Approximability of Some Classical Graph and Scheduling Problems

Doctoral Dissertation submitted to the
Faculty of Informatics of the University of Lugano
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by
Ola Nils Anders Svensson

under the supervision of
Dr. Monaldo Mastrolilli

June 2009

Dissertation Committee

Prof. Dr. Luca Maria Gambardella	IDSIA and University of Lugano, Switzerland
Prof. Dr. Klaus Jansen	Christian-Albrechts-Universität zu Kiel, Germany
Dr. Monaldo Mastrolilli	IDSIA, Switzerland
Prof. Dr. Andreas S. Schulz	Massachusetts Institute of Technology, USA
Prof. Dr. Thomas Shrimpton	University of Lugano, Switzerland

Dissertation accepted on 10 June 2009

Supervisor	PhD program director
Dr. Monaldo Mastrolilli	**Prof. Dr. Fabio Crestani**

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Ola Nils Anders Svensson
Lugano, 10 June 2009

Abstract

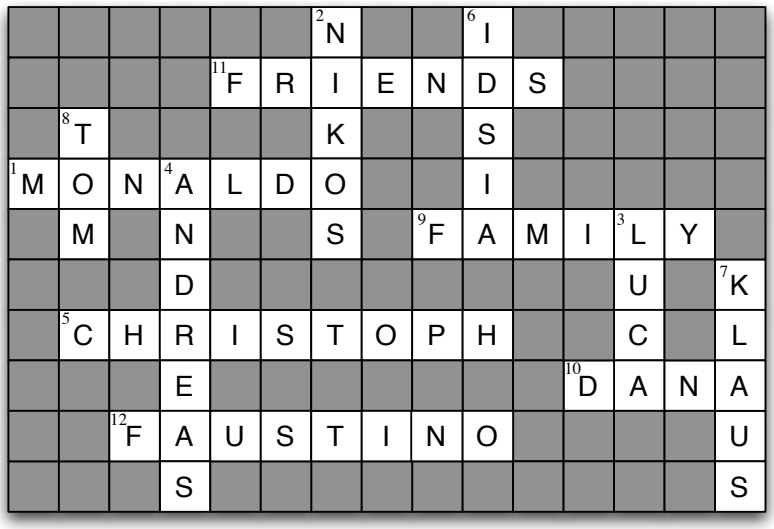
Approximation algorithms are motivated by the fact that for many important optimization problems we cannot hope to efficiently find an optimal solution (unless P=NP). Instead, we have to settle for second best — a solution that is close to being optimal. A natural question that arises is: how close to the optimal solution can one get with an efficient algorithm? The past two decades have seen major progress in answering this question for several fundamental optimization problems, including clique, set-cover, and graph coloring. In spite of this progress, our understanding of approximability for some classes of problems has remained weak. In this thesis we address several prominent open problems in the area of approximation algorithms for two such classes, namely machine scheduling and certain graph problems.

The first part of the thesis is devoted to the study of the classical precedence-constrained single machine scheduling problem with the weighted sum of completion times objective. By exploiting the scheduling problem's relationship to partial orders and vertex cover, we present a framework that achieves a better approximation guarantee as soon as the precedence constraints have low complexity (i.e. low dimension). We also complement these positive results by giving the first inapproximability result for the scheduling problem together with evidences that the various 2-approximation algorithms might be tight.

In the second part, we focus on the uniform sparsest cut and optimal linear arrangement graph problems. These classical graph problems are typical cases of problems for which neither a hardness of approximation result, nor a 'good' approximation algorithm exists. We use a recent so-called Quasi-random PCP construction to give the first hardness of approximation results for these problems that rule out the existence of a polynomial time approximation scheme for each of these problems.

We conclude the thesis by considering two notorious scheduling problems in shop environments, namely the job shop problem together with the more restricted flow shop problem. We close a major open problem in scheduling theory by providing stronger inapproximability results for these problems. More precisely, we give a gap-preserving reduction from graph coloring that gives an inapproximability result that matches the best known approximation algorithm for the general version of flow shops, where jobs are not required to be processed on each machine. Our result is also tight for the more general acyclic job shop problem and gives the first non-constant hardness of approximation result for the job shop problem.

Acknowledgements



ACROSS

1. Co-author, supervisor, and friend at IDSIA that has made this work possible.
5. Co-author with his feet on the ground and a special taste for good food and relaxing boat rides.
9. Persons that give unconditional love and support.
10. Great girl friend that tolerates approximate cleaning.
11. I have been fortunate with a lot of good ... (you know who you are).
12. His comments have significantly improved the writing in this thesis.

DOWN

2. Co-author and friend at IDSIA that cooks delicious Greek food.
3. Cheerful director and boss that is always ready to help.
4. Committee member, my host at MIT, and a good ping pong player that has had a great influence on this thesis.
6. An institute with great people and a friendly atmosphere.
7. Committee member and supervisor of the supervisor.
8. Committee member with the reputation to "look like a movie star and think like a professor".

Credits

Most of the results in this thesis have appeared previously in different forms.

- Chapter 3 is based on joint work with Christoph Ambühl, Monaldo Mastrolilli, and Nikolaus Mutsanas. Different parts of this chapter have previously appeared in [AMS06; AMMS07; AMS07; AMMS08].
- Chapter 4 is based on joint work with Christoph Ambühl and Monaldo Mastrolilli. A preliminary version appeared in [AMS07].
- Chapter 5 is based on joint work with Monaldo Mastrolilli. Our results regarding the job shop problem have previously appeared in [MS08]. Our results for flow shop scheduling were obtained recently and will appear in [MS09].

I had also the pleasure to discuss many of the topics covered in this thesis with many prominent researchers. Apart from my co-authors, these include Klaus Jansen, Maurice Queyranne, Andreas S. Schulz, Maxim Sviridenko, and Nelson Uhan. Also a special thanks to my director Luca who has given me the freedom to work freely on topics of my interest.

Finally, this work would not have been possible without the generous support from the Swiss National Science Foundation projects 200020-109854, "Approximation Algorithms for Machine scheduling Through Theory and Experiments II", and PBTI2-120966, "Scheduling with Precedence Constraints".

Contents

Chapter 1

Introduction

Picture yourself in the following situation. You and your rivals are competing for the precious prize of one million dollars. A big bowl full of coins is presented for exactly one minute, and the winner will be selected according to who gives the closest estimate of the number of coins. As the bowl is presented during only one minute, you do not have enough time to count each coin. Instead, you have to give a qualified guess — an approximation — of the number of coins. In order to win, it is crucial to give a good approximation. For example, if the bowl contains 1000 coins and you give a factor 3 approximation (you guess that the bowl contains $3 \cdot 1000$ coins), then the risk that one of your rivals gives a better approximation is overwhelming.

In computer science we face a similar dilemma. Many of the computational problems that arise in practice are NP-hard optimization problems. Assuming P≠NP[1], the NP-hardness of an optimization problem says that there is no efficient (polynomial time) algorithm that finds an optimal solution. Instead, one can only hope to efficiently find a good approximate solution — one that is not optimal, but is within a small factor away from optimal. A natural question that arises is how close to the optimal solution one can get with an efficient algorithm. In this thesis we address this question for some classical scheduling problems and the notorious uniform sparsest cut and optimal linear arrangement graph problems.

After briefly describing the methodology for analyzing the approximability of NP-hard optimization problems, we give an overview of our results followed by a short outline of this thesis.

[1]Whether P=NP or P≠NP is regarded as one of the most important open problems in mathematics and theoretical computer science; its resolution is worth one million dollars! Informally, it is the question whether it is easier to verify a solution than finding one. Most people believe that the latter is harder than the former, i.e., P≠NP.

1.1 Approximability of NP-hard Optimization Problems

In the field of approximation theory we are interested in understanding the approximability of NP-hard optimization problems. More precisely, given an NP-hard optimization problem, we would like to, on the one hand, provide an efficient (polynomial time) algorithm that is guaranteed to find a solution within a small factor β of optimal. We refer to such an algorithm as a β-approximation algorithm. On the other hand, we would like to know our limits by giving a so-called inapproximability or hardness of approximation result. This is a result stating that the problem is not efficiently approximable within some factor α. Invariably, such a result is based on a plausible complexity theoretic assumption, the weakest possible being P$\neq$ NP since if P=NP, all considered problems can be solved to optimality efficiently (in polynomial time).

The ultimate goal is to have a tight result, i.e., a result so that the bound β given by the best approximation algorithm matches the bound α given by the strongest inapproximability result. When this is the case, we fully understand the approximability of the addressed problem.

The motivation for the first type of results, also called *positive results*, is obvious: since we cannot hope to find an *exact* solution efficiently for all instances of our problem (unless P=NP) we have to settle for second best — a solution that is close to optimal. The second type of results, also called *negative results*, might seem harder to motivate. Why are we interested in knowing that we cannot efficiently find a good solution? There are many answers to this question. Here, we include three that we find especially convincing.

1. A negative result that matches the best known positive result prevents hours of hard (fruitless) work trying to improve on the positive result.

2. A negative result often gives insights as to what is precisely the difficulty of the addressed problem. After identifying that difficulty, we can restrict our attention to an easier but often relevant "sub"-problem that allows for better approximation.

3. Establishing that a problem is hard can have unexpected and useful consequences. For example, cryptography is based on the hardness of some problems.

As described above, the process of determining the approximability of an NP-hard optimization problem involves both positive and negative results. One of the first positive results can be traced back to 1966, when Graham [Gra66] analyzed a simple procedure, called list scheduling, for one of the most basic scheduling problems: minimizing makespan in an identical parallel machine

environment. Since then a large amount of work has been devoted to finding efficient approximation algorithms for a variety of optimization problems (see e.g. [ACG+99] and [Vaz01]). Nevertheless, our understanding of their approximability has, until recently, been impeded by the lack of techniques for proving negative results.

A turning point was the discovery of the now famous PCP Theorem [ALM+98; AS98] and its connection to negative results [FGL+96]. The PCP Theorem has had a similar impact on the theory of approximation as the establishment of a natural NP-hard problem had for analyzing exact problems [Coo71; Kar72]. Indeed, this theorem implies that the fundamental optimization problem MAX-3SAT is NP-hard to approximate within a small constant factor. Once this was established, negative results were subsequently obtained by "translating" the hardness of MAX-3SAT to other problems. In particular, negative results that match the best known positive results have been obtained for several fundamental NP-hard optimization problems, including MAX-3SAT [Hås01], set cover [Fei98], clique [Hås99], and graph coloring [FK98].

1.2 Motivation of this Thesis

Despite our increased understanding of the approximability of NP-hard optimization problems, today's techniques seem insufficient for obtaining tight or even nearly tight results for some classes of problems. Scheduling problems form one of those classes, where tight results are regarded as exceptional. In [SW99], Woeginger & Schuurman highlighted this point by listing ten of "the most outstanding open problems in the approximation of scheduling problems". Understanding their approximability is not only interesting from a theoretical point of view, it is also motivated by the fact that these classical scheduling problems arise frequently when analyzing more complex problems that occur in practice. Examples include problems from traffic planning, manufacturing, transportation, packet routing, etc.

The objective of this thesis is to improve our understanding of the approximability of these scheduling problems. While addressing these scheduling problems, we discovered a connection to a graph problem called maximum edge biclique. Interestingly, this graph problem is part of another class of problems, formed by graph problems that have a similar status as the scheduling problems, i.e., their approximability is not well understood. Other graph problems in this class include the notorious sparsest cut problem and the optimal linear arrangement problem. As these problems often appear as "sub"-problems when analyzing other problems, understanding their approximability is another major open problem in the theory of approximation (see e.g. [Vaz01; Tre04; DKSV06]).

1.3 Overview of our Contributions

The main contributions of this thesis regard three classical scheduling problems together with the uniform sparsest cut and optimal linear arrangement graph problems.

Single Machine Scheduling with Precedence Constraints

In Chapter 3, we study the classic precedence-constrained single machine scheduling problem with the weighted sum of completion times objective. Improving on the various known 2-approximation algorithms (or proving that none exists) is considered one of the most prominent open problems in scheduling theory (see e.g. "Open Problem 9" in [SW99]).

By exploiting the scheduling problem's relationship to partial orders and its close relationship to vertex cover [CS05; AM09], we present a framework that achieves a better approximation guarantee as soon as the precedence constraints have low complexity, where the complexity of the precedence constraints are measured by their dimension. This general framework gives the best known approximation ratios for all considered special cases of precedence constraints.

We also establish a connection between the scheduling problem and a graph problem called maximum edge biclique. This connection allows us to present the first hardness of approximation result for the scheduling problem. More precisely, we rule out the possibility of having an arbitrarily good approximation algorithm — a polynomial time approximation scheme (PTAS[2]) — for the addressed problem. Finally, we show that if we disregard the so-called "fixed-cost" always present in a feasible schedule, then the addressed scheduling problem is as hard to approximate as vertex cover. This gives further evidence that the various 2-approximation algorithms might be tight, since vertex cover is strongly conjectured to be NP-hard to approximate within a factor $(2-\epsilon)$ for any constant $\epsilon > 0$.

Maximum Edge Biclique, Optimal Linear Arrangement, and Sparsest Cut

Motivated by the connection to precedence-constrained single machine scheduling, we focus in Chapter 4 on the approximability of maximum edge biclique and the two related optimal linear arrangement and sparsest cut graph problems.

Optimal linear arrangement and (uniform) sparsest cut are typical cases of classical graph problems for which we have neither a hardness of approximation result, nor a 'good' approximation algorithm (they all have a worse than constant approximation guarantee). We give the first hardness of approximation re-

[2]For a formal definition of a PTAS, we refer the reader to Section 2.1.

sults for these problems by showing that they have no PTAS, unless NP-complete problems can be solved in randomized subexponential time. Our results use the recent Quasi-random PCP construction of Khot, who proved important inapproximability results for graph min-bisection, densest subgraph, and bipartite clique [Kho06]. The Quasi-random PCP construction, or even the stronger average-case assumptions recently used by Feige [Fei02], was not known to generalize to the uniform sparsest cut problem and the optimal linear arrangement problem (see e.g. [Tre04]) prior to this work.

Job and Flow Shops

Finally in Chapter 5, we consider two scheduling problems in shop environments, namely the classical job shop problem and the more restricted flow shop problem. The study of these problems can be traced back to the late 50's and is motivated by several natural applications, including packet routing and multistage production optimization. Even though the best approximation algorithms for both problems have worse than logarithmic performance guarantee, the only previously known inapproximability result says that they are NP-hard to approximate within a factor less than $5/4$ [WHH$^+$97].

For job shops, we provide stronger inapproximability results. We show that it is NP-hard to approximate job shops within any constant factor. Moreover, under a stronger assumption, we obtain a hardness result that matches (up to smaller terms) the best known approximation algorithm for acyclic job shops. When the number of machines *and* the number of operations per job are assumed to be constant the job shop problem is known to admit a PTAS [JSOS03]. We conclude by showing that both these restrictions are indeed necessary to obtain a PTAS.

Our inapproximability results for job shops do not apply to flow shops, whose approximability remains poorly understood. The current algorithm of choice for flow shops is Feige & Scheideler's approximation algorithm for acyclic job shops [FS02] which (roughly) guarantees a schedule with makespan at most a logarithmic factor away from a trivial lower bound on the optimal makespan. As flow shops are more structured than acyclic job shops, they raised as an open question if the approximation guarantee for flow shop scheduling can be improved significantly. We resolve this question (negatively) by providing flow shop instances with optimal makespan a logarithmic factor away from the trivial lower bound used by Feige & Scheideler [FS02]. By combining these new gap instances with our techniques for proving hardness of approximation for job shops, we also show that the current approximation algorithm is tight (up to smaller terms) for the more general variant of the flow shop problem, where jobs are not required to be processed on each machine.

1.4 How to Read this Thesis

In Chapter 2, we introduce the basic definitions and methods used when analyzing the approximability of NP-hard optimization problems. For other essential definitions regarding graphs, optimization problems, and complexity classes, we refer the reader to Appendix A, where we also give pointers to useful references.

In Chapters 3, 4 and 5, we present the contributions of this thesis. Each of these chapters starts with the definition(s) of the addressed problem(s), a review of the relevant literature, followed by a statement of the contributions together with an overview of the proof techniques. The formal proofs are given in the main body of the chapter. We have made an effort to present the proofs in order of increasing difficulty, with the hope that the reader will become familiar with the basic intuitions, before advancing to the more involved proofs. The main implications are then summarized in the chapter specific conclusions, where we also list some of our favorite open problems. In summary, these chapters are designed to be self-contained to allow the reader to jump directly to the part of most interest.

In the last part of this thesis, we discuss the contributions in a wider context together with future research directions we find interesting.

Chapter 2

Approximation: Coping with NP-Hardness

When confronted with an NP-hard optimization problem, we cannot hope to solve all instances to optimality in polynomial time. Instead we can hope to efficiently find an approximate solution – one that is not optimal, but is within a factor α of optimal. To understand how close to the optimal solution we can get in polynomial time, we need to provide both so-called positive and negative results. In Sections 2.1 and 2.2, we review some of the basic definitions and methods regarding positive and negative results.

2.1 Positive Results — Approximation Algorithms

We assume some familiarity with optimization problems, see e.g. [Vaz01] for an introduction. Let Π be an optimization problem. Given an instance I of Π, we let $OPT(I)$ denote the value of an optimal solution to instance I. We denote the value of another solution S to instance I by $\text{val}(S)$. For the considered optimization problems we have that both $OPT(I)$ and $\text{val}(S)$ are strictly positive. We say that S is an *α-approximate solution* to I if

$$\frac{\text{val}(S)}{OPT(I)} \leq \alpha \qquad \text{if } \Pi \text{ is a minimization problem;}$$

$$\frac{\text{val}(S)}{OPT(I)} \geq \alpha \qquad \text{if } \Pi \text{ is a maximization problem.}$$

An algorithm $\mathscr{A}$ is said to be an *α-approximation algorithm* for Π if, on each instance, $\mathscr{A}$ produces an α-approximate solution, and the running time of $\mathscr{A}$ is bounded by a polynomial in the instance size. The factor α is called the

(worst-case) *performance guarantee*, *approximation ratio* or *factor* of the algorithm. Note that the performance guarantee is at least 1 and at most 1 for minimization problems and maximization problems, respectively; an algorithm with performance guarantee 1 always finds an optimal solution.

Some NP-hard optimization problems allow approximability to any desired degree. Formally, we say that an algorithm $\mathscr{A}$ is a *polynomial time approximation scheme* (PTAS) for a minimization problem Π if, on each instance and for each fixed $\epsilon > 0$, $\mathscr{A}$ produces an $(1+\epsilon)$-approximate solution and the running time of $\mathscr{A}$, for a fixed ϵ, is bounded by a polynomial in the instance size. Note that the definition above allows the running time of the algorithm to depend arbitrarily on ϵ. If the running time of $\mathscr{A}$ is bounded by a polynomial in the instance size and $1/\epsilon$, then we say that $\mathscr{A}$ is a *fully* polynomial time approximation scheme (FPTAS) for problem Π. For example, on the one hand if $\mathscr{A}$ runs in time $O(n^{1/\epsilon})$ on an instance of size n then $\mathscr{A}$ is a PTAS but not an FPTAS. On the other hand, if $\mathscr{A}$ runs in time $O\left(n^4(1/\epsilon)^{100}\right)$ then $\mathscr{A}$ is an FPTAS.

The definitions of PTAS and FPTAS for maximization problems are obtained by substituting $(1+\epsilon)$ with $(1-\epsilon)$, $0 < \epsilon < 1$, in the above definitions.

2.1.1 Analyzing the Performance Guarantee

Consider a minimization problem Π. In order to establish the performance guarantee of an approximation algorithm $\mathscr{A}$ for Π, the cost of the solution produced by $\mathscr{A}$, on an instance I, needs to be compared with the value of an optimal solution, i.e., $OPT(I)$. However, calculating $OPT(I)$ is NP-hard for problems that we are interested in approximating. Instead, we have to settle for a lower bound $LB(I)$ on $OPT(I)$ [1]. The performance guarantee of $\mathscr{A}$ can then be measured with respect to $LB(I)$. It follows that $\mathscr{A}$ is an α-approximation algorithm if, on each instance I, $\mathscr{A}$ produces a solution S so that

$$\frac{\text{val}(S)}{OPT(I)} \leq \frac{\text{val}(S)}{LB(I)} \leq \alpha.$$

As the analyzed performance guarantee will depend on $LB(I)$, the task of achieving a good lower bound on the value of an optimal solution is crucial in the design of approximation algorithms. A powerful and popular method for this purpose, that we use in Chapter 3 and give an example of in the next section, is to first state the addressed problem as an integer program and then lower bound the value of an optimal solution, by solving the linear relaxation of this program.

[1] The issue described here also arises when considering a maximization problem. In that case, we need an *upper* bound on the value of an optimal solution to be able to analyze the performance guarantee of an approximation algorithm.

2.1.2 An Example of a Positive Result

As an example of a positive result we present the famous 2-approximation algorithm for minimum weighted vertex cover using linear programming. In the weighted vertex cover problem we are given a graph $G = (V, E)$ where each vertex $v \in V$ has a nonnegative weight w_v. A vertex cover is a subset $V' \subseteq V$ of vertices so that for each edge $\{u, v\} \in E$, at least one of u and v belongs to V'. *Minimum weighted vertex cover* is then the problem of finding a vertex cover $V' \subseteq V$ so as to minimize $w(V') = \sum_{v \in V'} w_v$. See Figure 2.1 for a small example.

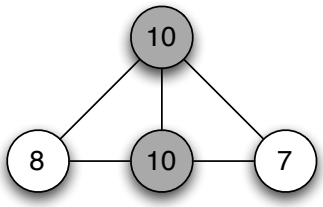


Figure 2.1: An example of a small weighted vertex cover instance. The vertices depicted in gray form a minimum vertex cover of value 20.

In 1973, Nemhauser & Trotter [NT73; NT75] used the following integer program to model the weighted vertex cover problem:

$$\begin{array}{lllll} \text{[VC-IP]} & \text{minimize} & \displaystyle\sum_{v \in V} w_v x_v & & \\ & \text{subject to} & x_u + x_v \geq 1 & \{u, v\} \in E, & (2.1) \\ & & x_v \in \{0, 1\} & v \in V. & \end{array}$$

It is not hard to see that the above integer program is equivalent to the weighted vertex cover problem. On the one hand, given a weighted vertex cover solution $V' \subseteq V$ we can define a feasible solution x^* to [VC-IP] with the same cost by setting $x^*_v = 1$ if and only if $v \in V'$. On the other hand, given a solution x^* to [VC-IP], the set $V' = \{v : x^*_v = 1\}$ is a feasible vertex cover with $w(V') = \sum_{v \in V} w_v x^*_v$. As [VC-IP] exactly models the weighted vertex cover problem, it is NP-hard and we cannot hope to find an optimal solution in polynomial time. Instead, we can relax the integrality constraint to obtain a linear program that can be solved in polynomial time.

The linear programming relaxation [VC-LP] of [VC-IP] is the linear program with the same objective and constraints as [VC-IP] with the exception that the integrality is relaxed, i.e., the constraint $x_v \in \{0, 1\}$ is replaced by the constraint

$0 \leq x_v \leq 1$. The linear program [VC-LP] is a relaxation of [VC-IP] (and the vertex cover problem) since each feasible solution to [VC-IP] is also a solution to [VC-LP], but the reverse is not true: setting all variables to $1/2$ is a feasible solution to [VC-LP] but not to [VC-IP].

However, we can obtain an approximate solution to [VC-IP] (and weighted vertex cover) by the following procedure.

1. Find an optimal solution x^{LP} to [VC-LP] in polynomial time.
2. Round x^{LP} to obtain a solution x^* to [VC-IP]. More specifically, set x_v^* to one if $x_v^{LP} \geq 1/2$, and to zero otherwise.

To show that the above procedure gives a 2-approximation algorithm for the weighted vertex cover problem, we need to prove that x^* is indeed a feasible solution to [VC-IP] and that $\sum_{v \in V} w_v x_v^* \leq 2 \cdot OPT$, where OPT is the value of an optimal solution to [VC-IP]. That x^* is binary is clear from the rounding. Now suppose toward contradiction that one of the constraints (2.1) is violated. Then there are two adjacent vertices u and v of G such that $x_u^* = 0$ and $x_v^* = 0$. By the way we rounded the solution this implies that $x_u^{LP} < 1/2$ and $x_v^{LP} < 1/2$ which contradicts the feasibility of x^{LP}, because the constraint $x_u + x_v \geq 1$ would be violated. Hence, x^* is a feasible solution to [VC-IP].

As [VC-LP] is a relaxation of [VC-IP], the solution x^{LP} is a lower bound on OPT. The performance guarantee of 2 now follows from

$$\sum_{v \in V} w_v x_v^* \leq 2 \cdot \sum_{v \in V} w_v x_v^{LP} \leq 2 \cdot OPT,$$

where the first inequality holds since for each $v \in V$ we have $x_v^* \leq 2 \cdot x_v^{LP}$.

2.2 Negative Results — Hardness of Approximation

A *negative result* (also called *hardness of approximation* or *inapproximability result*) is a result that states that a problem has no approximation algorithm with performance guarantee α. These results are invariably based on plausible complexity theoretic assumptions, the weakest possible being P≠NP since if P=NP, all considered problems can be solved exactly in polynomial time. For a list of complexity classes used in the assumptions in this thesis see Appendix A.3.

2.2.1 Gap-Introducing and Gap-Preserving Reductions

We start by recalling the methodology for establishing that an exact optimization problem is NP-hard. Suppose we would like to prove that it is NP-hard to

compute an optimal solution to a maximization problem Π_{MAX}. The traditional way to do so is to first select an NP-complete decision problem, say satisfiability (SAT), and then provide a polynomial time reduction from SAT to Π_{MAX}. The reduction maps an instance ϕ of SAT to an instance I of Π_{MAX} so that

- *(Completeness)* if ϕ is satisfiable then $OPT(I) \geq f(I)$, and
- *(Soundness)* if ϕ is not satisfiable then $OPT(I) < f(I)$,

where f is a function that comes with the reduction and has the instance as argument. Note that such a reduction does not yield any inapproximability result: it only says that it is hard to find an optimal solution but it leaves open the possibility to find a solution arbitrarily close to the optimal, i.e., a solution with value $(1-\epsilon) \cdot OPT(I)$ for an arbitrarily small $\epsilon > 0$.

Instead, we need more powerful reductions to establish hardness of approximation results. There are mainly two types of such reductions: gap-introducing and gap-preserving reductions.

Gap-Introducing Reductions

The formal definition of a gap-introducing reduction differs slightly for maximization and minimization problems. We give the definition for maximization problems; the definition for minimization problems is similar and left to the reader. A *gap-introducing reduction* from SAT to a maximization problem Π_{MAX} comes with an additional function α (apart from f also present in the exact reduction presented above). Given an instance ϕ of SAT, it computes, in polynomial time, an instance I of Π_{MAX} of size n, such that

- *(Completeness)* if ϕ is satisfiable then $OPT(I) \geq f(I)$, and
- *(Soundness)* if ϕ is not satisfiable then $OPT(I) < \alpha(n) \cdot f(I)$.

Notice that α is a function of the instance size and since Π_{MAX} is a maximization problem $\alpha(n) \leq 1$. Moreover, from the gap-introducing reduction above, it is not hard to conclude that Π_{max} has no $\alpha(n)$-approximation algorithm on an instance of size n, unless P=NP (see Corollary 2.2.2 for an example).

Any hardness of approximation result is ultimately based on a gap-introducing reduction. For many years there was limited progress in analyzing the approximability of NP-hard optimization problems, due to the lack of "good" gap-introducing reductions. A major breakthrough was the discovery of the PCP Theorem [AS98; ALM$^+$98] and its connection to inapproximability [FGL$^+$96]. The PCP Theorem can be seen as a (very complicated) gap-introducing reduction stated as follows:

Theorem 2.2.1 *There is a constant $\epsilon > 0$ and a polynomial time reduction that takes as input an instance ϕ of SAT and outputs an instance ψ of MAX-3SAT[2] with m clauses so that*

- (Completeness) *if ϕ is satisfiable then $OPT(\psi) = m$, and*
- (Soundness) *if ϕ is not satisfiable then $OPT(\psi) < (1-\epsilon)m$.*

Note that the reduction introduces a gap of factor $(1-\epsilon)$ in the number of ψ's clauses an optimal assignment satisfies, depending on whether the original SAT formula ϕ was satisfiable. As a direct consequence, we have the following corollary.

Corollary 2.2.2 *There is no $(1-\epsilon)$-approximation algorithm for MAX-3SAT, unless* P=NP.

Proof. Suppose MAX-3SAT has an $(1-\epsilon)$-approximation algorithm $\mathscr{A}$. We will show that $\mathscr{A}$ solves SAT in polynomial time. Given an instance ϕ of SAT we apply the reduction given in Theorem 2.2.1 to obtain an instance ψ of MAX-3SAT. Let val(S) be the value of the solution S returned by $\mathscr{A}$ on instance ψ. Then we have the following:

- if ϕ is satisfiable then $val(S) \geq (1-\epsilon) \cdot OPT(\psi) \geq (1-\epsilon) \cdot m$, and
- if ϕ is not satisfiable then $val(S) \leq OPT(\psi) < (1-\epsilon)m$.

We can thus use $\mathscr{A}$ to distinguish between satisfiable and unsatisfiable SAT instances. It follows that no such approximation algorithm exists, unless P=NP. □

Gap-Preserving Reductions

The negative results in this thesis are obtained by so called gap-preserving reductions. The idea is as follows. Once we have obtained a hardness result for an optimization problem, as we did in the previous section for MAX-3SAT, we can prove a hardness result for another optimization problem Π, by giving a special reduction, called gap-preserving reduction, from MAX-3SAT to Π so that the hardness (gap) of MAX-3SAT translates to a (potentially different) gap for Π.

The formal definition is slightly different for maximization and minimization problems. We present the formal definition, as given in [Vaz01], for the case

[2]Recall that MAX-3SAT is the following problem: given a 3-CNF formula ψ (i.e. with at most 3 literals per clause), find an assignment that satisfies the largest number of clauses.

when one reduces from a maximization problem Π_1 to a minimization problem Π_2. The three remaining cases where Π_1 and Π_2 are maximization or minimization problems are similar and left to the reader. To clarify the concept of gap-preserving reductions we also present a simple example in Section 2.2.2.

A *gap-preserving reduction*, from a maximization problem Π_1 to a minimization problem Π_2, comes with four parameters (functions), f_1, α, f_2, and β. Given an instance I_1 of Π_1 of size n_1, it computes, in polynomial time, an instance I_2 of Π_2 of size n_2 such that

- *(Completeness)* if $OPT(I_1) \geq f_1(I_1)$ then $OPT(I_2) \leq f_2(I_2)$,
- *(Soundness)* if $OPT(I_1) < \alpha(n_1) f_1(I_1)$ then $OPT(I_2) > \beta(n_2) f_2(I_2)$.

In keeping with the fact that Π_1 is a maximization problem and Π_2 is a minimization problem, $\alpha(n_1) \leq 1$ and $\beta(n_2) \geq 1$. Furthermore, by a similar argument as in the proof of Corollary 2.2.2, if it is NP-hard to distinguish between the bounds, in the completeness case and soundness case, on the value of an optimal solution to the instance of Π_1, then there is no $\beta(n_2)$-approximation algorithm for problem Π_2 on instances of size n_2, unless P=NP.

2.2.2 An Example of a Negative Result

As an example of a gap-preserving reduction we reduce MAX-3SAT to the (unweighted) vertex cover problem (see Section 2.1.2 for a definition). We remark that this is a classical example and can also be found, for example, in the book by Vazirani [Vaz01].

Given an instance ϕ of MAX-3SAT with m clauses we construct a vertex cover instance G with $3m$ vertices so that

- *(Completeness)* if $OPT(\phi) \geq m$ then $OPT(G) \leq 2 \cdot m$,
- *(Soundness)* if $OPT(\phi) < (1-\epsilon)m$ then $OPT(G) > (2+\epsilon) \cdot m$.

As Theorem 2.2.1 says that there is a constant $\epsilon > 0$ so that it is NP-hard to distinguish whether $OPT(\phi) = m$ or $OPT(\phi) = (1-\epsilon)m$, the reduction presented here implies that there is no approximation algorithm for the vertex cover problem with performance guarantee $(2+\epsilon)/2 = 1+\epsilon/2$, unless $P \neq NP$. We start by presenting the construction of the vertex cover instance followed by the completeness and soundness analysis.

Construction

Given an instance ϕ of *MAX*-3*SAT* with m clauses $C_1, \ldots, C_m$, we construct a vertex cover instance $G = (V, E)$ as follows (see Figure 2.2 for a small example). Corresponding to each clause of ϕ, G has a clique[3] of three vertices, one vertex for each literal in the clause. Hence $|V| = 3m$. The final graph G is then obtained by adding edges between any two vertices corresponding to literals that are negations of each other. The intuition behind the construction is that each vertex cover defines a partial truth assignment: each literal that corresponds to a vertex *not* in the vertex cover is set to be true. The value of a minimal vertex cover can then be seen to be the number of clauses satisfied by the (partial) truth assignment subtracted from $|V|$.

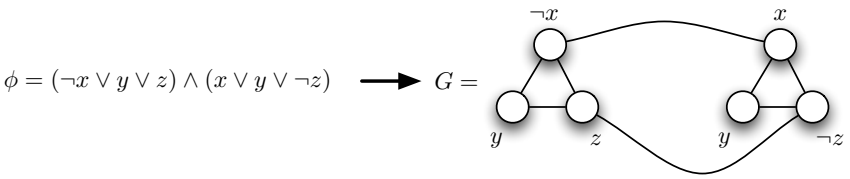


Figure 2.2: An example of the graph G constructed from ϕ.

Completeness and Soundness.

A subset $I \subseteq V$ of the vertices is called an *independent set* if no two vertices in I are adjacent. It is easy to see that vertex covers and independent sets are complements of each other, i.e., if $I \subseteq V$ is an independent set then $V \setminus I$ is a vertex cover, and vice versa, if $VC \subseteq V$ is a vertex cover then $V \setminus VC$ is an independent set. We claim that the size of the maximum independent set of G is exactly $OPT(\phi)$. In the completeness case consider an optimal truth assignment and pick one vertex, corresponding to a satisfied literal, from each satisfied clause. Clearly, the picked vertices form an independent set. Conversely, in the soundness case, consider an independent set I in G, and set the literals corresponding to its vertices to be true. Any extension of this truth setting to all variables will satisfy at least $|I|$ clauses. As the graph has $3m$ vertices and the complement of an independent set forms a vertex cover, we have that in the completeness case, G has a vertex cover of size

$$3m - \underbrace{OPT(\phi)}_{=m} = 2 \cdot m$$

[3] A clique is a set of pairwise adjacent vertices.

whereas in the soundness case, any vertex cover must have size

$$3m - \underbrace{OPT(\phi)}_{\leq(1-\epsilon)m} \geq (2+\epsilon)m,$$

for some $\epsilon > 0$.

Chapter 3

Single Machine Scheduling with Precedence Constraints

3.1 Introduction

We consider the classic precedence-constrained single machine scheduling problem with the weighted sum of completion times objective. In the notation of Graham et al. [GLLK79], the problem is known as $1|prec|\sum w_j C_j$. A set $N = \{1, \ldots, n\}$ of jobs are to be scheduled, without interruptions, on a single machine that can process at most one job at a time. Each job j has a processing time p_j and a weight w_j, where p_j and w_j are non-negative rationals. The precedence constraints are specified in the form of a *partially ordered set* (*poset*) $\mathbf{P} = (N, P)$, consisting of the set of jobs N and a partial order i.e. a reflexive, antisymmetric, and transitive binary relation P on N, where $(i, j) \in P$ $(i \neq j)$ implies that job i must be completed before job j can be started. The goal is to find a feasible schedule, i.e., a total ordering L of the jobs with $P \subseteq L$, so as to minimize $\sum_{j=1}^{n} w_j C_j$, where C_j is the time at which job j completes in the schedule. By rewriting the objective function we can divide it into a *variable-cost* and a *fixed-cost*:

$$\sum_{j=1}^{n} w_j C_j = \sum_{j=1}^{n} w_j \left(p_j + \sum_{(i,j)\in L} p_i \right) = \underbrace{\sum_{(i,j)\in L\setminus P} p_i w_j}_{\text{variable-cost}} + \underbrace{\sum_{(i,j)\in P} p_i w_j + \sum_{i=1}^{n} p_i w_i}_{\text{fixed-cost}}.$$

We note that the fixed-cost is only dependent on the instance and is present in all feasible schedules, whereas the variable-cost depends on the solution. As a result, finding a schedule that minimizes $\sum w_j C_j$ is equivalent to finding a schedule that minimizes the variable-cost. However, as the fixed-cost can always

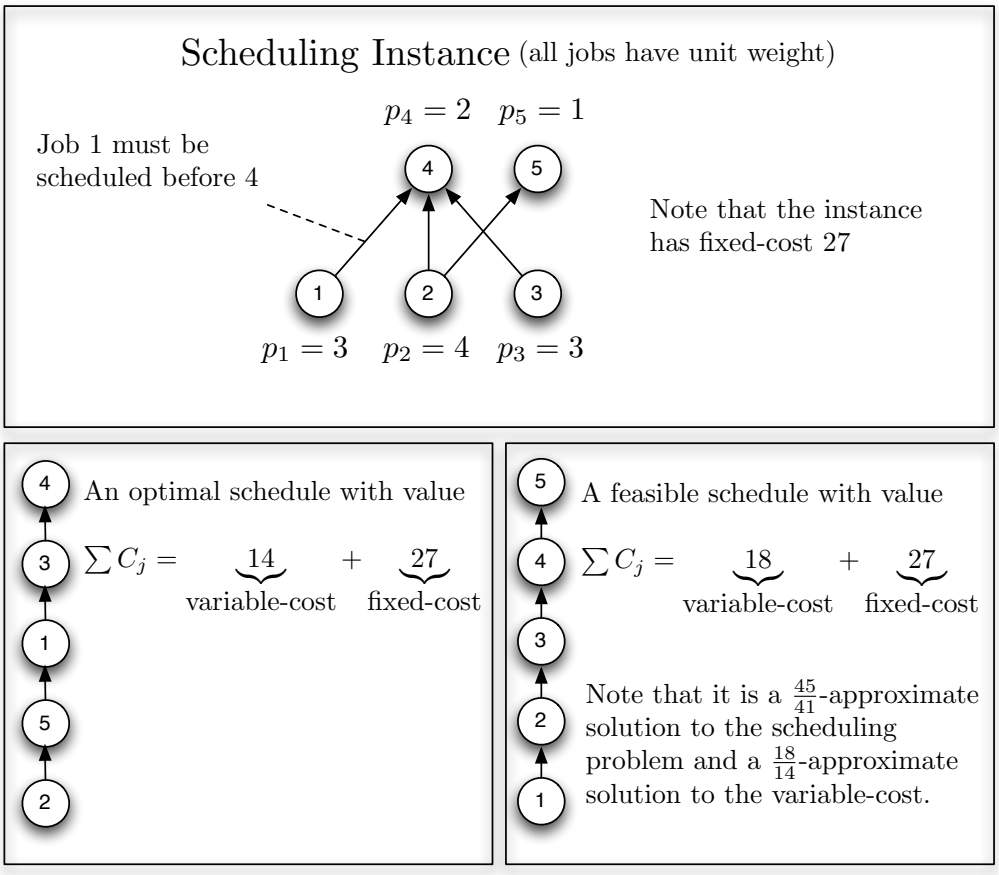


Figure 3.1: On the top we give an example instance of $1|prec|\sum w_j C_j$ (where all jobs have unit weight). An optimal schedule of this instance is depicted on the left and an approximate solution on the right. Note that the fixed-cost improves the approximation ratio: $45/41 < 18/14$.

be included in a lower bound on the value of an optimal schedule, it might help to improve the analysis of approximation algorithms. See Figure 3.1 for an example.

3.1.1 Literature Review

For the general version of $1|prec|\sum w_j C_j$, closing the approximability gap is considered an outstanding open problem in scheduling theory (see e.g. "Open Problem 9" in [SW99]). It was shown to be strongly NP-hard in 1978 by Lawler

[Law78] and Lenstra & Rinnooy Kan [LK78]. While no inapproximability results were known (other than that the problem does not admit a fully polynomial approximation scheme), several 2-approximation algorithms have been proposed [Pis92; Sch96; HSSW97; CM99; CH99; MQW03; Pis03].

Schulz [Sch96] gave 2-approximation algorithms using linear programming relaxations[1]. Chudak & Hochbaum [CH99] gave another algorithm based on a linear programming relaxation with two variables per constraint. Furthermore, they showed that their linear programming relaxation can be solved using one min-cut computation, making their algorithm combinatorial. Independently, Chekuri & Motwani [CM99] and Margot, Queyranne & Wang [MQW03], provided identical, simple combinatorial 2-approximation algorithms based on Sidney's decomposition theorem [Sid75] from 1975.

A Sidney decomposition partitions the set N of jobs into sets $S_1, S_2, \ldots, S_k$, such that there exists an optimal schedule where jobs from S_i are processed before jobs from S_{i+1}, for any $i = 1, \ldots, k-1$. Lawler [Law78] showed that a Sidney decomposition can be computed in polynomial time by performing a sequence of min-cut computations. Chekuri & Motwani [CM99] and Margot, Queyranne & Wang [MQW03] proved that every schedule that complies with a Sidney decomposition is a 2-approximate solution. It turns out that the fixed-cost is crucial for this analysis: a schedule that complies with a Sidney decomposition is not necessarily a 2-approximate solution if we only consider the variable-cost [Uha08]. Correa & Schulz [CS05] showed that all known 2-approximation algorithms follow a Sidney decomposition, and therefore belong to the class of approximation algorithms described by Chekuri & Motwani [CM99] and Margot, Queyranne & Wang [MQW03]. Recently, Schulz & Uhan [SU08] showed for a large class of random generated instances that almost all instances are not Sidney decomposable. Hence, for almost all of those instances, any feasible schedule is a 2-approximation. They actually proved the stronger statement that for almost all *randomly* generated instances, all feasible schedules are arbitrarily close to optimal.

Due to the difficulty of obtaining better than 2-approximation algorithms for the general case, it is interesting to understand for which special cases one can achieve a better performance guarantee. A particular successful and popular approach has been to consider special cases of precedence constraints. Indeed, Smith [Smi56] showed already in 1956 that, in the absence of precedence constraints, an optimal solution can be found by sequencing the jobs in non-increasing order of the ratio w_i/p_i. Later, several other results for special classes

[1] This work later appeared in a joint journal version [HSSW97] together with the work of Hall, Shmoys, & Stein [HSW96], who used linear programming relaxations to give a constant factor $(4+\epsilon)$-approximation algorithm for $1|prec|\sum w_j C_j$.

of precedence constraints were proposed (see [LLKS93] for a survey), most notably, Lawler's [Law78] $O(n \log n)$ time algorithm for series-parallel precedence constraints. Goemans & Williamson [GW00] provided a nice alternative proof for the correctness of this algorithm by using the two-dimensional Gantt charts of Eastman et al. [EEI64]. For interval orders and convex bipartite precedence constraints, Woeginger [Woe03] gave approximation algorithms with an approximation ratio arbitrarily close to the golden ratio $\frac{1}{2}(1+\sqrt{5}) \approx 1.61803$. Using a similar approach, Kolliopoulos and Steiner [KS02] gave an approximation algorithm with the same performance guarantee (≈ 1.61803) for the special case of two-dimensional precedence constraints.

Recently, Ambühl & Mastrolilli [AM09] settled an open problem first raised by Chudak & Hochbaum [CH99] and whose answer was subsequently conjectured by Correa & Schulz [CS05]. As showed by Correa & Schulz, the settlement of this conjecture has several interesting consequences for $1|prec|\sum w_j C_j$. The results in [CS05; AM09] imply the existence of an exact polynomial time algorithm for the special case of two-dimensional precedence constraints, improving on previously known approximation algorithms [KS02; CS05], and generalizing Lawler's exact algorithm for series-parallel orders [Law78].

The most significant implication of [CS05; AM09] is that $1|prec|\sum w_j C_j$ is in fact a special case of the weighted vertex cover problem. More precisely, they proved that every instance S of $1|prec|\sum w_j C_j$ can be translated in polynomial time into a weighted graph $G^S_{\mathbf{p}}$ (see Section 3.3 for details), such that finding an optimum of the variable-cost of S can be reduced to finding a minimum vertex cover in $G^S_{\mathbf{p}}$. This result even holds for approximate solutions: finding an α-approximate solution for the variable-cost of S can be reduced to finding an α-approximate vertex cover in $G^S_{\mathbf{p}}$. By using this relationship several previous results for the scheduling problem can be explained, and in some cases improved, by means of the vertex cover theory. In particular, as vertex cover can be approximated within a factor of 2, it follows that the variable-cost of the scheduling problem can be approximated within a factor of 2.

3.1.2 Results and Overview of Techniques

The vertex cover problem is one of the best studied problems in theoretical computer science. There are several 2-approximation algorithms (see [Pas97] for a survey) and it seems unlikely that one can do better: assuming the unique games conjecture [Kho02], it is NP-hard to approximate vertex cover within $(2-\epsilon)$ for any constant $\epsilon > 0$ [KR08].

Instead, one can hope that the special structure of the vertex cover graph $G^S_{\mathbf{p}}$ associated to $1|prec|\sum w_j C_j$, allows for better approximation. Indeed, the

structure of $G_{\mathbf{p}}^S$ heavily depends on the precedence constraints, which determine its vertices and edges. Moreover, Correa & Schulz [CS05] showed that the graph $G_{\mathbf{p}}^S$ is bipartite if and only if the precedence constraints are two-dimensional. As vertex cover is solvable in polynomial time on bipartite graphs, it follows that $1|prec|\sum w_jC_j$ with two-dimensional precedence constraints has an exact algorithm that runs in polynomial time.

In this chapter we make the connection between the structure of $G_{\mathbf{p}}^S$ and the precedence constraints explicit. More specifically, in Section 3.3.1 we show that the vertex cover graph $G_{\mathbf{p}}^S$ associated to $1|prec|\sum w_jC_j$ is exactly the graph of incomparable pairs $G_{\mathbf{p}}$ defined in dimension theory of partial orders [Tro92] (see also Section 3.2). This equivalence allows us to benefit from dimension theory to obtain both positive and negative results.

Positive Results

In dimension theory of partial orders, it is well known [FT00] that the chromatic number of $G_{\mathbf{p}}$ (and hence $G_{\mathbf{p}}^S$) is at most d, whenever the dimension of the precedence constraints at hand is (at most) d. Hochbaum [Hoc83] showed that if a given graph for the weighted vertex cover problem can be colored using d colors in polynomial time, then there exists a $(2-2/d)$-approximation algorithm for the corresponding weighted vertex cover problem. It follows that there exists a $(2-2/d)$-approximation algorithm for (the variable-cost of) $1|prec|\sum w_jC_j$ for special classes of precedence constraints that have dimension at most d (that can be computed in polynomial time; see Theorem 3.3.4 for an exact statement).

