On the Approximability of NP-complete Optimization Problems

Viggo Kann

Department of Numerical Analysis and Computing Science
Royal Institute of Technology
S-100 44 Stockholm
Sweden
`viggo@nada.kth.se`

Dissertation, May 1992

Akademisk avhandling för teknisk doktorsexamen vid
Kungl. Tekniska Högskolan, Stockholm
Maj 1992

ISBN 91-7170-082-X
TRITA-NA-9206 • ISSN 1101-2250 • ISRN KTH/NA–9206

Repro Print AB, Stockholm
MCMXCII

Abstract

This thesis deals with the polynomial approximability of combinatorial optimization problems which are NP-complete in their decision problem version. Different measures of approximation quality are possible and give rise to different approximation preserving reductions and different approximation classes of problems. We present definitions and characteristics of existing measures, reductions and classes and make comparisons. The main contributions comprise the following four subjects.

- We show that the definition of the approximability class Max SNP due to Panconesi and Ranjan differs from the original definition due to Papadimitriou and Yannakakis and that the same problem may or may not be included in the class depending on how it is encoded. A new definition of Max SNP without this limitation is proposed.
- We prove that maximum three dimensional matching is Max SNP-hard using a reduction constructing a structure of rings of trees. Using similar techniques we can show that several other matching and packing problems are Max SNP-hard, for example the maximum H-matching problem (the problem of determining the maximum number of node-disjoint copies of a fixed graph H contained in a variable graph). Most of the problems are Max SNP-complete when bounding the degree of the input structure.
- Some versions of the maximum common subgraph problem are studied and approximation algorithms are given. The maximum bounded common induced subgraph problem is shown to be Max SNP-hard and the maximum unbounded common induced subgraph problem is shown to be as hard to approximate as the maximum independent set problem. The maximum common induced connected subgraph problem is still harder to approximate and is shown to be NPO PB-complete, i.e. complete in the class of optimization problems with optimum value bounded by a polynomial.
- An algorithm which solves the Euclidean travelling salesperson problem in two dimensions optimally is given. The worst case time bound for the algorithm is $2^{O(\sqrt{n}\log n)}$.

As an appendix there is a list of NP optimization problems including their definitions, approximation properties and references.

Keywords: computational complexity, approximability, NP optimization problems, combinatorial problems, graph problems, optimization, approximation, approximation algorithms, approximation preserving reduction, approximability classes, relative error, performance ratio, Max SNP-completeness, maximum three dimensional matching, maximum H-matching, maximum common subgraph, travelling salesperson problem.

Sammanfattning

Denna avhandling handlar om hur väl NP-fullständiga kombinatoriska optimeringsproblem kan approximeras i polynomisk tid. Approximerbarhet kan mätas på flera sätt, vilket har lett till att det finns flera olika approximerbarhetsbevarande reduktioner och flera komplexitetsklasser av problem. I avhandlingen ges definitioner av, egenskaper hos och jämförelser mellan olika approximerbarhetsmått, reduktioner och klasser. Följande fyra punkter sammanfattar huvudsakligen avhandlingens nya resultat.

- Skillnader mellan Panconesis och Ranjans definition av komplexitetsklassen MAX SNP och Papadimitrious och Yannakakis ursprungliga definition påvisas. T ex kan det hända att samma problem antingen är med eller inte i klassen beroende på hur det kodas. En ny definition av MAX SNP som inte har denna brist läggs fram.

- Det tredimensionella matchningsproblemet visas vara MAX SNP-svårt med hjälp av en reduktion som konstruerar en speciell struktur av *trädringar*. Samma teknik kan användas för att visa att flera andra matchnings- och packningsproblem är MAX SNP-svåra, bl a maximal H-matchning (att hitta det största antalet nodskilda kopior av en fix graf H i en variabel graf). Dessa problem visas vara MAX SNP-fullständiga för inmatningar med begränsat gradtal.

- Några varianter av problemet att hitta den största gemensamma delgrafen i två grafer beskrivs. När grafernas gradtal är begränsat visas problemet vara MAX SNP-svårt och när gradtalet är obegränsat visas problemet vara ungefär lika svårt att approximera som problemet att hitta den största oberoende mängden hörn i en graf. Att approximera problemet att hitta den största sammanhängande gemensamma delgrafen visas vara NPO PB-fullständigt, dvs fullständigt i den klass av optimeringsproblem vars optimum är begränsat av ett polynom i indatas storlek.

- En algoritm som löser handelsresandeproblemet i det euklidiska planet beskrivs. Tidskomplexiteten för algoritmen är $2^{O(\sqrt{n}\log n)}$.

I ett appendix finns en lista över optimeringsproblem. För varje problem ges definitionen, approximationsegenskaper och referenser.

Nyckelord: komplexitetsteori, approximerbarhet, NP-optimeringsproblem, approximering, approximeringsalgoritmer, approximerbarhetsbevarande reduktioner, relativt fel, approximerbarhetsfaktor, MAX SNP-fullständighet, maximal tredimensionell matchning, maximal H-matchning, största gemensamma delgraf, handelsresandeproblemet.

Contents

Chapter 1

Introduction

Is there any way to find efficient algorithms which always give good approximations to optimization problems to which it is hard to find the optimal solution? This is the main question motivating the work in this thesis. The answer to the question is: it depends on the problem. Of problems which are equally hard to solve optimally, some can be very well approximated with efficient algorithms, while others, under the well-known and commonly believed assumption that $P \neq NP$, cannot be approximated with efficient algorithms at all.

1.1 Background

Identifying which combinatorial problems are easy to solve and which are hard is an important and challenging task, which has occupied theoretical computer scientists for many years. In order to translate the everyday expression "easy to solve" to mathematical theory the concept of *polynomial time algorithms* has been introduced.

An algorithm is said to run in polynomial time if there is a polynomial p such that the algorithm applied to an input of size n always finds a solution in time $p(n)$, that is after performing $p(n)$ simple instructions. Note that we measure the worst case complexity, that is the time in which we are sure that the algorithm ends regardless of which input of size n we have fed it.

The execution time of a polynomial time algorithm grows slowly enough with increasing input size to be able to be run on a computer, but if the execution time grows exponentially the algorithm is useless for all but the smallest inputs. One of the most accepted ways to prove that a problem is hard is to prove it NP-complete. If an optimization problem is NP-complete we are almost certain that it cannot be solved optimally in polynomial time.

If we are given an algorithm solving a problem, we can often use it to solve other, similar problems. We can even, given two problems A and B, specify in advance how to use any algorithm for problem B to solve problem A. Such a specification is called a *reduction* from A to B – we *reduce* A to B. If the reduction itself is easy to execute, then we have shown that problem B is at least as hard to solve as A, in the sense that if we can solve B we can also solve

A in about the same time – a reduction is thus a way to compare problems. Therefore one often symbolizes the reduction as an inequality $A \leq B$.

In the 1970s and 1980s a lot of decision problems (problems with the answer *yes* or *no*) were reduced to each other. All these problems have a common property: for every input to a problem with solution *yes* there is a proof that the input has solution *yes* and this proof can be verified in polynomial time. Any problem with this property is called an NP problem. Obviously all problems which can be solved by polynomial time algorithms satisfy this property. We say that P $\subseteq$ NP where P is the set of problems which can be solved by polynomial time algorithms and NP is the set of NP problems.

A problem which is greater that all NP problems, that is a problem to which every problem in NP can be reduced, is called NP-hard. If an NP-hard problem is itself an NP problem it is called NP-complete. Thus all NP-complete problems are equally hard to solve, since they are interreducible. If there is a polynomial time algorithm for any NP-complete problem then P = NP and every NP problem can be solved in polynomial time. Despite enormous efforts the question whether P = NP is still unanswered. The common belief nowadays is that P $\neq$ NP and a big part of the research in theoretical computer science, among them this thesis, have P $\neq$ NP as a fundamental assumption.

A well-known and practical example of an NP-complete problem is the *travelling salesperson problem* or TSP. The travelling salesperson has to visit n cities and knows in advance the distances between every pair of cities. She wants to find the tour through all n cities, starting and ending in the same city, which minimizes the distance travelled. In this formulation TSP is not really a decision problem – the answer is an optimal tour and not *yes* or *no*. It can be formulated as a decision problem by introducing a new parameter k in the input. The question now will be: is there a tour of length at most k through all cities?

TSP was one of the first problems shown to be NP-complete and there is no known polynomial time algorithm which can solve it for every value of k. Unfortunately, even several natural restrictions of the travelling salesperson problem are also NP-complete and thus probably intractable, for example when the distances satisfy the triangle inequality and when the cities lie in the plane.

Any optimization problem which in the same way as TSP can be seen as an NP decision problem is an NP optimization problem or an NPO problem. Provided that P $\neq$ NP there is no algorithm which finds the optimal solution of an NP-complete optimization problem in polynomial time.

1.2 Approximate solutions

In practice though, it is often sufficient to find an approximate solution, which is near the optimal solution, and in many cases this can be done quite fast. Even for NP-complete optimization problems there can exist *polynomial time approximation algorithms*. However the approximability of different NPO problems differs enormously.

For example the travelling salesperson problem with triangle inequality can in polynomial time be solved approximately within a factor $3/2$, i.e. one can find a trip of length at most 3/2 times the shortest possible trip [21]. The general Tsp cannot be approximated within any constant factor if P $\neq$ NP [35]. Another example is the knapsack problem, which is NP-complete but can be approximated within every constant in polynomial time [46]. Such a scheme for approximating within every constant is called a Ptas (polynomial time approximation scheme). The knapsack problem has in fact even an Fptas (fully polynomial time approximation scheme) which means that it can be approximated within $1 + \varepsilon$ in time polynomial in both the length of the input and $1/\varepsilon$, i.e. the time is bounded by a two variable polynomial where the input length is the first variable and $1/\varepsilon$ the second.

One could suspect that, since all NP-complete optimization problems are reducible to each other, an approximation algorithm for one problem would automatically, via the reduction, give an equally good approximation algorithm for other problems, but this is not true. The reason for this is that the NP-reductions are not strong enough to preserve all of the underlying structure of the optimization versions of the problems.

Several attempts have been made to find a theory explaining why a problem enjoys particular approximation properties. See [19] for a historic survey.

1.3 Degrees of approximability

A breakthrough was made in 1988 when Papadimitriou and Yannakakis defined the class Max SNP and a new concept of reduction, called an *L-reduction*, which preserves approximability within constants [88].

Max SNP contains the problems which can be defined syntactically in a certain manner. Papadimitriou and Yannakakis thereby characterized many polynomially approximable NPO problems in a non-computational way, since every problem in Max SNP can be approximated within a constant in polynomial time.

Furthermore, they called a problem in Max SNP to which every problem in Max SNP can be reduced Max SNP-*complete* and proved several NPO problems to be Max SNP-complete, for example the maximum 3-satisfiability problem (Max 3Sat). Given a boolean formula in 3CNF (a conjunction of clauses where each clause is a disjunction of at most three literals), Max 3Sat is the problem of finding a truth assignment which satisfies the maximum number of clauses.

The L-reduction was later generalized to the P-reduction. Both these reductions are sufficiently strong to preserve approximability within constants. For example, if A P-reduces to B and B has a Ptas then A has also a Ptas. Thus the Max SNP-complete problems are equally hard to approximate, either all have a Ptas or none.

The question of whether the Max SNP-complete problems can be approximated within every constant remained unanswered until recently, when it was answered in the negative (provided that P $\neq$ NP) [2]. Thus showing that a problem is Max SNP-complete gives a good characterization of the approximability: it can be approximated within a constant, but not within every constant.

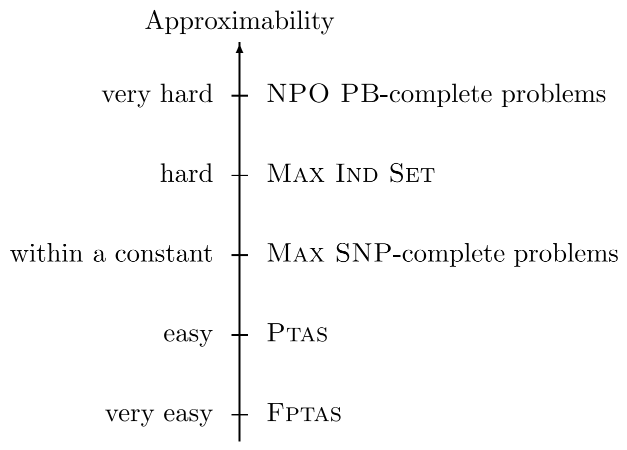

Figure 1.1: *Degrees of approximability of* NPO *problems.*

A considerably harder problem to approximate is the maximum independent set problem (Max Ind Set). The problem is to find the maximum size set of nodes which are independent in a graph, that is which are not directly connected with edges. There is a constant c such that Max Ind Set cannot be approximated within n^c where n is the number of nodes in the graph. If one manages to find a P-reduction from Max Ind Set to a problem, one has thus showed that the problem is hard to approximate.

The problem of finding the longest induced path in a graph (LIP) has the very interesting property that every NPO problem with not too large an optimum value (namely if the optimum value can be bounded by a polynomial in the size of the input) can be P-reduced to it. Thus it is among the hardest of the polynomially bounded NPO problems – an NPO PB-complete problem. Therefore it is even harder to approximate than the maximum independent set problem.

In this framework we can try to place the NPO problems, thereby giving both lower and upper bounds on the approximability of each problem, and thus telling us how good approximations we can hope to find for the problems.

1.4 Organization of the presentation

This thesis deals with polynomial approximation of NPO problems which are NP-complete in their decision problem version. It both presents new material and surveys the existing research. Most of our own material is contained in Section 4.10, Chapter 5, Chapter 6, Section 7.3.3 and the appendices.

In Chapter 2 we present the basic nomenclature that is used throughout the thesis. The class of NPO problems is defined and several measures of approximation are defined and compared. The chapter ends with a discussion why NP-complete problems have different approximability.

The important concept of approximability preserving reductions is introduced in Chapter 3. Different reductions exist which preserve approximability in different ways. We give definitions and properties of most of the approximability preserving reductions used in the literature and compare them with each other. We also introduce a new reduction, the S-reduction, which is suitable to use when reducing between problems which cannot be approximated within a constant.

Chapter 4 contains an exposition of the main approximability classes, both the classes defined using only the approximability of problems and the classes defined using the syntactical formulation of problems by logical formulas. We discuss two different definitions of the Max SNP and Max NP classes, show that they really contain different problems and propose new definitions of these classes. This section is based on our article [54].

Chapter 5 is a compilation and extension of [52] and [53] and contains analyses of several matching and packing problems. We show for example that bounded maximum three dimensional matching, bounded maximum three-set packing, bounded maximum triangle packing and bounded maximum H-matching are Max SNP-complete problems and thus cannot be approximated within every constant.

An interesting family of problems is the family of maximum common subgraph problems. It consists of problems which are formulated in a similar way but differ in approximability. In Chapter 6, which is based on [55], we show that some of these problems can be approximated within a constant, one problem is as hard to approximate as Max Ind Set and one problem is NPO PB-complete.

Chapter 7 deals with the travelling salesperson problem and describes the difference in approximability between the general problem and some restrictions of it. A relatively fast algorithm finding the optimal solution of Tsp in the plane is given. This was first published in [51].

In Chapter 8 we try to give a summary on how NPO problems are related approximability-wise. First we present two groups of related problems: master and slave problems and planar problems. The last section contains all NPO problems mentioned in the thesis, ordered by their approximability.

There are two appendices. The first contains some proofs of some reductions which do not naturally belong to any of the earlier chapters and the second is a list of all mentioned optimization problems including their definition and approximation properties.

1.5 Approximability of other types of problems

This thesis only covers sequential and polynomial time approximation of NPO problems which are NP-complete in their decision problem version. Some work has been done on the approximability of other types of problems.

In 1989 Serna and Spirakis showed that many P-complete problems cannot be approximated within a constant by an algorithm in NC unless P = NC [99]. They also showed other approximability behaviour for several problems and introduced a log-space reduction similar to the L-reduction. Recently some other problems have been shown to be hard to approximate in NC [58, 98].

Stockmeyer has shown some results regarding the approximability of problems in #P [104]. In general every function in #P can be approximated within any constant by an algorithm in the class Δ_3^p of the polynomial hierarchy. If the #P problem is based on an NP-complete problem there is no polynomial time algorithm which approximates it within a constant.

1.6 Approximation using other types of algorithms

A related area is the search for other types of approximation algorithms than sequential polynomial time algorithms.

Peters and Rudolph have shown that there is a logarithmic time parallel approximation scheme for the knapsack problem [91]. There are relatively few results on parallel approximation of NPO problems.

Yao has transferred some known approximability results for approximation of NPO problems to approximation by neural networks [111].

1.7 Approximation by local search

Another connecting active research field is approximation by local optimization. A local search algorithm for an optimization problem tries in every step to find improved solutions by considering perturbations of the current solution. If no perturbation gives a better solution, the solution, which is locally optimal, is output. One question is how good the locally optimal solution is relative to the optimum solution, and another question is whether the local search algorithm always finds a local optimum in polynomial time.

Johnson, Papadimitriou and Yannakakis have defined a complexity class PLS of local search problems and showed that there are PLS-complete problems [50]. Several problems have been shown to be PLS-complete and it has even been argued that PLS-completeness in some sense is the normal behaviour of NP-complete problems [49, 64, 109].

Chapter 2

Definitions

2.1 Basic notation

The following notation will be used throughout this thesis.
Sets:

$\mathbb{Z}$	the set of integers
$\mathbb{Z}^+$	the set of positive integers
$\mathbb{N}$	the set of natural numbers (non-negative integers)
$\mathbb{Q}$	the set of rational numbers
$\mathbb{R}$	the set of real numbers
$\mathbb{R}^+$	the set of positive real numbers
$[a..b]$	the set $\{a, a+1, \ldots, b-1, b\} \subset \mathbb{Z}$
2^A	the set of subsets of A
$\lvert A\rvert$	the cardinality of the set A
$A \subset B$	A is a proper subset of B

Graphs:

graph	undirected graph
$G = \langle V, E\rangle$	graph with node set V and edge set E with edges (v, v')
$G\|_C$	the restriction of the graph G to the nodes or edges in C

Functions:

log	$\log_2$, i.e. the logarithm in base 2
ln	$\log_e$, i.e. the natural logarithm

2.2 NP, NPO and PO

The first definition of NP optimization problems was made by Johnson [48]. The following, very similar, formulation is due to Crescenzi and Panconesi [24].

Definition 2.1 An NPO problem (over an alphabet Σ) is a four-tuple $F = (\mathcal{I}_F, S_F, m_F, opt_F)$ where

- $\mathcal{I}_\mathcal{F} \subseteq \Sigma^*$ is the space of *input instances.* It is recognizable in polynomial time.
- $S_F(x) \subseteq \Sigma^*$ is the space of *feasible solutions* on input $x \in \mathcal{I}_F$. The only requirement on S_F is that there exist a polynomial q and a polynomial time computable predicate π such that for all x in $\mathcal{I}_F$, S_F can be expressed as $S_F(x) = \{y : |y| \le q(|x|) \wedge \pi(x,y)\}$ where q and π only depend on F.
- $m_F : \mathcal{I}_F \times \Sigma^* \to \mathbb{N}$, the *objective function*, is a polynomial time computable function. $m_F(x,y)$ is defined only when $y \in S_F(x)$.
- $opt_F \in \{\max, \min\}$ tells if F is a *maximization* or a *minimization* problem.

Solving an optimization problem F given the input $x \in \mathcal{I}_F$ means finding a $y \in S_F(x)$ such that $m_F(x,y)$ is optimum, that is as big as possible if $opt_F = \max$ and as small as possible if $opt_F = \min$. Let $opt_F(x)$ denote this optimal value of m_F.

A *maximum* solution of a maximization problem is an optimal solution. A *maximal* solution is a locally optimal solution, thus an approximation to the optimal solution. Often it is easy to find a maximal solution but hard to find a maximum solution. Analogously, a *minimum* solution of a minimization problem is an optimal solution and a *minimal* solution is a locally optimal solution.

Example 2.1 Maximum three dimensional matching (Max 3DM) is an example of a problem in NPO. It is a tuple $(\mathcal{I}, S, m, opt)$ where the space of input instances $\mathcal{I}$ contains all sets of triples $T \subseteq X \times Y \times Z$ from pairwise disjunctive sets X, Y and Z. The set of feasible solutions $S(T)$ is the set of matchings of T, where a matching $M \in S(T)$ is a subset of T where no two triples agree in any coordinate. The objective function $m(T, M)$ is the cardinality $|M|$ of the solution M and $opt = \max$. 3DM is a well-known NP-complete problem. Thus Max 3DM can probably not be solved exactly in polynomial time.

Definition 2.2 PO is the class of NPO problems which can be solved in polynomial time.

Maximum matching in a bipartite graph (Max 2DM) is an example of a problem in PO.

2.3 How approximation is measured

Approximating an optimization problem F given the input $x \in \mathcal{I}_F$ means finding any $y' \in S_F(x)$. How good the approximation is depends on the relation between $m_F(x, y')$ and $opt_F(x)$. There are a number of ways to measure the approximability. Different problems have different approximabilities which must be measured in different ways. In this section the standard terminology on approximability measuring is presented. Unless otherwise stated we adopt the terminology used by Garey and Johnson [35].

For most problems which are not too hard to approximate we often measure the approximation using the *relative error.*

Definition 2.3 The *relative error* of a feasible solution with respect to the optimum of an NPO problem F is defined as

$$\mathcal{E}^r_F(x,y) = \frac{|opt_F(x) - m_F(x,y)|}{opt_F(x)}$$

where $y \in S_F(x)$.

If F is a maximization problem we have $opt_F(x) \geq m_F(x,y) \geq 0$ and

$$\mathcal{E}^r_F(x,y) = \frac{opt_F(x) - m_F(x,y)}{opt_F(x)} = 1 - \frac{m_F(x,y)}{opt_F(x)}$$

Thus $0 \leq \mathcal{E}^r_F(x,y) \leq 1$.

If F is a minimization problem we have $m_F(x,y) \geq opt_F(x) \geq 0$ and

$$\mathcal{E}^r_F(x,y) = \frac{m_F(x,y) - opt_F(x)}{opt_F(x)} = \frac{m_F(x,y)}{opt_F(x)} - 1 \geq 0$$

In order to avoid this lack of symmetry between maximization problems and minimization problems Ausiello, D'Atri and Protasi defined the *normalized relative error.*

Definition 2.4 [4] The *normalized relative error* of a feasible solution of an NPO problem F is defined as

$$\mathcal{E}^n_F(x,y) = \frac{|opt_F(x) - m_F(x,y)|}{|opt_F(x) - worst_F(x)|}$$

where $y \in S_F(x)$ and

$$worst_F(x) = \begin{cases} \min\{m_F(x,y) : y \in S_F(x)\} & \text{if } opt_F = \max, \\ \max\{m_F(x,y) : y \in S_F(x)\} & \text{if } opt_F = \min. \end{cases}$$

We observe that $0 \leq \mathcal{E}^n_F(x,y) \leq 1$. For maximization problems $worst_F(x)$ is usually zero, and then the normalized relative error will coincide with the relative error. The normalized relative error is hardly ever used in the literature, probably because it is non-intuitive. See Section 2.6 for an example of the use of the normalized relative error.

Definition 2.5 The *absolute error* of a feasible solution with respect to the optimum of an NPO problem F is defined as

$$\mathcal{E}^a_F(x,y) = |opt_F(x) - m_F(x,y)|$$

where $y \in S_F(x)$.

For most of the NP-complete optimization problems there is no hope of finding a solution with constant absolute error. Therefore this measure is seldom used. Still there are a few NPO problems which have approximation algorithms with constant absolute error guarantee, see Section 8.3.

Orponen and Mannila have generalized these measures and defined a general measure of approximation quality.

Definition 2.6 [83] A general *measure of approximation quality* of a feasible solution of an NPO problem F is a function $\mathcal{E}_F$ which, for each $x \in \mathcal{I}_F$, satisfies $\mathcal{E}_F(x, y) \geq 0$ if and only if $y \in S_F(x)$ and $\mathcal{E}_F(x, y) = 0$ if and only if $m_F(x, y) = opt_F(x)$. We say that the measure is *cost-respecting* if

$$\begin{array}{ll} m_F(x, y_1) \leq m_F(x, y_2) \Rightarrow \mathcal{E}_F(x, y_1) \geq \mathcal{E}_F(x, y_2) & \text{if } opt_F = \max, \\ m_F(x, y_1) \leq m_F(x, y_2) \Rightarrow \mathcal{E}_F(x, y_1) \leq \mathcal{E}_F(x, y_2) & \text{if } opt_F = \min. \end{array}$$

All the measures defined above are cost-respecting.

When analyzing an approximation algorithm the most common way to state the approximability is to give the performance ratio.

Definition 2.7 The *performance ratio* of a feasible solution with respect to the optimum of an NPO problem F is defined as

$$R_F(x, y) = \begin{cases} opt_F(x)/m_F(x, y) & \text{if } opt_F = \max, \\ m_F(x, y)/opt_F(x) & \text{if } opt_F = \min. \end{cases}$$

where $x \in \mathcal{I}_F$ and $y \in S_F(x)$.

The performance ratio and the relative error are obviously related in the following way.

$$R_F(x, y) = \begin{cases} \dfrac{1}{1 - \mathcal{E}_F^r(x, y)} = \displaystyle\sum_{i=0}^{\infty} (\mathcal{E}_F^r(x, y))^i & \text{if } opt_F = \max, \\ 1 + \mathcal{E}_F^r(x, y) & \text{if } opt_F = \min. \end{cases}$$

Definition 2.8 We say that an optimization problem F *can be approximated within* p for a constant p if there exists a polynomial time algorithm A such that for all instances $x \in \mathcal{I}_F$, $A(x) \in S_F(x)$ and $R_F(x, A(x)) \leq p$.

For a few problems it is more interesting to consider the performance ratio just for input instances with large optimal value. This asymptotical behaviour is covered by the following definition.

Definition 2.9 We say that an optimization problem F *can be approximated asymptotically within* p for a constant p if there exists a polynomial time algorithm A and a positive constant N such that for all instances $x \in \mathcal{I}_F$ with $opt_F(x) \geq N$, $A(x) \in S_F(x)$ and $R_F(x, A(x)) \leq p$.

Definition 2.10 The *best performance ratio* $\mathcal{R}[F]$ for an optimization problem F is defined as

$$\mathcal{R}[F] = \inf\{p : F \text{ can be approximated within } p\}.$$

Definition 2.11 The *best asymptotic performance ratio* $\mathcal{R}^\infty[F]$ for an optimization problem F is defined as

$$\mathcal{R}^\infty[F] = \inf\{p : F \text{ can be approximated asymptotically within } p\}.$$

Definition 2.12 We say that an optimization problem F has a *bounded approximation* if there exists a positive constant c such that F can be approximated within c.

Definition 2.8 can be extended to problems which do not have bounded approximation in the following way.

Definition 2.13 We say that an optimization problem F *can be approximated within* $p(n)$ for a function $p : \mathbb{Z}^+ \to \mathbb{R}^+$ if there exists a polynomial time algorithm A such that for every $n \in \mathbb{Z}^+$ and for all instances $x \in \mathcal{I}_F$ with $|x| = n$ we have that $A(x) \in S_F(x)$ and $R_F(x, A(x)) \le p(n)$.

2.4 Approximation schemes

An algorithm which approximates an NPO problem F within 1 by definition finds the optimal solution to the problem in polynomial time. For an NP-complete problem there is no such algorithm (unless P = NP). But for many NP-complete problems there are algorithms which approximate them within $1 + \varepsilon$ for each $\varepsilon > 0$. Such an algorithm is called a *polynomial time approximation scheme* and is defined as follows.

Definition 2.14 We say that an optimization problem F has a *polynomial time approximation scheme* (Ptas) if there is a polynomial time approximation algorithm (in the length of the input instance) that takes as input both an instance $x \in \mathcal{I}_\mathcal{F}$ and a constant $\varepsilon > 0$, and then outputs a solution which approximates F within $1 + \varepsilon$.

An alternative but equivalent definition is that F has a Ptas if there is a polynomial time approximation algorithm that takes as input both an instance $x \in \mathcal{I}_\mathcal{F}$ and a constant $\varepsilon > 0$, and then outputs a solution which approximates F with a relative error less than ε.

The problem of finding a maximum independent set of nodes in a planar graph is an example of a problem which has a Ptas [6]. But the time complexity of this Ptas is unfortunately $O(n \cdot 8^{1/\varepsilon}/\varepsilon)$ where n is the number of nodes in the graph, and thus the time increases very rapidly with decreasing ε. A better behaviour of the time complexity would be a polynomial dependence in $1/\varepsilon$. A Ptas with such a time complexity is called a *fully polynomial time approximation scheme.*

Definition 2.15 We say that an optimization problem F has a *fully polynomial time approximation scheme* (Fptas) if it has an approximation algorithm that takes as input both an instance $x \in \mathcal{I}_\mathcal{F}$ and a constant $\varepsilon > 0$, and then, in time polynomial in both $1/\varepsilon$ and the length of x, outputs a solution which approximates F within $1 + \varepsilon$.

An example of a problem with an Fptas is the knapsack problem [46].

2.5 NPO PB, polynomially bounded NP optimization problems

An important class of NP optimization problems is the *polynomially bounded* problems. These are NPO problems in which the size of the optimal solution is bounded by a polynomial in the length of the corresponding instance. [12, 19, 63]

Definition 2.16 An NPO problem F is *polynomially bounded* if there is a polynomial p such that

$$opt_F(x) \leq p(|x|) \text{ for all } x \in \mathcal{I}_F$$

The class of all polynomially bounded NPO problems is called NPO PB. This class was called OptP[$\log n$] by Krentel [63].

Maximum three-dimensional matching and maximum independent set are examples of problems in NPO PB. The travelling salesperson problem and minimum $0-1$ programming are examples of problems not in NPO PB.

Observe that if there is an NP-complete NPO PB problem which has an Fptas then we can use the Fptas to find the optimal solution in polynomial time. Thus, provided that P $\neq$ NP, no NP-complete NPO PB problem can have an Fptas.

2.6 Why do NP-complete problems have different approximability?

Formally an NP problem must be a decision problem. In order to define an NPO problem as an NP problem one has to introduce a bound on the objective function.

Given an NPO problem $F = (\mathcal{I}_F, S_F, m_F, opt_F)$, the corresponding decision problem $F^d = (\mathcal{I}_F, S_F, m_F, opt_F, K_F)$ is to decide, for an input $x \in \mathcal{I}_F$, whether there is a solution $y \in S_F(x)$ such that $m_F(x, y) \geq K_F$ if $opt_F = \max$ and $m_F(x, y) \leq K_F$ if $opt_F = \min$.

