Hardness of Vertex Cover

---

But for every constraint  $C_{S,A} \in \mathcal{C}$ , the vertices in  $V_{S,A}$  correspond to different partial assignments over the same set of vertices  $S$ , thus  $G[V_{S,A}]$ , the induced subgraph on  $V_{S,A}$  is a clique, and any independent can contain at most one vertex per set  $V_{S,A}$ . In other words,  $C_{S,A} \in \mathcal{C}$ ,  $|V_{S,A} \cap I| \in \{0, 1\}$ , and hence the number of satisfied constraints by  $x$  is the same as the cardinality of  $I$ . Combining all the previous observations, we get :

$$\mathcal{F}(x) = \frac{|\{C_{S,A} : C_{S,A}(x) = 1\}|}{|\mathcal{C}|} = \frac{|I|}{|\mathcal{C}|} = f \cdot \frac{|I|}{|V|} \geq s.$$

□

Recall that the complement of an independent set  $I \subseteq V$ , i.e.,  $V \setminus I$ , is vertex cover of  $G$ . Thus assume that, in a computational model where the FGLSS reduction can be applied<sup>5</sup>, we can prove that for some CONSTRAINT SATISFACTION PROBLEM  $\Pi_n$  and for some  $0 < s \leq c < 1$ , we *cannot* distinguish between

1. instances such that there exists an assignment that satisfies at least  $c$  fraction of the constraints, or
2. instances where no assignment satisfies more than  $s$  fraction of the constraints.

In other words, we have a  $(c, s)$ -hardness of  $\Pi_n$  in a computational model where the FGLSS reduction is allowed. Thus intuitively, Lemma 4.2 immediately implies a hardness of  $\frac{1-s}{1-c} = \frac{f-s}{f-c}$  for the VERTEX COVER problem in the same computational model, where  $f \in \mathbb{N}^+$  is as in Lemma 4.2.

Since the free bit complexities of  $1F\text{-CSP}_n$  and  $\kappa\text{-NOR}_{n,2}$  are *one* and *zero* respectively, it is reasonable to hope for a framework for LP-reduction in which, an *FGLSS-like* reduction should imply that

$$\begin{aligned} (c, s)\text{-hardness of } 1F\text{-CSP}_n &\iff \frac{2-s}{2-c}\text{-hardness of VERTEX COVER} \\ (c, s)\text{-hardness of } \kappa\text{-NOR}_{n,2} &\iff \frac{1-s}{1-c}\text{-hardness of VERTEX COVER.} \end{aligned}$$

This is indeed what the LP-reduction framework of [25] provides, and the LP-hardness of 2 for the VERTEX COVER problem that we present in Section 4.4 will follow from our  $(1 - \epsilon, \epsilon)$ -hardness of  $1F\text{-CSP}_n$  using a reduction in the same spirit as the FGLSS graph in the framework of [25]. We could also reach the same hardness if we start from our  $(\frac{1}{2} - \epsilon, \epsilon)$ -hardness of  $\kappa\text{-NOR}_{n,2}$ , however we only use the hardness of the

---

<sup>5</sup>For instance, to prove NP-hardness or UGC-hardness, the underlying reduction is only required to be computable in polynomial time. However, as we will see in Definition 4.4 in Section 4.3, LP-hardness results require more restricted reductions.

aforementioned problem for arbitrary domain  $q$  to prove the hardness of the  $q$ -UNIFORM-VERTEX-COVER problem.

### 4.2.2 Reduction to Hypergraphs

In order to establish a hardness for the  $q$ -UNIFORM-VERTEX-COVER problem, we develop a more involved reduction that can be seen as an extension to the FGLSS reduction when the starting CONSTRAINT SATISFACTION PROBLEM is the (not necessarily binary)  $\kappa$ -NOR.

Recall that a hypergraph  $H = (V, E)$  over the vertex set  $V$  and the *hyperedge* set  $E$  is said to be  $q$ -uniform, if every hyperedge  $e \in E$  covers exactly  $q$  vertices, i.e.,  $\forall e \in E, |e| = q$ . We say that an edge  $e \in E$  is hit by a subset  $S \subseteq V$  if  $|e \cap S| \geq 1$ . The notion of vertex cover readily generalizes to hypergraphs as follows: We call a subset  $S \subseteq V$  of vertices a vertex cover of  $H$ , if every edge  $e \in E$  is hit by  $S$ . The goal in the  $q$ -UNIFORM-VERTEX-COVER problem is then to find the minimum size subset  $S \subseteq V$  of vertices such that  $S$  is a vertex cover of the hypergraph  $H$ .

We now highlight our reduction from  $\kappa$ -NOR $_{n,q}$  to  $q$ -UNIFORM-VERTEX-COVER, for any integer  $q \geq 2$ . Let  $\mathcal{F}$  be a  $\kappa$ -NOR $_{n,q}$  instance over  $n$  variables  $x_1, \dots, x_n \in \mathbb{Z}_q$ , and a collection of  $m$  constraints  $\mathcal{C} = \{C_{S_1, A_1}, \dots, C_{S_m, A_m}\}$ , such that every constraint  $C_{S, A} \in \mathcal{C}$  is of predicate type  $\kappa$ -NOR :  $\mathbb{Z}_q^k \mapsto \{0, 1\}$ . Recall that the  $\kappa$ -NOR predicate has a *zero* free bit complexity.

Given  $\mathcal{F}$ , we construct the hypergraph  $H_{\mathcal{F}} = (V, E)$  as follows:

**Vertex set:** For every constraint  $C_{S, A} \in \mathcal{C}$ , we have one representative vertex  $v_{S, A}$ , i.e.,

$$V = \bigcup_{C_{S, A} \in \mathcal{C}} v_{S, A}.$$

Since  $C_{S, A}$  has exactly one partial satisfying assignment which is  $A$ ,  $v_{S, A}$  implicitly corresponds to the unique partial assignment  $\alpha = A$  of the variables in  $S$  satisfying  $C_{S, A}$ .

**Edge set:** Any  $q$  vertices  $v_{S_1, A_1}, \dots, v_{S_q, A_q}$  are connected with a hyperedge if there exists a variable  $x_i, i \in \bigcap_{j=1}^q S_j$ , such that no two out of the  $q$  corresponding constraints check  $x_i$  versus the same  $a \in [q]$ . Formally, we have a hyperedge  $(v_{S_1, A_1}, \dots, v_{S_q, A_q})$  if there exists a variable  $x_i$  with  $i \in \bigcap_{j=1}^q S_j$  such that  $v_{S_l, A_l}(i) \neq v_{S_m, A_m}(i)$  for all  $l, m \in [q]$  with  $l \neq m$ , where for a vertex  $v_{S, A}$  and an index  $i \in S$ ,  $v_{S, A}(i) \in [q]$  denotes the value  $a \in [q]$  against which  $x_i$  is compared in the constraint  $C_{S, A}$ .

Note that for  $q = 2$ , this boils down to the FGLSS reduction starting from  $\kappa$ -NOR $_{n,2}$ , by observing that in this case each constraint  $C_{S, A}$  has exactly one partially satisfying

## Chapter 4. LP Hardness of Vertex Cover

---

assignment, and hence its corresponding vertex  $v_{S,A}$  can be thought of as the only vertex  $\{v_{S,A,A}\} = V_{S,A}$  in the previous FGLSS reduction. The edges in this case only capture conflicting assignments in the natural way.

We can prove the following lemma, which is an analogue of Lemma 4.2 for the hypergraph case:

**Lemma 4.3.** *Let  $\mathcal{J}$  be a  $\mathcal{K}$ -NOR $_{n,q}$  instance over  $n$  variables and a collection of constraints  $\mathcal{C}$ , and let  $H_{\mathcal{J}} = (V, E)$  be its corresponding hypergraph. Then the following holds:*

1. (Completeness: ) *Assume there exists  $x \in \mathbb{Z}_q^n$  such that  $\mathcal{J}(x) \geq c \in [0, 1]$ . Then there exists an independent set  $I \subseteq V$ , such that  $\frac{|I|}{|V|} \geq c$ .*
2. (Soundness: ) *Assume that for every  $x \in \mathbb{Z}_q^n$ ,  $\mathcal{J}(x) < s \in [0, 1]$ . Then for every independent set  $I \subseteq V$ ,  $\frac{|I|}{|V|} < s$ .*

We delegate the proof of Lemma 4.3 to Section 4.5, where we prove it in the context of LP hardness. However, one can infer from the statement that a  $(1 - \frac{1}{q} - \epsilon, \epsilon)$ -hardness of the  $\mathcal{K}$ -NOR $_{n,q}$  problem would then yield a hardness of  $\frac{1 - (1 - \frac{1}{q} - \epsilon)}{1 - \epsilon} \approx q$  for the  $q$ -UNIFORM-VERTEX-COVER problem.

### 4.3 LP Reduction Framework

In the next section we establish LP-hardness of VERTEX COVER and INDEPENDENT SET via a reduction from 1F-CSP (see Definition 2.6).

We will now briefly introduce a formal framework for reducing between problems that is a stripped down version of the framework due to Braun *et al*, with a few notational changes; the interested reader is referred to [25] for more details.

In this framework problems can be naturally reduced to each other. We will use the following restricted form of reductions.

**Definition 4.4.** Let  $\Pi_1 = (\mathcal{S}_1, \mathcal{J}_1)$  be a maximization problem and  $\Pi_2 = (\mathcal{S}_2, \mathcal{J}_2)$  be a minimization problem. A *reduction from  $\Pi_1$  to  $\Pi_2$*  consists of two maps, one  $\mathcal{J}_1 \mapsto \mathcal{J}_2$  from  $\mathcal{J}_1$  to  $\mathcal{J}_2$  and the other  $\mathcal{S}_1 \mapsto \mathcal{S}_2$  from  $\mathcal{S}_1$  to  $\mathcal{S}_2$ , subject to

$$\text{Val}_{\mathcal{J}_1}(S_1) = \mu_{\mathcal{J}_1} - \zeta_{\mathcal{J}_1} \cdot \text{Cost}_{\mathcal{J}_2}(S_2) \quad \mathcal{J}_1 \in \mathcal{J}_1, S_1 \in \mathcal{S}_1,$$

where  $\mu_{\mathcal{J}_1}$  is called the *affine shift* and  $\zeta_{\mathcal{J}_1} \geq 0$  is a normalization factor.

We say that the reduction is *exact* if additionally

$$\text{OPT}(\mathcal{J}_1) = \mu_{\mathcal{J}_1} - \zeta_{\mathcal{J}_1} \cdot \text{OPT}(\mathcal{J}_2) \quad \mathcal{J}_1 \in \mathcal{J}_1.$$

The following result is a special case of a more general result by [25]. We give a proof for completeness.

**Theorem 4.5.** *Let  $\Pi_1$  be a maximization problem and let  $\Pi_2$  be a minimization problem. Suppose that there exists an exact reduction from  $\Pi_1$  to  $\Pi_2$  with  $\mu := \mu_{\mathcal{J}_1}$  constant for all  $\mathcal{J}_1 \in \mathcal{J}_1$ . Then,  $\text{fc}_+(\Pi_1, c_1, s_1) \leq \text{fc}_+(\Pi_2, \rho_2)$  where  $\rho_2 := \frac{\mu - s_1}{\mu - c_1}$  (assuming  $\mu > c_1 \geq s_1$ ).*

*Proof.* Let  $Ax \geq b$  by a  $\rho_2$ -approximate LP relaxation for  $\Pi_2 = (\mathcal{S}_2, \mathcal{J}_2)$ , with realizations  $x^{S_2}$  for  $S_2 \in \mathcal{S}_2$  and  $f_{\mathcal{J}_2} : \mathbb{R}^d \rightarrow \mathbb{R}$  for  $\mathcal{J}_2 \in \mathcal{J}_2$ . We use the same system  $Ax \geq b$  to define a  $(c_1, s_1)$ -approximate LP relaxation of the same size for  $\Pi_1 = (\mathcal{S}_1, \mathcal{J}_1)$  by letting  $x^{S_1} := x^{S_2}$  where  $S_2$  is the solution of  $\Pi_2$  corresponding to  $S_1 \in \mathcal{S}_1$  via the reduction, and similarly  $f_{\mathcal{J}_1} := \mu - \zeta_{\mathcal{J}_1} f_{\mathcal{J}_2}$  with  $\zeta_{\mathcal{J}_1} \geq 0$  where  $\mathcal{J}_2$  is the instance of  $\Pi_2$  to which  $\mathcal{J}_1$  is mapped by the reduction and  $\mu$  is the affine shift independent of the instance  $\mathcal{J}_1$ .

Then conditions (i) and (ii) of Definition 4.4 are automatically satisfied. It suffices to check (iii)' with our choice of  $\rho_2$ , for the given completeness  $c_1$  and soundness  $s_1$ . Assume that  $\text{OPT}(\mathcal{J}_1) \leq s_1$  for some instance  $\mathcal{J}_1$  of  $\Pi_1$ . Then

$$\begin{aligned}
 \text{LP}(\mathcal{J}_1) &= \mu - \zeta_{\mathcal{J}_1} \text{LP}(\mathcal{J}_2) && \text{(by definition of } f_{\mathcal{J}_1}, \text{ and since } \zeta_{\mathcal{J}_1} \geq 0) \\
 &\leq \mu - \frac{1}{\rho_2} \cdot \zeta_{\mathcal{J}_1} \cdot \text{OPT}(\mathcal{J}_2) && \text{(since } \text{OPT}(\mathcal{J}_2) \leq \rho_2 \text{LP}(\mathcal{J}_2)) \\
 &= \mu + \frac{\mu - c_1}{\mu - s_1} \cdot \underbrace{(\text{OPT}(\mathcal{J}_1) - \mu)}_{\leq s_1} && \text{(since the reduction is exact)} \\
 &\leq \mu + \frac{\mu - c_1}{\mu - s_1} \cdot (s_1 - \mu) \\
 &= c_1,
 \end{aligned}$$

as required. Thus  $Ax \geq b$  gives a  $(c_1, s_1)$ -approximate LP relaxation of  $\Pi_1$ . The theorem follows.  $\square$

We will also derive inapproximability of INDEPENDENT SET from a reduction between maximization problems. In this case the inapproximability factor obtained is of the form  $\rho_2 = \frac{\mu + c_1}{\mu + s_1}$ .

*Remark 4.6* (Reductions: the matrix view). In order to relate the above back to the slack matrix level for the reader familiar with extended formulations, let  $M_1$  be the slack matrix of problem  $\mathcal{P}_1$  and  $M_2$  the slack matrix of problem  $\mathcal{P}_2$ . A reduction then provides nonnegative matrices  $R, C$  with  $\mathbf{1}C = \mathbf{1}$  and a nonnegative column vector  $t$ , so that  $M_1 = R \cdot M_2 \cdot C + t\mathbf{1}$ , where  $\mathbf{1}$  is the all-one row vector. The matrices  $R, C$  and the vector  $t$  arise from the reduction maps and clearly  $\text{rk}_+(M_1) \leq \text{rk}_+(M_2) + 1$ ; we refer the interested reader to [25, Remark 4.3] for a reformulation of reductions on a matrix level.

## 4.4 LP-Hardness for Vertex Cover and Independent Set.

We will now reduce 1F-CSP to VERTEX COVER with the reduction mechanism outlined in the previous section, which will yield the desired LP hardness for the latter problem.

We start by recasting VERTEX COVER, INDEPENDENT SET and 1F-CSP in our language. The two first problems are defined on a fixed graph  $G = (V, E)$ .

**Problem 4.7** (VERTEX COVER( $G$ )). The set of feasible solutions  $\mathcal{S}$  consists of all possible vertex covers  $U \subseteq V$ , and there is one instance  $\mathcal{J} = \mathcal{J}(H) \in \mathfrak{J}$  for each induced subgraph  $H$  of  $G$ . For each vertex cover  $U$  we have  $\text{Cost}_{\mathcal{J}(H)}(U) := |U \cap V(H)|$  being the size of the induced vertex cover in  $H$ .

Note that the instances we consider have 0/1 costs, which makes our final result stronger: even restricting to 0/1 costs does not make it easier for LPs to approximate VERTEX COVER. Similarly, for the independent set problem we have:

**Problem 4.8** (INDEPENDENT SET( $G$ )). The set of feasible solutions  $\mathcal{S}$  consists of all possible independent sets of  $G$ , and there is one instance  $\mathcal{J} = \mathcal{J}(H) \in \mathfrak{J}$  for each induced subgraph  $H$  of  $G$ . For each independent set  $I \in \mathcal{S}$ , we have that  $\text{Val}_{\mathcal{J}(H)}(I) := |I \cap V(H)|$  is the size of the induced independent set of  $H$ .

Finally, we can recast 1F-CSP as follows. Let  $n, k \in \mathbb{N}$  be fixed, with  $k \leq n$ .

**Problem 4.9** (1F-CSP( $n, k$ )). The set of feasible solutions  $\mathcal{S}$  consists of all possible variable assignments, i.e., the vertices of the  $n$ -dimensional 0/1 hypercube and there is one instance  $\mathcal{J} = \mathcal{J}(\mathcal{C})$  for each possible set of constraints  $\mathcal{C} = \{C_1, \dots, C_m\}$ , where each constraint  $C \in \mathcal{C}$  is of type  $\mathcal{P}$ , and  $\mathcal{P} = \{P_1, \dots, P_q\}$  is the set of all one free bit predicates of arity  $k$ . As before, for an instance  $\mathcal{J} \in \mathfrak{J}$  and an assignment  $x \in \{0, 1\}^n$ ,  $\text{Val}_{\mathcal{J}}(x)$  is the fraction of constraints  $C_i$  that  $x$  satisfies (see Definition 2.6).

With the notion of LP relaxations and 1F-CSP from above we can now formulate LP-hardness of approximation for 1F-CSPs, which follows directly from Corollary 3.10 by the result of [31].

**Theorem 4.10.** *For every  $\varepsilon > 0$  there exists a constant arity  $k = k(\varepsilon)$  such that we have  $\text{fc}_+(1\text{F-CSP}(n, k), 1 - \varepsilon, \varepsilon) \geq 2^{n^{\Omega(1)}}$ .*

Following the approach in [25], we define a graph  $G$  over which we consider VERTEX COVER, which will correspond to our (family of) hard instances. This graph is a *universal* FGLSS graph as it encodes all possible choices of constraints simultaneously [43]. The constructed graph is similar to the one in [25], however now we consider *all* one free bit predicates and not just the MAX CUT predicate  $x \oplus y$ .

#### 4.4. LP-Hardness for Vertex Cover and Independent Set.

**Definition 4.11** (VERTEX COVER host graph). For fixed number of variables  $n$  and arity  $k \leq n$  we define a graph  $G^* = G^*(n, k)$  as follows. Let  $x_1, \dots, x_n$  denote the variables of the CSP.

*Vertices:* For every one free bit predicate  $P$  of arity  $k$ , ordered subset of indices  $S \subseteq [n]$  of size  $k$ , and literals assignment  $A \in \{0, 1\}^k$ , we have two vertices  $v_{P,S,A,1}$  and  $v_{P,S,A,2}$  corresponding to the two satisfying partial assignments for the one free bit constraint  $C_{P,k,2,1,n,S,A}$  in the language of section 2.1. For simplicity we identify the partial assignments with the respective vertices in  $G^*$ . Thus a partial assignment  $\alpha \in \{0, 1\}^S$  satisfying  $C$  has a corresponding vertex  $v_{C,\alpha} \in \{v_{P,S,A,1}, v_{P,S,A,2}\}$ .

*Edges:* Two vertices  $v_{C_1,\alpha_1}$  and  $v_{C_2,\alpha_2}$  are connected if and only if the corresponding partial assignments  $\alpha_1$  and  $\alpha_2$  are incompatible, i.e., there exists  $i \in S(C_1) \cap S(C_2)$  with  $\alpha_1(i) \neq \alpha_2(i)$ , where  $S(C)$  denotes the set of variables that  $S$  is applied to.

Note that the graph has  $k!(\binom{2^k}{2})2^{k+1}$  vertices, which is polynomial in  $n$  for fixed  $k$ . In order to establish LP-inapproximability of VERTEX COVER and INDEPENDENT SET it now suffices to define a reduction satisfying Theorem 4.5.

**Main Theorem 4.12.** *For every  $\varepsilon > 0$  and for every  $N$ , there exists a graph  $G$  with  $|V(G)| = N$  such that:*

1.  $\text{fc}_+(\text{VERTEX COVER}(G), 2 - \varepsilon) \geq 2^{N^{\Omega(1)}}$ , and
2.  $\text{fc}_+(\text{INDEPENDENT SET}(G), 1/\varepsilon) \geq 2^{N^{\Omega(1)}}$ .

*Proof.* Let  $k = k(\varepsilon)$  be a sufficiently large arity, chosen as in Theorem 4.10. Let  $n$  be the largest integer such that  $N \geq |V(G^*(n, k))|$ , that is,  $N \geq k!(\binom{2^k}{2})2^{k+1}$ . In particular, we have

$$n \geq \left\lfloor \sqrt[k]{\frac{N}{2\binom{2^k}{2}k^k}} \right\rfloor,$$

and so  $n = N^{\Omega(1)}$  for every fixed  $\varepsilon$ . Notice that we can make this lower bound on  $n$  as large as desired by taking  $N$  sufficiently large. In particular, we may assume that  $n \geq 1$ . Let  $G$  be any  $N$ -vertex graph that has  $G^*(n, k)$  as an induced subgraph.

We reduce 1F-CSP on  $n$  variables to VERTEX COVER over  $G \supseteq G^*(n, k)$ . Let  $\mathcal{P}$  be the set of all one free bit predicates of arity  $k$ , thus,  $|\mathcal{P}| = 2^k k!$ . For a 1F-CSP instance  $\mathcal{I}_1 := \mathcal{I}_1(\mathcal{C})$  and a set of constraints  $\mathcal{C} = \{C_1, \dots, C_m\}$ , let  $H(\mathcal{C})$  be the induced subgraph of  $G^*(n, k)$  on the set of vertices  $V(\mathcal{C})$  corresponding to the partial assignments satisfying some constraint in  $\mathcal{C}$ . So  $V(\mathcal{C}) = \{v_{P,S,A,i} \mid P \in \mathcal{P}, \text{ordered } S \subseteq [n], |S| \leq k, A \in \{0, 1\}^k, i = 1, 2\}$ .

## Chapter 4. LP Hardness of Vertex Cover

---

In Theorem 4.10 we have shown that no LP of size at most  $2^{n^{\Omega(1)}}$  can provide an  $(1 - \varepsilon, \varepsilon)$ -approximation for 1F-CSP for any  $\varepsilon > 0$ , provided the arity  $k$  is large enough. To prove that every LP relaxation with  $2 - \varepsilon$  approximation guarantee for VERTEX COVER has size at least  $2^{n^{\Omega(1)}} = 2^{N^{\Omega(1)}}$ , we provide maps defining a reduction from 1F-CSP to VERTEX COVER.

In the following, let  $\Pi_1 = (\mathcal{S}_1, \mathcal{I}_1)$  be the 1F-CSP problem and let  $\Pi_2 = (\mathcal{S}_2, \mathcal{I}_2)$  be the VERTEX COVER problem. In view of Definition 4.4, we map  $\mathcal{J}_1 = \mathcal{J}_1(\mathcal{C})$  to  $\mathcal{J}_2 = \mathcal{J}_2(H(\mathcal{C}))$  and let  $\mu := 2$  and  $\zeta_{\mathcal{J}_1} := \frac{1}{m}$  where  $m$  is the number of constraints in  $\mathcal{C}$ .

For a total assignment  $x \in \mathcal{S}_1$  we define  $U = U(x) := \{v_{C,\alpha} \mid \alpha \text{ satisfies } C \text{ and } x \text{ does not extend } \alpha\}$ . The latter is indeed a vertex cover: we only have edges between conflicting partial assignments, and all the partial assignments that agree with  $x$  are compatible with each other. Thus  $I = I(x) := \{v_{C,\alpha} \mid \alpha \text{ satisfies } C \text{ and } x \text{ extends } \alpha\}$  is an independent set and its complement  $U$  is a vertex cover.

We first verify the condition that  $\text{Val}_{\mathcal{J}_1}(x) = 2 - \frac{1}{m} \text{Cost}_{\mathcal{J}_2}(U(x))$  for all instances  $\mathcal{J}_1 \in \mathcal{I}_1$  and assignments  $x \in \mathcal{S}_1$ . Every constraint  $C$  in  $\mathcal{C}$  over the (possibly negated) variables in  $\{x_i \mid i \in S\}$  has exactly two representative vertices  $v_{C,\alpha_1}, v_{C,\alpha_2}$  where  $\alpha_1, \alpha_2 \in \{0, 1\}^S$  are the two partial assignments satisfying  $C$ . If an assignment  $x \in \mathcal{S}_1$  satisfies the constraint  $C$ , then exactly one of  $\alpha_1, \alpha_2$  is compatible with  $x$ . Otherwise, when  $C(x) = 0$ , neither of  $\alpha_1, \alpha_2$  do. This means that in the former case exactly one of  $v_{C,\alpha_1}, v_{C,\alpha_2}$  is contained in  $U$  and in the latter both  $v_{C,\alpha_1}$  and  $v_{C,\alpha_2}$  are contained in  $U$ . It follows that for any  $\mathcal{J}_1 = \mathcal{J}_1(\mathcal{C}) \in \mathcal{I}_1$  and  $x \in \mathcal{S}_1$  it holds

$$\text{Val}_{\mathcal{J}_1}(x) = 2 - \frac{1}{m} \text{Cost}_{\mathcal{J}_2}(U(x)).$$

In other words, for any specific  $\mathcal{C}$  the affine shift is 2, and the normalization factor is  $\frac{1}{m}$ .

Next we verify exactness of the reduction, i.e.,

$$\text{OPT}(\mathcal{J}_1) = 2 - \frac{1}{m} \text{OPT}(\mathcal{J}_2).$$

For this take an arbitrary vertex cover  $U \in \mathcal{S}_2$  of  $G$  and consider its complement. This is an independent set, say  $I$ . As  $I$  is an independent set, all partial assignments  $\alpha$  such that  $v_{C,\alpha} \in I$  are compatible and there exists a total assignment  $x$  that is compatible with each  $\alpha$  with  $v_{P,\alpha} \in I$ . Then the corresponding vertex cover  $U(x)$  is contained in  $U$ . Thus there always exists an optimum solution to  $\mathcal{J}_2$  that is of the form  $U(x)$ . Therefore, the reduction is exact.

It remains to compute the inapproximability factor via Theorem 4.5. We have

$$\rho_2 = \frac{2 - \varepsilon}{2 - (1 - \varepsilon)} \geq 2 - 3\varepsilon.$$



---

#### 4.5. LP-Hardness for $q$ -UNIFORM-VERTEX-COVER.

A similar reduction works for INDEPENDENT SET. This time, the affine shift is  $\mu = 0$  and we get an inapproximability factor of

$$\rho_2 = \frac{1 - \varepsilon}{\varepsilon} \geq \frac{1}{2\varepsilon},$$

for  $\varepsilon$  small enough. □

### 4.5 LP-Hardness for $q$ -UNIFORM-VERTEX-COVER.

We will now reduce  $\kappa$ -NOR to  $q$ -UNIFORM-VERTEX-COVER with the reduction mechanism outlined in Section 4.3, which will yield the desired LP hardness for the latter problem.

We start by recasting  $q$ -UNIFORM-VERTEX-COVER and  $\kappa$ -NOR in the language of Section 4.3. The first problem is defined on a fixed  $q$ -uniform hypergraph  $H = (V, E)$ .

**Problem 4.13** ( $q$ -UNIFORM-VERTEX-COVER( $H$ )). The set of feasible solutions  $\mathcal{S}$  consists of all possible vertex covers  $U \subseteq V$ , and there is one instance  $\mathcal{J} = \mathcal{J}(H') \in \mathfrak{I}$  for each induced subhypergraph  $H'$  of  $G$ . For each vertex cover  $U$  we have  $\text{Cost}_{\mathcal{J}(H')}(U) := |U \cap V(H')|$  being the size of the induced vertex cover in  $H'$ .

We similarly recast  $\kappa$ -NOR. Let  $n, q, k \in \mathbb{N}$  be fixed with  $k \leq n$ .

**Problem 4.14** ( $\kappa$ -NOR( $n, q, k$ )). The set of feasible solutions  $\mathcal{S}$  consists of all possible variable assignments, i.e., all possible elements in  $[q]^n$  and there is one instance  $\mathcal{J} = \mathcal{J}(\mathcal{P})$  for each possible set  $\mathcal{P} = \{P_1, \dots, P_m\}$  of  $\kappa$ -NOR predicates of arity  $k$ . As before, for an instance  $\mathcal{J} \in \mathfrak{I}$  and an assignment  $x \in [q]^n$ ,  $\text{Val}_{\mathcal{J}}(x)$  is the fraction of predicates  $P_i$  that  $x$  satisfies (see Definition 2.7).

With the notions of LP relaxations and  $\kappa$ -NOR from above, we can now formulate LP-hardness of approximation for  $\kappa$ -NORs, which follows directly from Theorem ?? by the result of [31] (See the discussion in [31] and Section 7 in [76]).

**Theorem 4.15.** *For every  $\varepsilon > 0$  and alphabet size  $q \geq 2$ , there exists a constant arity  $k = k(\varepsilon)$  such that for every sufficiently large  $n$  we have  $\text{fc}_+(\kappa\text{-NOR}(n, q, k), 1 - 1/q - \varepsilon, \varepsilon) \geq n^{\Omega(\log n / \log \log n)}$ .*

Similar to Section 4.4, we first define our host hypergraph, and then provide a reduction that will yield our hardness result for  $q$ -UNIFORM-VERTEX-COVER using Theorem 4.5.

**Definition 4.16** ( $q$ -UNIFORM-VERTEX-COVER host hypergraph). For fixed number of variables  $n$ , alphabet  $q$ , and arity  $k \leq n$  we define a hypergraph  $H^* = H^*(n, q, k)$  as follows. Let  $x_1, \dots, x_n$  denote the variables of the CSP.

## Chapter 4. LP Hardness of Vertex Cover

---

*Vertices:* For every subset  $S = \{i_1, \dots, i_k\} \subseteq [n]$ , and every value of  $A = (a_1, \dots, a_k) \in [q]^k$ , we have a vertex  $v_{S,A}$  corresponding to the  $\kappa$ -NOR predicate

$$P(x_{i_1}, \dots, x_{i_k}) = 1 \quad \text{if and only if} \quad \bigwedge_{j=1}^k (x_{i_j} \neq a_j).$$

For a vertex  $v_{S,A}$  and an index  $i \in S$ , we define  $v_{S,A}(i) \in [q]$  to be the value  $a \in [q]$  against which  $x_i$  is compared in the predicate.

*Hyperedges:* Any  $q$  vertices  $v_{S_1, A_1}, \dots, v_{S_q, A_q}$  are connected with a hyperedge if there exists a variable  $x_i$ ,  $i \in \bigcap_{j=1}^q S_j$ , such that no two out of the  $q$  predicates check  $x_i$  versus the same  $a \in [q]$ . Formally, we have a hyperedge  $(v_{S_1, A_1}, \dots, v_{S_q, A_q})$  if there exists a variable  $x_i$  with  $i \in \bigcap_{j=1}^q S_j$  such that  $v_{S_l, A_l}(i) \neq v_{S_m, A_m}(i)$  for all  $l, m \in [q]$  with  $l \neq m$ .

Similar to before the edges model which predicates (and assignments) are in conflict: for every hyperedge there exists an index  $i$ , so that each vertex in the hyperedge tests against a different  $a \in [q]$  and as there are  $q$  of those, there exists no assignment  $x$  that can satisfy the predicates belonging to the hyperedge simultaneously. Note that the graph has  $q^k \binom{n}{k}$  vertices, which is polynomial in  $n$  for fixed  $k$  and  $q$ . In order to establish LP-inapproximability of  $q$ -UNIFORM-VERTEX-COVER it now suffices to define a reduction satisfying Theorem 4.5.

**Main Theorem 4.17.** *For every  $\varepsilon > 0$ ,  $q \geq 2$  and for every sufficiently large  $n$ , there exists a hypergraph  $H$  with  $|V(H)| = n$  such that  $\text{fc}_+(q\text{-UNIFORM-VERTEX-COVER}(H), q - \varepsilon) \geq n^{\Omega(\log n / \log \log n)}$ .*

*Proof.* We reduce  $\kappa$ -NOR on  $n$  variables of alphabet  $[q]$  with sufficiently large arity  $k = k(\varepsilon)$  to  $q$ -UNIFORM-VERTEX-COVER over  $H := H^*(n, q, k)$ . For a  $\kappa$ -NOR instance  $\mathcal{F}_1 = \mathcal{F}_1(\mathcal{P})$  and a set of  $\kappa$ -NOR predicates  $\mathcal{P} = \{P_{S_1, A_1}, P_{S_2, A_2}, \dots, P_{S_m, A_m}\}$ , let  $H(\mathcal{P})$  be the induced subgraph of  $G$  on the set of vertices  $V(\mathcal{P}) = \{v_{S_i, A_i} \mid 1 \leq i \leq m\}$ .

Similarly to Section 4.4, we provide maps defining a reduction from  $\kappa$ -NOR to  $q$ -UNIFORM-VERTEX-COVER. The proof will then follow by combining Theorems 4.15 and 4.5.

In the following, let  $\Pi_1 = (\mathcal{S}_1, \mathcal{F}_1)$  be the  $\kappa$ -NOR problem and let  $\Pi_2 = (\mathcal{S}_2, \mathcal{F}_2)$  be the  $q$ -UNIFORM-VERTEX-COVER problem. In view of Definition 4.4, we map  $\mathcal{F}_1 = \mathcal{F}_1(\mathcal{P})$  to  $\mathcal{F}_2 = \mathcal{F}_2(H(\mathcal{P}))$  and let  $\mu := 1$  and  $\zeta_{\mathcal{F}_1} := \frac{1}{m}$  where  $m$  is the number of constraints in  $\mathcal{P}$ .

For a total assignment  $x \in \mathcal{S}_1$  we define  $U = U(x) := \{v_{S,A} : P_{S,A}(x) = 0\}$ . The latter is indeed a vertex cover. To see this, consider its complement  $I = I(x) := \{v_{S,A} \mid P_{S,A}(x) = 1\}$ . Since  $x$  satisfies all the constraints corresponding to vertices in  $I$  simultaneously, no hyperedge can be completely contained in  $I$ . Otherwise this would imply that there

exists a variable  $x_i$ , and  $q$  predicates  $P'_1, P'_2, \dots, P'_q \in \mathcal{P}$  requiring  $x_i \neq j$  for all  $j \in [q]$ , and yet are all simultaneously satisfied by  $x$ .

We first verify the condition that  $\text{Val}_{\mathcal{J}_1}(x) = 1 - \frac{1}{m} \text{Cost}_{\mathcal{J}_2}(U(x))$  for all instances  $\mathcal{J}_1 \in \mathcal{I}_1$  and assignments  $x \in \mathcal{S}_1$ . Every predicate  $P_{S,A}$  in  $\mathcal{P}$  over the variables in  $\{x_i \mid i \in S\}$  has exactly one representative vertex  $v_{S,A}$ , that will be inside  $U$  only if  $P_{S,A}(x) = 0$ , and hence our claim holds. In other words, for any specific  $\mathcal{P}$  the affine shift is 1, and the normalization factor is  $\frac{1}{m}$ .

Next we verify exactness of the reduction, i.e.,

$$\text{OPT}(\mathcal{J}_1) = 1 - \frac{1}{m} \text{OPT}(\mathcal{J}_2).$$

For this take an arbitrary vertex cover  $U \in \mathcal{S}_2$  of  $H$  and consider its complement. This is an independent set, say  $I$ . As  $I$  is an independent set<sup>6</sup>, we know that for any variable  $x_\ell$  with  $\ell \in \bigcup_{v_{S,A} \in I} S$ , there exists a least one  $\tilde{a}_{x_\ell} \in [q]$  such that  $x_\ell$  is *not checked* versus  $\tilde{a}_{x_\ell}$  in any of the predicates corresponding to vertices in  $I$ . Hence any assignment  $x$  setting each  $x_\ell$  to  $\tilde{a}_{x_\ell}$  sets  $P_{S,A}(x) = 1$  for all  $v_{S,A} \in I$ . Therefore the corresponding vertex cover  $U(x)$  is contained in  $U$  and so there always exists an optimum solution to  $\mathcal{J}_2$  that is of the form  $U(x)$ . Therefore, the reduction is exact.

It remains to compute the inapproximability factor via Theorem 4.5. We have

$$\rho_2 = \frac{1 - \varepsilon}{1 - (1 - 1/q - \varepsilon)} \geq q - \Theta(\varepsilon).$$

□

## 4.6 Upper bounds.

Here we give a size- $O(n)$  LP relaxation for approximating INDEPENDENT SET within a factor- $O(\sqrt{n})$ , which follows directly by work of Feige and Jozeph [44]. Note that this is strictly better than the  $n^{1-\varepsilon}$  hardness obtained assuming  $P \neq NP$  by [56]. This is possible because the *construction* of our LP is NP-hard while being still of small size, which is allowed in our framework.