Following a similar argument, we further generalize the above described approach to precedence constraints of *fractional* dimension [BS92b] (see Section 3.4). This generalization yields $(2-2/f)$-approximation algorithms for (the variable-cost of) $1|prec|\sum w_jC_j$ whenever precedence constraints have fractional dimension bounded by a constant f and satisfy an additional mild condition (see Theorem 3.4.1).

By following this general approach, we obtain improved approximation algorithms for relevant special classes of precedence constraints of low (fractional) dimension, such as *interval orders* (Section 3.5.1), *semi-orders* (Section 3.5.2), precedence constraints of *bounded up/down degree* (Section 3.5.3), and *convex bipartite* precedence constraints (Section 3.5.4)[2]. In fact, our approach yields the best known approximation ratios for all previously considered special classes of precedence constraints, and it provides the first results for precedence constraints of bounded up/down degree. We remark that the complexity of

[2]We refer the reader to the respectively sections for the definitions of these classes of precedence constraints.

$1|prec|\sum w_jC_j$ with these special classes of precedence constraints is unknown with the exception of precedence constraints of bounded up/down degree d which is known to be NP-hard for $d = 2$ [Law78].

For the considered special classes of precedence constraints, Table 3.1 summarizes the achieved approximation guarantees for $1|prec|\sum w_jC_j$.

Precedence Constraints	*Previous Best*	*Our*
Interval orders (*)	≈ 1.618 [Woe03]	1.5
Bounded up/down degree d (*)	2	$2-2/(d+1)$
Semi-orders	≈ 1.618 [Woe03]	4/3
Convex bipartite	≈ 1.618 [Woe03]	4/3

Table 3.1: A comparison between the previous best and our approximation guarantees for $1|prec|\sum w_jC_j$ with the considered special classes of precedence constraints. For the precedence constraints marked with (*), we need the generalization to fractional dimension as they have unbounded dimension.

Negative Results

In this chapter, we also show that $1|prec|\sum w_jC_j$ with interval order precedence constraints remains (weakly) NP-hard, a result that motivates the design of approximation algorithms for this special case. Moreover, we explain why our approach failed to improve the approximation guarantee in the general case by showing that minimizing the variable-cost of $1|prec|\sum w_jC_j$ is as hard to approximate as the vertex cover problem. Finally, we present the first inapproximability result for $1|prec|\sum w_jC_j$ that rules out, under a fairly standard assumption, the existence of a PTAS for the addressed scheduling problem.

Theorem 3.1.1 *Problem* $1|prec|\sum w_jC_j$ *restricted to interval order precedence constraints is* NP*-hard.*

Proof overview. In Section 3.6, we present a reduction to $1|prec|\sum w_jC_j$ with interval order precedence constraints from the NP-hard problem of finding a minimum (unweighted) vertex cover in a graph with bounded degree 3 [GJS76].

The main idea is to exploit the vertex cover nature of problem $1|prec|\sum w_jC_j$: finding an optimum solution to a scheduling instance S, where precedence constraints are given by an interval order $\mathbf{P}$, is equivalent to solving the minimum weighted vertex cover in the graph $G^S_{\mathbf{P}}$.

Given a graph G, we first transform it into another graph G' such that finding a minimum vertex cover of G is equivalent to finding a minimum vertex cover

of G' (see Lemma 3.6.1). The reason for this transformation is that the graph G' has some desirable structure, which allows us to construct a scheduling instance S, with interval order precedence constraints $\mathbf{P}$, such that:

1. A vertex of $G^S_{\mathbf{P}}$ has either weight 1 or weight at most $1/k$, where k can be selected to be arbitrarily large (see Lemma 3.6.4).

2. The subgraph of $G^S_{\mathbf{P}}$ induced by the vertices of weight 1 is isomorphic to G' (see Lemma 3.6.5).

As k can be selected to be arbitrarily large, we can conclude that finding an optimal schedule for S — a minimum vertex cover of $G^S_{\mathbf{P}}$ — is essentially equivalent to finding a minimum vertex cover of G', which in turn is equivalent to finding a minimum vertex cover of G.

□

The positive results of this chapter yield improved approximation algorithms for several classes of precedence constraints by exploiting the vertex cover nature of the problem and deriving better than 2-approximation algorithms for the variable-cost of $1|prec|\sum w_jC_j$. It is a natural and interesting question to ask whether a better than 2-approximate solution for the general version of the problem can be obtained in a similar vein. The following theorem shows this to be unlikely by showing that optimizing the variable-cost of $1|prec|\sum w_jC_j$ is in fact equivalent to vertex cover in terms of approximability.

Theorem 3.1.2 *Approximating the variable-cost of* $1|prec|\sum w_jC_j$ *is equivalent to approximating the vertex cover problem.*

Proof overview. The results in [CS05; AM09] imply that minimizing the variable-cost of $1|prec|\sum w_jC_j$ is a special case of vertex cover, and therefore is not harder to approximate. In Section 3.7.1 we prove the other direction (Theorem 3.7.1).

The main idea is similar to the one in the proof of Theorem 3.1.1, i.e., we again exploit the vertex cover nature of $1|prec|\sum w_jC_j$. As the precedence constraints are not restricted to form an interval order, we get the following stronger reduction. Given a graph G, we construct a scheduling instance S, with precedence constraints $\mathbf{P}$, such that:

1. A vertex of $G^S_{\mathbf{P}}$ has either weight 1 or weight at most $1/k$, where k can be selected to be arbitrarily large.

2. The subgraph of $G^S_{\mathbf{P}}$ induced by the vertices of weight 1 is isomorphic to G.

Recall that optimizing the variable-cost of S is equivalent to the weighted vertex cover problem on $G^S_{\mathbf{p}}$. As k can be selected to be arbitrarily large, we can conclude that approximating the variable-cost of S, i.e., approximating vertex cover on $G^S_{\mathbf{p}}$, is essentially equal to approximating vertex cover on G.

□

The best known hardness results for the vertex cover problem give us the following corollaries.

Corollary 3.1.3 (Theorem 3.1.2 + [DS05]) *The variable-cost of* $1|prec|\sum w_jC_j$ *is NP-hard to approximate within a factor of* 1.3606.

Corollary 3.1.4 (Theorem 3.1.2 + [KR08]) *Assuming the unique games conjecture, the variable-cost of* $1|prec|\sum w_jC_j$ *is NP-hard to approximate within a factor of* $2-\epsilon$ *for any constant* $\epsilon > 0$.

Unfortunately, the techniques used to prove Theorem 3.1.2 fail to yield any inapproximability results if the complete objective function (i.e. the fixed-cost plus the variable-cost) is considered: the fixed-cost introduced during the reduction can be seen to dominate the objective function value, which makes any feasible solution close to optimal.

By a different approach, we present the first inapproximability result for $1|prec|\sum w_jC_j$ that rules out, under a fairly standard assumption, the existence of a PTAS for $1|prec|\sum w_jC_j$.

Theorem 3.1.5 *Let* $\epsilon > 0$ *be an arbitrarily small constant. If there is a PTAS for* $1|prec|\sum w_jC_j$ *then SAT has a (probabilistic) algorithm that runs in time* 2^{n^ϵ} *on an instance of size n.*

Proof overview. The result is obtained in Section 3.7.2, by showing a nice relationship between $1|prec|\sum w_jC_j$ and the maximum edge biclique problem (MEB): given an n by n bipartite graph G, the maximum edge biclique problem is to find a k_1 by k_2 complete subgraph of G that maximizes $k_1 \cdot k_2$.

More precisely, given an n by n bipartite graph G, we construct a scheduling instance S_G such that if a maximum edge biclique of G has value an^2 for some $a \in (0,1]$, then

$$n^2 - an^2(\ln 1/a + 2) \leq val(\sigma^*) \leq n^2 - an^2,$$

where $val(\sigma*)$ denotes the value of an optimal schedule σ^* of S.

This relationship allows us to use hardness results for MEB to obtain hardness results for $1|prec|\sum w_jC_j$. In particular, the currently best inapproximability result for maximum edge biclique (that we obtain in Chapter 4) yields that the scheduling problem has no PTAS unless SAT can be solved by a (probabilistic) algorithm that runs in time 2^{N^ϵ}, where N is the instance size and $\epsilon > 0$ can be made arbitrarily close to 0. □

3.2 Dimension Theory of Partial Orders

A partially ordered set (or poset) formalizes the intuitive concept of an ordering, sequencing, or arrangement of the elements of a set. The theory of partially ordered sets is central to combinatorics and arises in many different contexts. We are going to introduce it by giving an example.

Consider the set $N = \{1,2,3,4,5\}$. The elements of the set N, called the *ground set*, can be partially ordered by the reflexive, antisymmetric and transitive relation P which we define, in the example, to be the precedence constraints of the jobs in Figure 3.1, i.e. $P = \{(1,4),(2,4),(2,5),(3,4)\}$. To emphasize the order concept, we write $x \leq y$ when $(x,y) \in P$.

For any $x, y \in N$, in case $x \leq y$ or $y \leq x$, we will say that the pair (x,y) is *comparable*, otherwise it is called *incomparable* (for $x \neq y$ we say that y is *above* x, if $x \leq y$, also denoted by $x < y$). For instance, 1 and 4 are comparable since $1 \leq 4$, whereas neither $1 \leq 5$ nor $5 \leq 1$ holds which makes the pairs $(1,5)$ and $(5,1)$ incomparable. The ground set N together with the partial order P define a *partially ordered set (poset)* $\mathbf{P} = (N,P)$. We denote the set of all incomparable pairs of a poset $\mathbf{P}$ by $inc(\mathbf{P})$. A poset that does not have any incomparable pairs is called a *linear order*.

A partial order P' on N is an *extension* of P (on the same set N), if $P \subseteq P'$. Moreover, if P' is a linear order, we call P' a *linear extension* of P. We can construct an extension of P by adding, for example, $3 \leq 5$ to P. The resulting poset $\mathbf{P}' = (N,P')$ is depicted in Figure 3.2. We say that the extension P' *reverses* the (ordered) incomparable pair $(5,3)$.

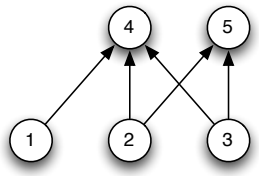


Figure 3.2: An extension of the partial order P with $P' = P \cup \{(3,5)\}$.

Constructing a linear extension that satisfies certain properties is central to several problems in combinatorics. The addressed scheduling problem falls in this category. Indeed, it can be formulated as the problem of finding a linear extension L of a partial order P on the set N of jobs so as to minimize $\sum_{(i,j)\in\mathbf{L}} p_i w_j + \sum_{i\in N} p_i w_i$. The construction of linear extensions of a poset also arises when determining its dimension, as explained in the following.

For a family $\mathscr{R}$ of linear extensions of a partial order P on N, we call $\mathscr{R}$ a *realizer* of $\mathbf{P} = (N, P)$, if $\bigcap \mathscr{R} = P$, i.e. for all $a, b, \in N$, $a \leq b$ in P if and only if $a \leq b$ in every $L \in \mathscr{R}$. Note that for each incomparable pair $(a, b) \in \text{inc}(\mathbf{P})$, there must be at least one linear extension in the realizer that reverses it. For a realizer of the example poset see Figure 3.3. The least positive integer t for which there is such a realizer $\mathscr{R} = \{L_1, \ldots, L_t\}$ of $\mathbf{P}$ is called the *dimension* of $\mathbf{P}$, denoted by $\dim(\mathbf{P})$ [DM41]. We will refer to a realizer $\mathscr{R} = \{L_1, \ldots, L_t\}$ of size t as a t-realizer. Generalizing this concept, a *k:t*-realizer is a multiset of t linear extensions $\mathscr{R} = \{L_1, \ldots, L_t\}$ in which each incomparable pair is reversed at least k times. The *fractional dimension* of $\mathbf{P}$, denoted by $\text{fdim}(\mathbf{P})$, is the infimum of the set of ratios t/k for which there exist *k:t*-realizers [BS92b]. Note that the fractional dimension of a poset is always bounded from above by its dimension and for some classes of posets we will see that the fractional dimension can be significantly smaller than the dimension.

A natural question is for which posets one can construct a t-realizer in polynomial time. In the general case, Yannakakis [Yan82] proved that determining whether the dimension of a poset is at most t is NP-complete for every $t \geq 3$. Moreover, Hedge & Jain [HJ07] recently proved that, in general, it is NP-hard to approximate the (fractional) dimension of a poset with n elements within a factor $n^{0.5-\epsilon}$, for any constant $\epsilon > 0$. However, for several special cases, a minimal realizer can be computed in polynomial time (see e.g. [Möh89; Tro92]).

3.2.1 The Hypergraph and Graph of Incomparable Pairs

It is easy to see that, when constructing an extension of a partial order P on a ground set N, there are some groups of incomparable pairs that cannot be reversed at the same time. Obviously, an extension of the example partial order P cannot reverse both $(3, 5)$ and $(5, 3)$ at the same time. This implies that, unless a poset is already a linear order, any realizer needs to contain more than one linear extensions in order to reverse every incomparable pair at least once. For the pairs of incomparable pairs mentioned above, it is obvious that they cannot be reversed at the same time. There are also less obvious pairs of incomparable pairs for which this is true. By examining $\mathbf{P}$, defined by the jobs and precedence constraints of Figure 3.1, one can conclude that reversing both $(2, 1)$ and $(1, 5)$ would lead to an inconsistency, i.e. a cycle in the ordering: adding $1 \leq 2$ and using transitivity leads to $1 \leq 2 \leq 5$, which contradicts $5 \leq 1$. In general, there can also be groups bigger than two pairs that cannot be all reversed at the same time without introducing contradictions (e.g. $(2, 1)$, $(1, 3)$ and $(3, 5)$).

The above observations naturally lead to the definition of the *hypergraph of incomparable pairs* $\mathbf{H_P}$ of a poset $\mathbf{P}$ [FT00] defined as follows. The vertices of

$\mathbf{H_P}$ are the incomparable pairs in $\mathbf{P}$. The edge set consists of those sets U of incomparable pairs such that:

1. No linear extension of P reverses all incomparable pairs in U.
2. For every proper subset of U there is a linear extension that reverses all incomparable pairs in U.

Let $G_\mathbf{P}$ denote the ordinary graph, called the *graph of incomparable pairs*, determined by all edges of size 2 in $\mathbf{H_P}$. Figure 3.3 depicts the hypergraph and graph of incomparable pairs for our example poset.

Recall that a *k:t*-coloring of a (hyper)graph is an assignment of sets of k colors to vertices by using a pool of at most t colors, such that the intersection of the sets assigned to vertices of any (hyper)edge is empty. Note that, with this terminology, the classical graph coloring problem is to find the minimal t such that there exists a *1:t* coloring. Felsner & Trotter [FT00] defined the hypergraph of incomparable pairs so as to capture the properties of a poset. In [BS92b] it is pointed out that these properties can be generalized to the fractional dimension.

The following result is known in dimension theory of partial orders (see e.g. [Tro92; BS92b]). We include the proof as it gives insights in the structure of the hypergraph (and graph) of incomparable pairs.

Proposition 3.2.1 ([Tro92; BS92b]) *Let $\mathbf{P} = (N, P)$ be a poset which is not a linear order, and let $\mathbf{H_P}$ be its hypergraph of incomparable pairs. Then*

a) *$\mathbf{H_P}$ can be k:t-colored if and only if $\mathbf{P}$ has a k:t-realizer.*

b) *For a linear extension L of P, the vertex sets $\{(x,y) \in \mathrm{inc}(\mathbf{P}) : (y,x) \in L\}$ and $\{(x,y) \in \mathrm{inc}(\mathbf{P}) : (x,y) \in L\}$ form an independent set and vertex cover of $\mathbf{H_P}$ (and $G_\mathbf{P}$), respectively.*

Proof. For statement a), note that a *k:t*-coloring suggests a way of reversing each incomparable pair in at least k linear extensions, with each color corresponding to a linear extension. Since a valid coloring does not use a color common to all vertices of a hyperedge, these extensions do not contain cycles and are therefore valid partial orders. On the other hand, a *k:t*-realizer suggests a way of coloring the vertices of the graph by assigning color i to each pair $(a,b) \in \mathrm{inc}(\mathbf{P})$ if and only if it is reversed in L_i. Since a linear extension can only reverse a proper subset of vertices for each hyperedge, there is no hyperedge with a color common to all of its vertices. Therefore, the coloring is valid. Finally, note that b) follows by the above arguments since all pairs that are reversed in a linear extension are assigned the same color and as the coloring is feasible, they form an independent set (and the complement, i.e., the pairs that are not reversed, form a vertex cover). □

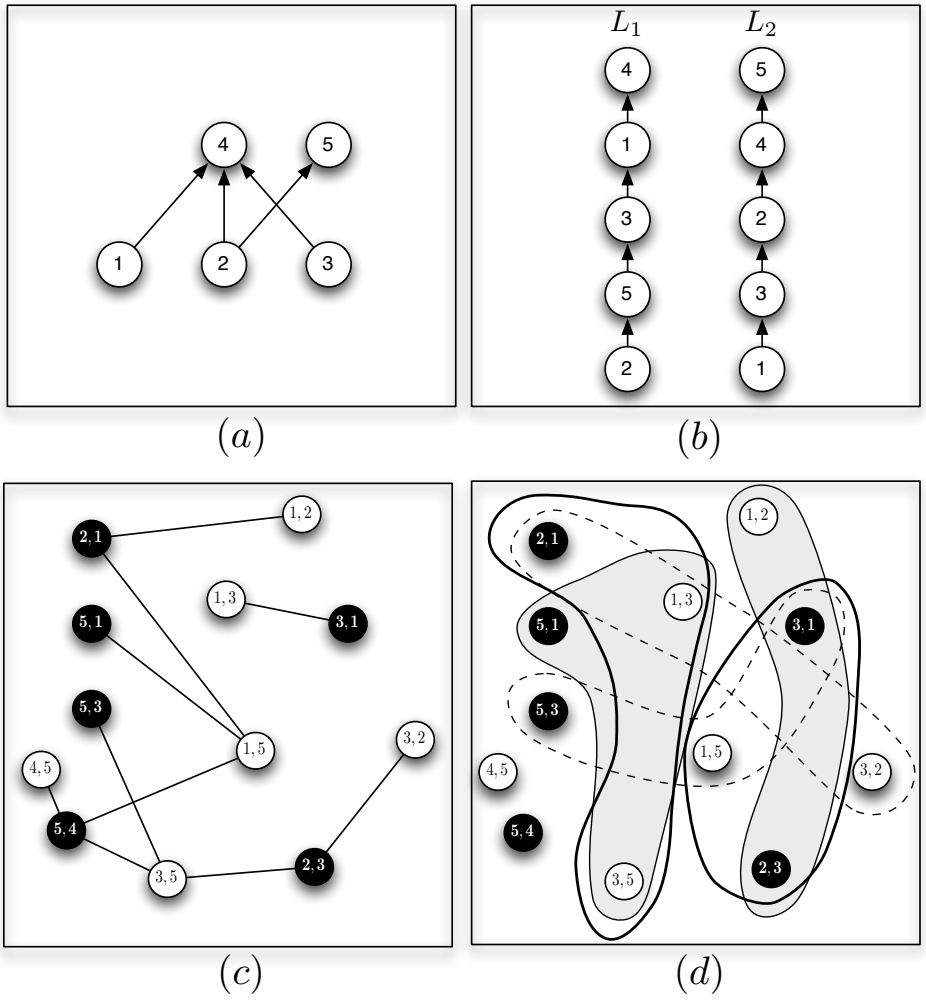


Figure 3.3: a) The example poset **P**. b) A realizer $\mathscr{R} = \{L_1, L_2\}$ of **P**. c) The graph $G_{\mathbf{P}}$, where the independent sets defined by L_1 and L_2 are depicted in white and black, respectively. d) The hyperedges of $\mathbf{H}_{\mathbf{P}}$; the remaining edges of $\mathbf{H}_{\mathbf{P}}$ are depicted in (c).

3.3 The Scheduling Problem and Vertex Cover

In a series of recent papers [CH99; CS05; AM09] it was proved that $1|prec|\sum w_jC_j$ is a special case of minimum weighted vertex cover. In this section we give an overview of how this result was obtained. We will also make the connection between the scheduling problem and dimension theory explicit by pointing out that the vertex cover graph obtained from a scheduling instance, with precedence constraints in the form of a poset $\mathbf{P}$, is in fact the graph of incomparable pairs $G_{\mathbf{P}}$, defined in dimension theory of partial orders (see Section 3.2).

To simplify notation, we implicitly assume hereafter that tuples and sets of jobs have no multiplicity. Therefore, $(a_1, a_2, \ldots, a_k) \in N^k$ and $\{b_1, b_2, \ldots, b_k\} \subseteq N$ denote a tuple and a set, respectively, with k distinct elements.

In the following, we introduce several linear programming formulations and relaxations of $1|prec|\sum w_jC_j$ using linear ordering variables δ_{ij}. The variable δ_{ij} has value 1 if job i precedes job j in the corresponding schedule, and 0 otherwise. The first formulation using linear ordering variables is due to Potts [Pot80], and it can be stated as follows.

$$\text{[P-IP]} \qquad \min \quad \sum_{j\in N} p_j w_j + \sum_{(i,j)\in N^2} \delta_{ij} p_i w_j \tag{3.1a}$$

$$\text{s.t.} \quad \delta_{ij} + \delta_{ji} = 1 \qquad \{i,j\} \subseteq N \tag{3.1b}$$

$$\delta_{ij} = 1 \qquad (i,j) \in P \tag{3.1c}$$

$$\delta_{ij} + \delta_{jk} + \delta_{ki} \leq 2 \qquad (i,j,k) \in N^3 \tag{3.1d}$$

$$\delta_{ij} \in \{0,1\} \qquad (i,j) \in N^2 \tag{3.1e}$$

Constraint (3.1b) ensures that either job i is scheduled before j or vice versa. If job i is constrained to precede j in the partial order P, then this is seized by Constraint (3.1c). The set of Constraints (3.1d) is used to capture the transitivity of the ordering relations (i.e., if i is scheduled before j and j before k, then i is scheduled before k). It is easy to see that [P-IP] is indeed a complete formulation of the problem [Pot80].

Chudak & Hochbaum [CH99] suggested to study the following relaxation of [P-IP]:

$$\text{[CH-IP]} \qquad \min \quad (3.1a)$$

$$\text{s.t.} \quad (3.1b), (3.1c), (3.1e)$$

$$\delta_{jk} + \delta_{ki} \leq 1 \qquad (i,j) \in P, \{i,j,k\} \subseteq N \tag{3.1d'}$$

In [CH-IP], Constraints (3.1d) are replaced by Constraints (3.1d'). These inequalities correspond in general to a proper subset of (3.1d), since only those

transitivity Constraints (3.1d) for which two of the participating jobs are already related to each other by a precedence constraint are kept.

Correa & Schulz [CS05] proposed the following integer programming formulation [CS-IP], which is a relaxation of [P-IP]. For $(i,j) \in N^2$, the term $i \| j$ means that $(i,j) \notin P$ and $(j,i) \notin P$. Note that in [CS-IP], we only need variables δ_{ij} for $i \| j$.

$$\text{[CS-IP]} \qquad \min \quad \sum_{j \in N} p_j w_j + \sum_{(i,j) \in P} p_i w_j + \sum_{i \| j} \delta_{ij} p_i w_j \tag{3.2a}$$

$$\text{s.t.} \quad \delta_{ij} + \delta_{ji} \geq 1 \qquad i \| j \tag{3.2b}$$

$$\delta_{ik} + \delta_{kj} \geq 1 \qquad (i,j) \in P, j \| k, k \| i \tag{3.2c}$$

$$\delta_{i\ell} + \delta_{kj} \geq 1 \qquad (i,j) \in P, j \| k, (k,\ell) \in P, \ell \| i \tag{3.2d}$$

$$\delta_{ij} \in \{0,1\} \qquad (i,j) \in N^2, i \| j$$

We remark that the objective function is split into the fixed-cost $\sum_{(i,j) \in P} p_i w_j + \sum_{i \in N} p_i w_i$ and the variable-cost $\sum_{(i,j) \in L \setminus P} p_i w_j$ defined in Section 3.1, where $L = \{(i,j) : \delta_{ij} = 1\}$ is not necessarily a total ordering of the jobs. It follows that solving [CS-IP] is equivalent to solving the same integer program but with the objective to only minimize the variable-cost, which in turn can be interpreted as a weighted vertex cover problem on a graph with a vertex of weight $p_i w_j$ for each variable δ_{ij} and an edge between two vertices if their corresponding variables occur together in a constraint.

Correa & Schulz [CS05] also proved that their formulation [CS-IP] is equivalent to [CH-IP] and they conjectured that an optimal solution to $1|prec|\sum w_j C_j$ gives an optimal solution to [CH-IP] as well. The conjecture in [CS05] was recently settled in the affirmative by Ambühl & Mastrolilli [AM09], who proved that any feasible solution to [CH-IP] can be turned, in polynomial time, into a feasible solution to $1|prec|\sum w_j C_j$ without deteriorating the objective value. As the fixed-cost remains unchanged during the transformations, the results in [CS05; AM09] imply that the problem of minimizing the variable-cost of $1|prec|\sum w_j C_j$ is a special case of minimum weighted vertex cover.

For a scheduling instance S with precedence constraints $\mathbf{P} = (N, P)$, we let $G^S_{\mathbf{P}}$ be the vertex cover graph obtained by interpreting [CS-IP] as a vertex cover problem. The following theorem summarizes the aforementioned results.

Theorem 3.3.1 ([CH99; CS05; AM09]) *Let S be an instance of $1|prec|\sum w_j C_j$ with precedence constraints $\mathbf{P}$. Then an α-approximate solution to the weighted vertex cover problem on $G^S_{\mathbf{P}}$ can, in polynomial time, be turned into an α-approximate solution to the variable-cost part of S.*

As the approximation guarantee might only improve by taking into account the fixed-cost, we have that an α-approximation algorithm for the weighted vertex cover problem associated to $1|prec|\sum w_j C_j$, yields *at least* an α-approximation algorithm for the scheduling problem. Understanding whether the approximation guarantee for $1|prec|\sum w_j C_j$ can be improved significantly by taking into account the fixed-cost turns out to be fundamental for understanding its approximability (see Section 3.7.1).

3.3.1 Structure of the Graph $G_{\mathbf{P}}^{S}$

The vertex cover problem is one of the best studied problems in theoretical computer science (see [Pas97] for a survey). In general, assuming the unique games conjecture [Kho02], it is NP-hard to approximate vertex cover within $(2-\epsilon)$ for any constant $\epsilon > 0$ [KR08]. However, the graphs obtained from $1|prec|\sum w_j C_j$ have a very nice structure; a fact that will lead to better approximation algorithms for several kinds of precedence constraints. To see this, consider an instance S of $1|prec|\sum w_j C_j$ with precedence constraints $\mathbf{P} = (N, P)$. Recall that the vertices of $G_{\mathbf{P}}^{S}$ are the incomparable pairs of $\mathbf{P}$. Graph $G_{\mathbf{P}}^{S}$ has three types of edges (see also Figure 3.4):

(i) Two vertices (i, j) and (j, i) are adjacent.

(ii) Two vertices (i, k) and (k, j) are adjacent if $(i, j) \in P$.

(iii) Two vertices (i, ℓ) and (k, j) are adjacent if $(i, j) \in P$ and $(k, \ell) \in P$.

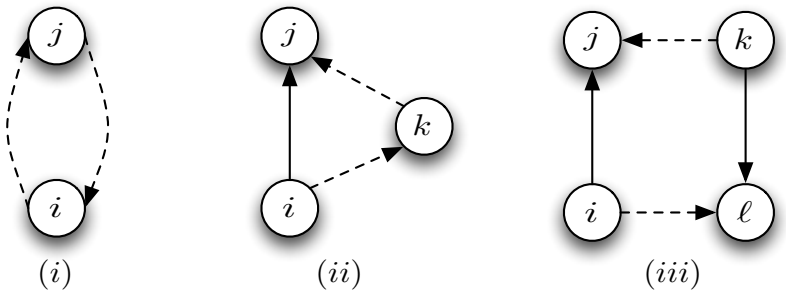


Figure 3.4: The three types of edges of $G_{\mathbf{P}}^{S}$. The incomparable pairs that are adjacent vertices in $G_{\mathbf{P}}^{S}$ are depicted by dashed arrows. Note that, in all three cases, no linear extension reverses both incomparable pairs.

It is not hard to see that the edge set of $G_{\mathbf{P}}^S$ consists of those sets U of *two* incomparable pairs such that no linear extension of P reverses both incomparable pairs in U. As a result (see the definition of graph of incomparable pairs in Section 3.2.1), we have the following proposition that establishes a strong relationship between dimension theory and $1|prec|\sum w_jC_j$.

Proposition 3.3.2 *The vertex cover graph $G_{\mathbf{P}}^S$ associated to $1|prec|\sum w_jC_j$ and the graph of incomparable pairs $G_{\mathbf{P}}$ coincide.*

The combinatorial theory of partially ordered sets is well studied. Tapping this source can help in designing approximation algorithms. The following theorem is such an example.

Theorem 3.3.3 ([Tro92; CS05]) *Let $\mathbf{P} = (N, P)$ be a poset that is not a linear order. Then the graph $G_{\mathbf{P}}$ is bipartite if and only if $dim(\mathbf{P}) = 2$.*

We note that the poset depicted in Figure (3.3-a) is two-dimensional, and hence the graph of incomparable pairs is bipartite (Figure (3.3-c)). Theorem 3.3.3 is a well-known result in dimension theory. Using a different approach, Correa & Schulz [CS05] rediscovered it for the vertex cover graph $G_{\mathbf{P}}^S$, independent of the connection pointed out by Proposition 3.3.2. Furthermore, Propositions 3.3.2 and 3.2.1 give us that graph $G_{\mathbf{P}}^S$ can be colored using $dim(\mathbf{P})$ colors and, given a realizer, this can be done in polynomial time (see the proof of Proposition 3.2.1). Combining this with Hochbaum's [Hoc83] $(2 - 2/t)$-approximation algorithm for the weighted vertex cover problem, whenever the vertex cover graph is t-colorable in polynomial time, gives us the following.

Theorem 3.3.4 *Problem $1|prec|\sum w_jC_j$, whenever precedence constraints are given by a t-realizer, has a polynomial time $(2 - \frac{2}{t})$-approximation algorithm.*

In the next section we generalize the above theorem to hold for fractional dimension.

3.4 Scheduling with Low Fractional Dimension

We prove that better than 2-approximation algorithms are possible provided that the set of precedence constraints has low fractional dimension.

We say that a poset $\mathbf{P}$ admits an *efficiently samplable k:t*-realizer if there exists a randomized algorithm that, in time polynomial in the size of the ground set (number of jobs), returns any linear extension from a k:t-realizer $\mathscr{R} = \{L_1, L_2, \ldots, L_t\}$ of $\mathbf{P}$ with probability $1/t$.

The following theorem generalizes Theorem 3.3.4. The main idea of the proof is the observation that Hochbaum's approach [Hoc83] for approximating the vertex cover problem can be extended to fractional coloring, yielding a similar approximation result.

Theorem 3.4.1 *The problem* $1|prec|\sum w_j C_j$*, whenever precedence constraints admit an efficiently samplable k:t-realizer, has a randomized* $(2-\frac{2}{t/k})$*-approximation algorithm.*

Proof. Let S be an instance of $1|prec|\sum w_j C_j$ where precedence constraints are given by a poset $\mathbf{P} = (N, P)$ that admits an efficiently samplable k:t-realizer $\mathscr{R} = \{L_1, L_2, \ldots, L_t\}$. Furthermore, we assume that $\mathrm{fdim}(\mathbf{P}) \geq 2$. The case when $\mathrm{fdim}(\mathbf{P}) = 1$, i.e., $\mathbf{P}$ is a linear order, is trivial.

Let $G_{\mathbf{P}}^S = (V_{\mathbf{P}}, E_{\mathbf{P}})$ be the weighted vertex cover instance associated to S where each vertex (incomparable pair) $v = (i, j) \in V_{\mathbf{P}}$ has weight $w_v = p_i \cdot w_j$, as specified in Section 3.3. In 1973, Nemhauser & Trotter [NT73; NT75] used the following integer program to model the minimum weighted vertex cover problem (see also Section 2.1.2):

$$
\begin{array}{llll}
\text{[VC-IP]} & \min & \displaystyle\sum_{v \in V_{\mathbf{P}}} w_v x_v & \\
& \text{s.t.} & x_u + x_v \geq 1 & \{u, v\} \in E_{\mathbf{P}} \\
& & x_v \in \{0, 1\} & v \in V_{\mathbf{P}}
\end{array}
$$

They also studied the linear relaxation [VC-LP] of [VC-IP], and proved that any basic feasible solution to [VC-LP] is *half-integral*, that is $x_v \in \{0, \frac{1}{2}, 1\}$ for all $v \in V_{\mathbf{P}}$.

We now proceed by solving the [VC-LP] formulation of $G_{\mathbf{P}}^S$. By a result of Nemhauser & Trotter [NT75], this can be done combinatorially and in polynomial time. We let V_i be the set of vertices with value i ($i \in \{0, \frac{1}{2}, 1\}$) in the optimum solution. Denote by $G_{\mathbf{P}}^S[V_{1/2}]$ the subgraph of $G_{\mathbf{P}}^S$ induced by the vertex set $V_{1/2}$. We consider the linear extensions of $\mathscr{R}$ as outcomes in a uniform sample space. For an incomparable pair (x, y), the *probability that* $y > x$ *in a linear extension picked from* $\mathscr{R}$ is given by

$$
Prob_{\mathscr{R}}[y > x] = \frac{1}{t}|\{i = 1, \ldots, t : y > x \in L_i\}| \geq \frac{k}{t} \tag{3.4}
$$

The last inequality holds because every incomparable pair is reversed in at least k linear extensions of $\mathscr{R}$.

By Proposition 3.2.1, a linear extension L of P defines an independent set of $G_{\mathbf{P}}^S$ by taking the incomparable pairs that are reversed in L, i.e., the vertex set

$\{(x,y) \in \text{inc}(\mathbf{P}) : (y,x) \in L\}$. Let us pick one linear extension L uniformly at random from $\mathscr{R} = \{L_1, \ldots, L_t\}$. Then, by linearity of expectation, the expected value of the independent set $I_{1/2}$, obtained by taking the incomparable pairs in $V_{1/2}$ that are reversed in L, is

$$E[w(I_{1/2})] = \sum_{(i,j)\in V_{1/2}} Prob_{\mathscr{R}}[j > i] \cdot w_{(i,j)} \geq \frac{k}{t} \cdot w(V_{1/2}) \tag{3.5}$$

A vertex cover solution C for the graph $G_{\mathbf{P}}^S[V_{1/2}]$ can be obtained by picking the nodes that are not in $I_{1/2}$, namely $C = V_{1/2} \setminus I_{1/2}$. The expected value of this solution is

$$E[w(C)] = w(V_{1/2}) - E[w(I_{1/2})] \leq \left(1 - \frac{k}{t}\right) w(V_{1/2})$$

As observed in [Hoc83], $V_1 \cup C$ gives a valid vertex cover for graph $G_{\mathbf{P}}^S$. Moreover, the expected value of the cover is bounded as follows

$$\begin{aligned} E[w(V_1 \cup C)] &\leq w(V_1) + \left(1 - \frac{k}{t}\right) w(V_{1/2}) && (3.6)\\ &\leq 2\left(1 - \frac{k}{t}\right)\left(w(V_1) + \frac{1}{2}w(V_{1/2})\right) && (3.7)\\ &\leq \left(2 - \frac{2}{t/k}\right) OPT && (3.8) \end{aligned}$$

where the last inequality holds since $w(V_1) + \frac{1}{2}w(V_{1/2})$ is the optimal value of [VC-LP]. Note that $t/k \geq \text{fdim}(\mathbf{P}) \geq 2$ was used for the second inequality. Theorem 3.3.1 implies that any α-approximation algorithm for $G_{\mathbf{P}}^S$ also gives an α-approximation algorithm for S. Thus we obtain a randomized $(2 - \frac{2}{t/k})$-approximation algorithm for S. □

3.5 Applications

In this section we will apply the framework described in Sections 3.3.1 and 3.4 to design approximation algorithms for special cases of posets with low (fractional) dimension.

3.5.1 Interval orders

A poset $\mathbf{P} = (N, P)$ is an *interval order* if it has an interval representation F, which assigns to each $x \in N$ a closed interval $F(x) = [a_x, b_x]$ of the real line $\mathbb{R}$, so that $x < y$ in P if and only if $b_x < a_y$ in $\mathbb{R}$. For an example see Figure 3.5.

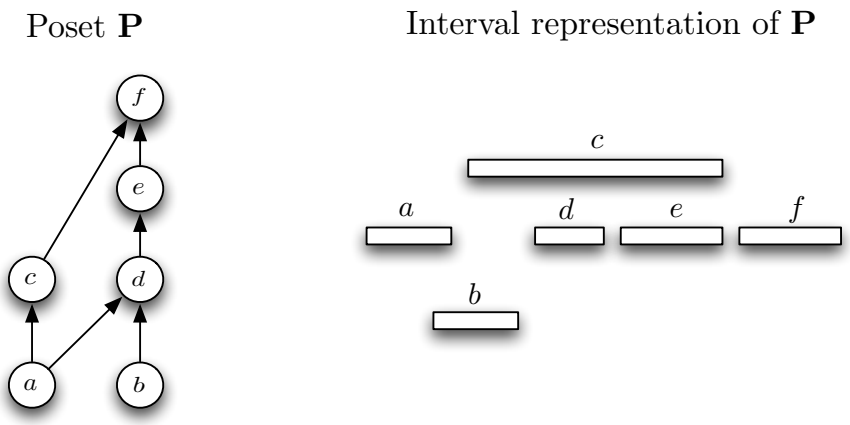


Figure 3.5: An example of an interval order.

Interval orders can be recognized in $O(n^2)$ time [PY79]. The dimension of interval orders can be of order $\log\log n$ [Tro92], whereas the fractional dimension is known to be less than 4 [BS92b], and this bound is asymptotically tight [FT94]. In the following we show how to obtain a 1.5-approximation algorithm for $1|prec|\sum w_jC_j$ with precedence constraints in the form of an interval order. By Theorem 3.4.1, it is sufficient to prove that interval orders admit an efficiently samplable k:t-realizer with $t/k \leq 4$.

Given a poset $\mathbf{P} = (N, P)$, disjoint subsets A and B of the ground set N, and a linear extension L of P, we say that *B is over A in L* if, for every incomparable pair of elements (a, b) with $a \in A$ and $b \in B$, one has $b > a$ in L. The following property of interval orders is fundamental for our approach.

Theorem 3.5.1 (Rabinovitch [Rab78]) *A poset $\mathbf{P} = (N, P)$ is an interval order if and only if for every pair (A, B) of disjoint subsets of N there is a linear extension L of P with B over A.*

We remark that given a pair (A, B) of disjoint subsets of N, we can find a linear extension L of P with B over A in polynomial time: add the relations $A \times B$ to P and complete the partial order to obtain a linear extension L. By using this property we can easily obtain a k:t-realizer $\mathscr{R} = \{L_1, \ldots, L_t\}$ with $k = 2^{n-2}$ and $t = 2^n$, where $n = |N|$. Indeed, consider every subset A of N and let L_A be a linear extension of P in which $B = N \setminus A$ is over A. Now let $\mathscr{R}$ be the multiset of all the L_A's. Note that $|\mathscr{R}| = 2^n$. Moreover, for any incomparable pair (x, y) there are at least $k = 2^{n-2}$ linear extensions in $\mathscr{R}$ for which $x \in B$ and $y \in A$. Finally, observe that we can efficiently pick uniformly at random one linear extension from $\mathscr{R}$: for every job $j \in N$ put j either in A or in B with the same probability $1/2$.

By the previous observations and Theorem 3.4.1, we have a randomized polynomial time 1.5-approximation for $1|prec|\sum w_j C_j$ with interval order precedence constraints. As described below, the algorithm can easily be derandomized by using the standard method of conditional probabilities.

Theorem 3.5.2 *Problem $1|prec|\sum w_j C_j$ for which the precedence constraints form an interval order has a 1.5-approximation algorithm.*

Derandomization for Interval Orders

We let $V_{1/2}$ be the set of vertices with value $1/2$ in the optimal solution to the [VC-LP] formulation of the scheduling problem (see Section 3.4).

Our goal is to partition the set of jobs into two sets A and B, such that any linear extension L with B over A will satisfy $w(V_{1/2} \setminus L) \geq w(V_{1/2})/4$, i.e., defines an independent set with weight at least $w(V_{1/2})/4$. For every pair (A_i, B_i) of disjoint sets of jobs, consider the following function where $(x, y) \in \text{inc}(P)$:

$$\Phi(A_i, B_i) = \sum_{(x,y)\in A_i \times B_i} p_y w_x + \sum_{\substack{x\in A_i, y\notin A_i\cup B_i \text{ or} \\ x\notin A_i\cup B_i, y\in B_i}} \frac{p_y w_x}{2} + \sum_{x,y\notin A_i\cup B_i} \frac{p_y w_x}{4}$$

Note that the pair (A_i, B_i) defines a partition of a subset of N and that $\Phi(A_i, B_i)$ gives a lower bound on the expected value of the independent set conditioned upon our current choices of A_i and B_i.

Set $A_0 = B_0 = \emptyset$ and observe that $\Phi(A_0, B_0) = w(V_{1/2})/4$. For $i = 1, \ldots n$ we have to decide if job i is in set A_i or in set B_i. We evaluate both possibilities:

1. $A_i^1 := A_{i-1} \cup \{i\}$ and $B_i^1 := B_{i-1}$
2. $A_i^2 := A_{i-1}$ and $B_i^2 := B_{i-1} \cup \{i\}$

Let $g = \arg\max_{h=1,2}\{\Phi(A_i^h, B_i^h)\}$ and observe that

$$\Phi(A_{i-1}, B_{i-1}) \leq \Phi(A_i^g, B_i^g).$$

We therefore set

$$\begin{aligned} A_i &:= A_i^g \\ B_i &:= B_i^g. \end{aligned}$$

At the end we have partitioned the set of jobs into two sets A_n and B_n such that $\Phi(A_n, B_n) \geq w(V_{1/2})/4$. Since P is an interval order $P \cup \{(a, b) \in A_n \times B_n : a||b\}$ is a valid extension of P and any linear extension of it gives an independent set of value $\geq w(V_{1/2})/4$.

3.5.2 Semi-Orders

An interval order $\mathbf{P} = (N, P)$ is called a *semi-order* (or unit interval order) if $\mathbf{P}$ has an interval representation F assigning to each $x \in N$ an interval $F(x) = [a_x, a_x + 1]$ so that $x < y$ in P if and only if $a_x + 1 < a_y$. Semi-orders can be recognized in $O(n^2)$ time (see e.g. [Möh89; Tro92]). In contrast to interval orders that have unbounded dimension [Tro92], Rabinovitch proved, by constructing a realizer, that the dimension of *semi-orders* is at most three [Rab78] (see also [Tro92] for a good explanation). The realizer proposed by Rabinovitch is constructed as follows (see Figure 3.6 for an overview). Given a semi-order $\mathbf{P} = (N, P)$, let $N_1 = N, P_1 = P$ and $\mathbf{P}_1 = (N_1, P_1)$. Then set $A_1 = \max(\mathbf{P}_1)$, where

$$\max(\mathbf{P}) := \{x \in N : \forall y \in N, \text{ either } (y, x) \in P \text{ or } (y, x) \in \text{inc}(\mathbf{P})\}.$$

If $N_i, P_i, \mathbf{P}_i$, and A_i have been defined for some $i : 1 \leq i < |N|$, set $N_{i+1} = N_i \setminus A_i$, $P_{i+1} = P_i \cap (N_{i+1} \times N_{i+1})$, $\mathbf{P}_{i+1} = (N_{i+1}, P_{i+1})$, and $A_{i+1} = \max(\mathbf{P}_{i+1})$. Note that the sets $A_1, A_2, \ldots, A_{|N|}$ form a partition of N and all elements in a set A_i are incomparable. Let $A = \{x \in N : x \in A_i \text{ for some odd } i\}$ and let $B = N \setminus A$. As semi-orders are a subset of interval orders, we can construct two linear extensions L_1 and L_2 so that B is over A in L_1 and A is over B in L_2 (see Theorem 3.5.1). Finally, we construct a third linear extension L_3 so that an incomparable pair $(i, j) \in L_3$, if either (i) $i \in A_k$ and $j \in A_\ell$ with $k > \ell$ or (ii) $\{i, j\} \subseteq A_k$ and $(j, i) \in L_1$.

It is not hard to see that we can construct $\mathscr{R} = \{L_1, L_2, L_3\}$ in polynomial time, which is a realizer of $\mathbf{P}$ [Rab78]. This together with Theorem 3.3.4, gives us the following theorem.

Theorem 3.5.3 *Problem* $1|prec|\sum w_j C_j$ *for which the precedence constraints form a semi-order has a (1+ 1/3)-approximation algorithm.*

3.5.3 Posets of Bounded Up/Down Degree

In the following we will see how to obtain, using Theorem 3.4.1, an approximation algorithm for $1|prec|\sum w_j C_j$ when the precedence constraints form a poset of bounded up/down degree. Before we proceed, we need to introduce some definitions.

Let $\mathbf{P} = (N, P)$ be a poset. For any job $j \in N$, define the *degree of* j, denoted $\deg(j)$, as the number of jobs comparable (but not equal) to j in $\mathbf{P}$. Let $\Delta(\mathbf{P}) = \max\{\deg(j) : j \in N\}$. Given a job j, let $D(j)$ denote the set of all jobs which are less than j, and $U(j)$ those which are greater than j in P. Define $\deg_D(j) = |D(j)|$ and $\Delta_D(\mathbf{P}) = \max\{\deg_D(j) : j \in N\}$. The quantities $\deg_U(j)$ and $\Delta_U(\mathbf{P})$ are defined analogously.

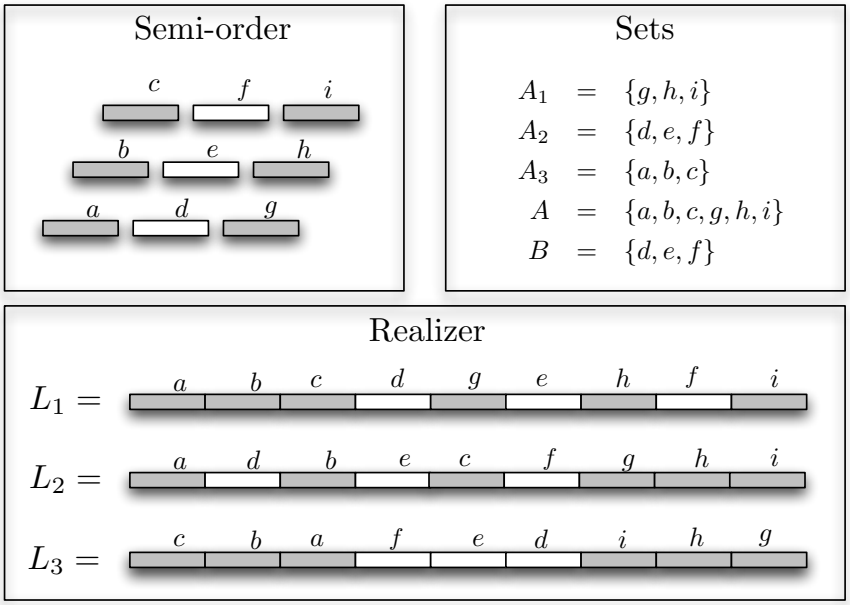


Figure 3.6: Overview of the construction of a 3-realizer for semi-orders. The jobs are partitioned into two sets A and B, depicted in gray and white, respectively.