Since all NPO problems can be written as decision problems and since all NP-complete problems are reducible to each other one could suspect that all NPO problems should have the same approximability. For several reasons this is not the case.

Firstly, an NP reduction between two NPO problems F and G (or strictly speaking between the decision problem version of two NPO problems) rarely preserves the objective function.

Secondly, even if the objective function of F transforms into the objective function of G, it might not be in a way that preserves the approximability. Consider for example the problems of finding a maximum size independent set of nodes and a minimum size node cover of an undirected graph $\langle V, E\rangle$ (see the definitions of the problems in Appendix B). A subset of V is independent if and only if its complement is a node cover. Thus, there is a node cover of size

σ if and only if there is an independent set of size $|V| - \sigma$. It is equally hard to find a solution to Max Ind Set with absolute error e and to find a solution to Min Node Cover with absolute error e, but it is much harder to find a solution to Max Ind Set with relative error ε than to find a solution to Min Node Cover with the same relative error.

The nodes of a maximal matching of the graph always form a node cover of at most twice the optimal size, whereas no algorithm approximating the independent set exists, provided that P $\neq$ NP.

Here we have two optimization problems which in a way are the same, but have different approximability due to the definition of relative error. In this specific case we are lucky, because we will get better results if we use the normalized relative error instead of the ordinary relative error, but generally we cannot hope for this.

Let F=Max Ind Set, G=Min Node Cover, $x = \langle V, E\rangle \in \mathcal{I}_F = \mathcal{I}_G$, $y \in S_F(x)$, $\bar{y} = V - y$, $worst_F(\langle V, E\rangle) = 0$ and $worst_G(\langle V, E\rangle) = |V|$. Now the normalized relative errors are

$$\mathcal{E}_F^n(x, y) = \frac{|opt_F(x) - m_F(x, y)|}{|opt_F(x) - worst_F(x)|} = \frac{opt_F(x) - m_F(x, y)}{opt_F(x)}$$

and

$$\mathcal{E}_G^n(x, \bar{y}) = \frac{|opt_G(x) - m_G(x, \bar{y})|}{|opt_G(x) - worst_G(x)|} = \frac{m_G(x, \bar{y}) - opt_G(x)}{worst_G(x) - opt_G(x)} =$$

$$= \frac{(|V| - m_F(x, y)) - (|V| - opt_F(x))}{|V| - (|V| - opt_F(x))} = \frac{opt_F(x) - m_F(x, y)}{opt_F(x)}.$$

Thus $\mathcal{E}_F^n(x, y) = \mathcal{E}_G^n(x, \bar{y})$.

Chapter 3

Reductions

3.1 Reductions preserving approximability

In Section 2.6 we saw that the NP-complete problems do not share the same approximability. If we use an ordinary NP-reduction to reduce one optimization problem to another the approximation properties will in general not be preserved.

In order to be able to speak about completeness of classes of optimization problems with respect to approximation we have to introduce some kind of stronger reduction between such problems. The reduction must preserve approximability in some way.

As we have seen earlier there are many measures of approximability and therefore there are many kinds of reductions. Various authors have proposed several more or less different conditions on what an approximation preserving reduction should look like. Most of them are based on the following idea.

An approximation preserving reduction f from an optimization problem F to an optimization problem G must be able to transform each instance of the F problem to an instance of G. Furthermore, we must be able to transform each solution of this instance of G to a solution of the original instance.

Thus, if we have an approximation algorithm A for G we can use it to get an approximation algorithm for F. Suppose we have an instance x of F. Use the first transformation t_1 to transform x to an instance $t_1(x)$ of G. Then use the approximation algorithm A to get an approximate solution y to $t_1(x)$. Finally transform y to a solution $t_2(y)$ of the original problem. See Figure 3.1 for an illustration of this.

The main point of interest is how good this solution $t_2(y)$ is compared to the optimal solution which has value $opt_F(x)$. This depends obviously both on the algorithm A and the reduction f.

Let us look at the most common approximation preserving reductions. In the rest of the thesis we will mainly use the reencoding, the L-reduction, the P-reduction and the parameter dependent S-reduction. We will start by presenting the strongest reduction, the reduction between two optimization problems which are so similar that they can easily be reencoded into each other.

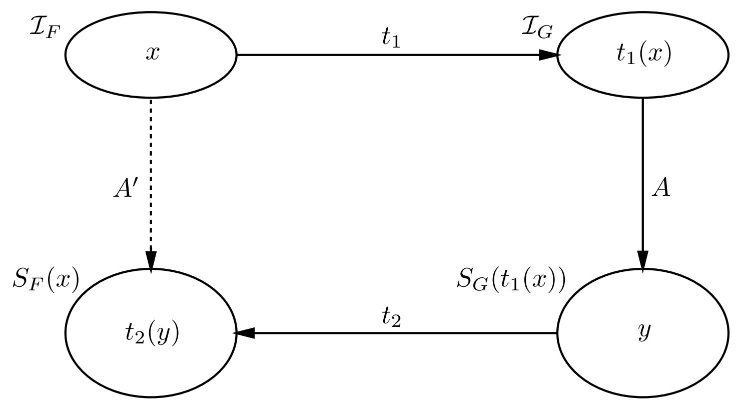
$$\begin{tikzcd} x \arrow[r, "t_1"] \arrow[d, dashed, "A'"'] & t_1(x) \arrow[d, "A"] \\ t_2(y) & y \arrow[l, "t_2"'] \end{tikzcd}$$

Figure 3.1: *The new approximation algorithm A' corresponding to the reduction $f = (t_1, t_2)$ and the approximation algorithm A.*

3.2 Reencoding of optimization problems

What is a legal reencoding of a problem instance? First of all the reencoding must be fast (polynomial time), invertible and the optimum should be preserved. Furthermore a solution (exact or approximate) to an instance should have a unique corresponding solution to the corresponding instance with the same value of the objective function.

Thus an optimal solution to the reencoded problem gives an optimal solution to the original problem with the same value and a solution with relative error ε gives a solution of the original problem with the same relative error. The same result holds for the normalized relative error, the performance ratio and the absolute error.

Definition 3.1 Given two NPO problems F and G (both maximization problems or both minimization problems), a *reencoding of F to G* is a pair $f = (t_1, t_2)$ such that

i) $t_1 : \mathcal{I}_F \to \mathcal{I}_G$ and $\forall x \in \mathcal{I}_F$ and $\forall y \in S_G(t_1(x))$, $t_2(x, y) \in S_F(x)$

ii) t_1 is an invertible function and t_2 is invertible for all fixed $x \in \mathcal{I}_F$. Both t_1, t_2 and their inverses should be polynomial time computable.

iii) $\forall x \in \mathcal{I}_F$ and $\forall y \in S_G(t_1(x))$, $m_F(x, t_2(x, y)) = m_G(t_1(x), y)$.

If there is a reencoding from F to G we write $F \equiv^p G$.

Proposition 3.1 *If $F \equiv^p G$ and $G \equiv^p H$ then $F \equiv^p H$. If $F \equiv^p G$ then $G \equiv^p F$.*

Example 3.1 Let $F =$ Max Clique (the problem of finding the largest set of nodes in a graph where each pair of nodes is connected) and $G =$ Max Ind Set (the problem of finding the largest set of nodes in a graph where no pair of nodes is connected).

Since a clique of size k in a graph corresponds to an independent set of size k in the complement graph and vice versa, $f = (t_1, t_2)$ defined as follows is a reencoding of F to G.

Let t_1 transform an input graph to its complement and let t_2 be the identity function. These functions are obviously invertible and both the functions and their inverses are polynomial time computable.

3.3 Strict reduction

The following reduction, introduced by Orponen and Mannila, is applicable to every measure of approximation. It completely preserves the approximability with respect to the measure of approximation.

Definition 3.2 [83] Given two NPO problems F and G, a *strict reduction* from F to G with respect to a measure $\mathcal{E}$ of approximation quality is a pair $f = (t_1, t_2)$ such that

i) t_1, t_2 are polynomial time computable functions.

ii) $t_1 : \mathcal{I}_F \to \mathcal{I}_G$ and $\forall x \in \mathcal{I}_F$ and $\forall y \in S_G(t_1(x))$, $t_2(x, y) \in S_F(x)$.

iii) $\forall x \in \mathcal{I}_F$ and $\forall y \in S_G(t_1(x))$, $\mathcal{E}_F(x, t_2(x, y)) \le \mathcal{E}_G(t_1(x), y)$.

If there is a strict reduction from F to G we write $F \le_s^p G$ and if at the same time $F \le_s^p G$ and $G \le_s^p F$ we write $F \equiv_s^p G$.

Proposition 3.2 (Orponen and Mannila [83])
Given three NPO *problems F, G and H. If $F \le_s^p G$ and $G \le_s^p H$ then $F \le_s^p H$.*

3.4 Relative error preserving reductions

Reductions which preserve the relative error in some way are the most commonly used. We shall here present five types of reductions which preserve the relative error to different extent. The definitions of the A-reduction, P-reduction, R-reduction and F-reduction are very similar. The reductions are approximation preserving, PTAS-preserving, RPTAS-preserving and FPTAS-preserving respectively, see Chapter 4. The fifth reduction, the L-reduction, is often most practical to use to show that a problem is as hard to approximate as another. If there is an L-reduction between two problems there is also a P-reduction, and if there is a P-reduction between two problems there is also an A-reduction. Unfortunately there is no such relation between the F-reduction and for example the P-reduction as we will see.

3.4.1 A-reduction

Definition 3.3 Given two NPO problems F and G, an *A-reduction* (approximation preserving reduction) from F to G is a triple $f = (t_1, t_2, c)$ such that

i) t_1, t_2 are polynomial time computable functions and $c : Q_G \to Q_F$ where

$$Q_x = \begin{cases} \mathbb{Q}^+ \cap (0,1) & \text{if } opt_x = \max, \\ \mathbb{Q}^+ & \text{if } opt_x = \min. \end{cases}$$

is a computable function.

ii) $t_1 : \mathcal{I}_F \to \mathcal{I}_G$ and $\forall x \in \mathcal{I}_F$ and $\forall y \in S_G(t_1(x))$, $t_2(x,y) \in S_F(x)$.

iii) $\forall x \in \mathcal{I}_F$ and $\forall y \in S_G(t_1(x))$, $\mathcal{E}^r_G(t_1(x),y) \le \varepsilon \ \Rightarrow\ \mathcal{E}^r_F(x,t_2(x,y)) \le c(\varepsilon)$.

If F A-reduces to G we write $F \le^p_A G$ and if at the same time $F \le^p_A G$ and $G \le^p_A F$ we write $F \equiv^p_A G$.

Proposition 3.3 (Orponen and Mannila [83])
Given two NPO *problems F and G, if $F \le^p_A G$ and G has a bounded approximation, then so does F.*

This proposition says that the A-reduction preserves bounded approximation. The A-reduction was called *bounded reduction* by Orponen and Mannila [83].

3.4.2 P-reduction

Definition 3.4 Given two NPO problems F and G, a *P-reduction* (Ptas preserving reduction) from F to G is a triple $f = (t_1, t_2, c)$ such that

i) t_1, t_2 are polynomial time computable functions and $c : Q_F \to Q_G$ is a computable function.

ii) $t_1 : \mathcal{I}_F \to \mathcal{I}_G$ and $\forall x \in \mathcal{I}_F$ and $\forall y \in S_G(t_1(x))$, $t_2(x,y) \in S_F(x)$.

iii) $\forall x \in \mathcal{I}_F$ and $\forall y \in S_G(t_1(x))$, $\mathcal{E}^r_G(t_1(x),y) \le c(\varepsilon) \ \Rightarrow\ \mathcal{E}^r_F(x,t_2(x,y)) \le \varepsilon$.

If F P-reduces to G we write $F \le^p_P G$ and if at the same time $F \le^p_P G$ and $G \le^p_P F$ we write $F \equiv^p_P G$.

Proposition 3.4 (Orponen and Mannila [83])
Given two NPO *problems F and G, if $F \le^p_P G$ and G has a* Ptas*, then so does F.*

Proof $A_F(x,\varepsilon)$ is a Ptas for F if $\forall\varepsilon\forall x \in \mathcal{I}_F, \mathcal{E}^r_F(x, A_F(x,\varepsilon)) \le \varepsilon$. Let $A_F(x,\varepsilon) = t_2(x, A_G(t_1(x), c(\varepsilon)))$. This is a polynomial time algorithm which, by condition iii), approximates x with relative error at most ε. $\square$

The P-reduction has been used by e.g. Crescenzi, Panconesi and Ranjan [24, 84]. The same reduction was called *continuous reduction* by Orponen and Mannila [83].

Proposition 3.5 *A reduction which is a P-reduction is at the same time an A-reduction (with the same functions t_1 and t_2 and another function c). If the c-function in an A- or P-reduction is of the form $c(\varepsilon) = k \cdot \varepsilon$, for some constant k, the reduction is both an A-reduction and a P-reduction. If the c-function in an A- or P-reduction is $c(\varepsilon) = \varepsilon$, the reduction is a strict reduction with respect to the relative error $\mathcal{E}^r$.*

The proof is trivial.

3.4.3 R-reduction

The R-reduction is a seldom used randomized variant of the P-reduction. In this thesis it will only be used in Section 4.11 to show that the maximum independent set is hard to approximate. See Section 4.5 for a definition of Rptas.

Definition 3.5 [12] Given two NPO problems F and G, an *R-reduction* (an Rptas preserving reduction) from F to G is a tuple $f = (t_1, t_2, c, p)$ such that

i) t_1, t_2 are random polynomial time computable functions, p is a real constant between zero and one, and $c : Q_F \to Q_G$ is a computable function.

ii) $t_1 : \mathcal{I}_F \to \mathcal{I}_G$ and $\forall x \in \mathcal{I}_F$ and $\forall y \in S_G(t_1(x))$, $t_2(x, y) \in S_F(x)$.

iii) $\forall x \in \mathcal{I}_F$ and $\forall y \in S_G(t_1(x))$, $\mathcal{E}^r_G(t_1(x), y) \le c(\varepsilon) \;\Rightarrow\; \mathcal{E}^r_F(x, t_2(x, y)) \le \varepsilon$ with probability at least p.

If F R-reduces to G we write $F \le^p_R G$ and if at the same time $F \le^p_R G$ and $G \le^p_R F$ we write $F \equiv^p_R G$.

Thus every P-reduction is also an R-reduction with the probability constant $p = 1$.

Proposition 3.6 *Given two* NPO *problems F and G, if $F \le^p_R G$ and G has an* Rptas*, then so does F.*

The proof is similar to that of Proposition 3.4.

3.4.4 L-reduction

The L-reduction was introduced in 1988 by Papadimitriou and Yannakakis in order to prove completeness in the classes Max NP and Max SNP, see Section 4.10.

Definition 3.6 [88] Given two NPO problems F and G, an *L-reduction* (linear reduction) from F to G is a tuple $f = (t_1, t_2, \alpha, \beta)$ such that

i) t_1, t_2 are polynomial time computable functions and α and β are positive constants.

ii) $t_1 : \mathcal{I}_F \to \mathcal{I}_G$ and $\forall x \in \mathcal{I}_F$ and $\forall y \in S_G(t_1(x))$, $t_2(x, y) \in S_F(x)$.

iii) $opt_G(t_1(x)) \leq \alpha \cdot opt_F(x)$,

iv) $\forall x \in \mathcal{I}_F$ and $\forall y \in S_G(t_1(x))$,

$$|opt_F(x) - m_F(x, t_2(x, y))| \leq \beta \, |opt_G(t_1(x)) - m_G(t_1(x), y)| \, .$$

If F L-reduces to G we write $F \leq^p_L G$ and if at the same time $F \leq^p_L G$ and $G \leq^p_L F$ we write $F \equiv^p_L G$.

Sometimes when mentioning the transformation f we will actually refer to the function t_1.

Proposition 3.7 (Papadimitriou and Yannakakis [88])
If F L-reduces to G with constants α and β and there is a polynomial time approximation algorithm for G with worst-case relative error ε, then there is a polynomial time approximation algorithm for F with worst-case relative error $\alpha\beta\varepsilon$.

Proof $\forall x \in \mathcal{I}_F$ and $\forall y \in S_G(t_1(x))$ we have

$$\mathcal{E}^r_F(x, t_2(x, y)) = \frac{|opt_F(x) - m_F(x, t_2(x, y))|}{opt_F(x)} \leq$$

$$\leq \frac{\beta \, |opt_G(t_1(x)) - m_G(t_1(x), y)|}{opt_G(t_1(x))/\alpha} = \alpha\beta \cdot \mathcal{E}^r_G(t_1(x), y)$$

and $\mathcal{E}^r_G(t_1(x), y) \leq \varepsilon$ implies that $\mathcal{E}^r_F(x, t_2(x, y)) \leq \alpha\beta\varepsilon$. □

Proposition 3.8 *A reduction which is an L-reduction is at the same time both an A-reduction and a P-reduction. An L-reduction with $\alpha\beta = 1$ is a strict reduction with respect to the relative error $\mathcal{E}^r$.*

Proof Follows immediately from Proposition 3.5 and Proposition 3.7. □

It is often easier to show that a reduction is an L-reduction than a P-reduction. Therefore, instead of directly showing that a reduction is a P-reduction, we will often show that it is an L-reduction and use Proposition 3.8 to see that there is a P-reduction.

3.4.5 F-reduction

Definition 3.7 [24] Given two NPO problems F and G, an *F-reduction* (that is an Fptas preserving reduction) from F to G is a triple $f = (t_1, t_2, c)$ such that

i) t_1, t_2 are polynomial time computable functions and $c : Q_F \times \mathcal{I}_F \to Q_G$.

ii) $t_1 : \mathcal{I}_F \to \mathcal{I}_G$ and $\forall x \in \mathcal{I}_F$ and $\forall y \in S_G(t_1(x))$, $t_2(x, y) \in S_F(x)$.

iii) $\forall x \in \mathcal{I}_F$ and $\forall y \in S_G(t_1(x))$,

$$\mathcal{E}^r_G(t_1(x), y) \le c(\varepsilon, x) \;\Rightarrow\; \mathcal{E}^r_F(x, t_2(x, y)) \le \varepsilon.$$

iv) The time complexity of c is $p_1(1/\varepsilon, |x|)$ where p_1 is a polynomial.

v) $\forall x \in \mathcal{I}_F$, $c(\varepsilon, x) \le 1/p_2(1/\varepsilon, |x|)$ where p_2 is a polynomial.

If F F-reduces to G we write $F \le^p_F G$ and if at the same time $F \le^p_F G$ and $G \le^p_F F$ we write $F \equiv^p_F G$.

Proposition 3.9 (Crescenzi and Panconesi [24])
Given two NPO *problems F and G, if $F \le^p_F G$ and G has a* Fptas*, then so does F.*

Proposition 3.10 (Crescenzi and Panconesi [24])
An F-reduction is not definable as a P-reduction with some additional constraint.

The reason for this is that the function c in the definition of the P-reduction must be independent of $|x|$ but in the definition of the F-reduction c can be polynomially dependent on $|x|$. A surprising fact shown by Crescenzi and Panconesi is that every problem which can be approximated within some constant can be F-reduced to a problem which has a Ptas [24]. Thus $F \le^p_F G$ and G has a Ptas $\not\Rightarrow$ F has a Ptas. See Section 4.6.

3.4.6 Transitivity

All relative error preserving reductions defined in this section are transitive.

Proposition 3.11 ([12, 24, 83, 88])
Given three NPO *problems F, G and H.*

$$\begin{array}{l}
\textit{If } F \le^p_A G \textit{ and } G \le^p_A H \textit{ then } F \le^p_A H.\\
\textit{If } F \le^p_P G \textit{ and } G \le^p_P H \textit{ then } F \le^p_P H.\\
\textit{If } F \le^p_R G \textit{ and } G \le^p_R H \textit{ then } F \le^p_R H.\\
\textit{If } F \le^p_L G \textit{ and } G \le^p_L H \textit{ then } F \le^p_L H.\\
\textit{If } F \le^p_F G \textit{ and } G \le^p_F H \textit{ then } F \le^p_F H.
\end{array}$$

3.5 Ratio preserving reduction

In this chapter we have seen a lot of definitions of reduction which preserve the relative error. Now we will switch to another type of reduction, introduced by Simon in 1990, which preserves the performance ratio. Recall from Definition 2.7 that the performance ratio of a solution y to an instance x of a problem F is

$$R_F(x, y) = \begin{cases} opt_F(x)/m_F(x, y) & \text{if } opt_F = \max, \\ m_F(x, y)/opt_F(x) & \text{if } opt_F = \min. \end{cases}$$

where $x \in \mathcal{I}_F$ and $y \in S_F(x)$.

Definition 3.8 Given two NPO problems F and G (either two maximization problems or two minimization problems) and two problem instances $x \in \mathcal{I}_F$ and $x' \in \mathcal{I}_G$, a mapping γ from $S_F(x)$ to $S_G(x')$ is called *a-bounded* if for all $y \in S(x)$

$$\begin{cases} m_G(x', \gamma(y)) \geq \dfrac{1}{a} \cdot m_F(x, y) & \text{if } opt_F = opt_G = \max, \\ m_G(x', \gamma(y)) \leq a \cdot m_F(x, y) & \text{if } opt_F = opt_G = \min. \end{cases}$$

Definition 3.9 Given two NPO problems F and G (either two maximization problems or two minimization problems), a *ratio preserving reduction with expansion* c_2c_3 from F to G is a tuple $f = (t_1, t_2, t_3, c_2, c_3)$ such that

i) t_1, t_2 are polynomial time computable functions.

ii) $t_1 : \mathcal{I}_F \to \mathcal{I}_G$ and $\forall x \in \mathcal{I}_F$ and $\forall y \in S_G(t_1(x))$, $t_2(t_1(x), y) \in S_F(x)$.

iii) $\forall x \in \mathcal{I}_F$ and $\forall y \in S_F(x)$, $t_3(x, y) \in S_G(t_1(x))$.

iv) For each $x \in \mathcal{I}_F$, t_2 is c_2-bounded and t_3 is c_3-bounded.

If f is a ratio preserving reduction with expansion c_2c_3 from F to G we write $F \leq_r^{c_2c_3} G$ and if at the same time $F \leq_r^1 G$ and $G \leq_r^1 F$ we write $F \equiv_r^p G$.

The definition is due to Simon [101], but he calls the reduction an *absolute continuous reduction*, which could be confused with Orponen's and Mannila's continuous reduction, see Section 3.4.2. The following propositions motivate why we have introduced the name *ratio preserving reduction*.

Proposition 3.12 (Simon [101])
If $f = (t_1, t_2, t_3, c_2, c_3)$ is a ratio preserving reduction from F to G, $x \in \mathcal{I}_\mathcal{F}$ and $y \in S_G(t_1(x))$, then $R_F(x, t_2(x, y)) \leq c_2c_3 \cdot R_G(t_1(x), y)$.

Proposition 3.13 (Simon [101])
If $f = (t_1, t_2, t_3, c_2, c_3)$ is a ratio preserving reduction from F to G, then $\mathcal{R}[F] \leq c_2c_3 \cdot \mathcal{R}[G]$.

Thus the product c_2c_3 controls the expansion of the reduction, that is, how much larger the performance ratio will be in the worst case.

Proposition 3.14 (Simon [101])
If F, G and H are either three maximization problems or three minimization problems and if $f = (t_1, t_2, t_3, c_2, c_3)$ is a ratio preserving reduction from F to G and $g = (t_4, t_5, t_6, c_5, c_6)$ is a ratio preserving reduction from G to H, then there is a ratio preserving reduction from F to H with expansion $c_2c_3c_5c_6$.

Proposition 3.15 *A ratio preserving reduction $f = (t_1, t_2, t_3, c_2, c_3)$ from F to G is an A-reduction.*

Proof The functions t_1 and t_2 satisfy the requirements i) and ii).
It only remains to show that given $x \in \mathcal{I}_\mathcal{F}, y \in S_G(t_1(x))$ and $\varepsilon > 0$ there is a function $c(\varepsilon)$ such that $\mathcal{E}^r_G(t_1(x), y) \leq \varepsilon \;\Rightarrow\; \mathcal{E}^r_F(x, t_2(x,y)) \leq c(\varepsilon)$.

We distinguish between the cases of minimization and maximization, and start with supposing that F and G are minimization problems. We know that the performance ratio expansion of f is bounded by a constant $e = c_2c_3$ since f is a ratio preserving reduction.

$$\begin{aligned} R_F(x, t_2(x,y)) &\leq e \cdot R_G(t_1(x), y) \\ \mathcal{E}^r_F(x, t_2(x,y)) + 1 &\leq e \cdot (\mathcal{E}^r_G(t_1(x), y) + 1) \\ \mathcal{E}^r_F(x, t_2(x,y)) &\leq e \cdot \mathcal{E}^r_G(t_1(x), y) + e - 1 \end{aligned}$$

$\mathcal{E}^r_G(t_1(x), y) \leq \varepsilon \Rightarrow \mathcal{E}^r_F(x, t_2(x,y)) \leq e\varepsilon + e - 1$.
Thus $c(\varepsilon) = e\varepsilon + e - 1$.

Now suppose that F and G are maximization problems.

$$\begin{aligned} R_F(x, t_2(x,y)) &\leq e \cdot R_G(t_1(x), y) \\ \frac{1}{1 - \mathcal{E}^r_F(x, t_2(x,y))} &\leq e \cdot \frac{1}{1 - \mathcal{E}^r_G(t_1(x), y)} \\ 1 - \mathcal{E}^r_G(t_1(x), y) &\leq e \cdot (1 - \mathcal{E}^r_F(x, t_2(x,y))) \\ \mathcal{E}^r_F(x, t_2(x,y)) &\leq 1 - \frac{1}{e} + \frac{1}{e}\mathcal{E}^r_G(t_1(x), y) \end{aligned}$$

$\mathcal{E}^r_G(t_1(x), y) \leq \varepsilon \Rightarrow \mathcal{E}^r_F(x, t_2(x,y)) \leq 1 + (\varepsilon - 1)/e$.
Thus $c(\varepsilon) = 1 + (\varepsilon - 1)/e < 1$ when $\varepsilon < 1$. □

Definition 3.10 Given two NPO problems F and G (either two maximization problems or two minimization problems), a *ratio preserving reduction scheme* is a sequence of ratio preserving reductions from F to G with gradually decreasing expansion values, such that for every constant $\varepsilon > 0$ some reduction in the sequence has an expansion less than ε.

Proposition 3.16 *If there is a ratio preserving reduction scheme from F to G and G has a* Ptas*, then F has a* Ptas.

The proof is trivial.

Proposition 3.17 *Given a ratio preserving reduction $f = (t_1, t_2, t_3, c_2, c_3)$ from F to G with $c_2c_3 = 1$. Then t_1 is an L-reduction with $\alpha = c_2$ and $\beta = c_3$ if $\text{opt}_F = \text{opt}_G = \max$ and $\alpha = c_3$ and $\beta = c_2$ if $\text{opt}_F = \text{opt}_G = \min$. Thus we in both cases have a strict reduction with respect to the relative error.*

PROOF Let $x \in \mathcal{I}_{\mathcal{F}}$ be an arbitrary instance of F.
First suppose that F and G are minimization problems. We know that

$$\begin{cases} y \in S_G(t_1(x)) & \Rightarrow \; m_F(x, t_2(t_1(x), y)) \leq c_2 \cdot m_G(t_1(x), y) \; \wedge \\ & \qquad \wedge \; m_G(t_1(x), y) \geq opt_G(t_1(x)) \\ z \in S_F(x) & \Rightarrow \; m_G(t_1(x), t_3(x, z)) \leq c_3 \cdot m_F(x, z) \; \wedge \\ & \qquad \wedge \; m_F(x, z) \geq opt_F(x) \end{cases}$$

As seen in Definition 3.6 the requirements of the L-reduction are

i) $opt_G(t_1(x)) \leq \alpha \cdot opt_F(x)$,

ii) for every solution of $t_1(x)$ with measure k_2 we can in polynomial time find a solution of x with measure k_1 such that

$$|opt_F(x) - k_1| \leq \beta \; |opt_G(t_1(x)) - k_2| .$$

Let $z \in S_F(x)$ be a minimum solution to x. Then

$$opt_F(x) = m_F(x, z) \geq \frac{1}{c_3} \cdot m_G(t_1(x), t_3(x, z)) \geq \frac{1}{c_3} \cdot opt_G(t_1(x)).$$

Thus the first requirement of the L-reduction is satisfied with $\alpha = c_3$.
Since $opt_F(x) \geq c_2 \cdot opt_G(t_1(x))$ and for $y \in S_G(t_1(x))$, $m_F(x, t_2(t_1(x), y)) \leq c_2 \cdot m_G(t_1(x), y)$ we have that

$$|opt_F(x) - m_F(x, t_2(t_1(x), y))| = m_F(x, t_2(t_1(x), y)) - opt_F(x) \leq$$

$$\leq c_2 \cdot m_G(t_1(x), y) - c_2 \cdot opt_G(t_1(x)) = c_2 \; |opt_G(t_1(x)) - m_G(t_1(x), y)|$$

and thus the second requirement of the L-reduction is satisfied with $\beta = c_2$.

Now suppose that F and G are maximization problems. We know that

$$\begin{cases} y \in S_G(t_1(x)) & \Rightarrow \; m_F(x, t_2(t_1(x), y)) \geq \frac{1}{c_2} \cdot m_G(t_1(x), y) \; \wedge \\ & \qquad \wedge \; m_G(t_1(x), y) \leq opt_G(t_1(x)) \\ z \in S_F(x) & \Rightarrow \; m_G(t_1(x), t_3(x, z)) \geq \frac{1}{c_3} \cdot m_F(x, z) \; \wedge \\ & \qquad \wedge \; m_F(x, z) \leq opt_F(x) \end{cases}$$

Let $y \in S_G(t_1(x))$ be a maximum solution to $t_1(x)$. Then

$$opt_G(t_1(x)) = m_G(t_1(x), y) \leq c_2 \cdot m_F(x, t_2(t_1(x), y)) \leq c_2 \cdot opt_F(x).$$

Thus the first requirement of the L-reduction is satisfied with $\alpha = c_2$.
Let $z \in S_F(x)$ be a maximum solution to x. Then

$$opt_F(x) = m_F(x, z) \leq c_3 \cdot m_G(t_1(x), t_3(x, z)) \leq c_3 \cdot opt_G(t_1(x)).$$

Let $y \in S_G(t_1(x))$.

$$m_F(x, t_2(t_1(x), y)) \geq \frac{1}{c_2} \cdot m_G(t_1(x), y) = c_3 \cdot m_G(t_1(x), y)$$

The two last inequalities imply that

$$opt_F(x) - m_F(x, t_2(t_1(x), y)) \leq c_3\left(opt_G(t_1(x)) - m_G(t_1(x), y)\right)$$

and thus the second requirement of the L-reduction is satisfied with $\beta = c_3$. □

3.6 Parameter dependent reductions

The relative error preserving reductions, like the L-reduction and the P-reduction, work well when reducing to problems with bounded approximation, but there are several problems which cannot be approximated within a constant, unless P = NP. When analyzing approximation algorithms for such problems one usually specifies the approximability using a one variable function where the parameter concerns the size of the input instance. For example the maximum independent set problem can be approximated within $O\left(n/(\log n)^2\right)$ where the parameter n is the number of nodes in the input graph. Which quantity of the input instance to choose as the parameter depends on the problem and the algorithm. In the example above the most relevant quantity turned out to be the number of nodes.