Start with a greedy coloring of  $G = (V, E)$ : let  $I_1$  be any maximum size independent set of  $G$ , let  $I_2$  be any maximum independent set of  $G - I_1$ , and so on. In general,  $I_{j+1}$  is any maximum independent set of  $G - I_1 - \dots - I_j$ . Stop as soon as  $I_1 \cup \dots \cup I_j$  covers the whole vertex set. Let  $k \leq n$  denote the number of independent sets constructed, that is, the number of colors in the greedy coloring.

<sup>6</sup>In a hypergraph  $H = (V, E)$  a set  $I \subseteq V$  is said to be *independent* if no hyperedge of  $H$  is fully contained in  $I$ .

Feige and Jozeph [44] made the following observation:

**Lemma 4.18.** *Every independent set  $I$  of  $G$  has a nonempty intersection with at most  $\lfloor 2\sqrt{n} \rfloor$  of the color classes  $I_j$ .*

Now consider the following linear constraints in  $\mathbb{R}^V \times \mathbb{R}^k \simeq \mathbb{R}^{n+k}$ :

$$0 \leq x_v \leq y_j \leq 1 \quad \forall j \in [k], v \in I_j \quad (4.2)$$

$$\sum_{j=1}^k y_j \leq \lfloor 2\sqrt{n} \rfloor. \quad (4.3)$$

These constraints describe the feasible set of our LP for INDEPENDENT SET on  $G$ . Each independent set  $I$  of  $G$  is realized by a 0/1-vector  $(x^I, y^I)$  defined by  $x_v^I = 1$  iff  $I$  contains vertex  $v$  and  $y_j^I = 1$  iff  $I$  has a nonempty intersection with color class  $I_j$ . For an induced subgraph  $H$  of  $G$ , we let  $f_{\mathcal{I}(H)}(x, y) := \sum_{v \in V(H)} x_v$ . By Lemma 4.18,  $(x^I, y^I)$  satisfies (4.2)–(4.3). Moreover, we clearly have  $f_{\mathcal{I}(H)}(x^I, y^I) = |I \cap V(H)|$ . Let  $\text{LP}(\mathcal{I}(H)) := \max\{f_{\mathcal{I}(H)}(x, y) \mid (4.2), (4.3)\} = \max\{\sum_{v \in V(H)} x_v \mid (4.2), (4.3)\}$ .

**Lemma 4.19.** *For every induced subgraph  $H$  of  $G$ , we have*

$$\text{LP}(\mathcal{I}(H)) \leq \lfloor 2\sqrt{n} \rfloor \text{OPT}(\mathcal{I}(H)).$$

*Proof.* When solving the LP, we may assume  $x_v = y_j$  for all  $j \in [k]$  and all  $v \in I_j$ . Thus the LP can be rewritten

$$\max \left\{ \sum_{j=1}^k |I_j \cap V(H)| \cdot y_j \mid 0 \leq y_j \leq 1 \forall j \in [k], \sum_{j=1}^k y_j \leq \lfloor 2\sqrt{n} \rfloor \right\}.$$

Because the feasible set is a 0/1-polytope, we see that the optimum value of this LP is attained by letting  $y_j = 1$  for at most  $\lfloor 2\sqrt{n} \rfloor$  of the color classes  $I_j$  and  $y_j = 0$  for the others. Thus some color class  $I_j$  has weight at least  $1/\lfloor 2\sqrt{n} \rfloor$  of the LP value.  $\square$

By Lemma 4.19, constraints (4.2)–(4.3) provide a size- $O(n)$  factor- $O(\sqrt{n})$  LP relaxation of INDEPENDENT SET.

**Theorem 4.20.** *For every  $n$ -vertex graph  $G$ ,  $\text{fc}_+(\text{INDEPENDENT SET}(G), 2\sqrt{n}) \leq O(n)$ .*

Although the LP relaxation (4.2)–(4.3) is NP-hard to construct, it is allowed by our framework because we do not bound the time needed to construct the LP. To our knowledge, this is the first example of a polynomial-size extended formulation outperforming polynomial-time algorithms.

We point out that a factor- $n^{1-\epsilon}$  LP-inapproximability of INDEPENDENT SET holds in a different model, known as the *uniform model* [4, 23]. In that model, we seek an LP

relaxation that approximates *all* INDEPENDENT SET instances with the same number of vertices  $n$ . This roughly corresponds to solving INDEPENDENT SET by approximating the correlation polytope in some way, which turns out to be strictly harder than approximating the stable set polytope, as shown by our result above.

## 4.7 SDP-Hardness for Independent Set.

We saw in Section 4.4 how to obtain an LP-hardness for VERTEX COVER and INDEPENDENT SET, starting from an LP-hardness for the 1F-CSP problem. Restricting our starting CSP to have only *one free bit* is crucial for the VERTEX COVER problem, since each constraint is then represented by a *cloud* containing exactly two vertices in the resulting graph. In this case, an assignment satisfying *almost all* the constraints, corresponds to a vertex cover containing *slightly more than half* of the vertices (i.e., one vertex in almost all the clouds, and both vertices in the *unsatisfied* clouds), whereas if no assignment can simultaneously satisfy more than an  $\varepsilon$ -fraction of the constraints, then any vertex cover should contain *almost all* the vertices. This extreme behaviour of the resulting graph is necessary to obtain a gap of 2 for the VERTEX COVER problem.

However, if we are only interested in the INDEPENDENT SET problem, any CSP with a sufficiently large gap between the soundness and completeness can yield the desired LP-Hardness, by virtue of the well-known FGLSS reduction [43]. Formally speaking, given reals  $0 < s < c \leq 1$ , and any CSP problem  $\Pi(P, n, k)$ , where  $n$  is the number of variables and  $P$  is a predicate of arity  $k$ , and knowing that no small linear program can provide a  $(c, s)$ -approximation for this CSP, then one can show that no small LP can as well approximate the INDEPENDENT SET problem within a factor of  $c/s$ . This can be simply done by tweaking the reduction of Section 4.4 in a way that the number of vertices in each cloud is equal to the number of satisfying assignments for the predicate. Hence dropping the *one free bit* requirement, and restricting ourselves to CSPs such that  $c/s = 1/\varepsilon$  for arbitrarily small  $\varepsilon := \varepsilon(k) > 0$ , would yield the desired  $\omega(1)$  LP-hardness for the INDEPENDENT SET problem.

Moreover, the reduction framework of [25] and our construction in Section 4.4 are agnostic to whether we are proving LP or SDP lower bounds, and hence having an analog of Theorem 4.10 in the SDP world will yield that any SDP of size less than  $n^{\Omega(\log n / \log \log n)}$  has an integrality gap of  $\omega(1)$  for the INDEPENDENT SET problem. In fact such SDP-hardness results for certain families of CSPs and hence an analog of Theorem 4.10 are known: if our starting CSP has a predicate that supports pairwise independence with a sufficiently large arity  $k$ , then the result of [11] by virtue of [76] gives us the desired SDP base hardness. By the argumentation from above we obtain:

**Corollary 4.21.** *For every  $\varepsilon > 0$  and for every sufficiently large  $n$ , there exists a graph  $G$  with  $|V(G)| = n$ , such that no polynomial size SDP is a  $(1/\varepsilon)$ -approximate SDP relaxation for*

INDEPENDENT SET( $G$ ).

### 4.8 Conclusion

Recall that we proved in Chapter 3 that if any LP relaxation of a certain size for the 1F-CSP problem has an  $\omega(1)$  integrality gap, then any LP relaxation of the same size for the VERTEX COVER problem and the INDEPENDENT SET problem has an integrality gap of  $2 - o(1)$  and  $\omega(1)$  respectively. We also proved that any Sherali-Adams LP relaxation whose size is at most sub-exponential for the 1F-CSP problem has an integrality gap of  $\omega(1)$ . Our results then imply a quasi-polynomial lower bound version of Theorem 4.1 by using the result of [31] as a black-box, and any strengthening of the LP size lower bounds in [31] directly implies a strengthening of Theorem 4.1. For instance as we mentioned here, using the recent result of Kothari, Meka and Raghavendra [71] as a black-box instead of [31] already yields our LP lower bounds in Theorem 4.1 (i.e., an improvement from quasi-polynomial to sub-exponential). Similarly, any further improvement upon the bounds in [71] would strengthen our results as well.

Our lower bound for the generalization of the VERTEX COVER problem on  $q$ -uniform hypergraphs follows a similar route; the proof however uses a different intermediate CSP,  $\kappa$ -NOR, that is not necessarily binary. Contrary to [31], the result of [71] does not yet have any implications on non-binary CSPs. Nevertheless, any stronger LP lower bound for  $\kappa$ -NOR would again directly improve our lower bound for the the VERTEX COVER problem on  $q$ -uniform hypergraphs to  $2^{n^{\Omega(1)}}$  as well.

Subsequent to our work, Braun, Pokutta and Roy [26] strengthened the reduction framework of [25] that we also employ here in order to allow more complex reductions. Using the new framework, they were able to establish the same quasi-polynomial LP lower bounds for the CSPs that we consider (i.e., 1F-CSP and  $\kappa$ -NOR) without requiring intermediate Sherali-Adams gaps. In contrast to this work, their lower bounds cannot however be directly improved to sub-exponential by virtue of [71], as they use the UNIQUE GAMES problem (a non-Binary CSP) as a starting point in their reduction, and the only known to-date LP lower bound for that problem is still quasi-polynomial.

## 5 Knapsack

We have thus far proved LP lower bounds for combinatorial problems. In this chapter, we prove upper bounds for any LP relaxation that uses the so-called knapsack cover inequalities.

Initially developed for the MIN-KNAPSACK problem, the knapsack cover inequalities are used in the current best relaxations for numerous combinatorial optimization problems of covering type. In spite of their widespread use, these inequalities yield linear programming (LP) relaxations of exponential size, over which it is not known how to optimize exactly in polynomial time. In this chapter, we address this issue and obtain LP relaxations of quasi-polynomial size that are at least as strong as that given by the knapsack cover inequalities.

For the MIN-KNAPSACK cover problem, our main result can be stated formally as follows: for any  $\varepsilon > 0$ , there is a  $(1/\varepsilon)^{O(1)}n^{O(\log n)}$ -size LP relaxation with an integrality gap of at most  $2 + \varepsilon$ , where  $n$  is the number of items. Prior to this work, there was no known relaxation of subexponential size with a constant upper bound on the integrality gap.

Our construction is inspired by a connection between extended formulations and monotone circuit complexity via Karchmer-Wigderson games. In particular, our LP is based on  $O(\log^2 n)$ -depth monotone circuits with fan-in 2 for evaluating weighted threshold functions with  $n$  inputs, as constructed by Beimel and Weinreb. We believe that a further understanding of this connection may lead to more positive results that would complement the numerous lower bounds recently proved for extended formulations.

## 5.1 Introduction

*Capacitated covering problems*<sup>1</sup> play a central role in combinatorial optimization. These are the problems modeled by Integer Programs (IPs) of the form  $\min\{\sum_{i=1}^n c_i x_i \mid Ax \geq b, x \in \{0,1\}^n\}$ , where  $A$  is a size- $m \times n$  non-negative matrix and  $b, c$  size- $n$  non-negative vectors. The MIN-KNAPSACK problem is the special case arising when there is a single covering constraint, that is, when  $m = 1$ . This is arguably the simplest interesting capacitated covering problem.

In terms of complexity, the MIN-KNAPSACK problem is well-understood: on the one hand it is weakly NP-hard [66] and, on the other hand, it admits an FPTAS [75, 88]. However, for its own sake and as it appears as a key substructure of numerous other IPs, improving our polyhedral understanding of the problem is important. By this, we mean finding “good” linear programming (LP) relaxations for the MIN-KNAPSACK problem. Indeed, the polyhedral study of this problem has led to the development of important tools, such as the knapsack cover inequalities, for the strengthening of LP relaxations. These inequalities and the generalizations thereof are now used in the current best known relaxations for several combinatorial optimization problems, such as single-machine scheduling [10] and capacitated facility location [1]. However, despite this important progress in the past, many fundamental questions remain open – even in the most basic setting.

**State of the Art.** The feasible region of a MIN-KNAPSACK instance is specified by positive *item sizes*  $s_1, \dots, s_n$  and a positive *demand*  $D$ . In this context, a vector  $x \in \{0,1\}^n$  is *feasible* if  $\sum_{i=1}^n s_i x_i \geq D$ . To specify completely an instance of the MIN-KNAPSACK problem, we are further given non-negative *item costs*  $c_1, \dots, c_n$ . Solving the resulting instance then amounts to solving the IP  $\min\{\sum_{i=1}^n c_i x_i \mid \sum_{i=1}^n s_i x_i \geq D, x \in \{0,1\}^n\}$ .

The *basic* LP relaxation, i.e.,  $\min\{\sum_{i=1}^n c_i x_i \mid \sum_{i=1}^n s_i x_i \geq D, x \in [0,1]^n\}$ , provides an estimate on the optimum value that can be quite bad. More precisely, defining the *integrality gap* as the supremum over all instances of the ratio of the optimum value of the IP to the optimum value of the LP relaxation, it is easy to see that the integrality gap is unbounded.

Several inequalities have been proposed for strengthening this basic LP relaxation. Already in the 70’s, Balas [5], Hammer, Johnson and Peled [55] and Wolsey [106] independently proposed to add the *uncapacitated* knapsack cover inequalities: for every subset  $A \subseteq [n]$  of the items such that  $\sum_{i \in A} s_i < D$ , add the inequality  $\sum_{i \notin A} x_i \geq 1$  (saying that at least one item in  $[n] \setminus A$  needs to be picked in order to satisfy the demand). Unfortunately, these (exponentially many) inequalities are not sufficient for bringing down

---

<sup>1</sup>The term “capacitated” is used in the literature to emphasize that the entries of matrix  $A$  can take any non-negative value in contrast to the uncapacitated version where entries are Boolean.



the integrality gap to a constant. A strengthening of these inequalities was therefore proposed more recently by Carr, Fleischer, Leung and Philipps [29]. They defined the following valid inequalities: for every set of items  $A \subseteq [n]$  such that  $\sum_{i \in A} s_i < D$ , there is a corresponding (capacitated) *knapsack cover inequality*

$$\sum_{i \notin A} s'_i x_i \geq U, \quad (5.1)$$

where  $U = U(A) := D - \sum_{i \in A} s_i$  is the *residual demand* and  $s'_i = s'_i(A) := \min\{s_i, U\}$ . The validity of (5.1) is due to the fact that every feasible solution  $x \in \{0, 1\}^n$  has to contain some object  $i \notin A$ . This object can be *large*, that is, have  $s_i \geq U$ , and in this case the inequality is clearly satisfied. Otherwise, in case every object  $i \notin A$  is *small*, the total size of the objects  $i \notin A$  picked by  $x$  has to be at least the residual demand  $U$ .

Carr *et al.* [29] proved that whenever  $x \in \mathbb{R}_{\geq 0}^n$  satisfies all knapsack cover inequalities,  $2x$  dominates a convex combination of feasible solutions, that is, there exist feasible solutions  $x^{(j)} \in \{0, 1\}^n$  ( $j \in [q]$ ) and coefficients  $\lambda_j \geq 0$  summing up to 1 such that  $2x \geq \sum_{j=1}^q \lambda_j x^{(j)}$ . Given any non-negative item costs, one of the  $x^{(j)}$  will have a cost that is at most 2 times that of  $x$ . This implies that the integrality gap of the corresponding LP relaxation is at most 2.

The LP relaxation defined by the knapsack cover inequalities is “good” in the sense that it has a constant integrality gap. However, it has exponential *size*, that is, exponentially many inequalities, over which it is not known how to optimize exactly in polynomial time; in particular, it is not known how to employ the Ellipsoid algorithm because the problem of separating the knapsack cover inequalities reduces to another knapsack problem (which is NP-hard in general).

In contrast, for the MAX-KNAPSACK problem, Bienstock [19] proved that for all  $\varepsilon > 0$  there exists a size- $n^{O(1/\varepsilon)}$  LP relaxation whose integrality gap<sup>2</sup> is at most  $1 + \varepsilon$ . That LP is defined by an extended formulation that uses  $n^{O(1/\varepsilon)}$  extra variables besides the  $x$ -variables. We remark that it is a notorious open problem to prove or disprove the existence of a  $f(1/\varepsilon) \cdot n^{O(1)}$ -size LP relaxation for MAX-KNAPSACK with integrality gap at most  $1 + \varepsilon$ , see e.g. the survey on extended formulations by Conforti, Cornuéjols and Zambelli [38]. Coming back to the MIN-KNAPSACK problem, it is not known whether there exists a polynomial-size LP relaxation with constant integrality gap or not.<sup>3</sup>

<sup>2</sup>For maximization problems, one takes the supremum of the ratio of the optimum value of the LP relaxation to the optimum value of the IP.

<sup>3</sup>We remark that Bienstock and McClosky [20] considered the easier case when the relaxation is allowed to depend on the objective function to be optimized (i.e., on the cost of the items). In this case, using techniques similar to those developed for polynomial time approximation schemes, they obtained polynomial size relaxations with integrality gap at most  $1 + \varepsilon$ , for any fixed  $\varepsilon > 0$ . This is, however, a very different setting and, as the developed inequalities depend on the objective function, they do not generalize to other problems.

**Main Result.** We come close to resolving the question and show that MIN-KNAPSACK admits a quasi-polynomial-size LP relaxation with integrality gap at most  $2 + \varepsilon$ . The upper bound on the integrality gap originates from the fact that our LP relaxation is at least as strong as that provided by a slightly weakened form of the knapsack cover inequalities. We point out that, under some conditions, we can bound the size of our relaxation by a polynomial, see Section 5.3.2. A more precise statement of our main result is as follows.

**Theorem 5.1.** *For all  $\varepsilon \in (0, 1)$ , item sizes  $s_1, \dots, s_n \in \mathbb{R}_+$  and demand  $D \in \mathbb{R}_+$ , there exists a size- $(1/\varepsilon)^{O(1)} n^{O(\log n)}$  extended formulation defining an LP relaxation of MIN-KNAPSACK with integrality gap at most  $2 + \varepsilon$ .*

As the result is obtained by giving quasi-polynomially many inequalities of roughly the same strength as the exponentially many knapsack cover inequalities, our techniques also lead to relaxations of quasi-polynomial size for the numerous applications of these inequalities. We mention some of these applications below when we discuss related works.

Beyond the result itself, the novelty of our approach lies in the concepts we rely on and the techniques we develop. Our starting point is a connection between monotone circuits and extended formulations that we explain below. This connection was instrumental in the recent *lower bounds* of Göös, Jain and Watson on the extension complexity of independent set polytopes [52], and can be traced back to a paper of Hrubeš [61]. Here we use it for the first time to prove an *upper bound*.

**From Monotone Circuits to Extended Formulations.** Each choice of item sizes and demand gives rise to a *weighted threshold function*  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  defined as

$$f(x) := \begin{cases} 1 & \text{if } \sum_{i=1}^n s_i x_i \geq D \\ 0 & \text{otherwise.} \end{cases} \quad (5.2)$$

Since we assume that the item sizes and demand are non-negative,  $f$  is *monotone* in the sense that  $a \leq b$  implies  $f(a) \leq f(b)$ , for all  $a, b \in \{0, 1\}^n$ .

Clearly, we have that  $x \in \{0, 1\}^n$  is feasible if and only if  $x \in f^{-1}(1)$ . Furthermore, for  $a \in f^{-1}(0)$ , we can rewrite the uncapacitated knapsack cover inequalities as  $\sum_{i:a_i=0} x_i \geq 1$ . Consider the *slack matrix*  $S_{a,b} := \sum_{i:a_i=0} b_i - 1$  indexed by pairs  $(a, b) \in f^{-1}(0) \times f^{-1}(1)$ . By Yannakakis' factorization theorem [107], the existence of a size- $r$  LP relaxation of MIN-KNAPSACK that is at least as strong as that given by the uncapacitated knapsack cover inequalities is equivalent to the existence of a decomposition of the slack matrix  $S$  as a sum of  $r$  non-negative rank-1 matrices.

Now suppose that there exists a depth- $t$  monotone circuit (that is, using only AND gates

and OR gates) of fan-in 2 for computing  $f(x)$ . A result of Karchmer and Wigderson [64] then implies a partition of the entries of  $S$  into at most  $2^t$  rectangles<sup>4</sup>  $R \subseteq f^{-1}(0) \times f^{-1}(1)$  such that in each one of these rectangles  $R$ , there exists some index  $i^* = i^*(R)$  such that  $a_{i^*} = 0$  and  $b_{i^*} = 1$  for all  $(a, b) \in R$ . Then we may write, for  $(a, b) \in R$ ,

$$S_{a,b} = \sum_{i:a_i=0} b_i - 1 = \sum_{i:a_i=0, i \neq i^*} b_i = \sum_{i \neq i^*} (1 - a_i) b_i, \quad (5.3)$$

so that  $S$  restricted to the entries of  $R$  can be expressed as a sum of at most  $n - 1$  non-negative rank-1 matrices of the form  $((1 - a_i) b_i)_{(a,b) \in R}$ , where  $i$  is a fixed index distinct from  $i^*$ . This implies a decomposition of the whole slack matrix  $S$  as a sum of at most  $2^t(n - 1)$  non-negative rank-1 matrices, and thus the existence of a  $2^t(n - 1)$ -size LP relaxation of MIN-KNAPSACK that captures the uncapacitated knapsack cover inequalities. Since  $f$  is a weighted threshold function, we can take  $t = O(\log^2 n)$ , as proved by Beimel and Weinreb [16]. Therefore, we obtain a  $n^{O(\log n)}$ -size extended formulation for the uncapacitated knapsack cover inequalities. Unfortunately, these inequalities do not suffice to guarantee a bounded integrality gap.

For the full-fledged knapsack cover inequalities (5.1), the simple idea described above breaks down. If the special index  $i^* = i^*(R)$  for some rectangle  $R$  corresponds to a large object, we can write

$$\sum_{i:a_i=0} s'_i b_i - U = \sum_{i:a_i=0, i \neq i^*} s'_i b_i = \sum_{i \neq i^*} s'_i (1 - a_i) b_i,$$

where each matrix  $(s'_i (1 - a_i) b_i)_{(a,b) \in R}$  has rank at most 1 because  $s'_i (1 - a_i)$  depends on  $a$  only. However,  $i^*$  may correspond to a small object, in which case we cannot decompose the slack matrix as above.

Nevertheless, we prove that it is possible to overcome this difficulty. Two key ideas we use to achieve this are to discretize some of the quantities (which explains why we lose an  $\varepsilon$  in the integrality gap) and resort to several weighted threshold functions instead of just one. If all these functions admit  $O(\log n)$ -depth monotone circuits of fan-in 2, then we obtain a size- $n^{O(1)}$  LP relaxation.

**Related Works.** Knapsack cover inequalities and their generalizations such as flow cover inequalities were used as a systematic way to *strengthen* LP formulations of other (seemingly unrelated) problems [29, 28, 78, 6, 7, 30, 10, 35, 41]. By strengthening we mean that one would start with a polynomial size LP formulation with a potentially unbounded integrality gap for some problem of interest, and then show that adding (adaptations) of knapsack cover inequalities reduces this integrality gap (we illustrate in Section 5.4 how this strengthening works for the Single Demand Facility Location

<sup>4</sup>A *rectangle* is the Cartesian product of a set of row indices and a set of column indices.

problem, reducing the integrality gap down to 2). However, similar to the case of MIN-KNAPSACK discussed above, the drawback of this approach is that the size of the resulting LP formulation becomes exponential. We can extend our result to show that it yields quasi-polynomial size LP formulation for many such applications. To name a few:

- Carr *et al.* [29] applied these inequalities to the Generalized Vertex Cover problem, Multi-color Network Design problem and the Fixed Charge Flow problem, and showed how these inequalities reduce the integrality gap of the starting LP formulations.
- Bansal and Pruhs [10] studied the Generalized Scheduling Problem (GSP) that captures many interesting scheduling problems such as Weighted Flow Time, Flow Time Squared and Weighted Tardiness. In particular, they showed a connection between GSP and a certain geometric covering problem, and designed an LP based approximation algorithm for the latter that yields an approximate solution for the GSP. The LP formulation that they use for the intermediate geometric cover problem is strengthened using knapsack cover inequalities, and yields an  $O(\log \log nP)$ -approximation for the GSW where  $n$  is the number of jobs, and  $P$  is the maximum job size. In the special case of identical release time of the jobs, their LP formulation yields a 16-approximation algorithm. This constant factor approximation was later improved by Cheung and Shmoys [35] and Mestre and Verschae [82] to a  $(4 + \epsilon)$ -approximation, where the authors added the knapsack cover inequalities directly to the LP formulation of the scheduling problem, i.e., without resorting to the intermediate geometric cover problem as in [10]. For both the GSP and its special case, our method yields an LP formulation whose size is quasi-polynomial in  $n$ , and polynomial in both  $\log P$  and  $\log W$ , where  $W$  is the maximum increase in the cost function of a job at any point in time.
- Efsandiari *et al.* [41] used a knapsack-cover-strengthened LP formulation to design an  $O(\log k)$ -approximation algorithm for Precedence-Constrained Single-Machine Deadline scheduling problem, where  $k$  is the number of distinct deadlines.
- Carnes and Shmoys [28] designed primal-dual algorithms for the Single-Demand Facility Location, where the primal LP formulation is strengthened by adding (generalizations) of knapsack cover inequalities.

Extended formulations have received a considerable amount of attention recently, mostly for proving impossibility results. Pokutta and Van Vyve [91] proved a worst-case  $2^{\Omega(\sqrt{n})}$  size lower bound for extended formulations of the MAX-KNAPSACK polytope, which directly implies a similar result for the MIN-KNAPSACK polytope. Other recent works include [45, 21, 31, 94, 76, 13].

**Outline.** We prove our main result in Section 5.3, after giving preliminaries in Section 5.2. Instead of explicitly constructing our extended formulation, we provide a non-negative factorization of the appropriate slack matrix. For this, we use the language of communication complexity — we give an  $O(\log^2 n + \log(1/\varepsilon))$ -complexity two-party communication protocol with private randomness and non-negative outputs whose expected output is the slack of a given feasible solution with respect to a given (weakened) knapsack cover inequality.

Next, in Section 5.4, we extend our communication protocol to the flow cover inequalities for the Single-Demand Facility Location problem, and show how to approximate the exponentially many flow cover inequalities using a smaller LP formulation.

Finally, in Section 5.5, we show that although we do not know how to write down our extended formulation for MIN-KNAPSACK in quasi-polynomial time, we can at least compute a  $(2 + \varepsilon)$ -approximation of the optimum from the extended formulation in quasi-polynomial time, given any cost vector, *without* relying on the ellipsoid algorithm. This is done via a new cutting-plane algorithm that might be of independent interest.

## 5.2 Preliminaries.

In this section, we introduce some key notions related to our problem. We review the relation between extended formulations and extension complexity of pairs of polyhedra, and the non-negative factorization of slack matrices in Section 5.2.1. Next, we define randomized communication protocols with non-negative outputs that compute entries of matrices in expectation. Finally, in Section 5.2.3, we review some constructions of low-depth monotone circuits, and the Karchmer-Wigderson game that relates circuit complexity and communication complexity.

### 5.2.1 Polyhedral Pairs, Extended Formulations and Slack Matrices.

**Definition 5.2.** Let  $(P, Q)$  be a polyhedral pair with  $P \subseteq Q \subseteq \mathbb{R}^n$ . Let  $P = \text{conv}(\{v_1, \dots, v_p\})$  be an inner description of  $P$  and  $Q = \{x \in \mathbb{R}^n \mid Ax \geq b\}$  be an outer description of  $Q$ , where  $A \in \mathbb{R}^{m \times n}$  and  $b \in \mathbb{R}^m$ . We now define the slack matrix  $S$  of the pair  $(P, Q)$  with respect to the given representations of  $P$  and  $Q$ . The  $i$ th row of  $S$  corresponds to the constraint  $A_i x \geq b_i$ , while the  $j$ th column of  $S$  corresponds to the point  $v_j$ . The value  $S_{i,j}$  measures how close the constraint  $A_i x \geq b_i$  is to being tight for point  $v_j$ . More specifically, the *slack matrix*  $S \in \mathbb{R}_{\geq 0}^{m \times p}$  is defined as  $S_{i,j} := A_i v_j - b_i$  for all  $i \in [m], j \in [p]$ .

Note that the slack matrix is not unique as it depends on the choices of points  $v_1, \dots, v_p$  and linear description  $Ax \geq b$ .

**Definition 5.3.** Given a non-negative matrix  $M \in \mathbb{R}_{\geq 0}^{m \times n}$ , we say that a pair of matrices

$T, U$  is a rank- $r$  non-negative factorization of  $M$  if  $T \in \mathbb{R}_{\geq 0}^{m \times r}$ ,  $U \in \mathbb{R}_{\geq 0}^{r \times n}$ , and  $M = TU$ . We define the non-negative rank of  $M$  as

$$\text{rk}_+(M) := \min\{r : M \text{ has a rank-}r \text{ non-negative factorization}\}.$$

Notice that a non-negative factorization of  $M$  of rank at most  $r$  is equivalent to a decomposition of  $M$  as a sum of at most  $r$  non-negative rank-1 matrices.

Yannakakis [107] proved that for a polytope  $P$  of dimension at least 1 and any of its slack matrices  $S$ , the extension complexity of  $P$  is equal to the non-negative rank of  $S$ . Namely,  $\text{xc}(P) = \text{rk}_+(S)$ . In particular, all the slack matrices of  $P$  have the same nonnegative rank.

This factorization theorem can be extended to polyhedral pairs: we have  $\text{xc}(P, Q) \in \{\text{rk}_+(S), \text{rk}_+(S) - 1\}$  whenever  $S$  is a slack matrix of  $(P, Q)$ , see e.g. [21].

### 5.2.2 Randomized Communication Protocols.

We now define a certain two-party communication problem and relate it to the non-negative rank discussed earlier, following the framework in Faenza, Fiorini, Grappe and Tiwary [42].

**Definition 5.4.** Let  $S \in \mathbb{R}_{\geq 0}^{\mathcal{A} \times \mathcal{B}}$  be a non-negative matrix whose rows and columns are indexed by  $\mathcal{A}$  and  $\mathcal{B}$ , respectively. Let  $\Pi$  be a communication protocol with private randomness between two players Alice and Bob. Alice gets an input  $a \in \mathcal{A}$  and Bob gets an input  $b \in \mathcal{B}$ . They exchange bits in a pre-specified way according to  $\Pi$ , and at the end either one of the players outputs some non-negative number  $\zeta \in \mathbb{R}_{\geq 0}$ . We say that  $\Pi$  computes  $S$  in expectation if for every  $a$  and  $b$ , the expectation of the output  $\zeta$  equals  $S_{a,b}$ .

The communication complexity of a protocol  $\Pi$  is the maximum of the number of bits exchanged between Alice and Bob, over all pairs  $(a, b) \in \mathcal{A} \times \mathcal{B}$  and the private randomness of the players. The size of the final output does not count towards the communication complexity of a protocol. The communication complexity of  $S$ , denoted  $R_{\text{exp}}^{\text{cc}}(S)$  is the minimum communication complexity of a randomized protocol  $\Pi$  computing  $S$  in expectation.

Faenza *et al.* [42] relate the non-negative rank of a non-negative matrix  $S$ , to the communication complexity  $R_{\text{exp}}^{\text{cc}}(S)$ . In particular, they prove that if  $\text{rk}_+(S) \neq 0$ , then  $R_{\text{exp}}^{\text{cc}}(S) = \log_2 \text{rk}_+(S) + \Theta(1)$ . Combining this with the factorization theorem, we get  $R_{\text{exp}}^{\text{cc}}(S) = \log_2 \text{xc}(P, Q) + \Theta(1)$  whenever  $(P, Q)$  is a polyhedral pair with slack matrix  $S$ , provided that  $\text{xc}(P, Q) \neq 0$ .

### 5.2.3 Weighted Threshold Functions and Karchmer-Wigderson Game.

An important part of our protocol depends on the communication complexity of (monotone) weighted threshold functions. We start with the following result from [15, 16] which gives low-depth circuits for such functions. Another construction was given in [34]. The circuits as stated in [15, 16, 34] have logarithmic depth, polynomial size and unbounded fan-in, thus it is straightforward to convert them into circuits with fan-in 2 with a logarithmic increase in depth. Below we state the result for circuits of fan-in 2 as will be used later. Recall that a circuit is *monotone* if it uses only AND and OR gates, but no NOT gates.

**Theorem 5.5** ([15, 16]). *Let  $w_1, \dots, w_n \in \mathbb{Z}_{>0}$  be positive weights, and  $T \in \mathbb{Z}_{\geq 0}$  be a threshold. Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  be the monotone function such that  $f(x_1, \dots, x_n) = 1$  if and only if  $\sum_{i=1}^n w_i x_i \geq T$ . Then there is a depth- $O(\log^2 n)$  monotone circuit of fan-in 2 that computes the function  $f$ .*

The well-known *Karchmer-Wigderson game* [64] connects the depth of monotone circuits and communication complexity. Given a monotone function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ , the *monotone Karchmer-Wigderson game* is the following: Alice receives  $a \in f^{-1}(0)$ , Bob receives  $b \in f^{-1}(1)$ , they communicate bits to each other, and the goal is to agree on a position  $i \in [n]$  such that  $a_i = 0$  and  $b_i = 1$ . Let  $D_{\text{mon-KW}}^{\text{cc}}(f)$  be the deterministic communication complexity of this game.

**Theorem 5.6** ([64]). *Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  be a monotone function,  $D_{\text{mon-KW}}^{\text{cc}}(f)$  be the deterministic communication complexity of the Karchmer-Wigderson game, and  $\text{depth}(f)$  be the minimum depth of a fan-in 2 monotone circuit that computes  $f$ . Then  $\text{depth}(f) = D_{\text{mon-KW}}^{\text{cc}}(f)$ .*

Combining Theorems 5.5 and 5.6, we immediately get that  $D_{\text{mon-KW}}^{\text{cc}}(f) = O(\log^2 n)$  for every weighted threshold function  $f$  on  $n$  inputs.

## 5.3 Small LP relaxation for MIN-KNAPSACK.

In this section, we show the existence of a  $(1/\varepsilon)^{O(1)} n^{O(\log n)}$ -size LP relaxation of MIN-KNAPSACK with integrality gap  $2 + \varepsilon$ , proving Theorem 5.1. First, we give a high-level overview of the construction in Section 5.3.1. The actual protocol is described and analyzed in Section 5.3.2.

### 5.3.1 Overview.

Consider the slack matrix  $S$  that has one row for each knapsack cover inequality and one column for each feasible solution of MIN-KNAPSACK. More precisely, let  $f : \{0, 1\}^n \rightarrow$

$\{0, 1\}$  denote the weighted threshold function defined by the item sizes  $s_i$  ( $i \in [n]$ ) and demand  $D$  as in (5.2). The rows and columns of  $S$  are indexed by  $a \in f^{-1}(0)$  and  $b \in f^{-1}(1)$  respectively. The entries of  $S$  are given by

$$S_{a,b} := \sum_{i:a_i=0} s'_i b_i - U,$$

where as preceedingly

$$U = U(a) := D - \sum_{i:a_i=1} s_i \quad \text{and} \quad s'_i = s'_i(a) = \min\{s_i, U\}.$$

Geometrically,  $S$  is the slack matrix of the polyhedral pair  $(P, Q)$  in which  $P$  is the MIN-KNAPSACK polytope and  $Q$  is the (unbounded) polyhedron defined by the knapsack cover inequalities.

Ideally, we would like to design a communication protocol for  $S$ , as those discussed in Section 5.2.2, with low communication complexity. This would imply a low-rank non-negative factorization of  $S$ . From the factorization theorem of Section 5.2.1, it would follow that there exists a small-size extended formulation yielding a polyhedron  $R$  containing the MIN-KNAPSACK polytope  $P$  and contained in the knapsack-cover relaxation  $Q$ . Hence, we would get a small-size LP relaxation for MIN-KNAPSACK that implies the exponentially many knapsack cover inequalities, and thus have integrality gap at most 2.

However, due to the fact that the quantities involved can be exponential in  $n$ , making them too expensive to communicate directly, we have to settle for showing the existence of small-size extended formulation that *approximately* implies the knapsack cover inequalities. Before discussing further these complications, we give an idealized version of the protocol to help with the intuition. Assume for now that all item sizes and the demand are polynomial in  $n$ . Thus Alice and Bob can communicate them with  $O(\log n)$  bits.

The goal of the two players is to compute the slack  $S_{a,b} = \sum_{i:a_i=0} s'_i b_i - U$ , when Alice is given an infeasible  $a \in \{0, 1\}^n$  and Bob is given a feasible  $b \in \{0, 1\}^n$ . That is, after several rounds of communication, either one of them outputs some non-negative value  $\xi$ , such that the expectation of  $\xi$  equals  $S_{a,b}$ .