We observe that the NP-completeness proof for $1|prec|\sum w_j C_j$ given by Lawler [Law78] was actually provided for posets $\mathbf{P}$ with $\Delta_D(\mathbf{P}) = 2$. By using fractional dimension we show that these posets (with bounded $\min\{\Delta_D, \Delta_U\}$) allow for a better than 2-approximation.

Theorem 3.5.4 *Problem* $1|prec|\sum w_j C_j$ *has a* $(2 - 2/f)$*-approximation algorithm, where* $f = 1 + \min\{\Delta_D, \Delta_U, 1\}$.

Proof. Let $\mathbf{P} = (N, P)$ be the poset representing the precedence constraints with bounded $\min\{\Delta_D, \Delta_U\}$. We will show that $\mathbf{P}$ has an efficiently samplable *k:t*-realizer with $t/k \leq \min\{\Delta_D, \Delta_U\} + 1$ by using a result by Felsner and Trotter [FT94]. To describe their approach we need to first introduce some concepts. Assume, without loss of generality, that $\mathbf{P}$ is *not* decomposable with respect to lexicographic sums (see Section 3.5.5). Otherwise, a decomposition with respect to lexicographic sums can be done in $O(n^2)$ time (see e.g. [Möh89]), and each component will have degree no larger than the degree of $\mathbf{P}$ and can be considered separately (see Theorem 3.5.9). We call an incomparable pair $(x, y) \in \text{inc}(\mathbf{P})$ a *critical pair* if for all $z, w \in N \setminus \{x, y\}$

1. $z < x$ in P implies $z < y$ in P, and

2. $y < w$ in P implies $x < w$ in P.

Critical pairs play an important role in dimension theory: any incomparable pair (a, b) can be associated with at least one critical pair (x, y) so that a linear extension reversing (x, y) also reverses (a, b) [Tro92]. It follows that if for each critical pair (x, y), there are at least k linear extensions in $\mathscr{R} = \{L_1, \ldots, L_t\}$ which reverse the pair (x, y) then $\mathscr{R}$ is a $k{:}t$-realizer of P and vice versa [BS92b].

For an element $x \in N$ and a set $A \subseteq N$ we write $x > A$, if $x > y$ for all $y \in A$. For any permutation M of N, consider the set $C(M)$ of critical pairs (x, y) that satisfy the following two conditions:

1. $x > (D(y) \cup \{y\})$ in M if $|D(y)| < \Delta_D$
2. $x > D(y)$ in M if $|D(y)| = \Delta_D$

In [FT94], Felsner & Trotter present an algorithm (for posets that are not decomposable with respect to lexicographic sums) that converts in polynomial time a permutation M of N to a linear extension L of P so that L reverses all critical pairs in the set $C(M)$. Now set $t = |N|!$ and consider the set $\mathscr{M} = \{M_1, M_2, \ldots, M_t\}$ of all permutations of the ground set N. Observe that for any critical pair (x, y) there are at least $n!/(\Delta_D + 1)$ different permutations $M_i \in \mathscr{M}$, in which either (i) $x > (D(y) \cup \{y\})$ in M_i if $|D(y)| < \Delta_D$ or (ii) $x > D(y)$ in M_i if $|D(y)| = \Delta_D$, i.e., $(x, y) \in C(M_i)$. It follows that for any critical pair there are at least $n!/(\Delta_D + 1)$ different permutations so that the critical pair is reversed in the associated linear extensions.

Applying the algorithm in [FT94] we obtain a $k{:}t$-realizer $\mathscr{R} = \{L_1, \ldots, L_t\}$ of P with $t = n!$ and $k = n!/(\Delta_D + 1)$. Moreover, we can efficiently pick uniformly at random one linear extension from $\mathscr{R}$: generate uniformly at random one permutation of jobs (e.g. by using Knuth's shuffle algorithm [Knu69]) and transform it into a linear extension with the described properties by using the algorithm in [FT94]. As described below, the algorithm can be derandomized by using the standard method of conditional probabilities. Finally observe that we can repeat a similar analysis by using Δ_U instead of Δ_D: let $\mathbf{P}^d = (N, P^d)$, where $P^d = \{(i, j) : (j, i) \in P\}$ then $\Delta_D(P^d) = \Delta_U(P)$ and a $k{:}t$-realizer $\mathscr{R}^d = \{L_1^d, \ldots, L_t^d\}$ of $\mathbf{P}^d$ gives a $k{:}t$-realizer $\mathscr{R} = \{L_1, \ldots, L_t\}$ of $\mathbf{P}$, where $L_i = \{(i, j) : (j, i) \in L_i^d\}$. □

We remark that it is necessary to use *fractional* dimension for obtaining the above result. To see this, consider the *incidence poset* $\mathbf{P}(G) = (N, P)$ defined as follows: given an undirected graph $G(V, E)$, let $N = V \cup E$ and for every $v \in V$ and $e = \{v_1, v_2\} \in E$, put $(v, e) \in P$ if and only if $v \in \{v_1, v_2\}$. Since every edge is adjacent to only two vertices, Δ_D is bounded by 2. For K_n the complete graph on n nodes, Spencer [Spe71] showed that $\dim(\mathbf{P}(K_n)) = \Theta(\log \log n)$ whereas from the above discussion we have $\mathrm{fdim}(\mathbf{P}(K_n)) \leq 1 + \min\{\Delta_U, \Delta_D\} = 3$.

Derandomization for Bounded Degree Posets

We let $V_{1/2}$ be the set of vertices with value $1/2$ in the optimal solution to the [VC-LP] formulation of the scheduling problem (see Section 3.4).

We consider the case when $\Delta_D \leq \Delta_U$. The case when $\Delta_U < \Delta_D$ is symmetric and omitted. It suffices to compute a permutation that gives a linear extension whose associated independent set has value at least

$$\frac{w(V_{1/2})}{\Delta_D + 1}.$$

We already mentioned that any incomparable pair (a, b) can be associated with at least one critical pair (x, y) so that a linear extension reversing (x, y) also reverses (a, b) [Tro92]. For simplicity, we associate every incomparable pair with *exactly* one critical pair such that the above condition holds and we denote by $C_{(x,y)}$ the set of incomparable pairs associated to the critical pair (x, y). Note that by convention we have $(x, y) \in C_{(x,y)}$ and the set $\{C_{(x,y)} : (x, y) \text{ is a critical pair}\}$ forms a partition of the incomparable pairs. From the proof of Theorem 3.5.4, it follows that the probability that any critical pair (x, y) (and thus the incomparable pairs in $C_{(x,y)}$) are reversed by a linear extension L obtained from a uniformly picked permutation is at least

$$N_{(x,y)}/D_{(x,y)},$$

where at the beginning

$$N_{(x,y)} := 1 \text{ and } D_{(x,y)} := \begin{cases} |D(y)| + 2, & \text{if } |D(y)| < \Delta_D \\ |D(y)| + 1, & \text{if } |D(y)| = \Delta_D \end{cases}.$$

Starting from the first position of the permutation, we consider the n possibilities corresponding to placing any job at that position and retain the best one. Then we remove the job that has been placed at the first position and continue with the second position, by considering the remaining $n - 1$ jobs, and so forth until the end of the permutation. Each time we consider a possibility, we update the probabilities accordingly. For example, if we decide to put job j at position i then the probability to reverse the incomparable pairs in $C_{(x,y)}$ is updated as follows.

- Set $N_{(x,y)} := 0$ if $x = j$, $|D(y)| < \Delta_D$ and y has not been placed at a previous location;
- Set $N_{(x,y)} := 0$ if $x = j$ and there exists a $z \in D(y)$ that has not been placed at a previous location;
- Set $D_{(x,y)} := D_{(x,y)} - 1$, if ($j = y$ and $D(y) < \Delta_D$) or $j \in D(y)$.

With the updated probabilities we can compute the associated value and retain the best choice.

3.5.4 Convex Bipartite Precedence Constraints

In this section we consider $1|prec|\sum w_j C_j$ for which the precedence constraints form a so called convex bipartite order. For this class of partial orders, we show how to construct a realizer of size 3 in polynomial time. By Theorem 3.3.4, this gives a $(1+\frac{1}{3})$-approximation algorithm.

A *convex bipartite order* $\mathbf{P} = (N = J^- \cup J^+, P)$ is defined as follows (see also Figure 3.7).

1. *The set of jobs are divided into two disjoint sets* $J^- = \{1, \ldots a\}$ *and* $J^+ = \{a+1, \ldots, n\}$*, called the* minus-jobs *and* plus-jobs*, respectively.*

2. *For every* $k = a+1, \ldots, n$ *there are two indices* $l(k)$ *and* $r(k)$ *with* $1 \leq l(k) \leq r(k) \leq a$ *such that* $(i,k) \in P$ *if and only if* $l(k) \leq i \leq r(k)$ (bipartiteness and convexity).

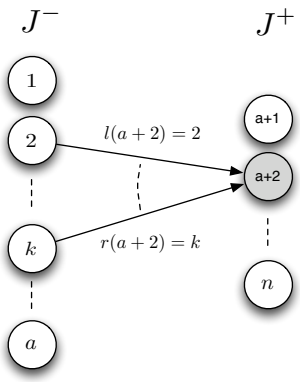


Figure 3.7: An example of a convex bipartite order. Only precedence constraints to $(a+2)$ are depicted.

It is not hard to check that convex bipartite orders can be recognized in polynomial time. Indeed, it is identical to the problem of deciding if the rows of a $(0,1)$-matrix can be permuted so as to make the 1's in each column appear consecutively. Fulkerson & Gross first gave a polynomial algorithm for this problem [FG65], which was later improved to linear time algorithms [BL76; MPT98]. Moreover, the class of convex bipartite orders forms a proper subset of the class of general bipartite orders, and a proper superset of the class of strong bipartite orders [Möh89]. Lemma 3.5.5 states that the class of convex bipartite orders has dimension at most 3. This is indeed a tight bound, since a bipartite order $\mathbf{P}$ is 2-dimensional if and only if it is a strong bipartite order [Möh89].

Finally, we note that $1|prec|\sum w_jC_j$ with strong bipartite orders is solvable in polynomial time [Möh89; CS05; AM09].

Lemma 3.5.5 *Given a convex bipartite order $\mathbf{P} = (N, P)$, a realizer of size three can be computed in polynomial time.*

The proof of this lemma can be found in the next subsection. Theorem 3.3.4 and Lemma 3.5.5 give us the following result.

Theorem 3.5.6 *Problem $1|prec|\sum w_jC_j$ for which the precedence constraints form a convex bipartite order has a (1+ 1/3)-approximation algorithm.*

Proof of Lemma 3.5.5

We create a realizer of size three for a given convex bipartite poset. In the sequel, we sometimes stress that a job j is a plus- or minus-job by writing j^+ and j^-, respectively. We also assume, without loss of generality, that the plus-jobs are numbered such that $i^+ < j^+$ if and only if $l(i^+) \leq l(j^+)$ (breaking ties arbitrarily).

Given a convex bipartite poset $\mathbf{P} = (N, P)$, we partition its incomparable pairs into three sets E_1, E_2, and E_3 as follows (see also Figure 3.8):

- For two incomparable minus-jobs k^- and ℓ^-, with $k < \ell$, we let (ℓ^-, k^-) and (k^-, ℓ^-) be members of E_1 and E_2, respectively.
- For two incomparable plus-jobs q^+ and r^+, with $q < r$, we let (q^+, r^+) and (r^+, q^+) be members of E_1 and E_3, respectively.
- A pair of incomparable jobs $(k^-, q^+) \in \text{inc}(\mathbf{P})$ is a member of E_1.
- A pair of incomparable jobs $(q^+, k^-) \in \text{inc}(\mathbf{P})$ is a member of E_2 if there exists a plus-job r^+, with $r > q$, so that $(k^-, r^+) \in P$; otherwise (if no such plus-job exists) (q^+, k^-) is a member of E_3.

The following lemma is a direct consequence of the definitions of E_1, E_2, and E_3.

Lemma 3.5.7 *Let $\mathbf{P}$ be a convex bipartite order then the sets E_1, E_2, and E_3 form a partition of $inc(\mathbf{P})$.*

Moreover, we have

Lemma 3.5.8 *Let $\bar{E}_1 = E_1 \cup P$, $\bar{E}_2 = E_2 \cup P$, and $\bar{E}_3 = E_3 \cup P$. Then $\bar{E}_1, \bar{E}_2$, and $\bar{E}_3$ are extensions of P.*

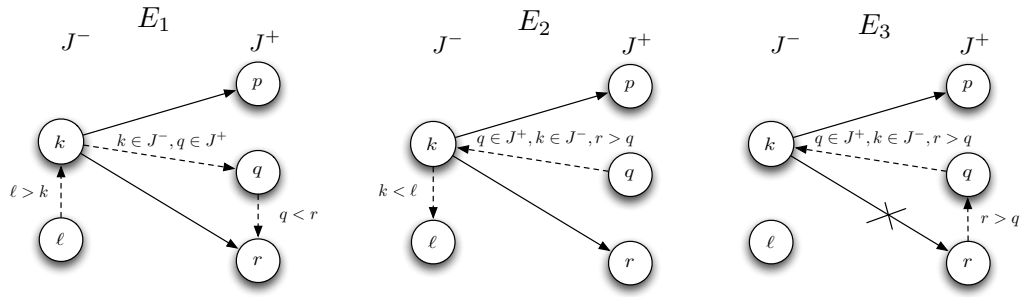


Figure 3.8: Solid edges correspond to precedence constraints, whereas dashed edges are examples of the relations in the different extensions. In this example we assume that $k < \ell$ and $p < q < r$.

Proof. By the definition of $\bar{E}_i$, it follows that if $(a, b) \in P$ then $(a, b) \in \bar{E}_i$, where $i = 1, 2, 3$. Moreover, it is easy to see (Figure 3.8) that the sets $\bar{E}_1$ and $\bar{E}_3$ do not contain cycles, i.e., are valid extensions of P.

Now, suppose toward contradiction that $\bar{E}_2$ is not a valid extension, i.e., contains a cycle $C = \{(j_1, j_2), (j_2, j_3), \ldots, (j_k, j_1)\}$. By the definition of E_2 we have $C \cap P \neq \emptyset$ and thus $C \cap (J^+ \times J^-) \neq \emptyset$. Let $i^- \in J^-$ be the minus-job with largest index in the cycle, i.e., there does not exist a $k > i$ such that $k \in J^-$ is part of the cycle. Then $(i^-, j^+) \in P \cap C$ and $(j^+, m^-) \in C$ for some jobs $j \in J^+$ and $m \in J^-$, where $m < i$. However, this implies that there exists an $\ell > j$ such that $(m^-, \ell^+) \in P$ (recall the definition of E_2). Together with convexity and the ordering of plus-jobs this implies $(m^-, j^+) \in P$, which contradicts the existence of $(j^+, m^-) \in C$. □

Let L_1, L_2, and L_3 be any linear extensions of $\bar{E}_1, \bar{E}_2$, and $\bar{E}_3$, respectively. That $\mathscr{R} = \{L_1, L_2, L_3\}$ is a realizer follows from the facts that all incomparable pairs are reversed (Lemma 3.5.7), and that $\bar{E}_1, \bar{E}_2$, and $\bar{E}_3$ are valid extensions of $\mathbf{P}$ (Lemma 3.5.8). Furthermore, all steps involved in creating $\mathscr{R}$ can clearly be accomplished in polynomial time.

3.5.5 Lexicographic Sums

In this section we show how to use previous results to obtain approximation algorithms for new ordered sets. The construction we use here, *lexicographic sums*, comes from a very simple pictorial idea (see [Tro92] for a more comprehensive discussion). Take a poset $\mathbf{P} = (N, P)$ and replace each of its points $x \in N$ with a partially ordered set $\mathbf{Q_x}$ such that each element of $\mathbf{Q_x}$ has the same relation to points outside it as x had before the replacement. A more formal definition

follows. For a poset $\mathbf{P} = (N, P)$ and a family of posets $\mathscr{S} = \{(Y_x, Q_x) \mid x \in N\}$ indexed by the elements in N, the lexicographic sum of $\mathscr{S}$ over (N, P), denoted $\sum_{x\in(N,P)}(Y_x, Q_x)$ is the poset (Z, R) where $Z = \{(x, y) \mid x \in N, y \in Y_x\}$ and $(x_1, y_1) \leq (x_2, y_2)$ in R if and only if one of the following two statements holds:

1. $x_1 < x_2$ in P.

2. $x_1 = x_2$ and $y_1 \leq y_2$ in Q_{x_1}.

We call $\mathscr{P} = \{\mathbf{P}\} \cup \mathscr{S}$ the *components* of the lexicographic sum. A lexicographic sum is *trivial* if $|N| = 1$ or if $|Y_x| = 1$ for all $x \in N$. A poset is *decomposable with respect to lexicographic sums* if it is isomorphic to a non-trivial lexicographic sum. Moreover, a decomposition of a poset with respect to lexicographic sums can be done in $O(n^2)$ time (see e.g. [Möh89])

In case the precedence constraints of every component admit an efficiently samplable realizer, we observe that this translates into a randomized approximation algorithm:

Theorem 3.5.9 *Problem* $1|prec|\sum w_j C_j$*, whenever precedence constraints form a lexicographic sum whose components admit efficiently samplable k:t-realizers, has a randomized* $(2 - \frac{2}{t/k})$*−approximation algorithm.*

Proof. Let $\mathbf{P} = (N, P)$ with $N = \{1, 2, \ldots, n\}$ and $\mathbf{Q}_1, \mathbf{Q}_2, \ldots, \mathbf{Q}_n$ be the components of the lexicographic sum $\sum_{x\in(N,P)} \mathbf{Q}_x$, where $\mathbf{Q}_x = (Y_x, Q_x)$.

Given a linear extension L_p of P and linear extensions $L_1, L_2, \ldots, L_n$ of the components $Q_1, Q_2, \ldots, Q_n$, we can construct a linear extension of $\sum_{x\in(N,P)} \mathbf{Q}_x$ by adding the relation $(x, y) \leq (x', y')$ between two incomparable elements of $\sum_{x\in(N,P)} \mathbf{Q}_x$, if either (i) $x < x'$ in L_p or (ii) $x = x'$ and $y < y'$ in L_x.

Now, suppose that all components have efficiently samplable *k:t*-realizers. Then we can sample each *k:t*-realizer independently to obtain linear extensions of all components, which in turn define a linear extension L of the lexicographic sum $\sum_{x\in(N,P)} \mathbf{Q}_x$. It is not hard to see that each incomparable pair of $\sum_{x\in(N,P)} \mathbf{Q}_x$ is reversed in L with probability at least k/t. Hence, the lexicographic sum has an efficiently samplable *k:t*-realizer. This together with Theorem 3.4.1 concludes the proof. □

Finally, we point out that, if the approximation algorithm for each component can be derandomized, this yields a derandomized approximation algorithm for the lexicographic sum. In particular this can be done when all components have low dimension.

3.6 Scheduling with Interval Orders is NP-Hard

In this section we prove Theorem 3.1.1, i.e., that $1|prec|\sum w_jC_j$ remains NP-hard even in the special case of interval order precedence constraints. To prove this, we exploit the vertex cover nature of problem $1|prec|\sum w_jC_j$: finding an optimum solution to a scheduling instance S, where precedence constraints are given by an interval order $\mathbf{P}$, is equivalent to solving the minimum weighted vertex cover problem in the graph $G^S_{\mathbf{P}}$ (see Section 3.3).

We provide a reduction to $1|prec|\sum w_jC_j$ with interval order precedence constraints from the NP-hard problem of finding a minimum vertex cover in a graph with bounded degree 3 [GJS76]. More precisely, given a connected graph $G=(V,E)$ with bounded degree 3[3], we construct a scheduling instance S (with interval order precedence constraints) so that the graph $G^S_{\mathbf{P}}$ has a weighted vertex cover with value less than $m+c+1$ if and only if G has a vertex cover of size at most m , where c is a fixed value that depends on G (see Equation (3.9)). We present the construction of S in two stages.

Stage 1 (Tree-layout of the graph)

Starting from an arbitrary, but fixed vertex $s \in V$, we obtain a tree $T=(V,E_T)$, with $E_T \subseteq E$, rooted at s by using, for example, breadth-first search. Furthermore, we number the vertices of T top-down and left-right. Figure 3.9 shows the breadth-first search tree T for K_4.

Let $G'=(V',E')$ be the graph obtained from T in the following way (see also Figure 3.9). For each vertex v_i in T we add two new vertices u_i^2, u_i^1 and edges $\{u_i^2,u_i^1\},\{u_i^1,v_i\}$. Furthermore, for each edge $\{v_i,v_j\} \in E \setminus E_T$ with $i<j$ we add vertices e_{ij}^1, e_{ij}^2 and edges $\{v_i,e_{ij}^1\},\{e_{ij}^1,e_{ij}^2\},\{e_{ij}^2,u_j^2\}$.

The following lemma relates the minimum unweighted vertex covers of G and G'. Its proof is similar to the proof in [AK00] for proving APX-completeness of vertex cover on cubic graphs.

Lemma 3.6.1 *Let $C_* \subseteq V$ and $C'_* \subseteq V'$ be minimum vertex cover solutions to G and G', respectively. Then $|C_*| = |C'_*| - |V| - |E \setminus E_T|$.*

Proof. On the one hand, it is easy to see that from every vertex cover $C \subseteq V$ of G we can construct a vertex cover $C' \subseteq V'$ of G' of size exactly $|C|+|V|+|E\setminus E_T|$. In C' we include

[3]A similar reduction can be seen to work without the requirement that the reduction is from graphs with bounded degree at most 3. However, this requirement can be done without loss of generality (as the vertex cover problem remains NP-hard on these graphs) and simplifies the reduction.

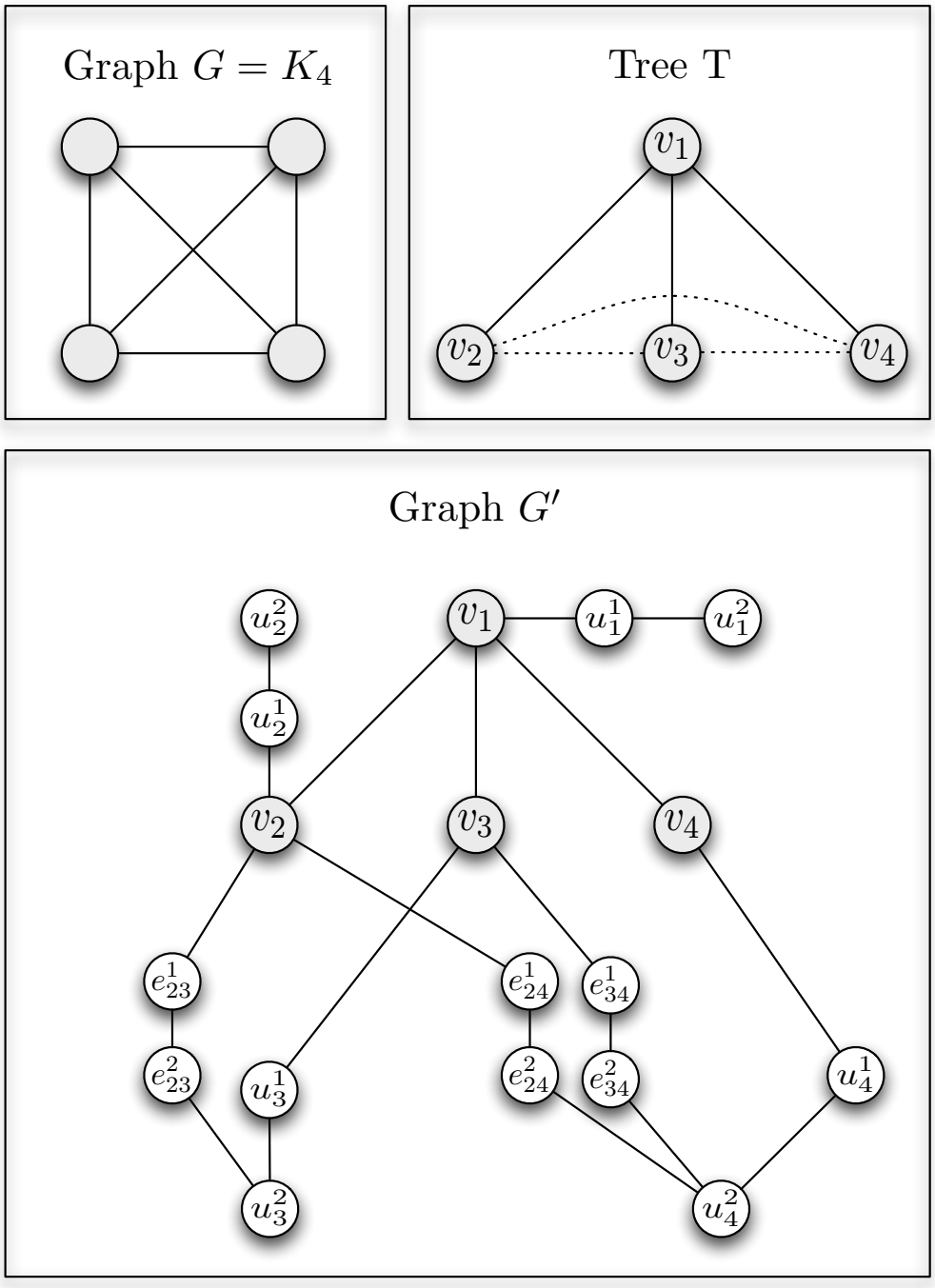


Figure 3.9: The breadth-first search tree $T = (V, E_T)$ for the graph $G = K_4$, and the graph G'. In the drawing of T the solid edges belong to E_T.

(i) every vertex in C;

(ii) u_i^1 for all i with $v_i \in V \setminus C$;

(iii) u_i^2 for all with $v_i \in C$;

(iv) e_{ij}^1 for each $\{v_i, v_j\} \in E \setminus E_T$ with $v_i \in V \setminus C$; and

(v) e_{ij}^2 for each $\{v_i, v_j\} \in E \setminus E_T$ with $v_i \in C$.

On the other hand, given a vertex cover $C' \subseteq V'$ of G' we transform it into a vertex cover $C \subseteq V$ of G in the following manner. Suppose there exist $v_i, v_j \in V$ with $i < j$ such that $\{v_i, v_j\} \in E$ and $v_i \notin C', v_j \notin C'$. Since C' is a feasible vertex cover of G' we have that $\{v_i, v_j\} \in E \setminus E_T$ and either $\{e_{ij}^1, e_{ij}^2, u_j^1\} \subseteq C'$ or $\{e_{ij}^1, u_j^2, u_j^1\} \subseteq C'$. Thus we can obtain a vertex cover $C'' \subseteq V'$ of G' with $|C''| \leq |C'|$ by letting $C'' = (C' \setminus \{u_j^1, e_{ij}^2\}) \cup \{v_j, u_j^2\}$. Repeating this procedure will result in a vertex cover $C''' \subseteq V'$ of G' with $|C'''| \leq |C'|$ such that $C = C''' \cap V$ is a feasible vertex cover of G. Furthermore, it is easy to see that $|C| \leq |C'''| - |V| - |E \setminus E_T|$. □

Stage 2 (Construction of scheduling instance)

Given a vertex cover graph $G = (V, E)$ and its corresponding tree $T = (V, E_T)$, we now proceed by constructing the instance S of $1|prec|\sum w_j C_j$ with precedence constraints in the form of an interval order. We will do so by defining several kinds of jobs along with several properties. Let k be a large value to be determined later.

Tree-jobs. Instance S has $|V| + 1$ so-called *tree-jobs* referred to as $s_0, s_1, \ldots, s_{|V|}$. Their interval representations, processing times and weights are defined as follows (see also Figure 3.11).

Job	Interval Representation	Processing Time	Weight
s_0	[-1,0]	1	0
s_1	[0, 1]	$1/k$	1
$s_j, j = 2, \ldots, \lvert V \rvert$	$[i, j]$, where $\{v_i, v_j\} \in E_T$, $i < j$	$1/k^j$	k^i

Recall that two jobs s_i and s_j are incomparable if their interval presentations overlap (even marginally). Let s_i and s_j with $i < j$ be two tree-jobs with interval representations $[a, i]$ and $[b, j]$ respectively. By the construction of the

scheduling instance S we have $p_{s_i} \leq 1/k^i$ and $w_{s_j} \leq k^b$. As i and b are integers, $p_{s_i} \cdot w_{s_j} = 1$ or $p_{s_i} \cdot w_{s_j} \leq 1/k$, if i and j are incomparable. Moreover, since $p_{s_i} \cdot w_{s_j} = 1$ only if $i = b$ it follows that $p_{s_i} \cdot w_{s_j} = 1$ only if either v_i is the parent of v_j in T or $(i, j) = (0, 1)$.

Now let

$$D_s := \{(s_0, s_1)\} \cup \{(s_i, s_j) : v_i \text{ is the parent of } v_j \text{ in } T\}.$$

By the above discussion we have the following lemma.

Lemma 3.6.2 *A pair of incomparable jobs (a,b) has $p_a \cdot w_b = 1$ if $(a, b) \in D_s$; otherwise if $(a, b) \notin D_s$ then $p_a \cdot w_b \leq 1/k$.*

Let $\mathbf{P}$ be the interval order defined on the tree-jobs and let $G^S_{\mathbf{P}}$ be the graph associated to the scheduling problem consisting of the tree-jobs. By the fact that an interval order does not contain any $\mathbf{2}+\mathbf{2}$ structures (depicted in Figure 3.10) as induced posets [Tro92], graph $G^S_{\mathbf{P}}$ has only two types of edges (see (i) and (ii) in Figure 3.4):

(i) Two vertices (i, j) and (j, i) are adjacent.

(ii) Two vertices (i, k) and (k, j) are adjacent if $(i, j) \in \mathbf{P}$.

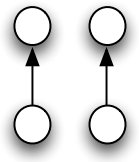


Figure 3.10: A $\mathbf{2}+\mathbf{2}$ poset.

We are now ready to relate the tree-jobs to the tree T. Two graphs G and H are said to be isomorphic if there exists a bijection f between the vertex sets of G and H so that two vertices u and v of G are adjacent in G if and only if $f(u)$ and $f(v)$ are adjacent in H.

Lemma 3.6.3 *Let $T_{\mathbf{P}}$ be the subgraph of $G^S_{\mathbf{P}}$ induced by the vertex subset D_s. Then $T_{\mathbf{P}}$ and T are isomorphic.*

Proof. See Figure 3.11 for an example of the graph $T_{\mathbf{P}}$. We relate the two graphs $T_{\mathbf{P}}$ and T by the bijection $f : D_s \to V$, defined by $f((s_i, s_j)) = v_j$. By the fact that $\mathbf{P}$ is an interval order (see the discussion above) together with the definition of D_s, we have that two incomparable pairs (s_i, s_j) and (s_k, s_ℓ) in D_s with $i < j \leq k < \ell$ are adjacent if and only if $j = k$ and $(s_i, s_\ell) \in P$. The proof is now completed by noting that

(i) if $\{(s_i, s_j), (s_j, s_\ell)\}$ is an edge in $T_{\mathbf{P}}$ then $\{f(s_i, s_j), f(s_j, s_\ell)\} = \{v_j, v_\ell\}$ is an edge in T, because, by definition, v_j is the parent of v_ℓ in T; and

(ii) if v_i and v_j are adjacent in T then $f^{-1}(v_i) = (s_a, s_i)$ and $f^{-1}(v_j) = (s_i, s_j)$, for some $a < i$, and as s_a's interval representation is to the left of s_j's interval representation no linear extension of $\mathbf{P}$ reverses both (s_a, s_i) and (s_i, s_j), i.e., they are adjacent in $T_{\mathbf{P}}$.

□

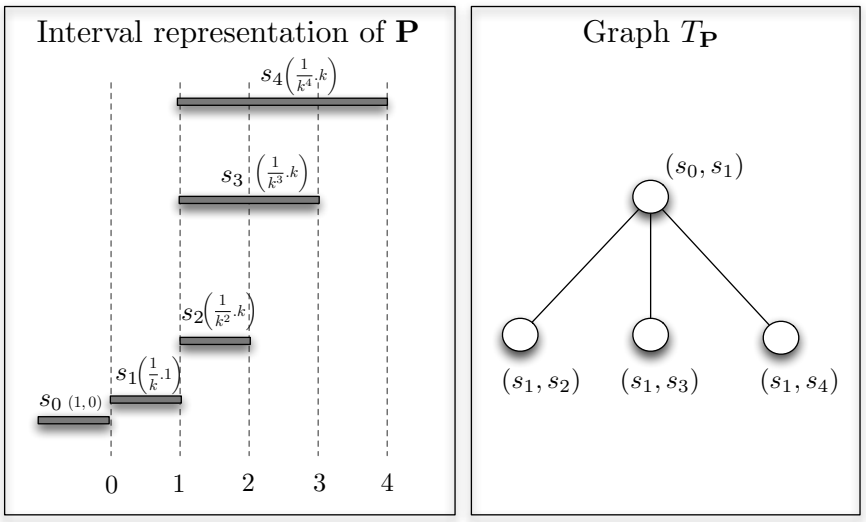


Figure 3.11: The interval representation of the tree-jobs obtained from K_4 (the processing time p and weight w of a job is depicted by the tuple (p, w)); $T_{\mathbf{P}}$ is the subgraph of $G^S_{\mathbf{P}}$ induced by the vertex subset D_s.

Slack- and edge-jobs. Instance S has $2|V|$ *slack-jobs*, referred to as $m_1, m_2, \ldots, m_{|V|}$ and $e_1, e_2, \ldots, e_{|V|}$. There is also an *edge-job* b_{ij} for each $\{v_i, v_j\} \in E \setminus E_T$. Their interval representations, processing times and weights are defined as follows (see also Figure 3.12).

Job	Interval Representation	Processing Time	Weight
$m_i, i = 1, \ldots, \lvert V\rvert$	$[i - \frac{1}{2}, \lvert V\rvert + i]$	$1/k^{(\lvert V\rvert+i)}$	k^i
$e_i, i = 1, \ldots, \lvert V\rvert$	$[\lvert V\rvert + i, \lvert V\rvert + i + 1]$	0	$k^{(\lvert V\rvert+i)}$
b_{ij}, where $\{v_i, v_j\} \in E \setminus E_T, i < j$	$[i, j - \frac{1}{2}]$	$1/k^j$	k^i

Let $D = D_s$
$\cup \{(s_i, m_i), (m_i, e_i) : i = 1, 2, \ldots, |V|\}$
$\cup \{(s_i, b_{ij}), (b_{ij}, m_j) : \{v_i, v_j\} \in E \setminus E_T, i < j\}$

Similar to Lemma 3.6.2 we have

Lemma 3.6.4 *A pair of incomparable jobs (a, b) has $p_a \cdot w_b = 1$ if $(a, b) \in D$; otherwise if $(a, b) \notin D$ then $p_a \cdot w_b \leq 1/k$.*

Proof. For an overview see Figure 3.12. By Lemma 3.6.2 we have that if $a = s_i$ and $b = s_j$ then $p_a \cdot w_b = 1$ if $(a, b) \in D$ and otherwise if $(a, b) \notin D$ then $p_a \cdot w_b \leq 1/k$. Since $p_{e_i} = 0$ for all $i = 1, \ldots |V|$ the incomparable pairs of the form (e_i, b) have weight $p_{e_i} \cdot w_b = 0$. Moreover, it is easy to see that an incomparable pair of the form (a, e_i) has $p_a \cdot w_{e_i} = 1$, if $a = m_i$ and $p_a \cdot w_{e_i} \leq 1/k$ otherwise. Similarly, an incomparable pair of the form (m_i, b) has $p_{m_i} \cdot w_b = 1$, if $b = e_i$ and $p_{m_i} \cdot w_b \leq 1/k$ otherwise.

Recall that a tree-job s_i has processing time $1/k^i$ and has interval representation $[x, i]$ for some x. Any other job j with interval representation $[q, r]$ has weight at most $k^{\lceil q \rceil}$. Hence, by the definition of the intervals, an incomparable pair (s_i, b), where b is no tree-job, has $p_{s_i} w_b = 1$, if $b = m_i$ or $b = b_{ij}$ for some $j > i$; otherwise we have that $p_{s_i} \cdot w_b \leq 1/k$. By a similar argument we have that an incomparable pair (b_{ij}, b) has $p_{b_{ij}} w_b = 1$ if $b = m_j$ and $p_{b_{ij}} w_b \leq 1/k$ otherwise. □

Let $\mathbf{P}$ be the interval order defined on the tree-, slack-, and edge-jobs; and let $G^S_{\mathbf{P}}$ be the graph associated to the scheduling problem S that consists of these jobs. The following lemma, whose proof can be found in the next subsection, motivates our construction.

Lemma 3.6.5 *Let $G'_{\mathbf{P}} = (D, E_{\mathbf{P}})$ be the subgraph of $G^S_{\mathbf{P}}$ induced by the vertex subset D. Then $G'_{\mathbf{P}}$ and G' are isomorphic.*

By Lemma 3.6.4, each incomparable pair of jobs $(i, j) \notin D$ satisfies $p(i) \cdot w(j) \leq 1/k$. Let n be the number of jobs in the scheduling instance S and select k to be $n^2 + 1$. Let $C^S_{\mathbf{P}}$ and $C'_{\mathbf{P}}$ be minimum vertex covers of $G^S_{\mathbf{P}}$ and $G'_{\mathbf{P}}$ and denote their respective values by $w(C^S_{\mathbf{P}})$ and $w(C'_{\mathbf{P}})$. Since $G'_{\mathbf{P}}$ is unweighted we have $w(C'_{\mathbf{P}}) = |C'_{\mathbf{P}}|$. By the selection of k and Lemma 3.6.4, we have

$$\sum_{(i,j) \in \mathrm{inc}(P) \setminus D} p_i w_j < 1, \text{ and thus } w(C'_{\mathbf{P}}) = \left\lfloor w(C^S_{\mathbf{P}}) \right\rfloor .$$

Since $G'_{\mathbf{P}}$ and G' are isomorphic (Lemma 3.6.5) we have by Lemma 3.6.1 that the original graph G has an optimal vertex cover of size at most m if and only if $\left\lfloor w(C^S_{\mathbf{P}}) \right\rfloor \leq m + c$, where

$$c = |V| + |E \setminus E_T|. \tag{3.9}$$

Finally, we note that, as finding a minimum schedule of S is equal to finding a minimum vertex cover of $G_{\mathbf{P}}^{S}$, which in turn is as hard as finding a minimum vertex cover in a graph G with bounded degree 3, we have that $1|prec|\sum w_j C_j$ remains (weakly) NP-hard in the special case of interval order precedence constraints.

3.6.1 Proof of Lemma 3.6.5

We relate the two graphs $G'_{\mathbf{P}}(D, E_{\mathbf{P}})$ and $G'(V', E')$ by the bijection $f : D \to V'$, defined as follows.

$$f : \begin{cases} (s_i, s_j) & \longmapsto & v_j, \\ (s_i, m_i) & \longmapsto & u_i^1, \\ (m_i, e_i) & \longmapsto & u_i^2, \\ (s_i, b_{ij}) & \longmapsto & e_{ij}^1, \\ (b_{ij}, m_j) & \longmapsto & e_{ij}^2. \end{cases}$$

To complete the proof we need to show that

$$\{(a,b),(c,d)\} \in E_{\mathbf{P}} \Leftrightarrow \{f((a,b)), f((c,d))\} \in E'. \tag{3.10}$$

If a, b, c, and d are all tree-jobs and hence $f((a,b)) = v_i, f((c,d)) = v_j$, for some i and j, then Lemma 3.6.3 guarantees (3.10). The remaining cases follow from a simple case analysis that is presented below for the sake of completeness (see also Figure 3.12 for an example).

On the one hand suppose $\{(a,b),(c,d)\} \in E_{\mathbf{P}}$. By the fact that $\mathbf{P}$ is an interval order (see discussion before Lemma 3.6.3) together with the definition of D, we can assume that $b = c$ and $a \neq d$. Now consider the possible cases of $\{(a,b),(b,d)\}$ (where not all of them are tree-jobs).

Case $a = s_i, b = s_j, d = b_{jk}, i < j < k$**:** Then $f((s_i,s_j)) = v_j$ and $f((s_j,b_{jk})) = e_{jk}^1$ and by the definition of G' we have $\{v_j, e_{jk}^1\} \in E'$.

Case $a = s_i, b = s_j, d = m_j, i < j$**:** Then $f((s_i,s_j)) = v_j$ and $f((s_j,m_j)) = u_j^1$ and by the definition of G' we have $\{v_j, u_j^1\} \in E'$.

Case $a = s_i, b = b_{ij}, d = m_j, i < j$**:** Then $f((s_i,b_{ij})) = e_{ij}^1$ and $f((b_{ij},m_j)) = e_{ij}^2$ and by the definition of G' we have $\{e_{ij}^1, e_{ij}^2\} \in E'$.

Case $a = s_i, b = m_i, d = e_i$**:** Then $f((s_i,m_i)) = u_i^1$ and $f((m_i,e_i)) = u_i^2$ and by the definition of G' we have $\{u_i^1, u_i^2\} \in E'$.

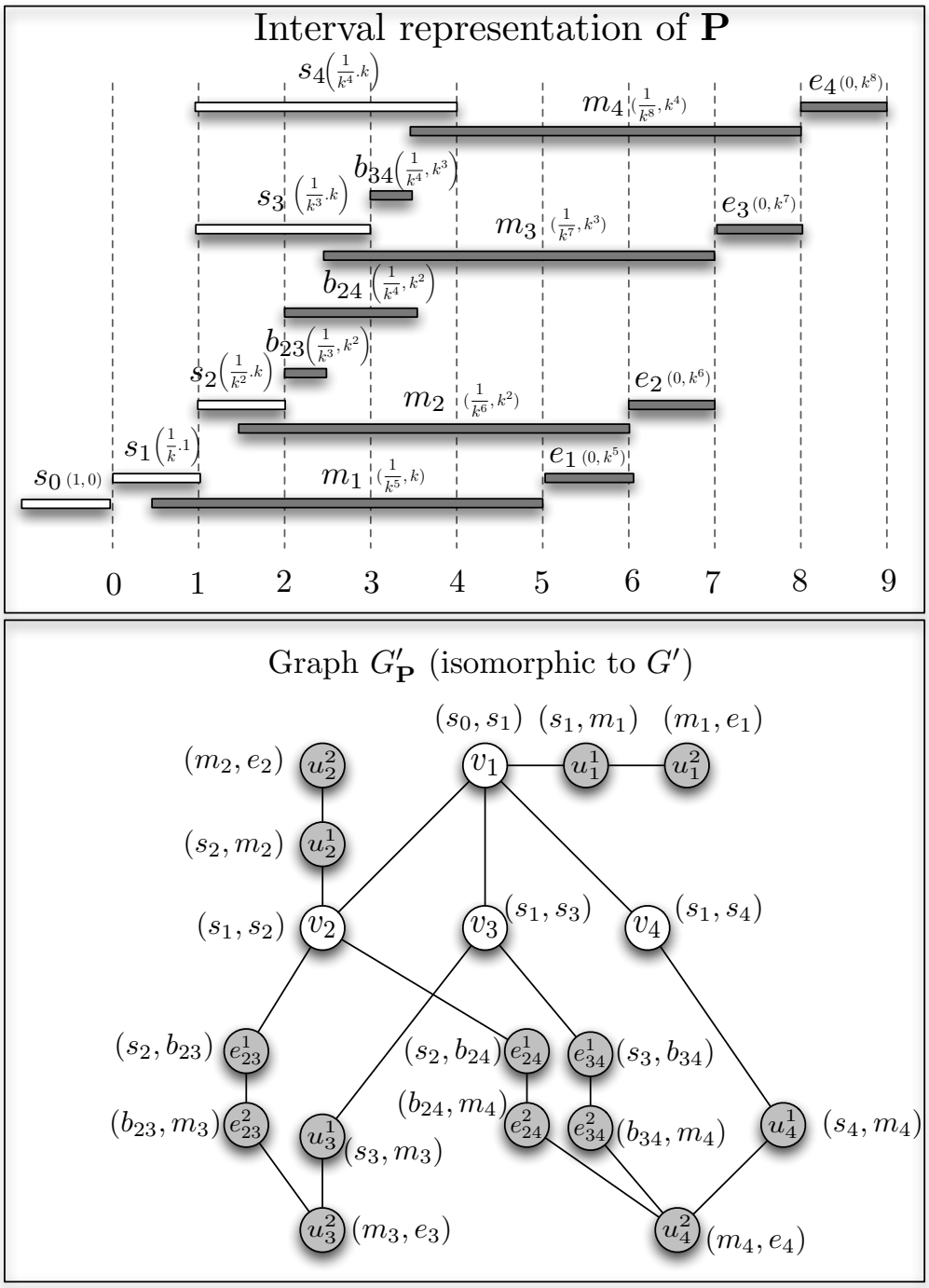


Figure 3.12: The interval representation of the tree-, slack, and edge-jobs obtained from K_4 (the processing time p and weight w of a job is depicted by the tuple (p, w)); $G'_{\mathbf{P}}$ is the subgraph of $G^S_{\mathbf{P}}$ induced by the vertex subset D. The additional intervals and vertices compared to Figure 3.11 are depicted in gray.

Case $a = b_{ij}, b = m_j, d = e_j, i < j$**:** Then $f((b_{ij}, m_j)) = e^2_{ij}$ and $f((m_j, e_j)) = u^2_j$ and by the definition of G' we have $\{e^2_{ij}, u^2_j\} \in E'$.

We have considered all possible cases and it follows that $\{(a,b),(b,d)\} \in E_{\mathbf{P}} \Rightarrow \{f((a,b)), f((b,c))\} \in E'$.

On the other hand, suppose $\{a,b\} \in E'$ and again consider the different possible cases (except the case when $a = v_i$ and $b = v_j$ that was already considered in Lemma 3.6.3).

Case $a = v_i, b = e^1_{ij}, i < j$**:** Then $f^{-1}(v_i) = (s_k, s_i)$ and $f^{-1}(e^{ij}_1) = (s_i, b_{ij})$ for some $k < i < j$. Since s_k's interval representation is completely to the left of b_{ij}'s interval representation in $\mathbf{P}$, the incomparable pairs (s_k, s_i) and (s_i, b_{ij}) cannot be reversed in the same linear extension, i.e., $\{(s_k, s_i), (s_i, b_{ij})\} \in E_{\mathbf{P}}$.