When reducing between two such problems, say from F to G, the relative error preserving reductions are not perfect. The trouble is that these reductions may transform an input instance of F to a much larger input instance of G. One purpose of a reduction is to be able to use an approximation algorithm for G to construct an equally good (within a constant) approximation algorithm for F. Because of the size amplification the constructed algorithm will not be as good as the original algorithm.

As an example (from Section 6.2.2) we take Max Ind Set as G, some other problem on graphs as F and transform the input graph to a graph where each node corresponds to a pair of nodes in the input graph. Thus, if the input graph of F contains n nodes, the input graph of G will contain $O(n^2)$ nodes, so the above mentioned approximation algorithm of Max Ind Set will only give us an algorithm approximating F within

$$O\left(\frac{n^2}{(\log n^2)^2}\right) = O\left(\frac{n^2}{(\log n)^2}\right).$$

In order to tell how the approximability, when given as a function, will be changed by a reduction, we have to specify how the size of the input instance will be amplified. The situation is complicated by not knowing in what parameter the input instances will be measured. For graphs both the number of nodes and the number of edges may be possible choices.

For every reduction mentioned in earlier sections in this chapter we may add a statement *with size amplification* $f(n)$ in order to specify this. If the size amplification is $O(n)$, i.e. if the size of the constructed structure is a constant times the size of the original structure, we say that the reduction is *without amplification.* Moreover we introduce a completely new size dependent reduction which we think is well suited for reductions between problems which cannot be approximated within a constant.

Definition 3.11 Given two NPO problems F and G, an *S-reduction with size amplification* $a(n)$ from F to G is a tuple $f = (t_1, t_2, a, c)$ such that

i) t_1, t_2 are polynomial time computable functions, a is a monotonously increasing positive function and c is a positive constant.

ii) $t_1 : \mathcal{I}_F \to \mathcal{I}_G$ and $\forall x \in \mathcal{I}_F$ and $\forall y \in S_G(t_1(x))$, $t_2(x, y) \in S_F(x)$.

iii) $\forall x \in \mathcal{I}_F$ and $\forall y \in S_G(t_1(x))$, $R_F(x, t_2(x, y)) \leq c \cdot R_G(t_1(x), y)$.

iv) $\forall x \in \mathcal{I}_F, |t_1(x)| \leq a(|x|)$.

Proposition 3.18 *Given two* NPO *problems F and G, if $F \leq_S^p G$ with size amplification $a(n)$ and G can be approximated within some monotonously increasing positive function $u(n)$ of the size of the input instance, then F can be approximated within $c \cdot u(a(n))$, which is a monotonously increasing positive function.*

Proof For each $x \in \mathcal{I}_F$ of size n we use the approximation function for G to find a solution $y \in S_G(t_1(x))$ so that

$$R_F(x, t_2(x, y)) \leq c \cdot R_G(t_1(x), y) \leq c \cdot u(|t_1(x)|) \leq c \cdot u(a(|x|)) = c \cdot u(a(n))$$

because u is a monotonously increasing positive function. □

For constant and polylogarithmic approximable problems the S-reduction preserves approximability within a constant for any polynomial size amplification, since $c \log^k(n^p) = p^k c \log^k n = O(\log^k n)$. For n^c approximable problems the S-reduction preserves approximability within a constant just for size amplification $O(n)$, since $c \cdot (O(n))^c = O(n^c)$.

3.7 Non-constructive reductions

Two of the first reductions between optimization problems to be introduced were the *non-constructive ratio preserving reduction* and the *non-constructive measure preserving reduction.* They were defined in 1979 by Paz and Moran. The reductions are called non-constructive because they don't tell how to find a solution, just what its objective function is.

Definition 3.12 [90] Given two NPO problems F and G (either two maximization problems or two minimization problems), a *non-constructive ratio preserving reduction* from F to G is a polynomial time computable function $g : \mathcal{I}_F \to \mathcal{I}_G$ together with two constants c_1 and c_2 such that $0 < c_1 \leq c_2$ and $\forall x \in \mathcal{I}_F, c_1\, opt_F(x) \leq opt_G(g(x)) \leq c_2\, opt_F(x)$.

Definition 3.13 [90] Given two NPO problems F and G (either two maximization problems or two minimization problems), a *non-constructive measure preserving reduction* from F to G is a polynomial time computable function $g : \mathcal{I}_F \rightarrow \mathcal{I}_G$ such that $\forall x \in \mathcal{I}_F, opt_F(x) = opt_G(g(x))$

We can see that the non-constructive measure preserving reduction is a non-constructive ratio preserving reduction with $c_1 = c_2 = 1$.

Suppose that A' is a non-constructive approximation algorithm for G, that is A' is a polynomial time computable function $A' : \mathcal{I}_G \rightarrow \mathbb{N}$ such that $\forall x' \in \mathcal{I}_G, A'(x') \leq opt_G(x')$ if $opt_G = \max$ and $A'(x') \geq opt_G(x')$ if $opt_G = \min$. If g is a non-constructive ratio preserving reduction from F to G, then a non-constructive approximation algorithm A for F is an algorithm which for an input $x \in \mathcal{I}_F$ computes

$$\left\lfloor \frac{A'(g(x))}{c_1} \right\rfloor .$$

Proposition 3.19 (Paz and Moran [90])
Given a non-constructive ratio preserving reduction g from F to G and a non-constructive approximation algorithm A' for G. If A' approximates G within a constant, then A (defined as above) approximates F within a constant. If $c_1 = c_2 = 1$ (i.e. g is a non-constructive measure preserving reduction) and A' approximates G within any constant, then A approximates F within any constant.

Thus the non-constructive ratio preserving reduction is a non-constructive variant of the A-reduction, and the non-constructive measure preserving reduction is a non-constructive variant of the P-reduction.

3.8 Structure preserving reduction

The last reduction in this chapter, the *structure preserving reduction*, differs from all the other reductions defined here because it preserves much more information about the instances but it is not immediately applicable to approximation. The reduction was invented by Ausiello, D'Atri and Protasi in 1977.

Definition 3.14 [4] Given an NPO problem F and an input instance $x \in \mathcal{I_F}$, the *structure* of x is the list $struct_F(x) = \langle a_0, \ldots, a_c \rangle$ where
$c = |opt_F(x) - worst_F(x)|$, $a_i = |\{y \in S_F(x) : i = |m_F(x, y) - worst_F(x)|\}|$.

Example 3.2 [4] Let F be the minimum set cover problem (given a number of sets, find the minimum number of sets covering all elements) and $x_1 = \{S_1, S_2, S_3, S_4, S_5\}$ where $S_1 = \{1, 2, 3, 4\}$, $S_2 = \{1, 2, 5\}$, $S_3 = \{3, 4, 6\}$, $S_4 = \{5, 6, 7\}$, $S_5 = \{1, 2, 3, 4, 5, 6, 7\}$. Now we have $worst_F(x_1) = 5$, $opt_F(x_1) = 1$, $struct_F(x_1) = \langle 1, 5, 9, 5, 1 \rangle$.

Example 3.3 [4] Let F be the minimum set cover problem and define x_2 as the family of sets $\{S_1, S_2, S_3, S_4\}$ where the sets are defined as in the previous example. In this case we have $worst_F(x_2) = 4$, $opt_F(x_2) = 2$, $struct_F(x_2) = \langle 1, 3, 1 \rangle$.

Definition 3.15 [4] Given two NPO problems F and G, a *structure preserving reduction* from F to G is a pair $f = (t_1, t_2)$ such that

i) t_1 and t_2 are polynomial time computable functions such that $t_1 : \mathcal{I}_F \to \mathcal{I}_G$ and $t_2 : \mathcal{I}_F \times \mathbb{N} \to \mathbb{N}$.

ii) $\forall x \in \mathcal{I}_F$ and $\forall k \in \mathbb{N}$

$$(\exists y \in S_F(x) : m_F(x, y) = k) \Leftrightarrow$$
$$\Leftrightarrow (\exists y' \in S_G(t_1(x)) : m_G(t_1(x), y') = t_2(x, k))$$

iii) $\forall x \in \mathcal{I}_F, struct_F(x) = struct_G(t_1(x))$

If there is a structure preserving reduction from F to G we write $F \leq^p_{sp} G$ and if at the same time $F \leq^p_{sp} G$ and $G \leq^p_{sp} F$ we write $F \equiv^p_{sp} G$.

Definition 3.16 An NPO problem F is said to be *convex* if for every $x \in \mathcal{I_F}$ and for every integer k between $opt_F(x)$ and $worst_F(x)$ there is at least one solution $y \in S_F(x)$ such that $m_F(y) = k$.

Definition 3.17 [4] Given two NPO problems F and G, a *parsimonious reduction* from F to G is a pair $f = (t_1, t_2)$ such that

i) t_1 and t_2 are polynomial time computable functions such that $t_1 : \mathcal{I}_F \to \mathcal{I}_G$ and $t_2 : \mathcal{I}_F \times \mathbb{N} \to \mathbb{N}$.

ii) $\forall x \in \mathcal{I}_F$ and $\forall k \in \mathbb{N}$

$$|\{y \in S_F(x) : m_F(x, y) = k\}| =$$
$$= |\{y' \in S_G(t_1(x)) : m_G(t_1(x), y') = t_2(x, k)\}|$$

Definition 3.18 [4] Given two convex NPO problems F and G, a structure preserving reduction $f = (t_1, t_2)$ from F to G is *strictly monotonous* if for every $x \in \mathcal{I_F}$ and for every pair of integers k_1 and k_2 between $opt_F(x)$ and $worst_F(x)$

$$k_1 < k_2 \Rightarrow t_2(x, k_1) < t_2(x, k_2).$$

Proposition 3.20 (Ausiello, D'Atri and Protasi [4])
Given two convex NPO *problems F and G and a parsimonious reduction $f = (t_1, t_2)$ from F to G such that*

i) there is a function $a : \mathcal{I_F} \to \mathbb{Z}$ such that for every $x \in \mathcal{I_F}$ and for every integer k between $\text{opt}_F(x)$ *and* $\text{worst}_F(x)$

$$t_2(x, k) = \begin{cases} a(x) + k & \textit{if } \text{opt}_F = \text{opt}_G, \\ a(x) - k & \textit{if } \text{opt}_F \neq \text{opt}_G. \end{cases}$$

ii) $\forall x \in \mathcal{I_F}, t_2(x, \text{worst}_F(x)) = \text{worst}_G(t_1(x))$

Then f is structure preserving and strictly monotonous.

Proposition 3.21 (Ausiello, D'Atri and Protasi [4])
Given two convex NPO *problems F and G and a structure preserving and strictly monotonous reduction $f = (t_1, t_2)$ from F to G, then f is parsimonious and satisfies the conditions of Proposition 3.20.*

Proposition 3.22 (Ausiello, D'Atri and Protasi [4])
Given two convex NPO *problems F and G, a structure preserving reduction $f = (t_1, t_2)$ from F to G and a structure preserving reduction $g = (u_1, u_2)$ from G to F such that*

i) *both f and g are strictly monotonous,*

ii) *there is a function $a : \mathcal{I_F} \to \mathbb{Z}$ such that for every $x \in \mathcal{I_F}$ and for every integer k between $\mathrm{opt}_F(x)$ and $\mathrm{worst}_F(x)$*

$$t_2(x,k) = \begin{cases} a(x) + k & \text{if } \mathrm{opt}_F = \mathrm{opt}_G, \\ a(x) - k & \text{if } \mathrm{opt}_F \neq \mathrm{opt}_G. \end{cases}$$

iii) *there is a function $b : \mathcal{I_G} \to \mathbb{Z}$ such that for every $x' \in \mathcal{I_G}$ and for every integer k' between $\mathrm{opt}_G(x')$ and $\mathrm{worst}_G(x')$*

$$u_2(x',k') = \begin{cases} b(x') + k' & \text{if } \mathrm{opt}_F = \mathrm{opt}_G, \\ b(x') - k' & \text{if } \mathrm{opt}_F \neq \mathrm{opt}_G. \end{cases}$$

iv) $\forall x \in \mathcal{I_F}$

$$\begin{cases} a(x) \geq -b(t_1(x)) & \text{if } \mathrm{opt}_F = \mathrm{opt}_G, \\ a(x) \leq b(t_1(x)) & \text{if } \mathrm{opt}_F \neq \mathrm{opt}_G, \end{cases}$$

then there is a non-constructive strict reduction from F to G with respect to the normalized relative error.

Note that conditions ii) and iii) are already satisfied since f and g are structure preserving and strictly monotonous due to Proposition 3.21.

Chapter 4

Approximability classes

4.1 Introduction

In order to sort out how different NPO problems behave regarding approximability we introduce approximability classes, that is classes of problems which we can approximate equally well. For example we have one class of problems which can be approximated within a constant and another class of problems which can be approximated within any constant.

As usual in complexity we would like to find the *hardest* problems in each class, problems which have the property that every problem in the class can be reduced to them.

Definition 4.1 Given an NPO problem F, a class C and an approximation preserving reduction $\leq$.

We say that F *is* C*-complete under the reduction* $\leq$ if

1. $F \in C$,
2. for all $G \in C$, $G \leq F$.

We say that F *is* C*-hard under the reduction* $\leq$ if for all $G \in C$, $G \leq F$.

The P-reduction is the most common reduction in this context, so whenever we mention completeness or hardness without specifying the type of reduction the P-reduction is implicit.

It is also possible to define classes of problems without using approximability behaviour explicitly. Much work has been done on classes defined by logical formulas. Such a class contains all NPO problems which can be formulated in the same way, for example all problems whose objective function is the size of a set which can be specified using a first-order formula with only one existential quantifier. For some of these classes every contained problem can be approximated in a uniform way.

Often it is useful to look at the closure (under some reduction) of a class defined by logical formulas. The closure is the set of problems which can be reduced to a problem in the original class.

Definition 4.2 Given an approximability class C and an approximation preserving reduction $\leq$.

The *closure of* C *under the reduction* $\leq$ is

$$\overline{C} = \{F \in \mathrm{NPO} : \exists G \in C : F \leq G\}.$$

In the following sections the most common approximability classes are defined and their characteristics are presented.

4.2 Fptas

An Fptas or a fully polynomial time approximation scheme was defined in Definition 2.15 as a polynomial approximation scheme with a time complexity bounded by a polynomial in $1/\varepsilon$. Let us define the class Fptas of optimization problems which can be approximated by a fully polynomial time approximation scheme. For simplicity the class Fptas has the same name as the approximation scheme Fptas. This should not create any confusion.

Definition 4.3

$$\text{Fptas} = \{F \in \mathrm{NPO} : F \text{ has an Fptas}\}$$

The property of having an Fptas is closely related to the property, introduced by Garey and Johnson, of having a pseudo-polynomial time algorithm. A pseudo-polynomial time algorithm is an algorithm whose time complexity is bounded by a polynomial in both the size of the input and the magnitude of the largest number in the input.

Definition 4.4 [35] Given an NPO problem F. Let $\mathit{maxint}_F(x)$ denote the magnitude of the largest integer occurring in x where $x \in \mathcal{I}_F$, or 0 if no integers occur in x.

Definition 4.5 [35] Given an NPO problem F. An algorithm that solves F is called a *pseudo-polynomial time algorithm* if its time complexity is bounded from above by a polynomial function of the two variables $\mathit{maxint}_F(x)$ and $|x|$ where $x \in \mathcal{I}_F$.

Proposition 4.1 (Garey and Johnson [34])
Given an NPO *problem F such that for each $x \in \mathcal{I}_F$, $\mathit{opt}_F(x)$ is bounded from above by a polynomial in both $\mathit{maxint}_F(x)$ and $|x|$. If $F \in$* Fptas *then it can be solved by a pseudo-polynomial time algorithm.*

This is also related to the concept of strong NP-completeness.

Definition 4.6 [35] A decision problem $F \in \mathrm{NP}$ is NP-*complete in the strong sense* if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ for which the problem, when restricting to the instances x that satisfy

$$\mathit{maxint}_F(x) \leq p(|x|),$$

is NP-complete.

Many of the problems mentioned in *Computers and Intractability: a guide to the theory of NP-completeness* by Garey and Johnson are NP-complete in the strong sense [34].

Proposition 4.2 (Garey and Johnson [34])
Given an NPO *problem F. If the decision problem version of F is* NP*-complete in the strong sense, then F cannot be solved by a pseudo-polynomial time algorithm unless* $\mathrm{P} = \mathrm{NP}$.

Thus, given an NPO problem F such that for each $x \in \mathcal{I}_F$, $opt_F(x)$ is bounded from above by a polynomial in both $maxint_F(x)$ and $|x|$; if the decision problem version of F is NP-complete in the strong sense, then $F \notin$ FPTAS unless $\mathrm{P} = \mathrm{NP}$.

Paz and Moran have completely characterized the problems in FPTAS using a property called p-simpleness. The connections between p-simpleness and NP-completeness in the strong sense are discussed in [5].

Definition 4.7 [90] An NPO problem F is *p-simple* if there is some polynomial $q : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that for each $k \in \mathbb{N}$ the set

$$\{x \in \mathcal{I}_F : opt_F(x) \leq k\}$$

is recognizable in time $q(|x|, k)$.

Proposition 4.3 (Paz and Moran [90])
Given an NPO *problem F.*

1. *If F is p-simple and for all $x \in \mathcal{I}_F$, $opt_F(x)$ is bounded from above by a polynomial in $|x|$, then F is polynomially solvable, i.e. $F \in$* PO.
2. *If F is p-simple and for all $x \in \mathcal{I}_F$, $opt_F(x)$ is bounded from above by a polynomial in $\mathrm{maxint}_F(x)$ and $|x|$, then F can be solved by a pseudo-polynomial time algorithm.*
3. *If F can be solved by a pseudo-polynomial time algorithm and for all instances $x \in \mathcal{I}_F$, $\mathrm{maxint}_F(x)$ is bounded from above by a polynomial in $|x|$ and $opt_F(x)$, then F is p-simple.*

Proposition 4.4 (Paz and Moran [90])
Given an NPO *problem F. $F \in$* FPTAS *if and only if the following conditions hold.*

1. *F is p-simple.*
2. *There is a function $b : \mathcal{I}_F \times \mathbb{Z}^+ \to \mathbb{N}$ and a polynomial $q : \mathbb{N} \to \mathbb{N}$ such that for each instance $x \in \mathcal{I}_F$ and each integer $h > 0$*

$$0 \leq \frac{opt_F(x)}{h} - b(x,h) \leq q(|x|) \quad \text{if } opt_F = \max,$$
$$0 \leq b(x,h) - \frac{opt_F(x)}{h} \leq q(|x|) \quad \text{if } opt_F = \min$$

where b has time complexity bounded by a polynomial in $|x|$ and $\dfrac{opt_F(x)}{h}$.

4.3 Ptas

A Ptas or a polynomial time approximation scheme is an algorithm that approximates an NPO problem within $1+\varepsilon$ for each $\varepsilon > 0$. It was defined formally in Definition 2.14. In the same way as in the preceding section we define the class Ptas as the optimization problems which can be approximated by a polynomial time approximation scheme.

Definition 4.8

$$\text{Ptas} = \{F \in \text{NPO} : F \text{ has a Ptas}\}$$

Apparently Fptas $\subseteq$ Ptas. There is a characterization of Ptas by Paz and Moran which is similar to the characterization of Fptas.

Definition 4.9 [90] An NPO problem F is *simple* if for each $k \in \mathbb{N}$ the set

$$\{x \in \mathcal{I}_F : opt_F(x) \le k\}$$

is recognizable in time polynomial in $|x|$ (and arbitrary in k).

Proposition 4.5 (Paz and Moran [90])
Given an NPO *problem F. F* $\in$ Ptas *if and only if the following conditions hold.*

1. *F is simple.*
2. *There is a function $b : \mathcal{I}_F \times \mathbb{Z}^+ \to \mathbb{N}$ and a constant $Q \in \mathbb{N}$ such that for each instance $x \in \mathcal{I}_F$ and each integer $h > 0$*

$$0 \le \frac{opt_F(x)}{h} - b(x,h) \le Q \quad \text{if } opt_F = \max,$$
$$0 \le b(x,h) - \frac{opt_F(x)}{h} \le Q \quad \text{if } opt_F = \min$$

where b has time complexity bounded by a polynomial in both $|x|$ and $opt_F(x)/h$.

There exist problems which are Ptas-complete, that is problems which are in Ptas and every problem in Ptas is reducible to them. Still no natural optimization problem has been shown to be Ptas-complete.

Proposition 4.6 (Crescenzi and Panconesi [24])
The problem Max Weighted Sat with Small Bound *is* Ptas*-complete under F-reductions.*

Maximum weighted satisfiability with small bound is a variant of the weighted maximum satisfiability problem. Given a boolean formula F over the variables U, non-negative integer weights $w(u)$ on the variables and a non-negative integer bound B such that

$$\sum_{u \in U} w(u) \le \left(1 + \frac{1}{|U|-1}\right) B.$$

The problem is to find a truth assignment $t : U \to \{\text{TRUE}, \text{FALSE}\}$ that maximizes the function which takes the value B when F is not satisfied and

$$\sum_{u:t(u)=\text{TRUE}} w(u)$$

otherwise.

4.4 Fptas$^\infty$ and Ptas$^\infty$

Suppose we are given a set of items with integer sizes and an unbounded number of bins of some fixed size c. We would like to pack all the items in as few bins as possible. This is the bin packing problem, which has been rigorously studied under many years, see [22] for a survey.

It is easy to show (using the NP-complete problem Partition) that it is NP-complete to decide whether two bins are enough. Therefore Min Bin Packing cannot be approximated better than within 3/2, supposing that P $\neq$ NP, and certainly Min Bin Packing $\notin$ Ptas.

However Karmarkar and Karp have shown that there is an algorithm which approximates Min Bin Packing within $1+\varepsilon$ in time polynomial in $1/\varepsilon$ where $\varepsilon = O(\log^2(opt(\langle U, c, s\rangle))/opt(\langle U, c, s\rangle))$ [56]. Thus there almost is an Fptas for the problem. The best asymptotic performance ratio for the problem is one and it has asymptotically an Fptas. We say that Min Bin Packing $\in$ Fptas$^\infty$.

Definition 4.10 Fptas$^\infty$ is the set of NPO problems F such that there exists an approximation algorithm that takes as input both an instance $x \in \mathcal{I}_\mathcal{F}$ and a constant $\varepsilon > 0$, and then, in time polynomial in both $1/\varepsilon$ and the length of x, outputs a solution which approximates F asymptotically within $1 + \varepsilon$.

In the same way we can define an asymptotical version of Ptas.

Definition 4.11

$$\text{Ptas}^\infty = \{F \in \text{NPO} : \mathcal{R}^\infty[\text{F}] = 1\}$$

Several problems on planar graphs can be placed in Ptas$^\infty$ using Lipton's and Tarjan's planar separator theorem, see Section 8.2.

4.5 Rptas

The Ptas class can be extended somewhat by introducing randomization. This leads to the class Rptas consisting of the problems which can be approximated by a randomized polynomial approximation scheme, that is a scheme which for each $\varepsilon > 0$ approximates a problem within $1+\varepsilon$ with some constant probability.

Definition 4.12

$$\text{Rptas} = \{F \in \text{NPO} : F \text{ has a randomized Ptas}\}$$

Apparently Ptas ⊆ Rptas. This class is used very little in the literature. The most interesting result is by Berman and Schnitger [12] and says that if any Max SNP-complete problem (see Section 4.10) is not in Rptas, then there is a constant $c > 0$ such that the maximum independent set problem cannot be approximated within n^c, where n is the number of nodes in the input graph, see Section 4.11.

4.6 Apx

The next class is very important: the class of problems which have bounded approximation, that is which can be approximated within a constant. Here are some famous examples of such problems.

1. Maximum satisfiability (Max Sat): given a set of disjunctive clauses, find the truth assignment that satisfies the largest number of clauses. Max Sat can be approximated within the constant 4/3 [110].
2. Maximum three-set packing (Max 3SP): given a collection of sets of size three, find the largest number of pairwise disjunctive sets. Max 3SP can be approximated within the constant 3 [52].
3. Euclidean travelling salesperson problem (Min ETsp): given a set of points in the plane, find the shortest tour which visits all the points. Min ETsp can be approximated within 3/2 [21].

Definition 4.13

$$\text{Apx} = \{F \in \text{NPO} : \exists c \geq 1 \text{ such that } F \text{ can be approximated within } c\}$$

Equivalently we can define Apx as the union of the set of minimization problems which can be approximated within a constant relative error and the set of maximization problems which can be approximated within a constant relative error less than 1.

Obviously Ptas ⊆ Apx.

Proposition 4.7 (Crescenzi and Panconesi [24])
The problem Max Weighted Sat with Bound *is* Apx*-complete under P-reductions.*

Max Weighted Sat with Bound is the same problem as Max Weighted Sat with Small Bound but with the more generous condition on B that

$$\sum_{u \in U} w(u) \leq 2\,B.$$

A startling result by Crescenzi and Panconesi is that every problem in Apx can be F-reduced to a problem which is Ptas-complete under F-reductions [24], in other terms: Apx ⊆ $\overline{\text{Ptas}}$ under F-reductions. The reason for this is that the F-reduction is not a P-reduction with some additional constraint, see Proposition 3.10.

4.7 NPO

The class NPO is the largest class that we will treat in this thesis. It contains every optimization problem which can be solved in nondeterministic polynomial time. If P = NP every NPO problem can be solved in polynomial time and we have no need for polynomial approximations.

We have that PO $\subseteq$ Fptas $\subseteq$ Ptas $\subseteq$ Apx $\subseteq$ NPO.

Proposition 4.8 (Orponen and Mannila [83])
The following problems are NPO*-complete under strict reductions with respect to any cost-respecting measure of approximation quality. For example they are* NPO*-complete under P-reductions and A-reductions.*

- *Minimum weighted 3-satisfiability* (Min Weighted 3Sat)*: given a boolean formula F in 3CNF with non-negative integer weights $w(u)$ on the variables; find a truth assignment $t : U \to \{\text{TRUE}, \text{FALSE}\}$ that satisfies F and minimizes*
$$\sum_{u:t(u)=\text{TRUE}} w(u).$$
- *Travelling salesperson problem* (Min Tsp)*: given a complete graph G with non-negative integer weights $s(e)$ on the edges; find a tour C in G that minimizes*
$$\sum_{e\in C} s(e).$$
- *Minimum $0-1$ programming* (Min $0-1$ Programming)*: given an integer $m \times n$-matrix A, integer m-vector b and a non-negative integer n-vector c; find a zero-one n-vector x that satisfies $Ax \geq b$ and minimizes $c^T x$.*

4.8 NPO problems with polynomially bounded optimum

NPO PB is the class of problems in NPO whose solution is bounded from above by a polynomial in the size of the input, see Definition 2.16.

Definition 4.14 For every class $C \subseteq$ NPO we define C PB to be the problems in C which have polynomially bounded optimum, that is

$$C \text{ PB} = C \cap \text{NPO PB}.$$

It is obvious that PO PB $\subseteq$ Fptas PB $\subseteq$ Ptas PB $\subseteq$ Apx PB $\subseteq$ NPO PB.

Proposition 4.9 Fptas PB $\subseteq$ PO PB.

Proof $F \in$ Fptas PB $\Rightarrow \exists$ algorithm $A(x, \varepsilon)$, polynomial $p(|x|, 1/\varepsilon)$ such that the time complexity of A is $p(|x|, 1/\varepsilon)$, A approximates F within $1+\varepsilon$ and there is a polynomial $q(|x|)$ such that $opt_F(x) \leq q(|x|)$.

Compute $A\left(x, \dfrac{1}{q(|x|)+1}\right)$ which approximates F within $1+1/(q(|x|)+1)$. The time for this is $p(|x|, q(|x|)+1)$, i.e. polynomial time.

Let $s = m_F(x, A(x, 1/(q(|x|)+1)))$. First suppose that $opt_F = \max$. Then $0 < s \leq opt_F(x) \leq q(|x|)$.

$$\frac{opt_F(x)}{s} \leq 1 + \frac{1}{q(|x|)+1} \Leftrightarrow s \geq opt_F(x) - \frac{s}{q(|x|)+1} > opt_F(x) - 1$$

Thus $opt_F(x) - 1 < s \leq opt_F(x) \Rightarrow s = opt_F(x)$ because s is an integer.

If $opt_F = \min$ then $0 < opt_F(x) \leq s$ and $opt_F(x) \leq q(|x|)$.

$$\frac{s}{opt_F(x)} \leq 1 + \frac{1}{q(|x|)+1} \Leftrightarrow s \leq opt_F(x) + \frac{opt_F(x)}{q(|x|)+1} < opt_F(x) + 1$$

Thus $opt_F(x) \leq s < opt_F(x) + 1 \Rightarrow s = opt_F(x)$.

In both cases we found an optimal solution in polynomial time. □

Proposition 4.10 (Berman and Schnitger [12])
For every NPO PB*-complete problem there is an* $\varepsilon > 0$ *such that the problem cannot be approximated within* $O(n^\varepsilon)$*, unless* P = NP.

Observe that the result above applies to NPO-complete problems as well.

Proposition 4.11 (Berman and Schnitger [12])
The following problems are NPO PB*-complete under P-reductions.*

- *Longest induced path in a graph: given a graph G, find the largest subset of nodes for which the induced subgraph is a simple path.*
- *Longest path with forbidden pairs: given a graph G and a collection P of pairs of nodes; find the longest simple path in G containing at most one node from each pair in P.*
- *Maximum bounded* $0-1$ *programming: given an integer $m \times n$-matrix A, an integer m-vector b and a zero-one integer n-vector c; find a zero-one n-vector x that satisfies $Ax \leq b$ and maximizes $c^T x$.*

There are more examples of NPO PB-complete problems in Chapter 8.