We define for a set of items  $J \subseteq [n]$  the quantities

$$s(J) := \sum_{j \in J} s_j, \quad \text{and} \quad s'(J) := \sum_{j \in J} s'_j.$$

Let  $A$  and  $B$  be the subsets of  $[n]$  corresponding to Alice's input  $a$  and Bob's input  $b$ ,



respectively. The slack we want to compute thus becomes

$$\sum_{i:a_i=0} s'_i b_i - U = \sum_{i \in B \setminus A} s'_i - U = s'(B \setminus A) - U.$$

At the beginning, Alice computes the residual demand  $U$  and sends it to Bob. Now observe that if there is some  $i^* \in B \setminus A$ , such that  $s_{i^*} \geq U$ , then we have  $s'_{i^*} = U$ , and we can easily write the slack as

$$s'(B \setminus A \setminus \{i^*\}) + (s'_{i^*} - U) = s'(B \setminus A \setminus \{i^*\}),$$

similarly to the uncapacitated case discussed in the introduction. Recall that we call an item  $i$  large if  $s_i \geq U$  and small otherwise. Let  $I_{\text{large}}$  be the set of large items and  $I_{\text{small}}$  be the set of small items.

The rest of the protocol is divided into two cases as follows, depending on whether Alice and Bob can easily find a large item  $i^* \in B \setminus A$ . To this end, Alice sends  $s(I_{\text{large}} \cap A)$  to Bob. Note that now Bob can compute

$$s(I_{\text{small}} \cap A) = D - U - s(I_{\text{large}} \cap A).$$

Bob computes the contribution of large items in  $B$ , that is,  $s(I_{\text{large}} \cap B)$ .

If  $s(I_{\text{large}} \cap B) > s(I_{\text{large}} \cap A)$ , then we are guaranteed that there is some  $i^* \in I_{\text{large}} \cap (B \setminus A)$ . Moreover, defining the threshold function

$$g(x) := \begin{cases} 1 & \text{if } \sum_{i \in I_{\text{large}}} s_i x_i \geq s(I_{\text{large}} \cap B), \\ 0 & \text{otherwise,} \end{cases} \quad (5.4)$$

then  $g(a) = 0$  and  $g(b) = 1$ . Hence, Alice and Bob can find such an item with  $O(\log^2 n)$  bits of communication, see Section 5.2.3. With that, it is not hard to compute  $s'(B \setminus A \setminus \{i^*\})$  with  $O(\log n)$  bits of communication:

- Alice samples a uniformly random item  $i$  and sends the index to Bob.
- Bob replies with  $b_i$ .
- Alice outputs  $s'_i \cdot n$  if  $b_i = 1$ ,  $i \neq i^*$  and  $i \notin A$ , and outputs 0 otherwise.

All her outputs are non-negative and their expectation is exactly the slack.

In the other case,  $s(I_{\text{large}} \cap B) \leq s(I_{\text{large}} \cap A)$ . Note that

$$s(B) = s(I_{\text{large}} \cap B) + s(I_{\text{small}} \cap B) \geq D = s(I_{\text{large}} \cap A) + s(I_{\text{small}} \cap A) + U,$$

thus

$$s(I_{\text{small}} \cap B) - s(I_{\text{small}} \cap A) - U \geq s(I_{\text{large}} \cap A) - s(I_{\text{large}} \cap B) \geq 0.$$

We now write the slack as

$$\begin{aligned} s'(B \setminus A) - U &= s'(I_{\text{large}} \cap (B \setminus A)) + s(I_{\text{small}} \cap (B \setminus A)) - U \\ &= s'(I_{\text{large}} \cap (B \setminus A)) + s(I_{\text{small}} \cap B) - s(I_{\text{small}} \cap (A \cap B)) - U \\ &= s'(I_{\text{large}} \cap (B \setminus A)) + s(I_{\text{small}} \cap B) - s(I_{\text{small}} \cap A) \\ &\quad + s(I_{\text{small}} \cap (A \setminus B)) - U \\ &= s'(I_{\text{large}} \cap (B \setminus A)) + s(I_{\text{small}} \cap (A \setminus B)) \\ &\quad + (s(I_{\text{small}} \cap B) - s(I_{\text{small}} \cap A) - U). \end{aligned}$$

Alice and Bob can compute the first and the second term in expectation using a protocol similar to that in the previous case. The last term can be computed by Bob with all the information he has at this stage. To conclude, in both cases, Alice and Bob can compute the exact slack  $S_{a,b}$  with  $O(\log^2 n)$  bits of communication.

### 5.3.2 The Protocol.

The actual slack matrix  $S^\varepsilon$  we work with is defined as

$$S_{a,b}^\varepsilon := \sum_{i:a_i=0} s'_i b_i - \frac{2}{2+\varepsilon} U, \quad (5.5)$$

where  $\varepsilon > 0$  is any small constant,  $a \in f^{-1}(0)$  and  $b \in f^{-1}(1)$ .  $S^\varepsilon$  is the slack matrix of the polyhedral pair  $(P, Q^\varepsilon)$  where  $P$  is the MIN-KNAPSACK polytope and  $Q^\varepsilon$  is the polyhedron defined by a slight weakening of the knapsack cover inequalities obtained by replacing the right hand side of (5.1) by  $\frac{2}{2+\varepsilon} U < U$ . For every  $x \in \mathbb{R}_{\geq 0}^n$  that satisfies all weakened knapsack cover inequalities, we have that  $\frac{2+\varepsilon}{2} x$  satisfies all original knapsack cover inequalities, and thus  $(2+\varepsilon)x$  dominates a convex combination of feasible solutions. Therefore the integrality gap of the resulting LP relaxation (obtained from a non-negative factorization of  $S^\varepsilon$ ) is at most  $2+\varepsilon$ .

In order to refer to the “derived” weighted threshold functions  $g$  as in (5.4), we need a last bit of terminology. We say that  $g : \{0,1\}^n \rightarrow \{0,1\}$  is a *truncation* of  $f$  if there exists  $U, T \in \mathbb{Z}_{>0}$  with  $T \leq D$  such that  $g(x) = 1$  iff  $\sum_{i=1}^n w_i x_i \geq T$ , where  $w_i = s_i$  if  $s_i \geq U$  and  $w_i = 0$  otherwise. We are now ready to state our main technical lemma.

**Lemma 5.7.** *For all constants  $\varepsilon \in (0, 1)$ , item sizes  $s_i \in \mathbb{Z}_{>0}$  ( $i \in [n]$ ), all smaller than  $2^{\lceil n \log n \rceil}$  and demand  $D \in \mathbb{Z}_{>0}$  with  $\max\{s_i \mid i \in [n]\} \leq D \leq \sum_{i=1}^n s_i$ , such that all truncations of the corresponding weighted threshold function admit depth- $t$  monotone circuits of fan-in 2, there is a  $O(\log(1/\varepsilon) + \log n + t)$ -complexity randomized communication protocol with non-negative outputs that computes the slack matrix  $S^\varepsilon$  in expectation. Since we may always take*

$t = O(\log^2 n)$ , this gives a  $O(\log(1/\varepsilon) + \log^2 n)$ -complexity protocol, unconditionally.

Before giving the proof, let us remark that Theorem 5.1 follows directly from this lemma. Indeed, the extra assumptions in the lemma are without loss of generality: the fact that we may assume without loss of generality that the item sizes  $s_i$  are positive integers that can be written down with at most  $\lceil n \log n \rceil$  bits, is due to a classic result from [86]; and the fact that we may also assume that the demand  $D$  is a positive integer with  $\max\{s_i \mid i \in [n]\} \leq D \leq \sum_{i=1}^n s_i$  should be clear.

Moreover, Lemma 5.7 implies that we can obtain a relaxation of polynomial size if all truncations of the weighted threshold function have monotone circuits of logarithmic depth. In particular, this is the case if all item sizes are polynomial in  $n$ . In that case the threshold function (and its truncations) can simply be written as the majority function on  $O(\sum_i s_i)$  input bits and, as such functions have monotone circuits of fan-in 2 of logarithmic depth, i.e., depth  $O(\log(\sum_i s_i))$ . Thus, using majority functions instead of threshold functions in our communication protocol, we get that for all  $\varepsilon \in (0, 1)$ ,  $c > 0$ , item sizes  $s_1, \dots, s_n \in \{0, 1, \dots, n^c\}$  and demand  $D \in \mathbb{N}$ , there exists a size- $(1/\varepsilon)^{O(1)} n^{O(c)}$  extended formulation defining an LP relaxation of MIN-KNAPSACK with integrality gap at most  $2 + \varepsilon$ . However, it is important to note here that when  $c$  is a constant (and hence the sizes  $s_1, \dots, s_n$  and the demand  $D$  are polynomial in  $n$ ), we can write down an exact polynomial size LP formulation of the MIN-KNAPSACK problem. For completeness, we elaborate more on that in Section 5.3.3

We now proceed by proving our main technical result, i.e., Lemma 5.7. We also pictorially illustrate the protocol in Figure 5.1.

*Proof of Lemma 5.7.* Let  $\alpha = \alpha(\varepsilon) := 2/(2 + \varepsilon)$  and  $\delta > 0$  be such  $(1 - 2\delta)/(1 + \delta) = \alpha$ . Thus  $\delta = \varepsilon/(6 + 2\varepsilon) = \Theta(\varepsilon)$ . As above, we denote by  $a \in f^{-1}(0)$  the input of Alice and  $b \in f^{-1}(1)$  that of Bob, and let  $A$  and  $B$  denote the corresponding subsets of  $[n]$ .

First, Alice tells Bob the identity of the set of large items  $I_{\text{large}} = \{i \in [n] \mid s_i \geq U\}$  and its complement, the set of small items  $I_{\text{small}}$ . This costs  $O(\log n)$  bits of communication. For instance, Alice can simply send the index of a smallest large item to Bob, or inform Bob that  $I_{\text{large}}$  is empty. Recall that

$$U = D - s(A) = D - s(I_{\text{large}} \cap A) - s(I_{\text{small}} \cap A).$$

Then, Alice sends Bob the unique nonnegative integer  $k$  such that  $(1 + \delta)^k \leq U < (1 + \delta)^{k+1}$ . This sets the scale at which the protocol is operating. Since  $U \leq n \cdot 2^{\lceil n \log n \rceil} \leq 2^{n^2}$ , we have  $(1 + \delta)^k \leq 2^{n^2}$ . This implies that  $k = O((1/\varepsilon)n^2)$ , thus  $k$  can be sent to Bob with  $\log(1/\varepsilon) + 2 \log n + O(1) = O(\log(1/\varepsilon) + \log n)$  bits. Let  $\tilde{U} := (1 + \delta)^k$ .

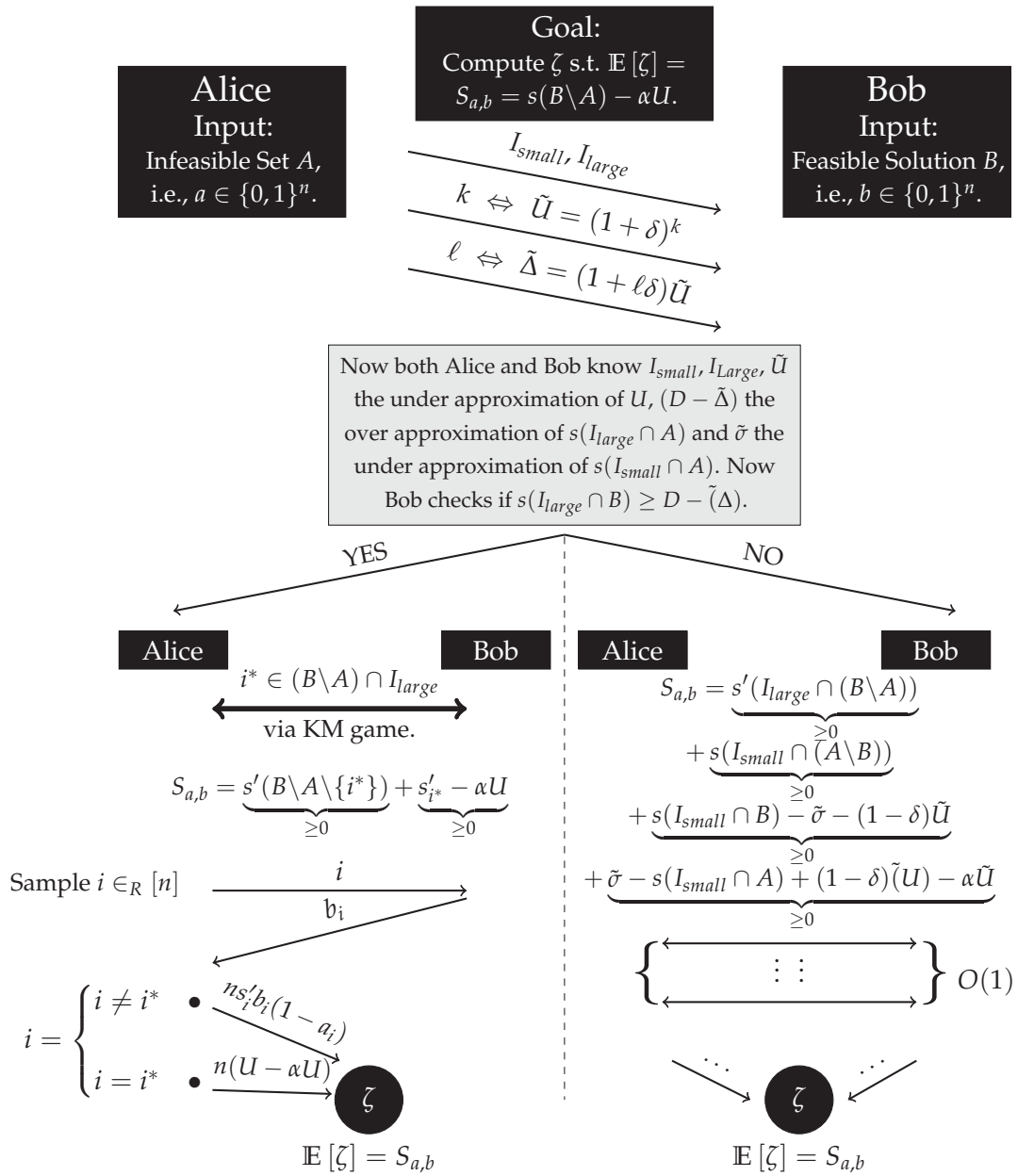


Figure 5.1 – Execution of the communication protocol between Alice and Bob. Thick arrows illustrate messages that require  $O(\log^2 n)$  bits to communicate, whereas thin arrows corresponds to  $O(\log n)$  and  $O(1)$  bit messages. The output  $\zeta$  of the protocol has the property that, over the randomness of the protocol,  $\mathbb{E}[\zeta] = S_{a,b}$ .

To efficiently communicate an approximate value of  $s(I_{large} \cap A)$ , Alice sends the unique nonnegative integer  $\ell$  such that

$$(1 + \ell\delta)\tilde{U} < D - s(I_{large} \cap A) \leq (1 + \ell\delta)\tilde{U} + \delta\tilde{U}.$$

Since small items have size at most  $U$  and we have at most  $n$  of them, we have  $s(I_{\text{small}} \cap A) \leq Un$ . Hence,

$$D - s(I_{\text{large}} \cap A) = U + s(I_{\text{small}} \cap A) \leq (n+1)U \leq (n+1)(1+\delta)\tilde{U}.$$

Since  $(1+\ell\delta)\tilde{U} < (n+1)(1+\delta)\tilde{U}$ , we have  $\ell = O((1/\varepsilon)n)$ . This means that Alice can communicate  $\ell$  to Bob with only  $O(\log(1/\varepsilon) + \log n)$  bits. Let  $\tilde{\Delta} = \tilde{\Delta}(\delta) := (1+\ell\delta)\tilde{U}$ . This is Bob's strict under-approximation of  $D - s(I_{\text{large}} \cap A)$ , so that  $D - \tilde{\Delta}$  is a strict over-approximation of  $s(I_{\text{large}} \cap A)$ .

Bob checks if  $s(I_{\text{large}} \cap B) \geq D - \tilde{\Delta}$ . If this is the case, then the weighted threshold function  $g$  such that  $g(x) = 1$  iff  $\sum_{i \in I_{\text{large}}} s_i x_i \geq D - \tilde{\Delta}$  separates  $a$  from  $b$  in the sense that  $g(a) = 0$  and  $g(b) = 1$ . Since  $g$  is a truncation of  $f$ , Alice and Bob can exchange  $t$  bits to find an index  $i^* \in I_{\text{large}}$  such that  $a_{i^*} = 0$  and  $b_{i^*} = 1$ .

We can rewrite the slack  $S_{a,b}^\varepsilon = s'(B \setminus A) - \alpha U$  as

$$\begin{aligned} s'(B \setminus A \setminus \{i^*\}) + s'_{i^*} - \alpha U &= s'(B \setminus A \setminus \{i^*\}) + (U - \alpha U) \\ &= \sum_{i: a_i=0, i \neq i^*} s'_i b_i + (U - \alpha U). \end{aligned} \quad (5.6)$$

With the knowledge of  $i^*$ , Alice and Bob can compute the slack as follows:

1. Alice samples a uniformly random number  $i \in [n]$ . If  $i \notin A$ , continue to the next step, otherwise Alice outputs 0 and terminates the communication.
2. If  $i = i^*$ , Alice outputs  $n \cdot (U - \alpha U)$  and terminates the communication, otherwise continue.
3. Alice sends  $i$  to Bob using  $\lceil \log n \rceil$  bits of communication, and Bob sends  $b_i$  back to Alice.
4. Alice outputs  $n \cdot s'_i b_i$ .

The above communication costs  $O(\log n)$  bits, all outputs are non-negative and can be computed with the information available to each player, and by linearity of expectation, the expected output is exactly the slack (5.6). Together with the  $O(\log(1/\varepsilon) + \log n + t)$  bits communicated previously, we conclude that in this case there is a protocol that computes the slack in expectation with  $O(\log(1/\varepsilon) + \log n + t)$  bits of communication.

In the other case, we have  $s(I_{\text{large}} \cap B) < D - \tilde{\Delta}$ . Because  $b \in \{0, 1\}^n$  is feasible, we get

$$s(B) \geq D \iff \underbrace{s(I_{\text{large}} \cap B)}_{< D - \tilde{\Delta}} + s(I_{\text{small}} \cap B) \geq D,$$

therefore we can bound  $s(I_{\text{small}} \cap B)$  as

$$\begin{aligned} s(I_{\text{small}} \cap B) &> \tilde{\Delta} \geq D - s(I_{\text{large}} \cap A) - \delta \tilde{U} \\ &= s(I_{\text{small}} \cap A) + U - \delta \tilde{U} \\ &\geq \tilde{\sigma} + (1 - \delta) \tilde{U}, \end{aligned} \tag{5.7}$$

where  $\tilde{\sigma}$  is the unique integer multiple of  $\delta \tilde{U}$  such that

$$\tilde{\sigma} \leq s(I_{\text{small}} \cap A) < \tilde{\sigma} + \delta \tilde{U}. \tag{5.8}$$

Since  $\tilde{\sigma} \leq s(I_{\text{small}} \cap A) \leq Un \leq (1 + \delta) \tilde{U}n$ , Alice can communicate  $\tilde{\sigma}$  to Bob with  $O(\log(1/\varepsilon) + \log n)$  bits.

This implies

$$\begin{aligned} s(I_{\text{small}} \cap (B \setminus A)) &= s(I_{\text{small}} \cap B) - s(I_{\text{small}} \cap (A \cap B)) \\ &> \tilde{\sigma} + (1 - \delta) \tilde{U} - s(I_{\text{small}} \cap (A \cap B)). \end{aligned}$$

Recall that by definition of  $\tilde{U}$ , we have  $(1 + \delta) \tilde{U} > U$ , therefore

$$(1 - 2\delta) \tilde{U} - \alpha U > (1 - 2\delta) \tilde{U} - \alpha(1 + \delta) \tilde{U} = 0. \tag{5.9}$$

We now rewrite the slack  $s'(B \setminus A) - \alpha U$  as

$$\begin{aligned} &\underbrace{s(I_{\text{large}} \cap (B \setminus A))}_{= \sum_{i \in I_{\text{large}} \setminus A} s'_i b_i} + \underbrace{s(I_{\text{small}} \cap B) - \tilde{\sigma} - (1 - \delta) \tilde{U}}_{\text{non-negative by (5.7)}} \\ &+ \underbrace{s(I_{\text{small}} \cap (A \setminus B))}_{\sum_{i \in I_{\text{small}} \cap A} s_i (1 - b_i)} + \underbrace{\tilde{\sigma} - s(I_{\text{small}} \cap A) + (1 - \delta) \tilde{U} - \alpha U}_{\text{non-negative by (5.8) and (5.9)}}. \end{aligned}$$

Similar to the previous case, we design a protocol to compute the slack as follows:

1. Alice samples a uniformly random number  $i \in [n + 2]$ . If  $i = n + 2$ , Alice outputs the normalized value of the last term, i.e.,  $(n + 2) \cdot (\tilde{\sigma} - s(I_{\text{small}} \cap A) + (1 - \delta) \tilde{U} - \alpha U)$ , and terminates the communication. Otherwise, she sends  $i$  to Bob using  $O(\log n)$  bits.
2. If  $i = n + 1$ , Bob outputs  $(n + 2) \cdot (s(I_{\text{small}} \cap B) - \tilde{\sigma} - (1 - \delta) \tilde{U})$ , and ends the communication. Otherwise, he replies to Alice with  $b_i$ .
3. If  $i \in I_{\text{large}} \setminus A$ , Alice outputs  $(n + 2) \cdot s'_i b_i$ ; if  $i \in I_{\text{small}} \cap A$ , she outputs  $(n + 2) \cdot s_i (1 - b_i)$ ; otherwise she outputs 0.

We can verify that the outputs of both players can be computed with information available to them, and that the outputs are non-negative due to Equation (5.7), (5.8) and (5.9), and the definition of the variables.  $\square$

### 5.3.3 MIN-KNAPSACK with Polynomial (and Integer) Demand and Sizes

Consider a MIN-KNAPSACK problem instance with demand  $D \in \mathbb{N}$ , where all the item sizes  $s_1, \dots, s_n \in [D]$  are integers<sup>5</sup>. If  $D = O(n^c)$  for some constant  $c > 0$ , it follows from Lemma 5.7 and the discussion thereafter that there exists an LP formulation of size  $\text{poly}(n, D)$  that *approximates* the MIN-KNAPSACK problem. However it is not hard to see that this is not the best that we can do in this case; we present in this section how to explicitly write an *exact* LP formulation for this case of the same size<sup>6</sup>.

Recall that to completely specify an instance of the MIN-KNAPSACK problem, we are further given nonnegative item costs  $c_1, c_2, \dots, c_n$ , and in our model of computation, we require that these costs *only* appear in the objective function of our LP, i.e., the LP constraints can only depend on  $D$  and the sizes  $s_1, \dots, s_n$ , but not on the costs  $c_1, \dots, c_n$ . We present in this section a construction of such an LP that is inspired by the folklore Dynamic Programming (DP) algorithm for the MIN-KNAPSACK problem.

**Lemma 5.8 (Folklore).** *There exists an exact LP formulation of the MIN-KNAPSACK problem with demand  $D \in \mathbb{N}$ , and integer item sizes  $s_1, \dots, s_n \in [D]$  of size  $\text{poly}(n, D)$ .*

**Folklore DP algorithm.** We briefly remind the reader of the well-known Dynamic Programming algorithm of the MIN-KNAPSACK problem, as it motivates the construction in the following section. Given a MIN-KNAPSACK instance  $\mathcal{J}$  with demand  $D \in \mathbb{N}$ , items sizes  $s_1, s_2, \dots, s_n \in [D]$  and items costs  $c_1, c_2, \dots, c_n \in \mathbb{R}_+$ , the Dynamic Programming algorithm proceeds as follows:

- Let  $M \in \mathbb{R}_+^{n \times D}$  be a matrix indexed by all pairs  $(i, d)$  for  $0 \leq i \leq n$  and  $0 \leq d \leq D$ .
- Recursively define  $M(i, d)$  for  $1 \leq i \leq n$  and  $1 \leq d \leq D$  to be:

$$M(i, d) = \begin{cases} 0 & \text{if } d = 0, \\ -\infty & \text{if } i = 0 \text{ and } d > 0, \\ \min\{c_i, M(i-1, d)\} & \text{if } s_i > d, \\ \min\{M(i-1, d), M(i-1, d-s_i) + c_i\} & \text{otherwise.} \end{cases}$$

<sup>5</sup>Note that since all size are integers, we can assume without of generality that  $s_i \leq D$  for all  $i \in [n]$  since rounding any size greater than  $D$  down to  $D$  does not change the problem.

<sup>6</sup>This fact was mentioned as a comment by the one of the reviewers for our SODA paper who also recommended adding it for completeness.

In other words,  $M(i, d)$  is the minimum cost subset  $S^*$  over all possible subsets  $S \subseteq [i]$  such that  $\sum_{k \in S} s_k \geq d$ , and  $M(n, D)$  is the optimal solution of the MIN-KNAPSACK instance.

**DP Algorithm as a Minimum Weight  $s$ - $t$  Flow Problem.** We will now describe how the above DP algorithm can be alternatively seen as minimum weight  $s$ - $t$  flow problem on a graph  $G = (V, E)$  of size  $\text{poly}(n, D)$ . Given a MIN-KNAPSACK instance  $\mathcal{F}$  with demand  $D \in \mathbb{N}$ , sizes  $s_1, s_2, \dots, s_n \in [D]$  and costs  $c_1, c_2, \dots, c_n \in \mathbb{R}_+$ , we construct our weighted graph  $G_{\mathcal{F}} = (V, E, w)$  as follows:

- For every  $0 \leq i \leq n$ , and every  $0 \leq d \leq n \times D$ , we have a vertex  $(i, d) \in V$ . Moreover, we have two vertices  $s, t$  that act as source and sink respectively. Formally,

$$\begin{aligned} V &= \{s, t\} \cup \{(i, d) : 0 \leq i \leq n, 0 \leq d \leq n \times D\} \\ &= \{s, t\} \cup \bigcup_{i=0}^n V_i, \end{aligned}$$

where  $V_i = \{(i, d) : 0 \leq d \leq n \times D\}$ .

- For every  $i \geq 1$  and  $d \geq s_i$ , we have an edge  $e = ((i, d), (i-1, d-s_i))$  of weight  $w_e = c_i$ , and an edge  $e' = ((i, d), (i-1, d))$  of weight  $w_{e'} = 0$ . We also have an edge  $(s, (0, 0))$  of weight 0, and edges  $((n, d), t)$  of weight 0 for all  $D \leq d \leq n \times D$ . This defines the edge set  $E$ , and the weight function  $w : E \mapsto \mathbb{R}^+$ .
- The capacity of each edge  $e \in E$  is 1.

Since the source  $s$  is only connected to the vertex  $(0, 0)$  with an edge of capacity 1, any  $s$ - $t$  flow in  $G_{\mathcal{F}}$  has a value at most 1. Moreover, for  $1 \leq i \leq n-1$ , a vertex in  $V_i$  can only be connected to vertices in  $V_{i-1}$  and  $V_{i+1}$ , and a vertex in  $V_n$  can only be connected to  $t$  and to vertices in  $V_{n-1}$ . Thus it follows that any simple path  $p$  from  $s$  to  $t$  is of length  $n+2$ . Namely, such a path  $p$  is the sequence of vertices  $p = (s, (0, d_0), (1, d_1), (2, d_2), \dots, (n, d_n), t)$  with  $d_0 = 0$ ,  $d_n \in \{D, D+1, \dots, n \times D\}$ , and  $(d_i - d_{i-1}) \in \{0, s_i\}$  for  $i = 1, 2, \dots, n$ . It is not hard to see that every such path  $p$  corresponds to a set  $S \subseteq [n]$  such that  $\sum_{i \in S} s_i \geq D$ . We formalize this in the following Claim.

**Claim 5.9.** *Given a MIN-KNAPSACK instance  $\mathcal{F}$  with demand  $D \in \mathbb{N}$  and integers sizes  $s_1, s_2, \dots, s_n \in [D]$ , there is a one to one correspondence between feasible sets  $S \subseteq [n]$  such that  $\sum_{i \in S} s_i \geq D$ , and simple  $s$ - $t$  paths in  $G_{\mathcal{F}}$ . Moreover, the weight of every  $s$ - $t$  path in  $G_{\mathcal{F}}$  is the cost of its corresponding feasible set in  $\mathcal{F}$ .*



*Proof.* Let  $\mathcal{A}$  be the set of feasible sets in  $\mathcal{S}$ , and  $\mathcal{B}$  be the set of  $s$ - $t$  paths in  $G_{\mathcal{S}}$ , i.e.,

$$\mathcal{A} = \{S : S \subseteq [n] \text{ and } \sum_{i \in S} s_i \geq D\}, \quad \mathcal{B} = \{p : p \text{ is a simple } s\text{-}t \text{ path in } G_{\mathcal{S}}\}.$$

We will construct two maps  $\mathbf{S} : \mathcal{B} \mapsto \mathcal{A}$ , and  $\mathbf{P} : \mathcal{A} \mapsto \mathcal{B}$  such that

- a)  $\mathbf{S}(p_1) \neq \mathbf{S}(p_2)$  for all  $p_1 \neq p_2 \in \mathcal{B}$ ,
- b)  $\mathbf{P}(S_1) \neq \mathbf{P}(S_2)$  for all  $S_1 \neq S_2 \in \mathcal{A}$ ,
- c)  $\mathbf{S}(\mathbf{P}(p)) = p$  for all  $p \in \mathcal{B}$ ,
- d)  $\mathbf{P}(\mathbf{S}(S)) = S$  for all  $S \in \mathcal{A}$ ,
- e) and for every  $p \in \mathcal{B}$ ,  $\sum_{e \in p} w_e = \sum_{i \in \mathbf{S}(p)} c_i$ .

Fix any simple  $s$ - $t$  path  $p = (s, (0,0), (1,d_1), (2,d_2), \dots, (n,d_n), t) \in \mathcal{B}$ , and construct the unique set  $\mathbf{S}(p) \in \mathcal{A}$  as follows:

$$\mathbf{S}(p) = \{i : 1 \leq i \leq n, d_i \neq d_{i-1}\},$$

where we let  $d_0 = 0$ . The uniqueness of  $\mathbf{S}(p)$  follows by construction since  $(d_i - d_{i-1}) \in \{0, s_i\}$ . Moreover,

$$\sum_{i \in \mathbf{S}(p)} s_i = \sum_{i \in \mathbf{S}(p)} (d_i - d_{i-1}) + \sum_{i \notin \mathbf{S}(p)} \underbrace{d_i - d_{i-1}}_{=0} = \sum_{i=1}^n (d_i - d_{i-1}) = d_n.$$

Since  $d_n \geq D$ , it follows that  $\mathbf{S}(p) \in \mathcal{A}$ . Moreover, we have by construction of  $G_{\mathcal{S}}$  that

$$\begin{aligned} \sum_{i \in \mathbf{S}(p)} c_i &= \sum_{1 \leq i \leq n: d_{i-1} = d_i - s_i} c_i \\ &= \sum_{1 \leq i \leq n: d_{i-1} = d_i - s_i} w_{((i,d_i), (i-1, d_i - s_i))} \\ &= \sum_{1 \leq i \leq n: d_{i-1} = d_i - s_i} w_{((i,d_i), (i-1, d_i - s_i))} + \sum_{1 \leq i \leq n: d_{i-1} = d_i} \underbrace{w_{((i,d_i), (i-1, d_i))}}_{=0} \\ &= \sum_{e \in p} w_e. \end{aligned}$$

Now consider any feasible set  $S \subseteq [n] \in \mathcal{A}$  and construct the unique sequence  $d_0, d_1, \dots, d_n$  as follows:

$$d_i = \begin{cases} 0 & \text{if } i = 0, \\ d_{i-1} & \text{if } i \notin S, \\ s_i + d_{i-1} & \text{if } i \in S. \end{cases}$$

Note that  $d_n = \sum_{i \in S} d_i \geq D$ . We claim that there exists a simple path  $\mathbf{P}(S)$  in  $G_{\mathcal{J}}$  defined as  $\mathbf{P}(S) = \{s, (0, d_0), (1, d_1), \dots, (n, d_n), t\}$  with the sequence  $d_0, d_1, \dots, d_n$  defined earlier. To see this, observe that:

1. The edge  $(s, (0, 0))$  is trivially in  $E$ , and the edge  $((n, d_n), t)$  is in  $E$  since  $d_n \geq D$ .
2. For every  $1 \leq i \leq n$ , we have both edges  $((i, d), (i-1, d))$  and  $((i, d), (i-1, d-s_i))$  in  $E$  by construction, where the first is in  $\mathbf{P}(S)$  if  $i \notin S$ , and the second is in  $\mathbf{P}(S)$  if  $i \in S$ .

Properties (b) and (c) follow since we have a fixed ordering on the items, and all items have non-zero sizes. □

It follows from Claim 5.9 that finding a feasible solution for a MIN-KNAPSACK instance  $\mathcal{J}$  amounts to finding an  $s$ - $t$  flow of value 1 in  $G_{\mathcal{J}}$ , and finding a minimum cost set  $S$  in  $\mathcal{J}$  amounts to finding an minimum weight  $s$ - $t$  flow of value 1. Moreover, given graph  $G = (\{s, t\} \cup V, E, c)$  where each edge  $e \in E$  has a capacity  $c \in \mathbb{N}^+$ , we know how to write down an exact LP formulation for the min-cost flow problem such that the weight of the edges only appear in the objective function (and hence the constraints are independent of  $w$ ), whose size is polynomial in the size of the graph  $G$ .

Since the size of  $G_{\mathcal{J}}$  is polynomial in  $n$  and  $D$ , Fact 5.8 follows.

## 5.4 Flow-cover inequalities.

A variant of the knapsack cover inequalities, known as the *flow cover inequalities*, was also used to strengthen LPs for many problems such as the Fixed Charge Network Flow problem [29] and the SINGLE-DEMAND FACILITY LOCATION problem [28]. In this section, we describe the application of flow cover inequalities to the SINGLE-DEMAND FACILITY LOCATION problem as used in [28], and then give an  $O(\log^2 n)$ -bit two-party communication protocol that computes a weakened version of these inequalities.

In the SINGLE-DEMAND FACILITY LOCATION problem, we are given a set  $F$  of  $n$  facilities, such that each facility  $i \in F$  has a capacity  $s_i$ , an opening cost  $f_i$ , and a per-unit cost  $c_i$  to serve the demand. The goal is to serve the demand  $D$  by opening a subset  $S \subseteq F$  of facilities such that the combined cost of opening these facilities and serving the demand is minimized. The authors of [28] cast this problem as an Integer Program, and showed that its natural LP relaxation has an unbounded integrality gap. To reduce this gap, they strengthened the relaxation by adding the so-called flow cover inequalities that we define shortly (See Section 3 in [28] for a more elaborate discussion).

A feasible solution  $(x, y)$  with  $y \in \{0, 1\}^n$  and  $x \in [0, 1]^n$  for the SINGLE-DEMAND

FACILITY LOCATION LP can be thought of as follows: for each  $i \in F$ ,  $y_i \in \{0, 1\}$  indicates if the  $i$ -th facility is open, and  $x_i \in [0, 1]$  indicates the fraction of the demand  $D$  being served by the  $i$ -th facility. A feasible solution  $(x, y)$  must then satisfy that

1. The demand is *met*, i.e.,  $\sum_i x_i = 1$ .
2. No facility is supplying more than its capacity, i.e.,  $0 \leq x_i D \leq y_i s_i$  for all  $i \in F$ .

For a subset  $J \subseteq F$  of facilities and a feasible solution  $(x, y)$ , we denote by  $B = \{i \in F : y_i = 1\} \subseteq F$  the set of *open* facilities according to  $y$ , and we define the quantity  $x(J)$  to be the overall demand served by the facilities in  $J$ , i.e.,  $x(J) = \sum_{i \in J} x_i D$ .<sup>7</sup> We also define the quantities  $s(\cdot)$  and  $s'(\cdot)$  as in Section 5.3.1.

Carnes and Shmoys [28] showed that adding the flow cover inequalities (FCI) reduces the integrality gap of the natural LP relaxation down to 2. These inequalities are defined as follows: for any *infeasible* set  $A \subseteq F$  (i.e.,  $A \subseteq F$  such that  $s(A) < D$ ), and for all partitions of  $F \setminus A = F_1 \sqcup F_2$ , the following inequality holds for all feasible solutions  $(x, y)$ :

$$s'(F_1 \cap B) + x(F_2 \cap B) \geq U, \tag{FCI}$$

where  $U = D - s(A)$  is the residual demand and  $s'_i = \min\{s_i, U\}$ . For brevity, we refer to an infeasible set  $A$  along with some partition  $F_1 \sqcup F_2 = F \setminus A$  as an *infeasible tuple*  $(A, F_1, F_2)$ . Note that for  $F_2 = \emptyset$ , the flow-cover inequalities are the same as the knapsack cover inequalities.