Case $a = e^1_{ij}, b = e^2_{ij}, i < j$**:** Then $f^{-1}(e^1_{ij}) = (s_i, b_{ij})$ and $f^{-1}(e^2_{ij}) = (b_{ij}, m_j)$. Since s_i's interval representation is completely to the left of m_j's interval representation in $\mathbf{P}$ the incomparable pairs (s_i, b_{ij}) and (b_{ij}, m_j) cannot be reversed in the same linear extension, i.e., $\{(s_i, b_{ij}), (b_{ij}, m_j)\} \in E_{\mathbf{P}}$.

Case $a = e^2_{ij}, b = u^2_j, i < j$**:** Then $f^{-1}(e^2_{ij}) = (b_{ij}, m_j)$ and $f^{-1}(u^2_j) = (m_j, e_j)$. Since b_{ij}'s interval representation is completely to the left of e_j's interval representation in $\mathbf{P}$ the incomparable pairs (b_{ij}, m_j) and (m_j, e_j) cannot be reversed in the same linear extension, i.e., $\{(b_{ij}, m_j), (m_j, e_j)\} \in E_{\mathbf{P}}$.

Case $a = u^1_j, b = u^2_j$**:** Then $f^{-1}(u^1_j) = (s_j, m_j)$ and $f^{-1}(u^2_j) = (m_j, e_j)$. Since s_j's interval representation is completely to the left of e_j's interval representation in $\mathbf{P}$ the incomparable pairs (s_j, m_j) and (m_j, e_j) cannot be reversed in the same linear extension, i.e., $\{(s_j, m_j), (m_j, e_j)\} \in E_{\mathbf{P}}$.

Case $a = v_j, b = u^1_j$ **:** Then $f^{-1}(v_j) = (s_i, s_j)$ and $f^{-1}(u^1_j) = (s_j, m_j)$ for some $i < j$. Since s_i's interval representation is completely to the left of m_j's interval representation in $\mathbf{P}$ the incomparable pairs (s_i, s_j) and (s_j, m_j) cannot be reversed in the same linear extension, i.e., $\{(s_i, s_j), (s_j, m_j)\} \in E_{\mathbf{P}}$.

We have considered all possible cases and it follows that $\{v_i, v_j\} \in E' \Rightarrow \{f^{-1}(v_i), f^{-1}(v_j)\} \in E_{\mathbf{P}}$. We have thus proved (3.10), i.e., that the function f defines an isomorphism between $G'_{\mathbf{p}}$ and G'.

3.7 Hardness of Approximation

Here, we prove hardness of approximation results for $1|prec|\sum w_jC_j$. Depending on whether we include the fixed-cost in the objective function, we obtain negative results of different strengths. In Section 3.7.1, we show that the variable-cost is as hard to approximate as the vertex cover problem. For the complete objective function (i.e. the fixed-cost plus the variable-cost), we rule out the existence of a PTAS for $1|prec|\sum w_jC_j$ (Section 3.7.2).

3.7.1 Hardness of Variable Part

We show Theorem 3.1.2, i.e., that approximating the variable-cost of scheduling problem $1|prec|\sum w_jC_j$ is equivalent to approximating the vertex cover problem. Theorem 3.3.1 implies that minimizing the variable-cost of $1|prec|\sum w_jC_j$ is a special case of vertex cover and therefore is not harder to approximate. It remains to prove the other direction. We do so by proving that, for any graph G, we can construct a scheduling instance for which minimizing the variable-cost is essentially equal to finding a minimum vertex cover of G.

Theorem 3.7.1 *Approximating the variable cost of* $1|prec|\sum w_jC_j$ *is as hard as approximating vertex cover.*

Proof. Let $G = (V, E)$ be a vertex cover instance and let $n = |V|$. We will construct a scheduling instance S as follows. The construction is inspired by the so-called *adjacency poset* of G. Let $r \geq 1, \epsilon > 0$ and choose $k > n^2r/\epsilon$. For each vertex $v_i \in V$, there are two jobs v_i' and v_i''. The processing time and weight for a job v_i' are $1/k^i$ and 0, respectively. Conversely, the processing time and weight for a job v_i'' are 0 and k^i, respectively.

S has the following precedence constraints: for each edge $\{v_i, v_j\} \in E$, the precedence constraints $v_i' < v_j''$ and $v_j' < v_i''$. Finally, we add $v_i' < v_j''$ for every i, j with $i < j$. See Figure 3.13 for a small example.

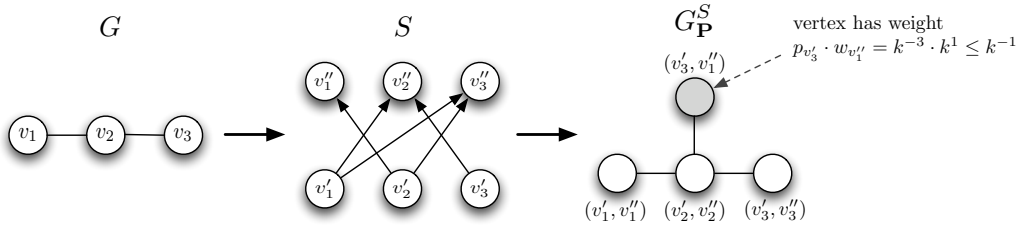


Figure 3.13: The transformation of a graph G.

Now consider the graph $G_{\mathbf{p}}^S$. It has at most n^2 vertices. The n vertices corresponding to the incomparable pairs (v_i', v_i'') have weight 1. All other vertices have weight at most $1/k$, which by the choice of k is very small. The total weight of these "light" vertices is no more than n^2/k.

Moreover, the subgraph induced by the vertices with weight 1 is isomorphic to G. To see this, recall that there is an edge between the vertices (v_i', v_i'') and (v_j', v_j'') in $G_{\mathbf{p}}^S$ if and only if both precedence constraints $v_i' \to v_j''$ and $v_j' \to v_i''$ are present in S. This in turn is the case if and only if $\{v_i, v_j\} \in E$.

Using the connection between S and $G_{\mathbf{p}}^S$ provided by Theorem 3.3.1 and the close relation between $G_{\mathbf{p}}^S$ and G, it is easy to see that an r-approximation algorithm for the variable-cost of $1|prec|\sum w_j C_j$ would imply an approximation algorithm for vertex cover with approximation ratio $r(1 + n^2/k) < (r + \epsilon)$. □

We point out that the above reduction fails to yield inapproximability results if the complete objective function (i.e. the fixed-cost plus the variable-cost) is considered: the fixed cost introduced during the reduction dominates the objective function value, which makes any feasible solution close to optimal. Nevertheless, one can rule out, under some fairly standard assumption, the existence of a PTAS for $1|prec|\sum w_j C_j$ by establishing a connection between the maximum edge biclique problem and $1|prec|\sum w_j C_j$. This is done in the following section.

3.7.2 Ruling out a PTAS

We show a nice relationship between $1|prec|\sum w_j C_j$ and the maximum edge biclique problem (see the proof overview of Theorem 3.1.5 for a definition). This relationship together with our inapproximability result for maximum edge biclique (MEB), given in Chapter 4, yields Theorem 3.1.5, i.e., that the scheduling problem has no PTAS unless SAT can be solved by a (probabilistic) algorithm that runs in time 2^{N^ϵ}, where N is the instance size and $\epsilon > 0$ can be made arbitrarily close to 0.

With an n by n bipartite graph $G = (U, V, E)$, we associate a bipartite scheduling instance S_G with jobs $U \cup V$ and precedence constraints $P = U \times V \setminus E$. The jobs of U have processing time 1 and weight 0, and the jobs of V have processing time 0 and weight 1. See Figure 3.14 for a small example.

The intuition behind the relationship between $1|prec|\sum w_j C_j$ and MEB is best seen by considering the 2D Gantt chart, first introduced by Eastman et al. [EEI64] and later revived by Goemans and Williamson [GW00] to give elegant proofs for various results related to $1|prec|\sum w_j C_j$. In a 2D Gantt chart, we have a horizontal axis of processing time and a vertical axis of weight. For a scheduling instance of the above form, the chart starts at point $(0, n)$ and ends

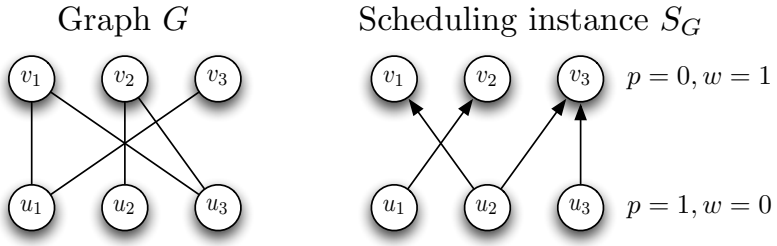


Figure 3.14: An example of a graph G with its associated scheduling instance S_G.

at point $(n, 0)$. A job j is represented by a rectangle of length p_j and height w_j. Hence, a job of U is represented by a horizontal line of length 1 and a job of V is represented by a vertical line of length 1. Any schedule (linear extension of the jobs) is represented in the 2D Gantt chart by placing the corresponding rectangles of the jobs in the order of the schedule such that the startpoint of a job is the endpoint of the previous job (or $(0, n)$ for the first job). The value $\sum_j w_j C_j$ of a schedule is then the area under the "work line" (see the shaded area in Figure 3.15), or equivalently, the area above the work line subtracted from n^2. The relationship to MEB now becomes clear from the following subtle observation: each point (s, t) on the work line of a schedule of S_G defines an edge biclique of G of size $(n - s)t$, by taking the vertices corresponding to the jobs of U that complete after s (there are $n - s$ of them) and the jobs of V that complete before s (there are t of them), see striped area in Figure 3.15. We can thus bound the area above the work line (and the value of an optimal schedule of S_G), in terms of the size of a maximum edge biclique of G.

Formalizing the above intuition we obtain the following result

Lemma 3.7.2 *Let $val(\sigma^*)$ denote the value of an optimal schedule σ^* of S_G. If a maximum edge biclique of G has value an^2 for some $a \in (0, 1]$, then*

$$n^2 - an^2(\ln 1/a + 2) \leq val(\sigma^*) \leq n^2 - an^2.$$

Proof. We start by showing that $\text{val}(\sigma^*) \leq n^2 - an^2$. Let $A \subseteq U, B \subseteq V$ be an edge biclique solution with value $|A| \cdot |B| = an^2$. Consider a schedule σ that schedules the jobs in the order $U \setminus A \to B \to A \to V \setminus B$. The feasibility of such a schedule can be seen by observing that there is no precedence constraints from the jobs in A to the jobs in B. The bound now follows since $\text{val}(\sigma^*) \leq \text{val}(\sigma)$ and

$$\text{val}(\sigma) \leq |U \setminus A| \cdot |B| + |U| \cdot |V \setminus B| = (n - |A|)|B| + n(n - |B|) = n^2 - |A||B| = n^2 - an^2.$$

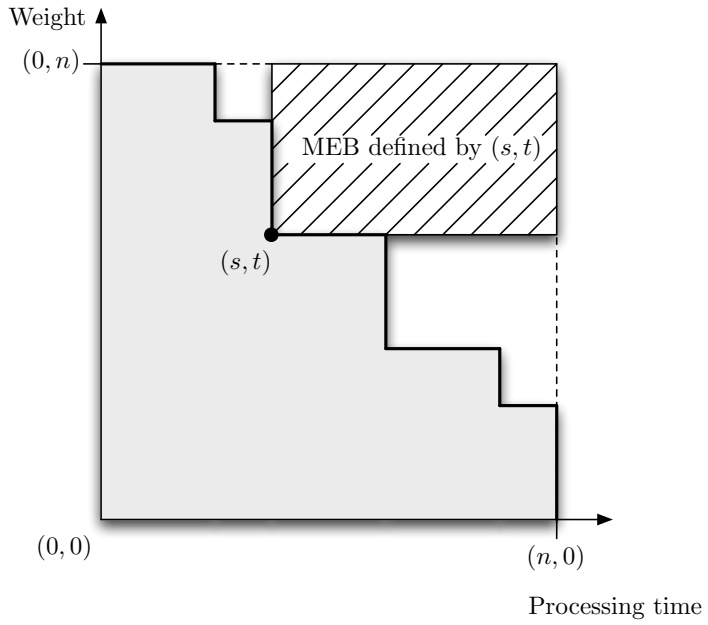


Figure 3.15: 2D Gantt chart representation of a schedule.

To prove the lower bound $n^2 - an^2(\ln 1/a + 2) \leq val(\sigma^*)$ we shall use $\sigma^*(i)$ to denote the total number of jobs of V scheduled before i jobs of U have been scheduled in σ^*. With this notation the value of σ^* (where we let $\sigma^*(n+1) = n$) is

$$\sum_{i=1}^{n} (\sigma^*(i+1) - \sigma^*(i))\, i = n^2 - \sum_{i=1}^{n} \sigma^*(i).$$

Note that in any point of the schedule σ^*, the set of jobs of U that have not been scheduled, say A, has no precedence constraints to the set of jobs of V that have been scheduled, say B. It follows that A and B form an edge biclique of G with value $|A||B|$. As a maximum edge biclique of G has value $a \cdot n^2$, we have that $\sigma^*(i)(n - i + 1) \leq an^2$ for $i = 1, \ldots, n$. Moreover, since $|V| \leq n$ we have that $\sigma^*(i) \leq n$ for $i = 1, \ldots, n$. Using these bounds on $\sigma^*(i)$, it follows that

$$\begin{aligned} n^2 - \sum_{i}^{n} \sigma^*(i) &= n^2 - \sum_{i=1}^{(1-a)n} \sigma^*(i) - \sum_{i=(1-a)n+1}^{n} \sigma^*(i) \\ &\geq n^2 - an^2 \sum_{i=1}^{(1-a)n} \frac{1}{n-i+1} - \sum_{i=(1-a)n+1}^{n} n \\ &= n^2 - an^2(H_n - H_{an}) - an^2. \end{aligned}$$

The statement now follows by the bounds $\ln(n) \leq H_n \leq \ln(n) + 1$ on the harmonic series. □

We can now use hardness results for maximum edge biclique to obtain hardness results for $1|prec|\sum w_j C_j$. The best known hardness result for MEB is presented in Chapter 4. For our purposes, it will be convenient to state it as follows (the statement is obtained by using the standard trick of graph products; see Section 4.5).

Theorem 3.7.3 *Let $\epsilon > 0$ be an arbitrarily small constant. There exist positive constants b and ϵ' (that depend on ϵ) so that for all constants $k > 0$, given a SAT instance ϕ of size N, we can probabilistically construct an n by n bipartite graph G in time $2^{O(N^\epsilon)}$ such that with high probability*

- *(Completeness) if ϕ is satisfiable then G has an edge biclique of value at least $(b+\epsilon')^k n^2$;*
- *(Soundness) if ϕ is not satisfiable then G has no edge biclique of value $b^k n^2$.*

By combining the above theorem with the bounds of Lemma 3.7.2, we have that, in the completeness case, S_G has a schedule of value at most

$$n^2\left(1-(b+\epsilon')^k\right)$$

whereas, in the soundness case, all schedules of S_G have value at least

$$n^2\left(1-b^k(\ln 1/b^k+2)\right).$$

Clearly, there is a sufficiently large k (that depends on b and ϵ' which in turn depend on ϵ) such that

$$(b+\epsilon')^k > b^k(\ln 1/b^k+2).$$

It follows that $1|prec|\sum w_j C_j$ has no PTAS unless SAT can be solved by a (probabilistic) algorithm that runs in time $2^{O(N^\epsilon)}$, where N is the instance size and $\epsilon > 0$ can be made arbitrarily close to 0.

3.8 Conclusions

We have considered the classic precedence-constrained single machine scheduling problem with the weighted sum of completion times objective. The objective function of the scheduling problem can be split into a so-called "fixed-cost" and a "variable-cost". Only the variable-cost depends on the schedule, whereas the fixed-cost is the same for all feasible schedules of a given instance. In a series of recent papers [CH99; CS05; AM09] it was established that (the variable-cost of) the scheduling problem is in fact a special case of minimum weighted vertex cover on a graph with certain structural properties [CS05].

We have continued to investigate the structure of the vertex cover graph associated with the scheduling problem. In particular, we observed that the obtained graph is exactly the graph of incomparable pairs defined in dimension theory of partial orders. Exploiting this relationship allowed us to present a framework for obtaining $(2-2/f)$-approximation algorithms, whenever the set of precedence constraints has fractional dimension f. Our approach yields the best known approximation ratios for all previously considered special classes of precedence constraints and it provides the first results for precedence constraints of bounded degree.

Unfortunately, our approach fails to improve the approximation guarantee in the general case. In order to explain this, we showed that the variable-cost of $1|prec|\sum w_jC_j$ is as hard to approximate as the vertex cover problem. This actually shows that any approach (including ours), which only takes into account the variable-cost, is unlikely to improve the approximation guarantee.

When the fixed-cost is taken into consideration, the scheduling problem might become easier to approximate since the techniques for proving hardness results for the variable-cost do not generalize to the complete objective function (i.e. the fixed-cost plus the variable-cost). For the complete objective function, we presented the first inapproximability result that rules out the existence of a PTAS for the scheduling problem. This result was obtained by establishing a connection to the maximum edge biclique problem. Consequently, an inapproximability result for maximum edge biclique under a weaker assumption would lead to a hardness result for the scheduling problem under the same assumption.

Below we mention two open problems that arise naturally from the work presented in this chapter. The second was also raised in [SW99] as "Open Problem 9".

1. *What is the complexity of* $1|prec|\sum w_jC_j$ *with convex bipartite or semi-order precedence constraints?*

   We showed that $1|prec|\sum w_jC_j$ restricted to interval order precedence constraints remains (weakly) NP-hard. The NP-hardness of the schedul-

ing problem with precedence constraints of bounded up/down degree follows from the work by Lawler [Law78]. However, the complexity of $1|prec|\sum w_j C_j$ with convex bipartite or semi-order precedence constraints remains open.

2. *Does the general version of* $1|prec|\sum w_j C_j$ *admit a* $(2-\epsilon)$*-approximation algorithm for some constant* $\epsilon > 0$*?*

   Our results show that an answer to this question is likely to be obtained by understanding the interplay between the fixed-cost and variable-cost. Indeed, for scheduling instances with very large fixed-cost we showed that it is unlikely to approximate the variable-cost within a ratio less than 2. What remains to be understood is if the variable-cost becomes easier to approximate when the fixed-cost is smaller than the variable-cost. In an exiting recent development, Bansal & Khot [BK09] addressed this issue and showed that if a new stronger version of the unique games conjecture is true then $1|prec|\sum w_j C_j$ is NP-hard to approximate within a factor $2-\epsilon$, for any $\epsilon > 0$.

Chapter 4

Maximum Edge Biclique, Optimal Linear Arrangement and Sparsest Cut

4.1 Introduction

In this chapter we consider the following notorious problems (see also Figure 4.1 for small examples):

- *Maximum edge biclique (MEB)* is the problem of finding a k_1 by k_2 complete bipartite subgraph of an n by n bipartite graph G so as to maximize $k_1 \cdot k_2$ (the number of edges).

- *(Uniform) sparsest cut (SPC)* on a graph $G(V, E)$ is the problem of finding a cut $(S, \bar{S})$, where $\bar{S} = V \setminus S$, that minimizes the sparsity $E(S, \bar{S})/(|S| \cdot |\bar{S}|)$, where $E(S, \bar{S})$ denotes the number of edges crossing the cut.

- *Optimal linear arrangement (OLA)* on a graph $G(V, E)$ is the problem of finding a permutation of the vertices — a bijective function $\pi : V \to \{1, 2, \ldots, |V|\}$ — so as to minimize $\sum_{\{u,v\} \in E} |\pi(v) - \pi(u)|$.

Maximum edge biclique, sparsest cut, and optimal linear arrangement are fundamental combinatorial problems. They have a rich number of applications in areas such as computational biology, circuit design, manufacturing optimization, and graph drawing (see e.g. [Shm97; CC00; DKST01; DPS02]). Moreover, as they often appear as sub-routines in algorithms, it is important to understand if we can efficiently find "good" solutions to these problems. For example, suppose we have a "good" algorithm for the sparsest cut problem. Then we can partition a graph into large pieces while minimizing the size of the "interface" between them, a property that is very useful when designing graph theoretic

algorithms via the divide-and-conquer paradigm (see [Shm97] for a comprehensive discussion).

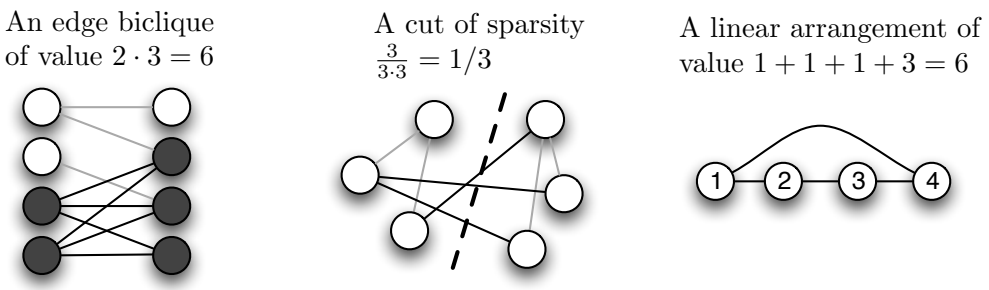


Figure 4.1: Examples of the addressed problems.

4.1.1 Literature Review

Since the addressed optimization problems are NP-hard [GJ79; MS90; Pee03], one is forced to settle for approximation algorithms. Unfortunately, there is no known approximation algorithm for the maximum edge biclique problem that achieves a significantly better approximation guarantee than the inverse of the number of edges in the bipartite graph. The situation for the sparsest cut problem and the optimal linear arrangement problem is more hopeful. Leighton & Rao [LR99] showed that the sparsest cut problem can be approximated within a factor $O(\log n)$ by using a linear programming relaxation. The approximation guarantee is tight in the sense that it matches the lower bound on the integrality gap of the corresponding relaxation up to constant factors [LR99]. Recently, Arora, Rao & Vazirani [ARV04] used semidefinite programming to obtain the best known approximation algorithm for uniform sparsest cut with performance guarantee $O(\sqrt{\log n})$ [1] . Subsequently, these techniques were also used to obtain the algorithm of choice for the non-uniform sparsest cut problem [CGR08], which is a generalization of the uniform sparsest cut problem. The situation is similar for optimal linear arrangement. Feige & Lee [FL07] and Charikar et al. [CHKR06] independently showed that combining the techniques in [ARV04] with the rounding algorithm of Rao and Richa [RR04] yields an $O(\sqrt{\log n} \log\log n)$-approximation algorithm for OLA. This improves over the

[1] The same approximation guarantee was later obtained without solving the semidefinite program and this approach has better running time [AHK04].

$O(\log n)$-approximation algorithm of Rao and Richa [RR04]. The semidefinite programming relaxations used for sparsest cut and OLA were recently shown to have integrality gap $\Omega(\log\log n)$ by Devanur, Khot, Saket & Vishnoi [DKSV06]. This result suggests that we cannot use those relaxations to obtain a constant factor approximation algorithm for the sparsest cut problem or the optimal linear arrangement problem.

Despite substantial efforts, it seems difficult to obtain good (constant factor) approximation algorithms for the considered problems. Instead, one can hope for negative results, i.e., results that indeed show the problems to be hard to approximate. For sparsest cut and optimal linear arrangement, the only known hardness results are based on the unique games conjecture [Kho02] and say that the *non-uniform* sparsest cut problem has no constant factor approximation algorithm [KV05; CKK$^+$06]. Feige & Kilian showed that the maximum edge biclique problem is hard to approximate within a factor of $2^{(\log n)^\delta}$ for some $\delta > 0$ under the plausible assumption that 3-SAT $\notin DTIME(2^{n^{3/4}})$. This was later improved by Feige [Fei02] who showed that maximum edge biclique is hard to approximate within $O(n^\epsilon)$, for some $\epsilon > 0$, by assuming a hypothesis about average-case hardness of Random 3-SAT.

In summary, no good approximation algorithms are known for maximum edge biclique, sparsest cut, and optimal linear arrangement. At the same time, the only known hardness of approximation results use non-standard assumptions and apply to non-uniform sparsest cut (a more general and thus possibly harder problem than uniform sparsest cut) and maximum edge biclique. Improving our understanding of the approximability of these problems is considered a major open problem in complexity theory (see e.g. [Vaz01; Tre04; DKSV06]).

4.1.2 Results and Overview of Techniques

Our results use the recent Quasi-random PCP construction of Khot [Kho06], who proved important inapproximability results for graph min-bisection, densest subgraph, and balanced bipartite clique. These inapproximability results were obtained under the (fairly) standard assumption that SAT has no probabilistic algorithm that runs in time 2^{n^ϵ}, where n is the instance size and $\epsilon > 0$ can be made arbitrarily close to 0. Prior to Khot's results, graph min-bisection, densest subgraph, and balanced bipartite clique had a similar status as maximum edge biclique, sparsest cut, and optimal linear arrangement, i.e., no good approximation guarantees, and the only hardness results where obtained by using non-standard assumptions [Fei02]. However, the results in [Kho06], or even the stronger average-case assumptions used by Feige in [Fei02], were not known to generalize to sparsest cut and optimal linear arrangement (see

e.g. [Tre04; DKSV06]).

The main contribution of this chapter is to show that the Quasi-random PCP [Kho06] and carefully designed constructions indeed suffice to rule out the existence of a polynomial time approximation scheme (PTAS) for sparsest cut, optimal linear arrangement, and maximum edge biclique.

Theorem 4.1.1 *Let $\epsilon > 0$ be an arbitrarily small constant. If there is a PTAS for sparsest cut, optimal linear arrangement or maximum edge biclique then SAT has a (probabilistic) algorithm that runs in time 2^{n^ϵ}, where n is the instance size.*

Proof overview. The hardness of approximation follows by presenting reductions from the Quasi-random PCP [Kho06]. The Quasi-random PCP is discussed in Section 4.1.3. Informally, it says that, given a set B of "bits" and a family T of tests that are subsets of B, it is hard to distinguish whether (i) there exists a subset $B' \subseteq B$ of half the bits so that "many" of the tests in T are subsets of B' or (ii) for any subset $B' \subseteq B$ of half the bits, only "few" tests in T are subsets of B'.

The reductions to maximum edge biclique, sparsest cut, and optimal linear arrangement are presented in Sections 4.2, 4.3, and 4.4, respectively. They all follow a general pattern that is sketched below. We start by building a graph instance of the addressed problem with vertices corresponding to the bits and tests of the Quasi-random PCP. The graph is created in such a way that vertices corresponding to tests ("test-vertices") have a relatively low impact on the total solution cost. This is achieved by having a relatively small number of test-vertices. Moreover, when test-vertices are disregarded, then any optimal solution is balanced; that is, bit-vertices are evenly partitioned into two parts in such a solution. Since test-vertices have low impact on the total cost, one can prove that any "good" solution must be quasi-balanced, i.e., bit-vertices are roughly evenly partitioned into two parts, B' and B'', in the solution. By the construction of the graph, test-vertices that correspond to tests that are subsets of B' have a lower cost (referred to as "good test-vertices"). The gap then follows by noting that, by the Quasi-random PCP, it is hard to decide whether there are "many" or "few" good test-vertices. □

The hardness factor for maximum edge biclique can be boosted as done for balanced bipartite clique in [Kho06] and we get the following theorem that is proved in Section 4.5.

Theorem 4.1.2 *Let $\epsilon > 0$ be an arbitrarily small constant. Assume that SAT does not have a probabilistic algorithm that runs in time 2^{n^ϵ} on an instance of size n. Then there is no polynomial (possibly randomized) algorithm for maximum edge biclique that achieves an approximation ratio of $1/N^{\epsilon'}$ on graphs of size N where ϵ' only depends on ϵ.*

4.1.3 Preliminaries: Quasi-random PCP

The famous PCP Theorem, by Arora & Safra [AS98] and Arora et al. [ALM+98], can be stated as follows.

Theorem 4.1.3 *Given a SAT formula ϕ of size n we can in time polynomial in n construct a set of M tests satisfying:*

1. *Each test queries a constant number d of bits from a proof and based on the outcome of the queries it either accepts or rejects ϕ.*
2. *(Yes Case/Completeness) If ϕ is satisfiable then there exists a proof so that all tests accept ϕ.*
3. *(No Case/Soundness) If ϕ is not satisfiable then no proof will cause more than $M/2$ tests to accept ϕ.*

Note that by picking one test at random, one can look at only a constant number of bits of a given proof and then with good probability know whether the given proof is correct or not. Therefore, such proofs are called probabilistically checkable proofs (PCP). The algorithm that constructs a set of such tests with the goal to distinguish between correct and incorrect proofs will be referred to as a PCP verifier.

Khot [Kho06] introduced the notion of Quasi-random PCPs. The idea is to focus on the distribution (as opposed to the outcome) of queries made by the verifier. The distribution is required to depend on whether the input to the PCP verifier is a YES or a NO instance. In the NO case, the queries are required to be distributed randomly over the proof, i.e., given any set B of half the bits, if each test queries d bits from the proof then only a fraction $(1/2)^d$ of the tests is expected to query bits only from B. In the YES case, the distribution is required to be far from random. Since the verifier does not know whether the input is a YES or NO instance, it seems quite counter-intuitive at a first sight that he can make his query pattern depend on the YES/NO case. However, consider the PCP verifier by Holmerin & Khot [HK03]: each test of their verifier queries three bits from a balanced proof, i.e. a proof with an equal number of 1 bits and 0 bits, and accepts if and only if the exclusive-or of the three queried bits is zero. Suppose the tests of this verifier query the same bits of the proof no matter if it is a YES or NO instance; then the tests that accepts in the YES case will also accept in the NO case (given the same proof). It is thus necessary that the query pattern depend on the YES/NO case, without the verifier knowing which case it is.

The following Quasi-random PCP construction by Khot [Kho06] will be the starting point for our reductions and can be stated as follows (for an overview see Figure 4.2).

Theorem 4.1.4 ([Kho06]) *For every $\epsilon > 0$, given a SAT formula ϕ of size n, we can in time $2^{O(n^\epsilon)}$ probabilistically construct a set of $M = 2^{O(n^\epsilon)}$ tests satisfying with high probability:*

1. *Each test queries $d = O\left(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\right)$ bits from a proof of length $N = 2^{O(n^\epsilon)}$.*
2. *Each bit of the proof is queried by dM/N tests (queries are uniformly distributed over the proof).*
3. *(Yes Case/Completeness) If ϕ is satisfiable then there exists a set of half the bits (corresponding to the 0-bits in a correct proof) so that βM tests query bits only from this set, where $\beta = (1 - O(1/d))\frac{1}{2^{d-1}}$.*
4. *(No Case/Soundness) For $p > 0$, let B be any subset of bits of size $(1/2+p)N$. If ϕ is not satisfiable then at most $(\alpha + pd)M$ tests query bits only from B, where $\alpha = \frac{1}{2^d} + \frac{1}{2^{20d}}$.*[2]

[2]We note that in [Kho06] the soundness says that for any set of half the bits, at most αM tests query bits only from this set. The soundness here follows easily by using that each bit is queried by dM/N tests.

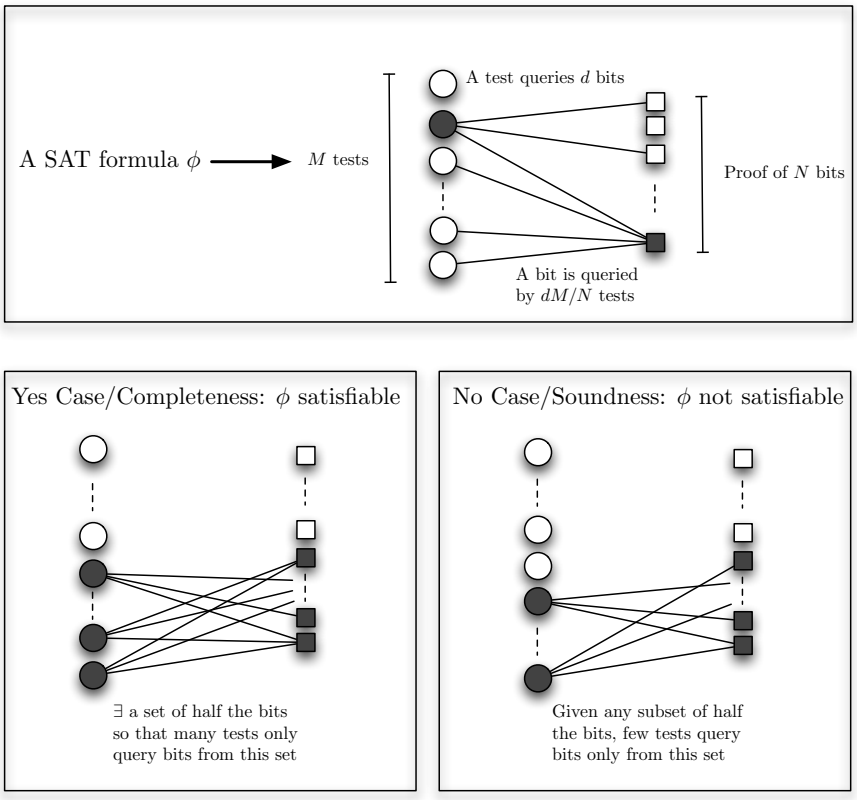


Figure 4.2: An overview of Theorem 4.1.4.

4.2 Maximum Edge Biclique

In this section we present a reduction from the Quasi-random PCP construction given by Theorem 4.1.4 to the maximum edge biclique problem so that in the completeness case the graph has an edge biclique with "large" value, whereas in the soundness case all edge bicliques have "small" value (see Section 4.2.5 for details on the achieved gap). We first present the construction (Section 4.2.1) followed by an important property of the constructed graph (Section 4.2.2). We then present the completeness and soundness analyses (Section 4.2.3 and Section 4.2.4).

Since the reduction and analysis are relatively easy, this section serves as a good starting point before continuing to the more complex reductions (that follow the same general pattern) in Sections 4.3 and 4.4.

4.2.1 Construction

Let N be the proof size and M be the total number of tests of the PCP verifier in Theorem 4.1.4. Both N and M are bounded by $2^{O(n^\epsilon)}$, where n is the size of the original SAT formula. Let d be the integer as in Theorem 4.1.4. Select w to be $\left(\frac{\beta-\alpha}{12\cdot d}\right)^2$ (very small), where $\beta := (1-O(1/d))\frac{1}{2^{d-1}}$ and $\alpha := \frac{1}{2^d} + \frac{1}{2^{20d}}$ are the bounds given by the completeness and soundness of Theorem 4.1.4.

Construct a bipartite graph $G(V, W, E)$ with $|V| = |W|$ as follows (for an overview of the construction see Figure 4.3). The right-hand side (RHS) consists of N bit-vertices corresponding to the bits in the PCP proof and M test-vertices corresponding to the tests of the PCP verifier. The left-hand side (LHS) consists of N bit-vertices corresponding to the bits in the PCP proof and M slack-vertices to keep the bipartite graph balanced. (The slack-vertices are not adjacent to any vertices and are thus not included in any bipartite clique). Connect a LHS bit-vertex to all RHS bit-vertices *except* the one corresponding to the same bit of the proof. Furthermore, connect it to a RHS test-vertex if and only if the bit is *not queried* by the test. Finally, assume that $w\frac{N}{2} = M$. (This can be achieved by simply copying vertices: every bit-vertex is replaced by c_N copies of itself, and every test-vertex is replaced by c_M copies of it such that now $wN/2 = M$ holds. Copies are connected if and only if the original vertices were. Any maximal biclique must take none or all the copies of a vertex on either side of G.)

The intuition behind the construction is the following. As there are many more bit-vertices than test-vertices, any maximum edge biclique must include approximately half of the bit-vertices of the LHS and the remaining bit-vertices of the RHS (see Section 4.2.2). We then use Theorem 4.1.4 together with the fact that bit-vertices are partitioned into two sets of approximately equal size to

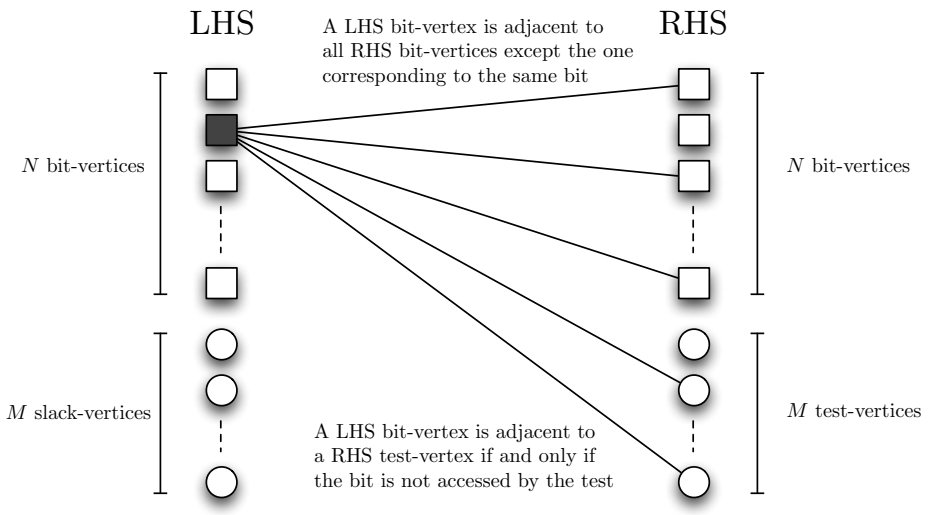


Figure 4.3: An example of the construction. Only the edges incident to the dark gray bit-vertex are depicted.

analyze the completeness and soundness (see Sections 4.2.3 and 4.2.4 respectively).

4.2.2 An Optimal Edge Biclique is Quasi-Balanced

Given a biclique let L and R denote respectively the number of bit-vertices of LHS and bit-vertices of RHS that are included in the biclique. Note that in any maximal edge biclique $L + R = N$. We say that a biclique is *quasi-balanced* if $|L - R| \leq \frac{\beta-\alpha}{6d}N$.

The following lemma follows in a straightforward manner from the fact that we have many more bit-vertices than test-vertices in our constructed biclique instance.

Lemma 4.2.1 *Any optimal edge biclique is quasi-balanced.*

Proof. Any balanced biclique of G, i.e., a biclique with $L = R = N/2$, has value at least $\left(\frac{N}{2}\right)^2$, which serves as a lower bound on the optimal solution. Now consider a biclique with $L = \frac{1+b}{2}N$ and $R = \frac{1-b}{2}N$ where $|b| > \frac{\beta-\alpha}{6d}$. Taking all

test-vertices in the biclique gives us the upper bound:

$$\begin{aligned} L(R+M) &= \frac{1+b}{2}N\left(\frac{1-b}{2}N+M\right) \\ &= \frac{1+b}{2}N\left(\frac{1-b}{2}N+w\frac{N}{2}\right) \\ &= (1-b^2+bw+w)\frac{1}{4}N^2. \end{aligned}$$

The statement follows by recalling $w = \left(\frac{\beta-\alpha}{12\cdot d}\right)^2$ and observing that

1. maximum of $f(x) = -x^2 + xw + w$ is achieved when $x = \frac{w}{2} < \frac{\beta-\alpha}{6d}$; and
2. $f(b) = -b^2 + bw + w \leq -(\frac{\beta-\alpha}{6d})^2 + \frac{\beta-\alpha}{6d}(\frac{\beta-\alpha}{12d})^2 + (\frac{\beta-\alpha}{12d})^2 < 0.$

We have thus that the value of $f(b)$ is always less than 0 when $|b| > \frac{\beta-\alpha}{6d}$. □

4.2.3 Completeness

We will see that there is an edge biclique of size at least

$$(1+\beta w)\left(\frac{N}{2}\right)^2. \tag{4.1}$$

This will be achieved by constructing a "balanced" solution, that is a biclique where the bit-vertices are partitioned into two equal sized sets. By Theorem 4.1.4, half the bits in the proof, namely the 1-bits in a correct proof, are such that a fraction β of tests do *not* query any of them. Let Γ denote the set of all such tests with $|\Gamma| = \beta M = \beta w\frac{N}{2}$. Now consider the biclique (see also Figure 4.4), where the LHS consists of the bit-vertices corresponding to the 1-bits in the proof and the RHS consists of the remaining bit-vertices (corresponding to the 0-bits in the proof) and the test-vertices corresponding to the tests in Γ. This gives an edge biclique of size $\frac{N}{2}\cdot\left(\frac{N}{2}+\beta M\right) = \frac{N}{2}\cdot\left(\frac{N}{2}+\beta w\frac{N}{2}\right) = (1+\beta w)\left(\frac{N}{2}\right)^2$.

4.2.4 Soundness

We will see that there is no edge biclique of size

$$\left(1+\frac{\alpha+\beta}{2}w\right)\left(\frac{N}{2}\right)^2. \tag{4.2}$$

By Lemma 4.2.1, it is enough to bound the value of quasi-balanced edge bicliques. Consider such a quasi-balanced biclique and let L, R, and T denote

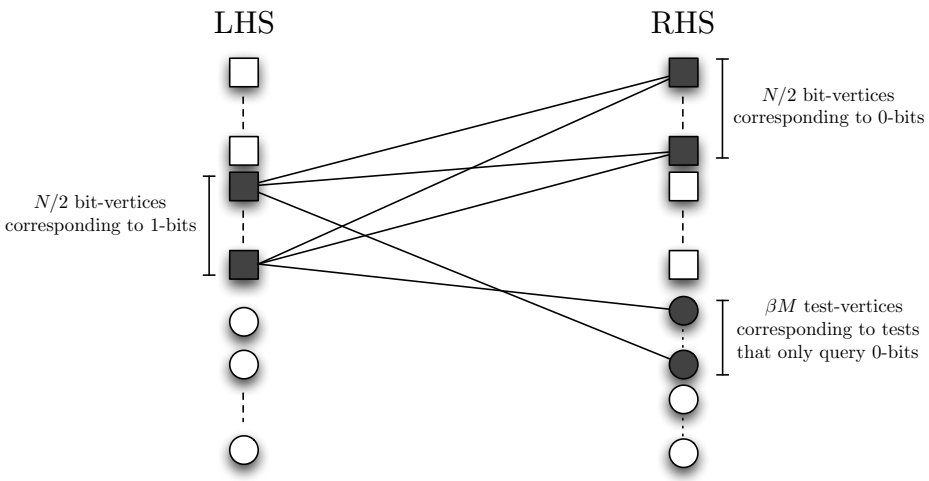


Figure 4.4: The edge biclique in the completeness case. The vertices included in the edge biclique are depicted in dark gray and only the edges incident to those vertices are depicted.

respectively the number of bit-vertices of LHS, bit-vertices of RHS, and test-vertices of RHS that are included in the biclique.

Note that a test-vertex can be included in a biclique only if it is adjacent to all bit-vertices in the LHS of the biclique. In other words, a test-vertex can be included in a biclique only if the corresponding test only queries bits that correspond to bit-vertices included in the RHS of the biclique. The soundness of Theorem 4.1.4 says that, for any given set of a fraction $1/2+p$ of the bits, at most a fraction $\alpha + p \cdot d$ of the tests only query those bits. Hence, any edge biclique with $L = \frac{1-b}{2}N$ and $R = \frac{1+b}{2}N$ has $T \leq \left(\alpha + \frac{|b|}{2}d\right) M \leq (\alpha + |b|d)w\frac{N}{2}$.

Assuming $|b| \leq \frac{\beta-\alpha}{6d}$ (Lemma 4.2.1), we have the following (rough) bound on the value of any edge biclique of G:

$$\begin{aligned} L(R+T) &\leq \frac{1-b}{2}N\left(\frac{1+b}{2}N + (\alpha+|b|d)w\frac{N}{2}\right) \\ &\leq (1+(1+|b|)(\alpha+|b|d)w)\left(\frac{N}{2}\right)^2 \\ &\leq (1+(\alpha+|b|(2d+\alpha))w)\left(\frac{N}{2}\right)^2 \\ &< \left(1+\frac{\alpha+\beta}{2}w\right)\left(\frac{N}{2}\right)^2. \end{aligned}$$

The last inequality holds because

$$\alpha + |b|(2d + \alpha) < \frac{\alpha + \beta}{2} \Leftrightarrow$$

$$2|b|(2d + \alpha) < \beta - \alpha,$$

which is easily seen to be true by recalling that $|b| \leq \frac{\beta - \alpha}{6d}$ and $\alpha < d$.

4.2.5 Inapproximability Gap

By using Theorem 4.1.4, we have provided a probabilistic reduction Γ from SAT to maximum edge biclique. For any fixed $\epsilon > 0$, given an instance ϕ of SAT of size n, Γ produces an edge biclique instance G in time $2^{O(n^\epsilon)}$ satisfying with high probability:

- *(Completeness)* If ϕ is satisfiable then G has an edge biclique of value

$$(1 + \beta w)\left(\frac{N}{2}\right)^2.$$

- *(Soundness)* If ϕ is not satisfiable then all edge bicliques of G have value at most

$$\left(1 + \frac{\alpha + \beta}{2} w\right)\left(\frac{N}{2}\right)^2.$$

The claimed hardness of approximation result now follows by recalling that (i) α, β, and w are all functions of parameter d of Theorem 4.1.4, which in turn is a function of ϵ and (ii) $\alpha < \beta$.

4.3 Sparsest Cut

We present a reduction from the Quasi-random PCP construction given by Theorem 4.1.4 to uniform sparsest cut so that in the completeness case the constructed graph has a cut with "small" sparsity, whereas in the soundness case all cuts have "large" sparsity (see Section 4.3.5 for details on the achieved gap). We first present the construction (Section 4.3.1) followed by an important property of the constructed graph (Section 4.3.2). We then present the completeness and soundness analyses (Section 4.3.3 and Section 4.3.4).

4.3.1 Construction

Let N be the proof size and M the total number of tests of the PCP verifier in Theorem 4.1.4. Both N and M are bounded by $2^{O(n^\epsilon)}$, where n is the size of the original SAT formula. Let d be the number of bits each test queries as in Theorem 4.1.4. Select $k := \left(\frac{10d}{\beta-\alpha}\right)^8$ and $h := k\left(k^2+k+\frac{1}{4}\right)$, where $\beta := (1-O(1/d))\frac{1}{2^{d-1}}$ and $\alpha := \frac{1}{2^d}+\frac{1}{2^{20d}}$ are the bounds given by the completeness and soundness of Theorem 4.1.4. Note that $h >> k >> 1$. We now describe the construction (for an overview see Figure 4.6). The graph $G = (V, E)$ consists of a bipartite graph G_b and two "huge" cliques of size kMN called C_ℓ and C_r. The graph G_b is a bipartite graph where the left-hand side (LHS) consists of M test-vertices corresponding to the tests of the PCP verifier. The right-hand side (RHS) consists of N clusters, one for each bit in the PCP proof, where each cluster consists of M bit-vertices. Place edges between a test-vertex to *all* vertices of a cluster if and only if the bit corresponding to that cluster is queried by the test.