The class NPO PB is the same as the class Krentel called OptP$[\log n]$ [63]. He also defined completeness in this class, but under a reduction which does not preserve approximability. Therefore the OptP$[\log n]$-complete problems under his reduction are different from the NPO PB-complete problems under P-reductions.

4.9 The existence of intermediate degrees

Are there problems between APX and the NPO-complete problems, that is problems which are not in APX and are not NPO-complete? The following proposition answers this question affirmative.

Proposition 4.12 (Crescenzi and Panconesi [24])
If P $\neq$ NP *there is an* NPO *problem F such that $F \notin$* APX*, F is not* NPO-*complete and F is* APX*-hard under P-reductions.*

Proposition 4.13 (Crescenzi and Panconesi [24])
If P $\neq$ NP *there is an* NPO *problem F such that $F \notin$* APX*, F is not* NPO-*complete and F is not* APX*-hard under P-reductions.*

Thus there are intuitively strange problems which are strong enough to be not approximable within a constant, but not strong enough to represent all problems which are approximable within a constant.

4.10 Classes defined by logical formulas

An alternative, logical characterization of NP was made by Fagin in 1974 [29]. It is interesting since it is the only definition which does not involve computation. As an example we shall show that SAT $\in$ NP using this characterization.

Example 4.1 Given a number of disjunctive clauses the SAT problem is to determine if there is a truth assignment that satisfies all clauses. Let the input be given as two predicates P and N where $P(x, c) = \text{TRUE}$ iff variable x appears positive in clause c and $N(x, c) = \text{TRUE}$ iff variable x appears negated in clause c. The clauses can be satisfied if and only if

$$\exists T \, \forall c \exists x (P(x, c) \wedge x \in T) \vee (N(x, c) \wedge x \notin T).$$

T shall be interpreted as the set of true variables.

In general, a language L is in NP if and only if there is a quantifier free first-order formula Φ_L such that

$$I \in L \Leftrightarrow \exists S \, \forall \overline{x} \exists \overline{y} \Phi_L(I, S, \overline{x}, \overline{y})$$

where I is a finite structure describing the instance, S is a finite structure, $\overline{x}$ and $\overline{y}$ are finite vectors.

This formalism was used by Papadimitriou and Yannakakis in 1988 to define two classes of NPO problems: MAX NP (maximization NP) and MAX SNP (maximization strict NP) [88].

Later Panconesi and Ranjan introduced the class MAX Π_1 [84] and recently Kolaitis and Thakur completed the picture with four class hierarchies MAX Π_i, MAX Σ_i, MIN Π_i and MIN Σ_i [61, 60]. All these classes will be presented later in this section.

The definitions of MAX NP and MAX SNP have been interpreted differently by different authors. In this theses the notations $\overline{\text{MAX NP}}$ and $\overline{\text{MAX SNP}}$

will be used for the classes as Papadimitriou and Yannakakis intended them to be, and SYNTACTIC MAX NP and SYNTACTIC MAX SNP will be used for the plainly syntactically defined classes used by for example Kolaitis and Thakur [61, 60] and by Panconesi and Ranjan [84]. SYNTACTIC MAX SNP and SYNTACTIC MAX NP are included in the hierarchies as MAX Σ_0 and MAX Σ_1 respectively.

Only some of these syntactically defined classes seem to capture approximation properties. However, the value of classifying problems using logical definability can be discussed because the same problem may or may not be included in a class depending on how it is encoded. Therefore we, in Section 4.10.5, propose new definitions of MAX SNP and MAX NP where we allow different encodings.

4.10.1 SYNTACTIC MAX NP

Definition 4.15 SYNTACTIC MAX NP is the class of NPO problems F which can be written in the form

$$opt_F(I) = \max_S |\{\overline{x} : \exists \overline{y}\ \Phi_F(I, S, \overline{x}, \overline{y})\}|$$

where Φ_F is a quantifier free formula, I an instance of F described as a finite structure and S a solution described as a finite structure.

Example 4.2 Show that the problem MAX SAT $\in$ SYNTACTIC MAX NP.

MAX SAT is the problem of, given a boolean formula in CNF, finding a truth assignment that satisfies the maximum number of clauses. Let the instance $I = \langle P, N \rangle$ where $P(x, c)$ is true when the variable x appears positive in clause c and $N(x, c)$ is true when x appears negated in c. Let S be the set of true variables. Now the optimum value can be written

$$opt(\langle P, N \rangle) = \max_T \left| \left\{ c : \exists x \left(P(x, c) \wedge x \in T \right) \vee \left(N(x, c) \wedge x \notin T \right) \right\} \right|$$

Compare this with how we showed that SAT $\in$ NP in Example 4.1. A main reason for introducing SYNTACTIC MAX NP was the following result.

Proposition 4.14 (Papadimitriou and Yannakakis [88])
Every problem in SYNTACTIC MAX NP *can be approximated within a constant, that is* SYNTACTIC MAX NP $\subseteq$ APX.

Proposition 4.15 (Kolaitis and Thakur [61])
The size of the optimum solution of any problem in SYNTACTIC MAX NP *is bounded above by a polynomial, that is* SYNTACTIC MAX NP $\subseteq$ NPO PB.

Proposition 4.14 and 4.15 thus say that SYNTACTIC MAX NP $\subseteq$ APX PB.

Proposition 4.16 (Papadimitriou and Yannakakis [88])
Every problem in SYNTACTIC MAX NP *can be P-reduced to maximum generalized k-satisfiability* (MAX G kSAT) *for some k.* MAX G kSAT *is similar to*

Max Sat *but each clause is a disjunction of conjunctions, where each conjunction contains at most k literals.* Max G kSat $\in$ Syntactic Max NP *for every $k \in \mathbb{Z}^+$. Thus there is a k such that* Max G kSat *is* Syntactic Max NP*-complete.*

Proposition 4.17 (Crescenzi, Fiorini and Silvestri [23])
Every problem in Syntactic Max NP *can be reduced to* Max Clique *using a strict reduction with respect to $\mathcal{E}^r$. Thus* Max Clique *is* Syntactic Max NP*-hard.*

Max Clique is Syntactic Max NP-hard but not Syntactic Max NP-complete since one can show that the problem is not contained in Syntactic Max NP.

Theorem 4.18 (Panconesi and Ranjan [84])

a) Max Clique *is not in* Syntactic Max NP.

b) Max 3SP *(maximum three-set packing) is not in* Syntactic Max NP.

c) Max 3DM *(maximum three dimensional matching) is not in* Syntactic Max NP.

d) Max 2DM *(maximum bipartite matching) is not in* Syntactic Max NP.

Proof The proofs are similar. Here follows Panconesi's and Ranjan's proof of c).

Assume to the contrary that Max 3DM $\in$ Syntactic Max NP. Then there exists a formula φ such that for all instances I of the problem

$$opt_{3DM}(I) = \max_S |\{\overline{x} \mid \exists \overline{y}\ \varphi(I, S, \overline{x}, \overline{y})\}|\,.$$

Consider an instance $I_1 = \{T_1, \ldots, T_n\}$ such that $opt_{3DM}(I_1) = n$. Given $T_i = (a_i, b_i, c_i)$ and $T_j = (a_j, b_j, c_j)$, we say that they are *compatible* if $a_i \neq a_j \ \wedge\ b_i \neq b_j \ \wedge\ c_i \neq c_j$. I_1 is a set of n mutually compatible 3-tuples.

From our contradictory assumption we have that there is S_1 such that

$$opt_{3DM}(I_1) \ = \ |\{\overline{x} \mid \exists \overline{y}\ \varphi(I_1, S_1, \overline{x}, \overline{y})\}| = n.$$

Let $\overline{x}_1, \ldots, \overline{x}_n$ be the tuples satisfying the above formula. Take $\overline{x}_1$ and suppose, without loss of generality, that it contains a_1, i.e. $\overline{x}_1 = (a_1, u_2, \ldots, u_k)$.

We now construct another instance I_2 by simply replacing a_1 with a brand new element a_0. Let $I_2 = \{T_0, T_2, \ldots, T_n\}$ where $T_0 = (a_0, b_1, c_1)$. I_2 is made of the same tuples as I_1 except the first, T_0. T_0 and T_1 only differ for the first component. We choose a_0 so that I_2 is made of n mutually compatible tuples. Now define S_2 to be the same set as S_1 provided any occurrence of a_1 is replaced by an occurrence of a_0, and define $\overline{z}_i$ to be the same tuple as $\overline{x}_i$ provided the same substitution takes place. Then $|\{\overline{z} \mid \exists \overline{y}\ \varphi(I_2, S_2, \overline{z}, \overline{y})\}| = n$.

If we now consider the new instance $I_{new} = I_1 \cup I_2$ and define $S_{new} = S_1 \cup S_2$, we have that $opt_{3DM}(I_{new}) = n$ but

$$\max_S |\{\overline{w} \mid \exists \overline{y}\ \varphi(I_{new}, S, \overline{w}, \overline{y})\}| \ \geq\ |\{\overline{w} \mid \exists \overline{y}\ \varphi(I_{new}, S_{new}, \overline{w}, \overline{y})\}| \ \geq\ n+1$$

because $|\{\overline{x}_1, \ldots, \overline{x}_n\} \ \cup\ \{\overline{z}_1, \ldots, \overline{z}_n\}| \geq n+1$.

Here we have used the general principle

$$\mathcal{A} \models \exists x \phi(x) \wedge \mathcal{A} \subseteq \mathcal{B} \Rightarrow \mathcal{B} \models \exists x \phi(x)$$

where $\phi(x)$ is a quantifier-free first-order formula and $\mathcal{A} \subseteq \mathcal{B}$ means that $\mathcal{A}$ is a submodel of $\mathcal{B}$. □

The same proof may be applied to the corresponding bounded problems as well, though this was not noticed by the original authors.

Theorem 4.19

a) Max Clique $-B$ *(Maximum clique in a graph with bounded degree) is not in* Syntactic Max NP.

b) Max 3SP $-B$ *(maximum bounded three-set packing) is not in* Syntactic Max NP.

c) Max 3DM $-B$ *(maximum bounded three dimensional matching) is not in* Syntactic Max NP.

Corollary 4.20 Syntactic Max NP *does not contain* PO PB.

Proof The problems Max Clique $-B$ and Max 2DM are not in Syntactic Max NP but are in PO PB. □

Thus Syntactic Max NP is a class that contains optimization problems which can be defined by the same type of formulas and can be approximated within a constant, but it is not entirely satisfactory that there are problems which can be solved in polynomial time which are not included in the class. Later we will see that the closure of Syntactic Max NP under P-reductions does contain PO PB.

4.10.2 Syntactic Max SNP

Papadimitriou and Yannakakis also defined a subclass Syntactic Max SNP (strict NP) of Syntactic Max NP using the same definition without the existence quantifier.

Definition 4.16 Syntactic Max SNP is the class of NPO problems F which can be written in the form

$$opt_F(I) = \max_S |\{\overline{x} : \Phi_F(I, S, \overline{x})\}|$$

where Φ_F is a quantifier free formula, I an instance of F described as a finite structure and S a solution described as a finite structure.

Example 4.3 Show that the problem MAX 3SAT, which is the same problem as MAX SAT with at most three literals in each clause, is in SYNTACTIC MAX SNP. Suppose that each clause is unique.

Encode the input instance as four relations C_0, C_1, C_2, C_3 where C_i contains all clauses with exactly i negative literals. Let $c = (x_1, x_2, x_3) \in C_i$ mean that the i first variables $x_1, \ldots, x_i$ occur negated in clause c and the $3 - i$ remaining variables occur positive in clause c. If a clause contains less than three literals we duplicate the last variable in the encoding to fill up the three places in the clause. Now MAX 3SAT can be defined as follows.

$$\begin{aligned} opt(\langle C_0, C_1, C_2, C_3 \rangle) = \max_T | (x_1, x_2, x_3) : & \\ & ((x_1, x_2, x_3) \in C_0 \wedge (x_1 \in T \vee x_2 \in T \vee x_3 \in T)) \vee \\ \vee \quad & ((x_1, x_2, x_3) \in C_1 \wedge (x_1 \notin T \vee x_2 \in T \vee x_3 \in T)) \vee \\ \vee \quad & ((x_1, x_2, x_3) \in C_2 \wedge (x_1 \notin T \vee x_2 \notin T \vee x_3 \in T)) \vee \\ \vee \quad & ((x_1, x_2, x_3) \in C_3 \wedge (x_1 \notin T \vee x_2 \notin T \vee x_3 \notin T)) \ | \end{aligned}$$

If we allow several copies of the same clause in the input the above definition will not work, but we can make it work by making the 3-ary relations C_i $(0 \leq i \leq 3)$ 4-ary by including a clause number. For example the clause $c_{17} = x_2 \vee \overline{x}_4 \vee x_{11}$ is encoded by adding $(x_4, x_2, x_{11}, c_{17})$ to relation C_1.

Example 4.4 Show that the problem MAX IND SET $-B$, which is the same problem as MAX IND SET on a graph of degree at most B, is in SYNTACTIC MAX SNP.

We encode the graph as a $(B + 1)$-ary relation A. There is one tuple $(u, v_1, \ldots, v_B)$ for every node u listing its neighbours $v_1, \ldots, v_B$. If u has less than B neighbours, just repeat the last one to fill up the tuple. Now we can define MAX IND SET $-B$ as follows, where S shall be interpreted as a set of independent nodes.

$$opt(A) = \max_S |\{(u, v_1, \ldots, v_B) \in A : u \in S \wedge v_1 \notin S \wedge \ldots \wedge v_B \notin S\}|$$

Since SYNTACTIC MAX SNP $\subseteq$ SYNTACTIC MAX NP we know that every problem in SYNTACTIC MAX SNP can be approximated within a constant. Several natural optimization problems are contained in SYNTACTIC MAX SNP and have been shown to be complete in the class with respect to P-reductions.

Proposition 4.21 (Papadimitriou and Yannakakis [88])
MAX 3SAT, MAX 3SAT $-B$ $(B \geq 6)$ *and* MAX IND SET $-B$ $(B \geq 7)$ *are* SYNTACTIC MAX SNP*-complete under P-reductions.* MAX 3SAT $-B$ *is the same problem as* MAX 3SAT *where the total number of occurrences of each variable is bounded by the constant B.* MAX IND SET $-B$ *is the maximum independent set problem on a graph with degree bounded by B.*

Many more problems were shown to be SYNTACTIC MAX SNP-complete in [88] and further more problems will be shown to be complete in the following chapters of this thesis. In Chapter 8 there is a list of all problems currently known to be MAX SNP-complete.

4.10.3 Closures of Max NP and Max SNP

Definition 4.17 [88] $\overline{\text{Max NP}}$ is the closure under P-reductions of Syntactic Max NP, that is

$$\{F \in \text{NPO} : \exists G \in \text{Syntactic Max NP} \; \exists \text{ P-reduction } f : F \leq_P^p G\}$$

Max NP was originally defined by Papadimitriou and Yannakakis [88], but their definition has been interpreted both as Definition 4.17 (which was their intention [86]) and as Definition 4.15 [84].

Definition 4.18 [88] $\overline{\text{Max SNP}}$ is the closure under P-reductions of Syntactic Max SNP, that is

$$\{F \in \text{NPO} : \exists G \in \text{Syntactic Max SNP} \; \exists \text{ P-reduction } f : F \leq_P^p G\}$$

Not only maximization problems, but also some minimization problems can be placed in $\overline{\text{Max NP}}$ and $\overline{\text{Max SNP}}$ with these definitions.

Example 4.5 Show that Min Node Cover $-B$ (minimum node cover in a graph with bounded degree) is in $\overline{\text{Max SNP}}$.

Let us (as in Section 2.6) use that given a graph without free nodes (nodes which do not have any neighbours), a set of nodes is a node cover if and only if its complement is an independent set. We observe that free nodes are trivially included in any maximal independent set. We would like to find a P-reduction from Min Node Cover $-B$ to Max Ind Set $-B$, which we in Example 4.4 showed is included in Syntactic Max SNP.

Often it is more convenient to show that a reduction is an L-reduction. Recall from Proposition 3.8 that an L-reduction satisfies the conditions of a P-reduction. Let F be Min Node Cover $-B$, G be Max Ind Set $-B$ and $f : \mathcal{I}_F \to \mathcal{I}_G$ be the identity function which takes a graph and returns it immediately. This function is certainly polynomial time computable.

1. Show that there is a constant α such that for every instance $x \in \mathcal{I}_F$

   $$opt_G(f(x)) \leq \alpha \cdot opt_F(x),$$

   i.e. that the size of the maximum independent set is bounded above by a constant times the size of the minimum node cover of a graph.

   Since every node in a maximum independent set (apart from the free nodes) must be connected to at least one node in the node cover and since the degree of the graph is bounded by B, the size of the maximum independent set cannot be larger that B times the size of the minimum node cover. Thus $\alpha = B$.

2. Show that there is a constant β such that for every instance $x \in \mathcal{I}_F$ and for every solution of $f(x)$ with measure c_2 we can in polynomial time find a solution of x with measure c_1 such that $|opt_F(x) - c_1| \leq \beta \, |opt_G(f(x)) - c_2|$.

Let S be a solution to G, that is an independent set in the graph, whose size is k less than the size of the maximum independent set. The complement set of S is a node cover whose size is at most k more than the size of the minimum node cover. Thus the second condition is satisfied with $\beta = 1$.

An obvious shortcoming of the classes Syntactic Max NP and Syntactic Max SNP is that they do not contain PO PB, the polynomially bounded optimization problems solvable in polynomial time. On the other hand one can show that PO PB is contained in $\overline{\text{Max SNP}}$.

Theorem 4.22 PO PB $\subseteq \overline{\text{Max SNP}}$.

Proof Given a maximization problem $F \in$ PO PB. Define a P-reduction $f = (t_1, t_2)$ from F to an optimization problem G in the following way. Since $F \in$ PO PB there is a polynomial time function $g : \mathcal{I}_F \to S_F(\mathcal{I}_F)$ which computes the optimum solution with the maximum value $m_F(I, g(I)) = opt_F(I)$. Let G be the simple maximization problem of counting: given a set A, give a (maximum) lower bound of the number of elements in A. Let $t_1(I)$ be the function that computes $m_F(I, g(I))$ and creates a set with this number of elements. Let $t_2(I, y) = g(I)$. t_1 and t_2 are polynomial time computable functions since g is polynomial and since $m_F(I, g(I))$ is polynomially bounded. For every y we have

$$\mathcal{E}_F^r(x, t_2(x, y)) = \mathcal{E}_F^r(x, g(x)) = \frac{|opt_F(x) - m_F(x, g(x))|}{opt_F(x)} = 0 \le \mathcal{E}_G^r(t_1(x), y)$$

Now we can define G as a Syntactic Max SNP problem with the input set A:

$$opt_G(A) = \max |\{x \in A\}|$$

□

Proposition 4.23 *Every problem in* $\overline{\text{Max NP}}$*,* Syntactic Max SNP *and* $\overline{\text{Max SNP}}$ *can be approximated within a constant, that is* $\overline{\text{Max NP}} \subseteq$ Apx*,* Syntactic Max SNP $\subseteq$ Apx*,* $\overline{\text{Max SNP}} \subseteq$ Apx*.*

Proof This follows immediately from Proposition 4.14, the fact that Syntactic Max SNP $\subseteq$ Syntactic Max NP and the definitions of $\overline{\text{Max NP}}$ and $\overline{\text{Max SNP}}$. □

Proposition 4.24 (Papadimitriou and Yannakakis [88])
If there exists a $\overline{\text{Max SNP}}$*-complete problem which has a* Ptas*, then every problem in* $\overline{\text{Max SNP}}$ *has a* Ptas*.*

Since it seemed very unlikely for a $\overline{\text{Max SNP}}$-complete problem to have a Ptas it was generally assumed that $\overline{\text{Max SNP}} \not\subseteq$ Ptas. Recently Arora, Lund, Motwani, Sudan and Szegedy confirmed this assumption.

Proposition 4.25 (Arora, Lund, Motwani, Sudan and Szegedy [2])
No $\overline{\text{Max SNP}}$*-complete problem can have a* Ptas *unless* P = NP*.*

4.10.4 Relations between the classes Syntactic Max SNP and $\overline{\text{Max SNP}}$

In Chapter 5 we will show that Max 3DM $-B$ and Max 3SP $-B$ are in $\overline{\text{Max SNP}}$ by reducing them to Max Ind Set $-B$. Since Theorem 4.19 says that Max 3DM $-B$ and Max 3SP $-B$ are not in Syntactic Max NP, the following theorem may be a bit surprising.

Theorem 4.26

a) Max 3DM $-B$ *is in* Syntactic Max SNP *with a different encoding.*

b) Max 3SP $-B$ *is in* Syntactic Max SNP *with a different encoding.*

Proof The proofs of a and b are similar. We just present the proof of b. Suppose that the Max 3SP $-B$ instance is given by specifying for every set s its neighbour sets, i.e. the sets which have at least one element in common with s. $B+1$ sets of sets $\#_0, \ldots, \#_B$ are used to specify the number of neighbour sets for every set.

$$s \in \#_k \Leftrightarrow s \text{ has exactly } k \text{ neighbour sets.}$$

Since every set has at most B neighbours we have B neighbour relations $N_1, \ldots, N_B$:

$$(s, s') \in N_i \Leftrightarrow s' \text{ is the } i\text{-th neighbour set of } s.$$

If s has $k < B$ neighbour sets, we let $(s, s) \in N_i$ for all $k < i \leq B$.

Now we can define the problem.

$$opt_{3SP-B}(I) = \max_S \left|\{\overline{x} \in S^{B+1} : \Phi(\overline{x}, I, S)\}\right|$$

where $\overline{x} = (s, s_1, \ldots, s_B)$, $I = (N_1, \ldots, N_B, \#_0, \ldots, \#_B)$ and $\Phi(\overline{x}, I, S) = s \in S \wedge$

$$\begin{aligned}
\wedge\ & \Big(s \in \#_B \Rightarrow s_1 \notin S \wedge \cdots \wedge s_B \notin S \wedge (s, s_1) \in N_1 \wedge \cdots \wedge (s, s_B) \in N_B\Big) \wedge \\
\wedge\ & \Big(s \in \#_{B-1} \Rightarrow s_1 \notin S \wedge \cdots \wedge s_{B-1} \notin S \wedge (s, s_1) \in N_1 \wedge \cdots \wedge \\
& \qquad\qquad \wedge (s, s_{B-1}) \in N_{B-1} \wedge s_B = s\Big) \wedge \\
& \vdots \\
\wedge\ & \Big(s \in \#_0 \Rightarrow s_1 = s \wedge s_2 = s \wedge \cdots \wedge s_B = s\Big).
\end{aligned}$$

□

Thus the problem Max 3SP $-B$ is not in Syntactic Max NP and therefore not in the smaller class Syntactic Max SNP with the usual encoding, but it is in Syntactic Max SNP with a different encoding. The same thing applies to Max 3DM too. It is not sound that the encoding of the problem influences the complexity class in such a brutal way. Some amount of reencoding of the problem before placing it in a class would be desirable.

4.10.5 New definitions of Max SNP and Max NP

We hereby define new versions of Max NP and Max SNP using the reencoding defined in Definition 3.1.

Definition 4.19 Max NP is the problems which can be reencoded to some problem in Syntactic Max NP, that is

$$\{F \in \text{NPO} : \exists G \in \text{Syntactic Max NP } \exists \text{ reencoding } F \equiv^p G\}$$

Definition 4.20 Max SNP is the problems which can be reencoded to some problem in Syntactic Max SNP, that is

$$\{F \in \text{NPO} : \exists G \in \text{Syntactic Max SNP } \exists \text{ reencoding } F \equiv^p G\}$$

We observe that the reencoding is a form of P-reduction and thus these new classes lie between the strictly syntactically defined classes and the closure under P-reductions. We have the following situation.

$$\begin{array}{ccccccc}
\text{Syntactic Max NP} & \subset & \text{Max NP} & \subseteq & \overline{\text{Max NP}} & \subseteq & \text{Apx} \\
\bigcup\!\mid & & \bigcup\!\mid & & \bigcup\!\mid & & \\
\text{Syntactic Max SNP} & \subset & \text{Max SNP} & \subseteq & \overline{\text{Max SNP}} & &
\end{array}$$

$$\text{Syntactic Max NP} \not\ni \text{Max 3DM } -B \in \text{Max SNP}$$
$$\text{Syntactic Max NP} \not\ni \text{Max 3SP } -B \in \text{Max SNP}$$

It is not known whether or not PO PB is contained in Max NP or Max SNP with the new definitions.

It seems to be harder to show that a problem is not in Max NP with the new definition than to show that it is not in Syntactic Max NP, but it might not be impossible in view of the restricted reencoding used.

The following proposition clears the relations between completeness in these classes.

Proposition 4.27 *A* Syntactic Max NP-*complete problem is always* Max NP-*complete and a* Max NP-*complete problem is always* $\overline{\text{Max NP}}$-*complete. A* $\overline{\text{Max NP}}$-*complete problem is* Max NP-*complete if it is in* Max NP *and a* Max NP-*complete problem is* Syntactic Max NP-*complete if it is in* Syntactic Max NP.

A Syntactic Max SNP-*complete problem is always* Max SNP-*complete and a* Max SNP-*complete problem is always* $\overline{\text{Max SNP}}$-*complete.*
A $\overline{\text{Max SNP}}$-*complete problem is* Max SNP-*complete if it is in* Max SNP *and a* Max SNP-*complete problem is* Syntactic Max SNP-*complete if it is in* Syntactic Max SNP.

Proof We will only prove one part of the proposition. The same type reasoning can be used to prove the rest.

An X-complete problem is, by Definition 4.1, a problem which is in X and to which every problem in X can be P-reduced.

Suppose we have a problem F and that every problem in Syntactic Max NP can be P-reduced to F. Since $\overline{\text{Max NP}}$ consists of the problems which can be P-reduced to some problem in SYNTACTIC MAX NP we know that every problem in $\overline{\text{MAX NP}}$ can be P-reduced to F. Thus F is $\overline{\text{MAX NP}}$-complete if it is in $\overline{\text{MAX NP}}$, and if F is in SYNTACTIC MAX NP we know that it is in $\overline{\text{MAX NP}}$. Thus a SYNTACTIC MAX NP-complete problem is always $\overline{\text{MAX NP}}$-complete.

A $\overline{\text{MAX NP}}$-hard problem is also SYNTACTIC MAX NP-hard because SYNTACTIC MAX NP $\subset$ $\overline{\text{MAX NP}}$. This means that a $\overline{\text{MAX NP}}$-complete problem is SYNTACTIC MAX NP-complete if it is in SYNTACTIC MAX NP. □

4.10.6 The hierarchies MAX Π_i, MAX Σ_i, MIN Π_i and MIN Σ_i

In analogy with the polynomial time hierarchy [103] with the notation Π_i^p and Σ_i^p Kolaitis and Thakur have introduced hierarchies MAX Π_i and MAX Σ_i of maximization problems and MIN Π_i and MIN Σ_i of minimization problems.

Definition 4.21 [61] Consider the set of maximization problems which can be written in a form such that for every problem F and instance $I \in \mathcal{I}_F$

$$opt_F(I) = \max_S |\{\overline{x} : \Phi_F(I, S, \overline{x})\}|$$

where Φ_F is a first-order formula in prenex normal form. We say that F is in the class MAX $\Sigma_i, i \in \mathbb{N}$ if Φ_F has i alternations of quantifiers and starts with a block of *existential* quantifiers and that F is in the class MAX $\Pi_i, i \in \mathbb{N}$ if Φ_F has i alternations of quantifiers and starts with a block of *universal* quantifiers.

Similarly we define MIN Σ_i and MIN $\Pi_i, i \in \mathbb{N}$ as the classes which contain the minimization problems which can be written in a form such that for every problem F and instance $I \in \mathcal{I}_F$

$$opt_F(I) = \min_S |\{\overline{x} : \Phi_F(I, S, \overline{x})\}|$$

where Φ_F is a first-order formula in prenex normal form having i alternations of quantifiers.

From the definition follows that MAX Σ_0 = MAX Π_0 = SYNTACTIC MAX SNP and MAX Σ_1 = SYNTACTIC MAX NP.

Definition 4.22 [61] Let MAX PB be the maximization problems which have polynomially bounded optimum value, that is

$$\text{MAX PB} = \{F \in \text{NPO PB} : opt_F = \max\}$$

and let MIN PB be the minimization problems which have polynomially bounded optimum value, that is

$$\text{MIN PB} = \{F \in \text{NPO PB} : opt_F = \min\}.$$

Thus NPO PB = Max PB ∪ Min PB. The hierarchy of maximization problems collapses at Max Π_2 and has four distinct levels. The hierarchy of minimization problems collapses at Min Π_1 and has three distinct levels.

Proposition 4.28 (Kolaitis and Thakur [61])

$$\text{Max }\Sigma_0 \subset \text{Max }\Sigma_1 \subset \text{Max }\Pi_1 = \text{Max }\Sigma_2 \subset \text{Max }\Pi_2$$

$$i \geq 2 \Rightarrow \text{Max }\Pi_i = \text{Max PB},\ i \geq 3 \Rightarrow \text{Max }\Sigma_i = \text{Max PB}$$

$$\text{Max }\Sigma_0 \not\ni \text{Max Sat} \in \text{Max }\Sigma_1$$
$$\text{Max }\Sigma_1 \not\ni \text{Max Clique} \in \text{Max }\Pi_1$$
$$\text{Max }\Pi_1 \not\ni \text{Max Connected Component} \in \text{Max }\Pi_2$$

Max Connected Component is the problem of finding the largest connected component in a given graph. Since Max Connected Component ∈ PO PB we also have that PO PB $\not\subseteq$ Max Π_1.

Example 4.6 Max Σ_1 $\not\ni$ Max Clique ∈ Max Π_1

Theorem 4.19 says that Max Clique ∉ Max Σ_1. We show that Max Clique ∈ Max Π_1 by giving its optimization formula $opt(\langle V, E\rangle)$:

$$\max_{V'} |\{v \in V' : \forall v_1 \forall v_2 (v_1 \in V' \wedge v_2 \in V' \wedge v_1 \neq v_2) \Rightarrow (v_1, v_2) \in E\}|$$

For minimization problems the hierarchy is even shorter.

Proposition 4.29 (Kolaitis and Thakur [61])

$$\text{Min }\Sigma_0 = \text{Min }\Sigma_1 \subset \text{Min }\Pi_1 = \text{Min }\Sigma_2$$

$$i \geq 1 \Rightarrow \text{Min }\Pi_i = \text{Min PB},\ i \geq 2 \Rightarrow \text{Min }\Sigma_i = \text{Min PB}$$

$$\text{Min }\Sigma_1 \not\ni \text{Min Graph Colouring} \in \text{Min }\Pi_1$$

Here, Min Graph Colouring is the problem of finding the minimum number of colours which can colour the nodes of a graph in such a way that no edge connects two nodes with the same colour.