Similar to the knapsack cover inequalities, the goal is to compute the slack of a *relaxed* version of (FCI) in expectation for any feasible solution  $(x, y)$  and any infeasible tuple  $(A, F_1, F_2)$ . Namely, for any  $\varepsilon \in (0, 1)$ , let  $\alpha = 2/(2 + \varepsilon)$ , then our goal is to design an  $O(\log^2 n + \log(1/\varepsilon))$ -complexity two-party communication protocol with private randomness and nonnegative outputs whose expected output equals  $s'(F_1 \cap B) + x(F_2 \cap B) - \alpha U$ . That is, we want to compute the slack with respect to a given (weakened) flow-cover inequality  $s'(F_1 \cap B) + x(F_2 \cap B) \geq \alpha U$ , where the RHS of (FCI) is replaced by  $\alpha U$ . This implies the existence of an LP of size  $(1/\varepsilon)^{O(1)} n^{O(\log n)}$  with an integrality gap at most  $2 + \varepsilon$  for the SINGLE-DEMAND FACILITY LOCATION problem.

In Section 5.4.1, we set up the notation and define a class of feasible solutions with a certain special structure which we refer to as *canonical* feasible solutions. We design the promised communication protocol restricted to canonical solutions in Section 5.4.2, and extend it to arbitrary feasible solutions in Section 5.4.3.

<sup>7</sup>Note that since we are assuming that  $(x, y)$  is feasible, we get that  $x(J) = x(J \cap B)$ .

### 5.4.1 Preliminaries.

Let  $(x, y)$  be a feasible solution for the flow-cover problem with demand  $D$ , and let  $B = \{i \in F : y_i = 1\}$  denote the support of  $y$ . In this terminology,  $B$  only indicates which facilities are open, but it does not capture the *relative* demand being served through each of them. However this distinction will be essential for designing the protocol, hence we partition  $B$  into three disjoint sets  $B = \tilde{F}_1 \sqcup \tilde{F}_2 \sqcup \tilde{F}_3$ , where  $\tilde{F}_1$  denotes the set of open facilities operating at *full* capacity,  $\tilde{F}_2$  denotes the set of open facilities operating at *partial* capacity and  $\tilde{F}_3$  denotes the set of facilities that are open but do *not* serve any demand. Formally, we define these sets as follows:

$$\begin{aligned}\tilde{F}_1 &= \{i \in B : x_i D = s_i y_i\}, \\ \tilde{F}_2 &= \{i \in B : 0 < x_i D < s_i y_i\}, \\ \tilde{F}_3 &= \{i \in B : x_i D = 0\}.\end{aligned}$$

We first focus on feasible solutions  $(x, y)$  that exhibit a certain structure, and then generalize to arbitrary solutions. Namely, we restrict our attention here and in Section 5.4.2 to *canonical* feasible solutions defined as follows:

**Definition 5.10.** A feasible solution  $(x, y)$  with associated sets  $\tilde{F}_1, \tilde{F}_2, \tilde{F}_3$  is *canonical* if  $\tilde{F}_2$  contains at most one facility, i.e.,  $|\tilde{F}_2| \leq 1$ . In other words, in a canonical feasible solution, there is at most one facility  $j$  that supplies a non-zero demand  $x_j D > 0$  which is *not* equal to its full capacity  $s_j$ .

Recall that we are interested in computing

$$s'(F_1 \cap B) + x(F_2 \cap B) - \alpha U \tag{5.10}$$

in expectation, which can be expanded as follows:

$$\underbrace{s'(F_1 \cap \tilde{F}_1) + s'(F_1 \cap \tilde{F}_2) + s'(F_1 \cap \tilde{F}_3)}_{s'(F_1 \cap B)} + \underbrace{x(F_2 \cap \tilde{F}_1) + x(F_2 \cap \tilde{F}_2) + x(F_2 \cap \tilde{F}_3)}_{x(F_2 \cap B)} - \alpha U. \tag{5.11}$$

We get from the definition of the set  $\tilde{F}_3$  that the second to last term in the above equation is 0 when restricted to canonical feasible solutions. In fact, one can completely get rid of the overall contribution of  $\tilde{F}_3$  in the above equation, since intuitively, *closing* down the facilities in  $\tilde{F}_3$  should not alter the feasibility of the solution, and hence Equation (5.11) should still be positive even without accounting for the contribution of  $s'(F_1 \cap \tilde{F}_3)$ . In the communication protocol setting, this intuition translates to designing a protocol that only deals with canonical feasible solutions restricted to  $\tilde{F}_3 = \emptyset$ .

To see that this is without loss of generality, consider a canonical feasible solution  $(x, y)$

such that  $\tilde{F}_3$  is *not* empty, and let  $(x, \bar{y})$  be the projection of  $(x, y)$  on  $\tilde{F}_1 \cup \tilde{F}_2$  — that is, for all  $i \in B \setminus \tilde{F}_3$ , set  $\bar{y}_i = y_i$ , and for all  $i \in \tilde{F}_3$ , set  $\bar{y}_i = 0$ . It follows that  $(x, \bar{y})$  is also a canonical feasible solution, as the items whose support is  $\tilde{F}_3$  do not contribute to the feasibility of the solution, and the cardinality of  $\tilde{F}_2$  does not change. Thus, for any infeasible tuple  $(A, F_1, F_2)$ , Equation (5.11) applied to  $(x, \bar{y})$  can be written as

$$s'(F_1 \cap \tilde{F}_1) + s'(F_1 \cap \tilde{F}_2) + x(F_2 \cap \tilde{F}_1) + x(F_2 \cap \tilde{F}_2) - \alpha U, \quad (5.12)$$

which is also non-negative, as it is the slack of  $(x, \bar{y})$  and  $(A, F_1, F_2)$ . Therefore, for any feasible solution  $(x, y)$ , the slack as given by Equation (5.11) can be viewed as the summation of Equation (5.12) and the non-negative term  $s'(F_1 \cap \tilde{F}_3)$ . The latter is easy to compute with a small communication protocol<sup>8</sup>, thus if Alice and Bob can devise a communication protocol  $\Pi$  that computes (5.12) in expectation, they can then easily compute (5.11) in expectation. For example, Alice can generate a uniformly random bit  $b \in \{0, 1\}$ , and

- if  $b = 0$ , then Alice and Bob run the protocol that computes  $s'(F_1 \cap \tilde{F}_3)$ , and return *twice* its output.
- if  $b = 1$ , then Alice and Bob run the protocol  $\Pi$  that computes (5.12), and return *twice* its output.

Moreover, since  $|\tilde{F}_2| \leq 1$ , and using the fact that  $x_i D = s_i y_i$  for  $i \in \tilde{F}_1$ , we can further simplify Equation (5.12) as follows:

$$s'(F_1 \cap \tilde{F}_1) + s(F_2 \cap \tilde{F}_1) + \gamma(x, y, A, F_1, F_2) - \alpha U, \quad (5.13)$$

where the function  $\gamma := \gamma(x, y, A, F_1, F_2)$  is defined as

$$\gamma = \begin{cases} s'_j y_j & \text{if } \tilde{F}_2 = \{j\} \subseteq F_1 \\ x_j D & \text{if } \tilde{F}_2 = \{j\} \subseteq F_2 \\ 0 & \text{if } \tilde{F}_2 = \{j\} \subseteq A, \text{ or } \tilde{F}_2 = \emptyset. \end{cases} \quad (5.14)$$

For simplicity of notation, we drop the parameters from  $\gamma(x, y, A, F_1, F_2)$  when it is clear from the context.

#### 5.4.2 Randomized Protocol for Canonical Feasible Solutions.

In what follows, we define a randomized communication protocol where Alice gets an infeasible tuple  $(A, F_1, F_2)$ , and Bob gets a *canonical* feasible solution  $(x, y)$  with  $\tilde{F}_3 = \emptyset$ , and the goal is to compute the value of (5.13) in expectation.

<sup>8</sup>To compute  $s'(F_1 \cap \tilde{F}_3)$ , Bob samples an index  $i \in [n]$ . If  $i \notin \tilde{F}_3$ , he outputs 0 and terminates the protocol, otherwise he sends  $i$  to Alice. If  $i \in F_1$ , Alice outputs  $n \cdot s'(i)$ , otherwise, she outputs 0.

## Chapter 5. Knapsack

---

For a fixed  $\varepsilon > 0$ , we define  $\alpha := \alpha(\varepsilon) = 2/(2 + \varepsilon)$ ,  $\delta := \delta(\varepsilon) = \varepsilon/(6 + 2\varepsilon)$  as in the MIN-KNAPSACK case. Similar to the protocol for the knapsack cover inequalities, Alice sends Bob  $O(\log n)$  bits at the beginning so that Bob knows  $I_{\text{large}}, I_{\text{small}}, \tilde{U}, \tilde{\sigma}$  and  $\tilde{\Delta}$ . Recall that:

$I_{\text{large}}$  is the set of large items (i.e.,  $i \in F$  such that  $s(i) \geq U$ ),

$I_{\text{small}}$  is the set of small items,

$\tilde{U}$  is an under-approximation of the residual demand  $U$ ,

$D - \tilde{\Delta}$  is an over-approximation of  $s(I_{\text{large}} \cap A)$

and  $\tilde{\sigma}$  is an under-approximation of  $s(I_{\text{small}} \cap A)$ .

Moreover, knowing his input  $(x, y)$ , Bob can construct the sets  $\tilde{F}_1$  and  $\tilde{F}_2$ . Thus, by exchanging an additional  $O(\log n)$  bits, Alice and Bob can both figure out which condition is satisfied for Equation (5.14).

To compute the value of (5.13) in expectation, we distinguish between the following cases:

**Case 1:** Either  $\tilde{F}_2 = \emptyset$ , or  $\tilde{F}_2 = \{j\}$  and  $j \in A \cup F_1$ . In this case, we have that the value  $\gamma$  is either 0 or  $s'_j y_j$ . Bob now checks if

$$s(I_{\text{large}} \cap (\tilde{F}_1 \cup \tilde{F}_2)) \geq D - \tilde{\Delta}. \quad (5.15)$$

**Equation (5.15) holds:** In the same way as in the MIN-KNAPSACK protocol, Alice and Bob exchange  $O(\log^2 n)$  bits to identify an index  $i^* \in I_{\text{large}}$  such that  $i^* \in ((\tilde{F}_1 \cup \tilde{F}_2) \setminus A)$ . More precisely, this index  $i^*$  belongs to one of the following three sets:

1.  $i^* \in F_2 \cap \tilde{F}_1$ ,
2.  $i^* \in F_1 \cap \tilde{F}_1$ ,
3. or  $i^* = j$  and  $\tilde{F}_2 = \{j\}$ .

Alice and Bob can thus exchange  $O(1)$  more bits to figure out the condition that  $i^*$  satisfies. In what follows, we design an  $O(\log n)$ -communication protocol to handle each of these cases.

If  $i^* \in F_2 \cap \tilde{F}_1$ , then Equation (5.13) can be rewritten as

$$s'(F_1 \cap \tilde{F}_1) + s((F_2 \cap \tilde{F}_1) \setminus \{i^*\}) + \gamma + (s_{i^*} - \alpha U). \quad (5.16)$$

One can see that each of the above four terms is non-negative, and similar to the MIN-KNAPSACK protocol, Alice and Bob can exchange  $O(\log n)$  bits and compute the value of (5.16) as follows:

1. Bob sends Alice the bit  $y_j$  and the index  $j$  using  $\lceil \log(n) \rceil + 1$  bits if and only if  $\tilde{F}_2 = \{j\}$ , and he sends 0 if  $\tilde{F}_2 = \emptyset$ .
2. Alice samples a uniformly random index  $i \in [n+1]$ . If  $i = n+1$ , Alice uses the knowledge of  $\tilde{F}_2$  (and thus  $\gamma$ ) to compute the normalized value of the last terms, that is, she outputs  $(n+1) \cdot (\gamma + s_{i^*} - \alpha U)$ , and terminates the communication. Otherwise, she sends  $i$  to Bob using  $\lceil \log(n) \rceil$  bits.
3. If  $i \in \tilde{F}_1$ , Bob sends  $y_i$  to Alice; otherwise, Bob outputs 0 and terminates the communication.
4. If  $i \in F_1$ , Alice outputs  $(n+1) \cdot s'_i y_i$ ; if  $i \in F_2 \setminus \{i^*\}$ , she outputs  $(n+1) \cdot s_i y_i$ ; otherwise she outputs 0.

The above communication costs  $O(\log n)$  bits, all outputs are non-negative and can be computed with the information available to each player, and by linearity of expectation, the expected output is exactly the slack (5.13) when  $i^* \in F_2 \cap \tilde{F}_1$ .

The case where  $i^* \in F_1 \cap \tilde{F}_1$  is handled similarly, since Equation (5.13) can then be rewritten as

$$s'((F_1 \cap \tilde{F}_1) \setminus \{i^*\}) + s(F_2 \cap \tilde{F}_1) + \underbrace{\gamma + s'_{i^*}}_{=U} - \alpha U,$$

and hence the only difference from the previous protocol would be that Alice has to output  $(n+1) \cdot (\gamma + U - \alpha U)$  instead of  $(n+1) \cdot (\gamma + s_{i^*} - \alpha U)$  in Step 2.

In the remaining case, we have  $\tilde{F}_2 = \{j\}$  and  $i^* = j \in F_1 \cap I_{\text{large}}$ , and hence  $\gamma = s'_j y_j > \alpha U$ . This can be handled by changing the second step of the protocol described earlier in such a way that Alice outputs  $(n+1) \cdot (s'_j - \alpha U)$  if  $i = n+1$ , since the slack can be rewritten in this case as

$$s'(F_1 \cap \tilde{F}_1) + s(F_2 \cap \tilde{F}_1) + s'_j - \alpha U.$$

**Equation (5.15) does not hold:** Recall that since  $(x, y)$  is a feasible solution (and  $\tilde{F}_3 = \emptyset$ ), we have

$$\begin{aligned} D &\leq x(\tilde{F}_1) + x(\tilde{F}_2) \\ &= x(I_{\text{small}} \cap \tilde{F}_1) + x(I_{\text{small}} \cap \tilde{F}_2) + x(I_{\text{large}} \cap \tilde{F}_1) + x(I_{\text{large}} \cap \tilde{F}_2) \\ &\leq x(I_{\text{small}} \cap \tilde{F}_1) + x(I_{\text{small}} \cap \tilde{F}_2) + s(I_{\text{large}} \cap \tilde{F}_1) + s(I_{\text{large}} \cap \tilde{F}_2) \\ &= x(I_{\text{small}} \cap \tilde{F}_1) + x(I_{\text{small}} \cap \tilde{F}_2) + s(I_{\text{large}} \cap (\tilde{F}_1 \cup \tilde{F}_2)). \end{aligned}$$

By the assumption that Equation (5.15) does not hold, together with the argument in Equation (5.7), we conclude that

$$x(I_{\text{small}} \cap \tilde{F}_1) + x(I_{\text{small}} \cap \tilde{F}_2) > \tilde{\Delta} \geq \tilde{\sigma} + (1 - \delta)\tilde{U}. \quad (5.17)$$

Note that since  $|\tilde{F}_2| \leq 1$ , we get that

$$x(I_{\text{small}} \cap \tilde{F}_2) = \begin{cases} 0 & \text{if } \tilde{F}_2 = \emptyset \\ 0 & \text{if } \tilde{F}_2 = \{j\} \subseteq I_{\text{large}} \\ x_j D & \text{if } \tilde{F}_2 = \{j\} \subseteq I_{\text{small}}. \end{cases}$$

We also have that  $x(I_{\text{small}} \cap \tilde{F}_1) = s(I_{\text{small}} \cap \tilde{F}_1)$  by the definition of  $\tilde{F}_1$ . Together this gives that the summation  $s(I_{\text{small}} \cap \tilde{F}_1) + x(I_{\text{small}} \cap \tilde{F}_2)$  is lower bounded by  $\tilde{\sigma} + (1 - \delta)\tilde{U}$ . We rewrite (5.13) as

$$\begin{aligned} & s'(F_1 \cap \tilde{F}_1) + s(F_2 \cap \tilde{F}_1) + \gamma - \alpha U & (5.18) \\ &= s'(I_{\text{large}} \cap F_1 \cap \tilde{F}_1) + s(I_{\text{large}} \cap F_2 \cap \tilde{F}_1) + s(I_{\text{small}} \cap (\tilde{F}_1 \setminus A)) + \gamma - \alpha U \\ &= s'(I_{\text{large}} \cap F_1 \cap \tilde{F}_1) + s(I_{\text{large}} \cap F_2 \cap \tilde{F}_1) + s(I_{\text{small}} \cap (A \setminus B)) \\ & \quad + s(I_{\text{small}} \cap A \cap \tilde{F}_2) + s(I_{\text{small}} \cap \tilde{F}_1) - s(I_{\text{small}} \cap A) + \gamma - \alpha U. \end{aligned}$$

The non-negativity of the first three terms is straightforward, and Alice and Bob can compute them by exchanging  $O(\log n)$  bits – For instance, to compute  $s'(I_{\text{large}} \cap F_1 \cap \tilde{F}_1)$ , Alice samples uniformly  $i \in [n]$  and sends it to Bob, Bob responds with  $b = 1$  if  $i \in \tilde{F}_1$  and  $b = 0$  otherwise. Alice then outputs  $n \cdot s'_i$  if  $i \in I_{\text{large}} \cap F_1$  and  $b = 1$ , and 0 otherwise. The protocols for the second and the third term are very similar.

As for the remaining terms, i.e.,

$$s(I_{\text{small}} \cap A \cap \tilde{F}_2) + s(I_{\text{small}} \cap \tilde{F}_1) - s(I_{\text{small}} \cap A) + \gamma - \alpha U,$$

we get that by adding and subtracting  $(\tilde{\sigma} + (1 - \delta)\tilde{U} - x(I_{\text{small}} \cap \tilde{F}_2))$ , we can rewrite them as

$$\begin{aligned} & \left( s(I_{\text{small}} \cap \tilde{F}_1) - \tilde{\sigma} - (1 - \delta)\tilde{U} + x(I_{\text{small}} \cap \tilde{F}_2) \right) \\ & + \left( \tilde{\sigma} + (1 - \delta)\tilde{U} - \alpha U - s(I_{\text{small}} \cap A) \right) \\ & + \left( s(I_{\text{small}} \cap A \cap \tilde{F}_2) + \gamma - x(I_{\text{small}} \cap \tilde{F}_2) \right). \end{aligned} \quad (5.19)$$

The non-negativity of the first part follows from (5.17), and Bob has all the information to compute it on his own. The non-negativity of the second part follows from our definition of  $\tilde{\sigma}$  and  $\tilde{U}$ , and their relation to  $\delta$  and  $\alpha$ . Moreover, Alice has all the information to compute this part.

To see that the third part (i.e.,  $s(I_{\text{small}} \cap A \cap \tilde{F}_2) + \gamma - x(I_{\text{small}} \cap \tilde{F}_2)$ ) is also non-negative and can easily be computed by one of the players, note that:

1. If  $x(I_{\text{small}} \cap \tilde{F}_2) = 0$ , then clearly it is non-negative. In this case, Bob communicates the set  $\tilde{F}_2$  to Alice using  $O(\log n)$  bits so that she knows whether



$\tilde{F}_2 = \emptyset$ , or the item  $j$  if  $\tilde{F}_2 = \{j\}$  and  $j \in I_{\text{large}}$ . Once  $\tilde{F}_2$  is known to Alice, she can compute both  $s(I_{\text{small}} \cap A \cap \tilde{F}_2)$  and  $\gamma$  (recall that  $\gamma$  would be either 0 or  $s'_j y_j = U$ ).

2. If  $x(I_{\text{small}} \cap \tilde{F}_2) = x_j D \neq 0$ , then we have that  $\tilde{F}_2 = \{j\}$  and  $j \in I_{\text{small}}$ . From our assumption of Case 1, we also have that  $j \in A \cup F_1$ . Since  $A$  and  $F_1$  are two disjoint sets, we get that:

(a) If  $j \in A$ , then

$$\underbrace{s(I_{\text{small}} \cap A \cap \tilde{F}_2)}_{s_j y_j} + \underbrace{\gamma}_0 - x_j D = s_j y_j - x_j D \geq 0.$$

(b) If  $j \in F_1$ , then

$$\underbrace{s(I_{\text{small}} \cap A \cap \tilde{F}_2)}_0 + \underbrace{\gamma}_{s_j y_j} - x_j D = s_j y_j - x_j D \geq 0.$$

Thus it is also non-negative, and Bob can compute it on his own in this case.

This concludes the communication problem in the case where either  $\tilde{F}_2 = \emptyset$ , or  $\tilde{F}_2 = \{j\}$  where  $j \in A \cup F_1$ .

**Case 2:**  $\tilde{F}_2 = \{j\}$  and  $j \in F_2$ . In this case  $\gamma = x_j D$ . This case is quite similar to Case 1, with the difference being that Bob checks at the beginning if

$$s(I_{\text{large}} \cap \tilde{F}_1) \geq D - \tilde{\Delta},$$

i.e., without including  $\tilde{F}_2$  compared to (5.15).

If the condition was indeed satisfied, then the same reasoning as the first part of Case 1 resolves this case. Otherwise, we get

$$s(I_{\text{small}} \cap \tilde{F}_1) + x_j D > \tilde{\sigma} + (1 - \delta)\tilde{U}, \quad (5.20)$$

and using Equation (5.18) from the second part of Case 1 yields that that *first four* terms in this case are non-negative and easy to compute. Similarly, adding and subtracting  $(\tilde{\sigma} + (1 - \delta)\tilde{U})$  to the *last four* terms of (5.18), and rearranging the terms we get

$$\left( s(I_{\text{small}} \cap \tilde{F}_1) - \tilde{\sigma} - (1 - \delta)\tilde{U} + x_j D \right) + \left( \tilde{\sigma} + (1 - \delta)\tilde{U} - \alpha U - s(I_{\text{small}} \cap A) \right).$$

The first part of the summation is non-negative by Equation (5.20) and can be computed by Bob. The second part is the same as the second part in Equation (5.19). It is non-negative by definition and can be computed by Alice. This completes the proof.

This concludes the promised communication problem in the case where Alice is given an infeasible tuple  $(A, F_1, F_2)$ , and Bob is given a canonical feasible solution with  $\tilde{F}_3 = \emptyset$ . As argued in Section 5.4.1, this generalizes to any canonical feasible solution *without* any restriction on  $\tilde{F}_3$ .

### 5.4.3 Randomized Protocol for Arbitrary Feasible Solutions.

We now extend the communication protocol of canonical feasible solutions to arbitrary feasible solutions. To that end, we denote by  $\mathcal{R} = \{(x^1, y^1), (x^2, y^2), \dots, (x^r, y^r)\}$  the set of all canonical feasible solutions.

In this non-restricted setting, Alice still gets an infeasible tuple  $(A, F_1, F_2)$ , but Bob gets a feasible solution  $(x, y)$  that is not necessarily canonical, and the goal remains to compute the slack of the corresponding flow-cover inequality (i.e., Equation (5.10)) in expectation. We show that the communication protocol that we developed in the previous section can be used as a black-box to handle this general case, by noting that any feasible solution  $(x, y)$  can be written as a convex combination of canonical feasible solutions  $(x^1, y^1), (x^2, y^2), \dots, (x^r, y^r)$ . In other words, there exists  $\lambda_1, \lambda_2, \dots, \lambda_r \geq 0, \sum_{k=1}^r \lambda_k = 1$ , such that

$$(x, y) = \sum_{k=1}^r \lambda_k (x^k, y^k). \quad (5.21)$$

This is formalized in Lemma 5.11.

To see that this is enough, note that the expansion in Equation (5.21) of  $(x, y)$  allows us to rewrite slack of the flow-cover inequalities in (5.10) as

$$\begin{aligned} & s'(F_1 \cap B) + x(F_2 \cap B) - \alpha U \\ &= \sum_{i \in F_1} s'_i \sum_{k=1}^r \lambda_k y_i^k + \sum_{i \in F_2} \sum_{k=1}^r \lambda_k x_i^k D - \alpha U \\ &= \sum_{k=1}^r \lambda_k \left( \sum_{i \in F_1} s'_i y_i^k + \sum_{i \in F_2} x_i^k D - \alpha U \right). \end{aligned}$$

Thus in order to compute the slack in expectation, Bob samples a canonical feasible solution  $(x^k, y^k) \in \mathcal{R}$  with probability  $\lambda_k$ , then together with Alice, they compute the slack of

$$\sum_{i \in F_1} s'_i y_i^k + \sum_{i \in F_2} x_i^k D - \alpha U$$

as discussed in the previous section.

It remains to prove that any feasible solution can indeed be written as a convex combi-

nation of canonical feasible solutions. This is formalized in Lemma 5.11.

**Lemma 5.11.** *Let  $\mathcal{R} = \{(x^1, y^1), (x^2, y^2), \dots, (x^r, y^r)\}$  be the set of all the canonical feasible solutions for the flow cover problem, then any feasible solution  $(x, y)$  can be written as*

$$(x, y) = \sum_{k=1}^r \lambda_k (x^k, y^k),$$

such that  $\lambda_k \geq 0$  for all  $1 \leq k \leq r$ , and  $\sum_k \lambda_k = 1$ .

*Proof.* Given a feasible solution  $(x, y)$ , define its support  $\tilde{F}^{x,y} = \{i : i \in F, \text{ and } y_i = 1\}$ , and define the set  $\mathcal{R}^{x,y}$  to be the set of all canonical feasible solutions whose support equals  $\tilde{F}^{x,y}$ , i.e.,

$$\mathcal{R}^{x,y} = \{(x', y) : (x', y) \in \mathcal{R}\} \subseteq \mathcal{R}.$$

Without loss of generality, we assume that  $\tilde{F}^{x,y} = [n]$  to simplify the presentation.

We now consider the following polytope  $P(y)$ :

$$P(y) = \left\{ \begin{array}{l} z \in [0, 1]^n, \\ (*) \sum_{i=1}^n z_i = 1, \\ (**) 0 \leq z_i \leq \frac{s_i y_i}{D} \end{array} \quad \begin{array}{l} \text{such that:} \\ \text{for all } 1 \leq i \leq n \end{array} \right\}$$

Note that for any feasible solution  $(x, y)$  to the flow cover problem, we have that  $x \in P(y)$ . Moreover, we get from Definition 5.10 that for any canonical feasible solution  $(x', y) \in \mathcal{R}^{x,y}$ , all except at most one item  $i \in [n]$ , either has  $x'_i = 0$  or  $x'_i D = s_i y_i$ . Thus  $x'$  satisfies at least  $n - 1$  linearly independent constraints of type  $(**)$  with equality. Conversely, if a point  $x \in P(y)$  satisfies at least  $n - 1$  constraints of type  $(**)$  with equality, then  $(x, y) \in \mathcal{R}^{x,y}$ .

Recall that a point  $z$  is an *extreme point* solution of  $P(y)$  iff there are  $n$  linearly independent constraints that are set to equality by  $z$ . Since constraint  $(*)$  is an equality constraint and is linearly independent from any set of  $n - 1$  constraints from  $(**)$ , we conclude that  $\{x' : (x', y) \in \mathcal{R}^{x,y}\}$  is the set of all extreme points of  $P(y)$ . This implies that for any  $x \in P(y)$ , there exists  $\lambda_k \geq 0$  for each  $1 \leq k \leq r$  such that  $\sum_k \lambda_k = 1$  and

$$x = \sum_{k=1}^r \lambda_k x^k.$$

Since all these points have the same  $y$ -support, it follows that

$$(x, y) = \sum_{k=1}^r \lambda_k (x^k, y^k).$$

□

## 5.5 Algorithmic Aspects.

Theorem 5.1 relies on the *existence* of a quasi-polynomial size extended formulation for the weakened knapsack cover inequalities. However, we do not know how to *construct* the full extended formulation in quasi-polynomial time. Nevertheless, there is a way to use the extended formulation algorithmically, which we describe here.

We adopt a more general point of view, since the findings of this section are applicable beyond the context of the knapsack cover inequalities. Consider any system of  $p$  inequalities  $A_1x \geq b_1, \dots, A_px \geq b_p$ , and  $q$  solutions  $x^{(1)}, \dots, x^{(q)} \in \mathbb{R}^n$  of this system. In the context of the MIN-KNAPSACK problem, the inequalities  $A_ix \geq b_i$  ( $i \in [p]$ ) are all the weakened knapsack cover inequalities and the solutions  $x^{(j)}$  ( $j \in [q]$ ) are all the feasible solutions  $x \in \{0, 1\}^n$ . Typically, both  $p$  and  $q$  are exponentially large as functions of  $n$ .

To this data corresponds a slack matrix  $S \in \mathbb{R}_{\geq 0}^{p \times q}$  defined by  $S_{ij} := A_ix^{(j)} - b_i$ . As observed by Yannakakis [107], every non-negative factorization  $S = FV$  where  $F \in \mathbb{R}_{\geq 0}^{p \times r}$  and  $V \in \mathbb{R}_{\geq 0}^{r \times q}$  determines a system

$$\begin{aligned} A_ix - b_i &= F_iy \quad \forall i \in [p] \\ y &\geq 0 \end{aligned} \tag{5.22}$$

whose projection to the  $x$ -space gives a polyhedron  $\{x \in \mathbb{R}^n \mid \exists y \in \mathbb{R}^r : Ax - b = Fy, y \geq 0\}$  containing each of the solutions  $x^{(j)}$  and being contained in each of the halfspaces  $A_ix \geq b_i$ .

Usually, the number  $p$  of equations in (5.22) is much bigger than both the number  $n$  of  $x$ -variables and rank  $r$  of the non-negative factorization. Thus the equation system is largely overdetermined and can be replaced by a smaller equivalent subsystem with at most  $n + r$  equations. However, it is not obvious to tell efficiently what are the indices  $i$  for which the corresponding equation in (5.22) should be kept.

To avoid this difficulty, we assume that the way in which we want to use the extended formulation (shorthand: EF)  $Ax - b = Fy, y \geq 0$  is to solve the LP  $\min\{c^T x \mid Ax \geq b\}$  for a *given* objective vector  $c \in \mathbb{R}^n$ , through the extended formulation.

For  $I \subseteq [p]$ , consider the linear program

$$\begin{aligned} \text{LP}(I) : \quad & \min c^T x \\ & \text{s.t. } A_ix - b_i = F_iy \quad \forall i \in I \\ & \quad y \geq 0. \end{aligned}$$

In fact, we will only need to consider sets  $I$  of size at most  $n + r \ll p$ .

Algorithm 4 solves the LP  $\min\{c^\top x \mid Ax \geq b\}$  in several steps. In each step, it solves the smaller LP( $I$ ) where  $I \subseteq [p]$  and calls a separation routine to check whether  $x^*$ , the  $x$ -part of the optimum solution found, satisfies  $Ax \geq b$  or not. In the first case, it returns  $x^*$  and stops. In the second case, it adds the index  $i^*$  of any violated constraint to  $I$  and continues. At the beginning of the algorithm,  $I$  is initialized to  $[n]$ . To avoid technicalities, we assume that LP( $[n]$ ) is bounded. For the sake of concreteness, we assume furthermore that the  $n$  first inequalities of the system  $Ax \geq b$  are the nonnegativity inequalities  $x_1 \geq 0, \dots, x_n \geq 0$ , and that  $c \in \mathbb{R}_{\geq 0}^n$ .

---

**Algorithm 4** Cutting-plane algorithm to solve  $\min\{c^\top x \mid Ax \geq b\}$  through EF  $Ax - b = Fy, y \geq 0$

---

```

1: initialize  $I \leftarrow [n]$ 
2: initialize feasible  $\leftarrow$  false
3: repeat
4:   solve LP( $I$ ), get optimum solution  $(x^*, y^*)$ 
5:   if there exists  $i^* \in [p]$  such that  $A_{i^*}x^* < b_{i^*}$  then
6:     add  $i^*$  to  $I$ 
7:   else
8:     set feasible  $\leftarrow$  true
9:   end if
10: until feasible = true
11: return  $x^*$ 

```

---

To analyze the running time of the algorithm, we make the following assumptions:

- the size of each coefficient in (5.22) and each  $c_i$  is upper-bounded by  $\Delta = \Delta(n)$ ;
- the separation problem (given  $x^* \in \mathbb{R}^n$ , find an index  $i^* \in [p]$  such that  $A_{i^*}x^* < b_{i^*}$  or report that no such index exists) can be solved in  $T_{\text{sep}}(n)$  time;
- each single equation in (5.22) can be written down in  $T_{\text{constr}}(n)$  time;
- LP( $I$ ) can be solved in time  $T_{\text{solve}}(n)$  for any set  $I$  of size at most  $n + r$ , where  $r = r(n)$  is the rank of the nonnegative factorization giving rise to the extended formulation  $Ax - b = Fy, y \geq 0$ .

Notice that  $T_{\text{solve}}(n) = O(n^3(n + r)\Delta)$  if an interior point method is used to solve LP( $I$ ).

**Lemma 5.12.** *Under the above assumptions, the main loop of Algorithm 4 is executed at most  $r + 1$  times. Thus the complexity of Algorithm 4 is  $O(r \cdot (T_{\text{solve}}(n) + T_{\text{sep}}(n) + T_{\text{constr}}(n)))$ .*

*Proof.* The result follows directly from the simple observation that each time a new equation  $A_{i^*}x - b_{i^*} = F_{i^*}y$  added to the system  $A_i x - b_i = F_i y$  ( $i \in I$ ), it is linearly

independent from the current equations in the system. Notice that by assumption, the algorithm starts with  $n$  linearly independent constraints. By the above observation, we always have  $|I| \leq n + r$ .  $\square$

From now on, we assume that the non-negative factorization of the slack matrix  $S$  comes from a communication protocol with non-negative outputs computing  $S$  in expectation. The protocol is specified by a binary *protocol tree*, in which each internal node is owned either by Alice or Bob, and each leaf corresponds to an output of the protocol. At each internal node  $u$  owned by Alice, a *branching probability*  $p_{\text{branch}}(i, u) \in [0, 1]$  is given for each input  $i \in [p]$  of Alice. Similarly for each internal node  $v$  owned by Bob, we are given a branching probability  $q_{\text{branch}}(j, v) \in [0, 1]$ , where  $j \in [q]$  is Bob's input. These branching probabilities specify the chance for the protocol of following the left branch. Finally, each leaf  $\ell$  has a nonnegative number  $\lambda(\ell) \in \mathbb{R}_{\geq 0}$  attached to it.

The corresponding extended formulation can be written as

$$\begin{aligned} A_i x - b_i &= \sum_{\ell \text{ leaf}} p_{\text{reach}}(i, \ell) \cdot y_\ell \quad \forall i \in [p] \\ y_\ell &\geq 0 \quad \forall \ell \text{ leaf} \end{aligned} \tag{5.23}$$

where  $p_{\text{reach}}(i, u)$  denotes the probability of reaching node  $u$  of the protocol tree on any input pair of the form  $(i, *)$ .

**Lemma 5.13.** *Let  $\Delta$  be any number that is at least*

$$\max\{-\log(p_{\text{reach}}(i, \ell)) \mid i \in [p], \ell \text{ leaf}, p_{\text{reach}}(i, \ell) > 0\},$$

and let  $h$  denote the height of the protocol tree. For any fixed  $i \in [p]$ , one can write down the right-hand side of the corresponding equation in (5.23) in  $O(2^h \Delta \log \Delta \log \log \Delta)$  time and  $O(2^h \Delta)$  space.

*Proof.* Clearly, at the root of the protocol tree, we have  $p_{\text{reach}}(i, \text{root}) = 1$ . At an internal node  $u$  owned by Alice with left child  $v$  and right child  $w$ , we have

$$p_{\text{reach}}(i, v) = p_{\text{reach}}(i, u) \cdot p_{\text{branch}}(i, u),$$

and

$$p_{\text{reach}}(i, w) = p_{\text{reach}}(i, u) \cdot (1 - p_{\text{branch}}(i, u)) = p_{\text{reach}}(i, u) - p_{\text{reach}}(i, v).$$

In case  $u$  is owned by Bob, we simply have  $p_{\text{reach}}(i, v) = p_{\text{reach}}(i, w) = p_{\text{reach}}(i, u)$  since the behavior of the communication protocol at node  $u$  on input pair  $(i, j)$  is independent of  $i$ .

Using this, we can compute recursively  $p_{\text{reach}}(i, u)$  for all nodes  $u$  of the protocol tree, and thus for the leaves of the tree. All arithmetic operations are performed on numbers of at most  $O(\Delta)$  bits. If we use the Schoolbook algorithm for subtraction and the Schönhage-Strassen algorithm for multiplication, we obtain the claimed bounds for the time- and space-complexity.  $\square$

Now, we discuss how Algorithm 4 and its analysis apply to the (weakened) knapsack cover inequalities and the corresponding slack matrix  $(S_{ab}^\varepsilon)_{a \in f^{-1}(0), b \in f^{-1}(1)}$  as in (5.5), where  $f$  is the weighted threshold function defining the knapsack. In order to do that, we first have to construct the protocol tree of the protocol described in the proof of Lemma 5.7. We claim that this can be done in time  $(1/\varepsilon)^{O(1)} n^{O(\log n)}$ .