Finally, we complete the construction of the graph G by connecting the bipartite graph G_b to C_ℓ and C_r as follows. Each bit-vertex has $h\frac{M}{N}$ edges to C_ℓ and $h\frac{M}{N}$ edges to C_r, and each test-vertex has $(d-\frac{\beta-\alpha}{5d})M$ edges to C_r. Furthermore, we distribute the edges incident to the cliques so that the difference of the degree between any two vertices in a clique is at most one.

The intuition behind the construction is the following. For a cut to have low sparsity it is good to divide the vertices into two sets of approximately the same size. As our construction has relatively few test-vertices compared to the number of bit-vertices and the size of the cliques, a cut of small sparsity must place the cliques on different sides and partition the bit-vertices into two sets of approximately the same size (see Section 4.3.2). We then use Theorem 4.1.4 together with the fact that in any good cut the bit-vertices are partitioned into two sets of approximately equal size, to analyze the completeness and soundness (see Sections 4.3.3 and 4.3.4 respectively).

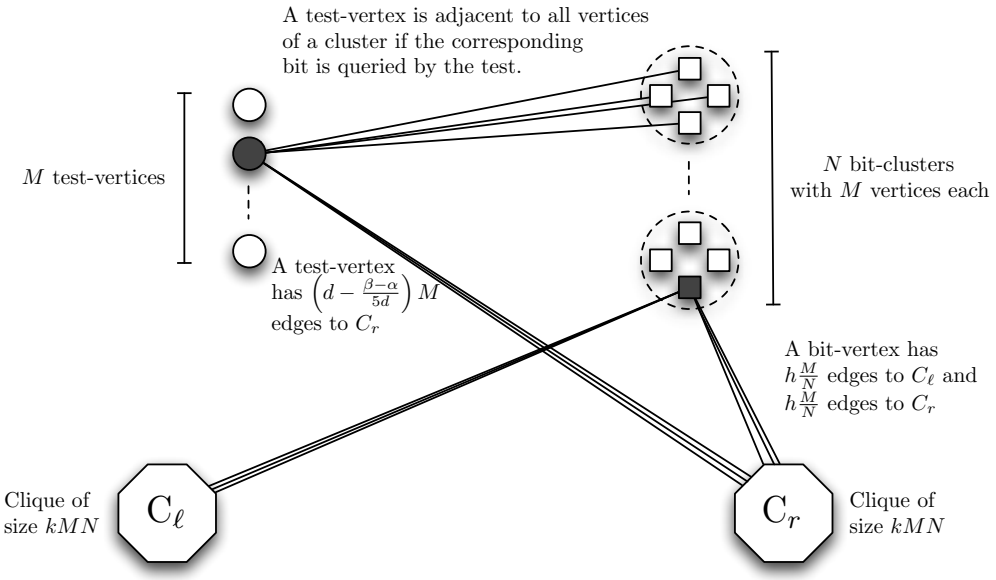


Figure 4.5: The graph G for sparsest cut. Cliques, bit-vertices, and test-vertices are depicted by polygons, squares and circles, respectively. For simplicity, only edges incident to dark gray vertices are depicted.

4.3.2 An Optimal Cut is Quasi-Balanced

We say that a cut $(S, \bar{S})$ is *quasi-balanced* if it satisfies the following properties:

1. The cliques C_ℓ and C_r are placed on different sides of the cut. Assume, without loss of generality, that the vertices of C_ℓ are included in S and the vertices of C_r are included in $\bar{S}$.

2. Let L and R be the number of bit-vertices in S and $\bar{S}$, respectively. Then $|L - R| < \left(\frac{\beta-\alpha}{10d}\right)^2 NM$.

The goal of this section is to prove that any optimal sparsest cut must be quasi-balanced. Indeed, if we consider the subgraph induced by all but the test-vertices, then it is easy to see that any sparsest cut is balanced, that is, quasi-balanced with $|L - R| = 0$. The intuition is now that the test-vertices have a relatively small impact on the cost and, hence, any optimal sparsest cut must be close to being balanced, i.e., quasi-balanced. For the formal proof, we will need some useful properties of the constructed graph G.

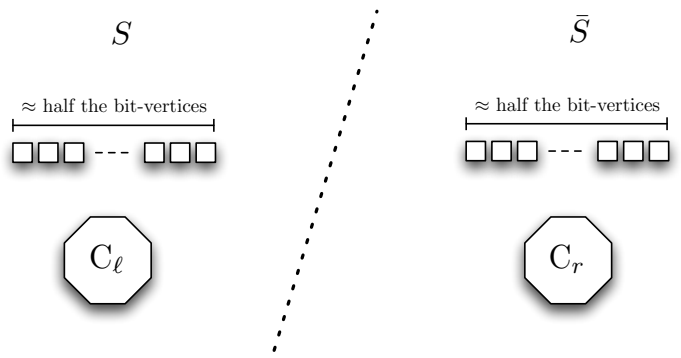


Figure 4.6: A quasi-balanced cut. (The edges and test-vertices are not depicted).

Observation 4.3.1

1. *The number of edges from bit-vertices and test-vertices to a clique is less than* $h\frac{M}{N} \cdot MN + dM^2 = (h+d)M^2$. *By the distribution of edges, a vertex* v *of* C_ℓ *or* C_r *is thus adjacent to at most* $\lceil \frac{(h+d)M^2}{kMN} \rceil = \lceil \frac{(h+d)M}{kN} \rceil < 3h\frac{M}{N}$ *vertices outside the clique.*

2. *A bit-vertex is adjacent to* $2h\frac{M}{N}$ *vertices of the cliques. As queries are uniformly distributed (see Theorem 4.1.4), a bit-vertex is adjacent to at most* $d\frac{M}{N}$ *test-vertices. It follows that a bit-vertex is adjacent to at most* $2h\frac{M}{N} + d\frac{M}{N} \leq 3h\frac{M}{N}$ *vertices.*

Lemma 4.3.2 *The graph* G *has a cut* $(S, \bar{S})$ *with sparsity*

$$\frac{E(S,\bar{S})}{|S||\bar{S}|} \leq \frac{1}{N^2}\left(k + \frac{\frac{d}{2}}{k^2+k+\frac{1}{4}}\right). \tag{4.3}$$

Moreover, $E(S, \bar{S}) = O(M^2)$ *in any optimal sparsest cut of* G.

Proof. Consider the cut $(S, \bar{S})$, where S contains all vertices of C_ℓ and the bit-vertices corresponding to half the bits ($\bar{S}$ contains the remaining vertices).

Since the cliques are on different sides of the cut and the solution is "balanced", i.e., the bit-vertices are partitioned into two sets of equal size, we have that $|S||\bar{S}| \geq \left(kMN + \frac{MN}{2}\right)\left(kMN + \frac{MN}{2}\right) = M^2N^2(k^2+k+\frac{1}{4})$.

We continue by calculating $E(S, \bar{S})$. Since all vertices of C_ℓ are in S and all vertices of C_r are in $\bar{S}$, we have that the number of edges between bit-vertices and the cliques that cross the cut is $MN \cdot h\frac{M}{N} = hM^2$. Consider the edges incident to test-vertices. Note that, as each test queries d bits and in G there is a cluster

of M bits for each bit, the total number of edges incident to test and bit-vertices is dM^2. By Theorem 4.1.4, the queries are uniformly distributed and thus the total number of edges between the test-vertices and the bit-vertices in S that corresponds to half the bits is $\frac{dM^2}{2}$. Summing up the above observations, we get $E(S,\bar{S}) = M^2\left(h + \frac{d}{2}\right)$. It follows that the sparsity of the cut is

$$\frac{E(S,\bar{S})}{|S||\bar{S}|} \leq \frac{M^2\left(h+\frac{d}{2}\right)}{M^2N^2(k^2+k+\frac{1}{4})},$$

which, by recalling that $h = k\left(k^2+k+\frac{1}{4}\right)$, can be written as

$$\frac{1}{N^2}\left(k + \frac{\frac{d}{2}}{k^2+k+\frac{1}{4}}\right),$$

which is the right-hand side of (4.3).

Finally, to see that any optimal sparsest cut $(S,\bar{S})$ has $E(S,\bar{S}) = O(M^2)$, note that the total number of vertices of G is $2kNM + NM + M$. Hence $|S||\bar{S}| \leq (|V|/2)^2 = \left(kNM + \frac{NM}{2} + \frac{M}{2}\right)^2 \leq ((k+1)NM)^2 = O\left((NM)^2\right)$ for any cut. Now suppose toward contradiction that there exists an optimal sparsest cut with $E(S,\bar{S}) = \omega(M^2)$. Then $\frac{E(S,\bar{S})}{|S||\bar{S}|} = \omega(1/N^2)$, which contradicts its optimality, since we proved that there exists a cut with sparsity $O(1/N^2)$.

□

We are now ready to prove the main result of this section.

Lemma 4.3.3 *Any optimal cut is quasi-balanced.*

Proof. We show that an optimal cut is quasi-balanced by first proving that the cliques are placed on different sides of the cut (Claim 4.3.4 and Claim 4.3.5) and then that bit-vertices are partitioned into two sets of almost equal sizes (Claim 4.3.6).

We say that a clique is divided in a cut $(S,\bar{S})$ if both sets S and $\bar{S}$ contain vertices of the clique. The intuition behind the following claim is that the cliques are so huge so that any cut dividing a clique will have a large number of edges crossing the cut.

Claim 4.3.4 *The cliques C_ℓ and C_r are not divided in any optimal sparsest cut.*

Proof of Claim. Given an optimal sparsest cut $(S,\bar{S})$, we prove that all vertices of C_r are placed in either S or $\bar{S}$. (The proof that C_ℓ is not divided is similar

and left to the reader). Let l and r be the number of vertices of C_r in S and $\bar{S}$, respectively. Suppose toward contradiction that $l > 0$ and $r > 0$.

If both l and r are big, say at least $\frac{kNM}{4}$, then we have $E(S,\bar{S}) \geq \left(\frac{kNM}{4}\right)^2$, which contradicts the optimality of the cut since an optimal cut has $E(S,\bar{S}) = O(M^2)$ (see Lemma 4.3.2).

Now consider case 1: when $0 < l < \frac{kNM}{4}$ (case 2: when $0 < r < \frac{kNM}{4}$ is symmetric). Let v be a vertex of C_r that is placed in S. We complete the proof by considering the following two sub-cases.

Case 1.a**:** Suppose there exists a bit-vertex v_b in $\bar{S}$ and consider what happens with the sparsity if we swap places of v and v_b. As the bit-vertex v_b is adjacent to at most $3h\frac{M}{N}$ vertices in total and v is adjacent to at most $3h\frac{M}{N} + \frac{kNM}{4}$ vertices in S (see Observation 4.3.1) and to at least $\frac{3kNM}{4}$ vertices in $\bar{S}$ (that belong to C_r), the number of edges that cross the cut will decrease by at least

$$\frac{3kNM}{4} - \frac{kNM}{4} - 3h\frac{M}{N} - 3h\frac{M}{N} > \frac{kNM}{4},$$

(for big enough N). The sizes of the two partitions S and $\bar{S}$ remain unchanged. It follows that the sparsity of the cut will decrease, which contradicts its optimality.

Case 1.b**:** Suppose there are no bit-vertices in $\bar{S}$. Then all bit-vertices are in S and we have $|S| \geq NM$ and since $r > 3kNM/4$ we have $|\bar{S}| \geq 3kNM/4$. Consider what happens if we move v to $\bar{S}$. Similar to the case above, the number of edges that cross the cut will decrease by at least $\frac{kNM}{4}$. The new value of the sparsest cut will thus be at most

$$\frac{E(S,\bar{S}) - \frac{kNM}{4}}{(|S|-1)(|\bar{S}|+1)} = \frac{E(S,\bar{S}) - \frac{kNM}{4}}{|S||\bar{S}|(1 - \frac{1}{|S|} + \frac{1}{|\bar{S}|} - \frac{1}{|S||\bar{S}|})}.$$

By using that both $|S|$ and $|\bar{S}|$ are at least NM, we have that the sparsity is at most

$$\frac{E(S,\bar{S}) - \frac{kNM}{4}}{|S||\bar{S}|(1 - \frac{2}{NM})},$$

which is strictly smaller than $\frac{E(S,\bar{S})}{|S||\bar{S}|}$ because (using that we have $E(S,\bar{S}) = O(M^2)$ in an optimal cut)

$$E(S,\bar{S})\left(1 - \frac{2}{NM}\right) \geq E(S,\bar{S}) - O\left(\frac{M}{N}\right) \geq E(S,\bar{S}) - \frac{kNM}{4},$$

again contradicting the optimality of the cut.

□

Given that the cliques are not divided in an optimal sparsest cut we now prove that they are placed on different sides. The intuition is that the cliques are so huge that a cut that places them on the same side is very unbalanced, i.e., the product $|S||\bar{S}|$ is small, which in turn will cause the cut to have large sparsity.

Claim 4.3.5 *The cliques C_ℓ and C_r are placed on different sides in any optimal sparsest cut.*

Proof of Claim. Suppose toward contradiction that both cliques are placed in say S in an optimal sparsest cut $(S, \bar{S})$. Recall that each bit-vertex has $2h\frac{M}{N}$ edges to the cliques and each test-vertex has $\left(d - \frac{\beta-\alpha}{5d}\right)M$ edges to the clique C_r. It follows that each vertex in $\bar{S}$ has at least $2h\frac{M}{N}$ edges that cross the cut (for big enough N), and the cut has sparsity:

$$\frac{E(S,\bar{S})}{|S||\bar{S}|} \geq \frac{2h\frac{M}{N} \cdot |\bar{S}|}{|S||\bar{S}|} \geq \frac{2h\frac{M}{N}}{4kMN} = \frac{k^2 + k + \frac{1}{4}}{2N^2}.$$

This contradicts the optimality of the cut, by recalling that G has a cut with sparsity (4.3). □

By the above claim we can assume that the cliques C_ℓ and C_r are placed on different sides of the cut. We continue by proving that the bit-vertices are partitioned into two sets of almost equal size. The following claim completes the proof of Lemma 4.3.3.

Claim 4.3.6 *Given an optimal cut $(S, \bar{S})$, let L and R be the number of bit-vertices in S and $\bar{S}$, respectively. Then*

$$|L - R| \leq \left(\frac{\beta - \alpha}{10d}\right)^2 NM.$$

Proof of Claim. Since the cliques are placed on different sides of the cut, each bit-vertex has at least $h\frac{M}{N}$ incident edges that cross the cut. It follows that $E(S,\bar{S}) \geq h\frac{M}{N} \cdot MN = hM^2$. Suppose toward contradiction that $L = \frac{1+p}{2}NM$ and $R = \frac{1-p}{2}NM$ with $|p| > \left(\frac{\beta-\alpha}{10d}\right)^2$. Then the calculations below show that the sparsity of such a cut is greater than (4.3), which contradicts its optimality.

$$\begin{aligned}
\frac{E(S,\bar{S})}{|S||\bar{S}|} &\geq \frac{hM^2}{\left(kMN+\frac{1+p}{2}MN+M\right)\left(kMN+\frac{1-p}{2}MN+M\right)} \\
&= \frac{h}{N^2\left(k^2+k+\frac{1-p^2}{4}+O(\frac{1}{N})\right)} \\
&\geq \frac{1}{N^2}\left(\frac{h+d}{k^2+k+\frac{1}{4}}\right) \text{ (for a big enough } N\text{)} \\
&= \frac{1}{N^2}\left(k+\frac{d}{k^2+k+\frac{1}{4}}\right) \text{ (recall } h=k(k^2+k+1/4)\text{)}
\end{aligned}$$

The last inequality holds because we assumed $|p| > \left(\frac{\beta-\alpha}{10d}\right)^2$ and we have

$$h\cdot\left(k^2+k+\frac{1}{4}\right) \geq (h+d)\cdot\left(k^2+k+\frac{1-p^2}{4}+O(1/N)\right) \Leftrightarrow$$

$$h\cdot\left(\frac{p^2}{4}-O(1/N)\right) \geq d\cdot\left(k^2+k+\frac{1-p^2}{4}+O(1/N)\right),$$

which can easily be seen to be true by recalling that $h = k(k^2+k+1/4)$ and $k = \left(\frac{10d}{\beta-\alpha}\right)^8$. □

The proof of the above claim concludes the proof of Lemma 4.3.3. □

4.3.3 Completeness

We will see that there is a cut with sparsity at most

$$\frac{1}{N^2}\left(k+\frac{\frac{d}{2}-\beta\frac{\beta-\alpha}{5d}}{k^2+k+\frac{1}{4}}\right). \tag{4.4}$$

By Theorem 4.1.4, half the bits in the proof, namely the 0-bits in a correct proof, are such that a fraction β of the tests access only these bits in their queries. Let Γ denote the set of all such tests with $|\Gamma| = \beta M$. We now partition the vertices of G as follows: S contain the vertices of C_ℓ, the vertices of the clusters corresponding to the 0-bits, and the test-vertices of Γ ($\bar{S}$ contains the remaining vertices).

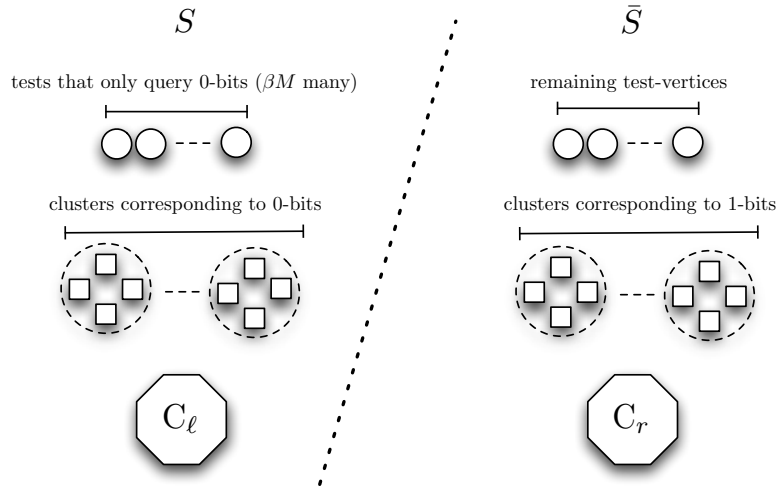


Figure 4.7: The cut $(S, \bar{S})$ in the completeness case. For simplicity, no edges are depicted.

Since the cliques are on different sides of the cut and the solution is "balanced", i.e., the bit-vertices are partitioned into two sets of equal size, we have that $|S||\bar{S}| \geq \left(kMN + \frac{MN}{2}\right)\left(kMN + \frac{MN}{2}\right) = M^2N^2(k^2 + k + \frac{1}{4})$.

We continue by calculating $E(S, \bar{S})$. Since all vertices of C_ℓ are in S and all vertices of C_r are in $\bar{S}$, we have that the number of edges between bit-vertices and the cliques that cross the cut is $MN \cdot h\frac{M}{N} = hM^2$. Consider the edges incident to test-vertices. Note that, as each test queries d bits and G has a cluster of M bits for each bit of the proof, the total number of edges incident to test and bit-vertices is dM^2. By Theorem 4.1.4, the queries are uniformly distributed and thus the total number of edges between the test-vertices and the bit-vertices corresponding to the 0-bits is $\frac{dM^2}{2}$. By observing that the test-vertices of Γ have βdM^2 edges to those bit-vertices and $\beta\left(d - \frac{\beta-\alpha}{5d}\right)M^2$ edges to C_r, the total number of edges incident to test-vertices that cross the cut is

$$\frac{dM^2}{2} - \beta dM^2 + \beta\left(d - \frac{\beta-\alpha}{5d}\right)M^2 = M^2\left(\frac{d}{2} - \beta\frac{\beta-\alpha}{5d}\right).$$

Summing up the above observations, we get $E(S, \bar{S}) = M^2\left(h + \frac{d}{2} - \beta\frac{\beta-\alpha}{5d}\right)$ and it follows that the sparsity of the cut is at most

$$\frac{M^2\left(h + \frac{d}{2} - \beta\frac{\beta-\alpha}{5d}\right)}{M^2N^2(k^2 + k + \frac{1}{4})},$$

which, by recalling that $h = k\left(k^2+k+\frac{1}{4}\right)$, can be written as

$$\frac{1}{N^2}\left(k+\frac{\frac{d}{2}-\beta\frac{\beta-\alpha}{5d}}{k^2+k+\frac{1}{4}}\right) = \text{(4.4)}.$$

4.3.4 Soundness

We will see that all cuts have sparsity at least

$$\frac{1}{N^2}\left(k+\frac{\frac{d}{2}-\frac{\alpha+\beta}{2}\frac{\beta-\alpha}{5d}}{k^2+k+\frac{1}{4}}\right). \tag{4.5}$$

We start by proving a useful property, which is later used to bound the number of "good" test-vertices. Since the construction of G does not necessarily enforce that all bit-vertices of a bit-cluster are placed on the same side of the cut, we cannot apply Theorem 4.1.4 in a straightforward way. The following lemma is a property of graph G_b (the same bipartite construction and property will be used for OLA in Section 4.4).

Lemma 4.3.7 *Consider the bipartite graph G_b, let B be a set of bit-vertices with $|B| \leq \frac{1+q}{2}NM$, where $q = \left(\frac{\beta-\alpha}{10d}\right)^2$, and let T be the set of test-vertices that each have at least $\left(d-\frac{\beta-\alpha}{10d}\right)M$ edges to the bit-vertices of B. Then for a NO-instance we have that $|T| < \frac{2\alpha+\beta}{3}M$.*

Proof. Note that all bits that are accessed by the test-vertices of T must have at least $(1-\frac{\beta-\alpha}{10d})M$ bit-vertices in B. Since $|B| \leq (1+q)NM/2$, we have that the number of bits in the proof accessed by the tests in T is at most

$$\frac{1+q}{1-\frac{\beta-\alpha}{10d}}\cdot\frac{N}{2} \leq \left(1+q+\frac{\beta-\alpha}{5d}\right)\cdot\frac{N}{2}.$$

The inequality holds because

$$1+q \leq \left(1-\frac{\beta-\alpha}{10d}\right)\left(1+q+\frac{\beta-\alpha}{5d}\right) \Leftrightarrow$$

$$\frac{\beta-\alpha}{10d}\left(1+q+\frac{\beta-\alpha}{5d}\right) \leq \frac{\beta-\alpha}{5d},$$

which is true since $q+\frac{\beta-\alpha}{5d}$ is less than one.

The soundness of Theorem 4.1.4 says that, for any given set of a fraction $(1+q+\frac{\beta-\alpha}{5d})/2$ of the bits, at most a fraction $\alpha+\left(q+\frac{\beta-\alpha}{5d}\right)\cdot d/2$ of the tests only query those bits. It follows that

$$|T| \leq \left(\alpha+\left(q+\frac{\beta-\alpha}{5d}\right)\cdot\frac{d}{2}\right)M,$$

which is less than $\frac{2\alpha+\beta}{3}M$, because

$$\alpha+\left(q+\frac{\beta-\alpha}{5d}\right)\cdot\frac{d}{2} < \frac{2\alpha+\beta}{3} \Leftrightarrow$$

$$\left(q+\frac{\beta-\alpha}{5d}\right)\cdot\frac{3d}{2} < \beta-\alpha,$$

which can be seen to be true by recalling that $q=\left(\frac{\beta-\alpha}{10d}\right)^2$. □

By Lemma 4.3.3 we only need to consider quasi-balanced cuts. (For an overview of the structure of an optimal cut in the soundness case see Figure 4.8). We continue by proving that for quasi-balanced cuts the value of $E(S,\bar{S})/(|V|/2)^2$, which is a lower bound on the sparsity of a cut $(S,\bar{S})$, is bounded from below by (4.5).

This is achieved by bounding $E(S,\bar{S})$ as follows. Consider a quasi-balanced cut $(S,\bar{S})$. Let L and R be the bit-vertices in S and $\bar{S}$, respectively. Let Γ be the set of test-vertices that each have at least $\left(d-\frac{\beta-\alpha}{10d}\right)M$ edges to the bit-vertices of L. By the fact that the cut is quasi-balanced we have that $\frac{1-q}{2}NM \leq |L| \leq \frac{1+q}{2}NM$, where $q=\left(\frac{\beta-\alpha}{10d}\right)^2$, which is sufficient for applying Lemma 4.3.7 and we get that $|\Gamma| \leq \frac{2\alpha+\beta}{3}M$. Since, by Theorem 4.1.4, the queries are uniformly distributed, the total number of edges between the test-vertices and the bit-vertices of L is at least $\frac{(1-q)dM^2}{2}$. If all test-vertices are placed in $\bar{S}$, all of these edges would cross the cut. The only way to decrease their number is to move test-vertices to S. But since every test-vertex has $\left(d-\frac{\beta-\alpha}{5d}\right)M$ edges to C_r, this is only profitable for test-vertices which have less than $\frac{\beta-\alpha}{10d}M$ edges to the bit-vertices of R, i.e., test-vertices that are in Γ. By the above argument we can assume, when calculating a lower bound of $E(S,\bar{S})$, that the only test-vertices placed in S are those in Γ and it is easy to see that assuming they are not adjacent to any bit-vertices of R might only decrease $E(S,\bar{S})$.

As in the completeness case, we have that the number of edges between bit-vertices and the cliques that cross the cut is $MN\cdot h\frac{M}{N}=hM^2$.

To summarize we have the following:

- The number of edges, incident to test-vertices that cross the cut, is at least $\frac{(1-q)dM^2}{2} - |\Gamma|dM + |\Gamma|\left(d - \frac{\beta-\alpha}{5d}\right)M = \frac{(1-q)dM^2}{2} - |\Gamma|\frac{\beta-\alpha}{5d}M$, which, by using that $|\Gamma| \leq \frac{2\alpha+\beta}{3}M$, can be bounded from below by $M^2\left(\frac{(1-q)d}{2} - \frac{2\alpha+\beta}{3}\frac{\beta-\alpha}{5d}\right)$.
- The number of edges, between bit-vertices and the cliques that cross the cut, is hM^2.

Since $|S||\bar{S}| \leq (|V|/2)^2$ we have that the sparsity of any cut of G is

$$\begin{aligned}\frac{E(S,\bar{S})}{|S||\bar{S}|} &\geq \frac{M^2\left(h + \frac{(1-q)d}{2} - \frac{2\alpha+\beta}{3}\frac{\beta-\alpha}{5d}\right)}{\left(kMN + \frac{MN}{2} + M\right)^2} \\ &= \frac{1}{N^2}\left(\frac{h + \frac{(1-q)d}{2} - \frac{2\alpha+\beta}{3}\frac{\beta-\alpha}{5d}}{k^2 + k + \frac{1}{4} + O(\frac{1}{N})}\right) \\ &\geq \frac{1}{N^2}\left(k + \frac{\frac{d}{2} - \frac{\alpha+\beta}{2}\frac{\beta-\alpha}{5d}}{k^2 + k + \frac{1}{4}}\right) = (4.5).\end{aligned}$$

The last inequality holds because $h = k(k^2 + k + 1/4)$ and

$$\begin{gathered}\frac{(1-q)d}{2} - \frac{2\alpha+\beta}{3}\frac{\beta-\alpha}{5d} > \frac{d}{2} - \frac{\alpha+\beta}{2}\frac{\beta-\alpha}{5d} \Leftrightarrow \\ \left(\frac{\alpha+\beta}{2} - \frac{2\alpha+\beta}{3}\right)\frac{\beta-\alpha}{5d} > \frac{qd}{2} \Leftrightarrow \\ \frac{\beta-\alpha}{6}\frac{\beta-\alpha}{5d} > \frac{qd}{2},\end{gathered}$$

which is true since $q = \left(\frac{\beta-\alpha}{10d}\right)^2$.

4.3.5 Inapproximability Gap

By using Theorem 4.1.4, we have provided a probabilistic reduction Γ from SAT to (uniform) sparsest cut. For any fixed $\epsilon > 0$, given an instance ϕ of SAT of size n, Γ produces a sparsest cut instance G in time $2^{O(n^\epsilon)}$ satisfying with high probability:

- *(Completeness)* If ϕ is satisfiable then G has a cut of sparsity at most

$$\frac{1}{N^2}\left(k + \frac{\frac{d}{2} - \beta\frac{\beta-\alpha}{5d}}{k^2 + k + \frac{1}{4}}\right).$$

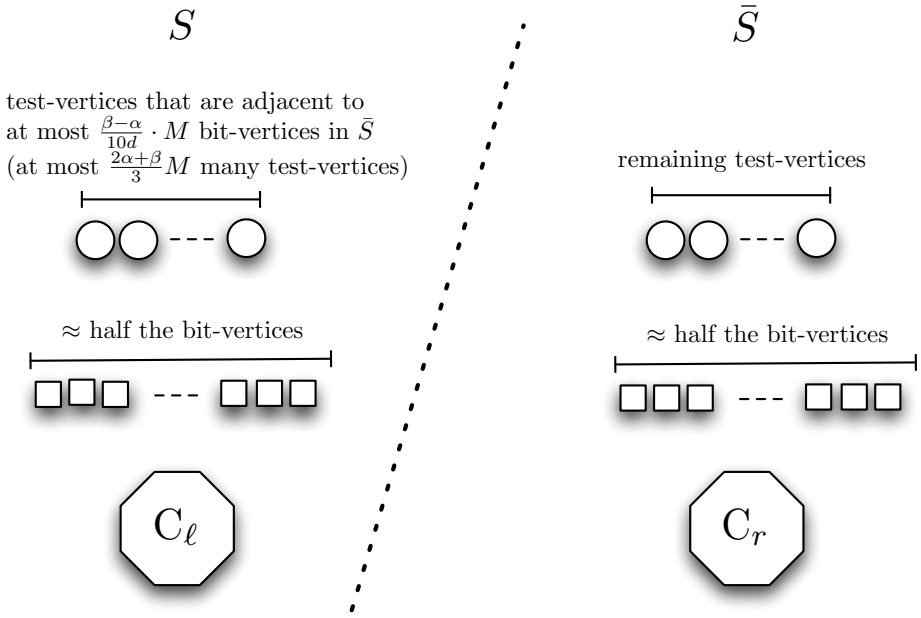


Figure 4.8: Structure of an optimal $(S, \bar{S})$ cut in the soundness case. (The edges are not depicted).

- *(Soundness)* If ϕ is not satisfiable then all cuts have sparsity at least

$$\frac{1}{N^2}\left(k + \frac{\frac{d}{2} - \frac{\alpha+\beta}{2}\frac{\beta-\alpha}{5d}}{k^2+k+\frac{1}{4}}\right).$$

The claimed hardness of approximation result now follows by recalling that (i) α, β, and k are all functions of parameter d of Theorem 4.1.4, which in turn is a function of ϵ and (ii) $\alpha < \beta$.

4.4 Optimal Linear Arrangement

For simplicity, we first consider the weighted version of OLA. That is, an edge $\{u, v\} \in E$ has weight w_{uv} and the objective is to find a permutation π of the vertices that minimizes $\sum_{\{u,v\}\in E} w_{uv}|\pi(u) - \pi(v)|$. We present a reduction from the Quasi-random PCP construction given by Theorem 4.1.4 to weighted OLA so that in the completeness case the constructed graph has a linear arrangement with "small" cost, whereas in the soundness case all linear arrangements have "large" cost (see Section 4.4.5 for details on the achieved gap). We first present the construction (Section 4.4.1) followed by an important property of the constructed graph (Section 4.4.2). We then present the completeness and soundness analyses (Section 4.4.3 and Section 4.4.4). Finally, we note in Section 4.4.6 that the arguments generalize in a straightforward manner to the unweighted case.

4.4.1 Construction

Let N be the proof size and M be the total number of tests of the PCP verifier in Theorem 4.1.4. Both N and M are bounded by $2^{O(n^\epsilon)}$, where n is the size of the original SAT formula. Let d be the number of bits each test queries in the Quasi-random PCP construction. Select k to be $\left(\frac{10d}{\beta-\alpha}\right)^8$, where $\alpha := \frac{1}{2^d} + \frac{1}{2^{20d}}$ and $\beta := (1 - O(1/d))\frac{1}{2^{d-1}}$ are the bounds given by the completeness and soundness of Theorem 4.1.4. Note that $k >> 1$. We now describe the construction (for an overview see Figure 4.9). The final graph G consists of the graphs G_b, G_ℓ, and G_r and is constructed as follows.

- The graph G_b is a bipartite graph where the left-hand side (LHS) consists of M test-vertices corresponding to the tests of the PCP verifier. The right-hand side (RHS) consists of N clusters, one for each bit in the PCP proof, where each cluster consists of M bit-vertices. Place edges, weighted by 1, between a test-vertex to *all* vertices of a cluster if and only if the bit, corresponding to that cluster, is queried by the test. (Note that G_b is the same bipartite graph as in Section 4.3.)
- The graph G_ℓ consists of a vertex C_ℓ and $2kMN$ additional slack-vertices We place an edge from each slack vertex to C_ℓ and weight these edges by $k^4\frac{M}{N}$.
- The graph G_r is constructed as G_ℓ, where instead of C_ℓ we have C_r.

Finally, we construct the graph G by connecting the bipartite graph G_b to G_ℓ and G_r as follows. Each test-vertex has edges to C_ℓ and C_r, weighted by $\frac{\beta-\alpha}{10d}M$ and $(d - \frac{\beta-\alpha}{10d})M$, respectively. Each bit-vertex has an edge to C_r of weight $k^2\frac{M}{N}$.

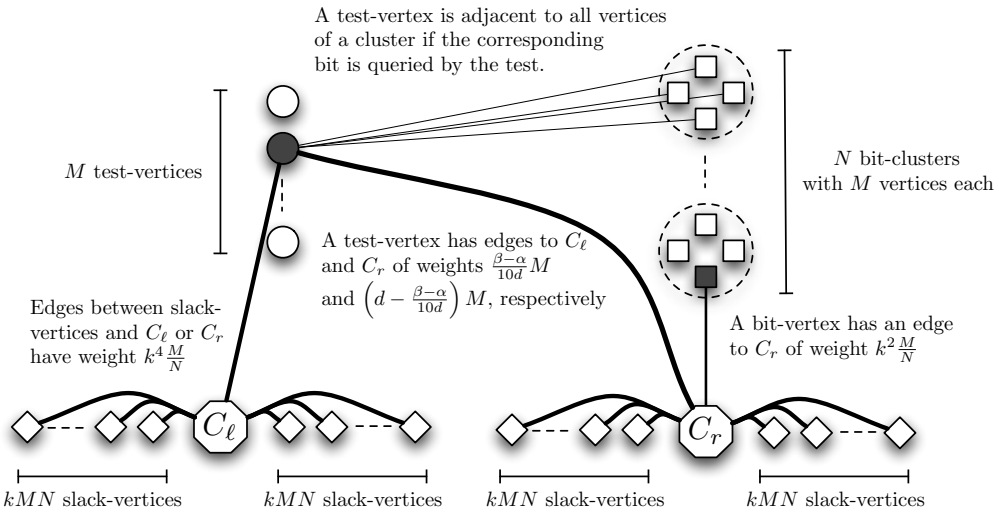


Figure 4.9: The graph G for OLA. Slack-vertices, bit-vertices, and test-vertices are depicted by diamonds, squares, and circles, respectively. For simplicity only some edges are depicted and the thickness of an edge is relative to its weight.

The intuition behind the construction is the following. As slack-vertices and bit-vertices have edges to C_ℓ and C_r of very large weight, any good optimal linear arrangement will locate these vertices evenly before and after the vertices of C_ℓ and C_r (see Figure 4.10). With this intuition in mind, we prove the important property that any good linear arrangement will partition the bit-vertices into two sets of approximately the same size (see Section 4.4.2). We then use Theorem 4.1.4 to analyze the completeness and soundness (see Sections 4.4.3 and 4.4.4 respectively).

Throughout the analyses, we restrict ourselves without loss of generality to linear arrangements where C_ℓ is placed to the left of C_r. The case when C_l is to the right of C_r is symmetric. Moreover, we use the following convention to simplify notation. Let π be a linear arrangement of G. For sets A, B of vertices we write $A <_\pi B$ (subscript omitted when π is clear from the context) whenever $\forall u \in A, \forall v \in B : \pi(u) < \pi(v)$.

4.4.2 An Optimal Linear Arrangement is Quasi-Balanced

Select $q := \left(\frac{\beta-\alpha}{10d}\right)^2$, i.e., a "small" number. We say that a linear arrangement π of G is *quasi-balanced* if (see also Figure 4.10)

- the slack-vertices of G_i can be partitioned into two sets S_L^i, S_R^i with $||S_L^i| -$

$|S_R^i|| \leq qkNM$, for $i \in \{l, r\}$; and

- the bit-vertices can be partitioned into two sets B_L and B_R with $||B_L| - |B_R|| \leq qNM$ so that

$$S_L^\ell < \{C_\ell\} < S_R^\ell < B_L < S_L^r < \{C_r\} < S_R^r < B_R.$$

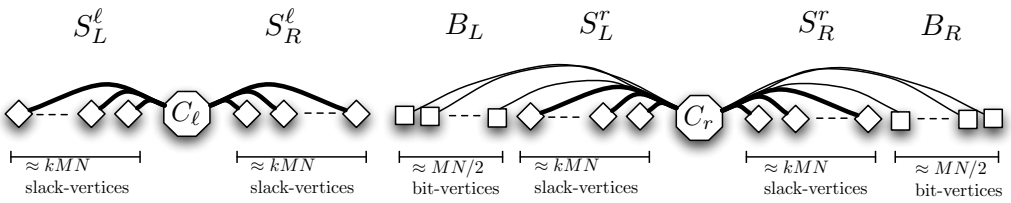


Figure 4.10: A quasi-balanced linear arrangement. (The test-vertices are not depicted).

The goal of this section is to prove that any optimal linear arrangement is quasi-balanced. Indeed, if we consider the subgraph induced on all but the test-vertices then it is easy to see that any optimal linear arrangement is balanced, that is, quasi-balanced with $|S_L^i| - |S_R^i| = 0$, for $i \in \{\ell, r\}$, and $|B_L| - |B_R| = 0$. The intuition is that the test-vertices have a relatively small impact on the cost and, hence, any optimal linear arrangement must be close to being balanced, i.e., quasi-balanced. For the formal proof, we will need the following upper bound on the cost of an optimal linear arrangement.

Lemma 4.4.1 *The graph G has a linear arrangement with cost at most*

$$M^3N\left(2k^6 + k^3 + \frac{k^2}{4} + 2dk\right). \tag{4.6}$$

Proof. Partition the slack-vertices of G_i into two sets S_L^i and S_R^i with $|S_L^i| = |S_R^i| = kNM$, for $i \in \{\ell, r\}$. Let B_L be the set of bit-vertices corresponding to a set of half the bits and let B_R be the remaining bit-vertices. Note that $|B_L| = |B_R| = NM/2$. We also let Γ with $|\Gamma| = M$ be all the test-vertices. Now consider a linear arrangement π of G so that (see Figure 4.10)

$$S_L^\ell < \{C_\ell\} < S_R^\ell < B_L < S_L^r < \Gamma < \{C_r\} < S_R^r < B_R.$$

We proceed by bounding the cost of π by considering the different edges.

- The edges incident to slack-vertices have cost at most

$$4 \cdot k^4 \frac{M}{N} \sum_{i=1}^{kNM} (i + M) = 4 \cdot k^4 \frac{M}{N} \left(\frac{kNM(kNM+1)}{2} + kNM^2) \right),$$

  which is bounded from above by $2k^6M^3N + o(M^3N)$.

- The edges between the bit-vertices and C_r have cost at most

$$2 \cdot k^2 \frac{M}{N} \sum_{i=1}^{NM/2} (i + kNM + M).$$

  Since

  (i) $\sum_{i=1}^{NM/2} i = (NM/2)(NM/2+1)/2 = \frac{(NM)^2}{8} + o\left((NM)^2\right)$,

  (ii) $\sum_{i=1}^{NM/2} kNM = k(NM)^2/2$, and

  (iii) $\sum_{i=1}^{NM/2} M = o\left((MN)^2\right)$

  the cost of the edges between bit-vertices and C_r is bounded from above by $M^3N(\frac{k^2}{4} + k^3) + o(M^3N)$.

- Now consider the edges incident to test-vertices. As an edge from a test-vertex to C_r has weight $\left(d - \frac{\beta-\alpha}{10d}\right) M$ and the length of such an edge is at most M in π, the cost of such an edge is at most $\left(d - \frac{\beta-\alpha}{10d}\right) M^2$. As there are M test-vertices, the cost of all edges from test-vertices to C_r is at most $\left(d - \frac{\beta-\alpha}{10d}\right) M^3 = o(M^3N)$. Each test-vertex also has an edge of weight $\frac{\beta-\alpha}{10d} M$ to C_ℓ and such an edge has length at most $(2kMN + MN/2 + M)$ in π. Hence, the cost of all edges from the M test-vertices to C_ℓ is at most $M^3N\frac{\beta-\alpha}{10d}\left(2k + \frac{1}{2}\right) + o(M^3N) \leq M^3Nk + o(M^3N)$. Finally, a test-vertex has at most dM edges to the bit-vertices, each of length at most $(kMN + MN/2 + M)$ in π. Thus the cost of all edges from the M test-vertices to bit-vertices is at most $M^3N\left(dk + \frac{d}{2}\right) + o(M^3N) \leq M^3N(d+1)k + o(M^3N)$.

  In summary, the total cost of the edges incident to test-vertices is at most $M^3Nk + M^3N(d+1)k + o(M^3N) = M^3N(d+2)k + o(M^3N)$.

Summing up the above observations gives us that the cost of π is at most

$$M^3N \left(2k^6 + k^3 + \frac{k^2}{4} + (d+2)k \right) + o(M^3N),$$

which is (for large enough N and M) less than

$$M^3N\left(2k^6+k^3+\frac{k^2}{4}+2dk\right)=(4.6).$$

□

We are now ready to prove the main result of this section.

Lemma 4.4.2 *Any optimal linear arrangement of G is quasi-balanced.*

Proof. We first prove (Claim 4.4.3) that in any optimal linear arrangement of G, G_i's slack-vertices can be partitioned into two sets S_L^i, S_R^i , for $i \in \{l, r\}$; and bit-vertices can be partitioned into two sets B_L and B_R so that

$$S_L^\ell < \{C_\ell\} < S_R^\ell < B_L < S_L^r < \{C_r\} < S_R^r < B_R. \tag{4.7}$$

Secondly (Claim 4.4.4), we will see that the sets must be almost "balanced" in an optimal linear arrangement that is, $||S_L^i| - |S_R^i|| \leq qkNM$ for $i \in \{l, r\}$ and $||B_L| - |B_R|| \leq qNM$.

Claim 4.4.3 *In any optimal linear arrangement π of G, vertices must be ordered as in* (4.7).

Proof of Claim. Since we only consider linear arrangements with C_ℓ to the left of C_r it is easy to see that

$$S_L^\ell <_\pi \{C_\ell\} <_\pi S_R^\ell <_\pi S_L^r <_\pi \{C_r\} <_\pi S_R^r.$$

Let v_b be a bit-vertex and v_s a slack-vertex of G_r. Suppose, toward contradiction, that v_b are placed between v_s and C_r, for example $\pi(v_s) < \pi(v_b) < \pi(C_r)$ (the remaining cases are symmetric and omitted). Consider what happens with the cost if we swap places of v_b and v_s:

- Vertex v_s is only adjacent to C_r, and this edge has weight $k^4\frac{M}{N}$.
- Vertex v_b has one edge to C_r of weight $k^2\frac{M}{N}$. Since queries are uniformly distributed (see Theorem 4.1.4) v_b has $d\frac{M}{N}$ edges to test-vertices, each of weight 1. Thus, the total weight of the edges incident to v_b is $d\frac{M}{N}+k^2\frac{M}{N}$.

It follows that by swapping v_b and v_s we *decrease* the cost by at least $(\pi(v_b) - \pi(v_s))\left(k^4\frac{M}{N} - \left(d\frac{M}{N} + k^2\frac{M}{N}\right)\right) > 0$, contradicting the optimality of π.

By the above arguments there are no bit-vertices placed in between slack-vertices and the corresponding vertex C_r (or C_ℓ). We can thus partition the bit-vertices into three sets B_1, B_L, B_R so that

$$B_1 < S_L^\ell < \{C_\ell\} < S_R^\ell < B_L < S_L^r < \{C_r\} < S_R^r < B_R.$$

We complete the proof of this claim by proving that $B_1 = \emptyset$ in an optimal linear arrangement π of G. Suppose, toward contradiction, that $B_1 \neq \emptyset$ in π. Recall that a bit-vertex has an edge of weight $k^2\frac{M}{N}$ to C_r and the total weight of its remaining edges (to test-vertices) is $d\frac{M}{N}$. Furthermore, the total weight of the edges incident to test-vertices are $2dM^2$ (the cost of the edges, that are incident to test-vertices and not to the bit-vertices in B_1, might also increase, since their length can increase when the bit-vertices in B_1 are moved) . Let π' be the linear arrangement

$$S_L^\ell < \{C_\ell\} < S_R^\ell < B_1 < B_L < S_L^r < \{C_r\} < S_R^r < B_R.$$

As $|S_L^\ell| + |S_R^\ell| = 2kNM$, the cost of π' is smaller than the cost of π by at least

$$|B_1|\left(2kMN\left(k^2\frac{M}{N} - d\frac{M}{N}\right) - 2dM^2\right),$$

which is positive whenever $B_1 \neq \emptyset$. □

The following claim completes the proof of Lemma 4.4.2.

Claim 4.4.4 *In any optimal linear arrangement π of G we have*

- $||S_L^i| - |S_R^i|| \leq qkNM$ *for* $i \in \{l, r\}$*; and*
- $||B_L| - |B_R|| \leq qNM$.

Proof of Claim. Let $|S_L^\ell| = (1+s_\ell)kMN$, $|S_R^\ell| = (1-s_\ell)kMN$, $|S_L^r| = (1+s_r)kMN$, $|S_R^r| = (1-s_r)kMN$, $|B_L| = (1+b)MN/2$, and $|B_R| = (1-b)MN/2$, where s_ℓ, s_r, and b may assume negative values.

We proceed by calculating a lower bound on the cost of π by considering the different types of edges.

- The cost of the edges incident to slack-vertices is at least

$$k^4\frac{M}{N}\left(\sum_{i=1}^{(1+s_\ell)kMN} i + \sum_{i=1}^{(1-s_\ell)kMN} i + \sum_{i=1}^{(1+s_r)kMN} i + \sum_{i=1}^{(1-s_r)kMN} i\right) \geq$$

$$k^6M^3N\left((1+s_\ell)^2/2 + (1-s_\ell)^2/2 + (1+s_r)^2/2 + (1-s_r)^2/2\right),$$

which is equal to $k^6M^3N(2+s_\ell^2+s_r^2)$.