Every problem in Max Σ_0 and Max Σ_1 can be approximated within a constant. We could suppose that this would be true for Min Σ_0 and Min Σ_1 as well, but this is not the case, as the following proposition shows.

Proposition 4.30 (Kolaitis and Thakur [61])
Min 3DNF Sat *cannot be approximated within a constant unless* P = NP.
Min 3DNF Sat *is* Min Σ_0*-complete under P-reductions.*

The problem minimum 3DNF satisfiability (Min 3DNF Sat) takes as input a boolean formula in 3DNF (disjunctive normal form with at most three literals in each conjunction) and finds the minimum number of satisfiable conjunctions.

Unlike the case of maximization problems, the pattern of the quantifier prefix does not have any influence on the approximability of minimization problems.

4.10.7 Closures of these classes

In Section 4.10.3 we studied closures of MAX Σ_0 and MAX Σ_1. We shall show that $\overline{\text{MAX } \Pi_1}$, the closure of MAX Π_1, is in fact the whole of NPO PB by showing that the maximum number of satisfiable formulas problem is in MAX Π_1 and is NPO PB-hard.

The maximum number of satisfiable formulas problem or MAX # SAT takes as input a set $\Psi = \{\psi_1, \ldots, \psi_n\}$ of 3CNF formulas. The problem is to find a truth assignment to the included variables such that the maximum number of the formulas are satisfied.

Proposition 4.31 (Panconesi and Ranjan [84])
MAX # SAT *is in* MAX Π_1.

Theorem 4.32 MAX # SAT *is* NPO PB-*complete.*

PROOF It is obvious that MAX # SAT is an NPO problem and that the optimum value is bounded by the number of formulas, i.e. polynomially bounded in the size of the input. (This also follows from the fact that MAX # SAT $\in$ MAX Π_1.)

We show that MAX # SAT is NPO PB-hard by indicating how an L-reduction may be obtained from the longest induced path problem, which is NPO PB-complete (see Proposition 4.11).

The idea is to construct a 3CNF formula for each edge in the input graph. The i-th edge gives birth to the formula $e_i \wedge C$ where e_i is a boolean variable indicating if the i-th edge is in the solution, and C is a very big formula which is true if and only if the solution specified by the variables is a valid solution of the problem. Hereby the number of satisfied formulas will be equal to the number of edges in the solution if the solution is legal and zero otherwise.

It is not very hard to see that such a reduction is an L-reduction with $\alpha = \beta = 1$. $\square$

Theorem 4.33 MIN 3DNF SAT *is* $\overline{\text{MAX SNP}}$-*hard.*

PROOF Let us show that there is an L-reduction from MAX 3SAT to MIN 3DNF SAT.

Let f be a function which takes a set of clauses and returns an equally large set of conjunctions by negating each clause. For example the clause $(x_1 \vee \overline{x}_2 \vee \overline{x}_3)$ transforms into $(\overline{x}_1 \wedge x_2 \wedge x_3)$.

If I is an instance of MAX 3SAT with n clauses and optimum value x then $f(I)$ is an instance of MIN 3DNF SAT with n conjunctions and optimum value $n - x$. Each solution of $f(I)$ with value v immediately gives a solution of I with value $n - v$.

Since we always can satisfy at least half of the clauses in a MAX 3SAT problem we see that $opt(f(I)) \leq n/2 \leq opt(I)$.

Thus f gives an L-reduction with $\alpha = \beta = 1$. $\square$

$$\begin{array}{ccccccccc}
\textsc{Max}\ \Sigma_0 & \subset & \textsc{Max}\ \Sigma_1 & \subset & \textsc{Max}\ \Pi_1 & \subset & \textsc{Max}\ \Pi_2 & = & \textsc{Max PB} \\
\cap & & \cap & & \cap & & \cap & & \\
\overline{\textsc{Max}\ \Sigma_0} & \subseteq & \overline{\textsc{Max}\ \Sigma_1} & \subset & \overline{\textsc{Max}\ \Pi_1} & = & \overline{\textsc{Max}\ \Pi_2} & \supset & \text{NPO PB} \\
\cap & & & & \| & & & & \\
\overline{\textsc{Min}\ \Sigma_0} & = & \overline{\textsc{Min}\ \Sigma_1} & \subseteq & \overline{\textsc{Min}\ \Pi_1} & \supset & \text{NPO PB} & & \\
\cup & & \cup & & \cup & & & & \\
\textsc{Min}\ \Sigma_0 & = & \textsc{Min}\ \Sigma_1 & \subset & \textsc{Min}\ \Pi_1 & = & \textsc{Min PB} & &
\end{array}$$

Figure 4.1: *Relations between classes defined by logical formulas.*

This theorem together with Proposition 4.30 shows that $\overline{\textsc{Max}\ \Sigma_0}$ (which is the same as $\overline{\textsc{Max SNP}}$) is a subset of $\overline{\textsc{Min}\ \Sigma_0}$. Figure 4.1 contains a summary of the relations between classes in the hierarchies and their closures.

4.10.8 Other syntactically defined classes

For many natural optimization problems a feasible solution can be expressed as a relation and the objective function is the cardinality of the relation. For example a feasible solution of Max 3DM is a set of triples which do not agree in any coordinate and the objective function is the cardinality of the set. Another example is Min Node Cover which has a set of nodes forming a node cover as its feasible solution. The objective function is the cardinality of the set.

In this section we will define new classes which are subclasses of syntactically defined classes and contain only problems which can be formulated as described above.

Definition 4.23 [60] A minimization problem G is in the class Min FΠ_1 iff its optimization function for the instance $I \in \mathcal{I}_G$ can be expressed as

$$opt_F(I) = \min_S |\{x : \forall y\ \Phi_G(I,S,y) \Rightarrow S(x)\}|$$

where Φ_G is a quantifier free formula. If there is no S such that $\Phi_G(I,S,y)$ is true then $opt_G(I)$ is equal to the trivial value $triv_G$. For technical reasons we define $triv_G$ as the number of possible different values that x can take.

Definition 4.24 [60] A maximization problem G is in the class Max FΠ_1 iff its optimization function for the instance $I \in \mathcal{I}_G$ can be expressed as

$$opt_F(I) = \max_S |\{x : \forall y\ \Phi_G(I,S,y) \wedge S(x)\}|$$

where Φ_G is a quantifier free formula. If there is no S such that $\Phi_G(I,S,y)$ is true then $opt_G(I)$ is equal to the trivial value 1.

Similarly we define Min FΠ_n and Max FΠ_n for $n = 2, 3, \ldots$ generalizing Min Π_n and Max Π_n in the corresponding way. By definition Min F$\Pi_n \subseteq$ Min Π_n and Max F$\Pi_n \subseteq$ Max Π_n but we can in fact say much more.

Proposition 4.34 (Kolaitis and Thakur [60])
For each $n \geq 1$

$$\textsc{Min}\ \mathrm{F}\Pi_n = \textsc{Min}\ \Sigma_n \textit{ and } \textsc{Max}\ \mathrm{F}\Pi_n = \textsc{Max}\ \Pi_n.$$

Proposition 4.35 (Kolaitis and Thakur [60])

$$\textsc{Min}\ \mathrm{PB} = \textsc{Min}\ \mathrm{F}\Pi_2 = \textsc{Min}\ \mathrm{F}\Pi_n,\ n \geq 2$$

$$\textsc{Max}\ \mathrm{PB} = \textsc{Max}\ \mathrm{F}\Pi_2 = \textsc{Max}\ \mathrm{F}\Pi_n,\ n \geq 2$$

Thus these hierarchies collapse at $n = 2$.

A family of subclasses of these classes are more interesting in the framework of approximability.

Definition 4.25 [60] A minimization problem F is in Min $\mathrm{F}^+\Pi_1(k)$ iff its optimization function for the instance $I \in \mathcal{I}_F$ can be expressed as

$$opt_F(I) = \min_S |\{x : \forall y\ \Phi_F(I, S, y) \Rightarrow S(x)\}|$$

where Φ_F is a quantifier free DNF formula, all occurrences of S in Φ_F are positive, and S appears at most k times in each disjunct.

Definition 4.26 [60] Min $\mathrm{F}^+\Pi_1$ is defined as

$$\bigcup_{k \in \mathbb{N}} \textsc{Min}\ \mathrm{F}^+\Pi_1(k)$$

Analogously we can define the class Min $\mathrm{F}^+\Pi_2$.

Example 4.7 [61] Min $\mathrm{F}^+\Pi_1(2)$ contains Min Node Cover as a member because the minimum node cover of a graph $G = \langle V, E\rangle$ is given by

$$opt(G) = \min_S |\{x : (\forall y \forall z, (y, z) \notin E \vee y \in S \vee z \in S) \Rightarrow x \in S\}|\,.$$

Example 4.8 [61] Min Node Cover can be generalized to k-hypergraphs. A k-hypergraph is a structure $\langle A, E\rangle$ where $E \subseteq A^k$. A hypernode cover is a set $S \subseteq A$ such that for every k-tuple $(a_1, a_2, \ldots, a_k) \in E$ we have that S contains some a_i. The minimum k-hypernode cover problem is to find the smallest hypernode cover in a k-hypergraph. It can be placed in Min $\mathrm{F}^+\Pi_1(k)$ by defining $opt(\langle A, E\rangle)$ as

$$\min_S |\{x : (\forall y_1 \ldots \forall y_k, (y_1, \ldots, y_k) \notin E \vee y_1 \in S \vee \cdots \vee y_k \in S) \Rightarrow x \in S\}|\,.$$

Note that for $k = 2$ the problem is the same as Min Node Cover. We can find a node cover (using a maximal matching) which is at most twice the size of the optimal node cover [34]. The same idea can be used to approximate Min k-hypernode Cover within the constant k.

Example 4.9 [60] Min $\mathrm{F}^+\Pi_2(1)$ contains Min SC as a member because the minimum set cover of a collection of subsets C of a finite set X is given by

$$opt(\langle X, C\rangle) = \min_S\{|S| : \forall y \in X \exists z \in C : z \in S \wedge y \in z\}$$

which can be written in the required way if we introduce a relation $M(y, z)$ which is true when the element $y \in X$ is contained in the subset $z \in C$:

$$opt(\langle X, M\rangle) = \min_S |\{x : (\forall y \exists z, y \in X \Rightarrow (z \in C \wedge M(y, z))) \Rightarrow x \in S\}| \,.$$

Proposition 4.36 (Kolaitis and Thakur [61])
For each $k \geq 2$ Min k-hypernode Cover *is complete for* Min $\mathrm{F}^+\Pi_1(k)$*. As a result* Min $\mathrm{F}^+\Pi_1 \in$ Apx.

Proposition 4.37 (Kolaitis and Thakur [61])
Min SC *and* Min Dominating Set *are* Min $\mathrm{F}^+\Pi_2(1)$*-complete. As a result every problem in* Min $\mathrm{F}^+\Pi_2(1)$ *can be approximated within $O(\log n)$.*

Analogously with the classes Min $\mathrm{F}^+\Pi_1(k)$ we can also define the classes Max $\mathrm{F}^-\Pi_1(k)$. These were originally introduced by Panconesi and Ranjan, but under the name Rmax(k).

Definition 4.27 [84] A maximization problem $F \in$ Max $\mathrm{F}^-\Pi_1(k)$ if its optimization function for the instance $I \in \mathcal{I}_F$ can be expressed as

$$opt_F(I) = \max_S |\{x : S(x) \wedge \forall y \; \Phi_F(I, S, y)\}|$$

where Φ_F is a quantifier free CNF formula, all occurrences of S in Φ_F are negative, and S appears at most k times in each clause.

Definition 4.28 [84] Max $\mathrm{F}^-\Pi_1$ is defined as

$$\bigcup_{k \in \mathbb{N}} \text{Max } \mathrm{F}^-\Pi_1(k)$$

Example 4.10 [84] Max Clique can also be generalized to k-hypergraphs. A hyperclique in a k-hypergraph $\langle A, E\rangle$ is a set $S \subseteq A$ such that for every k-tuple e of elements from S we have that $e \in E$. The maximum k-hyperclique problem is to find the largest hyperclique in a k-hypergraph. It can be placed in Max $\mathrm{F}^-\Pi_1(k)$ by defining $opt(\langle A, E\rangle)$ as

$$\max_S |\{x : x \in S \wedge (\forall y_1 \ldots \forall y_k, y_1 \notin S \vee \cdots y_k \notin S \vee (y_1, \ldots, y_k) \in E)\}| \,.$$

Proposition 4.38 (Panconesi and Ranjan [84])
For all $k \geq 2$, Max k-Hyperclique *is* Max $\mathrm{F}^-\Pi_1(k)$*-complete.*

Proposition 4.39 (Panconesi and Ranjan [84])
Max Clique*,* Max SP *and* Max k-colourable IS *are* Max $\mathrm{F}^-\Pi_1(2)$*-complete.*

All Max $\mathrm{F}^{-}\Pi_1(2)$-complete problems have the same approximability, see further Section 4.11.

Proposition 4.17 says that Max Clique is $\overline{\text{Max NP}}$-hard, which means that every problem in $\overline{\text{Max NP}}$ can be P-reduced to any Max $\mathrm{F}^{-}\Pi_1(2)$-complete problem. Thus

$$\overline{\text{Max }\Sigma_1} \subset \overline{\text{Max F}^{-}\Pi_1(2)}.$$

4.11 The Max Ind Set class

As we saw in Section 3.2 the Max Clique problem is actually the same problem as Max Ind Set on the complementary graph. We shall see that there are several problems which are as hard to approximate as these problems, for example the maximum set packing problem (see Section 5.4) and the maximum common induced connected subgraph problem (see Section 6.2). For simplicity we call the set of problems which are equivalent with these problems with respect to the P-reduction the Max Ind Set class.

Definition 4.29 Let *the* Max Ind Set *class* be the set of NPO problems F such that $F \equiv_P^p$ Max Ind Set.

The Max $\mathrm{F}^{-}\Pi_1(2)$-complete problems (defined in Section 4.10.8) are a subset of the Max Ind Set class.

There are a lot of results regarding the approximability of the problems in this class.

The best positive result is that Max Ind Set and Max Clique can be approximated within $O(n/(\log n)^2)$ (where n is the number of nodes in the input graph) and is due to Boppana and Halldórsson [18].

Garey and Johnson have shown that either there is a Ptas for Max Ind Set or the problem cannot be approximated within any constant, that is Max Ind Set $\notin$ Apx [34]. They used the fact that Max Ind Set is self-improvable.

Berman and Schnitger have shown a stronger result: if some Max SNP-complete problem does not have an Rptas (a randomized polynomial time approximation scheme) then for some constant $c > 0$ it is impossible to approximate Max Ind Set within n^c [12]. Thus this result relates the approximability of the Max SNP-complete problems with the approximability of Max Ind Set. This is done via an amplifying chain of problems starting with maximum 2-constraint satisfaction, ending with maximum $O(\log n)$-constraint satisfaction and using R-reductions. The maximum k-constraint satisfaction problem takes a set of n conjunctions, each involving at most k literals and returns a truth assignment which satisfies the largest number of conjunctions. Maximum 2-satisfiability can be L-reduced to maximum 2-constraint satisfaction (see Theorem A.3) and it is easy to see that maximum $O(\log n)$-constraint satisfaction can be L-reduced to Max Ind Set.

This result has been improved by Halldórsson [37] using a method by Blum [16]:

Proposition 4.40 (Halldórsson [37])
If Min Node Cover $\notin$ Rptas *then for some constant $c > 0$ it is impossible to approximate* Max Ind Set *within n^c where n is the number of nodes in the input graph.*

Crescenzi, Fiorini and Silvestri have shown that Max Clique is $\overline{\text{Max NP}}$-hard, see Proposition 4.17, and thus that if Max Clique can be approximated within a constant then $\overline{\text{Max NP}} \subset$ Ptas. As we will see this result is no longer interesting since Max Clique $\notin$ Apx.

A completely different approach to the problem has been done by Feige, Goldwasser, Lovász, Safra and Szegedy [30]. They have shown that Max Clique (and therefore also every other problem in the Max Ind Set class) cannot be approximated within $2^{(\log n)^{(1-\varepsilon)}}$ in polynomial time, unless $\text{NP} \subset \tilde{\text{P}}$, where $\tilde{\text{P}}$ denotes the set of languages accepted in quasi polynomial time (i.e. time $2^{\log^c n}$). This result was proved using interactive protocols. Recently Arora and Safra improved this result, finally confirming that Max Clique $\notin$ Apx and thereby eliminating the possibility of a Ptas for the problem [3].

And yet more recently Arora, Lund, Motwani, Sudan and Szegedy showed that a $\overline{\text{Max SNP}}$-complete problem cannot admit a Ptas, see Theorem 4.25, which with results by Feige, Goldwasser, Lovász, Safra and Szegedy [30] give us the following corollary.

Corollary 4.41 (to Theorem 4.25)
There is a constant $c \in \mathbb{R}^+$ such that Max Ind Set *cannot be approximated within n^c, unless* $\text{P} = \text{NP}$. *n is the number of nodes in the input graph.*

Chapter 5

Matching and packing problems

5.1 Introduction

In the following chapters we will concentrate on the approximability of certain categories of NPO problems. This chapter will deal with the approximability of several matching and packing problems. In Chapter 6 the family of maximum common subgraph problems will be studied, and Chapter 7 will be devoted to the travelling salesperson problem.

First we will show that Max 3DM, the optimization version of the NP-complete 3-dimensional matching problem is Max SNP-hard, even if the number of occurrences of any element is bounded by a constant. This is shown by a reduction from Max 3Sat $-B$. As we have already seen, Max 3DM is in Max SNP if the number of occurrences of each element in triples is bounded by a constant. Thus Max 3DM $-B$ is Max SNP-complete.

Two related problems, maximum bounded three-set packing and maximum bounded triangle packing, are also shown to be Max SNP-complete. The results can be extended to maximum k-dimensional matching and maximum k-set packing.

A generalization of the maximum bounded triangle packing problem is the maximum bounded H-matching problem, in which we want to find the largest collection of node-disjoint copies of the constant graph H in a bounded degree graph. This problem is shown to be $\overline{\text{Max SNP}}$-hard if H contains three or more nodes in some connected component. The proof is built (as most of the proofs in this chapter) on the proof that Max 3DM $-B$ is Max SNP-hard.

5.2 Maximum three dimensional matching

Theorem 5.1 Max 3DM $-B$, $B \geq 3$ *is* Max SNP*-complete.*

In the proof we will use a polynomial-time transformation f from Max 3Sat $-B$ to Max 3DM $-B$. It is an extension of the reduction used to prove that

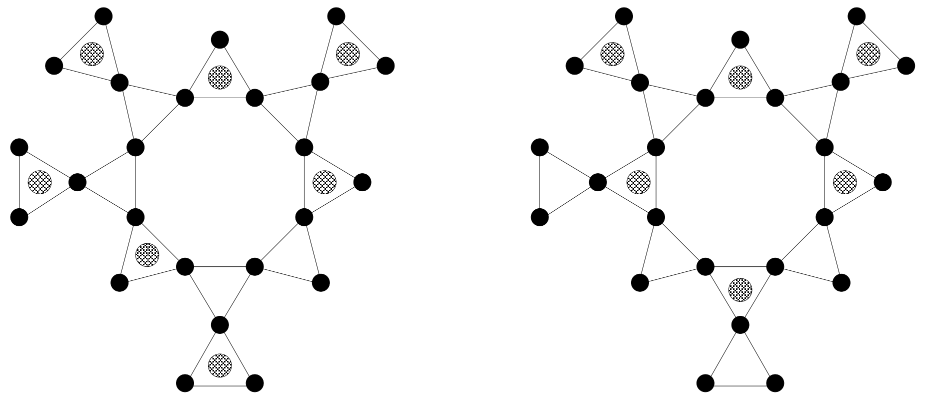

Figure 5.1: *Two matchings of the structure described in [34] corresponding to the clauses $\{x, x, \overline{x}, \overline{x}\}$. Each triangle is a triple in M and each dot is an element in either W, X, or Y. The matched triples are marked with rings. The left matching is optimal (with 7 matched triples) but the right is the matching corresponding to x true (with 6 matched triples).*

3DM is NP-complete in [34]. In that proof, for each variable a ring of triangles is formed (see Figure 5.1), the triangles alternately represent the variable and its negation, so that a perfect matching corresponds to fixing a truth value. The rings are interconnected by triangles representing clauses. For optimization, the problem with this reduction is that, since we allow solutions which are not perfect matchings, a better matching may result if, instead of including every other triple in the rings, we include many clause triples (see Figure 5.1). We are able to solve this problem by creating many rings for every variable, and connecting these rings with binary trees.

Let I be an instance of Max 3Sat $-B$ with $U = \{u_1, \ldots, u_n\}$ and $C = \{c_1, \ldots, c_m\}$. Let d_i be the total number of occurrences of u_i in C, either as u_i or $\overline{u}_i$. We know that $d_i \leq B$ for each i. Let $K = 2^{\lfloor \log(\frac{3}{2}B+1) \rfloor}$, i.e. the largest power of two such that $K \leq \frac{3}{2}B + 1$.

For every variable u_i we construct K identical rings, each ring containing $2d_i$ *ring triples*, connected as in Figure 5.2. The free elements in the ring triples of ring k are called $v_i[\gamma, k]$ and $\overline{v}_i[\gamma, k]$ ($1 \leq \gamma \leq d_i, 1 \leq k \leq K$) as in Figure 5.2.

The K rings are connected by *tree triples* in $2d_i$ binary trees in such a way that the elements $v_i[1, 1], v_i[1, 2], \ldots, v_i[1, K]$ are leaves in the first tree and the root of this tree is called $u_i[1]$; $\overline{v}_i[1, 1], \ldots, \overline{v}_i[1, K]$ are leaves in the second tree and the root is called $\overline{u}_i[1]$, et cetera.

Thus there are $2d_i(2K - 1)$ triples which originate in the single variable u_i. There are K rings with $2d_i$ ring triples in each which are connected with $2d_i$ binary trees with $K - 1$ tree triples in each. This structure we call the *ring of trees* corresponding to u_i. See Figure 5.3 for an example.

Finally *clause triples* connect some of the roots. For each clause c_j we introduce two new elements $s_1[j]$ and $s_2[j]$ and (at most) three triples connecting them to the appropriate root elements. If the variable u_i occurs in this clause

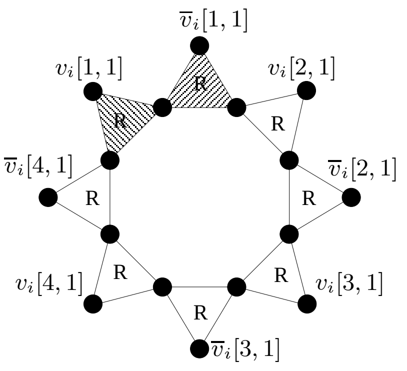

Figure 5.2: *The first of K rings for a variable u_i with d_i=4 occurrences. The shaded triangles are the same as the shaded triangles in Figure 5.4.*

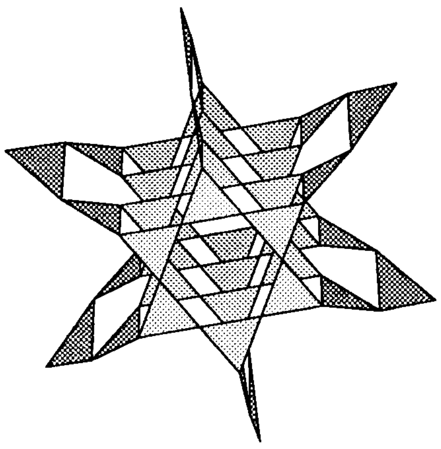

Figure 5.3: *A ring of trees with $d_i = 3$ and $K = 4$.*

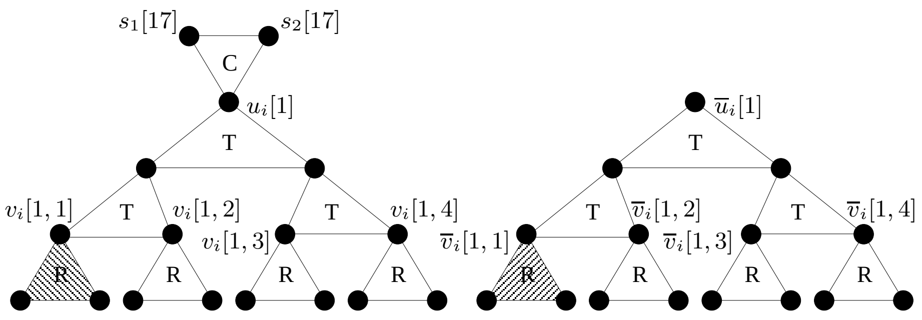

Figure 5.4: *An example of binary trees for u_i and the adjacent clause triple and ring triples where the first occurrence of u_i in C is in the 17-th clause. The triples are marked with R, T or C for ring, tree and clause triples, respectively. If u_i occurs as u_i and the tree contains an even number of levels (as in this example) or if u_i occurs as $\overline{u}_i$ and the tree contains an odd number of levels then the clause triple is connected to the root of the left tree. Otherwise it is connected to the right tree.*

and this is its γ-th occurrence in C then root element in the triple is to be $u_i[\gamma]$ or $\overline{u}_i[\gamma]$, depending on whether the occurrence was u_i or $\overline{u}_i$ and whether the binary tree contains an even or odd number of levels.

Now we have constructed all elements and triples which are needed to define the transformation f.

For the ring, tree and clause triples to define $M \subseteq W \times X \times Y$ we have to label the elements with w, x or y. All trees are labelled identically in the following way. Start at the root, label it w, and label the elements in every tree triple w, x and y anti-clockwise. The ring triples are labelled anti-clockwise in u_i-trees and clockwise in $\overline{u}_i$-trees, in some suitable planar representation of

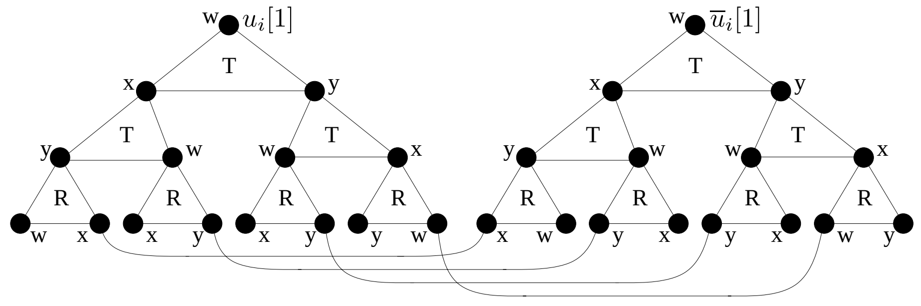

Figure 5.5: *An example of element labelling in a tree with two levels. Dots in the bottom layer which represent identical elements are connected with arcs.*

each ring. $s_1[j]$ and $s_2[j]$ are labelled x and y respectively. In this way every element gets a unique label, see Figure 5.5.

Now the transformation f is fully described. It contains $\sum_{i=1}^{n} d_i$ clause triples, $\sum_{i=1}^{n} K \cdot 2d_i$ ring triples and $\sum_{i=1}^{n} 2d_i \cdot (K-1)$ tree triples, that is

$$\begin{aligned} |M| &= \sum_{i=1}^{n} d_i + \sum_{i=1}^{n} 2d_i(K+K-1) \\ &\leq 3m + \sum_{i=1}^{n} 2d_i(3B+1) \\ &\leq 3m + \sum_{i=1}^{n} 6B^2 + 2\cdot 3m \\ &= 9m + 6B^2 n. \end{aligned}$$

f can be computed in polynomial time in m and n. Moreover, every element in a ring or tree triple occurs exactly two times in M, except for half of the root elements ($u_i[\gamma]$ or $\overline{u}_i[\gamma]$) which occur only once. The only remaining elements are $s_1[j]$ and $s_2[j]$ in the clause triples; they occur at most three times each, because a clause contains at most three literals. Thus $f(I)$ is a problem in Max 3DM $-B$ for $B \geq 3$.

We now study three dimensional matchings of the ring of trees corresponding to the variable u_i. First we show that in an optimal matching of this structure all rings belonging to the same variable are matched in the same way, i.e. all triples in the same positions (e.g. the ring triples which contain $v_i[1,k], 1 \leq k \leq K$) are either in the matching or not in the matching. We say that a triple is *chosen* if it is included in the matching.

Consider a *pair* of rings, that is, two rings connected to the same lowest-level tree triples. Suppose that the first ring has t_1 and the second ring t_2 chosen triples and that $t_1 \geq t_2$. Look at the matching for the two rings and the $2d_i$ tree triples that connect them. Construct a new matching of this structure from the old matching in the following way. Observe that for each of the t_1 chosen triples in the first ring, the connected tree triple cannot be chosen. Therefore we can match the second ring in the same way as the first. This matching is at least as large as the original matching. For every lowest-level tree triple, choose it if the connected ring triples are not chosen (if this is not possible because the connected tree triple on the next level is chosen, we first delete it from the matching before choosing the new tree triple). This will only make the matching larger.

Now we do the same thing one level up. We look at two adjacent pairs of rings (which are connected to the same $2d_i$ second lowest level tree triples) including the $2 \cdot 2d_i$ lowest-level tree triples which connect each pair and the $2d_i$ second lowest level tree triples which connect these tree triples. Suppose that the two rings in the first pair have been equally matched as in the preceding paragraph and likewise that the two rings in the second pair are equally matched. Suppose that the first half (the first pair of rings and the $2d_i$ tree triples that connect them) has t_1 chosen triples and that the second half (the second pair and $2d_i$ tree triples) has t_2 chosen triples, where $t_1 \geq t_2$. In exactly the same way as above we can get a larger matching if we match all rings in the same way as in the largest ring matching, for every lowest-level tree triple choose it if the connected ring triples are not chosen and for every second lowest level tree triple choose it if the connected lowest-level tree triples are not chosen.

This procedure can continue in $\log K - 1$ steps, adding a new tree level in every step. We have shown that in an optimal matching of the $2d_i(2K-1)$ triples in the ring of trees corresponding to u_i all rings must be equally matched and that the triples in every other level of the trees must be contained in the matching.

Now say that only $d_i - \delta$ triples from every ring are included in the matching but all rings are equally matched. Then we will get a matching of the ring of trees of size (# matched ring triples) + (# matched tree triples in trees connected to matched ring triples) + (# matched tree triples in trees connected to unmatched ring triples) which is less than

$$(d_i - \delta)K + (d_i - \delta)\frac{K-1}{3} + (d_i + \delta)\frac{2K-2}{3} = d_i(2K-1) - \delta\frac{2K+1}{3}$$

if $K = 2^\alpha$ and α is even, and less than

$$(d_i - \delta)K + (d_i - \delta)\frac{K-2}{3} + (d_i + \delta)\frac{2K-1}{3} = d_i(2K-1) - \delta\frac{2K-1}{3}$$

if $K = 2^\alpha$ and α is odd.