The protocol has several deterministic parts (in which the branching probabilities are in  $\{0, 1\}$  locally). Each corresponds to the resolution of a Karchmer-Wigderson game. For writing down the corresponding subtrees of the protocol tree, we just need  $\log^2(n)$ -depth monotone circuits of fan-in 2 for computing certain truncations of the weighted threshold function  $f$ . The translation of the circuit into a protocol tree follows the standard construction of Karchmer and Wigderson [64]. For constructing the circuits, we rely either on the construction of Beimel and Weinreb [15, 16] or the simpler and more recent construction of Chen, Oliveira and Servedio [34]. Both constructions can be executed in  $n^{O(1)}$  time.

The remaining parts of the protocol can be readily translated into the corresponding subtrees of the protocol tree.

Since the reaching probabilities in the protocol tree can be written down with  $O(\log n)$  bits, each coefficient in the right-hand side of (5.23) can be written down in  $O(\log n)$  bits. Assuming as before that all item sizes and demand can be written down with  $O(n \log n)$  bits (which is without loss of generality), the coefficients of the left-hand side of (5.23) can be written down with  $O(n \log n)$  bits. Therefore, we can take  $\Delta = O(n \log n)$

From what precedes and Lemma 5.13, we have that  $T_{\text{constr}}(n) = (1/\varepsilon)^{O(1)} n^{O(\log n)}$ . Moreover, Lemma 5.7 gives  $r(n) = (1/\varepsilon)^{O(1)} n^{O(\log n)}$ .

For the separation routine, we deviate significantly from Algorithm 4: instead of using an exact separation routine (efficient exact separation of the knapsack cover inequalities is an open problem), we rely on a separation trick from Carr *et al.* [29]. That is, we simply check if the knapsack cover inequality for  $A := \{i \in [n] \mid x_i^* \geq 1/2\}$  is satisfied. This is enough to guarantee that the modified Algorithm 4 computes a quantity that is within a  $2 + \varepsilon$  factor of the integer optimum for that particular cost function  $c$ . Unfortunately, by relying on the pseudo-separation of Carr *et al.*, we cannot guarantee that the modified Algorithm 4 optimizes exactly over all weakened knapsack cover inequalities.

If we further assume that the coefficients of  $c$  can be written with  $O(n \log n)$  bits, we conclude that one can find a  $(2 + \varepsilon)$ -approximation of  $\min\{\sum_{i=1}^n c_i x_i \mid \sum_{i=1}^n s_i x_i \geq D, x \in \{0, 1\}^n\}$  in time  $(1/\varepsilon)^{O(1)} n^{O(\log n)}$ , without relying on the ellipsoid algorithm, using our extended formulation.

### 5.6 Conclusion.

After the recent series of strong negative results on extended formulations, we have presented a positive result inspired by a connection to monotone circuits. Namely, we obtain the first quasi-polynomial-size LP relaxation of MIN-KNAPSACK with constant integrality gap from polylog-depth circuits for weighted threshold functions.



## 6 Scheduling Problems

In this chapter, we present an application for using integrality gap instances of LP relaxations of a combinatorial problem, in order to prove the inapproximability results of this problem in general computational models. These integrality gap instances in fact lead us to studying structural properties of  $k$ -partite graphs.

We show a close connection between structural hardness for  $k$ -partite graphs and tight inapproximability results for scheduling problems with precedence constraints. Assuming a natural but nontrivial generalization of the bipartite structural hardness result of [8], we obtain a hardness of  $2 - \epsilon$  for the problem of minimizing the makespan for scheduling precedence-constrained jobs with preemption on identical parallel machines. This matches the best approximation guarantee for this problem [53, 47]. Assuming the same hypothesis, we also obtain a super constant inapproximability result for the problem of scheduling precedence-constrained jobs on related parallel machines, making progress towards settling an open question in both lists of ten open questions by Williamson and Shmoys [105], and by Schuurman and Woeginger [97].

The study of structural hardness of  $k$ -partite graphs is of independent interest, as it captures the intrinsic hardness for a large family of scheduling problems. Other than the ones already mentioned, this generalization also implies tight inapproximability to the problem of minimizing the weighted completion time for precedence-constrained jobs on a single machine, and the problem of minimizing the makespan of precedence-constrained jobs on identical parallel machine, hence unifying the results of Bansal and Khot [8] and Svensson [101], respectively.

### 6.1 Introduction

The study of scheduling problems is motivated by the natural need to efficiently allocate limited resources over the course of time. Though some scheduling problems can be solved to optimality in polynomial time, others turn out to be NP-hard. This difference in

computational complexity can be altered by many factors, from the machines model that we adopt, to the requirements imposed on the jobs, as well as the optimality criterion of a feasible schedule. For instance, if we are interested in minimizing the completion time of the latest job in the schedule (known as the maximum *makespan*), then the scheduling problem is NP-hard to approximate within a factor of  $3/2 - \epsilon$ , for any  $\epsilon > 0$ , if the machines are *unrelated*, whereas it admits a Polynomial Time Approximation Scheme (PTAS) for the case of *identical parallel* machines [59]. Adopting a model in between the two, in which the machines run at different speeds, but doing so uniformly for all jobs (known as *uniform parallel* machines), also leads to a PTAS for the scheduling problem [60].

**Scheduling with Precedence Constraints.** Although the former discussion somehow suggests a similarity in the complexity of scheduling problems between identical parallel machines and uniform parallel machines, our hopes for comparably performing algorithms seem to be shattered as soon as we add precedence requirements among the jobs. On the one hand, we know how to obtain a 2-approximation algorithm for the problem where the parallel machines are identical [53, 47] (denoted as  $P|prec|C_{max}$  in the language of [54]); on the other hand, the best approximation algorithm known to date for the uniform parallel machines case (denoted as  $Q|prec|C_{max}$ ), gives a  $\log(m) / \log \log(m)$ -approximation guarantee [79],  $m$  being the number of machines. In fact, obtaining a constant factor approximation algorithm for the latter or ruling out any such result, is a major open problem in the area of scheduling algorithms. Perhaps as a testament to that is the fact that it is listed by Williamson and Shmoys [105] as Open Problem 8, and by Schuurman and Woeginger [97] as Open Problem 2.

**Parallel Machines with and without Preemption.** Moreover, our understanding of scheduling problems even on the same model of machines does not seem to be complete either. On the positive side, it is easy to see that the maximum makespan of any feasible schedule for  $P|prec|C_{max}$  is at least  $\max\{L, n/m\}$ , where  $L$  is the length of the longest chain of precedence constraints in our instance, and  $n$  and  $m$  are the number of jobs and machines, respectively. The same lower bound still holds when we allow preemption, i.e., the scheduling problem  $P|prec, pmtn|C_{max}$ . Given that both 2-approximation algorithms of [53] and [47] rely in their analysis on the aforementioned lower bound, they also yield a 2-approximation algorithm for  $P|prec, pmtn|C_{max}$ . However, on the negative side, our understanding for  $P|prec, pmtn|C_{max}$  is much less complete. For instance, we know that it is NP-hard to approximate  $P|prec|C_{max}$  within any constant factor strictly better than  $4/3$  [77], and assuming (a variant of) the Unique Games Conjecture (UGC), the latter lower bound is improved to 2 [101]. However for  $P|prec, pmtn|C_{max}$ , only NP-hardness is known. It is important to note here that the hard instances yielding the  $(2 - \epsilon)$  hardness for  $P|prec|C_{max}$  are easy instances for  $P|prec, pmtn|C_{max}$ . Informally speaking, the hard instances for  $P|prec|C_{max}$  can be thought of as  $k$ -partite graphs, where

each partition has  $m + 1$  vertices that correspond to  $n := m + 1$  jobs, and the edges from a layer to the layer above it emulate the precedence constraints. The goal is to schedule these  $nk$  jobs on  $m$  machines. If the  $k$ -partite graph is *complete*, then any feasible schedule has a makespan of at least  $2k$ , whereas if the graph was a collection of *perfect matchings* between each two consecutive layers, then there exists a schedule whose makespan is  $k + 1$ <sup>1</sup>. However, if we allow preemption, then it is easy to see that even if the  $k$ -partite graph is complete, one can nonetheless find a feasible schedule whose makespan is  $k + 1$ .

**From Graph Ordering to Scheduling Inapproximability.** The effort of closing the inapproximability gap between the best approximation guarantee and the best known hardness result for some scheduling problems was successful in recent years; two of the results that are of particular interest for us are [8] and [101]. Bansal and Khot study in [8] the scheduling problem  $1|\text{prec}|\sum_j w_j C_j$ , the problem of scheduling precedence constrained jobs on a single machine, with the goal of minimizing the weighted sum of completion time, and they prove tight inapproximability results for it, assuming a variant of the Unique Games Conjecture. Similarly, Svensson proves in [101] a hardness of  $2 - \epsilon$  for  $P|\text{prec}|C_{max}$ , assuming the same conjecture. In fact, both papers rely on a structural hardness result for bipartite graphs, first introduced in [8], by reducing a bipartite graph to a scheduling instance, which leads to the desired hardness factor.

**Our results.** We propose a natural but non-trivial generalization of the structural hardness result of [8] from bipartite to  $k$ -partite graphs; this generalization captures the intrinsic hardness of a large family of scheduling problems.

On the one hand, our generalization rules out any constant factor polynomial time approximation algorithm for the scheduling problem  $Q|\text{prec}|C_{max}$ . On the other hand, one may speculate that the preemption flexibility, when added to the scheduling problem  $P|\text{prec}|C_{max}$ , renders this problem easier, especially that the hard instances of the latter problem become easy when preemption is allowed. Contrary to such speculations, our generalization to  $k$ -partite graphs enables us to prove that it is NP-hard to approximate the scheduling problem  $P|\text{prec}, \text{pmtn}|C_{max}$  within any factor strictly better than 2. Formally, we prove the following:

**Theorem 6.1.** *Assuming Hypothesis 6.14, it is NP-hard to approximate the scheduling problems  $P|\text{prec}, \text{pmtn}|C_{max}$  within any constant factor strictly better than 2, and  $Q|\text{prec}|C_{max}$  within any constant factor.*

This suggests that the intrinsic hardness of a large family of scheduling problems seems

<sup>1</sup>In fact, the gap is between  $k$ -partite graphs that have nice structural properties in the completeness case, and behave like node expanders in the soundness case.

to be captured by structural hardness results for  $k$ -partite graphs. For the case of  $k = 2$ , our hypothesis coincides with the structured bipartite hardness result of [8], and yields the following result:

**Theorem 6.2.** *Assuming a variant of the Unique Games Conjecture, it is NP-hard to approximate the scheduling problem  $P|prec, pmtn|C_{max}$  within any constant factor strictly less than  $3/2$ .*

In fact, the  $3/2$  lower bound also holds even if we only assume that  $1|prec|\sum_j w_j C_j$  is NP-hard to approximate within any factor strictly better than 2, by noting the connection between the latter and a certain bipartite ordering problem. This connection was observed and used by Svensson [101] to prove tight hardness of approximation lower bounds for  $P|prec|C_{max}$ , and this yields a somehow stronger statement; even if the Unique Games Conjecture turns out to be false,  $1|prec|\sum_j w_j C_j$  might still be hard to approximate to within a factor of  $2 - \epsilon$ , and our result for  $P|prec, pmtn|C_{max}$  will still hold as well. Formally,

**Corollary 6.3.** *For any  $\epsilon > 0$ , and  $\eta \geq \eta(\epsilon)$ , where  $\eta(\epsilon)$  tends to 0 as  $\epsilon$  tends to 0, if  $1|prec|\sum_j w_j C_j$  has no  $(2 - \epsilon)$ -approximation algorithm, then  $P|prec, pmtn|C_{max}$  has no  $(3/2 - \eta)$ -approximation algorithm.*

*Remark 6.4.* Theorems 6.1 and 6.2 and Corollary 6.3 also hold for the special case of the  $P|prec, pmtn|C_{max}$  problem where all the jobs have size 1, denoted by  $P|prec, pmtn, p_j = 1|C_{max}$ .

Although we believe that Hypothesis 6.14 holds, the proof is still eluding us, even if we assume the Unique Games Conjecture. Nonetheless, understanding the structure of  $k$ -partite graphs seems to be a very promising direction for understanding the inapproximability of scheduling problems, due to its manifold implications in the latter problems. As mentioned earlier, a similar structure for bipartite graphs was proved assuming a variant of the Unique Games Conjecture in [8] (see Theorem 6.7).

**Further Related Work** The scheduling problem  $P|prec, pmtn|C_{max}$  was first shown to be NP-hard by Ullman [104]. However, if we drop the precedence rule, the problem can be solved to optimality in polynomial time [81]. Similarly, if the precedence constraint graph is a *tree* [84, 85, 51] or the number of machines is 2 [84, 85], the problem also becomes solvable in polynomial time. Yet, for an arbitrary precedence constraints structure, it remains open whether the problem is polynomial time solvable when the number of machines is a constant greater than or equal to 3 [105]. A closely related problem to  $P|prec, pmtn|C_{max}$  is  $P|prec|C_{max}$ , in which preemption is not allowed. In fact the best 2-approximation algorithms known to date for  $P|prec, pmtn|C_{max}$  were originally designed to approximate  $P|prec|C_{max}$  [53, 47], by noting the common lower bound for a makespan to any feasible schedule for both problems. As mentioned earlier, [77]

and [101] prove a  $4/3 - \epsilon$  NP-hardness and  $2 - \epsilon$  UGC-hardness, respectively, for  $P|prec|C_{max}$ , for any  $\epsilon > 0$ . However, to this date, only NP-hardness is known for the  $P|prec, pmtn|C_{max}$  scheduling problem. Although one may speculate that allowing preemption might enable us to get better approximation guarantees, no substantial progress has been made in this direction since [53] and [47].

One can easily see that the scheduling problem  $P|prec|C_{max}$  is a special case of  $Q|prec|C_{max}$ , since it corresponds to the case where the speed of every machine is equal to 1, and hence the  $(4/3 - \epsilon)$  NP-hardness of [77] and the  $(2 - \epsilon)$  UGC-hardness of [101] also apply to  $Q|prec|C_{max}$ . Nonetheless, no constant factor approximation algorithm for this problems is known; a  $\log(m)$ -approximation algorithm was designed by Chudak and Shmoys [36], and Chekuri and Bender [33] independently, where  $m$  is the number of machines. Very recently, the upper bound was improved to  $\log(m) / \log \log(m)$  by Li [79]

**Outline** We begin in Section 6.2 by formally defining our scheduling problems of interest, and introducing the notation that we use throughout this chapter. In Section 6.3, we present the natural LP formulation for the  $P|prec, pmtn, p_j = 1|C_{max}$  problem, and show that the integrality gap of this LP is 2. Although we are interested in the inapproximability of the problem in general computational models (i.e., not only the limitations of LP based algorithms), the structure of the integrality gap instance that we construct suggests a family of instances that seems to capture the intrinsic hardness of the problem. Inspired by these integrality gap instances, we then show in Section 6.4 that known structural hardness results for *bipartite* graphs yield a  $3/2$  inapproximability assuming a variant of the Unique Games Conjecture for the  $P|prec, pmtn|C_{max}$  problem, and motivate the need for a *generalization* to *k-partite* graphs in order to amplify our inapproximability result to 2 instead of  $3/2$ . We then propose our novel hypothesis for *k-partite* graphs (Hypothesis 6.14) that will play an essential role in the hardness proofs of Section 6.6. Namely, we use it in Section 6.6.1 to show a  $2 - \epsilon$  inapproximability for  $P|prec, pmtn|C_{max}$ , and in Section 6.6.2 to prove a super constant inapproximability result for the scheduling problem  $Q|prec|C_{max}$ .

## 6.2 Preliminaries

In the scheduling problems that we consider, we are given a set  $\mathcal{M}$  of  $m$  machines and a set  $\mathcal{J}$  of  $n$  jobs with precedence constraints, and the goal is to find a feasible schedule that minimizes the makespan, i.e., the maximum completion time. We say that a job  $J_i \in \mathcal{J}$  is a predecessor of a job  $J_j \in \mathcal{J}$ , and write it  $J_i \prec J_j$ , if in any feasible schedule,  $J_j$  cannot start executing before the completion of job  $J_i$ . Similarly, for two sets of jobs  $\mathcal{F}_i \subseteq \mathcal{J}$  and  $\mathcal{F}_j \subseteq \mathcal{J}$ ,  $\mathcal{F}_i \prec \mathcal{F}_j$  is equivalent to saying that all the jobs in  $\mathcal{F}_j$  are successors of all the jobs in  $\mathcal{F}_i$ .

The main problems that we consider in this chapter are the following variants of makespan minimization scheduling with precedence constraints:

**P|prec, pmtn|C<sub>max</sub>**: In this model, the machines are assumed to be parallel and identical, i.e., the processing time of a job  $J_j \in \mathcal{J}$  is the same on any machine  $M_i \in \mathcal{M}$  ( $p_{i,j} = p_j$  for all  $M_i \in \mathcal{M}$ ). Furthermore, preemption is allowed, and hence the processing of a job can be paused and resumed at later stages, not necessarily on the same machine.

**P|prec, pmtn,  $p_j = 1$ |C<sub>max</sub>**: This is the same as **P|prec, pmtn|C<sub>max</sub>** with the restriction that each job has processing time  $p_j = 1$ . Note that proving a lower bound on the approximability of **P|prec, pmtn,  $p_j = 1$ |C<sub>max</sub>** implies the same lower bound for **P|prec, pmtn|C<sub>max</sub>**.

**Q|prec|C<sub>max</sub>**: In this model, the machines are assumed to be parallel and uniform, i.e., each machine  $M_i \in \mathcal{M}$  has a speed  $s_i$ , and the time it takes to process job  $J_j \in \mathcal{J}$  on this machine is  $p_j/s_i$ .

Moreover, the complexity hypothesis that we suggest in this chapter (i.e., Hypothesis 6.14) also yields the known results for the following two scheduling problems:

**P|prec|C<sub>max</sub>**: This is a restricted version of the **P|prec, pmtn|C<sub>max</sub>** problem, where the preemption is not allowed.

**1|prec| $\sum_j w_j C_j$** : In this model, we only have *one* machine, and each job  $J_j \in \mathcal{J}$  has an associated weight  $w_j \in \mathbb{R}^+$ . The goal in this problem is to find a schedule for these jobs on this machine in a way that minimizes the weighted completion time of the jobs. Namely, let  $C_j^\sigma$  be the completion time for a job  $J_j \in \mathcal{J}$  in a feasible schedule  $\sigma$ , then the goal is to find a feasible schedule  $\sigma^*$  that minimizes

$$\sum_{J_j \in \mathcal{J}} w_j C_j^\sigma.$$

### 6.3 Integrality Gap For **P|prec, pmtn, $p_j = 1$ |C<sub>max</sub>**

Our inapproximability results for the scheduling problems of interest are motivated by a family of hard instances that fools a certain linear programming relaxation of **P|prec, pmtn,  $p_j = 1$ |C<sub>max</sub>**. Although this approach usually only rules out *good* approximation algorithms based on this particular LP relaxation, the structure of these hard instances naturally relates to the hardness of a certain graph ordering problem. This graph ordering problem is known to be hard on bipartite graphs assuming the Unique Games Conjecture [8] (See Theorem 6.7), and what we basically hypothesize in this chapter is that a natural generalization of this problem on  $k$ -partite graphs is also hard.

In order to motivate our  $k$ -partite graph ordering problem hypothesis, we present in this section the natural LP relaxation of the  $P|_{\text{prec, pmtn}, p_j = 1}|C_{\text{max}}$  problem, and construct its corresponding family of integrality gap instances. The connection between these instances, and our Hypothesis 6.14 will then become easy to see.

### 6.3.1 LP Relaxation of $P|_{\text{prec, pmtn}, p_j = 1}|C_{\text{max}}$

In this section, we will be interested in a feasibility Linear Program, that we denote by [SCHED-LP], for the scheduling problem  $P|_{\text{prec, pmtn}, p_j = 1}|C_{\text{max}}$ . For a makespan guess  $T$ , [SCHED-LP] has a set of indicator variables  $\{x_{j,t}\}$  for  $j \in [n]$  and  $t \in [T]$ . A variable  $x_{j,t}$  is intended to be the fraction of the job  $J_j$  scheduled between time  $t - 1$  and  $t$ . The optimal makespan  $T^*$  is then obtained by doing a binary search and checking at each step if the LP is feasible. [SCHED-LP] is defined as follows:

$$\sum_{j=1}^n x_{j,t} \leq m \quad \forall t \in [T] \quad (6.1)$$

$$\sum_{t=1}^T x_{j,t} = 1 \quad \forall j \in [n] \quad (6.2)$$

$$\sum_{t=1}^{t'-1} x_{\ell,t} + \sum_{t=t'+1}^T x_{k,t} \geq 1 \quad \forall J_\ell \prec J_k, \forall t' \in [T] \quad (6.3)$$

$$x_{j,t} \geq 0 \quad \forall j \in [n], \forall t \in [T]$$

To see that [SCHED-LP] is a valid relaxation for the scheduling problem  $P|_{\text{prec, pmtn}, p_j = 1}|C_{\text{max}}$ , note that constraint (6.1) guarantees that the number of jobs processed at each time unit is at most the number of machines, and constraint (6.2) says that in any feasible schedule, all the jobs must be assigned. Also any schedule that satisfies the precedence requirements must satisfy constraint (6.3).

### 6.3.2 Integrality Gap of the LP

In order to show that [SCHED-LP] has an integrality gap of 2, we start by constructing a family of instances showing an integrality gap  $3/2$  and gradually increase this gap to 2. The reason behind this incremental construction is that the  $3/2$  case captures the intrinsic hardness of the problem, and we show how to use it as basic building block for the construction of the target integrality gap instance of 2.

**Basic Building Block** We start by constructing an  $P|_{\text{prec, pmtn}, p_j = 1}|C_{\text{max}}$  scheduling instance  $\mathcal{S}(d)$  parametrised by a large constant  $d \geq 2$ , that shows that the integrality gap of [SCHED-LP] is  $3/2$ . This instance  $\mathcal{S}(d)$  will then constitute our main building block for the next reduction.

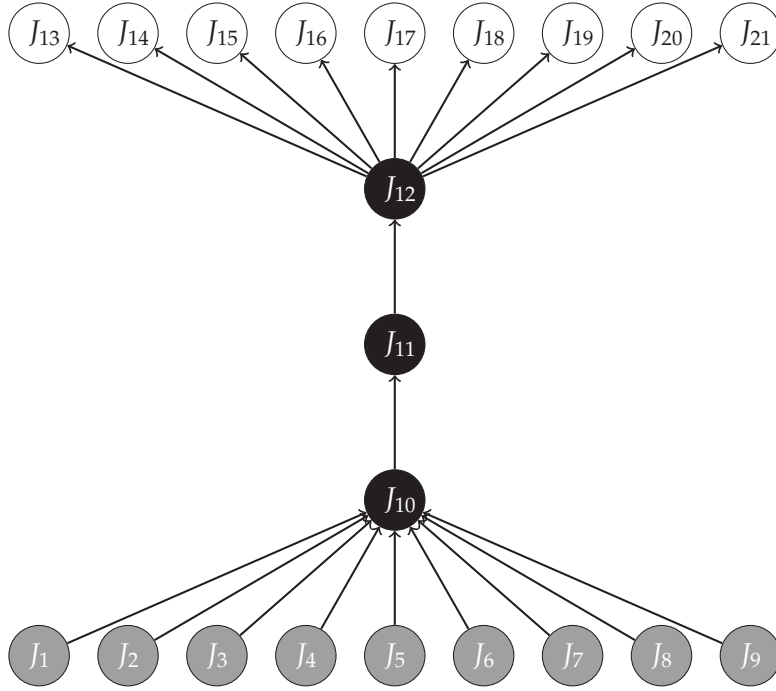


Figure 6.1 – Example of the integrality gap instance  $\mathcal{S}(4)$  with 3 machines of [SCHED-LP]. The number of machines is  $m$  and the number of jobs in this case is then  $n = 2md - (d - 1) = 21$ . Jobs whose color is grey, black and white correspond to jobs of layers 1, 2 and 3 respectively. A directed edge from a job  $J_i$  to a job  $J_k$  implies that  $J_i \prec J_k$ .

The instance  $\mathcal{S}(d)$  is defined over  $m$  machines and  $n$  jobs, where  $n = 2dm - (d - 1)$ . The dependencies between the jobs in  $\mathcal{S}(d)$  are as follows:

- The first  $dm - (d - 1)$  jobs  $J_1, \dots, J_{dm-(d-1)}$  have no predecessors [Layer 1].
- A chain of  $(d - 1)$  jobs  $J_{dm-(d-1)+1}, \dots, J_{dm}$  such that  $J_{dm-(d-1)+1}$  is the successor of all the jobs in the Layer 1, and  $J_{k-1} \prec J_k$  for  $k \in \{dm - (d - 1) + 2, \dots, dm\}$  [Layer 2].
- The last  $dm - (d - 1)$  jobs  $J_{dm+1}, \dots, J_{2dm-(d-1)}$  are successors of  $J_{dm}$  [Layer 3].

We illustrate an example of  $\mathcal{S}(d)$  for  $m = 3$  and  $d = 4$  in Figure 6.1.

We first show that  $\mathcal{S}(d)$  is an integrality gap of  $3/2$  for [SCHED-LP]. This basically follows from the following lemma:

**Lemma 6.5.** *Any feasible schedule for  $\mathcal{S}(d)$  has a makespan of at least  $3d - 2$ , however [SCHED-LP] has a feasible solution  $\{x_{j,t}\}$ , for  $t \in [2d]$  and  $j = 1, \dots, 2dm - (d - 1)$  of value  $2d$ . Moreover, for  $t = d + 1 + \ell$ ,  $\ell \in [d - 1]$ , the machines in the feasible LP solution can still execute a load of  $\frac{\ell}{d}$ , i.e.,  $m - \sum_{j \in \mathcal{S}} x_{j,t} \geq \frac{\ell}{d}$ .*



*Proof.* Consider the following fractional solution:

$$\begin{aligned}
 \text{[Layer 1]} \quad & x_{j,t} = \frac{1}{d} \\
 & \forall j \in \{1, 2, \dots, dm - (d - 1)\}, \forall t \in [d], \\
 \text{[Layer 2]} \quad & x_{dm-(d-1)+1,\ell+1} = x_{dm-(d-1)+2,\ell+2} = \dots = x_{dm,\ell+d-1} = \frac{1}{d} \\
 & \forall \ell \in [d], \\
 \text{[Layer 3]} \quad & x_{j,t} = \frac{1}{d} \quad \forall j \in \{dm + 1, dm + 2, \dots, 2dm - (d - 1)\}, \\
 & \forall t \in \{d + 1, d + 2, \dots, 2d\}.
 \end{aligned}$$

One can easily verify that each job  $J$  is completely scheduled, i.e.,  $\sum_{t=1}^{2d} x_{J,t} = 1$ . Moreover, the workload at each time step is at most  $m$ . To see this, we consider the following three types of time steps:

1. For  $t = 1$ , the workload is

$$\frac{1}{d} \times \left( \underbrace{dm - (d - 1)}_{\text{Layer 1}} \right) = m - 1 + \frac{1}{d} < m.$$

2. For  $t = 2, \dots, d$ , the workload is

$$\frac{1}{d} \times \left( \underbrace{dm - (d - 1)}_{\text{Layer 1}} + \underbrace{t - 1}_{\text{Layer 2}} \right) = m - 1 + \frac{t - 1}{d} < m.$$

3. For  $t = d + 1, \dots, 2d$ , the workload is

$$\frac{1}{d} \times \left( \underbrace{2d - t}_{\text{Layer 2}} + \underbrace{dm - (d - 1)}_{\text{Layer 3}} \right) = m - \frac{t - (d + 1)}{d} < m.$$

Note that in this feasible solution, we have that for  $t = d + 1 + i, i \in [d - 1]$ , the machines can still execute a load of  $\frac{i}{d}$ .

We have thus far verified that  $\{x_{j,t}\}$  satisfies the constraints (6.1) and (6.2) of [SCHED-LP]. Hence it remains to check (6.3), i.e., we need to check that for any two jobs  $J_k, J_\ell$  such that  $J_k$  is a predecessor of  $J_\ell$ , the following holds:

$$\sum_{\tilde{i}=1}^{t-1} x_{\ell,\tilde{i}} + \sum_{\tilde{i}=t+1}^T x_{k,\tilde{i}} \geq 1, \tag{6.4}$$

## Chapter 6. Scheduling Problems

---

for all  $t \in [2d]$ .

Note however that any two jobs  $J_k$  and  $J_\ell$  such that  $J_k$  is a *direct* predecessor of  $J_\ell$ , satisfy the following properties by construction: If  $t_k = \min \{t : x_{k,t} > 0\}$ , then

1.  $\max \{t : x_{k,t} > 0\} = t_k + d - 1$ .
2.  $\min \{t : x_{\ell,t} > 0\} = t_k + 1$ .
3.  $\max \{t : x_{\ell,t} > 0\} = t_k + d$ .

Thus Equation 6.4 holds since we have that for any such jobs  $J_k, J_\ell$ :

1. If  $t \in [t_k - 1]$ , then:

$$\underbrace{\sum_{\tilde{i}=1}^{t-1} x_{\ell,\tilde{i}}}_{=0} + \underbrace{\sum_{\tilde{i}=t+1}^T x_{k,\tilde{i}}}_{=1} = 1.$$

2. If  $t \in \{t_k, t_k + 1, \dots, t_k + d\}$ , then:

$$\begin{aligned} \sum_{\tilde{i}=1}^{t-1} x_{k,\tilde{i}} + \sum_{\tilde{i}=t+1}^{2d} x_{\ell,\tilde{i}} &= \sum_{\tilde{i}=t_k}^{t-1} x_{k,\tilde{i}} + \sum_{\tilde{i}=t+1}^{t_k+d} x_{\ell,\tilde{i}} \\ &= \frac{t-1-t_k+1}{d} + \frac{t_k+d-(t+1)+1}{d} \\ &= 1. \end{aligned}$$

3. If  $t \in \{t_k + d + 1, \dots, 2d\}$ , then:

$$\underbrace{\sum_{\tilde{i}=1}^{t-1} x_{\ell,\tilde{i}}}_{=1} + \underbrace{\sum_{\tilde{i}=t+1}^T x_{k,\tilde{i}}}_{=0} = 1.$$

This concludes the feasibility of an LP solution with makespan  $2d$ . It remains to show that any actual feasible schedule must have a large makespan. It is not hard to see that we should completely schedule all the jobs in Layer 1 so that we are able to start with the first job in Layer 2. Similarly, due to the *chain-like* structure of Layer 2, it requires  $d - 1$  times steps to be scheduled, before any job in Layer 3 can start executing. Hence the makespan of any feasible schedule is at least

$$\frac{dm - (d-1)}{m} + (d-1) + \frac{dm - (d-1)}{m} = 3d - \frac{2(d-1)}{m} - 1 \geq 3d - 2.$$

□

**Final Instance** We now construct our final integrality gap instance  $\mathcal{S}(k, d)$  using the basic building block  $\mathcal{S}(d)$ . This is basically done by *replicating* the structure of  $\mathcal{S}(d)$ , and arguing that any feasible schedule for  $\mathcal{S}(k, d)$  must have a makespan of roughly  $2kd$ , whereas we can *extend* the LP solution of Lemma 6.5 for the instance  $\mathcal{S}(d)$ , to a feasible LP solution for  $\mathcal{S}(k, d)$  of value  $(k + 1)d$ . A key point that we use here is that the structure of the LP solution of Lemma 6.5 enables us to schedule a fraction of the *chain* jobs of a layer, while executing the *non-chain* jobs of the previous layer. We now proceed to prove that the integrality gap of [SCHED-LP] is 2, by constructing a family  $\mathcal{S}(k, d)$  of scheduling instances, using the basic building block  $\mathcal{S}(d)$ .

**Theorem 6.6.** [SCHED-LP] has an integrality gap of  $c$ , where  $c$  is a constant arbitrarily close to 2.

*Proof.* Consider the following family of instances  $\mathcal{S}(k, d)$  for constant integers  $k$  and  $d$ , constructed as follows:

- We have  $k + 1$  layers  $\{\mathcal{L}_1^1, \mathcal{L}_2^1, \dots, \mathcal{L}_{k+1}^1\}$  similar to Layer 1 in  $\mathcal{S}(d)$ , and  $k$  layers  $\{\mathcal{L}_1^2, \mathcal{L}_2^2, \dots, \mathcal{L}_k^2\}$  similar to Layer 2, i.e.,
  - $\mathcal{L}_i^1$  has  $dm - (d - 1)$  jobs  $J_{i,1}^1, \dots, J_{i, dm - (d - 1)}^1$  for all  $i \in [k + 1]$ , and
  - $\mathcal{L}_i^2$  has  $(d - 1)$  jobs  $J_{i,1}^2, \dots, J_{i, (d - 1)}^2$  for all  $i \in [k]$ .
- For  $i \in [k]$ :
  - Connect  $\mathcal{L}_i^1$  to  $\mathcal{L}_i^2$  in the same way that Layer 1 is connected to Layer 2 in  $\mathcal{S}(d)$ , that is, the job  $J_{i,1}^2 \in \mathcal{L}_i^2$  is a successor for all the jobs in  $\mathcal{L}_i^1$ .
  - Connect  $\mathcal{L}_i^2$  to  $\mathcal{L}_{i+1}^1$  in the same way that Layer 2 is connected to Layer 3 in  $\mathcal{S}(d)$ , that is, all the jobs in  $\mathcal{L}_{i+1}^1$  are successors for the job  $J_{i, (d - 1)}^2 \in \mathcal{L}_i^2$ .

Notice that for  $k = 1$ , the scheduling instance  $\mathcal{S}(1, d)$  is the same as the previously defined instance  $\mathcal{S}(d)$ . The construction is depicted in Figure 6.2.

In any feasible schedule, we need to first schedule the jobs in  $\mathcal{L}_1^1$ , then those in  $\mathcal{L}_1^2$ , then  $\mathcal{L}_2^1$ , and so on, until  $\mathcal{L}_{k+1}^1$ . Hence the makespan of any such schedule is at least

$$(k + 1) \frac{dm - (d - 1)}{m} + k(d - 1) > 2kd + d - k - 1.$$

We now show that [SCHED-LP] has a feasible solution of value  $(k + 1)d$ . Let  $\{x_{j,t}\}$  for  $t = 1, \dots, 2d$  and  $j = 1, \dots, 2dm - (d - 1)$  be the feasible solution of value  $2d$  obtained in Lemma 6.5. It would be easier to think of  $\{x_{j,t}\}$  as  $\{x_{j,t}^1\} \cup \{x_{j,t}^2\} \cup \{x_{j,t}^3\}$  where for

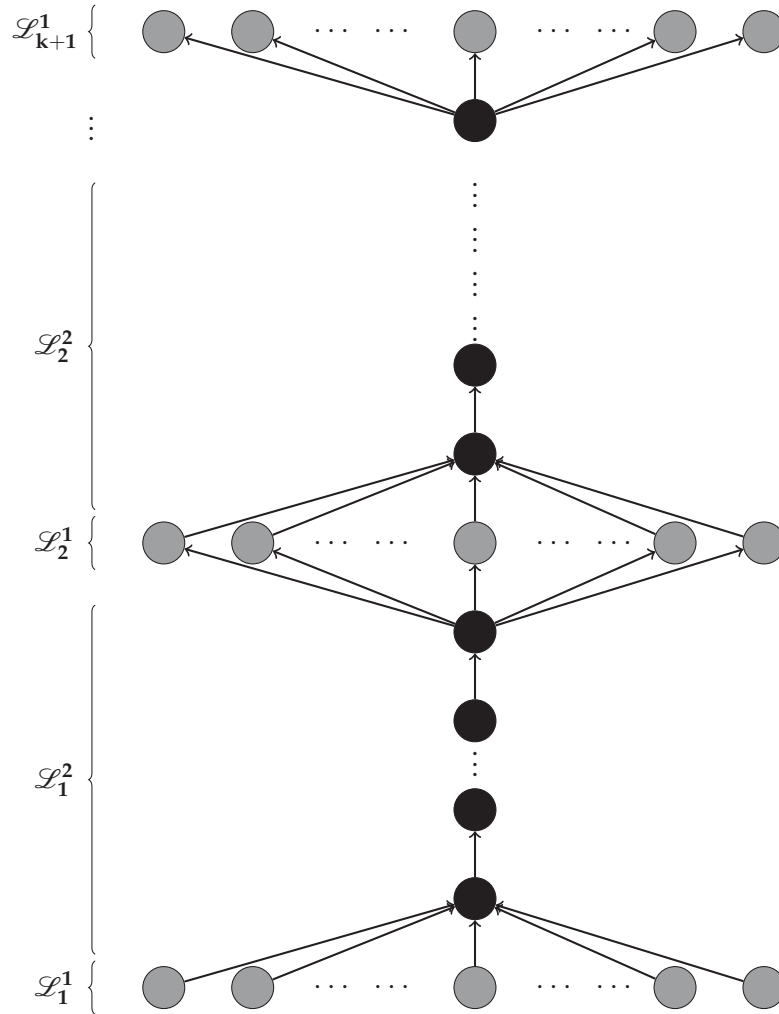


Figure 6.2 – Illustration of the integrality gap instance  $\mathcal{S}(k, d)$ .