- The cost of the edges incident to bit-vertices is at least

$$k^2\frac{M}{N}\left(\sum_{i=1}^{(1+b)MN/2}(i+(1+s_r)kNM)+\sum_{i=1}^{(1-b)MN/2}(i+(1-s_r)kNM)\right).$$

  Since

  (i) $\sum_{i=1}^{(1+b)NM/2} i+\sum_{i=1}^{(1-b)NM/2} i \geq \frac{(NM)^2}{4}\left(\frac{(1+b)^2}{2}+\frac{(1-b)^2}{2}\right)=(NM)^2\frac{1+b^2}{4}$,

  (ii) $\sum_{i=1}^{(1+b)NM/2}(1+s_r)kNM+\sum_{i=1}^{(1-b)NM/2}(1-s_r)kNM \geq (NM)^2(1-|s_r|)k$

  the cost of the edges incident to bit-vertices is bounded from below by $M^3N((1-|s_r|)k^3+\frac{1+b^2}{4}k^2)$.

Summing up the above observations we have that the cost of π is at least

$$M^3N\left((2+s_\ell^2+s_r^2)k^6+(1-|s_r|)k^3+\frac{1+b^2}{4}k^2\right).$$

As $k=\left(\frac{10d}{\beta-\alpha}\right)^8$ (a huge number) and $q=\left(\frac{\beta-\alpha}{10d}\right)^2=\left(\frac{1}{k}\right)^{1/4}$, the cost of π is greater than the upper bound on an optimal linear arrangement (4.6) whenever $|s_\ell|>q/2, |s_r|>q/2$, or $|b|>q$ and the statement follows.

□

The proof of Claim 4.4.4 concludes the proof of Lemma 4.4.2. □

4.4.3 Completeness

We will see that there is a linear arrangement with value at most

$$M^3N\left[2k^6+k^3+\frac{k^2}{4}+\left(d+\left(1-\frac{2\beta+\alpha}{3}\right)\frac{\beta-\alpha}{5d}\right)k\right]. \tag{4.8}$$

This will be achieved by constructing a "balanced" linear arrangement. Partition the slack-vertices of G_i into two sets S_L^i and S_R^i with $|S_L^i|=|S_R^i|=kNM$, for $i\in\{\ell,r\}$. Let B_L be the bit-vertices corresponding to the 0-bits in a correct proof and let B_R be the remaining bit-vertices. Note that $|B_L|=|B_R|=NM/2$. By the completeness of Theorem 4.1.4, half the bits in the proof, namely the 0-bits in a correct proof, are such that a fraction β of tests only access them in their queries. Let Γ denote the set of all such test-vertices with $|\Gamma|=\beta M$ and let $\bar{\Gamma}$ be the set of the remaining test-vertices.

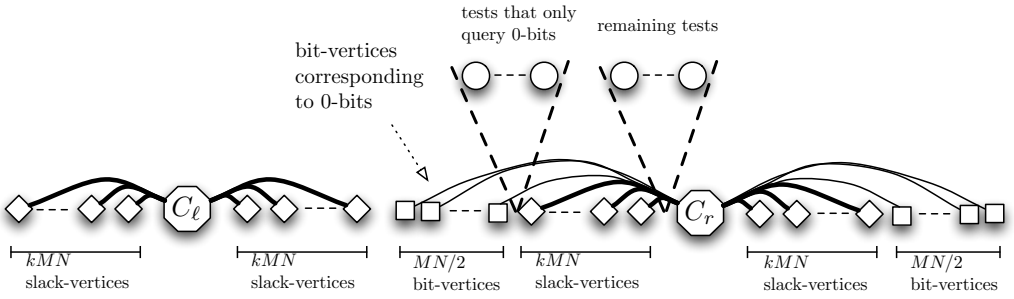


Figure 4.11: The linear arrangement π in the completeness case.

Now consider the balanced linear arrangement π of G:

$$S_L^\ell < \{C_\ell\} < S_R^\ell < B_L < \Gamma < S_L^r < \bar{\Gamma} < \{C_r\} < S_R^r < B_R$$

The following lemma concludes the completeness analysis.

Lemma 4.4.5 *The cost of π is at most* (4.8) *(for big enough M and N).*

Proof. We need to bound the cost of each edge in the linear arrangement π.

1. As in the proof of Lemma 4.4.1, both the cost of edges incident to slack-vertices and the cost of edges between the bit-vertices and C_r can be seen to be at most $M^3N(2k^6 + k^3 + \frac{k^2}{4}) + o(M^3N)$.

2. Consider a test-vertex $t \in \Gamma$. As the weight of the edge $\{t, C_r\}$ is $\left(d - \frac{\beta-\alpha}{10d}\right) M$ and its length in π is at most $kMN + M$, the cost of edge $\{t, C_r\}$ is at most $\left(d - \frac{\beta-\alpha}{10d}\right) kM^2N + o(M^2N)$. Similarly, as edge $\{t, C_\ell\}$ has weight $\frac{\beta-\alpha}{10d}M$ and its length in π is at most $(kMN + MN/2 + M)$, the cost of $\{t, C_\ell\}$ is at most $M^2N\frac{\beta-\alpha}{10d}(k + 1/2) + o(M^2N)$. Finally t has dM edges of weight 1 to bit-vertices in B_L. Since these edges have length at most $(MN/2 + M)$ in π, their cost is at most $M^2Nd/2 + o(M^2N)$.

   By the above arguments, the cost of the edges incident to the test-vertices in Γ is at most

$$|\Gamma|M^2N\left(\left(d - \frac{\beta-\alpha}{10d}\right)k + \frac{\beta-\alpha}{10d}(k + 1/2) + d/2\right) + |\Gamma|o(M^2N),$$

   which is less than $|\Gamma|M^2N(dk + d) + |\Gamma|o(M^2N)$. Using $|\Gamma| = \beta M$, we have that the edges incident to the test-vertices in Γ have cost at most $\beta M^3N(dk + d) + o(M^3N)$.

3. Similarly to the above calculations for test-vertices in Γ, the cost of edges incident to test-vertices in $\bar{\Gamma}$ can be seen to be at most

$$(1-\beta)M\left[\underbrace{\frac{\beta-\alpha}{10d}M(2kMN+MN/2)}_{\text{edges to } c_\ell}+\underbrace{dM(kMN+MN/2)}_{\text{edges to bit-vertices}}\right]+o(M^3N)=$$

$$(1-\beta)M\left[\left(d+\frac{\beta-\alpha}{5d}\right)kM^2N+\left(\frac{\beta-\alpha}{20d}+d/2\right)M^2N\right]+o(M^3N),$$

which is less than $(1-\beta)M^3N\left(\left(d+\frac{\beta-\alpha}{5d}\right)k+d\right)+o(M^3N)$.

We have considered all types of edges of G and by summing up the above costs we get that the total cost of π is at most

$$M^3N\left[2k^6+k^3+\frac{k^2}{4}+\left(d+(1-\beta)\frac{\beta-\alpha}{5d}\right)k+d\right]+o(M^3N)<$$

$$M^3N\left[2k^6+k^3+\frac{k^2}{4}+\left(d+\left(1-\frac{2\beta+\alpha}{3}\right)\frac{\beta-\alpha}{5d}\right)k\right]=(4.8).$$

The last inequality holds because

$$\left(d+(1-\beta)\frac{\beta-\alpha}{5d}\right)k+d<\left(d+\left(1-\frac{2\beta+\alpha}{3}\right)\frac{\beta-\alpha}{5d}\right)k\Leftrightarrow$$

$$d<\left(\beta-\frac{2\beta+\alpha}{3}\right)\frac{\beta-\alpha}{5d}k\Leftrightarrow$$

$$d<\frac{\beta-\alpha}{3}\cdot\frac{\beta-\alpha}{5d}k,$$

which is easily seen to be true by recalling that $k=\left(\frac{10d}{\beta-\alpha}\right)^8$. □

4.4.4 Soundness

We will see that all linear arrangements of G have value at least

$$M^3N\left[2k^6+k^3+\frac{k^2}{4}+\left(d+\left(1-\frac{\alpha+\beta}{2}\right)\frac{\beta-\alpha}{5d}\right)k\right]. \tag{4.9}$$

By Lemma 4.4.2 we only need to consider quasi-balanced linear arrangements. We proceed by bounding the cost of such linear arrangements from

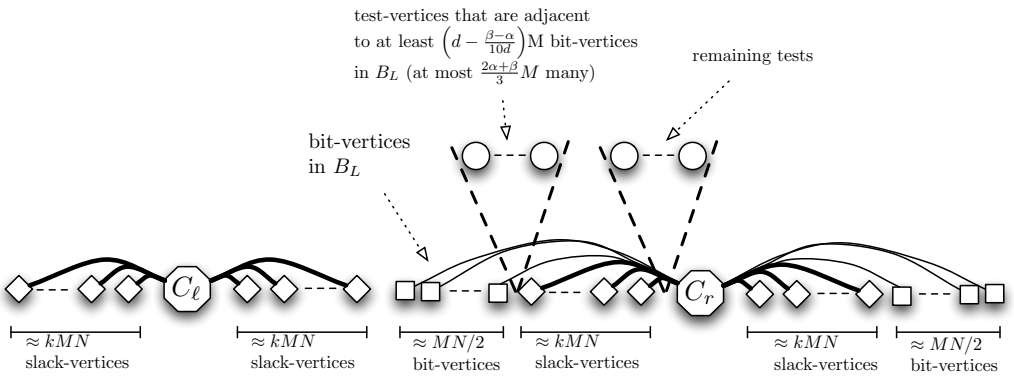


Figure 4.12: The structure of an optimal linear arrangement π in the soundness case.

below by (4.9). Given a quasi-balanced linear arrangement π of G, let Γ be the set of test-vertices that have at least $\left(d-\frac{\beta-\alpha}{10d}\right)M$ edges to B_L in π. Since $|B_L| \leq \frac{1+q}{2}NM$, we can apply Lemma 4.3.7 and get $|\Gamma| < \frac{2\alpha+\beta}{3}M$.

The following lemma follows from an easy case analysis and its proof is given in the next subsection.

Lemma 4.4.6 *In any quasi-balanced linear arrangement π of G, the cost of the edges incident to a test-vertex t is at least*

$$\begin{cases} (1-q)M^2Ndk & \text{if } t \in \Gamma, \\ (1-q)M^2N\left(d+\frac{\beta-\alpha}{5d}\right)k & \text{if } t \notin \Gamma \end{cases}.$$

The above lemma, together with $|\Gamma| < \frac{2\alpha+\beta}{3}M$, implies that the total cost of the edges incident to the M test-vertices is at least

$$(1-q)M^3N\left(d+\left(1-\frac{2\alpha+\beta}{3}\right)\frac{\beta-\alpha}{5d}\right)k. \tag{4.10}$$

As noted in Section 4.4.2, the cost of the edges not incident to test-vertices is minimized by a balanced linear arrangement (see Figure 4.10) and is thus bounded from below by

$$4k^4\frac{M}{N}\sum_{i=1}^{kMN} i + 2k^2\frac{M}{N}\sum_{i=1}^{MN/2}(i+kMN), \tag{4.11}$$

which is greater than $M^3N(2k^6+k^3+\frac{k^2}{4})$.

Summing up (4.10) and (4.11), we have that the total cost of a quasi-balanced linear arrangement is at least

$$M^3N\left[2k^6+k^3+\frac{k^2}{4}+(1-q)\left(d+\left(1-\frac{2\alpha+\beta}{3}\right)\frac{\beta-\alpha}{5d}\right)k\right] >$$

$$M^3N\left[2k^6+k^3+\frac{k^2}{4}+\left(d+\left(1-\frac{\alpha+\beta}{2}\right)\frac{\beta-\alpha}{5d}\right)k\right] = (4.9).$$

The last inequality holds because

$$(1-q)\left(d+\left(1-\frac{2\alpha+\beta}{3}\right)\frac{\beta-\alpha}{5d}\right) > \left(d+\left(1-\frac{\alpha+\beta}{2}\right)\frac{\beta-\alpha}{5d}\right) \Leftrightarrow$$

$$\left(\frac{\alpha+\beta}{2}-\frac{2\alpha+\beta}{3}\right)\frac{\beta-\alpha}{5d} > q\left(d+\left(1-\frac{2\alpha+\beta}{3}\right)\frac{\beta-\alpha}{5d}\right) \Leftrightarrow$$

$$\frac{\beta-\alpha}{6}\cdot\frac{\beta-\alpha}{5d} > q\left(d+\left(1-\frac{2\alpha+\beta}{3}\right)\frac{\beta-\alpha}{5d}\right),$$

which is true since $\left(1-\frac{2\alpha+\beta}{3}\right)\frac{\beta-\alpha}{5d} < 1$ and $q=\left(\frac{\beta-\alpha}{10d}\right)^2$.

Proof of Lemma 4.4.6

We will repeatedly use the fact that, in any quasi-balanced linear arrangement, $S_L^\ell, S_R^\ell, S_L^r$, and S_R^r have all size at least $(1-q)kNM$, where $q=\left(\frac{\beta-\alpha}{10d}\right)^2$.

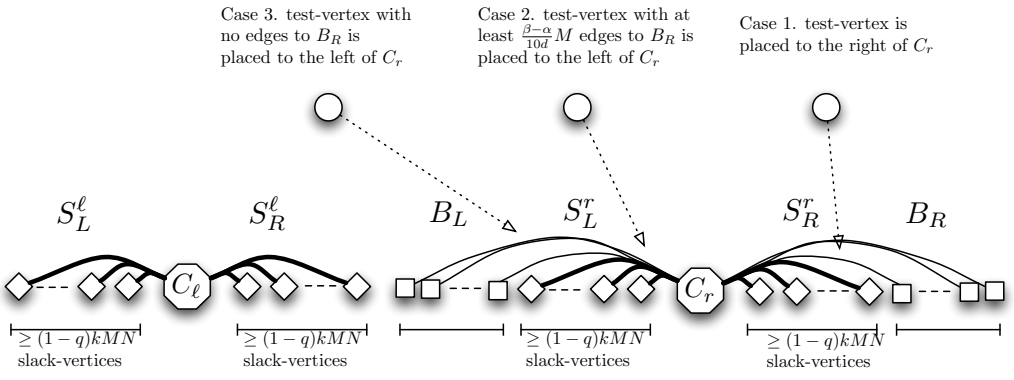


Figure 4.13: Overview of the cases considered in the proof of Lemma 4.4.6

We start by proving that any test-vertex that is placed to the right of C_r (Case 1 in Figure 4.13) will have edges of total value at least $(1-q)M^2N\left(d+\frac{\beta-\alpha}{5d}\right)k$.

Let $p > 0$ and suppose that test-vertex t is placed to the right of $p(1-q)kMN$ slack-vertices of C_r. Since t is placed to the right of C_r, we might only decrease the cost by assuming that all bit-vertices adjacent to t are in B_R. Then the cost of the edges incident to t is at least

$$\underbrace{dM(1-p)(1-q)kNM}_{\text{edges to bit-vertices}} + \underbrace{\left(d - \frac{\beta-\alpha}{10d}\right)Mp(1-q)kNM}_{\text{edge to } C_r} +$$

$$\underbrace{\frac{\beta-\alpha}{10d}M(2+p)(1-q)kNM}_{\text{edge to } C_\ell} =$$

$$M^2N(1-q)k\left(d(1-p) + \left(d - \frac{\beta-\alpha}{10d}\right)p + \frac{\beta-\alpha}{10d}(2+p)\right) =$$

$$M^2N(1-q)k\left(d + \frac{\beta-\alpha}{5d}\right).$$

Recall that Γ is the set of test-vertices with at least $\left(d - \frac{\beta-\alpha}{10d}\right)M$ edges to B_L. Now let $p > 0$ and suppose that t is placed to the left of $p(1-q)kMN$ slack-vertices that are to the left of C_r. On the one hand, if t is not in Γ then it has at least $\frac{\beta-\alpha}{10d}M$ edges to B_R (Case 2 in Figure 4.13) and the cost of the edges incident to t are at least

$$\underbrace{\left(d - \frac{\beta-\alpha}{10d}\right)M\max[1-p,0](1-q)kNM}_{\text{edges to } B_L} + \underbrace{\left(\frac{\beta-\alpha}{10d}\right)M(1+p)(1-q)kNM}_{\text{edges to } B_R} +$$

$$\underbrace{(d - \frac{\beta-\alpha}{10d})Mp(1-q)kNM}_{\text{edge to } C_r} + \underbrace{\frac{\beta-\alpha}{10d}M(2-p)(1-q)kNM}_{\text{edge to } C_\ell},$$

which can be written as

$$M^2N(1-q)k\left(\left(d - \frac{\beta-\alpha}{10d}\right)\max[1-p,0] + \frac{\beta-\alpha}{10d}(1+p)\right) +$$

$$M^2N(1-q)k\left(\left(d - \frac{\beta-\alpha}{10d}\right)p + \frac{\beta-\alpha}{10d}(2-p)\right),$$

which is easily seen to be at least

$$M^2N(1-q)k\left(d + \frac{\beta-\alpha}{5d}\right).$$

On the other hand, if t is in Γ (Case 3 in Figure 4.13) the cost of the edges incident to t is at least

$$\underbrace{\left(d-\frac{\beta-\alpha}{10d}\right)M\max[1-p,0](1-q)kNM}_{\text{edges to } B_L}+\underbrace{\left(d-\frac{\beta-\alpha}{10d}\right)Mp(1-q)kNM}_{\text{edge to } C_r}+$$

$$\underbrace{\frac{\beta-\alpha}{10d}M(2-p)(1-q)kNM}_{\text{edge to } C_\ell},$$

which can be written as

$$M^2N(1-q)k\left(\left(d-\frac{\beta-\alpha}{10d}\right)\max[1-p,0]+\left(d-\frac{\beta-\alpha}{10d}\right)p+\frac{\beta-\alpha}{10d}(2-p)\right),$$

and this is easily seen to be at least (bound tight when $p=1$)

$$M^2N(1-q)kd.$$

The above case distinction concludes the proof of Lemma 4.4.6.

4.4.5 Inapproximability Gap

By using Theorem 4.1.4, we have provided a probabilistic reduction Γ from SAT to weighted optimal linear arrangement. For any fixed $\epsilon>0$, given an instance ϕ of SAT of size n, Γ produces a weighted optimal linear arrangement instance G in time $2^{O(n^\epsilon)}$ satisfying with high probability:

- *(Completeness)* If ϕ is satisfiable then G has a linear arrangement with cost at most

$$M^3N\left[2k^6+k^3+\frac{k^2}{4}+\left(d+\left(1-\frac{2\beta+\alpha}{3}\right)\frac{\beta-\alpha}{5d}\right)k\right].$$

- *(Soundness)* If ϕ is not satisfiable then all linear arrangements have cost at least

$$M^3N\left[2k^6+k^3+\frac{k^2}{4}+\left(d+\left(1-\frac{\alpha+\beta}{2}\right)\frac{\beta-\alpha}{5d}\right)k\right].$$

The hardness of approximation result now follows by recalling that (i) α,β, and k are all functions of parameter d of Theorem 4.1.4, which in turn is a function of ϵ and (ii) $\alpha<\beta$.

4.4.6 Unweighted OLA

In this section we will show that the analysis for weighted OLA can also be used in the unweighted case. Let the graph G be defined as in the construction of weighted OLA (see Section 4.4.1). Note that the edges with weight different than 1 are incident to either C_r or C_ℓ. Recall that $k = \left(\frac{10d}{\beta-\alpha}\right)^8$. Now consider the graph G_U obtained from G, where we

1. replace vertices C_r and C_ℓ by two "huge" cliques of size k^6M, called C'_r and C'_ℓ respectively;

2. each edge from a vertex v to C_i with weight w is replaced by w edges from v to w different vertices of C'_i, for $i \in \{c, l\}$; and

3. we distribute the edges to a clique C'_i so that the difference in the degree of two vertices of a clique is no bigger than one.

With this construction, there are at most $2kMN \cdot k^4\frac{M}{N} + MN \cdot k^2\frac{M}{N} + dM^2 = M^2(2k^5 + k^2 + d)$ edges adjacent to C'_i for $i = \{c, l\}$. Since the edges adjacent to a clique are evenly distributed among its vertices, we have that a vertex of C'_r or C'_ℓ has less than M edges to vertices not belonging to the cliques.

We will now see that the soundness and completeness analyses for G_U do not differ much from the analyses done for G.

Completeness Let π' be the linear arrangement of G_U obtained from the linear arrangement π of G as defined in the completeness analysis of OLA (Section 4.4.3), where the vertices of C'_ℓ and C'_r are placed on the location of C_ℓ and C_r, respectively. By noting that the number of vertices of the cliques is relatively small (of order M) and that the total number of edges is $4kMN \cdot k^4\frac{M}{N} + NM \cdot k^2\frac{M}{N} + M2dM = O(M^2)$, it follows that the value of π' of G_U is only $o(M^3N)$ greater than the value of π of G and the same bound (4.8) holds (for big enough M and N).

Soundness We say that a clique is *divided* in a linear arrangement π if there exists a bit-, slack-, or test-vertex w and two vertices of the clique u and v such that $\pi(u) < \pi(w) < \pi(v)$. Note that if neither C'_ℓ nor C'_r is divided in an optimal solution of G_U, it follows, by treating the cliques as the vertices C_ℓ and C_r respectively, that the value of an optimal linear arrangement of G_U must be at least as big as the value of an optimal linear arrangement of G. Thus, the following lemma is enough to complete the soundness analysis of G_U.

Lemma 4.4.7 *In any optimal linear arrangement π of G_U, the cliques C'_r and C'_ℓ are not divided.*

Proof. We will present our arguments for the clique C'_r. Since the arguments are the same for C'_ℓ, we leave this case to the reader. Given an optimal linear arrangement π of G_U, let l and r denote the left-most and right-most vertices of C'_r in π, respectively, and let $S = \{v \text{ is a slack-, test-, or bit-vertex} : \pi(l) < \pi(v) < \pi(r)\}$. Suppose, toward contradiction, that S is non-empty. Select

$$v_L = \arg\min_{v \in S}(\pi(v)) \quad \text{and} \quad v_R = \arg\max_{v \in S}(\pi(v))$$

(the left-most vertex and right-most vertex of S, respectively). Let A denote the set of vertices of C'_r that are placed between l and v_L, i.e., $A = \{v \text{ is a vertex of } C'_r : \pi(l) < \pi(v) < \pi(v_L)\}$. Similarly, let B denote the set of vertices of C'_r that are placed between v_R and r.

Note that the selection of v_L and v_R implies that either $|A|$ or $|B|$ is less than $k^6M/2$. Suppose $|A| \leq k^6M/2$ and consider what happens with the cost if we swap places of l and v_L.

1. *Edges leaving l.* The number of edges from l to vertices outside the clique is at most M. The cost of these edges will thus *increase* by at most $(\pi(v_L) - \pi(l))M$. The cost of the edges from l to vertices in A will *increase* by

   $$\sum_{i \in A} \underbrace{(\pi(v_L) - \pi(i))}_{\text{new cost}} - \underbrace{(\pi(i) - \pi(l))}_{\text{old cost}} \leq \sum_{i \in A} (\pi(v_L) - \pi(l)) + \pi(l) - \pi(i),$$

   which is bounded from above by

   $$\sum_{i=1}^{|A|} (\pi(v_L) - \pi(l)) - i \leq (\pi(v_L) - \pi(l))|A| - |A|^2/2.$$

   Finally, the cost of the edges from l to the vertices of C'_r that are not in A will *decrease* by $(\pi(v_L) - \pi(l))(k^6M - |A|) \geq (\pi(v_L) - \pi(l))k^6M/2$.

2. *Edges leaving v_L.* Note that slack-, bit-, and test-vertices have degree at most $2dM$ (for large enough N). The cost of the edges incident to v_L will thus *increase* by at most $(\pi(v_L) - \pi(l))2dM$.

Summing up the above observations we have that the increase of cost will be at most

$$(\pi(v_L) - \pi(l))(M + 2dM + |A| - k^6M/2) - |A|^2/2 < 0,$$

i.e, the cost will decrease which contradicts the optimality of the linear arrangement. The last inequality follows easily by recalling that (i) $\pi(v_L) - \pi(l) \leq 2k^6M$ (since by the definition of v_L there can only be vertices belonging to the cliques that are placed between l and v_L) and (ii) $|A| \leq k^6M/2$ (by assumption).

The remaining case when $|B| \leq k^6M/2$ is symmetric and omitted. □

4.5 Boosting the Hardness Factor for Maximum Edge Biclique

In this section we prove Theorem 4.1.2. The techniques used here have previously been used by Khot [Kho06] for the balanced bipartite clique problem and by Berman & Schnitger [BS92a] for the clique problem, and are included for the sake of completeness.

The following lemma shows that with high probability we can boost the hardness factor for maximum edge biclique without increasing the instance size too much.

Lemma 4.5.1 *Let $G(V, W, E)$ be a bipartite graph with $|V| = |W| = n$. Then for an integer k and a constant $\delta > 0$, we can construct a bipartite graph $H(V_*, W_*, E_*)$ in randomized time $O(n^2/\delta^k)$, with $|V_*| = |W_*| = O(n^2/\delta^k)$, so that with high probability:*

- *(Completeness) For $\beta \geq \delta$, if G has an edge biclique of size $\beta|V||W|$ then H has an edge biclique of size $\frac{1}{4} \cdot \beta^k |V_*||W_*|$.*
- *(Soundness) For $\alpha \geq \delta$, if G has no edge biclique of size $\alpha|V||W|$ then H has no edge biclique of size $4\alpha^k|V_*||W_*|$.*

Proof. Let $G^k(V', W', E')$ denote the product graph defined as follows:

- $V' = V^k, W' = W^k$. Thus $|V'| = |W'| = n^k$.
- $\{(v_1, v_2, \ldots, v_k), (w_1, w_2, \ldots, w_k)\} \in E' \Leftrightarrow \forall i, j, 1 \leq i, j \leq k, \{v_i, w_j\} \in E$.

Observe that if $V_1 \subseteq V, W_1 \subseteq W$ form an edge biclique of G then V_1^k, W_1^k form an edge biclique of G^k. Moreover, it is easy to see that any *maximal* edge biclique of G^k must be of the form V_1^k, W_1^k where V_1, W_1 is an edge biclique of G.

Let V_* be $O(n^2/\delta^k)$ vertices of V^k that are picked independently uniformly at random. Similarly, let W_* be $O(n^2/\delta^k)$ vertices of W^k that are picked independently uniformly at random. Graph $H(V_*, W_*, E_*)$ is now the subgraph of G^k induced on the vertices V_*, W_*. Note that we do not need to actually create graph G^k to obtain H. Indeed we can construct the sets V_* and W_* by randomly picking k vertices $O(n^2/\delta^k)$ times from V and W, respectively. Given V_*, W_* it is easy to complete the edge set E_*. It follows that we can create graph H in randomized time $O(n^2/\delta^k)$.

We now proceed by analyzing the completeness case. Assume that G has an edge biclique $V_1 \subseteq V, W_1 \subseteq W$ with $|V_1| = \beta_1 n, |W_1| = \beta_2 n$ so that $\beta_1 \cdot \beta_2 n^2 = \beta n^2$, where $\beta \geq \delta$. Note that both β_1 and β_2 are at most 1 and at least $\beta \geq \delta$.

For every vertex in V_* the probability that it belongs to V_1^k is β_1^k. Thus the expected size of $|V_* \cap V_1^k|$ is $\beta_1^k|V_*|$. As the vertices in V_* are picked randomly and independently from V^k, we can use Chernoff bounds[3]

$$\Pr\left[|V_* \cap V_1^k| \leq \frac{1}{2}\beta_1^k|V_*|\right] \leq 2^{-\Omega(\beta_1^k|V_*|)} \leq 2^{-\Omega(\delta^k|V_*|)} = 2^{-\Omega(n^2)}.$$

Similarly, w.h.p., $|W_* \cap W_1^k| \geq \frac{1}{2}\beta_2^k|W_*|$. Clearly, $V_* \cap V_1^k, W_* \cap W_1^k$ form an edge biclique of H and with high probability it has size at least $\frac{1}{2}\beta_1^k|V_*| \cdot \frac{1}{2}\beta_2^k|W_*| = \frac{1}{4}\beta^k|V_*||W_*|$. Hence we have proved the completeness case.

We now consider the soundness case. For some $\alpha \geq \delta$, assume that graph G has no edge biclique of size $\alpha|V||W|$ and thus G^k has no edge biclique of size $\alpha^k|V^k||W^k|$. It suffices to prove that w.h.p., for *every maximal* edge biclique V_1^k, W_1^k of G^k, we have that $|V_* \cap V_1^k| \cdot |W_* \cap W_1^k| < 4\alpha^k|V_*||W_*|$. Note that the number of maximal edge bicliques of G^k is bounded by 2^{2n}.

Fix any maximal edge biclique V_1^k, W_1^k of G^k. Then there exists α_1 and α_2 with $\alpha_1 \cdot \alpha_2 = \alpha$ so that $|V_1^k| \leq \alpha_1^k|V^k|$ and $|W_1^k| \leq \alpha_2^k|W^k|$. Note that both α_1 and α_2 are at most 1 and at least $\alpha \geq \delta$. For every vertex in V_* the probability that it belongs to V_1^k is at most α_1^k. Thus the expected size of $|V_* \cap V_1^k|$ is at most $\alpha_1^k|V_*|$. Again using Chernoff bounds,

$$\Pr\left[|V_* \cap V_1^k| \geq 2\alpha_1^k|V_*|\right] \leq 2^{-\Omega(\alpha_1^k|V_*|)} \leq 2^{-\Omega(\delta^k|V_*|)} = 2^{-\Omega(n^2)}.$$

Similarly, the size of $|W_* \cap W_1^k|$ is greater than $2\alpha_2^k|V_*|$ with probability at most $2^{-\Omega(n^2)}$. Now taking the union bound over all possible (at most 2^{2n} many) maximal edge bicliques of G^k proves the soundness case. □

Proof of Theorem 4.1.2

Fix $\epsilon > 0$ to be an arbitrarily small constant. Let $G(V, W, E)$ be the bipartite graph given by the reduction in Section 4.2.1. Let $N = |V| = |W|$. If the size of the original SAT instance is n then note that $N = 2^{O(n^\epsilon)}$. By the completeness and soundness (see Section 4.2.5) there exist constants $\beta, \alpha > 0$ that depend on ϵ with $\beta > \alpha$ so that (with high probability)

- *(Completeness)* If the SAT instance is satisfiable, then G has an edge biclique of size $\beta|V||W|$.

[3]Let random variables $X_1, X_2, \ldots, X_n$ be independent random variables taking on values 0 or 1. Then if we let $X = \sum_i X_i$ and μ be the expectation of X, for any $\epsilon > 0$ $\Pr[X > (1+\epsilon)\mu] \leq \left(\frac{e^\epsilon}{(1+\epsilon)^{1+\epsilon}}\right)^\mu$ and $\Pr[X < (1-\epsilon)\mu] \leq e^{-\epsilon^2\mu/2}$.

- *(Soundness)* If the SAT instance is not satisfiable, then G has no edge biclique of size $\alpha|V||W|$.

We now apply Lemma 4.5.1 with $k = \frac{\log(2\cdot 16)}{\log(\beta/\alpha)} \log N$ and $\delta = \alpha$ to obtain a graph $H(V_*, W_*, E_*)$ of size $S = O(N^2/\delta^k)$ that with high probability satisfies:

- *(Completeness)* If the SAT instance is satisfiable, then H has an edge biclique of size $\frac{1}{4}\beta^k|V_*||W_*|$.
- *(Soundness)* If the SAT instance is not satisfiable, then H has no edge biclique of size $4\alpha^k|V_*||W_*|$.

Thus the hardness factor is

$$16(\alpha/\beta)^k = \frac{1}{\frac{1}{16}(\beta/\alpha)^k} = \frac{1}{\frac{1}{16}(\beta/\alpha)^{\frac{\log(2\cdot 16)}{\log(\beta/\alpha)}\log N}} \leq \frac{1}{2^{\log N}} = \frac{1}{N}.$$

On the other hand,

$$\begin{aligned} S &= O(N^2/\delta^k) = O(N^2/\alpha^k) = O(N^2 2^{k\log(1/\alpha)}) \\ &= O(N^2 2^{\frac{\log(2\cdot 16)}{\log(\beta/\alpha)}\log N \log(1/\alpha)}) = O(N^{2+\frac{\log(2\cdot 16)}{\log(\beta/\alpha)}\log(1/\alpha)}). \end{aligned}$$

As α and β only depend on ϵ, the hardness factor can be expressed as $1/S^{\epsilon'}$, where ϵ' is a function of ϵ. We conclude by noting that, assuming $SAT \notin BPTIME(2^{n^\epsilon})$, it is hard to approximate maximum edge biclique on a graph of size S within a factor $1/S^{\epsilon'}$, for some ϵ' that only depends on ϵ.

4.6 Conclusions

In this chapter we have proved the first hardness of approximation results for the classical optimal linear arrangement and (uniform) sparsest cut graph problems. We also improved previous hardness results for the maximum edge biclique problem by using a more standard assumption.

All our results are obtained by using the Quasi-random PCP construction by Khot [Kho06]. Hence, our results are under the assumption that SAT is not solvable in probabilistic time 2^{n^ϵ}, where n is the instance size and $\epsilon > 0$ can be made arbitrarily close to 0. Moreover, the hardness factors obtained for optimal linear arrangement and sparsest cut by using our reductions from the Quasi-random PCP are tiny. This raises two prominent open problems:

1. Show that it is hard to approximate the addressed problems, by using a weaker assumption (ideally $P \neq NP$).

2. Provide a constant factor approximation algorithm for optimal linear arrangement and uniform sparsest cut, or rule out this possibility.

A natural approach for proving that uniform sparsest cut has no constant approximation algorithm would be to assume the unique games conjecture [Kho02] and to do a similar reduction as done for non-uniform sparsest cut [KV05; CKK$^+$06]. In [AKK$^+$08], Arora et al. pointed out that such an approach only seems to work if the unique games conjecture would be true on expander graphs, which they disproved. This makes it unlikely to obtain a hardness result for (uniform) sparsest cut with the above described approach.

Chapter 5

Job and Flow Shops

5.1 Introduction

We consider the classical job shop scheduling problem together with the more restricted flow shop problem. In the *job shop problem* we have a set of n jobs that must be processed on a given set M of m machines. Each job j consists of a chain of μ_j operations $O_{1j}, O_{2j}, \ldots, O_{\mu_j j}$. Operation O_{ij} must be processed during p_{ij} time units without interruption on machine $m_{ij} \in M$. A feasible schedule is one in which each operation is scheduled only after all its preceding operations have been completed, and each machine processes at most one operation at a time. For any given schedule, let C_j be the completion time of the last operation of job j. We consider the natural and typically considered objectives of minimizing the *makespan* $C_{\max} = \max_j C_j$ and minimizing the *sum of weighted completion times* $\sum w_j C_j$. The goal is to find a feasible schedule which minimizes the considered objective function. In the notation of Graham et al. [GLLK79] this problem is denoted as $J||\gamma$, where γ denotes the objective function to be minimized.

In the *flow shop scheduling problem* $(F||\gamma)$ each job has exactly one operation per machine, and all jobs go through all the machines in the same order. A natural generalization of the flow shop problem is to not require jobs to be processed on all machines, i.e., a job still requests the machines in compliance with some fixed order but may skip some of them. We will refer to this more general version as *generalized flow shops* or *flow shops with jumps* $(F|jumps|\gamma)$. Generalized flow shop scheduling (and thus flow shop) is a special case of acyclic job shop scheduling, which only requires that within each job all operations are performed on different machines, which in turn is a special case of job shop scheduling. See Figure 5.1 for example instances of the addressed problems.

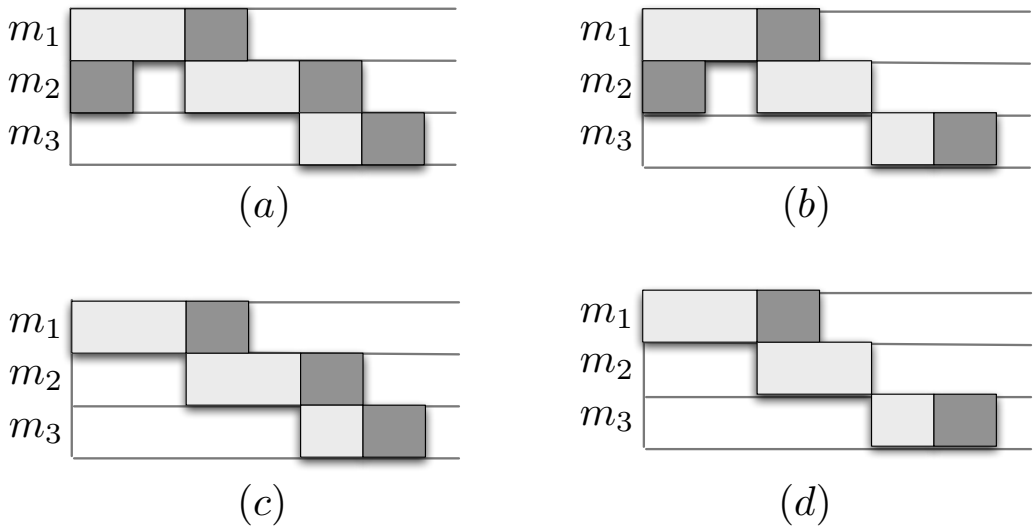


Figure 5.1: Example of scheduling instances with three machines and two jobs depicted in light and dark gray. a) Job shop instance – jobs may have several operations on each machine. b) Acyclic job shop instance – a job has at most one operation per machine. c) Flow shop instance – each job has exactly one operation per machine and all the jobs are processed on the machines in the same order. d) Generalized flow shop instance – jobs have at most one operation per machine and jobs process the machines in the same order.

5.1.1 Literature Review

Job and flow shops have long been identified as having a number of important practical applications and have been widely studied since the late 50's (the reader is referred to the survey papers of Lawler et al. [LLKS93] and of Chen, Potts & Woeginger [CPW98]). To find a schedule that minimizes the makespan, or one that minimizes the sum of completion times, was proved to be strongly NP-hard for flow shops (and thus job shops) in the 70's, even for severely restricted instances (see e.g. [CPW98]).

From then, many approximation methods have been proposed. Since the quality of an approximation algorithm is measured by comparing the returned solution value with a polynomial time computable lower bound on the optimal value, the latter is very important. For a given instance, let C^*_{max} denote the minimum makespan taken over all possible feasible schedules. If D denotes the length of the longest job (the *dilation*), and C denotes the time units requested by all jobs on the most loaded machine (the *congestion*), then $\mathrm{lb} = \max[C, D]$ is a known trivial lower bound on C^*_{max}. There is no known significantly stronger lower bound on C^*_{max}, and all the proposed approximation algorithms for flow shops, acyclic job shops, job shops, and the more constrained case of permutation flow shops have been analyzed with respect to this lower bound (see, e.g., [LMR94; SSW94; LMR99; GPSS01; FS02; NS08]).

Even though the trivial lower bound might seem weak, a surprising result by Leighton, Maggs & Rao [LMR94] says that for acyclic job shops, if all operations are of unit length, then $C^*_{max} = \Theta(\mathrm{lb})$. If we allow operations of any length, then Feige & Scheideler [FS02] showed that $C^*_{max} = O(\mathrm{lb} \cdot \log \mathrm{lb} \log\log \mathrm{lb})$ for acyclic job shops. They also showed their analysis to be nearly tight by providing acyclic job shop instances with $C^*_{max} = \Omega(\mathrm{lb} \cdot \log \mathrm{lb} / \log\log \mathrm{lb})$. The proofs of the upper bounds in [LMR94; FS02] are nonconstructive and make repeated use of (a general version) of the Lovàsz local lemma. Algorithmic versions of the Lovàsz local lemma [Bec91] and the general version [CS00], have later been used to obtain constructive upper bounds yielding a constant approximation algorithm for unit time acyclic job shops [LMR99] and an $O(\log \mathrm{lb}^{1+\epsilon})$-approximation algorithm for acyclic job shops [CS00], where $\epsilon > 0$ is any constant.

Feige & Scheideler's upper bound for acyclic job shops [FS02] is also the best upper bound for the special case of flow shops. As flow shops have more structure than acyclic job shops and no flow shop instances with $C^*_{max} = \omega(\mathrm{lb})$ were known, one could hope for a significantly better upper bound for flow shops. The existence of such a bound was raised as an open question in [FS02]. The strength of the lower bound lb is better understood for the related *permutation* flow shop problem, that is a flow shop problem with the additional constraint that each machine processes all the jobs in the same order. Potts, Shmoys &

Williamson [PSW91] gave a family of permutation flow shop instances with $C^*_{max} = \Omega(\text{lb} \cdot \sqrt{\min[m,n]})$. This lower bound was recently shown to be tight, by Nagarajan & Sviridenko [NS08], who gave an approximation algorithm that returns a permutation schedule with makespan $O(\text{lb} \cdot \sqrt{\min[m,n]})$.

The best approximation algorithms known for general $J||C_{max}$ are an approximation algorithm by Shmoys, Stein & Wein [SSW94] with performance guarantee $O((\log \text{lb})^2 / \log\log \text{lb})$; later improved by a $\log\log \text{lb}$ factor by Goldberg, Paterson, Srinivasan and Sweedyk [GPSS01].

When preemption is allowed, every operation can be temporarily interrupted and resumed later without any penalty. For any $\varepsilon > 0$, it is well-known that with only ε loss in the approximation factor, the preemptive job shop scheduling problem is equivalent to the nonpreemptive job shop scheduling problem with unit processing times (see e.g. [BKS06]). For acyclic job shop and flow shop scheduling with preemption, the best known result is due to Feige & Scheideler [FS02] who showed that there always exists a preemptive schedule within a $O(\log\log \text{lb})$ factor of lb. For the general preemptive job shop problem, Bansal et al. [BKS06] showed an $O(\log m / \log\log m)$-randomized approximation algorithm, and a $(2+\varepsilon)$-approximation for a constant number of machines.

Whether the above algorithms for $J||C_{\max}$ and $F||C_{\max}$ are tight or even nearly tight, is a long standing open problem (see "Open problems 6 and 7" in [SW99]). The only known inapproximability result is due to Williamson et al. [WHH+97], and states that when the number of machines and jobs are part of the input, it is NP-hard to approximate $F||C_{\max}$ with unit time operations, and at most three operations per job, within a ratio better than $5/4$.

The situation is similar for the weighted sum of completions times objective. Queyranne & Sviridenko [QS02] showed that an approximation algorithm for the above mentioned problems that produces a schedule with makespan a factor $O(\rho)$ away from the lower bound lb can be used to obtain an $O(\rho)$-approximation algorithms for other objectives, including the sum of weighted completion times. As a result, the above mentioned approximation guarantees for the makespan criteria also hold for the the sum of weighted completion times objective. The only known inapproximability result is by Hoogeveen, Schuurman & Woeginger[HSW01], who showed that $F||\sum C_j$ is NP-hard to approximate within a ratio better than $1+\epsilon$ for some small $\epsilon > 0$.

Another open problem [SW99] is to understand whether there is a PTAS for the general job shop problem with a constant number of machines. For those instances where the number of machines and the number of operations per job are constant, Jansen, Solis-Oba & Sviridenko [JSOS03] gave a PTAS for the makespan criteria. A similar result was later obtained by Fishkin, Jansen & Mastrolilli [FJM08] for the weighted sum of completion times objective.

5.1.2 Results and Overview of Techniques

Feige & Scheideler [FS02] showed their analysis to be essentially tight for acyclic job shops. As flow shops are more structured than acyclic job shops, they raised as an open question whether the upper bound for flow shop scheduling can be improved significantly. Our first result resolves this question negatively.

Theorem 5.1.1 *There exist flow shop instances with optimal makespan* $\Omega(\mathrm{lb} \cdot \log \mathrm{lb} / \log \log \mathrm{lb})$.

Proof overview. The construction of job shop instances with "large" makespan is presented in Section 5.2.1 and serves as a good starting point for reading Section 5.2.2 where the more complex construction of flow shop instances with "large" makespan is presented.

The job shop construction closely follows the construction in [FS02] with the main difference being that we do not require all operations of a job to be of the same length, which leads to a slightly better analysis. The main idea is to introduce jobs of different "frequencies", with the property that two jobs of different frequencies essentially cannot be processed at the same time in any feasible schedule. Hence, a job shop instance with d jobs of different frequencies, all of them of length D, has optimal makespan $\Omega(d \cdot D)$. Moreover, the construction satisfies $\mathrm{lb} = C = D = 3^d$ and it follows that the job shop instance has optimal makespan $\Omega(d \cdot 3^d)$, which can be written as $\Omega(\mathrm{lb} \cdot \log \mathrm{lb})$.

The construction of flow shop instances with "large" makespan is more complicated, as each job is required to have exactly one operation for every machine, and all jobs are required to go through all the machines in the same order. The main idea is to start with the aforementioned job shop construction, which has "very cyclic" jobs, i.e., jobs have many operations on a single machine. The flow shop instance is then obtained by "copying" the job shop instance several times and instead of having cyclic jobs we let the i-th "long-operation" of a job to be processed by a machine in the i-th copy of the original job shop instance. Finally, we insert additional zero-length operations to obtain a flow shop instance. By carefully choosing the different frequencies we can show that the constructed flow shop instance will have essentially the same optimal makespan as the job shop instance we started with. The slightly worse bound, $\Omega(\mathrm{lb} \cdot \log \mathrm{lb} / \log \log \mathrm{lb})$ instead of $\Omega(\mathrm{lb} \cdot \log \mathrm{lb})$, arises from additional constraints on the design of frequencies. □

If we do not require a job to be processed on *all* machines, i.e. generalized flow shops, we prove that it is hard to improve the approximation guarantee. Theorem 5.1.2 shows that generalized flow shops, with the objective to either

minimize makespan or sum of completion times [1], have no constant approximation algorithm unless $P = NP$.

Theorem 5.1.2 *For all sufficiently large constants K, it is* NP*-hard to distinguish between generalized flow shop instances that have a schedule with makespan $2K \cdot \text{lb}$ and those that have no solution that schedules more than half of the jobs within $(1/8)K^{\frac{1}{25}(\log K)} \cdot \text{lb}$ time units. Moreover this hardness result holds for generalized flow shop instances with bounded number of operations per job, that only depends on K.*

Proof overview. The reduction is from the NP-hard problem of deciding whether a graph G with degree bounded by a constant Δ can be colored using "few" colors or has no "large" independent set (see Theorem 5.1.5). In Section 5.3.1 we present a relatively easy reduction to the job shop problem which serves as a good starting point for reading the more complex reduction for the generalized flow shop problem presented in Section 5.3.2. The main idea is as follows. Given a graph G with bounded degree Δ, we construct a generalized flow shop instance S, where all jobs have the same length D and all machines the same load $C = D$. Hence, $\text{lb} = C = D$. Instance S has a set of jobs for each vertex in G. By using jobs of different frequencies, as done in the gap construction, we have the property that "many" of the jobs corresponding to adjacent vertices *cannot* be scheduled in parallel in any feasible schedule. On the other hand, by letting jobs skip the machines corresponding to non-adjacent vertices, jobs corresponding to an independent set in G *can* be scheduled in parallel (their operations can overlap in time) in a feasible schedule. For the reduction to be polynomial it is crucial that the number of frequencies is relatively small. However, to ensure the desired properties, jobs corresponding to adjacent vertices must be of different frequencies. We resolve this by using the fact that G has bounded degree. Since the graph G has degree of at most Δ we can in polynomial time partition its vertices into $\Delta + 1$ independent sets. As two jobs only need to be assigned different frequencies if they correspond to adjacent vertices, we only need a constant $(\Delta + 1)$ number of frequencies.