Obviously $\delta = 0$ gives us a maximum matching with $d_i(2K-1)$ triples. $\delta = 1$ gives us a matching of the structure which is at least $\frac{2K-1}{3}$ triples smaller than the maximum matching.

Finally, still concentrating on the i-th variable's triples, we have d_i clause triples which are situated at the root of some of the binary trees, corresponding to the clauses to which the variable belongs and whether it is negated or not in that clause. We would like the property that every other ring triple is included in the matching to be more valuable than the inclusion of any number of clause triples. Assume that we sacrifice this property by including only $d_i - \delta$ triples from every ring in the matching and that we include p clause triples instead. From these p triples one can always choose $\lceil \frac{p}{2} \rceil$ triples which occur at even distances from each other (corresponding to the fact that one of u_i or $\overline{u}_i$ appears in at least $\lceil \frac{p}{2} \rceil$ clauses). Thus we can always obtain a matching with $d_i(2K-1) + \lceil \frac{p}{2} \rceil$ triples without sacrificing the aforementioned property. We want that

$$d_i(2K-1) + \left\lceil \frac{p}{2} \right\rceil > d_i(2K-1) - \delta \frac{2K-1}{3} + p \text{ for } 1 \le \delta \le d_i, p \le d_i$$

$$\Leftarrow \frac{1}{3}(2K-1) > p - \left\lceil \frac{p}{2} \right\rceil = \left\lfloor \frac{p}{2} \right\rfloor \text{ for } p \le d_i \Leftarrow K > \frac{3}{2}\left\lfloor \frac{d_i}{2} \right\rfloor + \frac{1}{2}$$

This is true because

$$\begin{aligned} K &= 2^{\lfloor \log(\frac{3}{2}B+1) \rfloor} \ge 2^{\lfloor \log(\frac{3}{2}d_i+1) \rfloor} = 2^{\lfloor \log(\frac{3}{4}d_i+\frac{1}{2}) \rfloor + 1} > 2^{\log(\frac{3}{4}d_i+\frac{1}{2})} = \\ &= \frac{3}{4}d_i + \frac{1}{2} \ge \frac{3}{2}\left\lfloor \frac{d_i}{2} \right\rfloor + \frac{1}{2}. \end{aligned}$$

Let A be the structure consisting of the rings of trees corresponding to all variables (i.e. A is M without the clause triples). It is easy to find a maximum matching for A, namely every other triple, and there are two ways to obtain it for every i. These two ways correspond to the truth values of u_i. With the choice of K above we know that any maximum matching of the whole problem must contain a maximum matching of the substructure A. In order to obtain the optimal matching one has to include as many of the clause triples as possible. At most one triple from every C_j can be included. A triple $(u_i[\gamma], s_1[j], s_2[j])$ can be included only if the root triple in the corresponding tree is not included, and this depends on the manner in which the substructure for variable i is matched.

We see that solving a MAX 3SAT $-B$-problem I is equivalent to solving the MAX 3DM-problem $f(I)$.

Lemma 5.2 *The transformation* f : MAX 3SAT $-B \to$ MAX 3DM -3 *is an L-reduction.*

PROOF Assume that I is an instance of MAX 3SAT $-B$. We have to show the two following inequalities.

1. Show that $opt\big(f(I)\big) \le \alpha\, opt(I)$ for a constant $\alpha > 0$.

$$\begin{aligned} opt\big(f(I)\big) &= \sum_{i=1}^{n} \big(d_i(2K-1)\big) + opt(I) \\ &= \Big(\sum_{i=1}^{n} d_i\Big)\big(2K-1\big) + opt(I) \\ &\le 3m\big(2 \cdot (\frac{3}{2}B+1) - 1\big) + opt(I) \\ &\le (18B+7)\, opt(I) \end{aligned}$$

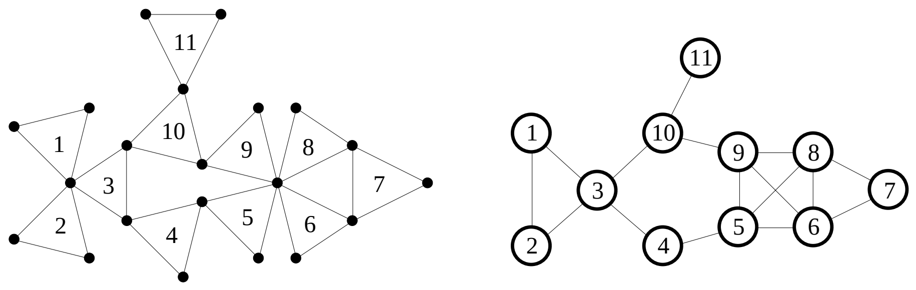

Figure 5.6: *An example of a matching problem I and the* Max Ind Set *problem $g(I)$.*

because $opt(I) \geq \frac{m}{2}$ (it is always possible to satisfy at least half of the clauses in I). Thus $\alpha = 18B + 7$ in the inequality above.

2. Show that for every matching for $f(I)$ of size c_2 we can, in polynomial time, find a solution of I with c_1 clauses satisfied and $opt(I) - c_1 \leq \beta\big(opt\big(f(I)\big) - c_2\big)$ where $\beta = 1$.

   If the given matching is not optimal on the substructure A (defined above) then it will increase in size if we make it optimal on A (as seen above). Thus we may presume that the given matching for $f(I)$ is optimal on A. By setting the variables of I as the matching indicates (i.e. by looking at the matching for every ring) we will get an approximate solution of I which satisfies c_1 clauses and $opt(I) - c_1 = opt\big(f(I)\big) - c_2$.

□

In order to show that Max 3DM $-B$ is in $\overline{\text{Max SNP}}$ we define a new polynomial-time transformation g from Max 3DM $-B$ to Max Ind Set $-B$. Let $V = M$, that is, for every triple in M we have a node in the graph. There is an edge in E between two triples in M if they have at least one element in common. Thus, for every element in $W \cup X \cup Y$ which occurs k times in M we have a k-clique between the corresponding nodes in V, see Figure 5.6.

The degree of a node in $g(I)$ is at most $3 \cdot (B - 1)$ where B is the bound of occurrences in M. Thus $g(I)$ is a bounded independent set problem.

Lemma 5.3 *The transformation g from the problem* Max 3DM $-B$ *to the problem* Max Ind Set $-\big(3(B-1)\big)$ *is an L-reduction.*

Proof Assume that I is an instance of Max 3DM $-B$ with set of triples M. Two triples in M are adjacent if and only if the two corresponding nodes in $g(I)$ are adjacent. We see that the three dimensional matching problem of I corresponds exactly to the independent set problem of $g(I)$. Thus we get $opt\big(g(I)\big) = opt(I)$ and for every independent set for $g(I)$ of size c we immediately have a three dimensional matching for I of size c. g is an L-reduction since both of the needed inequalities are satisfied (with $\alpha = \beta = 1$).

□

Proof of Theorem 5.1 By Lemma 5.2 Max 3Sat $-B$ L-reduces to Max 3DM -3 and by Lemma 5.3 Max 3DM $-B$ L-reduces to Max Ind Set $-\big(3(B-1)\big)$. Since Max 3Sat $-B$ and Max Ind Set $-B$ are Max SNP-complete [88] Max 3DM $-B$ is $\overline{\text{Max SNP}}$-complete for $B \geq 3$. Theorem 4.26 says that Max 3DM $-B$ is in Syntactic Max SNP with a different encoding. Thus Max 3DM $-B$ is Max SNP-complete with the new definition of Max SNP (Definition 4.20). □

We have seen that maximum bounded three dimensional matching is Max SNP-complete, but how about the harder, unbounded problem? The transformation g above takes an unbounded Max 3DM problem to an unbounded Max Ind Set problem. Unfortunately the latter problem is not in Max SNP. If there is no polynomial-time approximation scheme for some Max SNP-complete problem, then Max Ind Set cannot be approximated within any constant, see Section 4.11. But Max 3DM is not this hard. The trivial algorithm described below approximates the optimal solution within 3. Thus it is still possible that the unbounded version of Max 3DM is in Max SNP or $\overline{\text{Max SNP}}$.

Algorithm 5.1
Input: A set M of triples.
Output: A maximal matching $M' \subseteq M$.
Algorithm:

```
$T := M; M' := \emptyset;$
while $T \neq \emptyset$ do begin
    $t :=$ any element in $T$;
    $M' := M' \cup \{t\}$;
    $T := T - \{t$ and all its neighbours in $T\}$;
end;
```

Analysis: For any triple in the found matching M', consider it and all its neighbour triples. At most three of these triples can be in the optimal matching. Thus the algorithm approximates Max 3DM within 3.

5.3 Maximum three-set packing

We are given a collection of sets of size three. The maximum three-set packing problem or Max 3SP is to find the largest number of pairwise disjunctive sets.

In Max 3SP $-B$, the bounded version of the problem, the same element cannot be included in more than B three-sets.

Theorem 5.4 Max 3SP $-B$*, $B \geq 3$ is* Max SNP*-complete.*

Proof The difference between maximum three dimensional matching and maximum three-set packing is that in the former problem the elements are of different types (W, X or Y) but in the latter problem all elements are of the same type A.

Define f : MAX 3DM $-B \rightarrow$ MAX 3SP $-B$ by $A = W \cup X \cup Y$ and $C = M$. Now $opt\big(f(I)\big) = opt(I)$ and every solution of $f(I)$ can be translated to an equally good solution of I by f^{-1}.

Let g : MAX 3SP $-B \rightarrow$ MAX IND SET $-\big(3(B-1)\big)$ be the same transformation as in the reduction from MAX 3DM $-B$, that is, two nodes in the graph are connected if the corresponding sets have a common element.

Both f and g are L-reductions and MAX 3DM $-B$, $B \geq 3$ and MAX IND SET $-B$ are MAX SNP-complete. Thus MAX 3SP $-B$ is $\overline{\text{MAX SNP}}$-complete for $B \geq 3$. It is also MAX SNP-complete by Theorem 4.26. □

MAX 3SP can be approximated within 3 using Algorithm 5.1.

5.4 Maximum k-set packing

The maximum k-set packing problem, where k is a positive integer, is the natural generalization of the maximum three-set packing problem, that is, we are given a collection of sets of size k and want to find the largest number of pairwise disjunctive sets.

For $k = 1$ the problem is trivial, for $k = 2$ it is the same problem as maximum matching and can be solved optimally in polynomial time [78]. For $k > 3$ the problem is at least as hard to approximate as maximum three-set packing, since we get an L-reduction with $\alpha = \beta = 1$ from MAX 3SP to MAX kSP by extending all sets to k elements by introducing extra dummy elements. Thus MAX kSP is $\overline{\text{MAX SNP}}$-hard for $k \geq 3$, even if the same element cannot be included in more than a bounded number of k-sets.

Using the same ideas as in Theorem 4.26 one can show that MAX kSP $-B$ $\in$ MAX SNP. This leads us to the following theorem.

Theorem 5.5 MAX kSP $-B$ *is* MAX SNP-*complete for all* $k \geq 3$.

The optimal solution of the maximum k-set packing problem is at most k times as big as any solution which is a maximal set packing, so if we once again use Algorithm 5.1 we approximate MAX kSP within k.

The general maximum set packing without any restrictions on the number of elements in the sets is much harder to approximate. It is in fact as hard to approximate as MAX CLIQUE and thus cannot be approximated within any constant unless P = NP.

Proposition 5.6 (Ausiello, D'Atri and Protasi [4]) MAX SP $\equiv^p_{sp}$ MAX CLIQUE *and* MAX SP $\equiv^p_L$ MAX CLIQUE *with* $\alpha = \beta = 1$ *and without node amplification.*

PROOF We will give the reductions but omit the proof that they are structure preserving.

Let an instance of MAX CLIQUE be given by the graph $G = \langle V, E \rangle$ and an instance of MAX SP be given by the collection C of subsets of some finite set.

Define a transformation f from Max Clique to Max SP by

$$C = \{S_1, \ldots, S_{|V|}\} \text{ where } S_i = \{\{v_i, v_j\} : v_j \in V \wedge (v_i, v_j) \notin E\}\,.$$

A set packing now corresponds to a clique of the same size. Thus f is an L-reduction with $\alpha = \beta = 1$.

Define a transformation g from Max SP to Max Clique by $V = C$ and $(S_i, S_j) \in E$ iff $S_i \cap S_j = \emptyset$. Since a maximum clique corresponds to a set packing of the same size g is an L-reduction with $\alpha = \beta = 1$.

In both the reductions $|C| = |V|$ so the reductions are S-reductions with size amplification n in the number of nodes and sets. Thus an approximation algorithm for one of the problems gives us an equally good approximation algorithm for the other problem. $\square$

Since Max SP $\equiv_L^p$ Max Clique it is in the Max Ind Set class, see Section 4.11.

5.5 Maximum k-dimensional matching

Using the same idea as in the preceding section we generalize maximum three dimensional matching to maximum k-dimensional matching, where k is a positive integer. We are given k-tuples from k pairwise disjoint sets and want to find the largest number of k-tuples which do not agree in any coordinate.

For $k = 1$ the problem is trivial, for $k = 2$ it is the same problem as maximum bipartite matching and can be solved optimally in polynomial time [78]. For $k > 3$ the problem is at least as hard to approximate as maximum three dimensional matching, since we get an L-reduction with $\alpha = \beta = 1$ from Max 3DM to Max kDM by extending all triples to k-tuples introducing extra dummy elements. Thus Max kDM is $\overline{\text{Max SNP}}$-hard for $k \geq 3$, even if the same element cannot appear in more than a bounded number of k-tuples.

Once again we can use the ideas in Theorem 4.26 to show that Max kDM $-B \in$ Max SNP and thus Max kSP $-B$ is Max SNP-complete for all $k \geq 3$.

Algorithm 5.1 will approximate Max kDM within k.

5.6 Maximum triangle packing

The Max Triangle Packing problem is the optimization version of the problem partition into triangles: given a graph $G = \langle V, E \rangle$, find the largest collection $\{V_i\}$ of mutually disjoint 3-sets of nodes in V such that every V_i induces a triangle (i.e. a 3-clique) in G. When bounding the degree of the input graph by B we get the problem Max Triangle Packing $-B$.

Theorem 5.7 Max Triangle Packing *$-B$, $B \geq 4$ is* Max SNP*-complete.*

Proof Since we can easy build a list of every possible triangle in the input graph it is clear that any Max Triangle Packing input can be written as a Max 3SP input. If the degree of the graph is bounded by B, every element in the Max 3SP input will occur in at most $B(B-1)/2$ three-sets. Thus Max Triangle Packing $-B$ is in Max SNP.

To show that Max Triangle Packing $-B$ is Max SNP-hard we can use the reduction by Garey and Johnson from 3DM to Partition into Triangles [34] to get an L-reduction from Max 3SP $-B$ to Max Triangle Packing $-2B$. This is proved in [52].

Another way is to use the reduction from Max 3Sat $-B$ to Max 3DM -3 in Theorem 5.1 and view the constructed structure as a graph. We observe that all possible triangles in the graph are the triples in the structure. Thus the same proof can be used to show that this is an L-reduction from Max 3Sat $-B$ to Max Triangle Packing -4. □

Max Triangle Packing can be approximated within 3 using Algorithm 5.1 where M is the set of possible triangles in the graph.

5.7 Maximum H-matching

Let H be a fixed graph. Max H-matching takes as input an undirected graph $G = \langle V, E\rangle$. The problem is to find the largest H-matching, that is, the largest collection of node-disjoint copies of H in G. The Max Induced H-matching problem is to find the largest induced H-matching, that is, the largest collection of disjoint subsets of V such that every subset induces a subgraph of G which is isomorphic to H.

Thus a solution of Max Induced H-matching is at the same time always a solution of Max H-matching, but not vice versa, since given a copy H′ of H in a solution of Max H-matching there may exist edges in G which are not in H between the vertices in H′.

If the input graph has degree bounded by a constant B we get the problems Max H-matching $-B$ and Max Induced H-matching $-B$.

Example 5.1 Let H be a 3-clique. Since H is a clique Max H-matching and Max Induced H-matching coincide. This problem is the maximum triangle packing problem known from Section 5.6.

Berman et al [10] have shown that the maximum planar H-matching problem is NP-complete for any connected planar graph H containing at least three nodes. Given a planar graph G and an integer k, this is the decision problem of determining whether G contains k node-disjoint copies of H. They also gave a fast approximation algorithm guaranteed to find $k(1 - O(1/\sqrt{\log k}))$ node-disjoint copies of H where k is the optimal number. Baker has shown that there exists a Ptas for the H-matching problem on planar graphs [6]. For general graphs the problem is likely to be harder to approximate in polynomial time.

We will show that for general graphs with at least three nodes in a connected component the problem is Max SNP-hard. This is shown by a reduction from the problem of maximum bounded 3-satisfiability. The same result holds for the induced H-matching problem. The reduction is an extension of the reductions used by Berman et al [10] and in Section 5.2 when proving that Max 3DM $-B$ is Max SNP-hard.

Theorem 5.8 *Maximum bounded* H*-matching is* Max SNP*-hard for any* H *with three or more nodes in some connected component.*

Proof We divide the possible H-graphs into three classes depending on whether they are connected and whether the largest 2-connected (i.e. 2-*node*-connected) component is unique. A cut edge is here considered to be a degenerate form of a 2-connected component. In each case we reduce Max 3Sat $-B$ to an input structure with generator rings connected by binary trees to receptors. The receptors correspond to the clauses in the 3Sat problem and the generator rings and trees correspond to the variables.

There are three types of base elements: generator elements, tree elements and receptor elements. Each element has three special nodes which are the only points connecting the element to other elements. In a maximum matching we want all three special nodes in every element to be matched in the same way.

Every variable in the 3Sat problem gives birth to a number K (which has to be a power of two) of generator rings, each consisting of $2d$ generator elements where d is the number of occurrences of the variable in clauses (see Figure 5.7). Two of the three contact points of a generator element connect it to neighbouring generator elements and the third connects it to a tree leaf.

Place all K generator rings created by a single variable on top of each other. Connect all tree contact points which are in the same position in the rings with a $\log K$-level binary tree consisting of $K-1$ tree elements. Two of the contact points of a tree element connect it to its children (or to generator elements if it is a leaf) and the third connects it to its father (if it is not the root of the tree). Thus the K rings are connected to $2d$ trees. This structure with all the rings and trees originating in the same variable we call a *ring of trees* (see Figure 5.8).

Every matching of the structure where all three special nodes in each base element are matched in the same way, we can look at as a matching of base elements. A base element is considered matched in the base element matching iff its three special nodes are matched in the original matching. A maximum base element matching of a ring of trees can be obtained in two ways (see Section 5.2) by including every other ring element and all tree elements in every other tree level in the matching, which means Kd ring elements and $d(K-1)$ tree elements. Mark the roots of the trees alternating $+$ and $-$ (walking around the rings in any direction). We say that the ring of trees is in *false mode* if it has a maximum matching and all its $+$ roots are in the matching but no $-$ root, and in *true mode* if all $-$ roots are in the matching but no $+$ root. The mode corresponds to whether the variable is true or false.

For every clause in the 3Sat problem we construct a receptor consisting of one receptor element per variable in the clause (see Figure 5.9) where each receptor element has one special node which should be identified with a root node in the ring of trees originating in the corresponding variable. If the variable appears uncomplemented in the clause we use a $+$ root and if the

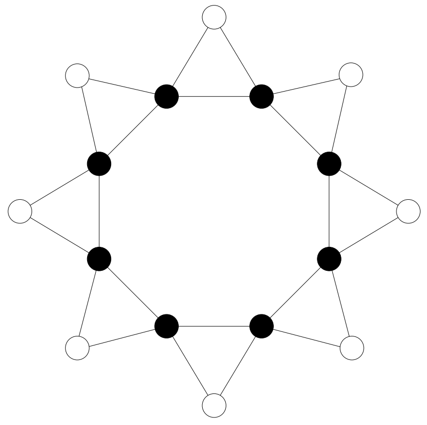
Figure 5.7: *A generator ring with* $d = 4$.

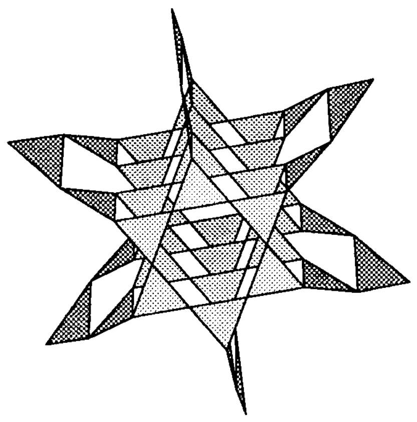
Figure 5.8: *A ring of trees with* $d = 3$ *and* $K = 4$.

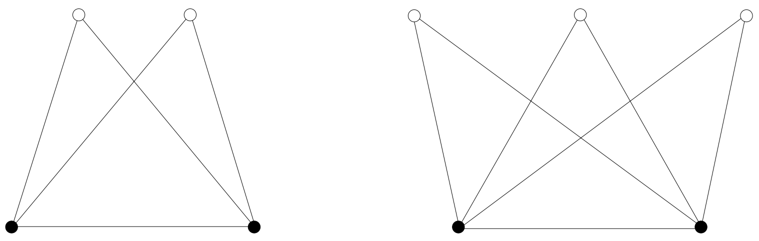
Figure 5.9: *A receptor with two elements and a receptor with three elements.*

variable appears complemented we use a $-$ node. Thus half of the root nodes in every ring of trees will be identified to receptor nodes and the rest of them will be free.

In order to specify the structure completely we now just have to define the generator, tree and receptor elements and the constant K. K should be chosen in such a way that for every matching of the structure we can in polynomial time find an at least as big matching where every ring of trees is in either true or false mode.

Class 1. H is connected and contains three or more nodes and a unique maximum-size 2-connected component.

The special case where H is a triangle (i.e. a 3-cycle) is the problem Max Triangle Packing $-B$ which was proved to be Max SNP-complete in Section 5.6. That proof uses the described structure with $K = 2^{\lfloor \log(\frac{3}{2}B+1) \rfloor}$ and all the generator, tree and receptor elements as triangles.

Class 1, general case.

Let H be any connected graph with at least three nodes for which the maximum-size 2-connected component, which we henceforth will denote by $\overline{\mathrm{H}}$, is unique. Choose three nodes from any cycle in $\overline{\mathrm{H}}$ and call them a, b and c.

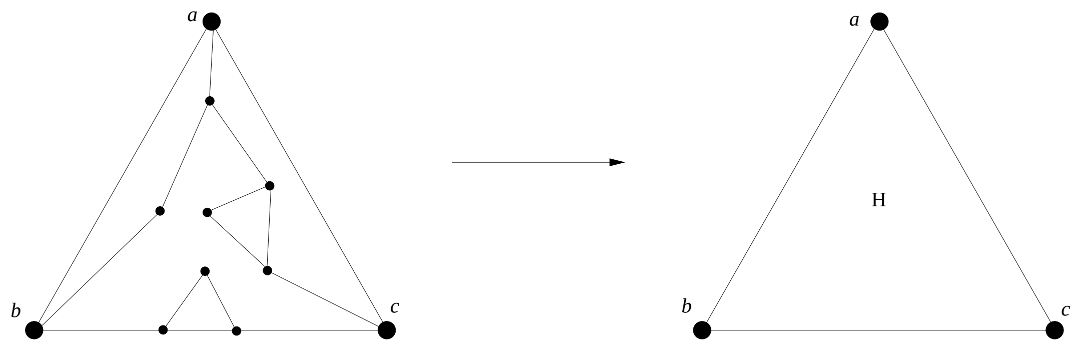

Figure 5.10: *Representation of H as a triangle.*

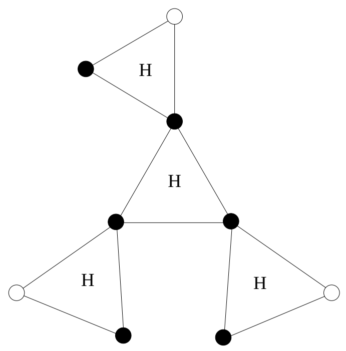

Figure 5.11: *A generator element in class 1. The three special nodes are open circles.*

Now we can represent H as a triangle (see Figure 5.10). H with a, b and c as special nodes will be used as both tree element and receptor element. We will use the same K as in the special case above, and a generator element consisting of four copies of H, connected as in Figure 5.11. The reason for complicating the generator element in this way is that it helps us to get a matching consisting just of fully matched triangles, as will be explained later.

The graph will get degree at most $\max\{\deg(H), 3\deg(a), 3\deg(b), 3\deg(c)\}$. We have to show that for every matching of the structure we can in polynomial time find an at least as big matching where no copy of H occurs across several triangles. First note that if the $\overline{\text{H}}$ part of H is inside one triangle it must contain the a, b and c nodes (because H contains only one maximum-size 2-connected component), and we can always move the whole copy of H into the triangle without disturbing the rest of the matching. Thus we need only look at copies of $\overline{\text{H}}$.

Every copy of $\overline{\text{H}}$ which occurs across several triangles must visit a cycle of triangles (because $\overline{\text{H}}$ is 2-connected) and for each triangle in the cycle it must use at least two of the nodes a, b and c. By inspection of the structure we see that every cycle must go through a generator element, and in the generator element through a triangle with an a, b or c node not connected to any other triangle, because it is only the center triangle in a generator element in which all three special nodes are connected to other triangles. Thus we can move the whole copy of H into this one triangle, without disturbing anything else.

Now we are back to the special case with the only difference that the generator elements contain four triangles instead of one.

In the same way as in Section 5.2 we see that given a matching of a ring of trees we can always modify it to a matching with all the rings matched in the same way. Ring triangles which are connected to the tree leaves we call *top triangles.* All ring triangles except the top triangles compose a chain of $6d$ triangles, and at most $3d$ of them can be included in a maximum matching. If $3d$ of these triangles are in the matching, at most d of the top triangles could be included. Suppose that $d-\delta$ top triangles are in the matching where $\delta > 0$ and that all rings are matched in the same way. Then we will get a matching of the ring of trees of size at most $K \cdot 3d +$ (# matched ring triples) + (# matched tree triples in trees connected to matched ring triples) + (# matched tree triples in trees connected to unmatched ring triples) which is less than

$$3dK + (d-\delta)K + (d-\delta)\frac{K-1}{3} + (d+\delta)\frac{2K-2}{3} = d(5K-1) - \delta\frac{2K+1}{3}$$

if $K = 2^\alpha$ and α is even, and less than

$$3dK + (d-\delta)K + (d-\delta)\frac{K-2}{3} + (d+\delta)\frac{2K-1}{3} = d(5K-1) - \delta\frac{2K-1}{3}$$

if $K = 2^\alpha$ and α is odd.

Obviously $\delta = 0$ gives us the maximum matching with $d(5K-1)$ triples. $\delta = 1$ gives us a matching of the structure which is at least $\frac{2K-1}{3}$ triples smaller.

If more than d top triangles are in the matching, say $d+\delta$, we can at most include $4d-1$ ring triangles, so (# matched ring triples) + (# matched tree triples in trees connected to matched ring triples) + (# matched tree triples in trees connected to unmatched ring triples) which is less than

$$(4d-1)K + (d+\delta)\frac{K-1}{3} + (d-\delta)\frac{2K-2}{3} = d(5K-1) - K - \delta\frac{K-1}{3}$$

if $K = 2^\alpha$ and α is even, and less than

$$(4d-1)K + (d+\delta)\frac{K-2}{3} + (d-\delta)\frac{2K-1}{3} = d(5K-1) - K - \delta\frac{K+1}{3}$$

if $K = 2^\alpha$ and α is odd, which is at least $\frac{4K-1}{3}$ less than the maximum matching.

We have d receptor elements connected to the roots of some of the trees. Which root a receptor element is connected to depends on to which clauses the corresponding variable belongs and whether it is negated or not in the clauses. We would like the property that the ring of trees is in either true or false mode to be more valuable than the inclusion of any number of the connected receptor elements in the matching. Assume that we sacrifice this property and that we include p receptor elements instead. From these p triples one can always choose $\lceil \frac{p}{2} \rceil$ triples which occur at even distances from each other (corresponding to the fact that one of u or $\overline{u}$ appears in at least $\lceil\frac{p}{2}\rceil$ clauses). Thus we can always obtain a matching with $d(5K-1)+\lceil\frac{p}{2}\rceil$ triples without sacrificing the aforementioned property. We want that

$$d(5K-1)+\left\lceil\frac{p}{2}\right\rceil > d(5K-1)-\frac{2K-1}{3}+p \text{ for } p\le d \Leftarrow$$

$$\Leftarrow \frac{1}{3}(2K-1) > \left\lfloor\frac{p}{2}\right\rfloor \text{ for } p\le d \Leftarrow K > \frac{3}{2}\left\lfloor\frac{d}{2}\right\rfloor+\frac{1}{2}.$$

This is true because

$$\begin{aligned} K &= 2^{\lfloor\log(\frac{3}{2}B+1)\rfloor} \ge 2^{\lfloor\log(\frac{3}{2}d+1)\rfloor} = 2^{\lfloor\log(\frac{3}{4}d+\frac{1}{2})\rfloor+1} > 2^{\log(\frac{3}{4}d+\frac{1}{2})} = \\ &= \frac{3}{4}d+\frac{1}{2} \ge \frac{3}{2}\left\lfloor\frac{d}{2}\right\rfloor+\frac{1}{2}. \end{aligned}$$

Given a matching of the whole structure we can, as seen above, modify it to an at least as big matching where every ring of trees is in either true or false mode. By setting the variables of the 3Sat problem as the modes indicate we will get an approximate solution of this problem which does not differ from the optimal solution more than the size of the matching differs from the size of the optimal matching.

This is one of the two properties of an L-reduction. The other is that the optimum for the matching problem $opt(f(I))$ is bounded by a constant times the optimum for the 3Sat problem $opt(I)$.

$$\begin{aligned} opt\big(f(I)\big) &= \sum_{i=1}^{n}\big(d_i(5K-1)\big)+opt(I) \le \Big(\sum_{i=1}^{n} d_i\Big)\big(5K-1\big)+opt(I) \le \\ &\le 3m\big(5\cdot(\frac{3}{2}B+1)-1\big)+opt(I) \le (45B+25)\,opt(I) \end{aligned}$$

because $opt(I) \ge \frac{m}{2}$ (it is always possible to satisfy at least half of the clauses in I).