$i = 1, 2, 3$ ,  $\{x_{j,t}^i\}$  is the set of LP variables corresponding to variables in Layer  $i$  in  $\mathcal{S}(d)$ . We now construct a feasible solution  $\{y_{j,t}\}$  for  $\mathcal{S}(k, d)$ . We similarly think of  $\{y_{j,t}\}$  as  $\{y_{j,t}^1\} \cup \{y_{j,t}^2\}$ , where  $y_{j,t}^i$  is the set of LP variables corresponding to jobs in  $\mathcal{L}_\ell^i$ , for some  $1 \leq \ell \leq k + 1$ . The set  $\{y_{j,t}^i\}$  for  $\mathcal{S}(k, d)$  can then be readily constructed as follows:

- for  $J_{1,j}^1 \in \mathcal{L}_1^1$ ,  $y_{j,t}^1 = x_{j,t}^1$  for  $t \leq 2d$ , and 0 otherwise.
- for  $J_{i,j}^2 \in \mathcal{L}_i^2$ ,  $y_{j,t+(i-1)d}^2 = x_{j,t}^2$  for  $i = 1, 2, \dots, k$ , and  $t \leq 2d$ , and 0 otherwise.
- for  $J_{i,j}^1 \in \mathcal{L}_i^1$ ,  $y_{j,t+(i-1)d}^1 = x_{j,t}^3$  for  $i = 2, 3, \dots, k + 1$ , and  $t \leq 2d$ , and 0 otherwise.

Using Lemma 6.5, we get that for  $t = d + 1 + i$ ,  $i \in [d - 1]$ , the machines in the feasible LP solution can still execute a load of  $\frac{i}{d}$ , and hence invoking the same analysis of Lemma 6.5

with the aforementioned observation for every two consecutive layers of jobs, we get that  $\{y_{j,t}\}$  is a feasible solution for [SCHED-LP] of value  $(k + 1)d$ .  $\square$

To recap, the results in this section can be summarized as follows: For a certain LP relaxation of the  $P|_{\text{prec, pmtn}}, p_j = 1|C_{\max}$  problem, namely [SCHED-LP], we know the following:

1. There exists a family of instances illustrated in Figure 6.1 for which [SCHED-LP] has an integrality gap of  $3/2$ .
2. If we appropriately replicate the structure in Figure 6.1 as shown in Figure 6.2, then we can show that the integrality gap of [SCHED-LP] is 2.

Motivated by the structure of Figure 6.1, we show in the next section that  $P|_{\text{prec, pmtn}}, p_j = 1|C_{\max}$  is in fact NP-hard to approximate within a factor of  $3/2$ , assuming a variant of the Unique Games Conjecture. In particular, that conjecture implies that given a bipartite graph  $G$ , it is NP-hard to distinguish between the case where  $G$  behaves like an *expander*, and the case where  $G$  behaves has a *nice ordered* structure. By carefully defining a reduction from bipartite graphs to  $P|_{\text{prec, pmtn}}, p_j = 1|C_{\max}$  scheduling instances in the same spirit as in Figure 6.1, we get the UG-hardness result of  $3/2$ .

## 6.4 Hardness of $P|_{\text{prec, pmtn}}|C_{\max}$ assuming the UGC

Assuming a variant of the Unique Games Conjecture, Bansal and Khot [8] proved the following structural hardness result for bipartite graphs:

**Theorem 6.7.** [Section 7.2 in [8]] *For every  $\varepsilon, \delta > 0$ , and positive integer  $Q$ , the following problem is NP-hard assuming a variant of the UNIQUE GAMES Conjecture: given an  $n$ -by- $n$  bipartite graph  $G = (V, W, E)$ , distinguish between the following two cases:*

- *YES Case:  $V$  can be partitioned into  $V_0, \dots, V_{Q-1}$  and  $W$  can be partitioned into  $W_0, \dots, W_{Q-1}$ , such that*
  - *There is no edge between  $V_i$  and  $W_j$  for all  $0 \leq j < i < Q$ .*
  - *$|V_i| \geq \frac{(1-\varepsilon)}{Q}n$  and  $|W_i| \geq \frac{(1-\varepsilon)}{Q}n$ , for all  $i \in [Q]$ .*
- *NO Case: For any  $S \subseteq V, T \subseteq W, |S| = \delta n, |T| = \delta n$ , there is an edge between  $S$  and  $T$ .*

The YES case of Theorem 6.7 is depicted in Figure 6.3.

The main result of this Section is that the scheduling problem  $P|_{\text{prec, pmtn}}|C_{\max}$  is NP-hard to approximate within a factor of  $3/2 - \varepsilon$ , assuming the variant of the Unique Games Conjecture. This is formally stated in Theorem 6.8.

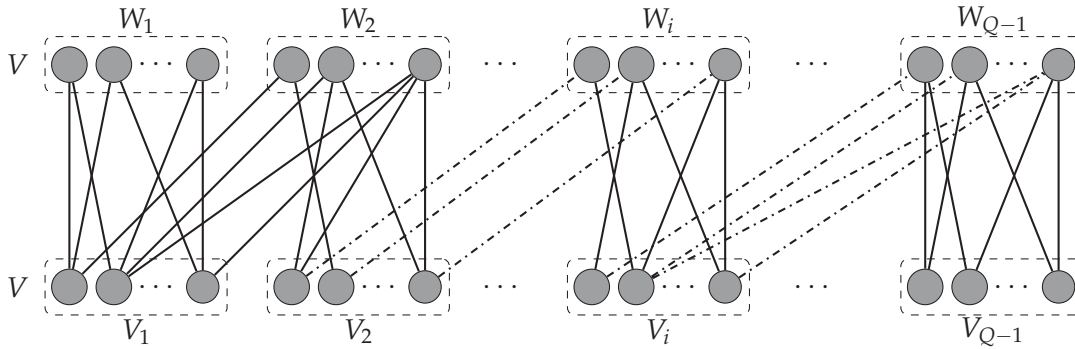


Figure 6.3 – Bipartite Graph  $G = (V, W, E)$  satisfying the YES case of Theorem 6.7. For each  $i = 0, \dots, Q - 1$ ,  $|V_i|, |W_i| \geq \frac{1-\epsilon}{Q}|V|$ . Moreover, the only allowed edges are from a vertex  $w \in W_i$  to vertices  $v \in V_j$ ,  $j \leq i$ . In other words, an edge  $(w, v)$  such that  $w \in W_i$ ,  $v \in V_j$ , and  $j > i$  is prohibited.

**Theorem 6.8.** For any  $\epsilon > 0$ , it is NP-hard to approximate  $P|prec, pmtn|C_{max}$  within a factor of  $3/2 - \epsilon$ , assuming (a variant of) the Unique Games Conjecture.

In order to prove Theorem 6.8, we will provide a reduction that, given a bipartite graph  $G = (V, W, E)$ , creates in polynomial time a scheduling instance  $\mathcal{F}$  of  $P|prec, pmtn, p_j = 1|C_{max}$  such that:

*Completeness:* If  $G$  satisfies the YES case of Theorem 6.7, then there exists a schedule for  $\mathcal{F}$  that has a makespan of roughly  $2Q$ .

*Soundness:* If  $G$  satisfies the NO case of Theorem 6.7, then any valid schedule for  $\mathcal{F}$  must have a makespan of at least roughly  $3Q$ .

Note that since  $P|prec, pmtn, p_j = 1|C_{max}$  is a special case of  $P|prec, pmtn|C_{max}$ , a lower bound on the former directly implies the same lower bound on the latter. Hence hereinafter, we restrict the discussion to the  $P|prec, pmtn, p_j = 1|C_{max}$  problem.

### Reduction

We present a reduction from an  $n$ -by- $n$  bipartite graph  $G = (V, W, E)$  to a scheduling instance  $\mathcal{F}$ . The reduction has a parameter  $Q$ , which corresponds to the constant in Theorem 6.7:

- For each vertex  $v \in V$ , we create a set  $\mathcal{F}_v$  of  $Qn$  jobs each of size 1, and let  $\mathcal{F}_V := \bigcup_{v \in V} \mathcal{F}_v$ .

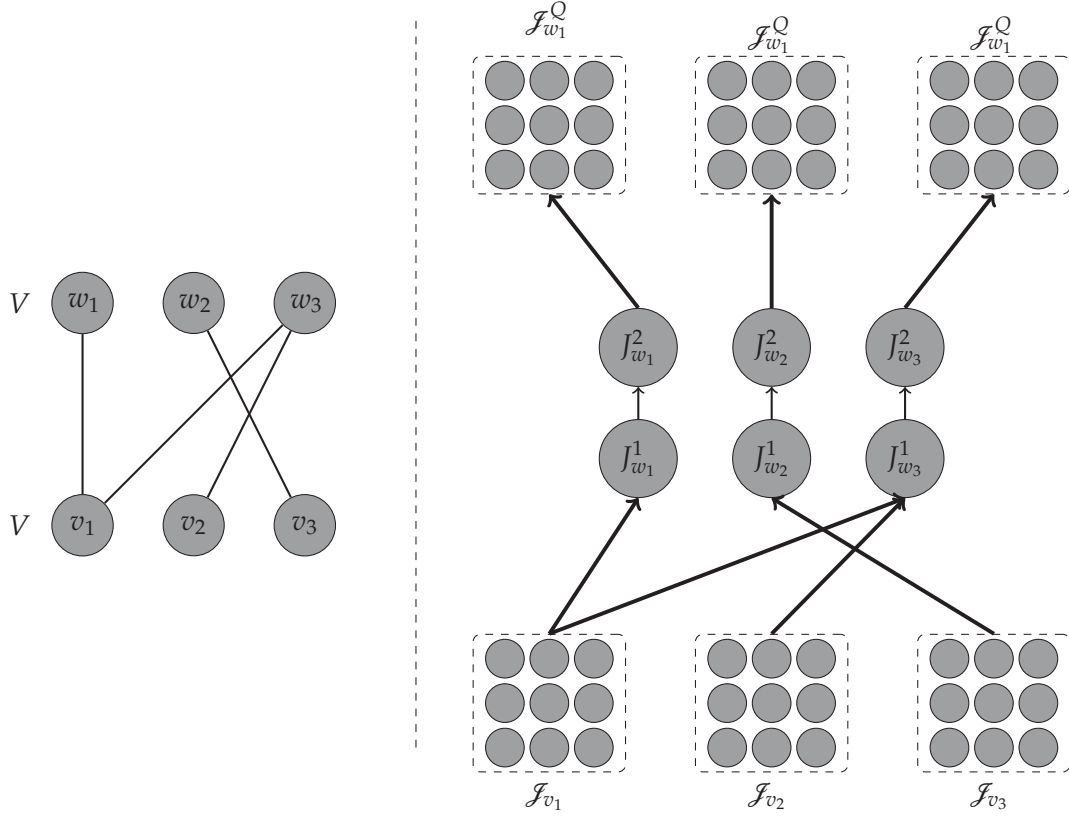


Figure 6.4 – Reduction from a bipartite graph  $G = (V, W, E)$  to a scheduling instance  $\mathcal{F}$ . On the left side, we have the starting  $3 \times 3$  bipartite graph  $G = (V, W, E)$ . On the right hand side, we have the resulting scheduling instance  $\mathcal{F}$  for  $Q = 3$ . A thick directed edge from (or to) a set of jobs means that all the jobs in this set has this precedence constraint.

- For each vertex  $w \in W$ , we create a set  $\mathcal{F}_w$  of  $Q(n + 1) - 1$  jobs

$$\mathcal{F}_w = \{J_w^1, J_w^2, \dots, J_w^{Q-1}\} \cup \mathcal{F}_w^Q,$$

where  $\mathcal{F}_w^Q$  is the set of the last  $Qn$  jobs, and the first  $Q - 1$  jobs are the *chain jobs*. We also define  $\mathcal{F}_W$  to be  $\mathcal{F}_W := \bigcup_{w \in W} \mathcal{F}_w^Q$ .

- For each edge  $e = (v, w) \in E$ , we have  $J_v \prec J_w^1$  for all  $J_v \in \mathcal{F}_v$ .
- For each  $w \in W$ , we have the following precedence constraints:

$$\begin{aligned} J_w^i &\prec J_w^{i+1}, & \forall i \in [Q - 2], \\ J_w^{Q-1} &\prec \mathcal{F}_w^Q. \end{aligned}$$

An illustration of this reduction is shown in Figure 6.4.

In total, the number of jobs and precedence constraints is polynomial in  $n$  since

$$\text{number of the jobs} \leq Qn^2 + n(Q(n+1) - 1) = 2Qn^2 + Qn - n.$$

For a subset  $\mathcal{S}$  of jobs in our scheduling instance  $\mathcal{J}$ , we denote by  $\Psi(\mathcal{S}) \subseteq V \cup W$  the set of their *representative* vertices in the starting graph  $G$ . Similarly, for a subset  $S \subseteq V \cup W$ ,  $\Psi^{-1}(S) \subseteq \mathcal{J}_V \cup \mathcal{J}_W$  is the set of *all* jobs, except for chain jobs, corresponding to vertices in  $S$ , i.e.,

$$\Psi^{-1}(S) = \left( \bigcup_{v \in (S \setminus W)} \mathcal{J}_v \right) \cup \left( \bigcup_{w \in (S \setminus V)} \mathcal{J}_w^Q \right).$$

A subset  $\mathcal{S}$  of jobs with  $S = \Psi(\mathcal{S})$  is said to be *complete* if  $\mathcal{S} = \Psi^{-1}(S)$ .

W.l.o.g. assume that  $Q$  divides  $n$ . Finally the number of machines is  $n^2$ . Before proceeding with the proof of Theorem 6.8, we record the following easy observations:

*Observation 6.9.* If for some  $w \in W$ , there exist a feasible schedule  $\sigma$  in which a job  $J \in \mathcal{J}_w^Q$  starts before time  $T$ , then the set  $\mathcal{A} \subseteq \mathcal{J}_V$  of all its predecessors in  $\mathcal{J}_V$  must have finished executing in  $\sigma$  prior to time  $T - Q$ . Moreover  $\mathcal{A}$  is complete, i.e.,  $\mathcal{A} = \Psi^{-1}(w)$ .

*Observation 6.10.* For any subset  $\mathcal{A} \subseteq \mathcal{J}_V \cup \mathcal{J}_W$ , we have that

$$|\Psi(\mathcal{A})| \geq \frac{|\mathcal{A}|}{nQ},$$

where the bound is met with equality if  $\mathcal{A}$  is complete.

### Completeness

**Lemma 6.11.** *Let  $G = (V, W, E)$  be an  $n$ -by- $n$  bipartite graph, and let  $\mathcal{J}$  be the  $P|prec, pmtn, p_j = 1|C_{max}$  scheduling instance resulting from the reduction. If  $G$  satisfies the YES case of Theorem 6.7 for some real  $\varepsilon > 0$  and integer  $Q$ , then there exists a feasible schedule for  $\mathcal{J}$  of makespan  $2Q + \varepsilon'$ , for some  $\varepsilon' = \varepsilon'(\varepsilon, Q)$ .*

*Proof.* Let  $V_0, V_1, \dots, V_{Q-1}, W_0, W_1, \dots, W_{Q-1}$  be the partitions as in the YES Case of Theorem 6.7. Note that this implies that for all  $i \in [Q]$ , any vertex  $w \in W_i$  is only connected to vertices in  $V_j$  where  $j \leq i$ . We have also have that

$$|W_i|, |V_i| \geq \frac{1 - \varepsilon}{Q} n \quad \forall i \in [Q],$$

which in turn implies that

$$|W_i|, |V_i| \leq \left( \frac{1 - \varepsilon}{Q} + \varepsilon \right) n \quad \forall i \in [Q]. \quad (6.5)$$



For a subset  $S \subseteq V \cup W$ , we denote by  $\mathcal{J}_S$  the set of jobs corresponding to vertices in  $S$ , i.e.,  $\mathcal{J}_S = \cup_{u \in S} \mathcal{J}_u$ . Also, for an index  $i \in [2Q]$ , we define a job set  $\mathcal{T}_i$  as follows:

$$\mathcal{T}_i = \begin{cases} \mathcal{S}_i \cup \mathcal{J}_{V_{i-1}} & 0 < i < Q \\ \mathcal{S}_i \cup \mathcal{J}_{W_{i-Q-1}}^Q & Q \leq i < 2Q \end{cases}$$

where

$$\mathcal{S}_i = \{J_{W_{k-1}}^\ell : 1 \leq \ell < Q, k \in [Q], \text{ and } k + \ell = i\}.$$

To see the intuition behind this partitioning, note that we get in this case for instance that

$$\mathcal{T}_1 = \mathcal{J}_{V_0} \quad \mathcal{T}_2 = \{\mathcal{J}_{V_1}, J_{W_0}^1\} \quad \mathcal{T}_3 = \{J_{V_2}, J_{W_0}^2, J_{W_1}^1\}.$$

as shown in Figure 6.5.

Given the structure of the graph  $G$  and the properties of its partitions, we get that any predecessor for a job in  $\mathcal{T}_2$  must be in  $\mathcal{T}_1$ , and any predecessor of a job in  $\mathcal{T}_3$  must be in either  $\mathcal{T}_2$  or  $\mathcal{T}_1$ . In general, we get by construction that if there exists two jobs  $J, J'$  such that  $J' \prec J$  and  $J \in \mathcal{J}_{W_k}^\ell$ , then  $J'$  can only be in one of the following two sets:

$$J' \in \mathcal{J}_{V_{k'}} \text{ s.t. } k' \leq k, \quad \text{or,} \quad J' \in \mathcal{J}_{W_{k'}}^{\ell'} \text{ s.t. } k' = k, \ell' < \ell.$$

This then implies that a schedule  $\sigma$  in which we first schedule  $\mathcal{T}_1$  then  $\mathcal{T}_2$ , and so on up to  $\mathcal{T}_{2Q}$  is indeed a valid schedule. Now using equation (6.5) and the construction of our scheduling instance  $\mathcal{J}$ , we get that

$$|\mathcal{T}_i| \leq Qn^2 \left( \frac{1-\varepsilon}{Q} + \varepsilon \right) + nQ \left( \frac{1-\varepsilon}{Q} + \varepsilon \right) \leq n(n+1)(1+\varepsilon Q).$$

Hence the total makespan of  $\sigma$  is at most

$$\sum_{i=1}^{2Q} \frac{|\mathcal{T}_i|}{n^2} = 2Q \left( 1 + \varepsilon Q + O\left(\frac{1}{n}\right) \right),$$

which tends to  $2Q + \varepsilon'$  for large values of  $n$ , because  $Q$  is a constant.  $\square$

### Soundness

**Lemma 6.12.** *Let  $G = (V, W, E)$  be an  $n$ -by- $n$  bipartite graph, and let  $\mathcal{J}$  be the  $P|_{\text{prec, pmtn}}, p_j = 1|C_{\max}$  scheduling instance resulting from the reduction. If  $G$  satisfies the NO case of Theorem 6.7 for some real  $\delta > 0$ , then any feasible schedule for  $\mathcal{J}$  must have a makespan of at least  $3Q - \varepsilon'$ , for some  $\varepsilon' = \varepsilon'(\delta, Q)$ .*

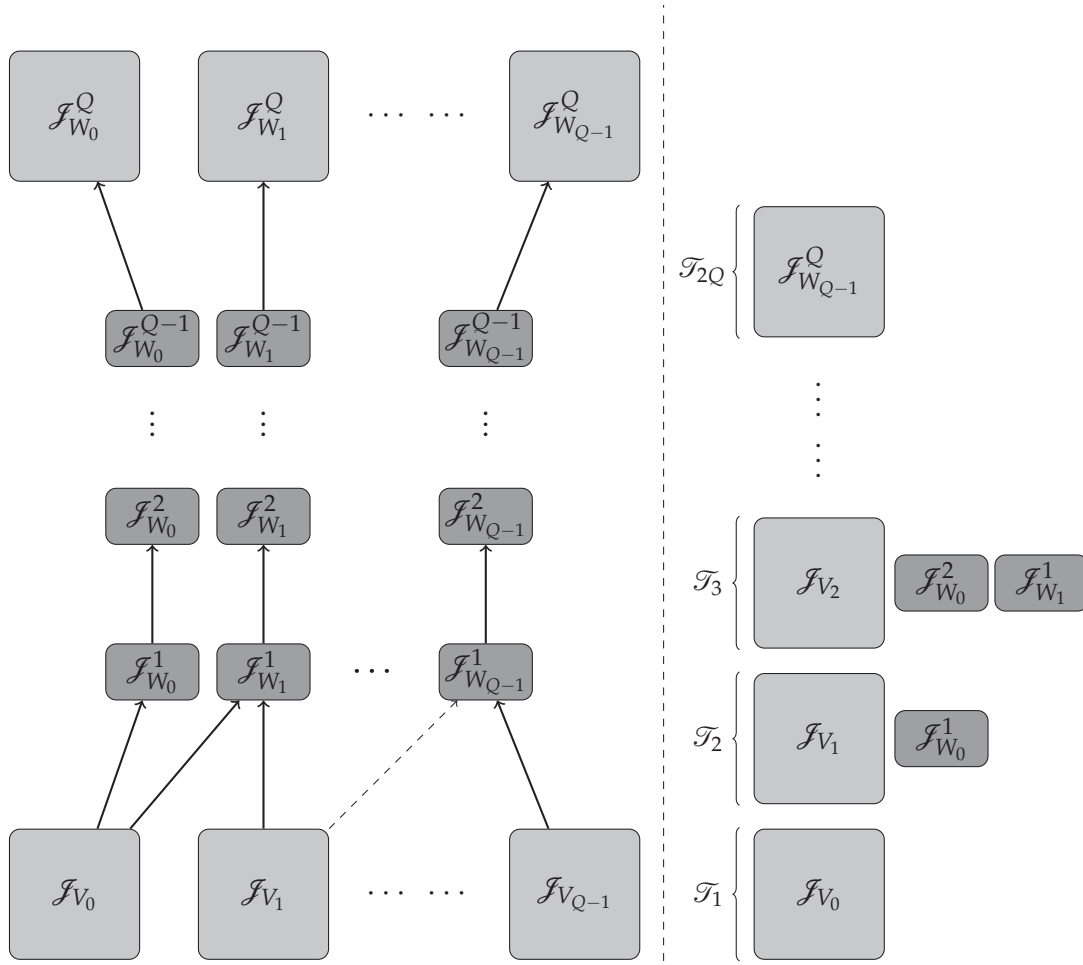


Figure 6.5 – The structure of the scheduling instance in the YES case of Theorem 6.7. On the left, we have the directed graph depicting the precedence constraints of a scheduling instance with sets  $\mathcal{J}_{(\cdot)}^{(\cdot)}$  as defined in the proof of Lemma 6.11. On the right, we illustrate the corresponding sets  $\mathcal{T}_i$  for  $i = 1, \dots, 2Q$  defined in the proof of Lemma 6.11.

*Proof.* Assume towards contradiction that there exists a schedule for  $\mathcal{J}$  with a maximum makespan less than  $t := 3Q - 1 - 2\delta Q$ , and let  $\mathcal{A}$  be the set of jobs in  $\mathcal{J}_W$  that started executing by or before time  $s := 2Q - 1 - \delta Q$ , and denote by  $\mathcal{B}$  the set of their predecessors in  $\mathcal{J}_V$ . Note that  $\mathcal{B}$  is complete by Observation 6.9. Now since  $t - s = Q - \delta Q$ , we get that  $|\mathcal{A}| \geq \delta Q n^2$ , and hence, by Observation 6.10,  $|\mathcal{A}| := |\Psi(\mathcal{A})| \geq \frac{\varepsilon Q n^2}{Q n} = \delta n$ . Applying Observation 6.9 one more time, we get that all the jobs in  $\mathcal{B}$  must have finished executing in  $\sigma$  by time  $Q - \delta Q$ , and hence  $|\mathcal{B}| \leq Q n^2 (1 - \delta)$ . Using the fact that  $\mathcal{B}$  is complete, we get that  $|\mathcal{B}| := |\Psi(\mathcal{B})| \leq \frac{Q n^2 (1 - \delta)}{Q n} = (1 - \delta)n$ , which contradicts with the NO Case of Theorem 6.7.  $\square$

Theorem 6.8 now follows by combining Lemmata 6.11 and 6.12.

We note here that we can settle for a weaker structure of the graph corresponding to the completeness case of Theorem 6.7. In particular, we can use a graph resulting from Theorem 2 in [101], and yet get a hardness of  $3/2 - \epsilon$ . Using the same reduction used here, this will yield this somehow stronger statement:

**Theorem 6.13.** *For any  $\epsilon > 0$ , and  $\eta \geq \eta(\epsilon)$ , where  $\eta(\epsilon)$  tends to 0 as  $\epsilon$  tends to 0, if  $1|\text{prec}|\sum_j w_j C_j$  has no  $(2 - \epsilon)$ -approximation algorithm, then  $P|\text{prec}, \text{pmtn}|C_{\max}$  has no  $(3/2 - \eta)$ -approximation algorithm.*

## 6.5 Structured $k$ -Partite Problem

Motivated by the resemblance between the basic building block of the integrality gap in Section 6.3, and the hard structure of the bipartite graph yielding the inapproximability of  $3/2$  in Section 6.4, we propose in this section a natural but nontrivial generalization of Theorem 6.7 to  $k$ -partite graphs. Assuming hardness of this problem, we can get the following hardness of approximation results:

1. It is NP-hard to approximate  $P|\text{prec}, \text{pmtn}|C_{\max}$  within a  $2 - \epsilon$  factor.
2. It is NP-hard to approximate  $Q|\text{prec}|C_{\max}$  within any constant factor.
3. It is NP-hard to approximate  $1|\text{prec}|\sum_j w_j C_j$  within a  $2 - \epsilon$  factor.
4. It is NP-hard to approximate  $P|\text{prec}|C_{\max}$  within a  $2 - \epsilon$  factor.

The first and second result are presented in Sections 6.6.1 and 6.6.2, respectively. Moreover, one can see that the reduction presented in [8] for the scheduling problem  $1|\text{prec}|\sum_j w_j C_j$  holds using the hypothesis for the case that  $k = 2$ . The same holds for the reduction in [101] for the scheduling problem  $P|\text{prec}|C_{\max}$ . This suggests that this structured hardness result for  $k$ -partite graphs somehow unifies a large family of scheduling problems, and captures their common intrinsic hard structure.

Recall that a graph  $G = (V, E)$  is said to be  $k$ -partite for an integer  $k \geq 2$ , if the vertex set  $V$  can be partitioned into  $k$  disjoint sets of  $V_1, \dots, V_k \subset V$ , and the edge set  $E$  can be partitioned into  $k - 1$  disjoint sets  $E_1, \dots, E_{k-1} \subseteq E$  such that:

1.  $V = V_1 \sqcup \dots \sqcup V_k$  and  $E = E_1 \sqcup \dots \sqcup E_{k-1}$ , and
2.  $E_i \subseteq (V_i \times V_{i+1})$  for  $i = 1, \dots, k - 1$ .

In other words, if we have an edge  $(v_\ell, v_j) \in E$  with  $v_\ell \in V_i$  for some  $i = 2, \dots, k - 1$ , then  $v_j$  can either be in  $V_{i-1}$  or in  $V_{i+1}$ . If we have that  $v_\ell \in V_1$ , then  $v_j$  must be in  $V_2$ . Similarly, if  $v_\ell \in V_k$ , then  $v_j$  must be in  $V_{k-1}$ . For our purposes, we will be interested in  $k$ -partite graphs where  $|V_1| = \dots = |V_k| = n$ , i.e.,  $|V| = kn$ .

**Hypothesis 6.14.** [*k*-Partite Problem] For every  $\epsilon, \delta > 0$ , and constant integers  $k, Q > 1$ , the following problem is NP-hard: given a *k*-partite graph  $G = (V_1, \dots, V_k, E_1, \dots, E_{k-1})$  with  $|V_i| = n$  for all  $1 \leq i \leq k$  and  $E_i$  being the set of edges between  $V_i$  and  $V_{i+1}$  for all  $1 \leq i < k$ , distinguish between following two cases:

- YES Case: every  $V_i$  can be partitioned into  $V_{i,0}, \dots, V_{i,Q-1}$ , such that
  - There is no edge between  $V_{i,j_1}$  and  $V_{i-1,j_2}$  for all  $1 < i \leq k, j_1 < j_2 \in [Q]$ .
  - $|V_{i,j}| \geq \frac{(1-\epsilon)}{Q}n$ , for all  $1 \leq i \leq k, j \in [Q]$ .
- NO Case: For any  $1 < i \leq k$  and any two sets  $S \subseteq V_{i-1}, T \subseteq V_i, |S| = \delta n, |T| = \delta n$ , there is an edge between  $S$  and  $T$ .

This says that if the *k*-partite graph  $G = (V_1, \dots, V_k, E_1, \dots, E_{k-1})$  satisfies the YES Case, then for every  $1 \leq i \leq k - 1$ , the induced subgraph  $\tilde{G} = (V_i, V_{i+1}, E_i)$  behaves like the YES Case of Theorem 6.7, and otherwise, every such induced subgraph corresponds to the NO case.

Recall that in order to prove an integrality gap of 2 in Section 6.3 for the LP corresponding to the  $P|_{\text{prec}}, \text{pmtn}, p_j = 1|_{C_{\max}}$  problem, we started from basic building block that had an integrality gap of  $3/2$ , and carefully replicated its structure to get the desired gap 2. This is exactly where Hypothesis 6.14 comes into play; its structure can be seen a generalization of Theorem 6.7 from bipartite graphs to *k*-partite graphs, and a carefully tailored replication of the reduction in Section 6.4 should allow us to amplify the hardness result in Theorem 6.8 to 2, instead of  $3/2$ . Moreover, this *k*-partite structure turns out be manifold as we show in Section 6.6.2; it yields a super constant inapproximability for the  $Q|_{\text{prec}}|_{C_{\max}}$  problem.

## 6.6 Lower Bounds for Scheduling Problems

In this section, we show that, assuming Hypothesis 6.14, there is no *c*-approximation algorithm for the scheduling problem  $P|_{\text{prec}}, \text{pmtn}|_{C_{\max}}$ , for any constant *c* strictly better than 2, and there is no constant factor approximation algorithm for the scheduling problem  $Q|_{\text{prec}}|_{C_{\max}}$ .

### 6.6.1 $P|_{\text{prec}}, \text{pmtn}|_{C_{\max}}$

We present in this section a reduction from a *k*-partite graph to an instance of the scheduling problem  $P|_{\text{prec}}, \text{pmtn}, p_j = 1|_{C_{\max}}$ , and we prove a tight inapproximability result for its generalization  $P|_{\text{prec}}, \text{pmtn}|_{C_{\max}}$ , assuming Hypothesis 6.14. Formally, we prove the following result:

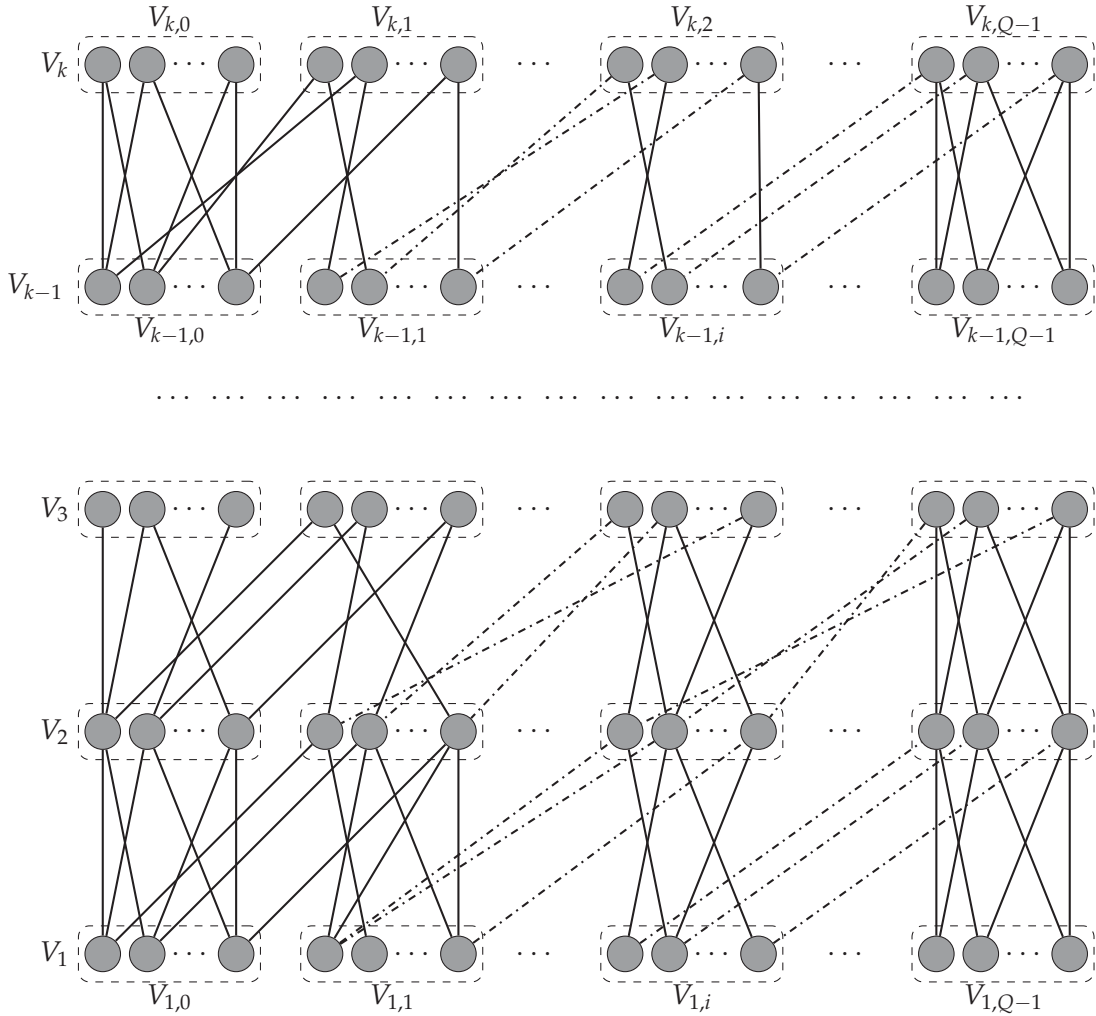


Figure 6.6 –  $k$ -partite Graph  $G = (V_1, V_2, \dots, V_k, E_1, E_2, \dots, E_{k-1})$  satisfying the YES case of Hypothesis 6.14. For each  $i = 0, \dots, Q-1, j = 1, \dots, k, |V_{i,j}| \geq \frac{1-\epsilon}{Q} |V_i|$ . Moreover, if we assume that the paths are oriented upwards, then a path leading to a vertex  $v \in V_{i,j}$  can only contain vertices from  $V_{i',j'}$  such that  $i' < i, j' \leq j$ .

**Theorem 6.15.** Assuming Hypothesis 6.14, it is NP-hard to approximate the scheduling problem  $P|prec, pmtn|C_{max}$  within any constant factor strictly better than 2.

To prove this, we first reduce a  $k$ -partite graph  $G = (V_1, \dots, V_k, E_1, \dots, E_{k-1})$  to a scheduling instance  $\mathcal{S}(k)$ , and then show that

1. If  $G$  satisfies the YES Case of Hypothesis 6.14, then  $\mathcal{S}(k)$  has a feasible schedule whose makespan is roughly  $kQ/2$ .
2. if  $G$  satisfies the NO Case of Hypothesis 6.14, then any schedule for  $\mathcal{S}(k)$  must

have a makespan of roughly  $kQ$ .

### Reduction

The reduction has three parameters: an odd integer  $k$ , an integer  $Q$  such that  $Q \gg k$  and  $n$  divides  $Q$ , and a real  $\epsilon \gg 1/Q^2 > 0$ .

Given a  $k$ -partite graph  $G = (V_1, \dots, V_k, E_1, \dots, E_{k-1})$ , we construct an instance  $\mathcal{F}(k)$  of the scheduling problem  $P|\text{prec, pmtn, } p_j = 1|C_{\max}$  as follows:

- For each vertex  $v \in V_{2i-1}$  and every  $1 \leq i \leq (k+1)/2$ , we create a set  $\mathcal{F}_{2i-1,v}$  of  $Qn - (Q-1)$  jobs.
- For each vertex  $v \in V_{2i}$  and every  $1 \leq i < (k+1)/2$ , we create a chain of length  $Q-1$  of jobs, i.e., a set  $\mathcal{F}_{2i,v}$  of  $Q-1$  jobs

$$\mathcal{F}_{2i,v} = \{J_{2i,v}^1, J_{2i,v}^2, \dots, J_{2i,v}^{Q-1}\},$$

where we have  $J_{2i,v}^l \prec J_{2i,v}^{l+1}$  for all  $l \in [Q-2]$ .

- For each edge  $e = (v, w) \in E_{2i-1}$  and every  $1 \leq i < (k+1)/2$ , we have  $\mathcal{F}_{2i-1,v} \prec J_{2i,w}^1$ .
- For each edge  $e = (v, w) \in E_{2i}$  and every  $1 \leq i < (k+1)/2$ , we have  $J_{2i,v}^{Q-1} \prec \mathcal{F}_{2i+1,w}$ .