The analysis follows naturally: a set of jobs corresponding to an independent set can be scheduled in parallel. Hence, if the graph G can be colored with few, say F, colors then there is a schedule of S with makespan $O(F \cdot \text{lb})$. Finally, if there is a schedule where at least half the jobs finish within $L \cdot \text{lb}$ time units then jobs corresponding to at least a fraction $\Omega(1/L)$ of the vertices overlap at some point in the schedule. As the jobs overlap, they correspond to a fraction $\Omega(1/L)$

[1] Note that Theorem 5.1.2 implies that for sufficiently large constants K, it is NP-hard to distinguish whether $\sum C_j \leq n \cdot 2K \cdot \text{lb}$ or $\sum C_j \geq \frac{n}{2} \cdot (1/8) \cdot K^{\frac{1}{25}(\log K)} \cdot \text{lb}$, where n is the number of jobs.

of vertices that form an independent set. It follows that if the graph has no large independent set, then the generalized flow shop instance has no short schedule. □

By making a stronger assumption, we give a hardness result that essentially shows that the current approximation algorithms for generalized flow shops (and acyclic job shops), with both makespan and sum of weighted completion times objectives, are tight.

Theorem 5.1.3 *Let $\epsilon > 0$ be an arbitrarily small constant. There is no $O\left((\log \mathrm{lb})^{1-\epsilon}\right)$-approximation algorithm for $F|jumps|C_{max}$ or $F|jumps|\sum C_j$, unless $NP \subseteq ZTIME(2^{\log n^{O(1/\epsilon)}})$.*

Proof overview. The construction of the generalized flow shop instance is the same as in the proof of Theorem 5.1.2. To obtain the stronger result we use a stronger hardness result for graph coloring (see Theorem 5.1.6). The tricky part is that the graph no longer has a small bounded degree. We overcome this difficulty by a randomized process that preserves the desired properties of the graph with an overwhelming probability (see Lemma 5.3.1). □

For flow shops, the consequences of Theorem 5.1.1 and Theorem 5.1.3 are among others that in order to improve the approximation guarantee, it is necessary to (i) improve the used lower bound on the optimal makespan and (ii) use the fact that a job needs to be processed on *all* machines.

In [JSOS03], a PTAS was given for the job shop problem, where both the number of machines and the number of operations per job are assumed to be constant. Our second result shows that both these restrictions are necessary to obtain a PTAS (that one needs to restrict the number of machines follows from the work in [WHH+97]).

Theorem 5.1.4 *Problem $J2||C_{\max}$ has no PTAS unless $NP \subseteq DTIME(n^{O(\log n)})$.*

Proof overview. In Section 5.4, we prove the result by presenting a gap-preserving reduction from the independent set problem in cubic graphs, i.e., graphs where all vertices have degree three (see Theorem 5.1.7).

Given a cubic graph G we construct an instance S of $J2||C_{\max}$ as follows. The instance has a "big" job, called j_b, whose length will equal the makespan in the completeness case. Its operations are divided into four parts, called the *edge-, tail-, slack-,* and *remaining-part*. There is also a vertex job for each vertex. We again use the technique of introducing different frequencies of jobs; this time to ensure that, without delaying job j_b, two jobs corresponding to adjacent vertices cannot both complete before the end of the tail-part of job j_b.

The analysis now follows from selecting the lengths of the different parts of j_b such that in the completeness case we can schedule all jobs, corresponding to a "big" independent set of G, in parallel with the edge- and tail-part of job j_b and the remaining jobs are scheduled in parallel with the slack- and remaining-part of job j_b. In contrast, in the soundness case, as G has no "big" independent set, we can, without delaying the schedule, only schedule relatively few jobs in parallel with the edge- and tail-part of job j_b. The remaining jobs, relatively many, will then require more time units than the total length of the slack- and remaining-part of job j_b and it follows that the schedule will have makespan larger than the length of j_b.

The reduction runs in time $n^{O(f)}$, where f is the number of frequencies. With our current techniques we need $O(\log n)$ frequencies and hence the assumption used in the statement of Theorem 5.1.4.

□

5.1.3 Preliminaries

When considering a schedule we shall say that two jobs (or operations) overlap or are scheduled in parallel for t time units if t time units of them are processed at the same time on different machines. For a given graph G, we let $\chi(G)$ and $\alpha(G)$ denote the chromatic number of G and the size of a maximum independent set of G, respectively. We shall also denote the maximum degree of graph G by $\Delta(G)$, where we sometimes drop G when it is clear from the context.

Our reductions to the job shop problem with unbounded number of machines use results by Khot [Kho01], who proved that even though we know that the graph is colorable using K colors it is NP-hard to find a coloring that uses less than $K^{\frac{1}{25}(\log K)}$ colors, for sufficiently large constants K. The result was obtained by presenting a polynomial time reduction that takes as input a SAT formula ϕ together with a sufficiently large constant K and outputs an n-vertex graph G with degree at most $2^{K^{O(\log K)}}$ such that (completeness) if ϕ is satisfiable then G can be colored using at most K colors and (soundness) if ϕ is not satisfiable then G has no independent set containing $n/K^{\frac{1}{25}(\log K)}$ vertices (see Section 6 in [Kho01]). Note that the soundness case implies that any feasible coloring of the graph uses at least $K^{\frac{1}{25}(\log K)}$ colors. We let $\mathcal{G}[c, f]$ be the family of graphs that either can be colored using c colors or have no independent set containing a fraction f of the vertices. The following theorem (not stated in this form in [Kho01]) summarizes the result obtained.

Theorem 5.1.5 ([Kho01]) *For all sufficiently large constants K, it is* NP*-hard to decide if a graph in $\mathcal{G}[K, 1/K^{\frac{1}{25}(\log K)}]$ can be colored using K colors or has no*

independent set containing a fraction $1/K^{\frac{1}{25}(\log K)}$ *of the vertices. Moreover this hardness result holds for graphs with degree at most* $2^{K^{O(\log K)}}$.

By using a stronger assumption, we can let K be a function of the number of vertices. Again, the stronger statement (not explicitly stated in [Kho01]) follows from the soundness analysis.

Theorem 5.1.6 ([Kho01]) *There exists an absolute constant* $\gamma > 0$ *such that for all* $K \leq 2^{(\log n)^\gamma}$*, there is no polynomial time algorithm that decides if an n-vertex graph in* $\mathcal{G}[K, 1/K^{\Omega(\log K)}]$ *can be colored using K colors or has no independent set containing a fraction* $1/K^{\Omega(\log K)}$ *of the vertices, unless* $NP \subseteq DTIME(2^{O(\log n)^{O(1)}})$.

Our reduction to $J2||C_{\max}$ uses the following result by Alimonti and Kann [AK00]. A cubic graph is a graph where all vertices have degree three.

Theorem 5.1.7 ([AK00]) *There exist positive constants* β, α *with* $\beta > \alpha$*, so that it is NP-hard to distinguish between n-vertex cubic graphs that have an independent set of size* $\beta \cdot n$ *and those that have no independent set of size* $\alpha \cdot n$.

5.2 Job and Flow Shop Instances with Large Makespan

We first exhibit a family of instances of general job shop scheduling for which it is relatively simple to show that any optimal schedule is of length $\Omega(\mathrm{lb} \cdot \log \mathrm{lb})$. This complements[2] and builds on the bound by Feige & Scheideler [FS02], who showed the existence of job shop instances with optimal makespan $\Omega(\mathrm{lb} \cdot \log \mathrm{lb}/\log\log \mathrm{lb})$. We then use this construction as a building block in the more complicated flow shop construction.

5.2.1 Job Shops with Large Makespan

Construction

For any integer $d \geq 1$ consider the job shop instance with d machines $m_1, \ldots, m_d$ and d jobs $j_1, \ldots, j_d$. We say that job j_i has *frequency* i, which means that it has 3^i so-called *long-operations* on machine m_i, each one of them requires 3^{d-i} time units. Between any two consecutive long-operations, job j_i has *short-operations* that require 0 time units on the machines $m_1, \ldots, m_{i-1}$. Note that the length of all jobs and the load on all machines are 3^d, which we denote by lb. See Figure 5.2 for an example of the construction.

[2] In their construction all operations of a job have the same length which is not the case for our construction.

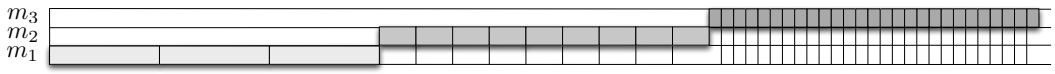


Figure 5.2: An example of the construction with $d = 3$. The jobs of frequency 1, 2 and 3 are depicted in light, medium, and dark gray, respectively.

Analysis

Fix an arbitrary feasible schedule of the jobs. We shall show that the length of the schedule must be $\Omega(\mathrm{lb} \cdot \log \mathrm{lb})$.

Lemma 5.2.1 *For $i, j : 1 \leq i < j \leq d$, the number of time units during which both j_i and j_j perform operations is at most $\frac{\mathrm{lb}}{3^{j-i}}$.*

Proof. During the execution of a long-operation of j_i (that requires 3^{d-i} time units), job j_j can complete at most one long-operation that requires 3^{d-j} time units (since its short-operation on machine m_i has to wait). As j_i has 3^i long-operations, the two jobs can perform operations at the same time during at most $3^i \cdot 3^{d-j} = \frac{3^d}{3^{j-i}} = \frac{\mathrm{lb}}{3^{j-i}}$ time units. □

It follows that, for each $i = 1, \ldots, d$, at most a fraction $1/3 + 1/3^2 + \cdots + 1/3^i \leq 1/3 + 1/3^2 + \cdots + 1/3^d \leq \frac{1}{3-1} = 1/2$ of the time spent for long-operations of a job j_i is performed at the same time as long-operations of jobs with lower frequency. Hence a feasible schedule has makespan at least $d \cdot \mathrm{lb}/2$. As $d = \Omega(\log \mathrm{lb})$ (recall that $\mathrm{lb} = 3^d$), the optimal makespan of the constructed job shop instance is $\Omega(\mathrm{lb} \cdot \log \mathrm{lb})$.

5.2.2 Flow Shops with Large Makespan

Construction

For sufficiently large integers d and r, consider the flow shop instance defined as follows (see also Figure 5.3 for an example of the construction):

- There are r^{2d} groups of machines[3], denoted by $M_1, M_2, \ldots, M_{r^{2d}}$. Each group M_g consists of d machines $m_{g,1}, m_{g,2}, \ldots, m_{g,d}$ (one for each frequency). Finally the machines are ordered in such a way that $m_{g,i}$ is before $m_{h,j}$ if either (i) $g < h$ or (ii) $g = h$ and $i > j$. The latter case will ensure that, within each group of machines, long-operations of jobs with high frequency will be scheduled before long-operations of jobs with low frequency, a fact that will be used in the proof of Lemma 5.2.3.

[3]These groups of machines "correspond" to copies of the job shop instance in subsection 5.2.1.

- For each frequency $f = 1, \ldots, d$, there are $r^{2(d-f)}$ groups of jobs, denoted by $J_1^f, J_2^f, \ldots, J_{r^{2(d-f)}}^f$. Each group J_g^f consists of r^{2f} copies, referred to as $j_{g,1}^f, j_{g,2}^f, \ldots, j_{g,r^{2f}}^f$, of the job that must be processed during $r^{2(d-f)}$ time units on the machines

$$m_{a+1,f}, m_{a+2,f}, \ldots, m_{a+r^{2f},f} \text{ where } a = (g-1) \cdot r^{2f}$$

  and during 0 time units on all the other machines that are required to create a flow shop instance. Let J^f be the set of jobs that correspond to frequency f, i.e., $J^f = \{j_{g,a}^f : 1 \leq g \leq r^{2(d-f)}, 1 \leq a \leq r^{2f}\}$.

Note that the length of all jobs and the load on all machines are r^{2d}, which equals lb. Moreover, the total number of machines and the total number of jobs are both $r^{2d} \cdot d$. In the subsequent we will call the operations that require more than 0 time units *long-operations* and the operations that only require 0 time units *short-operations*.

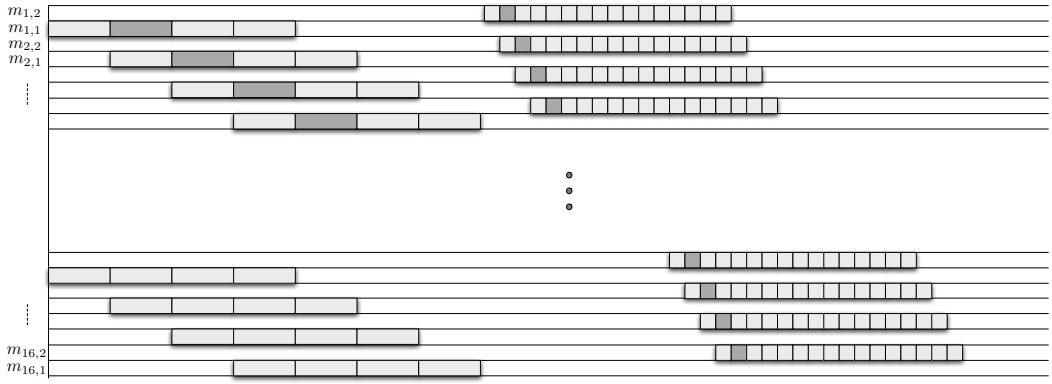


Figure 5.3: An example of the construction for flow shop scheduling with $r = d = 2$. Only long-operations on the first 4 and last 4 groups of machines are depicted. The long-operations of one job of each frequency are highlighted in dark gray.

Analysis

We shall show that the length of any feasible schedule must be $\Omega(\text{lb} \cdot \min[r, d])$. As $\text{lb} = r^{2d}$, instances constructed with $r = d$ have optimal makespan $\Omega(\text{lb} \cdot \log \text{lb} / \log \log \text{lb})$.

Fix an arbitrarily feasible schedule for the jobs. We start by showing a useful property. For a job j, let $d_j(i)$ denote the delay between its i-th and $(i+1)$-th

long-operations, i.e., the time units between the end of job j's i-th long-operation and the start of its $(i+1)$-th long-operation (let $d_j(i) = \infty$ for the last long-operation). We say that the i-th long-operation of a job j of frequency f is *good* if $d_j(i) \leq \frac{r^2}{4} \cdot r^{2(d-f)}$.

Lemma 5.2.2 *If the schedule has makespan less than $r \cdot \text{lb}$ then the fraction of good long-operations of each job is at least $\left(1 - \frac{4}{r}\right)$.*

Proof. Assume that the considered schedule has makespan less than $r \cdot \text{lb}$. Suppose toward contradiction that there exists a job j of frequency f so that j has at least $\frac{4}{r} r^{2f}$ long-operations that are not good. But then the length of j is at least $\frac{4}{r} r^{2f} \cdot \frac{r^2}{4} \cdot r^{2(d-f)} = r \cdot r^{2d} = r \cdot \text{lb}$, which contradicts that the makespan of the considered schedule is less than $r \cdot \text{lb}$. □

We continue by analyzing the schedule with the assumption that its makespan is less than $r \cdot \text{lb}$ (otherwise we are done). In each group M_g of machines we will associate a set $T_{g,f}$ of time intervals with each frequency $f = 1, \ldots, d$. The set $T_{g,f}$ contains the time intervals corresponding to the *first half* of all good long-operations scheduled on the machine $m_{g,f}$. For intuition of the following lemma see Figure 5.4.

Lemma 5.2.3 *Let $k, \ell : 1 \leq k < \ell \leq d$ be two different frequencies. Then the sets $T_{g,k}$ and $T_{g,\ell}$, for all $g : 1 \leq g \leq r^{2d}$, contain disjoint time intervals.*

Proof. Suppose toward contradiction that there exist time intervals $t_k \in T_{g,k}$ and $t_\ell \in T_{g,\ell}$ that overlap, i.e., $t_k \cap t_\ell \neq \emptyset$. Note that t_k and t_ℓ correspond to good long-operations of jobs of frequencies k and ℓ, respectively. Let us say that the good long-operation corresponding to t_ℓ is the a-th operation of some job j. Note that, since j has a long-operation on machine $m_{g,\ell}$, job j has frequency ℓ. As t_ℓ and t_k overlap, the a-th long-operation of j must overlap the first half of the long-operation corresponding to t_k. As job j has a short operation on machine $m_{g,k}$ after its long-operation on machine $m_{g,\ell}$ (recall that machines are ordered $m_{g,d}, m_{g,d-1}, \ldots, m_{g,1}$ and $\ell > k$), job j's $(a+1)$-th operation must be delayed by at least $r^{2(d-k)}/2 - r^{2(d-\ell)}$ time units and thus $d_j(a) > r^{2(d-k)}/2 - r^{2(d-\ell)} > \frac{r^2}{4} r^{2(d-\ell)}$ (using $\ell > k$), which contradicts that the a-th long-operation of job j is good. □

Let $L(T_{g,f})$ denote the total time units covered by the time intervals in $T_{g,f}$. We continue by showing that there exists a g such that $\sum_{f=1}^{d} L(T_{g,f}) \geq \frac{\text{lb}}{4} \cdot d$. With this in place, it is easy to see that any schedule has makespan $\Omega(d \cdot \text{lb})$ since all the time intervals $\{T_{g,f} : f = 1, \ldots, d\}$ are disjoint (Lemma 5.2.3).

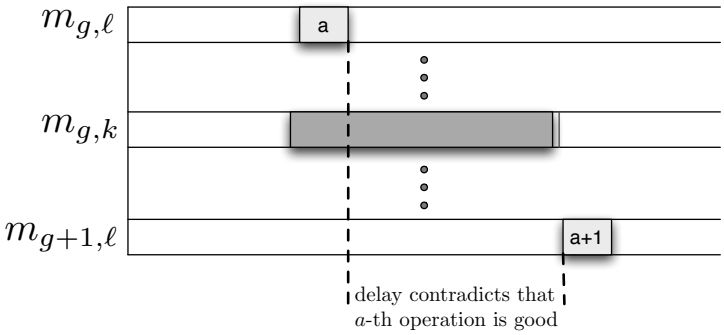


Figure 5.4: The intuition behind the proof of Lemma 5.2.3.

Lemma 5.2.4 *There exists a* $g \in \{1, \ldots, r^{2d}\}$ *such that*

$$\sum_{f=1}^{d} L(T_{g,f}) \geq \frac{\text{lb}}{4} \cdot d$$

Proof. As $\sum_{f=1}^{d} L(T_{g,f})$ adds up the time units required by the *first half* of each good long-operation scheduled on a machine in M_g, the claim follows by showing that there exists a group of machines M_g from $\{M_1, M_2, \ldots, M_{r^{2d}}\}$ so that the total time units required by the good long-operations on the machines in M_g is at least $\frac{\text{lb} \cdot d}{2}$.

By lemma 5.2.2 we have that the good long-operations of each job require at least $\text{lb} \cdot \left(1 - \frac{4}{r}\right)$ time units. Since the total number of jobs is $r^{2d}d$ the total time units required by all good long-operations is at least $\text{lb} \cdot \left(1 - \frac{4}{r}\right) \cdot r^{2d}d$. As there are r^{2d} many groups of machines, a simple averaging argument guarantees that in at least one group of machines, say M_g, the total time units required by the good long-operations on the machines in M_g is at least $\text{lb} \cdot \left(1 - \frac{4}{r}\right) d > \text{lb} \cdot d/2$ for a sufficiently large r. □

5.3 Hardness of Job Shops and Generalized Flow Shops

Theorem 5.1.2 and Theorem 5.1.3 are proved by presenting a gap-preserving reduction, Γ, from the graph coloring problem to the generalized flow shop problem, that has two parameters, r and d. Given an n-vertex graph G whose vertices are partitioned into d independent sets, it computes in time polynomial in n and r^d, a generalized flow shop instance $S(r, d)$ where all jobs have the same length r^{2d} and all machines the same load r^{2d}. Hence, $\text{lb} = r^{2d}$. Instance $S(r, d)$

has a set of r^{2d} jobs and a set of r^{2d} machines for each vertex in G. The total number of jobs and the total number of machines are thus both $r^{2d}n$. Moreover, each job has at most $(\Delta+1)r^{2d}$ operations.

By using jobs of different frequencies, as done in the gap construction, we have the property that "many" of the jobs corresponding to adjacent vertices cannot be scheduled in parallel in any feasible schedule. On the other hand, by letting jobs skip those machines corresponding to non-adjacent vertices, jobs corresponding to an independent set in G *can* be scheduled in parallel (their operations can overlap in time) in a feasible schedule. This ensures that the following completeness and soundness hold for the resulting generalized flow shop instance $S(r,d)$.

- *(Completeness)* If G can be colored using L colors then $C^*_{max} \leq \text{lb} \cdot 2L$.
- *(Soundness)* For any $L \leq r$. Given a schedule where at least half the jobs finish within $\text{lb} \cdot L$ time units, we can find an independent set of G of size $n/(8L)$, in time polynomial in n and r^d.

In Section 5.3.1, we present a reduction with somewhat stronger properties for the *job shop* problem. As the reduction is relatively simple, it serves as a good starting point before reading the similar but more complex reduction Γ for the *generalized flow shop* problem. Before continuing, let us see how the reduction Γ, with the aforementioned properties, is sufficient for proving Theorem 5.1.2 and Theorem 5.1.3.

Proof of Theorem 5.1.2

By Theorem 5.1.5, for sufficiently large K and $\Delta = 2^{K^{O(\log K)}}$, it is NP-hard to decide if an n-vertex graph G in $\mathcal{G}[K, 1/K^{\frac{1}{25}(\log K)}]$ with bounded degree Δ has

$$\chi(G) \leq K \quad \text{or} \quad \alpha(G) \leq \frac{n}{K^{\frac{1}{25}(\log K)}}.$$

As the vertices of a graph with bounded degree Δ can, in polynomial time, be partitioned into $\Delta+1$ independent sets, we can use Γ with parameters $d = \Delta+1$ and $r = K^{\frac{1}{25}(\log K)}$ (r is chosen such that the condition $L \leq r$ in the soundness case of Γ is satisfied for $L = K^{\frac{1}{25}(\log K)}/8$). It follows by the completeness case and soundness case of Γ that it is NP-hard to distinguish if the obtained scheduling instance has a schedule with makespan at most $\text{lb} \cdot 2K$ or no solution schedules more than half of the jobs within $\text{lb} \cdot K^{\frac{1}{25}(\log K)}/8$ time units. Moreover, each job has at most $(\Delta+1)r^{2d}$ operations, which is a constant that only depends on K.

Proof of Theorem 5.1.3

The proof is similar to the proof of Theorem 5.1.2 with the exception that the graphs have no longer bounded degree. To this end the lemma below will be useful. When using probabilistic arguments for graphs with n vertices, we shall use the term *overwhelming* (*negligible*, respectively) to denote probability that tends to 1 (to 0, respectively) as n tends to infinity.

Lemma 5.3.1 *For any constant $\delta \geq 1$, given an n-vertex graph $G = (V, E)$, we can construct in randomized polynomial time a subgraph $G' = (V, E')$ of G with $E' \subseteq E$ such that*

1. *The vertices are partitioned into $(\log n)^\delta$ sets, each set forms an independent set in G'.*
2. *$\chi(G') \leq \chi(G)$.*
3. *With overwhelming probability the following holds: given an independent set of G', with $\frac{n}{(\log n)^{\delta-1}}$ vertices, we can find an independent set of G with $\frac{n}{(\log n)^\delta}$ vertices, in polynomial time.*

Proof. Given an n-vertex graph $G = (V, E)$, we give a probabilistic construction of $G' = (V, E')$. Each vertex $v \in V$ is assigned independently, uniformly at random to one of the sets $I_1, I_2, \ldots, I_{(\log n)^\delta}$. Let $E' \subseteq E$ be those edges that are incident to vertices placed in different sets, i.e., an edge is deleted from E to yield E' if and only if it is adjacent to two vertices $u \in I_i$ and $v \in I_i$ for some $i : 1 \leq i \leq (\log n)^\delta$.

The graph G' obviously satisfies the first two properties in the lemma. We continue by showing that G' satisfies property 3. In fact, we show that the following stronger property holds with overwhelming probability: any independent set I' of G', with $|I'| = \frac{n}{(\log n)^{\delta-1}}$, induces a subgraph of G with at least $\frac{n}{(\log n)^\delta}$ maximal connected components. This is done by proving that if a subset V' of $n/(\log n)^{\delta-1}$ vertices induces a subgraph of G with less than $\frac{n}{(\log n)^\delta}$ maximal connected components, then the probability that V' is an independent set of G' is negligible.

Fix a set $V' \subseteq V$ of $n/(\log n)^{\delta-1}$ vertices and let H be the subgraph of G induced by V'. Assuming that H can be partitioned into s maximal connected components, with $s < \frac{n}{(\log n)^\delta}$, we calculate the probability that V' forms an independent set in G'. Let $H_1, H_2, \ldots, H_s$ denote the maximal connected components of H. We use $|H_\ell|$, for $\ell : 1 \leq \ell \leq s$, to denote the number of vertices of H_ℓ. If the vertices of H form an independent set in G' then all vertices of a connected component must be placed in the same set I_i, for some $i : 1 \leq i \leq (\log n)^\delta$. The probability that this happens, for a connected component with k vertices, is at

most $\left(\frac{1}{\log n}\right)^{\delta(k-1)}$. As the different maximal connected components are independent, the probability that V' forms an independent set in G' is at most

$$\left(\frac{1}{\log n}\right)^{\delta(|H_1|-1)}\left(\frac{1}{\log n}\right)^{\delta(|H_2|-1)}\cdots\left(\frac{1}{\log n}\right)^{\delta(|H_s|-1)}=\left(\frac{1}{\log n}\right)^{\delta\left(\sum_{i=1}^{s}|H_i|-s\right)}.$$

As $\sum_{i=1}^{s}|H_i|=|V'|=n/(\log n)^{\delta-1}$ and $s<\frac{n}{(\log n)^{\delta}}$, the probability that V' forms an independent set in G' is at most

$$\left(\frac{1}{\log n}\right)^{\delta\cdot n/(\log n)^{\delta-1}\cdot(1-1/\log n)}.$$

The number of ways to fix the set V' is at most

$$\binom{n}{n/(\log n)^{\delta-1}}\leq\left(e\cdot\log n\right)^{(\delta-1)n/(\log n)^{\delta-1}}.$$

Hence, the union bound implies that the probability that graph G' fails to satisfy property 3 is at most

$$\left(\frac{1}{\log n}\right)^{\delta\cdot n/(\log n)^{\delta-1}\cdot(1-1/\log n)}\cdot\left(e\cdot\log n\right)^{(\delta-1)n/(\log n)^{\delta-1}}$$

which tends to 0 as n tends to infinity.

□

Assuming $NP\not\subseteq DTIME(2^{O(\log n)^{O(1)}})$, Theorem 5.1.6 with $K=\log n$ says that there is no polynomial algorithm that decides if an n-vertex graph G in $\mathcal{G}[\log n,1/(\log n)^{\Omega(\log\log n)}]$ has

$$\chi(G)\leq\log n\quad\text{or}\quad\alpha(G)\leq\frac{n}{(\log n)^{\Omega(\log\log n)}}.$$

Given an n-vertex graph G in $\mathcal{G}[\log n,1/(\log n)^{\Omega(\log\log n)}]$, we construct graph G' from G by applying Lemma 5.3.1 with $\delta=3/\epsilon$, where $\epsilon>0$ is an arbitrarily small constant. We then obtain a generalized flow shop instance S from G' by using Γ with parameters $r=(\log n)^{\delta-1}$ and $d=(\log n)^{\delta}$. The size of S is $O(r^{2d}n\cdot\Delta r^{2d})=O(2^{O(\log n)^{O(1/\epsilon)}})$ and $\mathrm{lb}=r^{2d}=(\log n)^{2(\delta-1)(\log n)^{\delta}}$ and hence $\log\mathrm{lb}\leq(\log n)^{\delta+1}$ (for large enough n).

The analysis is straightforward:

- *(Completeness)* If $\chi(G)\leq\log n$ then $\chi(G')\leq\log n$ and, by the completeness case of Γ, there is a schedule of S with makespan $\mathrm{lb}\cdot 2\log n$.

- *(Soundness)* Assuming that the probabilistic construction of G' succeeded, we have $\alpha(G) \leq \frac{n}{n^{\Omega(\log\log n)}} \Rightarrow \alpha(G') \leq \frac{n}{(\log n)^{\delta-1}}$, which in turn implies by the soundness case of Γ that no solution schedules more than half the jobs within $\mathrm{lb} \cdot (\log n)^{\delta-1}/8$ time units (recall that r was chosen to be greater than $(\log n)^{\delta-1}/8$).

The probabilistic construction of G' succeeds with overwhelming probability. Furthermore, given a schedule, we can detect such a failure in polynomial time and repeat the reduction. It follows that an approximation algorithm for $F|jumps|C_{max}$ with performance guarantee $\frac{(\log n)^{\delta-1}/8}{2\log n} = (\log n)^{\delta-2}/16$ or an approximation algorithm for $F|jumps|\sum C_j$ with performance guarantee $\frac{(n/2)\cdot(\log n)^{\delta-1}/8}{2n\cdot\log n} = (\log n)^{\delta-2}/32$ would imply that $NP \subseteq ZTIME(2^{O(\log n)^{O(1/\epsilon)}})$. Finally we note that δ was chosen so that for large enough n we have

$$(\log \mathrm{lb})^{1-\epsilon} \leq (\log n)^{(1-\epsilon)(\delta+1)} = (\log n)^{3/\epsilon+1-3-\epsilon} =$$

$$(\log n)^{\delta-2-\epsilon} < 1/32 \cdot (\log n)^{\delta-2}.$$

5.3.1 Job Shops

In this section we give and analyze a somewhat stronger reduction than Γ for the *general* job shop problem. The number of operations per job is at most $(\Delta+1)r^d$, the number of jobs and the number of machines are n, and the soundness case says that, given a schedule with makespan $\mathrm{lb} \cdot L$, we can, in time polynomial in n and r^d, find an independent set of G of size $(1 - \frac{\Delta}{r})n/L$.[4] As the reduction is relatively simple, it serves as a good starting point for reading the more complex reduction to the generalized flow shop problem.

Construction

Given an n-vertex graph $G = (V, E)$ whose vertices are partitioned into d independent sets, we create a job shop instance $S(r, d)$, where r and d are the parameters of the reduction. Instance $S(r, d)$ has a machine m_v and a job j_v for each vertex $v \in V$. We continue by describing the operations of the jobs. Let $I_1, I_2, \ldots I_d$ denote the independent sets that form a partition of V. A job j_v that corresponds to a vertex $v \in I_i$, for some $i : 1 \leq i \leq d$, has a chain of r^i *long-operations* $O_{1,j_v}, O_{2,j_v}, \ldots, O_{r^i,j_v}$, each of them requiring r^{d-i} time units, that

[4]The condition that r needs to be greater than Δ can be removed as done in the analysis of generalized flow shops. In this section we have chosen to provide a simpler analysis, which still describes most of the ideas used in the flow shop analysis.

must be processed on the machine m_v. Between two consecutive long-operations O_{p,j_v}, O_{p+1,j_v}, for $p : 1 \leq p < r^i$, we have a set of *short-operations* placed on the machines $\{m_u : \{u,v\} \in E\}$ (the machines corresponding to adjacent vertices) in some order. A short-operation requires time 0. For an example of the construction see Figure 5.5.

Remark. The construction has n machines and n jobs. Each job has length r^d and each machine has load r^d. Hence, $\text{lb} = r^d$. Moreover, the number of operations per job is at most $(\Delta + 1)r^d$. □

Figure 5.5: An example of the reduction with $r = 4, d = 2, I_1 = \{A\}$, and $I_2 = \{B, C\}$. Note that jobs only have short-operations on machines corresponding to adjacent vertices. (The jobs corresponding to A, B, and C are depicted to the left, center, and right respectively.)

Completeness

We prove that if the graph G can be colored with L colors then there is a "relatively short" solution to the job shop instance (see Figure 5.6).

Lemma 5.3.2 *If $\chi(G) = L$ then there is a schedule of $S(r,d)$ with makespan* $\text{lb} \cdot L$.

Proof. Let $V_1, V_2, \ldots, V_L$ be a partition of V into L independent sets. Consider one of these sets, say V_i. As the vertices of V_i form an independent set, no short-operations of the jobs $\{j_v\}_{v \in V_i}$, are scheduled on the machines $\{m_v\}_{v \in V_i}$. Since short-operations require time 0 we can schedule the jobs $\{j_v\}_{v \in V_i}$ within lb time units. We can thus schedule the jobs in L-"blocks" in the order $\{j_v\}_{v \in V_1}, \{j_v\}_{v \in V_2}$, $\ldots, \{j_v\}_{v \in V_L}$. The total length of this schedule is $\text{lb} \cdot L$. □

Soundness

We prove that, given a schedule where many jobs are completed "early", we can, in polynomial time, find a "big" independent set of G.

Lemma 5.3.3 *Given a schedule of $S(r,d)$ where at least half the jobs finish within* $\text{lb} \cdot L$ *time units, we can, in time polynomial in n and r^d, find an independent set of G of size at least* $(1 - \frac{\Delta}{r})n/(2L)$.

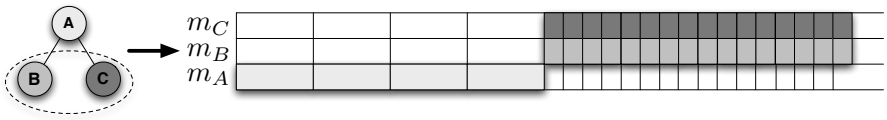


Figure 5.6: An example of how the jobs are scheduled in the completeness case. Here $V_1 = \{A\}$ and $V_2 = \{B, C\}$.

Proof. First we show that two jobs corresponding to adjacent vertices cannot be scheduled in parallel. (The proof of the following claim is similar to the proof of Lemma 5.2.1 in the gap construction).

Claim 5.3.4 *Let $u \in I_i$ and $v \in I_j$ be two adjacent vertices in G with $i < j$. Then at most a fraction $\frac{1}{r}$ of the long-operations of j_v can overlap the long-operations of j_u.*

Proof of Claim. There are r^i and r^j long-operations of j_u and j_v, respectively. As the vertices u and v are adjacent, job j_v has a small-operation on machine m_u between any two long-operations. Hence, at most one long-operation of j_v can be scheduled in parallel with a long-operation of j_u and the total number of such operations in any schedule is at most $r^i \leq \frac{r^j}{r}$ (using $i < j$). □

Now consider a schedule where at least half the jobs finish within $\text{lb} \cdot L$ time units. For each $i : 1 < i \leq d$ and for each $v \in I_i$, we disregard those long-operations of job j_v that overlap long-operations of the jobs $\{j_u : \{u, v\} \in E \text{ and } u \in I_j \text{ for some } j < i\}$. After disregarding operations, no two long-operations corresponding to adjacent vertices will overlap in time. Furthermore, by applying Claim 5.3.4 and using that the maximum degree of G is Δ, we know that at most a fraction $\frac{\Delta}{r}$ of a job's long-operations have been disregarded. Thus the remaining long-operations of a job require at least $(1 - \frac{\Delta}{r}) \cdot \text{lb}$ time units. As at least half the jobs ($n/2$ many) finish within $L \cdot \text{lb}$ time units, we have that at least $(1 - \frac{\Delta}{r}) \cdot \text{lb} \cdot n/2$ time units are scheduled on the machines within $L \cdot \text{lb}$ time units. By a simple averaging argument we have that at least $(1 - \frac{\Delta}{r})n/(2L)$ of the remaining long-operations must overlap at some point within the first $L \cdot \text{lb}$ time units in the schedule. As the remaining long-operations that overlap correspond to different vertices that are non-adjacent, the graph has an independent set of size $(1 - \frac{\Delta}{r})n/(2L)$. Moreover, we can find such a point (corresponding to an independent set) in the schedule , e.g. by considering the start and end points of all long-operations that were not disregarded. □

5.3.2 Generalized Flow Shops

Here, we present the reduction Γ for the general flow shop problem where jobs are allowed to skip machines. The idea is similar to the reduction presented in Section 5.3.1 for the job shop problem. The main difference is to ensure, without using cyclic jobs, that jobs corresponding to adjacent vertices cannot be scheduled in parallel.

Construction

Given an n-vertex graph $G = (V, E)$ whose vertices are partitioned into d independent sets, we create a generalized flow shop instance $S(r, d)$, where r and d are the parameters of the reduction. Let $I_1, I_2, \ldots I_d$ denote the independent sets that form a partition of V.

The instance $S(r, d)$ is very similar to the gap instance described in Section 5.2.2. The main difference is that in $S(r, d)$ distinct jobs can be scheduled in parallel if their corresponding vertices in G are not adjacent. This is obtained by letting a job skip those machines corresponding to non-adjacent vertices. (The gap instance of Section 5.2.2 can be seen as the result of the following reduction when the graph G is a complete graph with d nodes). For convenience, we give the complete description with the necessary changes.

- There are r^{2d} groups of machines, denoted by $M_1, M_2, \ldots, M_{r^{2d}}$. Each group M_g consists of n machines $\{m_{g,v} : v \in V\}$ (one for each vertex in G). Finally the machines are ordered in such a way that $m_{g,u}$ is before $m_{h,v}$ if either (i) $g < h$ or (ii) $g = h$ and $u \in I_k, v \in I_\ell$ with $k > \ell$. The latter case will ensure that, within each group of machines, long-operations of jobs with high frequency will be scheduled before long-operations of jobs with low frequency, a fact that is used to prove Lemma 5.3.9.

- For each $f : 1 \leq f \leq d$ and for each vertex $v \in I_f$ there are $r^{2(d-f)}$ groups of jobs, denoted by $J^v_1, J^v_2, \ldots, J^v_{r^{2(d-f)}}$. Each group J^v_g consists of r^{2f} copies, referred to as $j^v_{g,1}, j^v_{g,2}, \ldots, j^v_{g,r^{2f}}$, of the job that must be processed during $r^{2(d-f)}$ time units on the machines

  $$m_{a+1,v}, m_{a+2,v}, \ldots, m_{a+r^{2f},v} \text{ where } a = (g-1) \cdot r^{2f}$$

  and during 0 time units on machines corresponding to adjacent vertices, i.e., $\{m_{a,u} : 1 \leq a \leq r^{2d}, \{u, v\} \in E\}$ (in an order so that it results in a generalized flow shop instance).

  Let J^v be the jobs that correspond to the vertex v, i.e., $J^v = \{j^v_{g,i} : 1 \leq g \leq r^{2(d-f)}, 1 \leq i \leq r^{2f}\}$.

Remark. The construction has $r^{2d}n$ machines and $r^{2d}n$ jobs. Each job has length r^{2d} and each machine has load r^{2d}. Hence, lb $= r^{2d}$. Moreover, the number of operations per job is at most $(\Delta+1)r^{2d}$. □

In the subsequent we will call the operations that require more than 0 time units *long-operations* and the operations that only require 0 time units *short-operations*. For an example of the construction see Figure 5.7.

Figure 5.7: An example of the reduction with $r = 2, d = 2, I_1 = \{A\}$ and $I_2 = \{B, C\}$. Only the first two out of $r^{2d} = 16$ groups of machines are depicted with the jobs corresponding to *A*,*B*, and *C* to the left, center, and right respectively.

Completeness

We prove that if the graph G can be colored with L colors then there is a relatively "short" solution to the general flow shop instance.

Lemma 5.3.5 *If $\chi(G) = L$ then there is a schedule of $S(r,d)$ with makespan* lb·2L.

Proof. We start by showing that all jobs corresponding to non-adjacent vertices can be scheduled within $2 \cdot$ lb time units.

Claim 5.3.6 *Let IS be an independent set of G. Then all the jobs $\bigcup_{v\in IS} J^v$ can be scheduled within* $2 \cdot$ lb *time units.*

Proof of Claim. Consider the schedule defined by scheduling the jobs corresponding to each vertex $v \in IS$ as follows. Let I_f be the independent set with $v \in I_f$. A job $j^v_{g,i}$ corresponding to vertex v is then scheduled without interruption starting at time $r^{2(d-f)} \cdot (i-1)$.

The schedule has makespan at most $2 \cdot$ lb since a job is started at latest at time $r^{2(d-f)} \cdot (r^{2f} - 1) <$ lb and requires lb time units in total.

To see that the schedule is feasible, observe that no short-operations of the jobs in $\bigcup_{v\in IS} J^v$ need to be processed on the same machines as the long-operations of the jobs in $\bigcup_{v\in IS} J^v$ (this follows from the construction and from the fact that the jobs correspond to non-adjacent vertices). Moreover, two jobs $j^v_{g,i}, j^{v'}_{g',i'}$ with either $g \neq g'$ or $v \neq v'$ have no two long-operations that must be

processed on the same machine. Hence, the only jobs that might delay each other are jobs belonging to the same vertex v and the same group g, but these jobs are started with appropriate delays (depending on the frequency of the job). □

Assuming $\chi(G) = L$ we partition V into L independent sets $V_1, V_2, \ldots, V_L$. By the above claim, the jobs corresponding to each of these independent sets can be scheduled within $2 \cdot \text{lb}$ time units. We can thus schedule the jobs in L-"blocks", one block of length $2 \cdot \text{lb}$ for each independent set. The total length of this schedule is $\text{lb} \cdot 2L$. □

Soundness

We prove that, given a schedule where many jobs are completed "early", we can find a "big" independent set of G, in polynomial time.

Lemma 5.3.7 *For any $L \leq r$, given a schedule of $S(r,d)$ where at least half the jobs finish within $\text{lb} \cdot L$ time units, we can, in time polynomial in n and r^d, find an independent set of G of size at least $n/(8L)$.*

Proof. Fix an arbitrarily schedule of $S(r,d)$ where at least half the jobs finish within $\text{lb} \cdot L$ time units. In the subsequent we will disregard the jobs that do not finish within $\text{lb} \cdot L$ time units. Note that the remaining jobs are at least $r^{2d}n/2$ many. As for the gap construction (see Section 5.2.2), we say that the i-th long-operation of a job j of frequency f is *good* if the delay $d_j(i)$ between job j's i-th and $(i+1)$-th long-operations is at most $\frac{r^2}{4} \cdot r^{2(d-f)}$. In each group M_g of machines we will associate a set $T_{g,v}$ of time intervals with each vertex $v \in V$. The set $T_{g,v}$ contains the time intervals corresponding to the *first half* of all good long-operations scheduled on the machine $m_{g,v}$. We also let $L(T_{g,v})$ denote the total time units covered by the time intervals in $T_{g,v}$. Scheduling instance $S(r,d)$ has similar structure and similar properties as the gap instances created in Section 5.2.2. By using the fact that all jobs (that were not disregarded) have completion time at most $L \cdot \text{lb}$, which is by assumption at most $r \cdot \text{lb}$, Lemma 5.3.8 follows from the same arguments as Lemma 5.2.2.

Lemma 5.3.8 *The fraction of good long-operations of each job is at least $\left(1 - \frac{4}{r}\right)$.*

Consider a group M_g of machines and two jobs corresponding to adjacent vertices that have long-operations on machines in M_g. Recall that jobs corresponding to adjacent vertices have different frequencies. By the ordering of the machines, we are guaranteed that the job of higher frequency has, after its long-operation on a machine in M_g, a short-operation on the machine in M_g where

the job of lower frequency has its long-operation. The following lemma now follows by observing, as in the proof of Lemma 5.2.3, that the long-operation of the high frequency job can only be good if it is *not* scheduled in parallel with the first half of the long-operation of the low frequency job.

Lemma 5.3.9 *Let $u \in I_k$ and $v \in I_l$ be two adjacent vertices in G with $k > l$. Then the sets $T_{g,u}$ and $T_{g,v}$, for all $g : 1 \leq g \leq r^{2d}$, contain disjoint time intervals.*

Finally, Lemma 5.3.10 is proved in the very same way as Lemma 5.2.4. Their different inequalities arise because in the gap instance we had $d \cdot r^{2d}$ jobs and here we are considering at least $r^{2d}n/2$ jobs that were not disregarded.

Lemma 5.3.10 *There exists a $g \in \{1, \ldots, r^{2d}\}$ such that*

$$\sum_{v \in V} L(T_{g,v}) \geq \frac{\mathrm{lb} \cdot n}{8}.$$

We conclude by a simple averaging argument. Consider a g, such that $\sum_{v \in V} L(T_{g,v})$ is at least $\frac{\mathrm{lb} \cdot n}{8}$. This is guaranteed to exist by the lemma above. As all jobs that were not disregarded finish within $L \cdot \mathrm{lb}$ time units, at least $\frac{\mathrm{lb} \cdot n}{8}/(L \cdot \mathrm{lb}) = \frac{n}{8L}$ time intervals must overlap at some point during the first $L \cdot \mathrm{lb}$ time units of the schedule, and, since they overlap, they correspond to different vertices that form an independent set in G (Lemma 5.3.9). Moreover, we can find such a point in the schedule, for example, by considering all different blocks and in each block verify the start and end points of the time intervals. □

5.4 Hardness of Job Shops with Two Machines

In this section we prove Theorem 5.1.4. We show that $J2||C_{\max}$ has no PTAS by presenting a gap-preserving reduction from the NP-hard problem to distinguish between n-vertex cubic graphs that have an independent set of size $\beta \cdot n$ and those that have no independent set of size $\alpha \cdot n$, for some $\beta > \alpha$ (see Theorem 5.1.7). More specifically, given a cubic graph $G(V, E)$, we construct an instance S of $J2||C_{max}$ so that, for some L defined later, we have the following completeness and soundness analyses.

- *(Completeness)* If G has an independent set of size βn then S has a schedule with makespan L.
- *(Soundness)* If G has no independent set of size αn then all schedules of S have makespan at least $(1 + \Omega(1))L$.

Throughout this section we will use the following notation to define jobs (see Figure 5.8 for an example). An operation is defined by a pair $[m_i, p]$, where p is the processing time required on machine m_i. Let $s_1, \ldots, s_y$ be sequences of operations, and let $(s_1, \ldots, s_y)$ stand for the sequence resulting by their concatenation in the given order. We use $(s_1, \ldots, s_y)^x$ to denote the sequence obtained by repeating $(s_1, \ldots, s_y)$ x times.

Figure 5.8: An example of the representation. The light gray job is defined by $([m_1, 2], [m_2, 2])$ and the dark gray job is defined by $([m_2, 1], [m_1, 1])^2$.

Before presenting the reduction (Section 5.4.1) and the analysis (Section 5.4.2), we have the following lemma (whose standard proof is included for the sake of completeness), which will be useful in our construction.