Class 2. H is connected and contains two or more maximum-size 2-connected components.

We first consider the special case where H is a path with three nodes. As both generator elements and tree elements we use gadgets, described in Figure 5.12. A receptor element is just a 3-node path, see Figure 5.13. Examples of a ring and a tree are shown in Figure 5.14.

We can always modify a matching of the structure so that every 3-node path is contained totally within a gadget or a receptor, without changing the size of the matching, in the following way. Since every 3-node path between two elements will isolate a leaf in one of the elements, we can move the matched 3-node path one step and use the isolated leaf instead. Next we rearrange (and complete when needed) the matchings inside every gadget in order to obtain one of the matchings in Figure 5.15. This is always possible and easy to do. In these two matchings all special nodes in the gadget are matched in the same way and we can look at the gadgets just as triangles. Thus we are back to the special case in class 1, and we can see from that proof that this special case is Max SNP-hard. The optimal solution of the matching problem

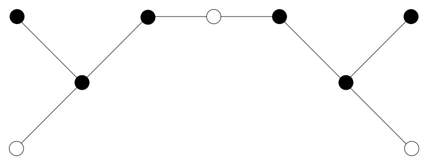

Figure 5.12: *A* gadget*, which can be used as a generator element or a tree element.*

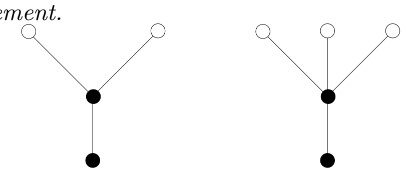

Figure 5.13: *A receptor with two elements and a receptor with three elements.*

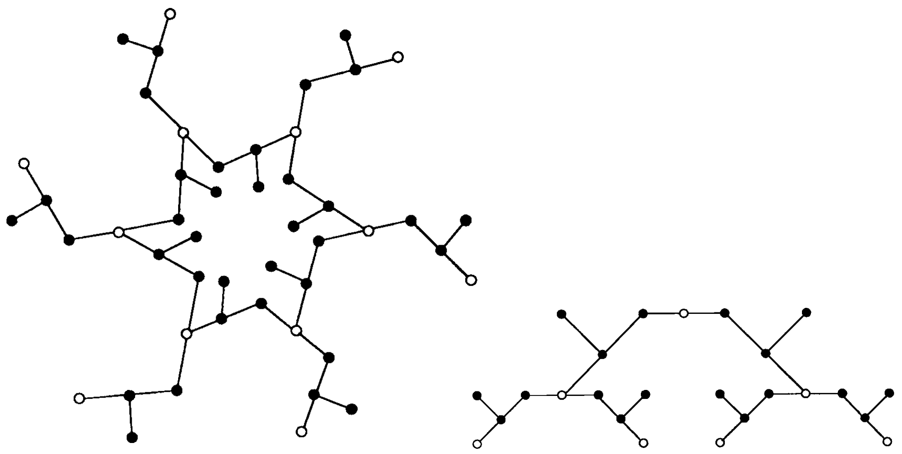

Figure 5.14: *A generator in class 2 with $d = 3$ and a tree with $K = 4$.*

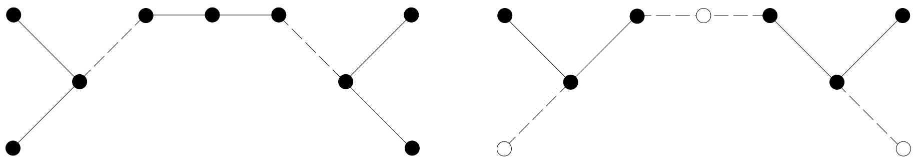

Figure 5.15: *The two good ways to match a gadget. The continuous lines correspond to edges included in the matching.*

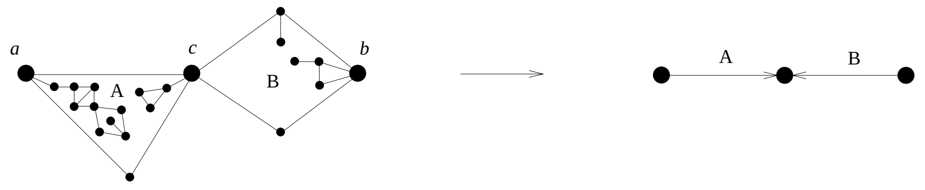

Figure 5.16: *The representation of H with the two components A and B as two directed edges A and B.*

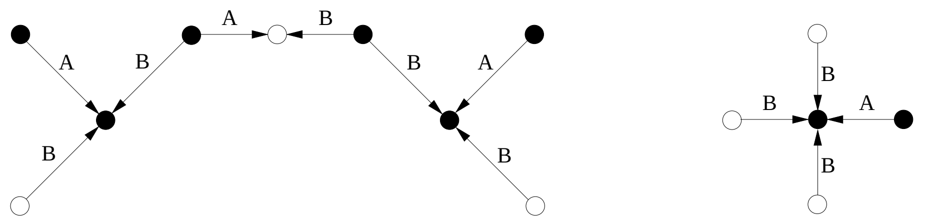

Figure 5.17: *A gadget and a receptor element in class 2, general case.*

is here bounded by the constant $72B + 28$ times the optimal solution of the satisfiability problem.

Class 2, general case.

Let H be any connected graph in which there are at least two maximum-sized 2-connected components. Let $\overline{\text{H}}$ be one of the maximum-size 2-connected components in H. In the same way as in [10] we select a cut point c which is contained in $\overline{\text{H}}$ and which separates it from the rest of the maximum-size 2-connected components in H. This may not be possible for every $\overline{\text{H}}$, but there is always a choice of $\overline{\text{H}}$ such that c can be selected. Let B be the union of $\{c\}$ and the connected component containing $\overline{\text{H}} - \{c\}$ which is cut off by c, and let A be the union of $\{c\}$ and the rest of H. Choose a node b from $\overline{\text{H}} - \{c\}$. Let C be the 2-connected component in A which contains c. If C is one of the maximum-size 2-connected components then let a be any node in C but c. Otherwise let a be a node in C which separates $\overline{\text{H}}$ from a maximum-size 2-connected component in A.

Now H can be represented as something reminiscent of a 3-path with the three nodes a, c and b, see Figure 5.16. Using this similarity we can construct generator elements, tree elements and receptor elements as in the special case, see Figure 5.17, where the edges A and B are represented as arrows, pointing towards the c node. The resulting graph has degree at most $\max\{\deg(\text{H}), 4\deg(a), 4\deg(b), 4\deg(c)\}$.

We now have to see that a matching can be modified in such a way that every matched copy of H covers exactly one A and one B edge, and does not spread across different elements. This is proved for a similar structure in [10] in the following way.

Suppose that we in the matching have a copy D of H which utilizes two or fewer of the a, b and c nodes. If a c node which corresponds to a leaf copy

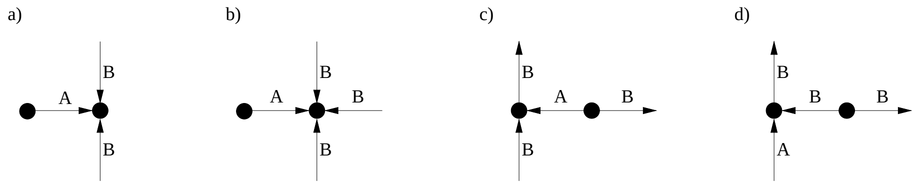

Figure 5.18: *Possible cases for occurrences of H using two or less of the a, b and c nodes.*

of A is used, then the rest of the leaf A cannot be used by any other copy of H and we may use the whole of A in D. By inspection of the structure we see that D must occur in one of the four ways shown in Figure 5.18. In the first three cases a, b and c in Figure 5.18, there is one less copy of $\overline{\mathrm{H}}$ available than is needed to make a copy of H. Hence a copy of H cannot occur in this fashion. In the last case (Figure 5.18d) there are just enough maximum-size 2-connected components available (provided that a is not in a maximum-size 2-connected component of H – otherwise we are done) but there are not enough nodes available which are in 2-connected components separating maximum-size 2-connected components. This is because all such nodes in H are contained in A and cannot be replaced by nodes in B. Since a is such a node, it must be used in order to complete a copy of H. Hence H cannot be contained in any of the structures shown in Figure 5.18 and hence we can always modify a matching in such a way that every matched copy of H covers exactly an A and a B edge.

Now we just rearrange the matchings in every gadget, as in the special case, once again obtaining the structure of the special case in class 1.

Class 3. H is not connected.

Consider the set of maximum-size connected components in H. Let H_1 be one of the connected components in this set which have the largest number of edges. First we look at the case when H_1 is unique in H. We construct the normal structure, as described above, with H_1 as the connected graph, and call the structure S. S will have a maximum H_1-matching of size not exceeding $3m\big(5(\frac{3}{2}B+1)-1\big)+m=\big(\frac{45}{2}B+13\big)m$ if H_1 is in class 1 and $5\cdot 3m\big(2(\frac{3}{2}B+1)-1\big)+m=(45B+16)m$ if H_1 is in class 2. Add this number of copies of $\mathrm{H}-\mathrm{H}_1$ to S. Now it is possible to match as many copies of H as the number of matched H_1 in S. Since H_1 is a unique maximum-size connected component, we cannot use the added components to match H_1. Therefore, given a matching of the new structure, we first move every matched copy of a small component to its "right place", that is to one of the added components which are isomorphic to the component. Then only copies of H_1 are left in S and we proceed as in the proof of class 1 or 2.

If H_1 is not unique, we proceed in the following way. Suppose there are i isomorphic maximum-size connected components H_1, H_2,..., H_i in H. We construct the normal structure, but duplicate the whole structure $i-1$ times so we have i identical substructures. Then we add $\big\lfloor m\big(\frac{45}{2}B+13\big)\big\rfloor$ or $\big\lfloor m(45B+16)\big\rfloor$ copies of $\mathrm{H}-\sum_{j=1}^{i}\mathrm{H}_j$, depending on whether H_1 is in class 1 or 2, and proceed as in the first case. If we have a matching which is δ from the optimal matching we can find a solution of the 3SAT problem which is less than δ/i from the optimal solution. Thus we have an L-reduction. □

By inspection of the constructed H-matching instances we see that they are instances of induced H-matching as well, with the same properties. Thus, using the same proof we have the following theorem.

Theorem 5.9 *Maximum bounded induced* H*-matching is* MAX SNP*-hard for any* H *with three or more nodes in some connected component.*

Theorem 5.10 *Both the problems* MAX H-MATCHING $-B$ *and* MAX INDUCED H-MATCHING $-B$ *are* $\overline{\text{MAX SNP}}$*-complete for any connected* H *with three or more nodes.*

PROOF Theorem 5.8 and 5.9 say that MAX H-MATCHING $-B$ and MAX INDUCED H-MATCHING $-B$ are MAX SNP-hard when H is connected and contains at least three nodes. We will show that the problems are in $\overline{\text{MAX SNP}}$ by reducing them to MAX IND SET $-B$ which is in MAX SNP.

In order to show this we define a polynomial-time transformation from the problem MAX H-MATCHING $-B$ to MAX IND SET $-B$. We construct the instance graph $G_I = \langle V_I, E_I \rangle$ of the independent set problem from the H-matching graph $G = \langle V, E \rangle$ in the following way. For every possible subset of V which has the same number of nodes as H, and which induces a subgraph of G which contains H (or in the induced H-matching problem: which is isomorphic to H) we construct a node in V_I. We have an edge $e \in E_I$ between two nodes v_1 and v_2 in V_I if and only if the two corresponding subsets of V are not disjoint. This means that every H-matching of G is an independent set of G_I of the same size and vice versa. This is a polynomial time transformation since H is fixed and connected, and the degree of G is bounded by B. The degree of G_I is the maximum number of H-copies in G which are not node-disjoint to any copy of H in G. This degree is bounded by a function of B and H. Therefore the transformation is an L-reduction. □

Chapter 6

Maximum common subgraph problems

6.1 Introduction

The Subgraph isomorphism problem is a famous NP-complete problem. It is one of the first problems mentioned in *Computers and Intractability* by Garey and Johnson [35]. Given two graphs the problem is to decide whether the second graph is isomorphic to any subgraph of the first graph. The problem is shown to be NP-complete by the following simple reduction from the Clique problem. Let the first input graph to Subgraph isomorphism be the input graph to Clique and let the second input graph be a K-clique where K is the bound in the Clique problem. Now the K-clique is isomorphic to a subgraph of the first graph if and only if there is a clique of size K or more in the graph.

A related optimization problem is called Maximum common subgraph. In this problem we are given two graphs and we want to find the largest subgraphs which are isomorphic. The corresponding decision problem was shown to be NP-complete by Garey and Johnson using the same reduction as above [35]. The approximation properties of various versions of this problem are studied in this chapter.

6.2 Maximum common induced subgraph

The *maximum common induced subgraph problem* (Max CIS) takes two undirected graphs $G_1 = \langle V_1, E_1 \rangle$ and $G_2 = \langle V_2, E_2 \rangle$ as input. The problem is to find ${V_1}' \subseteq V_1$ and ${V_2}' \subseteq V_2$ such that $G_1 |_{{V_1}'}$ and $G_2 |_{{V_2}'}$ (the graphs induced by ${V_1}'$ and ${V_2}'$) are isomorphic and $\left|{V_1}'\right| = \left|{V_2}'\right|$ is as large as possible. The isomorphism between $G_1 |_{{V_1}'}$ and $G_2 |_{{V_2}'}$ is expressed as a bijective function $f : V_1' \to V_2'$ such that for all $v_1, v_2 \in V_1$ we have $(v_1, v_2) \in E_1 \Leftrightarrow (f(v_1), f(v_2)) \in E_2$. We say that v *is matched with* $f(v)$ and that $f(v)$ *is matched with* v.

The *maximum bounded common induced subgraph problem* (Max CIS $-B$) is the same problem but with restricted space of input instances: $G_1 = \langle V_1, E_1 \rangle$

and $G_2 = \langle V_2, E_2 \rangle$ have degree bounded by a constant B.

6.2.1 Approximation of maximum bounded common induced subgraph

Theorem 6.1 *Maximum bounded common induced subgraph* (Max CIS $-B$) *can be approximated within $B+1$.*

Proof We use that independent sets of the same size are always isomorphic. The following trivial algorithm finds an independent set V_1' in the graph $G_1 = \langle V_1, E_1 \rangle$.

- $V_1' \leftarrow \emptyset$
- Pick nodes from V_1 in any order and add each node to V_1' if none of its neighbours are already added to V_1'.

This algorithm will create a set of size $|V_1'| \geq |V_1|/(B+1)$ because for each node in V_1 either the node itself or one of its at most B neighbour nodes must be in V_1'.

Applying the algorithm to G_1 and G_2 gives us two independent sets V_1' and V_2'. If they are of different size, remove nodes from the largest set until they have got the same size. These sets are a legal solution of the problem of size at least $\min(|V_1|, |V_2|)/(B+1)$. Since the optimal solution has size at most $\min(|V_1|, |V_2|)$ the algorithm approximates the problem within the constant $B+1$. □

Theorem 6.2 Max CIS $-B$ *is* $\overline{\text{Max SNP}}$*-hard when* $B \geq 25$.

Proof The problem Max 3Sat -6, where each variable is in at most six clauses is known to be Max SNP-complete [88]. We assume that each variable x_i occurs both as x_i in some clause and as $\overline{x}_i$ in some other clause, that no clause is trivially satisfied (e.g. $x_i \vee \overline{x}_i$) and that there are more clauses than variables. The problem is still Max SNP-complete under these assumptions. We will show that there is an L-reduction f_1 from this problem to Max CIS $-B$. Let $U = \{x_1, x_2, \ldots, x_n\}$ be the variables and $C = \{c_1, c_2, \ldots, c_m\}$ be the clauses of the input instance.

f_1 takes the sets U and C and constructs a Max CIS $-B$ instance (the graphs G_1 and G_2) in the following way. G_1 and G_2 are similar and consist of $6n$ *literal nodes* (six for each variable), $18m$ *clique nodes* (18 for each clause) and a number of *clause nodes*. G_1 has $7m$ clause nodes (seven for each clause) and G_2 has m clause nodes (one for each clause). The clique nodes are connected in 18-cliques (m in each graph). In both graphs the six literal nodes for a variable x_i are connected in two 3-cliques — one 3-clique we call the x_i *clique* and the other 3-clique we call the $\overline{x}_i$ *clique*.

In G_2 each clause node is connected with one of the clique nodes in the corresponding 18-clique and with all the literal nodes corresponding to the at most three literals which are contained in the corresponding clause in the Max 3Sat -6 problem. This completes the description of graph G_2. G_1 has edges between each pair of literal nodes which corresponds to the same variable (i.e. building a 6-clique). Finally there are some edges from the clause nodes to the clique nodes and literal nodes in G_1. Number the seven clause nodes of clause c_j from 1 to 7 and the 18 clique nodes in the corresponding clique from 1 to 18. Now add edges between clause node i and clique node i for i from 1 to 7. Call the three literal 3-cliques corresponding to the three literals in c_i A, B and C. Add edges between clause node 1 and each node in A, 2 and B, 3 and A, 3 and B, 4 and C, 5 and A, 5 and C, 6 and B, 6 and C, 7 and A, 7 and B, 7 and C. If c_i only has two literals just add the edges from clause nodes 1, 2 and 3. If c_i only has one literal just add three edges, between node 1 and the literal 3-clique. See Figure 6.1 for an example.

The idea is that a truth assignment shall be encoded in the subgraph problem by including the corresponding literal 3-cliques in the subgraphs. For example, if x_4 is true then the literal 3-clique x_4 is in the subgraphs, and if x_4 is false then the literal 3-clique $\overline{x}_4$ is in the subgraphs. The included literal 3-cliques of G_1 and G_2 are matched with each other. A clause node in graph G_2 is included in the subgraph iff it is satisfied by the truth assignment. If a clause node in G_2 is included then it is matched with one of the corresponding seven clause nodes in G_1, namely with the node which is connected with exactly those literals in the clause which are true in the truth assignment. All the clique nodes are included in the subgraphs and are matched with each other clause-wise.

A solution of the Max 3Sat -6 problem with k satisfied clauses will be encoded as two subgraphs (of G_1 and G_2), each with k clause nodes, $3n$ literal nodes and $18m$ clique nodes. Since a literal can be contained in at most five clauses at the same time, a literal node in the graph G_1 has degree at most $5 \cdot 4 + 5 = 25$, a clause node has degree at most 4 and a clique node has degree 17 or 18. In G_2 a literal node has degree at most 7, a clause node at most 4 and a clique node at most 18. Thus $B \geq 25$.

We will now prove that the maximum solution of the Max 3Sat -6 problem will be encoded as a maximum solution of the Max CIS $-B$ problem, and that given a solution of the Max CIS $-B$ problem we can in polynomial time find an at least as good solution which is a legal encoding of a Max 3Sat -6 solution.

Suppose we have any solution of the Max CIS $-B$ problem, that is an induced subgraph of G_1, an induced subgraph of G_2 and a matching between each node in the G_1 subgraph and the corresponding node in the isomorphic G_2 subgraph.

- First we would like to include all clique nodes in the subgraphs and match each 18-clique in the first graph with some 18-clique in the second. Observe that, besides the clique nodes, no node in the graph G_2 is in a clique of size five or more. This means that if five or more clique nodes in the same 18-clique are included in the subgraph of G_1, then they must be matched with clique nodes in the other subgraph. In the same way we see that besides the clique nodes, no node in the graph G_1 is in a

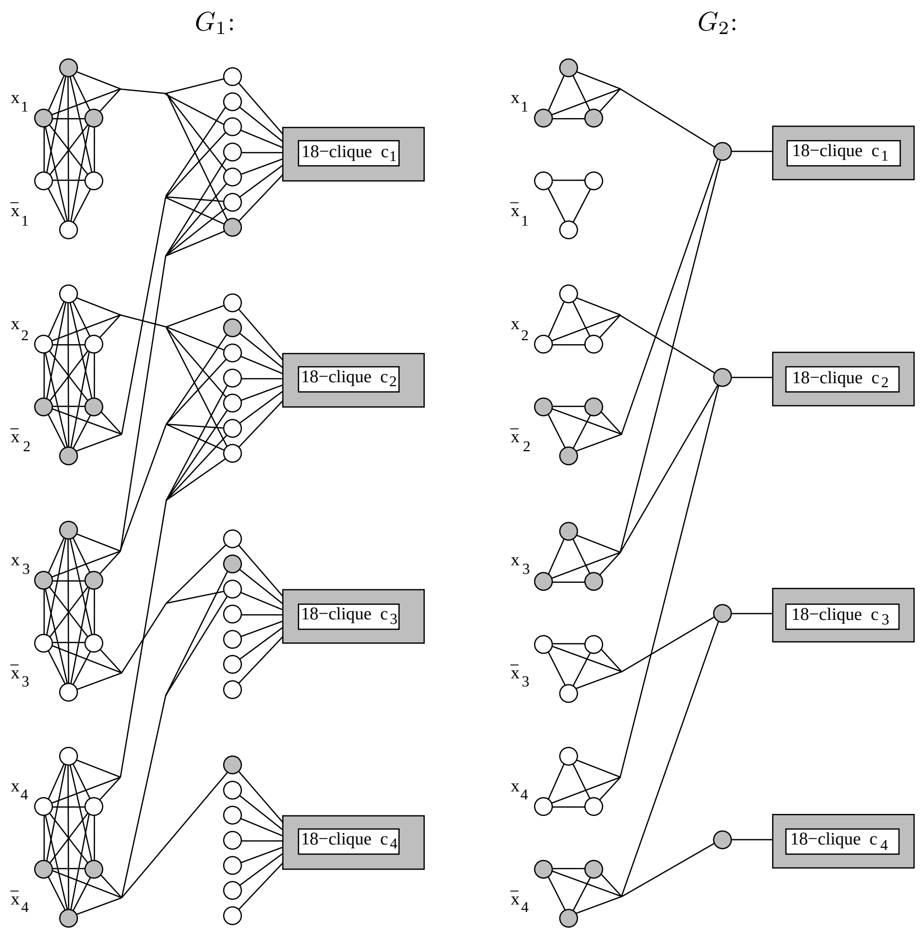

Figure 6.1: *The constructed instance of the* Max CIS $-B$ *problem for the* Max 3Sat -6 *input* $U = \{x_1, x_2, x_3, x_4\}, C = \{(x_1 \vee \overline{x}_2 \vee x_3), (x_2 \vee x_3 \vee x_4), (\overline{x}_3 \vee \overline{x}_4), (\overline{x}_4)\}$*. One of the possible maximum common subgraphs is formed by including the shaded nodes.*

clique of size seven or more, so if seven or more clique nodes in the same 18-clique are included in the subgraph of G_2, then they must be matched with clique nodes in the subgraph of G_1.

In each 18-clique in G_1 which has $5 \leq p < 18$ nodes included in the subgraph, we add the rest of the 18-clique to the subgraph and remove every clause node which is connected to an added clique node. We have added $18-p$ clique nodes and removed at most the same number of clause nodes.

The matched 18-clique in G_2 must also have p clique nodes in the subgraph. Add the remaining $18 - p$ clique nodes and remove the nodes which are matched with the removed clause nodes in G_1. It is easy to see that we now can match the two 18-cliques with each other without problems.

Perform the same operation for each 18-clique in G_1 which has at least five but not all nodes included in the subgraph. The resulting subgraphs are at least as large as the original subgraphs, and they are still isomorphic.

Now every 18-clique in G_1 either has all nodes in the subgraph or at most four nodes in the subgraph. For each 18-clique in G_1 with $0 \leq p \leq 4$ nodes we do the following.

We add $18 - p$ clique nodes to the subgraph of G_1 and remove every clause node which is connected to a clique node in the current 18-clique. We have added $18-p$ nodes and removed at most 7. In G_2 we choose one 18-clique with $0 \leq q \leq 6$ nodes in the subgraph and add the remaining $18 - q$ clique nodes. We remove the p nodes which are matched with the old clique nodes in the first subgraph and the at most 7 nodes which are matched with the removed nodes in the first subgraph. Furthermore we have to remove the q nodes in G_1 which are matched with the q old clique nodes in G_2. If the clause node in G_2 (which is a neighbour to the current 18-clique in G_2) is included in the second subgraph we remove it and its matched node in G_1. We have now added $18 - p \geq 14$ nodes to the first subgraph and removed at most $7+q+1 \leq 14$. We have added $18-q \geq 12$ nodes to the second subgraph and removed at most $7 + p + 1 \leq 12$. As before, since the 18-cliques are now separate connected components we can match them with each other without problems.

The two subgraphs are still isomorphic, and thus a solution of the problem. They are at least as large as before, but now all clique nodes are included in the subgraphs and are matched with each other (but they are not necessarily matched in order yet).

- We observe that in each 7-group of clause nodes in G_1 at most one node is in the subgraph. The explanation is that every clause node is in contact with an 18-clique, which is completely in the subgraph, but in the subgraph of G_2 only one node can be in contact with each 18-clique (namely the corresponding clause node). Hence a structure with two or more nodes connected to an 18-clique cannot be isomorphic with any structure in G_2. Furthermore we can see that clause nodes in one of the subgraphs can only be matched with clause nodes in the other subgraph and, since all clique nodes are matched with clique nodes, literal nodes must be matched with literal nodes.

- We would like to change the subgraphs so that each literal 3-clique is either totally included in a subgraph or is not included at all. Furthermore at most one of the two literal 3-cliques in G_1 corresponding to the same variable may be included in the subgraph. Suppose that there is (at least) one node in the literal 3-clique x which is included in the subgraph of G_1. Let y be the literal node in the subgraph of G_2 with which this node is matched. We examine two cases.

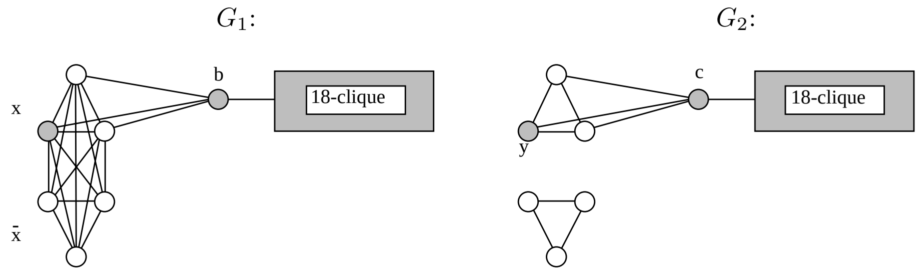

Figure 6.2: *The structure in case 1.*

Case 1. At least one of the clause nodes connected to x is included in the subgraph of G_1. Let b be one of these clause nodes and let c be the clause node in G_2 with which b is matched. See Figure 6.2. First we shall see that no node in the $\overline{x}$ literal 3-clique can be in the subgraph. This is because the nodes in the $\overline{x}$ clique are connected to the nodes in the x clique but not to b (since we have assumed that x and $\overline{x}$ cannot be in the same clause), but in G_2 there are no literal nodes which are connected to y but not to c. Since all three nodes in the x clique have the same connections to the environment in G_1 and all three nodes in the literal 3-clique containing y have the same environment in G_2 we still have isomorphic subgraphs if we include the whole x 3-clique in the subgraph of G_1 and the whole 3-clique containing y in the subgraph of G_2.

Case 2. None of the clause nodes connected to x are in the subgraph of G_1. If one or more nodes in $\overline{x}$ are in the subgraph of G_1 then none of the clause nodes which are connected to $\overline{x}$ can be in the subgraph, since we in that case would be in case 1 with the clause node as b and $\overline{x}$ as x. Thus we have a separate k-clique (with $1 \leq k \leq 6$) of literal nodes which by the above must be matched with a separate k-clique of literal nodes in G_2. In G_2 the largest possible clique of literal nodes is of size 3. Therefore the only possible cases are $1 \leq k \leq 3$. We remove those k nodes and instead include the whole x 3-clique in the subgraph of G_1 and the whole 3-clique containing one of the matched nodes in the subgraph of G_2.

In both cases we get a subgraph of G_1 where each literal 3-clique is either totally included in a subgraph or is not included at all and where both of the literal 3-cliques corresponding to the same variable are never included in the subgraph of G_1 at the same time.

- We now forget about the subgraph of G_2 and concentrate on the subgraph of G_1. It contains all clique nodes, at most one of each 7-group of clause nodes and at most one of each pair of literal 3-cliques. First we will include literal nodes so that every pair of literal 3-cliques has exactly one of the 3-cliques in the subgraph. We will have to remove some of the clause nodes, but the subgraph should contain at least as many nodes as before. Then we reorder the clause nodes to form a legal encoding of a Max 3Sat −6 solution.

1. Suppose there are k variables which do not have any of their literal 3-cliques in the subgraph and that there are j clauses which contain these variables. We know that each variable can occur in at most six clauses, thus $j \leq 6k$. Using a simple algorithm (see for example [48]) we can give values to the k variables so that at least half of the j clauses are satisfied. We first remove all of the j clause nodes which are in the subgraph from the subgraph and then include one clause node for each clause which was satisfied by the algorithm and the literal 3-cliques corresponding to the k variable values. We will then have removed at most j nodes and included at least $3k + j/2 \geq j/2 + j/2 = j$ nodes.
2. In order to create a subgraph of G_1 which is a legal encoding of a Max 3Sat −6 solution we may have to substitute some clause nodes in the subgraph for other clause nodes in the same 7-groups. Every clause node in the resulting subgraph should have connections to exactly those literal 3-cliques which are included in the subgraph of G_1 and correspond to literals in the corresponding clause. It is easy to see that this operation is always possible.

- As the subgraph of G_2 choose nodes as shown in the description of the encoding above. This is possible since the subgraph of G_1 is a legal encoding of a Max 3Sat −6 solution. The isomorphic matching is then trivial.

We have now shown that every solution of the Max CIS $-B$ problem can be transformed to an at least as large solution which is a legal encoding of a Max 3Sat −6 solution. Moreover this transformation can be done in polynomial time.

If the optimal number of satisfied clauses is r and we do not have more variables than clauses then the optimal number of nodes in a subgraph is $3n + 18m + r \leq (3+18)m + r \leq 21 \cdot 2r + r = 43r$, since we can always satisfy at least half of the clauses. Thus the transformation f_1 of a Max 3Sat −6 problem to a Max CIS $-B$ problem, where $B \geq 25$, is an L-reduction with $\alpha = 43$ and $\beta = 1$. □

6.2.2 Approximation of maximum unbounded common induced subgraph

Theorem 6.3 *There is a reduction from* Max CIS *to* Max Clique *which is an L-reduction and an S-reduction with node amplification* n^2*.*

Proof From the input graphs G_1 and G_2 we form a *derived graph* $G = \langle V, E \rangle$ in the following way (due to Barrow and Burstall [9]).