Finally the number of machines is  $(1 + Q\epsilon)n^2$ . Note that this reduction can be thought of as replicating the reduction of Section 6.4 depicted in Figure 6.4. We illustrate this reduction on an example in Figure 6.7.

Theorem 6.15 now follows from the following lemma, that in turn follows from combining Lemmata 6.17 and 6.18.

**Lemma 6.16.** *Scheduling instance  $\mathcal{F}(k)$  has the following two properties.*

1. *If  $G$  satisfies the YES Case of Hypothesis 6.14, then  $\mathcal{F}(k)$  has a feasible schedule whose makespan is  $(1 + \epsilon)kQ/2$ , where  $\epsilon$  can be arbitrary close to zero.*
2. *if  $G$  satisfies the NO Case of Hypothesis 6.14, then any feasible schedule for  $\mathcal{F}(k)$  must have a makespan of  $(1 - \epsilon)kQ$ , where  $\epsilon$  can be arbitrary close to zero.*

**Lemma 6.17 (Completeness).** *If the given  $k$ -partite graph  $G$  satisfies the properties of the YES case of Hypothesis 6.14, then there exists a valid schedule for  $\mathcal{F}(k)$  with maximum makespan  $(1 + \epsilon')kQ/2$ , where  $\epsilon'$  can be made arbitrary close to zero.*

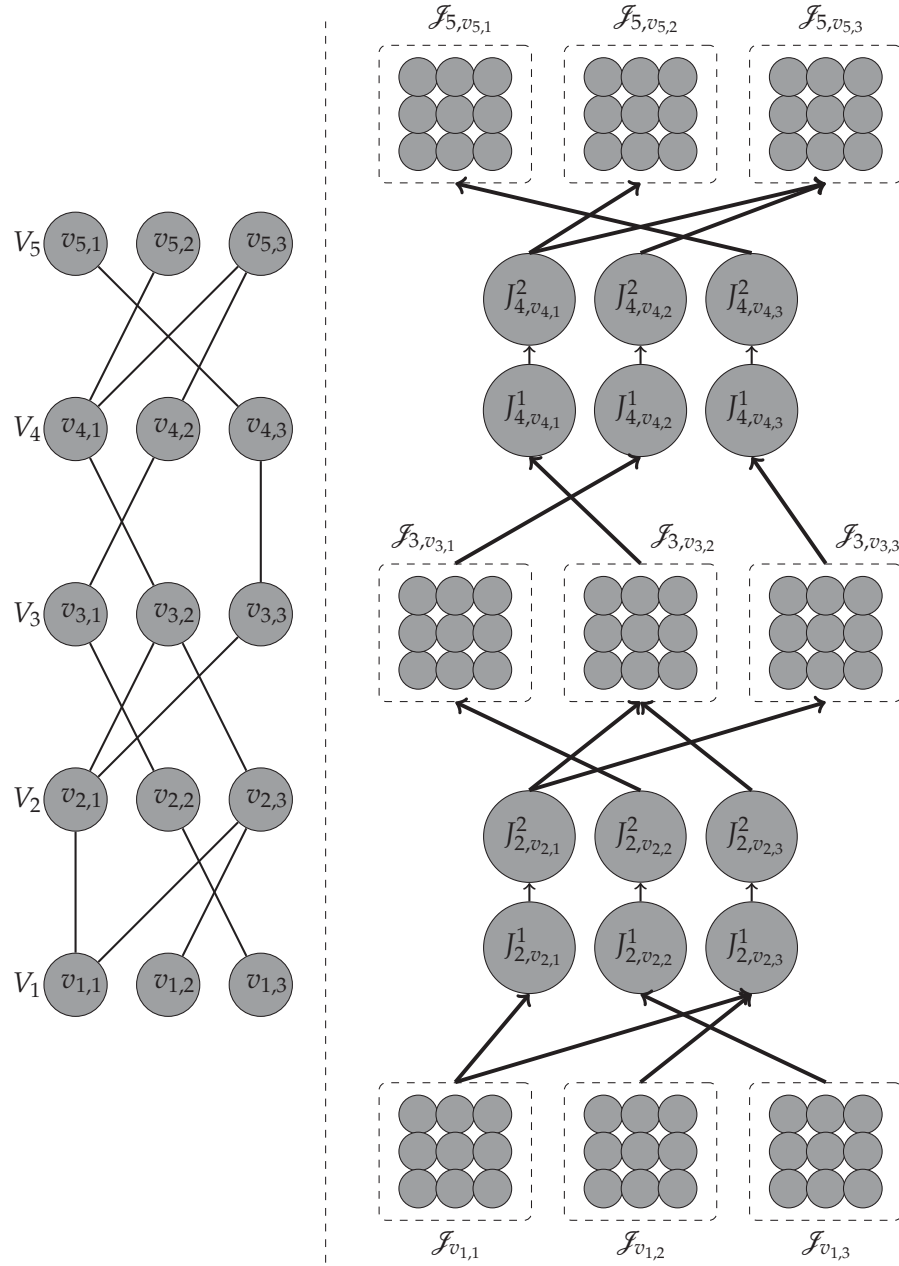


Figure 6.7 – Reduction from a 5-partite graph  $G = (V_1, V_2, V_3, V_4, V_5, E_1, E_2, E_3, E_4)$  to a scheduling instance  $\mathcal{F}$ . On the left side, we have the starting 5-partite graph  $G$ . On the right hand side, we have the resulting scheduling instance  $\mathcal{F}$  for  $Q = 3$ . A thick directed edge from (or to) a set of jobs means that all the jobs in this set has this precedence constraint.

*Proof.* Assume that  $G$  satisfies the properties of the YES Case of Hypothesis 6.14, and let  $\{V_{s,\ell}\}$  for  $s \in [k], \ell \in [Q]$  denote the good partitioning of the vertices of  $G$ . We use this partitioning to derive a partitioning  $\{\mathcal{S}_{i,j}\}$  for the jobs in the scheduling instance  $\mathcal{F}(k)$  for  $1 \leq i \leq (k-1)Q/2 - 1, j \in [Q]$ , where a set of jobs  $\mathcal{S}_{i,j}$  can be either *big* or *small*.

## Chapter 6. Scheduling Problems

The intuition behind this big/small distinction is that a job  $J$  is in a big set if it is part of the  $Qn - (Q - 1)$  copies of a vertex  $v \in V_{2i-1}$  for  $1 \leq i \leq (k + 1)/2$ , and in a small set otherwise.

These sets can now be formally defined as follows:

$$\begin{aligned} \text{Big sets: } \mathcal{S}_{Q(i-1)+1,j} &:= \bigcup_{v \in V_{2i-1,j}} \mathcal{J}_{2i-1,v} \quad \forall 1 \leq i \leq \frac{k+1}{2}, j \in [Q] \\ \text{Small sets: } \mathcal{S}_{Q(i-1)+1+l,j} &:= \bigcup_{v \in V_{2i,j}} J_{2i,v}^l \quad \forall 1 \leq i < \frac{k+1}{2}, j \in [Q], l \in [Q-1]. \end{aligned}$$

We first provide a brief overview of the schedule before defining it formally. Since  $\mathcal{S}_{1,0}$  is the set of the jobs corresponding to the vertices in  $V_{1,0}$ , scheduling all the jobs in  $\mathcal{S}_{1,0}$  in the first time step enables us to start the jobs at the first layer of the chain corresponding to vertices in  $V_{2,0}$  (i.e.,  $\mathcal{S}_{2,0}$ ). Therefore in the next time step we can schedule the jobs corresponding to the vertices in  $V_{1,1}$ , (i.e.  $\mathcal{S}_{1,1}$ ) and  $\mathcal{S}_{2,0}$ . This further enables us to continue to schedule the jobs in the second layer of the chain corresponding to the vertices in  $V_{2,0}$  (i.e.,  $\mathcal{S}_{3,0}$ ), the jobs at the first layer of the chain corresponding to vertices in  $V_{2,1}$  (i.e.,  $\mathcal{S}_{2,1}$ ), and the jobs corresponding to the vertices in  $V_{1,2}$  (i.e.,  $\mathcal{S}_{1,2}$ ). We can keep going the same way, until we have scheduled all the jobs. Since the number of partitions of each vertex set  $V_i$  is  $Q$ , and length of each of our chains is  $Q - 1$ , we can see that in the suggested schedule, we are scheduling in each time step at most  $Q$  sets, out of which exactly one is big, and none of the precedence constraints are violated. An example of a scheduling instance of this case is depicted in Figure 6.8.

Formally speaking, let  $\mathcal{T}_t$  be the union of  $\mathcal{S}_{i,j}$  such that  $t = i + j - 1$ , where  $1 \leq i \leq (k - 1)Q/2 + 1$  and  $j \in [Q]$ , hence each  $\mathcal{T}_t$  consist of at most  $Q$  sets of the jobs in which exactly one of them is a big set and at most  $Q - 1$  of them are small sets. Therefore, for  $t \in \lceil (k + 1)Q/2 \rceil$ , we have

$$\begin{aligned} |\mathcal{T}_t| &\leq |V_{2i-1,j}| \cdot (Qn - (Q - 1)) + |V_{2i,j}| \cdot (Q - 1) \\ &\leq (1/Q + \epsilon)n \cdot (Qn - (Q - 1)) + (1/Q + \epsilon)n \cdot (Q - 1) \\ &\leq (1/Q + \epsilon)n \cdot (Qn) \leq (1 + Q\epsilon)n^2. \end{aligned}$$

One can easily see that all the jobs in a set  $\mathcal{T}_t$  can be scheduled in a single time step since the number of machines is  $(1 + Q\epsilon)n^2$ . Hence consider the following schedule: for each  $t \in \lceil (k + 1)Q/2 \rceil$ , schedule all the jobs in  $\mathcal{T}_t$  between time  $t$  and  $t + 1$ . We claim that this schedule does not violate any precedence constraint. This is true because we first schedule the predecessors of the job, and then the job in the following steps. Formally, if  $J_1 \prec J_2$  with  $J_1 \in \mathcal{T}_{t_1}$  and  $J_2 \in \mathcal{T}_{t_2}$ , then  $t_1 < t_2$ .  $\square$

**Lemma 6.18 (Soundness).** *If the given  $k$ -partite graph  $G$  satisfies the properties of the No Case of Hypothesis 6.14, then any feasible schedule for  $\mathcal{F}(k)$  has a maximum makespan of at*



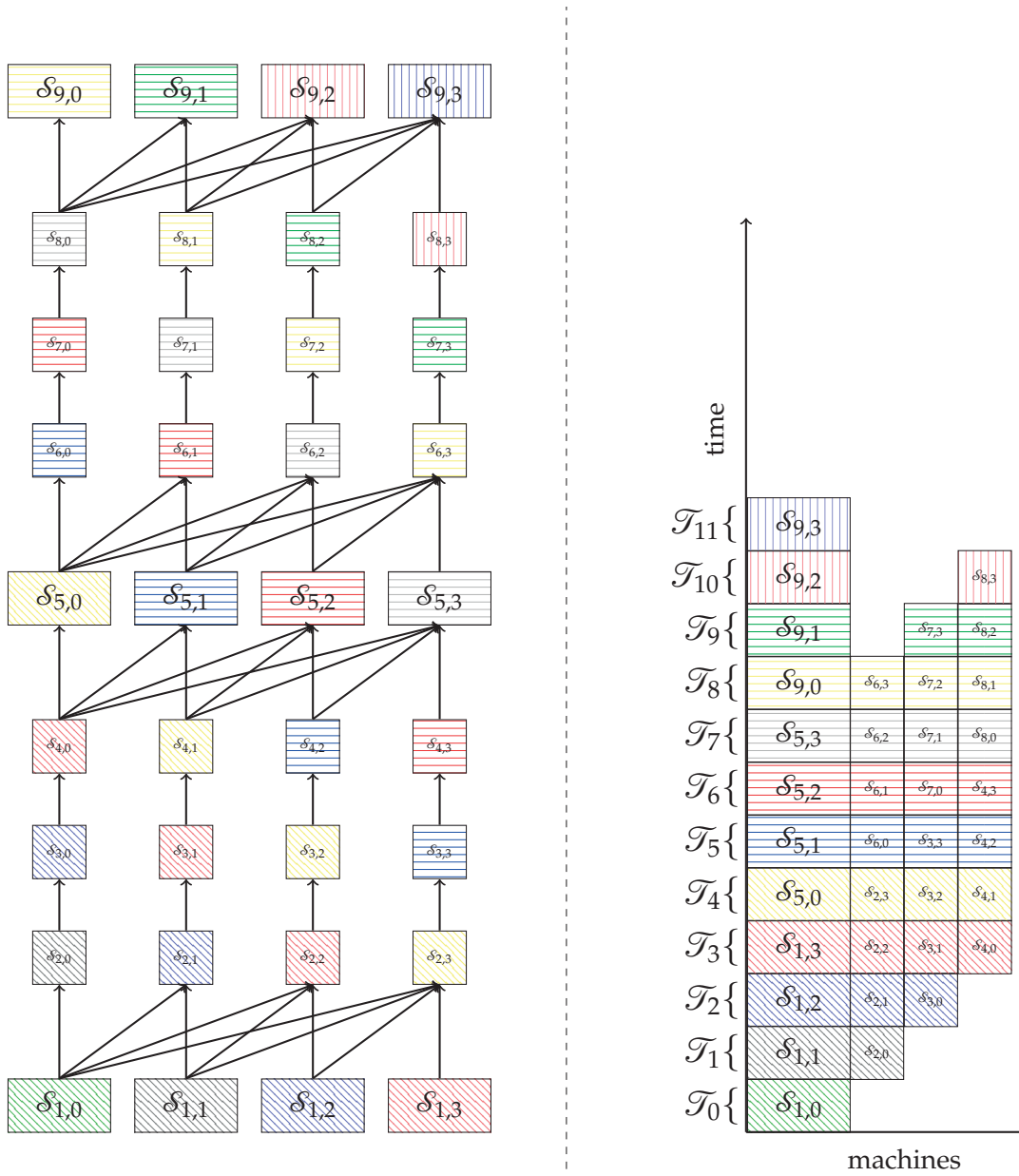


Figure 6.8 – An example of a scheduling instance corresponding to the YES case of Hypothesis 6.14. The figure on the left corresponds to scheduling instance corresponding to the YES case of Hypothesis 6.14 for  $Q = 4$ . Big rectangles correspond to big sets  $S_{i,j}$ , whereas small rectangle corresponds to small sets. Note that there are no precedence constraints between the sets of jobs who has the same fill pattern. The figure on the right depicts the corresponding scheduling. On time  $t \in [12]$ , we are scheduling the jobs in set  $\mathcal{T}_t$ , i.e., the sets of jobs  $S_{i,j}$  such that  $i + j - 1 = t$ .

least  $(1 - \epsilon')kQ$ , where  $\epsilon'$  can be made arbitrary close to zero.

## Chapter 6. Scheduling Problems

---

*Proof.* Assume that  $G$  satisfies the NO Case of Hypothesis 6.14, and consider the following partitioning of the jobs:

$$\begin{aligned} \text{Big partitions: } \mathcal{S}_{Q(i-1)+1} &:= \cup_{v \in V_{2i-1}} \mathcal{J}_{2i-1,v} & \forall 1 \leq i \leq (k+1)/2, \\ \text{Small partitions: } \mathcal{S}_{Q(i-1)+1+l} &:= \cup_{v \in V_{2i}} \mathcal{J}_{2i,v}^l & \forall 1 \leq i < (k+1)/2, l \in [Q-1]. \end{aligned}$$

Note that  $\{\mathcal{S}\}$  partitions the jobs into  $(k-1)Q/2 + 1$  partitions such that the size of a big partition is  $n(nQ - c) \geq n(n-1)Q$  and the size of a small partition is  $n$ . Let  $f_i$  be the first time that a  $(1-\delta)$  fraction of the jobs in  $\mathcal{S}_i$  is completely executed, and let  $s_i$  be the first time that more than  $\delta$  fraction of the jobs in  $\mathcal{S}_i$  is started. Because of the expansion property of the NO Case, we can not start more than  $\delta$  fraction of the jobs in the second partition, before finishing at least  $1-\delta$  fraction of the jobs in the first partition. This implies that  $f_1 \leq s_2$ . Similarly,  $f_1 + 1 \leq s_3$  and  $f_1 + Q - 2 \leq s_Q$ . The same inequalities hold for any big partition and the small partitions following it. This means that, beside  $\delta$  fraction of the jobs in the  $i$ -th and  $(i+1)$ -th big partitions, the rest of the jobs in the  $(i+1)$ -th big partition start  $Q-1$  steps after finishing the jobs in the  $i$ -th big partition. Also we need at least  $\frac{(1-\delta)n(n-1)Q}{(1+Q\epsilon)n^2} = (1-\epsilon_1)Q$  time to finish  $1-\delta$  fraction of the jobs in a big partition. This gives that the makespan is at least:

$$(1-\epsilon_1)(k+1)Q/2 + (k-1)(Q-1)/2 \geq (1-\epsilon_2)kQ,$$

where  $\epsilon_2 = \epsilon_2(Q, k, \epsilon, \delta)$ , which can be made small enough for an appropriate choice of  $Q, k, \epsilon$  and  $\delta$ .  $\square$

We illustrate the difference between the schedules corresponding to the completeness case and the soundness case in Figure 6.9.

### 6.6.2 $Q|\text{prec}|C_{max}$

In this section, we reduce a given  $k$ -partite graph  $G$  to an instance  $\mathcal{S}(k)$  of the scheduling problem  $Q|\text{prec}|C_{max}$ , and show that if  $G$  corresponds to the YES Case of Hypothesis 6.14, then the maximum makespan of  $\mathcal{S}(k)$  is roughly  $n$ , whereas a graph corresponding to the NO Case leads to a scheduling instance whose makespan is roughly the number of vertices in the graph, i.e.,  $nk$ . Formally, we prove the following theorem.

**Theorem 6.19.** *Assuming Hypothesis 6.14, it is NP-hard to approximate the scheduling problem  $Q|\text{prec}|C_{max}$  within any constant factor.*

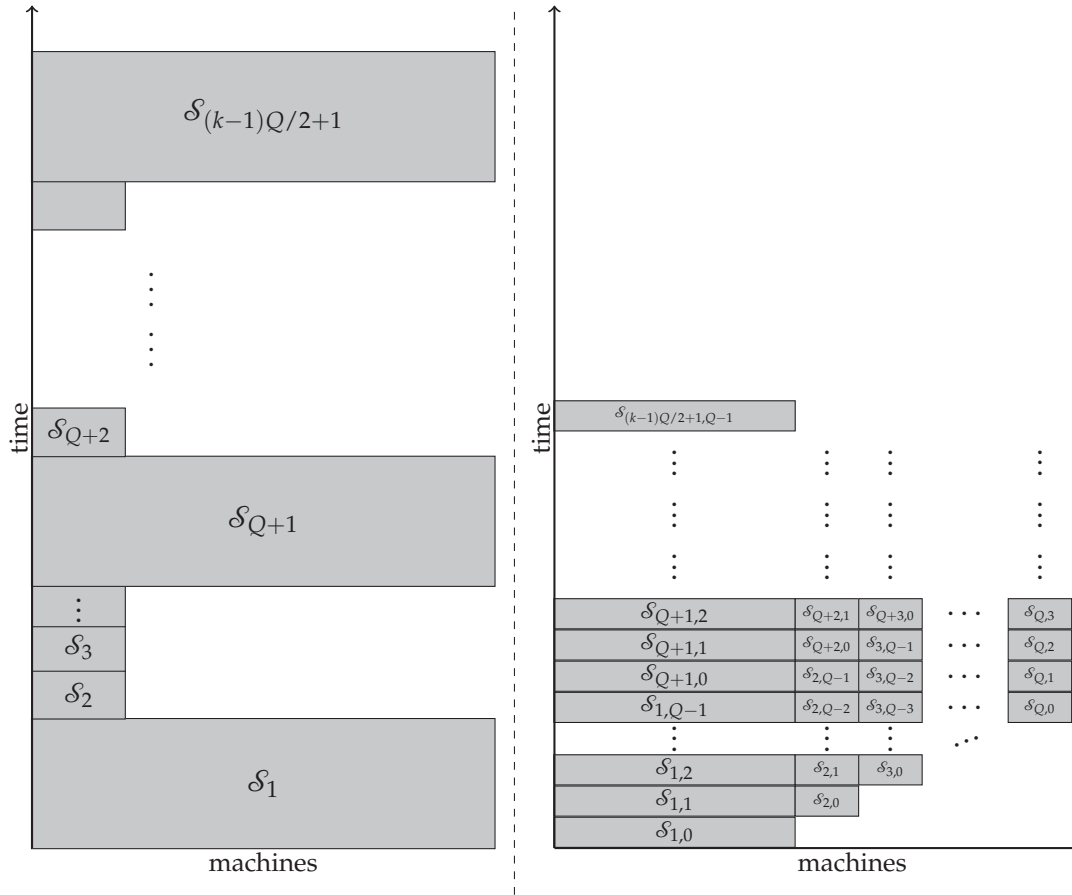


Figure 6.9 – Structure of the soundness versus completeness of  $P|prec, pmtn, p_j = 1|C_{max}$  assuming Hypothesis 6.14. The schedule on the left corresponds to the case where the starting graph represents the NO Case of Hypothesis 6.14; note the most of the machines are idle most of the time in this case. The schedule on the right corresponds to the case where the starting graph represents the YES Case of the hypothesis; Note that the all the machines are packed almost all the time. This case also illustrates our partitioning of the jobs in sets  $\{\mathcal{F}_t\}$ , where  $\mathcal{F}_t = \bigcup_{i,j:i+j-1=t} \mathcal{S}_{i,j}$

### Reduction

We present a reduction from a  $k$ -partite graph  $G = (V_1, \dots, V_k, E_1, \dots, E_{k-1})$  to an instance  $\mathcal{I}(k)$  of the scheduling problem  $Q|prec|C_{max}$ . The reduction is parametrised by a constant  $k$ , a constant  $Q \gg k$  such that  $n$  divides  $Q$  where  $n$  is the number of jobs in the scheduling instance, and a large enough number of machines  $m$  such that  $m \gg nk$ .

- For each vertex in  $v \in V_i$ , let  $\mathcal{F}_{v,i}$  be a set of  $m^{2(k-i)}$  jobs with processing time  $m^{i-1}$ , for every  $1 \leq i \leq k$ .
- For each edge  $e = (v, w) \in E_i$ , we have  $\mathcal{F}_{v,i} \prec \mathcal{F}_{w,i+1}$ , for  $1 \leq i < k$ .

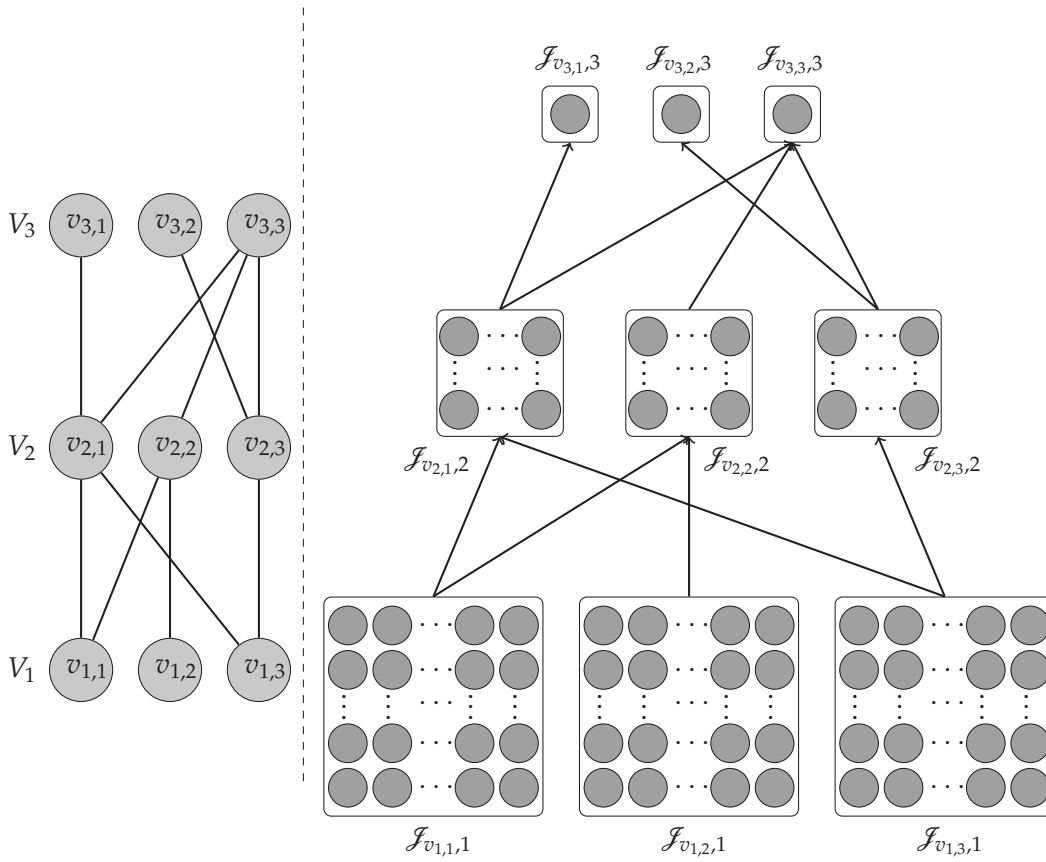


Figure 6.10 – The reduction from a 3-partite graph to a scheduling instance  $\mathcal{F}$  of  $Q|\text{prec}|C_{\max}$ . On the left, have the starting 3-partite graph  $G = (V_1, V_2, V_3, E_1, E_2)$ . On the right, we have the resulting scheduling instance over  $m$  machines according to the reduction in Section 6.6.2. Big rectangles correspond to sets of  $m^{2(k-1)} = m^4$  jobs mapping to vertices in  $V_1$ . Smaller rectangles correspond to sets of  $m^{2(k-2)} = m^2$  jobs mapping to vertices in  $V_2$ . Circles corresponds to jobs corresponding to vertices of  $V_3$  since in this case  $m^{2(k-i)} = 1$ .

- For each  $1 \leq i \leq k$  we create a set  $\mathcal{M}_i$  of  $m^{2(k-i)}$  machines with speed  $m^{i-1}$ .

We illustrate this reduction in Figure 6.10. For convenience, we later on refer to all the jobs corresponding to vertices in  $i$ -th partition as  $\mathcal{F}_i$ , i.e.,

$$\mathcal{F}_i = \bigcup_{v \in V_i} \mathcal{F}_{v,i}.$$

### Completeness

We show that if the given  $k$ -partite graph satisfies the properties of the YES Case, then there exist a schedule with makespan  $(1 + \epsilon_1)n$  for some small  $\epsilon_1 > 0$ . Towards this end, assume that the given  $k$ -partite graph satisfies the properties of the YES Case and let  $\{V_{i,j}\}$  for  $1 \leq i \leq k$  and  $0 \leq j \leq Q - 1$  be the claimed partitioning of Hypothesis 6.14.

The partitioning of the vertices naturally induces a partitioning  $\{\tilde{\mathcal{J}}_{i,j}\}$  for the jobs for  $1 \leq i \leq k$  and  $0 \leq j \leq Q - 1$  in the following way:

$$\tilde{\mathcal{J}}_{i,j} = \bigcup_{v \in V_{i,j}} \mathcal{J}_{v,i}.$$

Consider the schedule where for each  $1 \leq i \leq k$ , all the jobs in a set  $\tilde{\mathcal{J}}_{i,0}, \dots, \tilde{\mathcal{J}}_{i,Q-1}$  are scheduled on the machines in  $\mathcal{M}_i$ . Moreover, we start the jobs in  $\tilde{\mathcal{J}}_{i,j}$  after finishing the jobs in both  $\tilde{\mathcal{J}}_{i-1,j}$  and  $\tilde{\mathcal{J}}_{i,j-1}$  (if such sets exist). In other words, we schedule the jobs as follows (see Figure 6.11):

- For each  $1 \leq i \leq k$ , we first schedule the jobs in  $\tilde{\mathcal{J}}_{i,0}$ , then those in  $\tilde{\mathcal{J}}_{i,1}$  and so on up until  $\tilde{\mathcal{J}}_{i,Q-1}$ . The scheduling of the jobs on machines in  $\mathcal{M}_0$  starts at time 0 in the previously defined order.
- For each  $2 \leq i \leq k$ , we start the scheduling of jobs  $\tilde{\mathcal{J}}_{i,0}$  right after the completion of the jobs in  $\tilde{\mathcal{J}}_{i-1,0}$ .
- To respect the remaining precedence requirements, we start scheduling the jobs in  $\tilde{\mathcal{J}}_{i,j}$  right after the execution of jobs in  $\tilde{\mathcal{J}}_{i,j-1}$  and as soon as the jobs in  $\tilde{\mathcal{J}}_{i-1,j}$  have finished executing, for  $2 \leq i \leq k$  and  $1 \leq j \leq Q - 1$ .

By the aforementioned construction of the schedule, we know that the precedence constraints are satisfied, and hence the schedule is feasible. That is, since we are in YES Case, we know that vertices in  $V_{i',j'}$  might only have edges to the vertices in  $V_{i,j}$  for all  $1 \leq i' < i \leq k$  and  $1 \leq j' \leq j < Q$ , which means that the precedence constraints may only be from the jobs in  $\tilde{\mathcal{J}}_{i',j'}$  to jobs in  $\tilde{\mathcal{J}}_{i,j}$  for all  $1 \leq i' < i \leq k$  and  $0 \leq j' \leq j < Q$ . Therefore the precedence constraints are satisfied.

Moreover, we know that there are at most  $m^{2(k-i)}n(1 + \epsilon)/Q$  jobs of length  $m^{i-1}$  in  $\tilde{\mathcal{J}}_{i,j}$ , and  $m^{2(k-i)}$  machines with speed  $m^{i-1}$  in each  $\mathcal{M}_i$  for all  $1 \leq i \leq k$ ,  $j \in [Q]$ . Thus, it takes  $(1 + \epsilon)n/Q$  time to schedule all the jobs in  $\tilde{\mathcal{J}}_{i,j}$  on the machines in  $\mathcal{M}_i$  for all  $1 \leq i \leq k$ ,  $j \in [Q]$ , which in turn implies that we can schedule all the jobs in a set  $\tilde{\mathcal{J}}_{i,j}$  between time  $(i + j - 1)(1 + \epsilon)n/Q$  and  $(i + j)(1 + \epsilon)n/Q$ . Hence, the makespan is at most  $(k + Q)(1 + \epsilon)n/Q$  which is equal to  $(1 + \epsilon_1)n$ , by the assumption that  $Q \gg k$ .

### Soundness

We shall now show that if the  $k$ -partite graph  $G$  corresponds to the NO Case of Hypothesis 6.14, then any feasible schedule for  $\mathcal{F}(k)$  must have a makespan of at least  $cnk$ , where  $c := (1 - 2\delta)(1 - k^2/m)$  can be made arbitrary close to one.

**Lemma 6.20.** *In a feasible schedule  $\sigma$  for  $\mathcal{F}(k)$  such that the makespan of  $\sigma$  is at most  $nk$ , the following is true: for every  $1 \leq i \leq k$ , at least a  $(1 - k^2/m)$  fraction of the jobs in  $\mathcal{L}_i = \cup_{v \in V_i} \mathcal{F}_{v,i}$  are scheduled on machines in  $\mathcal{M}_i$ .*

*Proof.* We first show that no job in  $\mathcal{L}_i$  can be scheduled on machines in  $\mathcal{M}_j$ , for all  $1 \leq j < i \leq k$ . This is true, because any job  $J \in \mathcal{F}_i$  has a processing time of  $m^{i-1}$ , whereas the speed of any machine  $M \in \mathcal{M}_j$  is  $m^{j-1}$  by construction, and hence scheduling the job  $J$  on the machine  $M$  would require  $m^{i-1}/m^{j-1} \geq m$  time steps. But since  $m \gg nk$ , this contradicts the assumption that the makespan is at most  $nk$ .

We now show that at most  $k^2/m$  fraction of the jobs in  $\mathcal{L}_i$  can be scheduled on the machines in  $\mathcal{M}_j$  for  $1 \leq i < j \leq k$ . Fix any such pair  $i$  and  $j$ , and assume that all the machines in  $\mathcal{M}_j$  process the jobs in  $\mathcal{L}_i$  during all the  $T \leq nk$  time steps of the schedule. This accounts for a total  $T \frac{m^{2(k-j)m^{j-1}}}{m^{i-1}} \leq m^{2k-j-i}nk$  jobs processed from  $\mathcal{L}_i$ , which constitutes at most  $\frac{m^{2k-j-i}nk}{nm^{2(k-i)}} \leq \frac{k}{m}$  fraction of the total number of jobs in  $\mathcal{L}_i$ .  $\square$

Let  $\sigma$  be a schedule whose makespan is at most  $nk$ , and fix  $\gamma > k^2/m$  to be a small constant. From Lemma 6.20 we know that for every  $1 \leq i \leq k$ , at least an  $(1 - \gamma)$  fraction of the jobs in  $\mathcal{L}_i$  is scheduled on machines in  $\mathcal{M}_i$ . From the structure of the graph in the NO Case of the  $k$ -partite Problem, we know that we cannot start more than  $\delta$  fraction of the jobs in  $\mathcal{L}_i$  before finishing  $(1 - \delta)$  fraction of the jobs in  $\mathcal{L}_{i-1}$ , for all  $2 \leq i \leq k$ . Hence the maximum makespan of any such schedule  $\sigma$  is at least  $(1 - 2\delta)(1 - \gamma)nk$ . See figure 6.11.

## 6.7 Conclusion

We proposed in this chapter a natural but nontrivial generalization of Theorem 6.7, that seems to capture the hardness of a large family of scheduling problems with precedence constraints. Namely, we showed that if the structural  $k$ -partite hypothesis in Hypothesis 6.14 holds, then we close the approximability gap of  $P|prec, pmtn|C_{max}$  and we rule out any constant factor approximation for  $Q|prec|C_{max}$ .

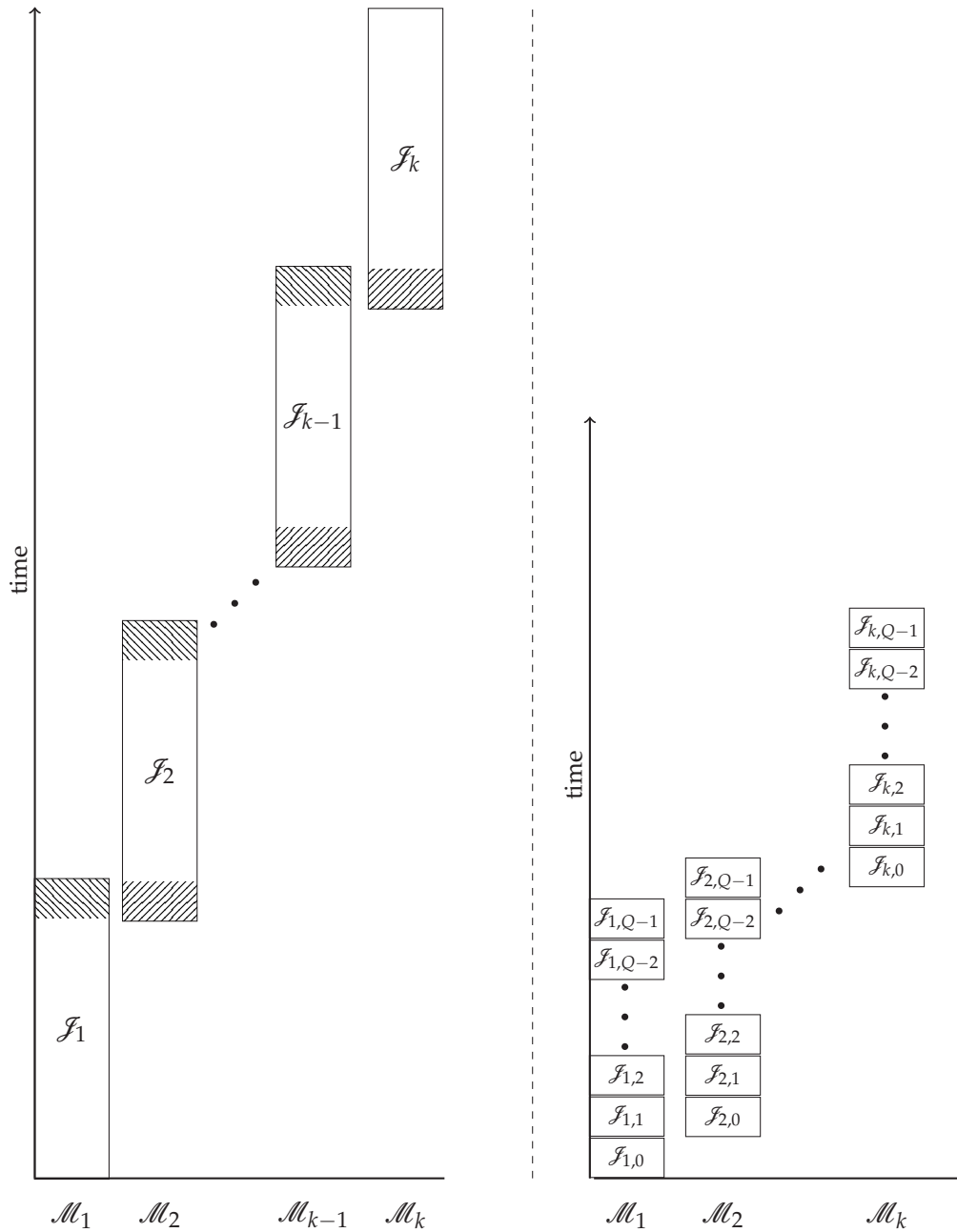


Figure 6.11 – Structure of the soundness versus completeness of  $Q|prec|C_{max}$  assuming Hypothesis 6.14. The schedule on the left corresponds to the case where the graph represents the NO Case of Hypothesis 6.14; note that most of the machines are idle but for a small fraction of times. The schedule on the right corresponds to the case where the graph represents the YES Case; the schedule is almost packed. This case also illustrates the ordering of the jobs within each machine according to the partitioning of the jobs in the  $k$ -partite graph.