Lemma 5.4.1 *For any sufficiently small fixed $\epsilon > 0$, we can, in time polynomial in n, construct a family of sets $\mathscr{C} = \{C_1, C_2, \ldots, C_{n^2}\}$ with the following properties:*

1. *Each set $C_i \in \mathscr{C}$ is a subset of $\{1, 2, \ldots, (1/\epsilon)^{1/\epsilon} \log n\}$ and has size $\log n$.*
2. *Two sets $C_i \in \mathscr{C}$ and $C_j \in \mathscr{C}$, with $i \neq j$, satisfy $|C_i \cap C_j| \leq \epsilon \log n$.*

Proof. Consider the following procedure to obtain such a family $\mathscr{C}$.

1: Initiate S with all binary strings of length $(1/\epsilon)^{1/\epsilon} \log n$ with $\log n$ many 1's
2: Let $\mathscr{C} = \emptyset$
3: **repeat**
4: Pick a binary string $x \in S$, and add the set $\{i : x_i = 1\}$ to $\mathscr{C}$
5: Remove all the binary strings from S that share at least $\epsilon \log n$ many 1's (elements) with x
6: **until** S is empty

It is clear that the family $\mathscr{C}$ returned by the above procedure satisfies properties (1) and (2). We continue by analyzing the size of the returned $\mathscr{C}$. We will use that $\binom{a}{b}$ is bounded from above by both $\binom{a}{\lceil a/2 \rceil}$ and $\left(\frac{a \cdot e}{b}\right)^b$. For simplicity

we assume all numbers to be integral. At each iteration, the number of sets of S that we remove is at most

$$\sum_{i=\epsilon \log n}^{\log n} \underbrace{\binom{\log n}{i}}_{\#\text{choices for 1's}} \cdot \underbrace{\binom{((1/\epsilon)^{1/\epsilon}-1)\log n}{\log n - i}}_{\#\text{choices for 0's}} \leq$$

$$(1-\epsilon)\log n \binom{\log n}{\frac{\log n}{2}} \cdot \binom{(1/\epsilon)^{1/\epsilon}\cdot \log n}{(1-\epsilon)\log n} \leq$$

$$(1-\epsilon)\log n \left((2e)^{\frac{\log n}{2}} \cdot \left((1/\epsilon)^{1/\epsilon} \cdot \frac{e}{1-\epsilon}\right)^{(1-\epsilon)\log n}\right) \leq$$

$$(1-\epsilon)\log n \left(4e^2 \cdot (1/\epsilon)^{1/\epsilon}\right)^{(1-\epsilon)\log n}.$$

As the number of elements in S at the beginning is $\binom{(1/\epsilon)^{1/\epsilon}\log n}{\log n} \geq (1/\epsilon)^{1/\epsilon \cdot \log n}$, the number of iterations and thus the size of C at the end of the process is at least

$$\frac{(1/\epsilon)^{1/\epsilon \cdot \log n}}{(1-\epsilon)\log n \left(4e^2 \cdot (1/\epsilon)^{1/\epsilon}\right)^{(1-\epsilon)\log n}} =$$

$$\frac{1}{(1-\epsilon)\log n \left((4e^2)^{1-\epsilon}\right)^{\log n}} \cdot \frac{(1/\epsilon)^{1/\epsilon \cdot \log n}}{(1/\epsilon)^{1/\epsilon \log n(1-\epsilon)}} =$$

$$\frac{(1/\epsilon)^{\log n}}{(1-\epsilon)\log n \left((4e^2)^{1-\epsilon}\right)^{\log n}},$$

which is greater than n^2 for sufficiently small ϵ.

□

5.4.1 Construction

The construction will be presented together with some useful properties that will later be used in the analysis. Before defining the jobs we will define "types" and "blocks" of operations. The jobs will later be defined as concatenations of blocks, which in turn will be defined as concatenations of types.

Types

Let $d = O(\log n)$. For each frequency $f : 1 \leq f \leq d$ we define type T_f and type $\bar{T}_f$ as

$$T_f := \left([m_1, n^{4(d-f)}], [m_2, 0]\right)^{n^{4f}}$$
$$\bar{T}_f := \left([m_2, n^{4(d-f)}], [m_1, 0]\right)^{n^{4f}}.$$

We will call the operations of T_f and $\bar{T}_f$ that require $n^{4(d-f)}$ time units *long-operations* and the operations that require 0 time units *short-operations*. Note that a type requires time $n^{4f} n^{4(d-f)} = n^{4d}$. We say that the two types T_f and $\bar{T}_f$ are *compatible*, for $f : 1 \leq f \leq d$. Note that two compatible types can be scheduled in parallel, i.e., both can be scheduled within n^{4d} time units (see Figure (5.9-a)). Moreover, we have the following lemma (for intuition see Figure (5.9-b)).

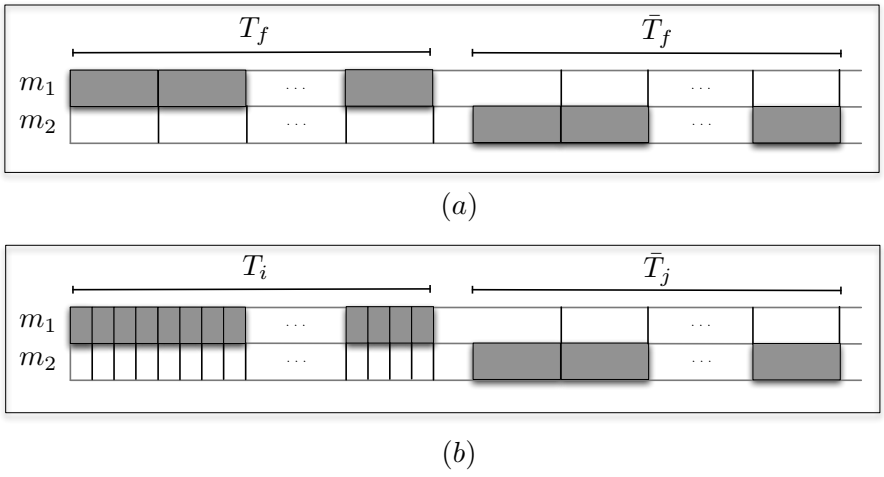


Figure 5.9: a) An example of two compatible types T_f and $\bar{T}_f$ that can be scheduled in parallel. b) Two types $T_i, \bar{T}_j$ with $i > j$, for which there are "many" time units during which they cannot be scheduled in parallel.

Lemma 5.4.2 *Two types T_i and $\bar{T}_j$ with $i \neq j$ can be scheduled in parallel during at most n^{4d}/n^4 time units in any feasible schedule.*

Proof. If $i > j$ then, as T_i has a short-operation on machine m_2 between any two consecutive long-operations on machine m_1 at most one long-operation of T_i is scheduled in parallel with any long-operation of $\bar{T}_j$. Since $\bar{T}_j$ has n^{4j} long-operations and each long-operation of T_i requires $n^{4(d-i)}$ time units, it follows,

by using $i > j$, that the operations of $\bar{T}_j$ and T_i overlap at most $n^{4(d-1)}$ time units. The same result can be obtained when $i < j$ by using symmetric arguments. □

Configurations

For $i = 1, \ldots, |E|$, a *configuration* $C_i = (T_{\pi_{i,1}}, \ldots, T_{\pi_{i,\log n}})$ is an ordered sequence of $\log n$ types, where $\pi_{i,j} \in \{1, \ldots, d\}$ denotes the frequency of the j-th type of configuration C_i. Lemma 5.4.1 shows that we can define a set of configurations $\mathscr{C} = \{C_i : i = 1, \ldots, |E|\}$ such that any two configurations $C_i \in \mathscr{C}$ and $C_j \in \mathscr{C}$ with $i \neq j$ have at most $\varepsilon \log n$ types in common, for $\varepsilon > 0$ arbitrarily small. The set $\bar{\mathscr{C}} = \{\bar{C}_i : i = 1, \ldots, |E|\}$ is defined in a similar way by using the types $\bar{T}_i$, i.e., for $i = 1, \ldots, |E|$ we have $\bar{C}_i = (\bar{T}_{\pi_{i,1}}, \ldots, \bar{T}_{\pi_{i,\log n}})$. Note that a configuration requires $n^{4d} \log n$ time units.

Blocks

We are now ready to define the different blocks. For $i = 1, \ldots, |E|$, *block* B_i is obtained by concatenating C_i for n^2-times, i.e. $B_i := (C_i)^{n^2}$; similarly $\bar{B}_i := (\bar{C}_i)^{n^2}$. Let $D = n^{4d+2} \log n$ be the length of a block. As in the case of compatible types, it is easy to see that two blocks B_i and $\bar{B}_i$ can be scheduled in parallel. However, only a tiny fraction of the operations of a configuration $\bar{C}_i$ can overlap the operations of a block B_j, if $i \neq j$.

Lemma 5.4.3 *The operations of a configuration $\bar{C}_i$ and a block B_j, with $i \neq j$, can be scheduled in parallel during at most $\epsilon n^{4d} \log n$ time units in any feasible schedule, where $\epsilon > 0$ can be made an arbitrarily small constant.*

Proof. The block B_j is composed of n^2 repetitions of C_j. Note that configuration $\bar{C}_i$ has at most $\epsilon \log n$ compatible types with configuration C_j, where $\epsilon > 0$ can be made an arbitrarily small constant. By Lemma 5.4.2 together with the fact that B_j is a sequence of $n^2 \log n$ types, we have that the remaining $(1-\epsilon)\log n$ types of $\bar{C}_i$ can be scheduled in parallel with B_j during at most $(1-\epsilon)\log n \cdot n^2 \log n \cdot n^{4d}/n^4 = o(n^{4d} \log n)$ time units. Hence, the operations of $\bar{C}_i$ and block B_j can be scheduled in parallel during at most $\epsilon n^{4d} \log n + o(n^{4d} \log n) < \epsilon' n^{4d} \log n$ time units, for some $\epsilon' > 0$ that can be made an arbitrarily small constant. □

Jobs

The blocks are now used as building blocks for defining the *jobs*. We will define two kinds of jobs: a big job and vertex jobs. The *big job* j_b is composed of

an *edge-part* $P_{E(b)}$, followed by a *tail-part* $P_{T(b)}$, a *slack-part* $P_{S(b)}$, and finally a *remaining-part* $P_{R(b)}$, defined as follows:

$$\begin{aligned} P_{E(b)} &:= (B_1, B_2, \ldots, B_{|E|}) \\ P_{T(b)} &:= [m_2, D \cdot \beta n] \\ P_{S(b)} &:= ([m_1, 1])^{D \cdot 3(1-\beta)n} \\ P_{R(b)} &:= [m_2, D \cdot (1-\beta)n] \end{aligned}$$

Note that the length of job j_b is $L := D(|E| + n + 3(1-\beta)n) = O(nD)$. A high level representation of the long job j_b is given in Figure (5.10) (for simplicity the structure of the edge-part is omitted; the building blocks of this part have been previously described).

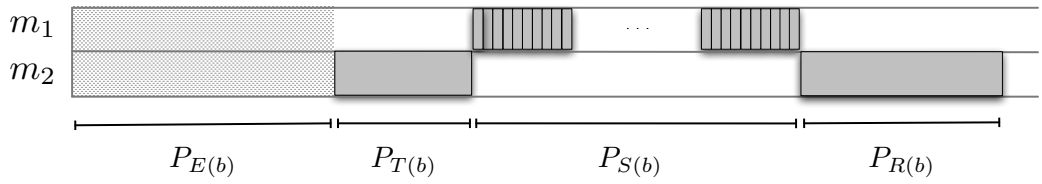


Figure 5.10: An overview of j_b.

We have a *vertex job* j_v for each vertex $v \in V$. Let e_i, e_j, e_k be the 3 edges incident to v with $i < j < k$. Job j_v is composed of an *edge-part* $P_{E(v)}$ followed by a *tail-part* $P_{T(v)}$ defined as follows:

$$\begin{aligned} P_{E(v)} &:= (\bar{B}_i, \bar{B}_j, \bar{B}_k) \\ P_{T(b)} &:= [m_1, D] \end{aligned}$$

The length of a vertex job is $4D$.

The following fundamental lemma motivates our construction. It shows that for any pair $\{u, v\} \in E$ of adjacent vertices, either j_u or j_v cannot be completed before the end of j_b's tail-part without delaying job j_b $\Omega(D)$ time units. It follows that, without delaying job j_b, only jobs corresponding to vertices that form an independent set can be completed before the end of j_b's tail-part.

Lemma 5.4.4 *For any $i \in \{1, \ldots, |E|\}$, if there are two copies of block $\bar{B}_i$ to be scheduled, then at least $\gamma \cdot D$ time units of these two blocks cannot be scheduled in parallel with the edge-part of job j_b, for some $\gamma < 1$ that can be made arbitrarily close to* 1.

Proof. Recall that $\bar{B}_i$ is composed of n^2 repetitions of the configuration $\bar{C}_i$. Hence, the two copies contain $2n^2$ copies of configuration $\bar{C}_i$. At most n^2 of these configurations, that in total require D time units, can be scheduled in parallel with block B_i of job j_b's edge-part. Let $\mathscr{R}$ denote the remaining configurations that may only be scheduled in parallel with the blocks $\{B_j : j \neq i\}$ of job j_b's edge-part. Note that $|\mathscr{R}| \geq n^2$. By Lemma 5.4.3, we have that a configuration $\bar{C}_i$ can be scheduled in parallel with a block B_j, with $i \neq j$, during at most $\epsilon n^{4d} \log n$ time units, for an arbitrarily small constant $\epsilon > 0$. As the edge-part of j_b consists of $|E| = 3n/2$ blocks, $|\mathscr{R}| \geq n^2$, and configurations belonging to a job must be scheduled in a fixed order, almost all configurations in $\mathscr{R}$ are scheduled in parallel with at most one block of job j_b's edge-part. It follows that at most $D + n^2 \cdot n^{4d} \log n\epsilon = D(1+\epsilon)$ time units of the two copies (requiring $2D$ time units in total) can be scheduled in parallel with job j_b's edge-part, where $\epsilon > 0$ can be made an arbitrarily small constant. □

5.4.2 Analysis

Completeness

We will see that if graph G has an independent set of size βn then all vertex jobs can be scheduled in parallel with the big job j_b. Thus the makespan of the schedule will equal L (the length of job j_b).

Let $V' \subseteq V$ denote an independent set of G with $|V'| = \beta n$. Since V' forms an independent set, no two vertices are incident to the same edge. Recall that a block $\bar{B}_i$ can be scheduled in parallel with a block B_i, the tail-part of a vertex job requires time D on machine m_1 and the tail-part $P_{T(b)}$ of j_b requires time $D\beta n$ on machine m_2. It follows that the vertex jobs corresponding to the vertices in V' can all be scheduled in parallel with the edge-part $P_{T(b)} = (B_1, B_2, \ldots, B_{|E|})$ and the tail-part $P_{T(b)}$ of j_b. As (i) the slack-part of job j_b consists of $D \cdot 3(1-\beta)n$ unit time operations on m_1, (ii) a block $\bar{B}_i$ can be scheduled in parallel with D slack-operations, and (iii) the remaining-part of job j_b consists of one operation on machine m_2 that requires time $D(1-\beta)n$, the $(1-\beta)n$ jobs corresponding to the vertices of $V \setminus V'$ can be scheduled in parallel with the slack-part and remaining-part of j_b.

Soundness

As the makespan equals L — the length of job j_b — in the completeness case, we will analyze the soundness case by showing that there is a fraction of the operations belonging to the vertex jobs that are not scheduled in parallel with

j_b. Then it follows that if graph G has no independent set of size αn then the length of any schedule is at least $(1+\Omega(1))L$.

For any given schedule, let t_1 be the time at which the tail-part $P_{T(b)}$ of j_b is completed, and t_2 the time at which the remaining-part $P_{R(b)}$ of j_b starts. Let $T := n \cdot D$ denote the sum of the lengths of tail-parts of the vertex jobs. Let τ_1,τ_2 and τ_3 be the fraction of T spent to schedule tail-operations of the vertex jobs during time interval $[0, t_1)$, $[t_1, t_2)$ and $[t_2, \infty)$, respectively.

It is easy to observe that any positive value of τ_2 implies that $\tau_2 \cdot T$ time units are not scheduled in parallel with job j_b. Similarly a positive value of τ_3 implies that $\max\{0, (\tau_3 - (1-\beta))T\}$ time units are not scheduled in parallel with j_b. Finally, note that there are at least $\tau_1 \cdot n$ vertex jobs that complete their edge-part before time t_1. Since G has no independent set of size αn, it follows that there are at least $\max\{0, (\tau_1 - \alpha)n\}$ conflicting pairs of vertex jobs (i.e., corresponding to adjacent pairs of vertices). There are thus two "conflicting" copies of $\max\{0, (\tau_1 - \alpha)n\}$ different blocks from $\{\bar{B}_i : i = 1, \ldots, |E|\}$ to be scheduled before time t_1. By using Lemma 5.4.4, we can easily check that at least $(\tau_1 - \alpha)n \cdot \gamma \cdot D$ time units of these conflicting blocks cannot be scheduled in parallel with job j_b.

By the above arguments, it follows that the makespan of the schedule is at least the length of job j_b plus $(\beta - \alpha)\gamma \cdot n \cdot D$, where $\gamma < 1$ can be made arbitrarily close to 1. Hence, as $L = O(nD)$, the length of any schedule in the soundness case is at least $(1+\Omega(1))L$.

5.5 Conclusions

Woeginger & Schuurman [SW99] highlighted the poor understanding of the approximability of job shop and flow shop scheduling as two of the ten most prominent open problems in the area of approximation algorithms for NP-hard machine scheduling problems.

In this chapter we have resolved many of these questions by using strong hardness results for coloring by Khot [Kho01] together with novel "gap" constructions that build upon previous work by Feige & Scheideler [FS02]. The main results of our work can be summarized as follows.

1. The $O((\log \mathrm{lb})^{1+\epsilon})$-approximation algorithm [CS00; FS02], where $\epsilon > 0$ can be made arbitrarily close to 0, for acyclic job shops and generalized flow shops is essentially the best possible.

2. To improve the approximation guarantee for flow shops, one needs to (i) improve the polynomial time computable lower bound on the opti-

mal makespan and (ii) use the fact that all jobs are processed on every machine.

3. It is necessary to restrict both the machines *and* the number of operations per job to obtain a PTAS for the job shop problem with makespan objective. That it is sufficient follows from the work by Jansen, Solis-Oba & Sviridenko [JSOS03].

With our current techniques we have been unable to address some shop scheduling problems, whose approximability remains poorly understood. Below we list three prominent problems, all of them with a significant "gap" between the best known approximation guarantees and inapproximability results.

1. The job shop problem admits an $O((\log \mathrm{lb})^2/(\log\log \mathrm{lb})^2)$-approximation algorithm [GPSS01]. Our results imply that it is unlikely to approximate job shops within a factor $O((\log \mathrm{lb})^{1-\epsilon})$, for any $\epsilon > 0$. To the best of our knowledge no instances of the job shop problem are known with optimal makespan a $\omega(\log \mathrm{lb})$ factor away from the lower bound lb. This leaves open the possibility that job shop scheduling might have an $O(\log \mathrm{lb})$-approximation algorithm.

2. Job shop scheduling with preemption admits an $O(\log m/\log\log m)$-approximation algorithm [BKS06] and preemptive acyclic job shops admits an $O(\log\log \mathrm{lb})$-approximation algorithm [FS02], which is currently also the algorithm of choice for flow shops with preemption. The only negative result [WHH$^+$97], says that it is NP-hard to approximate these problems within a factor less than $5/4$.

3. The flow shop problem has an $O((\log \mathrm{lb})^{1+\epsilon})$-approximation algorithm, where $\epsilon > 0$ can be made arbitrarily close to 0 [CS00; FS02]. On the other hand it is only known that it is NP-hard to approximate flow shops within a factor less than $5/4$ [WHH$^+$97]. This leaves open the possibility that one can use the fact that all jobs have to be processed on every machine to even obtain a constant factor approximation algorithm for flow shop scheduling.

Chapter 6

Conclusions

In this thesis we addressed the approximability of several classical scheduling problems where there is a large "gap" in our understanding. Our hypothesis was that stronger hardness of approximation results were needed to close this gap. This was motivated by the fact that for these problems there were few negative results that took advantage of new methods, such as probabilistic checkable proofs. We showed our hypothesis to be valid and the main negative results that we obtained are the following:

- For precedence-constrained scheduling ($1|prec|\sum w_j C_j$), (uniform) sparsest cut, and optimal linear arrangement we gave the first negative results that rule out a PTAS for each of those problems.
- For the general version of flow shops, where jobs are allowed to skip machines, we gave a negative result that matches the best known approximation algorithm [FS02]. Our result is also tight for the more general acyclic job shop problem and gives the first non-constant inapproximability result for the job shop problem.
- We showed that restricting the number of machines *and* the number of operations per job is necessary to obtain a PTAS for the job shop problem. That it is sufficient follows from the work in [JSOS03].

Apart from the negative results, we also gave positive results for the precedence-constrained single machine scheduling problem with the weighted sum of completion times objective. More precisely, we derived a framework that guarantees a better approximation ratio as soon as precedence constraints have low complexity, where the complexity of the precedence constraints is measured by their dimension. Perhaps the most interesting part of this framework is not the improved approximation guarantees, but rather the concrete connection established between the scheduling problem and dimension theory of partial orders.

Indeed, in [CS05; AM09] it was proved that this classical scheduling problem is a special case of the fundamental weighted vertex cover problem and, here, we pointed out that the graph obtained is in fact the graph of incomparable pairs previously studied in the dimension theory of partial orders.

6.1 Future Directions

In what follows, we briefly describe some interesting future research directions that are related to this thesis.

Tailor-Made PCPs for Ordering/Scheduling Problems

A successful approach for proving strong inapproximability results has been to design tailor-made PCPs for the specific problems. An interesting continuation of our work would be to do the same for scheduling and ordering problems. The first result in this direction was the recent so-called Quasi-random PCP by Khot [Kho06], which we used to obtain inapproximability results for precedence-constrained scheduling on a single machine to minimize the sum of weighted completion times.

The current Quasi-random PCP construction assumes that NP-complete problems are not solvable in randomized subexponential time (a stronger assumption than the standard P$\neq$ NP) and the lower bounds obtained are weak (no $(1+\epsilon)$-approximation algorithms exist). A long term goal would be to obtain special tailored PCP constructions for precedence-constrained scheduling that either achieve stronger lower bounds or use more standard assumptions like P$\neq$NP.

Stronger Inapproximability Results Assuming the Unique Games Conjecture

Today's techniques seem insufficient to analyze the approximability for several NP-hard optimization problems. A possible way to overcome this difficulty is to adopt a stronger assumption. The unique games conjecture (UGC) is such an assumption asserting that a certain optimization problem is NP-hard [Kho02]. The unique games conjecture has fueled a lot of development in hardness of approximation, and whether the conjecture is true or false has become a central open problem. Assuming the UGC, researchers have been able to obtain hardness of approximation results that match the best known positive results for several optimization problems for which no tight or nearly tight results were previously known (see e.g. [KKMO04; Aus07; KR08; Rag08]).

It would be interesting to investigate if the UGC can be used to obtain similar results for basic scheduling problems, such as the ones listed in [SW99]. In an exiting recent development, Bansal & Khot [BK09] showed that a new stronger version of the unique games conjecture implies the tight negative result that $1|prec|\sum w_j C_j$ (the scheduling problem addressed in Chapter 3) is NP-hard to approximate within a factor $2-\epsilon$, for any $\epsilon > 0$. Following their work, it would be interesting to understand (i) if their assumption is equivalent to the unique games conjecture and (ii) if similar techniques can be extended to other fundamental scheduling problems.

Semidefinite Programming for Scheduling Problems

A large fraction of the current approximation algorithms are built around linear programming. A powerful generalization of linear programming (that still can be solved in polynomial time) is semidefinite programming.

In the seminal paper [GW95], Goemans & Williamson introduced the use of semidefinite programming in the design of approximation algorithms. They considered the max cut problem and gave a 0.87856-approximation algorithm; a big improvement over the previous best approximation ratio of 1/2. Since then semidefinite programming has found numerous applications and is today one of the most powerful tools in the design of approximation algorithms.

To the best of our knowledge, the only approximation algorithm for scheduling problems that uses semidefinite programming is Skutella's 1.5-approximation algorithm for the problem of scheduling unrelated parallel machines so as to minimize the total weighted completion time [Sku01]. As semidefinite programming is a generalization of linear programming, we believe that a natural and fruitful research direction would be to extend the use of semidefinite programming in the design of approximation algorithms for scheduling problems.

Appendix A

Basic Definitions

In this appendix, we give some basic definitions that are used throughout this thesis. We start with common graph terminology, followed by the definitions of essential classical decision and optimization problems. Finally, we review the definitions of some complexity classes.

For a more comprehensive discussion of these topics, we refer the interested reader to the textbooks [GJ79; Hoc95; ACG+99; Vaz01].

A.1 Graph Terminology

While reading the definitions, we also recommend the reader to see Figure A.1. We denote a *graph* G with vertex set V and edge set E, by $G(V,E)$ or $G=(V,E)$. Graph G is *bipartite* if its vertex set can be partitioned into two sets V' and W' so that each edge is incident to one vertex in V' and one vertex in W', i.e., $E \subseteq \{\{v,w\} : v \in V', w \in W'\}$. We will sometimes refer to such a bipartite graph as $G(V',W',E)$ and it is said to be m by n bipartite if $|V'|=m$ and $|W'|=n$.

We say that a vertex v has degree $d(v)$, if v is adjacent to $d(v)$ other vertices. The *degree of a graph* $G(V,E)$, denoted by $d(G)$ or $\Delta(G)$, is then $\max_{v\in V} d(v)$. If all vertices of a graph have degree 3, then the graph is said to be a cubic graph.

A *complete graph* $G(V,E)$ (also called *clique*) is the graph where any two vertices u and v are adjacent. Similarly, a bipartite graph $G(V,W,E)$ is said to be complete (or a *bipartite clique*) if any two vertices $v \in V$ and $w \in W$ are adjacent.

Given a graph $G(V,E)$, let $V' \subseteq V$ be a subset of the vertices. We say that graph $H(V',E')$ is the subgraph of G *induced* by V', if $E' = \{\{u,v\} \subseteq V' : \{u,v\} \in E\}$. Finally, two graphs $G(V,E)$ and $H(W,E')$ are isomorphic if there exists a bijection $f : V \to W$ such that $\{u,v\} \in E \Leftrightarrow \{f(u),f(v)\} \in E'$.

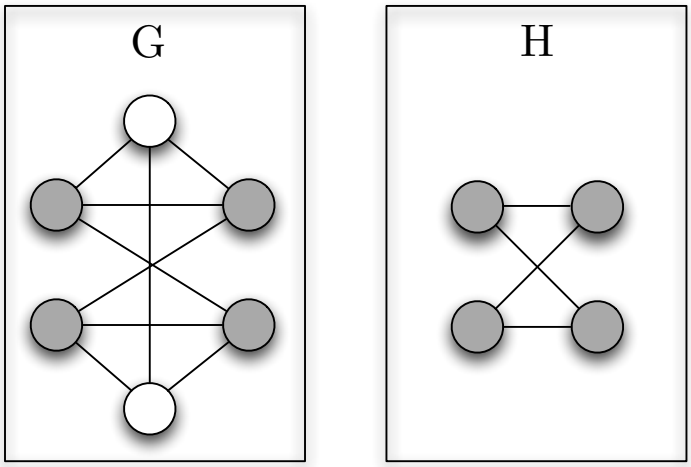


Figure A.1: Graph G is a cubic graph, i.e., all vertices have degree 3. Graph H is the subgraph of G induced by the gray vertices, which in turn is a 2 by 2 complete bipartite graph.

A.2 Some Decision and Optimization Problems

Arguably, the most basic NP-complete decision problems are satisfiability and its restriction 3SAT.

- *Satisfiability (SAT):* Given a Boolean formula ϕ, decide if there is some truth assignment to the variables in ϕ such that ϕ is true.
- 3*SAT:* Given a Boolean 3CNF formula ϕ (a formula that is a conjunction of clauses where each clause is a disjunction of at most 3 literals), decide if there is some truth assignment to the variables in ϕ such that ϕ is true.

Below, we list some of the more fundamental optimization problems that appear in this thesis.

- *MAX-3SAT:* Given a Boolean 3CNF formula ϕ, find an assignment to the variables in ϕ that satisfies the largest number of clauses.
- *Weighted vertex cover:* Given a graph $G(V, E)$ where each vertex v has a nonnegative weight w_v, find a vertex cover $V' \subseteq V$ — a subset of the vertices so that for each $\{u, v\} \in E$, $\{u, v\} \cap V' \neq \emptyset$ — that minimizes $\sum_{v \in V'} w_v$. (In the unweighted version all vertices have weight one.)

- *Weighted independent set:* Given a graph $G(V,E)$ where each vertex v has a nonnegative weight w_v, find an independent set $V' \subseteq V$ — a subset of the vertices so that for any two vertices $u \in V'$ and $v \in V'$, $\{u,v\} \notin E$ — that maximizes $\sum_{v \in V'} w_v$. (In the unweighted version all vertices have weight one.)

- *Graph coloring:* Given a graph $G(V,E)$ assign a color to each vertex such that two adjacent vertices receive different colors and the number of used colors is minimized.

A.3 Complexity Classes

Here, we list the definitions of some complexity classes that appear in the plausible assumptions we use when proving hardness of approximation results.

- P — class of decision problems that can be solved in polynomial time.

- NP — class of decision problems whose solutions can be verified in polynomial time.

- $DTIME(f(n))$ — class of decision problems that can be solved using a deterministic algorithm that runs in time $O(f(n))$ on an instance of size n.

- $ZTIME(f(n))$ — class of decision problems that can be solved using a randomized algorithm that always gives the correct answer and has expected running time $O(f(n))$ on an instance of size n.

- $BPTIME(f(n))$ — class of decision problems that can be solved using a randomized algorithm that gives the correct answer with probability at least $2/3$ and runs in time $O(f(n))$ on an instance of size n.

An example of an assumption used in this thesis is $NP \not\subseteq DTIME\left(n^{\log n}\right)$, i.e., that there are problems in NP that cannot be solved using an algorithm that runs in time $O\left(n^{\log n}\right)$ on an instance of size n.

Bibliography

[ACG+99] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation*. Springer, 1999.

[AHK04] S. Arora, E. Hazan, and S. Kale. $O(\sqrt{\log n})$-approximation to sparsest cut in $O(n^2)$ time. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 238–247, 2004.

[AK00] P. Alimonti and V. Kann. Some APX-completeness results for cubic graphs. *Theoretical Computer Science*, 237(1-2):123–134, 2000.

[AKK+08] S. Arora, S. Khot, A. Kolla, D. Steurer, M. Tulsiani, and N. K. Vishnoi. Unique games on expanding constraint graphs are easy: extended abstract. In *Proceedings of the 40th annual ACM symposium on Theory of computing (STOC)*, pages 21–28, 2008.

[ALM+98] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[AM09] C. Ambühl and M. Mastrolilli. Single machine precedence constrained scheduling is a vertex cover problem. *Algorithmica*, 53(4):488–503, 2009.

[AMMS07] C. Ambühl, M. Mastrolilli, N. Mutsanas, and O. Svensson. Scheduling with precedence constraints of low fractional dimension. In *Proceedings of the 12th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 130–144, 2007.

[AMMS08] C. Ambühl, M. Mastrolilli, N. Mutsanas, and O. Svensson. Precedence constraint scheduling and connections to dimension theory of partial orders. *Bulletin of the European Association for Theoretical Computer Science (EATCS)*, 95:45–58, 2008.

[AMS06] C. Ambühl, M. Mastrolilli, and O. Svensson. Approximating precedence-constrained single machine scheduling by coloring. In *Proceedings of the 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 15–26, 2006.

[AMS07] C. Ambühl, M. Mastrolilli, and O. Svensson. Inapproximability results for sparsest cut, optimal linear arrangement, and precedence constrained scheduling. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 329–337, 2007.

[ARV04] S. Arora, S. Rao, and U. V. Vazirani. Expander flows, geometric embeddings and graph partitioning. In *Proceedings of the 36th annual ACM symposium on Theory of computing (STOC)*, pages 222–231, 2004.

[AS98] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[Aus07] P. Austrin. Towards sharp inapproximability for any 2-CSP. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 307–317, 2007.

[Bec91] J. Beck. An algorithmic approach to the lovasz local lemma. *Random Structures and Algorithms*, 2(4):343–365, 1991.

[BK09] N. Bansal and S. Khot. Optimal long code test with one free bit. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2009. To appear.

[BKS06] N. Bansal, T. Kimbrel, and M. Sviridenko. Job shop scheduling with unit processing times. *Mathematics of Operations Research*, 31:381–389, 2006.

[BL76] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and planarity using pq-tree algorithms. *Journal of Computational Systems Science*, 13:335–379, 1976.

[BS92a] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. *Information and Computation*, 96(1):77–94, 1992.

[BS92b] G. R. Brightwell and E. R. Scheinerman. Fractional dimension of partial orders. *Order*, 9:139–158, 1992.

[CC00] Y. Cheng and G. M. Church. Biclustering of expression data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 93–103, 2000.

[CGR08] S. Chawla, A. Gupta, and H. Räcke. Embeddings of negative-type metrics and an improved approximation to generalized sparsest cut. *ACM Transactions on Algorithms*, 4(2):1–18, 2008.

[CH99] F. A. Chudak and D. S. Hochbaum. A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine. *Operations Research Letters*, 25:199–204, 1999.

[CHKR06] M. Charikar, M. T. Hajiaghayi, H. Karloff, and S. Rao. l_2^2 spreading metrics for vertex ordering problems. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm (SODA)*, pages 1018–1027, 2006.

[CKK+06] S. Chawla, R. Krauthgamer, R. Kumar, Y. Rabani, and D. Sivakumar. On the hardness of approximating multicut and sparsest-cut. *Computational Complexity*, 15(2):94–114, 2006.

[CM99] C. Chekuri and R. Motwani. Precedence constrained scheduling to minimize sum of weighted completion times on a single machine. *Discrete Applied Mathematics*, 98(1-2):29–38, 1999.

[Coo71] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing (STOC)*, pages 151–158, 1971.

[CPW98] B. Chen, C. N. Potts, and G. J. Woeginger. A review of machine scheduling: Complexity, algorithms and approximability. *Handbook of Combinatorial Optimization*, 3:21–169, 1998.

[CS00] A. Czumaj and C. Scheideler. A new algorithm approach to the general lovász local lemma with applications to scheduling and satisfiability problems (extended abstract). In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 38–47, 2000.

[CS05] J. R. Correa and A. S. Schulz. Single machine scheduling with precedence constraints. *Mathematics of Operations Research*, 30(4):1005–1021, 2005.

[DKST01] M. Dawande, P. Keskinocak, J. M. Swaminathan, and S. Tayur. On bipartite and multipartite clique problems. *Journal of Algorithms*, 41(2):388–403, 2001.

[DKSV06] N. R. Devanur, S. Khot, R. Saket, and N. K. Vishnoi. Integrality gaps for sparsest cut and minimum linear arrangement problems. In *Proceedings of the 38th annual ACM symposium on Theory of computing (STOC)*, pages 537–546, 2006.

[DM41] B. Dushnik and E. W. Miller. Partially ordered sets. *American Journal of Mathematics*, 63:600–610, 1941.

[DPS02] J. Díaz, J. Petit, and M. Serna. A survey of graph layout problems. *ACM Computing Surveys*, 34(3):313–356, 2002.

[DS05] I. Dinur and S. Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162(1):439–485, 2005.

[EEI64] W. L. Eastman, S. Even, and I. M. Isaacs. Bounds for the optimal scheduling of n jobs on m processors. *Management Science*, 11(2):268–279, 1964.

[Fei98] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[Fei02] U. Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the 34th annual ACM symposium on Theory of computing (STOC)*, pages 534–543, 2002.

[FG65] D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15(3):835–855, 1965.

[FGL+96] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. *Journal of the ACM*, 43(2):268–292, 1996.

[FJM08] A. V. Fishkin, K. Jansen, and M. Mastrolilli. Grouping techniques for scheduling problems: Simpler and faster. *Algorithmica*, 51(2):183–199, 2008.

[FK98] U. Feige and J. Kilian. Zero knowledge and the chromatic number. *Journal of Computer and System Sciences*, 57(2):187–199, 1998.

[FL07] U. Feige and J. R. Lee. An improved approximation ratio for the minimum linear arrangement problem. *Information Processing Letters*, 101(1):26–29, 2007.

[FS02] U. Feige and C. Scheideler. Improved bounds for acyclic job shop scheduling. *Combinatorica*, 22(3):361–399, 2002.

[FT94] S. Felsner and W. T. Trotter. On the fractional dimension of partially ordered sets. *Discrete Mathematics*, 136:101–117, 1994.

[FT00] S. Felsner and W. T. Trotter. Dimension, graph and hypergraph coloring. *Order*, 17(2):167–177, 2000.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and intractability; a guide to the theory of NP-completeness*. W.H. Freeman, 1979.

[GJS76] M. R. Garey, D. S. Johnson, and L. J. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[GLLK79] R. Graham, E. Lawler, J.K. Lenstra, and A. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[GPSS01] L. A. Goldberg, M. Paterson, A. Srinivasan, and E. Sweedyk. Better approximation guarantees for job-shop scheduling. *SIAM Journal on Discrete Mathematics*, 14(1):67–92, 2001.

[Gra66] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal (BSTJ)*, 45:1563–1581, 1966.

[GW95] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

[GW00] M. X. Goemans and D. P. Williamson. Two-dimensional Gantt charts and a scheduling algorithm of Lawler. *SIAM Journal on Discrete Mathematics*, 13(3):281–294, 2000.

[Hås99] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182:105–142, 1999.

[Hås01] J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

[HJ07] R. Hegde and K. Jain. The hardness of approximating poset dimension. *Electronic Notes in Discrete Mathematics*, 29:435 – 443, 2007.

[HK03] J. Holmerin and S. Khot. A strong inapproximability gap for a generalization of minimum bisection. In *Proceedings of 18th IEEE Annual Conference on Computational Complexity (CCC)*, pages 371–378, 2003.

[Hoc83] D. S. Hochbaum. Efficient bounds for the stable set, vertex cover and set packing problems. *Discrete Applied Mathematics*, 6:243–254, 1983.

[Hoc95] D.S. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Company, Boston, 1995.

[HSSW97] L. A. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein. Scheduling to minimize average completion time: off-line and on-line approximation algorithms. *Mathematics of Operations Research*, 22:513–544, 1997.

[HSW96] L. A. Hall, D. B. Shmoys, and J. Wein:. Scheduling to minimize average completion time: Off-line and on-line algorithms. In *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 142–151, 1996.

[HSW01] H. Hoogeveen, P. Schuurman, and G. J. Woeginger. Non-approximability results for scheduling problems with minsum criteria. *INFORMS Journal on Computing*, 13(2):157–168, 2001.

[JSOS03] K. Jansen, R. Solis-Oba, and M. Sviridenko. Makespan minimization in job shops: A linear time approximation scheme. *SIAM Journal on Discrete Mathematics*, 16(2):288–300, 2003.

[Kar72] R. M. Karp. *Reducibility Among Combinatorial Problems*, pages 85–103. Plenum Press, 1972.

[Kho01] S. Khot. Improved inapproximability results for maxclique, chromatic number and approximate graph coloring. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 600–609, 2001.

[Kho02] S. Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the 34th annual ACM symposium on Theory of computing (STOC)*, pages 767–775, 2002.

[Kho06] S. Khot. Ruling out PTAS for graph min-bisection, dense k-subgraph, and bipartite clique. *SIAM Journal on Computing*, 36(4):1025–1071, 2006.

[KKMO04] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal inapproximability results for max-cut and other 2-variable CSPs? In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 146–154, 2004.

[Knu69] D. E. Knuth. *The Art of Computer Programming volume 2: Seminumerical algorithms*. Reading, MA: Addison-Wesley, 1969.

[KR08] S. Khot and O. Regev. Vertex cover might be hard to approximate to within $2-\varepsilon$. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

[KS02] S. G. Kolliopoulos and G. Steiner. Partially-ordered knapsack and applications to scheduling. In *Proceedings of the 10th Annual European Symposium on Algorithms (ESA)*, pages 612–624, 2002.

[KV05] S. Khot and N. K. Vishnoi. The unique games conjecture, integrality gap for cut problems and embeddability of negative type metrics into l_1. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 53–62, 2005.

[Law78] E. L. Lawler. Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Annals of Discrete Mathematics*, 2:75–90, 1978.

[LK78] J. K. Lenstra and A. H. G. Rinnooy Kan. The complexity of scheduling under precedence constraints. *Operations Research*, 26:22–35, 1978.

[LLKS93] E. L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. Sequencing and scheduling: Algorithms and complexity. *Handbook in Operations Research and Management Science*, 4:445–522, 1993.

[LMR94] F. T. Leighton, B. M. Maggs, and S. B. Rao. Packet routing and job-shop scheduling in O(congestion + dilation) steps. *Combinatorica*, 14(2):167–180, 1994.

[LMR99] F. T. Leighton, B. M. Maggs, and A. W. Richa. Fast algorithms for finding O(congestion + dilation) packet routing schedules. *Combinatorica*, 19:375–401, 1999.

[LR99] F. T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46(6):787–832, 1999.

[Möh89] R. H. Möhring. Computationally tractable classes of ordered sets. In I. Rival, editor, *Algorithms and Order*, pages 105–193. Kluwer Academic, 1989.

[MPT98] J. Meidanis, O. Porto, and G. P. Telles. On the consecutive ones property. *Discrete Applied Mathematics*, 88(1-3):325–354, 1998.

[MQW03] F. Margot, M. Queyranne, and Y. Wang. Decompositions, network flows and a precedence constrained single machine scheduling problem. *Operations Research*, 51(6):981–992, 2003.

[MS90] D. W. Matula and F. Shahrokhi. Sparsest cuts and bottlenecks in graphs. *Discrete Applied Mathematics*, 27(1-2):113–123, 1990.

[MS08] M. Mastrolilli and O. Svensson. (Acyclic) job shops are hard to approximate. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 583–592, 2008.

[MS09] M. Mastrolilli and O. Svensson. Improved bounds for flow shop scheduling. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 677–688, 2009.

[NS08] V. Nagarajan and M. Sviridenko. Tight bounds for permutation flow shop scheduling. In *Proceedings of the 13th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 154–168, 2008.

[NT73] G. L. Nemhauser and L. E. Trotter. Properties of vertex packing and independence system polyhedra. *Mathematical Programming*, 6:48–61, 1973.

[NT75] G. L. Nemhauser and L. E. Trotter. Vertex packings: Structural properties and algorithms. *Mathematical Programming*, 8:232–248, 1975.

[Pas97] V. T. Paschos. A survey of approximately optimal solutions to some covering and packing problems. *ACM Computing Surveys*, 29(2):171–209, 1997.

[Pee03] R. Peeters. The maximum edge biclique problem is NP-complete. *Discrete Applied Mathematics*, 131:651–654, 2003.

[Pis92] N. N. Pisaruk. The boundaries of submodular functions. *Computational Mathematics and Mathematical Physics*, 32(12):1769–1783, 1992.

[Pis03] N. N. Pisaruk. A fully combinatorial 2-approximation algorithm for precedence-constrained scheduling a single machine to minimize average weighted completion time. *Discrete Applied Mathematics*, 131(3):655–663, 2003.

[Pot80] C. N. Potts. An algorithm for the single machine sequencing problem with precedence constraints. *Mathematical Programming Study*, 13:78–87, 1980.

[PSW91] C. Potts, D. Shmoys, and D. Williamson. Permutation vs. nonpermutation flow shop schedules. *Operations Research Letters*, 10:281–284, 1991.

[PY79] C. H. Papadimitriou and M. Yannakakis. Scheduling interval-ordered tasks. *SIAM Journal on Computing*, 8:405–409, 1979.

[QS02] M. Queyranne and M. Sviridenko. Approximation algorithms for shop scheduling problems with minsum objective. *Journal of Scheduling*, 5(4):287–305, 2002.

[Rab78] I. Rabinovitch. The dimension of semiorders. *Journal of Combinatorial Theory, Series A*, 25:50–61, 1978.

[Rag08] P. Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *Proceedings of the 40th annual ACM symposium on Theory of computing (STOC)*, pages 245–254, 2008.

[RR04] S. Rao and A. W. Richa. New approximation techniques for some linear ordering problems. *SIAM Journal on Computing*, 34(2):388–404, 2004.

[Sch96] A. S. Schulz. Scheduling to minimize total weighted completion time: Performance guarantees of LP-based heuristics and lower bounds. In *Proceedings of the 5th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 301–315, 1996.

[Shm97] D. B. Shmoys. Cut problems and their application to divide-and-conquer. In *Approximation algorithms for NP-hard problems*, pages 192–235. 1997.

[Sid75] J. B. Sidney. Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs. *Operations Research*, 23:283–298, 1975.

[Sku01] M. Skutella. Convex quadratic and semidefinite programming relaxations in scheduling. *Journal of the ACM*, 48(2):206–242, 2001.

[Smi56] W. E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3:59–66, 1956.

[Spe71] J. Spencer. On minimum scrambling sets of simple orders. *Acta Mathematica*, 22:349–353, 1971.

[SSW94] D.B. Shmoys, C. Stein, and J. Wein. Improved approximation algorithms for shop scheduling problems. *SIAM Journal on Computing*, 23:617–632, 1994.

[SU08] A. S. Schulz and N. A. Uhan. Near-optimal solutions and integrality gaps for almost all instances of single-machine precedence-constrained scheduling. Preprint, 2008.

[SW99] P. Schuurman and G. J. Woeginger. Polynomial time approximation algorithms for machine scheduling: ten open problems. *Journal of Scheduling*, 2(5):203–213, 1999.

[Tre04] L. Trevisan. Inapproximability of combinatorial optimization problems. Survey paper, 2004.

[Tro92] W. T. Trotter. *Combinatorics and Partially Ordered Sets: Dimension Theory*. Johns Hopkins Series in the Mathematical Sciences. The Johns Hopkins University Press, 1992.

[Uha08] N. A. Uhan. *Algorithmic and Game-Theoretic Perspectives on Scheduling*. PhD thesis, Massachusetts Institute of Technology (MIT), 2008.

[Vaz01] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2001.

[WHH+97] D. P. Williamson, L. A. Hall, J. A. Hoogeveen, C. A. J. Hurkens, J. K. Lenstra, S. V. Sevastianov, and D. B. Shmoys. Short shop schedules. *Operations Research*, 45:288–294, 1997.

[Woe03] G. J. Woeginger. On the approximability of average completion time scheduling under precedence constraints. *Discrete Applied Mathematics*, 131(1):237–252, 2003.

[Yan82] M. Yannakakis. On the complexity of partial order dimension problem. *SIAM Journal on Algebraic and Discrete Methods*, 22(3):351–358, 1982.