## 7 Conclusion and Future Directions

We conclude in this section with some natural remaining open problems suggested by the work presented in the previous chapters.

### 7.1 LP-lower bounds

We presented in Chapter 4 Linear Programming lower bounds for the VERTEX COVER problem and the INDEPENDENT SET problem. We believe that our approach extends to many other related problems, as we proved that it also applies to  $q$ -UNIFORM-VERTEX-COVER. Moreover, we would like to stress that our reduction is agnostic to whether it is used for LPs or SDPs, and Lasserre gap instances for 1F-CSP together with [76] and our reduction would provide SDP hardness of approximation for the VERTEX COVER problem. This already holds for the INDEPENDENT SET problem as we saw in Section 4.7; there it only matters that the starting CSP has a (large) gap between the soundness and completeness, and does not necessarily need to have only one free bit.

Note that we are only able to establish hardness of approximations for the stable set problem within any constant factor, whereas assuming  $P \neq NP$  we can establish hardness of approximation within  $n^{1-\epsilon}$ . The reason for this gap is that the standard amplification techniques via graph products do not fall into the reduction framework in [25]. Also, there will be limits to amplification, as established by the upper bounds in Section 4.6. Thus it would be actually interesting to narrow down this gap between the lower bound of  $\Omega(1)$  and the upper bound of  $O(\sqrt{n})$ .

Moreover, it would be interesting to understand the structure of the graph used in the proof of Theorem 4.12. For our purposes, arguing about the size of the vertex cover in the soundness and completeness case followed from the soundness and completeness of the original 1F-CSP instance that, in turn, followed from the starting UNIQUE GAMES instance. In particular, it is plausible that a further understanding of the structure of this graph could yield SOS lower bounds for the VERTEX COVER problem, or perhaps certify

that this family of graphs is actually *easy* as far as the SOS hierarchy is concerned.

### 7.2 Knapsack

We presented in Chapter 5 the first quasi-polynomial-size LP relaxation of MIN-KNAPSACK with constant integrality gap from polylog-depth circuits for weighted threshold functions.

This result sheds new light on the approximability of MIN-KNAPSACK via small LPs by connecting it to the complexity of monotone circuits. For instance, it follows from our results that proving that no  $n^{O(1)}$ -size LP relaxation for MIN-KNAPSACK can have integrality gap at most  $\alpha$  for some  $\alpha > 2$  would rule out the existence of  $O(\log n)$ -depth monotone circuits with bounded fan-in for weighted threshold functions on  $n$  inputs, which is an open problem.

Finally, let us further mention two open questions following this work. First, it would be interesting to find an efficient (quasi-polynomial time) procedure to explicitly write down our linear program for MIN-KNAPSACK. Second, it would be interesting to understand whether there is a “combinatorial” interpretation of our relaxation.

### 7.3 The Scheduling Problems and the $k$ -Partite Hypothesis

We proposed in Chapter 6 a natural but nontrivial generalisation of Theorem 6.7, that seems to capture the hardness of a large family of scheduling problems with precedence constraints. We have shown that if the structural  $k$ -partite hypothesis (i.e., Hypothesis 6.14) holds, then we close the approximability gap of  $P|_{\text{prec, pmtn}}|C_{\max}$  and we rule out any constant factor approximation for  $Q|_{\text{prec}}|C_{\max}$ . It is interesting to investigate whether this generalisation also illustrates potential intrinsic hardness of other scheduling problems, for which the gap between the best known approximation algorithm and the best known hardness result persists.

A natural direction would be to prove Hypothesis 6.14; we know how to prove a *less-structured* version of it using the bipartite graph resulting from the variant of the UNIQUE GAMES Conjecture in [8] (See Appendix B in [14]). One can also tweak the dictatorship  $T_{\epsilon,t}$  of [8], to yield a  $k$ -partite graph instead of a bipartite one. However, composing this test with a UNIQUE GAMES instance adds a noisy component to our  $k$ -partite graph, that we do not know how to control, since it is due to the non-perfect completeness of the UNIQUE GAMES instance. One can also try to impose (a variant of) this dictatorship test on D-TO-1 GAMES instances, and perhaps prove the hypothesis assuming the D-TO-1 Conjecture, although we expect the size of the partitions will deteriorate as  $k$  increases.

# A List of Problems

To facilitate the reading of the various chapters of the thesis, we repeat here the definition of all the problems that we tackle, and include the definition of some of the problems that we only mention throughout the thesis for completeness.

## Graph Problems

We start by defining the VERTEX COVER problem, and its generalization on  $k$ -uniform hypergraphs that we denote by  $q$ -UNIFORM-VERTEX-COVER. We also define the INDEPENDENT SET problem, the complement of the VERTEX COVER problem.

**VERTEX COVER:** Given a graph  $G = (V, E)$ , we say that a subset  $C \subseteq V$  of vertices is a vertex cover of  $G$  if every edge  $e \in E$  has at least one of its two endpoints in  $C$ . In the VERTEX COVER problem, the goal is to find the minimum cardinality vertex cover, or equivalently the minimum weighted vertex cover if  $G$  was vertex-weighted.

**$q$ -UNIFORM-VERTEX-COVER:** Given a  $k$ -uniform hypergraph  $H = (V, E)$  (i.e., each hyperedge  $e \in E$  contains exactly  $k$  vertices), we say that a subset  $C \subseteq V$  is a vertex cover of  $G$ , if each hyperedge  $e \in E$  has at least one its  $k$  vertices in  $C$ . The goal in the  $q$ -UNIFORM-VERTEX-COVER problem is then to find the minimum cardinality (weight) vertex cover of  $G$ .

**INDEPENDENT SET:** Given a graph  $G = (V, E)$ , we say that a subset  $I \subseteq V$  of vertices is an independent set of  $G$  if no edge  $e \in E$  has both its endpoints in  $C$ . In the INDEPENDENT SET problem, the goal is to find the maximum cardinality (weight) independent set. This notion naturally generalize to hypergraphs.

**MAX CUT:** Given a graph  $G = (V, E)$ , the value of a cut  $S \subseteq V$  is the number of edges going from  $S$  to  $V \setminus S$ . In the MAX CUT problem, the goal is to find the maximum value cut of  $G$ .

## Appendix A. List of Problems

---

**UNIQUE GAMES:** A UNIQUE GAMES instance  $\mathcal{U} = (G, [R], \Pi)$  is defined by a graph  $G = (V, E)$ , where every edge  $uv \in E$  is associated with a bijection map  $\pi_{u,v} \in \Pi$  such that  $\pi_{u,v} : [R] \mapsto [R]$  (we set  $\pi_{v,u} := \pi_{u,v}^{-1}$ ). Here,  $[R]$  is known as the label set. The goal is to find a labeling  $\Lambda : V \mapsto [R]$  that maximizes the number of satisfied edges, where an edge  $uv$  is satisfied by  $\Lambda$  if  $\pi_{u,v}(\Lambda(u)) = \Lambda(v)$ .

## Constraint Satisfaction Problems

A CONSTRAINT SATISFACTION PROBLEM  $\Pi_{n,R,\mathcal{P}}$  is defined by specifying:

1. The number of variables  $n$ , where the variables in the cases are denoted by  $x_1, \dots, x_n$ .
2. The domain  $[R]$  of variables (or equivalently the domain size  $R$ ), which means that every variable  $x_i$  is allowed to take values from  $\{0, 1, \dots, R-1\}$ .
3. The family of predicates  $\mathcal{P} = \{P_1, \dots, P_\ell\}$ , where each predicate  $P_i$  for  $i = 1, \dots, \ell$  has an arity  $k_i \in \mathbb{N}^+$ . In other words, each  $P_i$  is of the form

$$P_i : [R]_{i_1}^{k_i} \mapsto \{0, 1\}.$$

An instance  $\mathcal{J} \in \Pi_{n,R,\mathcal{P}}$  is further specified by a collection  $\mathcal{C} = \{C_{S_1, A_1}, \dots, C_{S_m, A_m}\}$  of constraints, where each constraint  $C_{S,A} \in \mathcal{C}$  is of predicate type  $P_C$  for some  $P_C \in \mathcal{P}$ , and evaluates as follows for an assignment  $x \in [R]^n$  of the variables:

$$C_{S,A}(x) = P_C(x_{i_1} \ominus A_1, \dots, x_{i_k} \ominus A_k),$$

where

- $k$  is the arity of the predicate  $P_C$ .
- $S \subset \{1, 2, \dots, n\}$  with  $|S| = k$  is the ordered set of indices of the variables on which  $C_{S,A}$  evaluates.
- $A \in [R]^k$  is the literals assignment of the variables in  $S$ .

For the Boolean case, the entry  $A_j$  of the vector  $A = (A_1, \dots, A_k)$  dictates whether the variable  $x_{i_j}$  appears negated in  $C_{S,A}$  or not.

Given an instance  $\mathcal{J} \in \Pi_{n,R,\mathcal{P}}$ , and an assignment  $x \in [R]^n$  the value of  $\mathcal{J}$  evaluated on  $x$  is

$$\mathcal{J}(x) = \frac{1}{m} \sum_{i=1}^m C_{S_i, A_i}(x).$$

---

Furthermore, the goal in these problems is to find an optimal assignment that maximizes the number of satisfied constraints, i.e., to find

$$\operatorname{argmax}_{x \in [R]^k} \mathcal{F}(x).$$

Before we proceed with the definition of the CONSTRAINT SATISFACTION PROBLEMS of interest, we shall first define the following three (families) of predicates:

**The family of 1F-CSP predicates:** A binary predicate  $P$  of arity  $k$  is said to be an 1F-CSP predicate if  $P$  has only two accepting configurations out of the  $2^k$  possible ones. For a fixed arity  $k$ ,  $1\text{F-CSP} := 1\text{F-CSP}^k$  is the family of all such predicates.

**The  $\kappa$ -NOR predicate:** The  $\kappa$ -NOR  $:= \kappa\text{-NOR}_{k,R}$  is an  $R$  domain predicate of arity  $k$  that has exactly one accepting configuration out of the  $R^k$  possible ones. Namely, this accepting configuration is the all-zeros input.

**The 3-SAT predicate:** The 3-SAT predicate is the boolean OR predicate of arity 3, i.e.,  $3\text{-SAT}(x_1, x_2, x_3) = x_1 \vee x_2 \vee x_3$ .

Equipped with this, we can readily define the CONSTRAINT SATISFACTION PROBLEMS that appear throughout this thesis:

**1F-CSP $_n$ :** The  $1\text{F-CSP}_n$  is a boolean CONSTRAINT SATISFACTION PROBLEM over  $n$  variables where the family of predicates  $\mathcal{P}$  is the set of all  $1\text{F-CSP}_k$  predicates, where the arity  $k$  is usually clear from the context.

**$\kappa$ -NOR $_n$ :** The  $\kappa\text{-NOR}_n$  is a domain  $R$  CONSTRAINT SATISFACTION PROBLEM over  $n$  variables where  $\mathcal{P}$  contains only the  $\kappa\text{-NOR}_{k,R}$  predicate, where the arity  $k$  and the domain size  $R$  are usually clear from the context.

**MAX 3-SAT $_n$ :** The  $\text{MAX 3-SAT}_n$  is a boolean CONSTRAINT SATISFACTION PROBLEM over  $n$  variables where the only allowed predicate is the 3-SAT predicate.

Note that both MAX CUT and UNIQUE GAMES defined earlier can also be seen as CONSTRAINT SATISFACTION PROBLEMS, where the predicate in the first is the XOR predicate, and the predicate in the second is only satisfied if the variable assignment (i.e., vertices labels) satisfy the bijective maps associated with the edges.

## MIN-KNAPSACK Problem

In the Minimum Knapsack Problem (MIN-KNAPSACK), we are given a demand  $D \geq 0$ , and  $n$  items where each item  $i \in \{1, 2, \dots, n\}$  is associated with a cost  $c_i \in \mathbb{R}^+$  and size

## Appendix A. List of Problems

---

$s_i \in \mathbb{R}^+$ . A feasible solution of the knapsack problem is a set  $S \subseteq \{1, 2, \dots, n\}$  such that  $\sum_{i \in S} s_i \geq D$ . The goal in the MIN-KNAPSACK problem is to find the minimum cost feasible solution.

### SINGLE-DEMAND FACILITY LOCATION Problem

In the SINGLE-DEMAND FACILITY LOCATION problem, we are given a set  $F$  of  $n$  facilities, such that each facility  $i \in F$  has a capacity  $s_i$ , an opening cost  $f_i$ , and a per-unit cost  $c_i$  to serve the demand. The goal is to serve the demand  $D$  by opening a subset  $S \subseteq F$  of facilities such that the combined cost of opening these facilities and serving the demand is minimized.

### Scheduling Problems

In the scheduling problems that we consider, we are given a set  $\mathcal{M}$  of  $m$  machines and a set  $\mathcal{J}$  of  $n$  jobs with precedence constraints, and the goal is find a feasible schedule that minimizes the makespan, i.e., the maximum completion time. We will be interested in the following two variants of this general setting:

**P|prec, pmtn|C<sub>max</sub>**: In this model, the machines are assumed to be parallel and identical, i.e., the processing time of a job  $J_j \in \mathcal{J}$  is the same on any machine  $M_i \in \mathcal{M}$  ( $p_{i,j} = p_j$  for all  $M_i \in \mathcal{M}$ ). Furthermore, preemption is allowed, and hence the processing of a job can be paused and resumed at later stages, not necessarily on the same machine.

**P|prec|C<sub>max</sub>**: This is a restricted version of the P|prec, pmtn|C<sub>max</sub> problem, where the preemption is not allowed.

**Q|prec|C<sub>max</sub>**: In this model, the machines are assumed to be parallel and uniform, i.e., each machine  $M_i \in \mathcal{M}$  has a speed  $s_i$ , and the time it takes to process job  $J_j \in \mathcal{J}$  on this machine is  $p_j/s_i$ .

Moreover, we also consider the single machine scheduling problem  $1|prec|\sum_j w_j C_j$  where we have *one* machine, and precedence constraints between the  $n$  jobs, but the goal is the problem is to find the schedule that has the minimum weighted completion time of the jobs.

# Bibliography

- [1] H. C. An, M. Singh, and O. Svensson. LP-based algorithms for capacitated facility location. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 256–265, 2014.
- [2] S. Arora, B. Bollobás, and L. Lovász. Proving integrality gaps without knowing the linear program. In *Proc. FOCS 2002*, pages 313–322, 2002.
- [3] S. Arora, B. Bollobás, L. Lovász, and I. Turlakis. Proving integrality gaps without knowing the linear program. *Theory Comput.*, 2:19–51, 2006.
- [4] Mark B. and Ankur M. An information complexity approach to extended formulations. In *Proc. STOC 2013*, pages 161–170, 2013.
- [5] E. Balas. Facets of the knapsack polytope. *Math. Program.*, 8:146–164, 1975.
- [6] N. Bansal, N. Buchbinder, and J. S. Naor. Randomized competitive algorithms for generalized caching. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 235–244. ACM, 2008.
- [7] N. Bansal, A. Gupta, and R. Krishnaswamy. A constant factor approximation algorithm for generalized min-sum set cover. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1539–1545. Society for Industrial and Applied Mathematics, 2010.
- [8] N. Bansal and S. Khot. Optimal long code test with one free bit. In *Proc. FOCS 2009*, FOCS '09, pages 453–462, Washington, DC, USA, 2009. IEEE Computer Society.
- [9] N. Bansal and S. Khot. Optimal long code test with one free bit. In *Proc. FOCS 2009*, pages 453–462, 2009.
- [10] N. Bansal and K. Pruhs. The geometry of scheduling. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 407–414. IEEE, 2010.
- [11] B. Barak, S. O. Chan, and P. K. Kothari. Sum of squares lower bounds from pairwise independence. In *Proc. SODA 2015*, pages 97–106, 2015.

## Bibliography

---

- [12] A. Bazzi, S. Fiorini, S. Huang, and O. Svensson. Small extended formulation for knapsack cover inequalities from monotone circuits. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2326–2341. SIAM, 2017.
- [13] A. Bazzi, S. Fiorini, S. Pokutta, and O. Svensson. Small linear programs cannot approximate Vertex Cover within a factor of  $2 - \epsilon$ . In *IEEE 56th Annual Symp. on Foundations of Computer Science (FOCS)*, pages 1123–1142, 2015.
- [14] A. Bazzi and A. Norouzi-Fard. Towards tight lower bounds for scheduling problems. In *Algorithms-Esa 2015*, pages 118–129. Springer, 2015.
- [15] A. Beimel and E. Weinreb. Monotone circuits for weighted threshold functions. In *Proc. of the 20th Annual IEEE Conference on Computational Complexity*, pages 67–75, 2005.
- [16] A. Beimel and E. Weinreb. Monotone circuits for monotone weighted threshold functions. *Information Processing Letters*, 97:12–18, 2006.
- [17] S. Benabbas, K. Georgiou, A. Magen, and M. Tulsiani. SDP gaps from pairwise independence. *Theory Comput.*, 8(12):269–289, 2012.
- [18] A. Bhaskara, M. Charikar, A. Vijayaraghavan, V. Guruswami, and Y. Zhou. Polynomial integrality gaps for strong SDP relaxations of Densest  $k$ -subgraph. In *Proc. SODA 2012*, pages 388–405, 2012.
- [19] D. Bienstock. Approximate formulations for 0-1 knapsack sets. *Operations Research Letters*, pages 317–320, 2008.
- [20] D. Bienstock and B. McClosky. Tightening simple mixed-integer sets with guaranteed bounds. *Mathematical Programming*, 133(1):337–363, 2012.
- [21] G. Braun, S. Fiorini, S. Pokutta, and D. Steurer. Approximation limits of linear programs (beyond hierarchies). In *53rd IEEE Symp. on Foundations of Computer Science (FOCS 2012)*, pages 480–489, 2012.
- [22] G. Braun, S. Fiorini, S. Pokutta, and D. Steurer. Approximation limits of linear programs (beyond hierarchies). *Math. Oper. Res.*, 40(3):756–772, 2015.
- [23] G. Braun and S. Pokutta. Common information and unique disjointness. *Algorithmica*, 76(3):597–629, 2016.
- [24] G. Braun, S. Pokutta, and D. Zink. Inapproximability of combinatorial problems via small LPs and SDPs. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC)*, pages 107–116, 2015.
- [25] G. Braun, S. Pokutta, and D. Zink. Inapproximability of combinatorial problems via small LPs and SDPs. In *Proc. STOC 2015*, pages 107–116, 2015.



- 
- [26] G. Braun, A. Roy, and S. Pokutta. Stronger Reductions for Extended Formulations. In *Proc. IPCO 2016*, pages 350–361, 2016.
- [27] J. Briët, D. Dadush, and S. Pokutta. On the existence of 0/1 polytopes with high semidefinite extension complexity. *Math. Program.*, 153(1):179–199, 2015.
- [28] T. Carnes and D. Shmoys. Primal-dual schema for capacitated covering problems. In *Proceedings of the 13th International Conference on Integer Programming and Combinatorial Optimization, IPCO'08*, pages 288–302, Berlin, Heidelberg, 2008. Springer-Verlag.
- [29] R. D. Carr, L. K. Fleischer, V. J. Leung, and C. A. Philipps. Strengthening integrality gaps for capacitated network design and covering problems. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms (SODA '00)*, pages 106–115, 2000.
- [30] D. Chakrabarty, E. Grant, and J. Konemann. On column restricted and priority integer covering programs. In *Conference on Integer Programming and Combinatorial Optimization*, 2010.
- [31] S. O. Chan, J. R. Lee, P. Raghavendra, and D. Steurer. Approximate Constraint Satisfaction Requires Large LP Relaxations. In *Proc. FOCS 2013*, pages 350–359, 2013.
- [32] M. Charikar, K. Makarychev, and Y. Makarychev. Integrality Gaps for Sherali-Adams Relaxations. In *Proc. STOC 2009*, pages 283–292, 2009.
- [33] C. Chekuri and M. Bender. An efficient approximation algorithm for minimizing makespan on uniformly related machines. *Journal of Algorithms*, 41(2):212–224, 2001.
- [34] X. Chen, I. Carboni Oliveira, and R. Servedio. Addition is exponentially harder than counting for shallow monotone circuits. arXiv:1508.03061, 2015.
- [35] M. Cheung and D. Shmoys. A primal-dual approximation algorithm for min-sum single-machine scheduling problems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 135–146. Springer, 2011.
- [36] F. Chudak and D. Shmoys. Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds. *Journal of Algorithms*, 30(2):323–343, 1999.
- [37] M. Conforti, G. Cornuéjols, and G. Zambelli. Extended formulations in combinatorial optimization. *4OR*, 8:1–48, 2010.
- [38] M. Conforti, G. Cornuéjols, and G. Zambelli. Extended formulations in combinatorial optimization. *4OR*, 8:1–48, 2010.

## Bibliography

---

- [39] W. F. de la Vega and C. Kenyon-Mathieu. Linear programming relaxations of Maxcut. In *Proc. SODA 2007*, pages 53–61, 2007.
- [40] I. Dinur and S. Safra. On the hardness of approximating minimum vertex cover. *Ann. of Math.*, 162(1):439–485, 2005.
- [41] H. Efsandiari, M. Hajiaghyi, J. Könemann, H. Mahini, D. Malec, and L. Sanita. Approximate deadline-scheduling with precedence constraints. In *Algorithms-ESA 2015*, pages 483–495. Springer, 2015.
- [42] Y. Faenza, S. Fiorini, R. Grappe, and H. R. Tiwary. Extended formulations, non-negative factorizations and randomized communication protocols. *Math. Programming*, 153:75–94, 2015.
- [43] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *J. ACM*, 43:268–292, 1996.
- [44] U. Feige and S. Jozeph. Demand queries with preprocessing. In *Proc. ICALP 2014*, pages 477–488, 2014.
- [45] 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 44th Annual ACM Symposium on Theory of Computing (STOC)*, 2012.
- [46] S. Fiorini, S. Massar, S. Pokutta, H.R. Tiwary, and R. de Wolf. Exponential lower bounds for polytopes in combinatorial optimization. *J. ACM*, 62(2), 2015.
- [47] D. Gangal and A. Ranade. Precedence constrained scheduling in  $(2-7/(3p+1))$  optimal. *Journal of Computer and System Sciences*, 74(7):1139–1146, 2008.
- [48] K. Georgiou and A. Magen. Limitations of the Sherali-Adams lift and project system: Compromising local and global arguments. *Technical Report CSRG-587*, 2008.
- [49] K. Georgiou, A. Magen, T. Pitassi, and I. Turlakis. Integrality gaps of  $2 - o(1)$  for vertex cover SDPs in the Lovász-Schrijver hierarchy. *SIAM J. Comput.*, 39(8):3553–3570, 2010.
- [50] K. Georgiou, A. Magen, and M. Tulsiani. Optimal Sherali-Adams Gaps from Pairwise Independence. In *Proc. APPROX 2009*, pages 125–139, 2009.
- [51] T. Gonzalez and D. Johnson. A new algorithm for preemptive scheduling of trees. *Journal of the ACM (JACM)*, 27(2):287–312, 1980.
- [52] M. Göös, R. Jain, and T. Watson. Extension complexity of independent set polytopes. arXiv:1604.07062, April 2016.

- 
- [53] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(9):1563–1581, 1966.
- [54] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, 5(2):287–326, 1979.
- [55] P.L. Hammer, E.L. Johnson, and U.N. Peled. Facets of regular 0-1 polytopes. *Math. Program.*, 8:179–206, 1975.
- [56] J. Håstad. Clique is hard to approximate within  $n^{1-\epsilon}$ . *Acta Math.*, 182(1):105–142, 1999.
- [57] J. Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.
- [58] D. Hochbaum. Approximating covering and packing problems: Set cover, vertex cover, independent set and related problems. In *Approximation Algorithms for NP-hard Problems*, pages 94–143. PWS Publishing Company, 1997.
- [59] D. Hochbaum and D. Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *Journal of the ACM (JACM)*, 34(1):144–162, 1987.
- [60] D. Hochbaum and D. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM journal on computing*, 17(3):539–551, 1988.
- [61] P. Hrubeš. On the nonnegative rank of distance matrices. *Information Processing Letters*, 112:457–461, 2012.
- [62] V. Kaibel. Extended formulations in combinatorial optimization. *Optima*, 85:2–7, 2011.
- [63] V. Kaibel, K. Pashkovich, and D.O. Theis. Symmetry matters for the sizes of extended formulations. In *Proc. IPCO 2010*, pages 135–148, 2010.
- [64] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM J. Discrete Math.*, 3:255–265, 1990.
- [65] A. R. Karlin, C. Mathieu, and C. T. Nguyen. Integrality Gaps of Linear and Semi-definite Programming Relaxations for Knapsack. In *Proc. IPCO 2011*, pages 301–314, 2011.
- [66] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

## Bibliography

---

- [67] S. Khot. On the power of unique 2-prover 1-round games. In *Proc. STOC 2002*, pages 767–775, 2002.
- [68] S. Khot and O. Regev. Vertex cover might be hard to approximate to within  $2 - \epsilon$ . *J. Comput. System Sci.*, pages 335–349, 2008.
- [69] S. Khot and O. Regev. Vertex cover might be hard to approximate to within  $2 - \epsilon$ . *J. Comput. Syst. Sci.*, 74(3):335–349, 2008.
- [70] S. Khot and R. Saket. SDP Integrality Gaps with Local  $\ell_1$ -Embeddability. In *Proc. FOCS 2009*, pages 565–574, 2009.
- [71] P. Kothari, R. Meka, and P. Raghavendra. Approximating rectangles by juntas and weakly-exponential lower bounds for lp relaxations of csps. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 590–603. ACM, 2017.
- [72] J. B. Lasserre. An explicit exact SDP relaxation for nonlinear 0-1 programs. In *Proc. IPCO 2001*, pages 293–203, 2001.
- [73] J. B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM J. Optim.*, pages 796–817, 2001.
- [74] J. B. Lasserre. An explicit equivalent positive semidefinite program for nonlinear 0-1 programs. *SIAM J. Optim.*, 12:756–769, 2002.
- [75] E. L. Lawler. Fast approximation algorithms for knapsack problems. *Mathematics of Operations Research*, 4:339–356, 1979.
- [76] J.R. Lee, P. Raghavendra, and D. Steurer. Lower bounds on the size of semidefinite programming relaxations. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC)*, pages 567–576, 2015.
- [77] J. K. Lenstra and A. H. G. Rinnooy Kan. Computational complexity of discrete optimization problems. *Annals of Discrete Mathematics*, 4:121–140, 1979.
- [78] R. Levi, A. Lodi, and M. Sviridenko. Approximation algorithms for the capacitated multi-item lot-sizing problem via flow-cover inequalities. *Mathematics of Operations Research*, 33(2):461–474, 2008.
- [79] Shi Li. Scheduling to minimize total weighted completion time via time-indexed linear programming relaxations. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, 2017.
- [80] L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM J. Optim.*, 1(2):166–190, 1991.

- 
- [81] R. McNaughton. Scheduling with deadlines and loss functions. *Management Science*, 6(1):1–12, 1959.
- [82] J. Mestre and J. Verschae. A 4-approximation for scheduling on a single machine with general cost function. *arXiv preprint arXiv:1403.0298*, 2014.
- [83] E. Mossel, R. O’Donnell, and K. Oleszkiewicz. Noise stability of functions with low influences: invariance and optimality. *Ann. of Math.*, 171(1):295–341, 2010.
- [84] R. Muntz and E. Coffman Jr. Optimal preemptive scheduling on two-processor systems. *Computers, IEEE Transactions on*, 100(11):1014–1020, 1969.
- [85] R. Muntz and E. Coffman Jr. Preemptive scheduling of real-time tasks on multi-processor systems. *Journal of the ACM (JACM)*, 17(2):324–338, 1970.
- [86] S. Muroga. *Threshold Logic and Its Applications*. Wiley-Interscience, 1971.
- [87] R. O’Donnell. *Analysis of Boolean Functions*. Cambridge University Press, New York, NY, USA, 2014.
- [88] M. W. Padberg, T. J. Van Roy, and L. A. Wolsey. Valid inequalities for fixed charge problems. *Oper. Res.*, 33:842–861, 1985.
- [89] P. Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, California Institute of Technology, 2000.
- [90] K. Pashkovich. *Extended Formulations for Combinatorial Polytopes*. PhD thesis, Magdeburg Universität, 2012.
- [91] S. Pokutta and M. Van Vyve. A note on the extension complexity of the knapsack polytope. *Operations Research Letters*, 41:347–350, 2013.
- [92] P. Raghavendra and D. Steurer. Integrality Gaps for Strong SDP Relaxations of Unique games. In *Proc. FOCS 2009*, pages 575–585, 2009.
- [93] T. Rothvoß. Some 0/1 polytopes need exponential size extended formulations. *Math. Program.*, 142(1–2):255–268, 2013.
- [94] T. Rothvoß. The matching polytope has exponential extension complexity. In *Proc. STOC 2014*, pages 263–272, 2014.
- [95] G. Schoenebeck. Linear Level Lasserre Lower Bounds for Certain k-CSPs. In *Proc. FOCS 2008*, pages 593–602, 2008.
- [96] G. Schoenebeck, L. Trevisan, and M. Tulsiani. Tight Integrality Gaps for Lovász-Schrijver LP Relaxations of Vertex Cover and Max Cut. In *Proc. STOC 2007*, pages 302–310, 2007.

## Bibliography

---

- [97] P. Schuurman and G. Woeginger. Polynomial time approximation algorithms for machine scheduling: Ten open problems. *Journal of Scheduling*, 2(5):203–213, 1999.
- [98] H. D. Sherali and W. P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM J. Discrete Math.*, 3:411–430, 1990.
- [99] H. D. Sherali and W. P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM J. Discrete Math.*, 3(3):411–430, 1990.
- [100] M. Singh. Bellairs workshop on approximation algorithms. Open problem session #1, 2010.
- [101] O. Svensson. Hardness of precedence constrained scheduling on identical machines. *SIAM Journal on Computing*, 40(5):1258–1274, 2011.
- [102] O. Svensson. Hardness of vertex deletion and project scheduling. *Theory Comput.*, 9:759–781, 2013.
- [103] M. Tulsiani. CSP gaps and reductions in the Lasserre hierarchy. In *Proc. STOC 2009*, pages 303–312, 2009.
- [104] J. Ullman. Complexity of sequencing problems. *Computer and Job-Shop Scheduling Theory*, EG Co man, Jr.(ed.), 1976.
- [105] D. Williamson and D. Shmoys. *The design of approximation algorithms*. Cambridge University Press, 2011.
- [106] L. A. Wolsey. Faces for linear inequalities in 0-1 variables. *Math. Program.*, 8:165–178, 1975.
- [107] M. Yannakakis. Expressing combinatorial optimization problems by linear programs. *J. Comput. System Sci.*, 43:441–466, 1991.

## Abbas BAZZI

---

CONTACT INFORMATION	École Polytechnique Fédérale de Lausanne INJ-110, 1015, Lausanne Switzerland	mobile: +41 78 640 49 66 e-mail: abbas.bazzi@epfl.ch
EDUCATION	<b>ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, LAUSANNE, SWITZERLAND</b> <i>Computer, communication and information sciences - PhD</i> <b>September 2013 – Present</b> <ul style="list-style-type: none"><li>• <b>Expected Graduation Date:</b> October 2017</li></ul> <b>AMERICAN UNIVERSITY OF BEIRUT, BEIRUT, LEBANON</b> <i>Electrical and Computer Engineering - MS</i> <b>September 2012 – July 2013</b> <ul style="list-style-type: none"><li>• <b>GPA:</b> 4.0</li><li>• I dropped out of the MS program at AUB before earning the degree to join the PhD program at EPFL.</li></ul> <b>AMERICAN UNIVERSITY OF BEIRUT, BEIRUT, LEBANON</b> <i>Electrical and Computer Engineering - BE</i> <b>October 2008 – June 2012</b> <ul style="list-style-type: none"><li>• <b>Core Courses GPA:</b> 4.0</li></ul>	
PROFESSIONAL INDUSTRY EXPERIENCE	<b>GOOGLE, NYC, UNITED STATES</b> <b>SWE Intern, Geo Data</b> <b>June 2016 – September 2016</b> Worked on training a machine learning model to detect business closure from their corresponding websites.  <b>VELOXENT, BEIRUT, LEBANON</b> <b>Systems Engineer, PHY layer</b> <b>june 2012 – April 2013</b> Designed and implemented a low-power transmitter operating on the 802.11b standard. I also helped in the design of the receiver's side.  <b>TENSORCOM, CARLSBAD, CA, UNITED STATES</b> <b>Systems Engineer, PHY layer</b> <b>june 2011 – September 2011</b> Worked on the implementation of (mainly transmitter's) blocks that are used in 802.11 b/g wifi transmitter as well as the WiGig standard.	
PROFESSIONAL ACADEMIC EXPERIENCE	<b>UNIVERSITY OF WASHINGTON, SEATTLE, WASHINGTON</b> <b>Visiting Researcher</b> <b>November 2016 – May 2017</b> I worked on a research project with my host Prof. Thomas Rothvoß on different open problems in (hyper)graph theory, and scheduling theory.  <b>ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, LAUSANNE, SWITZERLAND</b> <b>Research Assistant</b> <b>October 2013 – Present</b> Worked with my advisor Prof. Ola Svensson on different Scheduling and Allocation problems, as well as the tools used to show the hardness of approximation of certain NP-Complete problems, in different computational models.  <b>AMERICAN UNIVERSITY OF BEIRUT, BEIRUT, LEBANON</b> <b>Research Assistant</b> <b>October 2012 – July 2013</b> Worked on a research project with Prof. Ibrahim Abou Faycal to assess the performance of mismatched decoding when an ISI channel is used under timing phase uncertainty.	
PUBLICATIONS	<b>A. Bazzi, S. Fiorini, S. Huang, O. Svensson.</b> <i>Small Extended Formulation for Knapsack Cover Inequalities from Monotone Circuits.</i> ACM-SIAM Symposium on Discrete Algorithms (SODA), 2017.	

A. Norouzi-Fard, **A. Bazzi**, I. Bogunovic, M. El-Halabi, Y. Hsieh, V. Cevher. *An Efficient Streaming Algorithm for the Submodular Cover Problem*. 30<sup>th</sup> Neural Information Processing Systems (**NIPS**), 2016.

**A. Bazzi**, S. Fiorini, S. Pokutta, O. Svensson. *No Small Linear Program Approximates Vertex Cover within a Factor  $2 - \epsilon$* . 56th Annual Symposium on Foundations of Computer Science (**FOCS**), 2015

**A. Bazzi**, A. Norouzi-Fard. *Towards Tight Lower Bounds for Scheduling Problem*. 23rd Annual European Symposium on Algorithms (**ESA**), 2015

TEACHING  
EXPERIENCE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, LAUSANNE, SWITZERLAND

**Teaching assistant** **Spring 2014, 2015 & 2016**

Teaching assistant for the (master-level) course *Topics in Theoretical Computer Science*.

**Teaching assistant** **Fall 2014 & 2015**

Teaching assistant for the (bachelor-level) course *Algorithms*.

AMERICAN UNIVERSITY OF BEIRUT, BEIRUT, LEBANON

**Lab Instructor** **October 2012 – June 2013**

Lab Instructor for the *Introduction to Programming* course.

**Lab assistant** **Fall 2009-2010 & Spring 2011-2012**

Lab assistant for the *Introduction to Programming* course. I was also a problem setter for the solving sessions.

**Teaching assistant** **Spring 2009-2010 & Spring 2011-2012**

Teaching assistant for the *Data Structures and Algorithms* course and the *Operating Systems* course respectively.

AWARDS &  
HONORS

- I wrote and acquired a Swiss National Foundation project funding for a six month visit to UW, totalling 25700CHF, for the following scientific project: "Approximate Formulation Complexity of NP-hard Optimization Problems", 2016.
- Ranked second in the second ACM Lebanese Collegiate Programming Contest, LAU June 2010.
- Dean's Honor List at AUB for the fall 2011-2012, spring 2011-2012.

LANGUAGES

Fluent in English and Arabic. Advanced knowledge in French.

REFERENCES

*Available upon request*