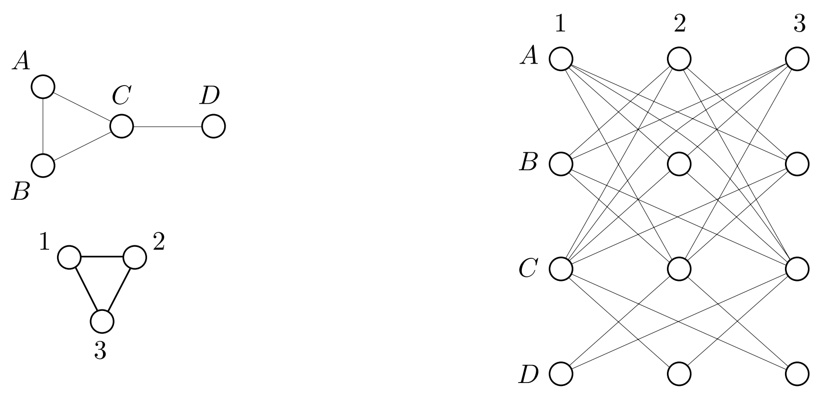

Figure 6.3: *An example of the transformation from* Max CIS *to* Max Clique *in Theorem 6.3. The maximum size cliques in the right graph have size three, e.g. (A1,B3,C2) which corresponds to a common subgraph matching between node A and 1, B and 3, C and 2 in the left graphs.*

Let $V = V_1 \times V_2$ and call V a set of *pairs*. Call two pairs $\langle v_1, v_2\rangle$ and $\langle w_1, w_2\rangle$ *compatible* if $v_1 \neq w_1$ and $v_2 \neq w_2$ and if they preserve the edge relation, that is there is an edge between v_1 and w_1 if and only if there is an edge between v_2 and w_2. Let E be the set of compatible pairs. See Figure 6.3.

A k-clique in the derived graph G can be interpreted as a matching between two induced k-node subgraphs. The subgraphs are isomorphic since the compatible pairs preserve the edge relation.

Thus, if we have a polynomial time approximation algorithm for the Max Clique problem which finds a solution within p we can apply it to the derived graph and use the answer to get an approximate solution for the Max CIS problem of exactly the same size. Since the maximum clique corresponds to the maximum common induced subgraph this yields a solution within p for the Max CIS problem and we have an L-reduction from Max CIS to Max Clique with $\alpha = \beta = 1$.

For example we can use the Max Clique approximation algorithm by Boppana and Halldórsson [18]. This algorithm will, for a graph of size n, find a clique of size at least $O((\log n)^2/n)$ times the size of the maximum clique.

Note that in spite of the size of the optimal solution being preserved by the reduction, the size of the problem instance is increased. If the two input graphs of the Max CIS problem contain m_1 and m_2 nodes respectively, the constructed graph will contain $m_1 m_2$ nodes and the algorithm will only guarantee a common induced subgraph of size $O((\log m_1 m_2)^2/(m_1 m_2))$ times the size of the maximum common induced subgraph.

Thus the reduction is an S-reduction with node amplification n^2. □

Theorem 6.4 *There is a reduction from* Max Clique *to* Max CIS *which is an L-reduction and an S-reduction without node amplification.*

Proof The reduction is the same as the one Garey and Johnson used to prove that Subgraph isomorphism is NP-complete. The Max Clique input is given as a graph G. Let the first of the Max CIS graphs G_1 be this graph. Let G_2 be a $|V_1|$-clique, that is a complete graph with the same number of nodes as G_1. The constructed graphs have the same number of nodes as the input graph.

Every induced subgraph in G_2 is a clique. Thus each common induced subgraph is a clique. The optimal solution of the Max Clique problem is a clique of size at most $|V|$, and this clique is also the largest clique in G_1 and is therefore the largest common induced subgraph.

In order to prove that this is an L-reduction we also have to show that given a solution of the Max CIS problem we in polynomial time can find an at least as good solution of the Max Clique problem. But since every common subgraph is a clique we can directly use the subgraph as a solution to the Max Clique problem. The solutions have the same size. $\square$

The Max Clique problem is actually the same problem as Max Ind Set on the complementary graph. Thus the unbounded Max CIS problem is hard to approximate, see Section 4.11.

6.3 Maximum common edge subgraph

The *maximum common edge subgraph problem* (Max CES) also takes two undirected graphs $G_1 = \langle V_1, E_1 \rangle$ and $G_2 = \langle V_2, E_2 \rangle$ as input. This problem differs from the maximum common induced subgraph problem in that the subgraphs do not need to be node induced. The problem is to find $E_1{}' \subseteq E_1$ and $E_2{}' \subseteq E_2$ such that $G_1\left|_{E_1{}'}\right.$ and $G_2\left|_{E_2{}'}\right.$ are isomorphic and $\left|E_1{}'\right| = \left|E_2{}'\right|$ is as large as possible.

The *maximum bounded common edge subgraph problem* (Max CES $-B$) is the same problem where the input graphs have degree bounded by a constant B.

6.3.1 Approximation of maximum bounded common edge subgraph

Lemma 6.5 *A maximum matching of a graph $G = \langle V, E \rangle$ with degree at most B contains at least $|E|/(B+1)$ edges.*

Proof Let ν be the number of edges in the maximum matching and p be the number of nodes in the graph. If there exists a perfect matching, then $p = 2\nu$ and

$$\frac{|E|}{B+1} \leq \frac{p \cdot B/2}{B+1} < \frac{p}{2} = \nu.$$

If $p \geq 2\nu + 1$ the inequality $|E| \leq (B+1)\nu$ follows from [75, theorem 3.4.6]. $\square$

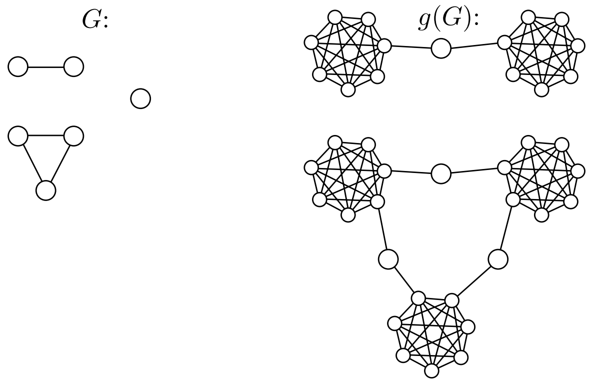

Figure 6.4: *An example of the transformation g in Theorem 6.7.*

Theorem 6.6 *Maximum bounded common edge subgraph* (Max CES $-B$) *can be approximated within $B+1$.*

Proof We use the same idea as in the proof of Theorem 6.1, but create an independent set of *edges* instead. In polynomial time we can find a maximum matching of the graphs. The size of the smallest maximum matching is by Lemma 6.5 at least $\min(|E_1|, |E_2|)/(B+1)$ and the size of the optimal solution is at most $\min(|E_1|, |E_2|)$, so we can approximate this problem too within the constant $B+1$. $\square$

Now we will show that if the degrees of the graphs are bounded then the Max CIS problem is at least as hard to approximate as the Max CES problem.

Theorem 6.7 *There is an L-reduction from the problem* Max CES $-B$ *to the problem* Max CIS$-(2B+3)$.

Proof Let f_3 be the following transformation from Max CES $-B$ to Max CIS $-(2B+3)$. An input instance $\{{G_1}^E, {G_2}^E\}$ of Max CES $-B$ shall be transformed to an instance $\{{G_1}^I, {G_2}^I\}$ of Max CIS $-(2B+3)$. Let ${G_1}^I = g({G_1}^E)$ and ${G_2}^I = g({G_2}^E)$ where g transforms each node with degree greater than zero to a $(2B+3)$-clique and each edge to two edges connected with an *edge node*. The two edges are connected to one node in each of the two $(2B+3)$-cliques corresponding to the end points of the edge in the original graph. This shall be done so that every clique node is connected to at most one edge node. The constructed graphs have degree $2B+3$. See Figure 6.4.

Solutions $\{{E_1}', {E_2}'\}$ to the Max CES $-B$ problem is encoded as solutions $\{{V_1}', {V_2}'\}$ to the Max CIS problem where an edge node is in V_i' iff the corresponding edge is in E_i' and where $\min(|V_1'|, |V_2'|)$ $(2B+3)$-cliques, among them all cliques corresponding to nodes adjacent to edges in E_1' and E_2', are included in each subgraph.

In the same way as in the proof of Theorem 6.2 we will show that given a solution of the Max CIS problem, which is d smaller than the optimal solution, we can in polynomial time find a solution of the Max CES $-B$ problem which is at most d smaller than the optimal solution.

Given solutions V_1' and V_2' we first modify them so that all clique nodes are included. Observe that cliques of size three or more only can be found in the $(2B+3)$-cliques in ${G_1}^I$ and ${G_2}^I$ and thus that a $(2B+3)$-clique in ${G_1}^I$ with k nodes ($k \geq 3$) in V_1' only can be matched with a $(2B+3)$-clique in ${G_2}^I$ with exactly k nodes in V_2' (and vice versa). For each $(2B+3)$-clique in ${G_1}^I$ with k nodes ($3 \leq k < 2B+3$) in V_1' we can add the remaining $2B+3-k$ nodes if we remove all edge nodes in V_1' which are connected to added nodes and perform the same operations on the matched clique in the other graph.

Now every clique either has all nodes in the subgraph or at most two nodes in the subgraph. For each clique in ${G_1}^{I'}$ with $0 \leq p \leq 2$ nodes we do the following (until there are no nonfull cliques left in one of the subgraphs).

We add $2B+3-p$ clique nodes to the subgraph of ${G_1}^I$ and remove every edge node which is connected to a clique node in the current clique (at most B nodes). In ${G_2}^I$ we choose one clique with $0 \leq q \leq p$ nodes in the subgraph. If one of the p nodes in the first subgraph is matched with a clique node in the second subgraph (which is always the case if $p = 2$ because two edge nodes never are connected) we choose this clique. We add the remaining $2B+3-q$ clique nodes and remove every edge node which is connected to a clique node in the current clique (at most B nodes).

If one of the q nodes in the second subgraph is matched with an edge node in the first subgraph we have to remove this edge node from the first subgraph. If one of the p nodes in the first subgraph is matched with an edge node in the second subgraph we have to remove this edge node from the second subgraph.

Furthermore we have to remove the at most B nodes in the first subgraph which are matched with edge nodes which are connected with nodes in the current clique in the second subgraph. And symmetrically we have to remove the at most B nodes in the second subgraph which are matched with edge nodes which are connected with nodes in the current clique in the first subgraph.

We have now added $2B+3-p \geq 2B+1$ nodes to the first subgraph and $2B+3-q \geq 2B+1$ nodes from the second subgraph and removed at most $B+1+B = 2B+1$ nodes from the first subgraph and $B+1+B = 2B+1$ nodes from the second. If we match the cliques with each other the two subgraphs are still isomorphic.

Now every clique node in at least one of the graphs, say in ${G_1}^I$, is included in the corresponding subgraph and are matched with clique nodes in the other subgraph. Therefore the edge nodes in the second subgraph must be matched with edge nodes in the first subgraph. Every edge node in the first subgraph must be matched with an edge node in the second subgraph, because it is adjacent to a $(2B+3)$-clique in the first subgraph, and no clique node in the second subgraph is adjacent to a $(2B+3)$-clique. Thus we have subgraphs which are an encoding of a Max CES $-B$ solution, where an edge node is in the Max CIS subgraph if and only if the corresponding edge is in the Max CES $-B$ subgraph.

If the subgraphs in an optimal solution of the Max CES $-B$ problem contain k edges then the number of nodes in the subgraphs in the optimal

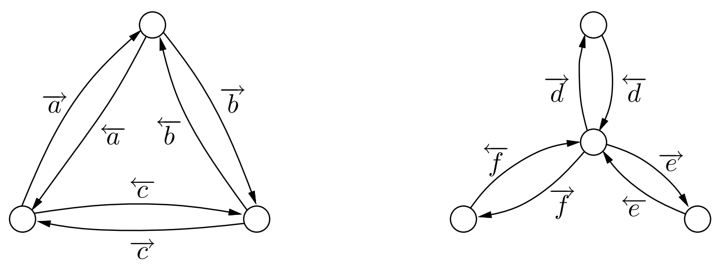

Figure 6.5: *The digraphs resulting from a triangle and a 3-star in Theorem 6.8.*

solution to the Max CIS problem is

$$k+(2B+3)\cdot 2\cdot \min(\left|{E_1}^E\right|, \left|{E_2}^E\right|) \leq k+(2B+3)\cdot 2\cdot (B+1)k = (4B^2+10B+7)k$$

Thus the reduction f_3 is an L-reduction with $\alpha = 4B^2+10B+7$ and $\beta = 1$. □

6.3.2 Approximation of maximum unbounded common edge subgraph

Theorem 6.8 *There is a reduction from* Max CES *to* Max Clique *which is an L-reduction and an S-reduction with node amplification* n^2.

Proof We use the same idea as in the proof of Theorem 6.3 but the pairs are now pairs of directed edges instead of pairs of nodes. Let $V = A_1 \times A_2$, where A_i consists of two directed edges, $\overleftarrow{e}$ and $\overrightarrow{e}$, for each edge $e \in E_i$. We say that two pairs $\langle \overrightarrow{m_1}, \overrightarrow{m_2} \rangle$ and $\langle \overrightarrow{n_1}, \overrightarrow{n_2} \rangle$ are compatible if $\overrightarrow{m_1} \neq \overrightarrow{n_1}$, $\overrightarrow{m_1} \neq \overleftarrow{n_1}$, $\overrightarrow{m_2} \neq \overrightarrow{n_2}$, $\overrightarrow{m_2} \neq \overleftarrow{n_2}$ and they preserve the node relation, that is $\overrightarrow{m_1}$ and $\overrightarrow{n_1}$ are incident to the same node if and only if $\overrightarrow{m_2}$ and $\overrightarrow{n_2}$ are incident to the same node *in the same way*. For example, in Figure 6.5 $\langle \overrightarrow{a}, \overleftarrow{d} \rangle$ is compatible with $\langle \overrightarrow{b}, \overrightarrow{e} \rangle$, $\langle \overrightarrow{b}, \overrightarrow{f} \rangle$, $\langle \overleftarrow{b}, \overleftarrow{e} \rangle$ and $\langle \overleftarrow{b}, \overleftarrow{f} \rangle$ but not with e.g. $\langle \overleftarrow{b}, \overrightarrow{e} \rangle$ or $\langle \overrightarrow{b}, \overleftarrow{e} \rangle$.

A k-clique in the derived graph can be interpreted as a matching between two edge subgraphs with k edges in each subgraph. The subgraphs are isomorphic since the compatible pairs preserve the node relation.

Thus we get an L-reduction from Max CES to Max Clique with $\alpha = \beta = 1$ which we can use to transform a Max Clique approximation algorithm to a Max CES approximation algorithm. As in Theorem 6.3 the reduction has node amplification n^2. □

$$\begin{array}{ccccc}
\textsc{Max 3Sat} -B & < & \textsc{Max Clique} & < & \text{LIP} \\
\wedge & & \wedge \vee & & \wedge \vee \\
\textsc{Max} \text{ CIS } -B & < & \textsc{Max} \text{ CIS} & < & \textsc{Max} \text{ CICS} \\
\vee & & \vee & & \\
\textsc{Max} \text{ CES } -B & < & \textsc{Max} \text{ CES} & &
\end{array}$$

Figure 6.6: *Summary of how the common subgraph problems are related. Here $P_1 < P_2$ means that there is an L-reduction from problem P_1 to problem P_2.*

6.4 Maximum common induced connected subgraph

When we gave an approximation algorithm for the Max CIS $-B$ problem in Theorem 6.1 we constructed a maximal independent set to use as the common subgraph. Somehow this feels like cheating, because an independent set of nodes is usually not the type of common subgraph we want. Perhaps we would rather like to find a big common *connected* subgraph. Unfortunately the Max CIS problem, where we demand that the common subgraph is connected, is *provably* hard to approximate. We will prove that this problem is NPO PB-complete under L-reductions.

In general, for similar graph problems, the demand that the solution subgraph is connected seems to lead to harder problems [107].

Theorem 6.9 Maximum common induced connected subgraph (Max CICS) *is* NPO PB-*complete under L-reductions.*

Proof NPO PB consists of all NPO problems which are polynomially bounded, see Definition 2.16.

We know that Longest induced path in a graph G is NPO PB-complete under P-reductions (see Proposition 4.11) so we will L-reduce this problem to Max CICS in order to show that the latter problem is NPO PB-complete.

Choose G as the first graph G_1 and choose a simple path with $|V|$ nodes as the second graph G_2. We observe that every induced connected subgraph in G_2 must be a simple path. The maximum induced connected subgraph is the longest induced path that can be found in G_1, that is the optimal solution of the LIP problem with input G. Since every non-optimal solution of size c immediately gives a solution of the LIP problem of size c the transformation is an L-reduction with $\alpha = \beta = 1$ and without node amplification. □

In this chapter we have studied the approximability of some maximum common subgraph problems. Figure 6.6 illustrates the situation.

Yannakakis observed in [108] that problems on edges tend to be easier to solve than their node-analogues. We have seen that this is valid for the approximability of the maximum common graph problem as well.

Chapter 7

The travelling salesperson problem

7.1 Introduction

The general travelling salesperson problem (MIN TSP) is the problem of, given a sequence of cities $c_1, c_2, \ldots, c_n$ and distances $d(c_i, c_j)$ for each pair of cities, finding the shortest tour between the cities, that is, finding a permutation π of the cities that minimizes

$$\sum_{i=1}^{n-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(n)}, c_{\pi(1)}).$$

In the symmetric travelling salesperson problem the distances for all pairs of cities satisfy $d(c_i, c_j) = d(c_j, c_i)$. This problem has many applications, for example VLSI chip fabrication. See [65] for an exposition of the problem's applications and history.

An important special case of the symmetric TSP is the Euclidean travelling salesperson problem. In this problem the n cities are given as coordinates in the Euclidean space and the distances are simply the Euclidean distances between the cities.

All the mentioned variants of the TSP are known to be NP-hard [65]. Thus it is almost certain that they cannot be solved in polynomial time. The trivial algorithm which tests all $n!$ permutations takes time $2^{O(n \log n)}$. In this chapter we present an algorithm solving the Euclidean TSP in the plane in time $2^{O(\sqrt{n} \log n)}$ using ring formed separators. Recently Warren Smith found, independently of us, another algorithm which also solves the problem in time $2^{O(\sqrt{n} \log n)}$ [102].

Much attention has been addressed to the question of efficient approximation algorithms. There exist many heuristics for the different variants of TSP [49]. An algorithm by Christofides finds in polynomial time a tour of length at most $\frac{3}{2}$ times the shortest tour possible for the symmetric TSP where the distances obey the triangle inequality [21]. This is still the best approximation performance ratio known.

Papadimitriou and Yannakakis have shown that the TSP with all distances either one or two is $\overline{\text{MAX SNP}}$-hard, and therefore it does not have a polynomial time approximation scheme provided that P $\neq$ NP. It is unknown whether the Euclidean TSP is $\overline{\text{MAX SNP}}$-hard [89].

A result of a quite different nature for the Euclidean travelling salesperson problem where all n cities are in the unit square is that the optimal tour in the worst case is $\alpha\sqrt{n} + o(\sqrt{n})$, where $1.075 \le \alpha \le \sqrt{2}$ [105].

7.2 General TSP

Given a complete undirected graph $G = \langle V, E\rangle$ with non-negative integer edge weights $s : E \to \mathbb{N}$. A solution to the symmetric general travelling salesperson problem is a permutation π of the nodes in the graph. The problem is to find the permutation that minimizes the sum of the edge weights in the tour, that is

$$\sum_{i=1}^{|V|-1} s(v_{\pi(i)}, v_{\pi(i+1)}) + s(v_{\pi(|V|)}, v_{\pi(1)}).$$

MIN TSP is in NPO but the optimal solution is not polynomially bounded, so MIN TSP $\notin$ NPO PB. Garey and Johnson have shown that if MIN TSP could be approximated within a constant, then MIN TSP can be solved optimally in polynomial time and thus P = NP [34].

In fact it is extremely hard to approximate MIN TSP. Orponen and Mannila have shown that MIN TSP is NPO-complete under strict reductions with respect to any cost-respecting measure of approximation quality [83], that is every NPO problem can be strictly reduced to MIN TSP, see Proposition 4.8.

7.3 TSP with triangle inequality

The general travelling salesperson problem is obviously very hard to approximate, but a restricted problem may be easier. Let us demand that the edge weight function must satisfy the triangle inequality, that is

$$\forall i, j, k \in [1..|V|], s(v_i, v_k) \le s(v_i, v_j) + s(v_j, v_k) \text{ if } i \neq j, i \neq k \text{ and } j \neq k.$$

This is called the travelling salesperson problem with triangle inequality or MIN ΔTSP. This problem is relatively easy to approximate as we will see. It can be approximated within a constant, but it cannot be approximated within every constant.

7.3.1 Approximation algorithm by Christofides

Christofides has given a beautiful approximation algorithm for the travelling salesperson problem with triangle inequality. It approximates the optimal solution within 3/2.

Algorithm 7.1 (Christofides [21])
INPUT: A complete undirected graph $G = \langle V, E\rangle$ with non-negative integer edge weights $s : E \to \mathbb{N}$ which satisfy the triangle inequality.
OUTPUT: A travelling salesperson tour of G of length at most 3/2 times the length of the minimum tour.
ALGORITHM:

1. Find a minimum spanning tree of G.
2. Let V' be the set of nodes which have odd degree in the spanning tree. There is always an even number of such nodes.
3. Find a minimum matching of the graph induced by V' and add the edges in the matching to the spanning tree. Now every node in the augmented spanning tree has even degree. Thus the graph is Eulerian.
4. Find an Eulerian tour (a circuit which traverses every edge once) of the augmented spanning tree.
5. Convert the Eulerian tour into a TSP tour by shortcutting nodes which have already been visited.

ANALYSIS: First only consider the graph induced in G by V'. The minimum TSP tour of this graph gives us two matchings of V', simply by taking every second edge in the tour. The lengths of these two matchings can of course not be smaller than the length of the minimum matching and the sum of the lengths of the two matchings is the length of the minimum tour of the induced graph which is less than the length of the minimum tour of the original graph G. Thus the length of the minimum matching is less than half the length of the minimum tour of G.

The sum of the length of the edges in the spanning tree is less than the length of the minimum tour. The length of the Eulerian tour in step 4 is therefore less than 3/2 times the length of the minimum TSP tour.

Because of the triangle inequality the shortcuts in step 5 can only make the tour shorter. We have found that the length of the constructed tour is at most 3/2 times the minimum tour.

Each step in the algorithm can be done in polynomial time.

7.3.2 TSP with distances one and two

A very restricted variant of TSP is the travelling salesperson problem with distances one and two (MIN $(1,2)$TSP). In this problem the edge weight function in the complete graph can only take values one and two, that is $s : E \to \{1, 2\}$. Observe that the sum of the lengths of two edges is at least two, i.e. greater than or equal to the length of one edge. Thus this edge weight function satisfies the triangle inequality and thus any negative result obtained on the approximability of MIN $(1,2)$TSP will be a negative result on the approximability of MIN ΔTSP.

Proposition 7.1 (Papadimitriou and Yannakakis [89])
MIN $(1,2)$TSP *is* $\overline{\text{MAX SNP}}$*-hard.*

Thus the general polynomially bounded travelling salesperson problem with triangle inequality is $\overline{\text{MAX SNP}}$-hard.

Papadimitriou and Yannakakis also gave an algorithm which approximates MIN $(1,2)$TSP within $7/6$ and therefore is better than Christofides's algorithm in this case.

7.3.3 2-dimensional Euclidean TSP

Another special case of the travelling salesperson problem with triangle inequality is the k-dimensional Euclidean TSP. In this problem the n cities are situated in a k-dimensional Euclidean space. The input instance consists of n integer k-dimensional vectors, the coordinates of the cities.

The distance between two cities at coordinates $(v_1, \ldots, v_k)$ and $(u_1, \ldots, u_k)$ is the discretized Euclidean length

$$\left\lceil \sqrt{\sum_{i=1}^{k} (v_i - u_i)^2} \right\rceil$$

The decision problem version of the Euclidean TSP is NP-complete for $k \geq 2$ [85].

In spite of the efforts made by many researchers over the years, no algorithm approximating the Euclidean TSP better than Christofides's algorithm has been found. This is surprising, since of all the characteristics of Euclidean space, Christofides's algorithm only uses the triangle inequality. Therefore it seems likely that the problem cannot be approximated much better, at least not within every constant, so it may be possible to prove the problem to be $\overline{\text{MAX SNP}}$-hard.

While working on this we found an algorithm which solves the 2-dimensional Euclidean TSP exactly. The algorithm is of course exponential but it is much faster than previously known algorithms.

7.3.3.1 Exact solution

Algorithm 7.3 solves the Euclidean TSP in the plane. It uses two subroutines: Algorithm 7.2, which finds the separators, and Algorithm 7.4, which solves the problem of finding the shortest Hamiltonian paths with starting points and end points given. Given a set of points and m pairs of distinguished points, the Hamiltonian paths are m paths with the given pairs as starting and end points, which together pass through all points in the set and for which the sum of the lengths is minimal.

In order to prove the correctness of Algorithm 7.3 we have to prove five lemmas. We will start with these and conclude with the three algorithms.

Lemma 7.2 *Given two concentric circles with radii r_1 and r_2, $r_1 < r_2$. If the optimal* TSP *tour contains $k\sqrt{n}$ edges which cross both the circles, then $r_2 - r_1 < \frac{3\pi(r_1+r_2)}{k\sqrt{n}}$.*

PROOF The idea is to find a shorter tour if the distance $r_2 - r_1$ between the circles is too big. This is done by identifying three close neighbouring crossing

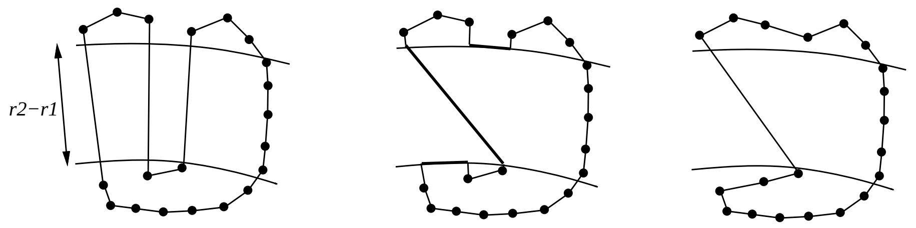

Figure 7.1: *Three edges crossing the circles and the result after shortening and taking shortcuts.*

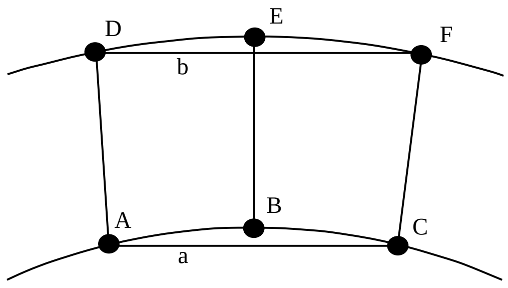

Figure 7.2: *Definition of A, B, C, D, E, F, a and b.*

edges, substituting the parts of the edges between the circles by shorter edges and taking shortcuts as in Figure 7.1. Observe that the optimal tour can never cross itself.

We define the points A, B, C, D, E, F as the intersection of the edges and the circles and the distances a and b as AC and DF respectively. See Figure 7.2.

First we look at the possible values of $a + b$ for three neighbouring crossing edges. Starting at any edge we compute the sum of the values of $a + b$ around the ring. This sum is less than the sum of the circumferences $2\pi(r_1 + r_2)$. The mean value of $a + b$ (supposing there is an even number of crossing edges) is $\frac{2\pi(r_1+r_2)}{k\sqrt{n}/2}$. Thus there are always three neighbouring crossing edges with $a + b < \frac{4\pi(r_1+r_2)}{k\sqrt{n}}$.

Suppose the distance between the circles is greater than $\frac{3\pi(r_1+r_2)}{k\sqrt{n}}$. We will show that we now can find a shorter tour by substituting these three crossing edges. The substitution will depend on how the edges are connected with each other in the tour. Figure 7.3 shows the three possible cases (reflections not counted) and how to make the substitutions in each case. Note that the substitutions in case 2 and 3 are the same. We have to show that the sum of the lengths of the new edges is less than before.
Case 1. Show that $AF + BC + DE < AD + BE + CF$.

$$AD + BE + CF \geq AD + 2(r_2 - r_1) \text{ and } AD + BE + CF \geq CF + 2(r_2 - r_1).$$

$$AF + BC + DE < AF + a + b < AD + b + a + b$$

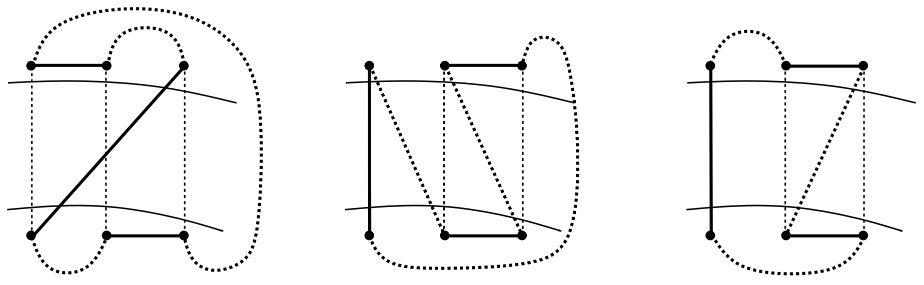

Figure 7.3: *How to make edge substitutions in the three different cases.*

which is less than $AD + 2(r_2 - r_1)$ if $a + 2b \leq 2(r_2 - r_1)$.

$$AF + BC + DE < AF + a + b < CF + a + a + b$$

which is less than $CF + 2(r_2 - r_1)$ if $2a + b \leq 2(r_2 - r_1)$.

$$\frac{3}{2}(a+b) \leq 2(r_2 - r_1) \Rightarrow (a + 2b \leq 2(r_2 - r_1)) \vee (2a + b \leq 2(r_2 - r_1)) \Rightarrow$$

$$\Rightarrow AF + BC + DE < AD + BE + CF.$$

Case 2 and 3. Show that $AD + BC + EF < AD + BE + CF$.

$$AD + BC + EF < AD + a + b < AD + 2(r_2 - r_1) \leq AD + BE + CF$$

if $\frac{3}{2}(a + b) \leq 2(r_2 - r_1)$.

Thus we can make the tour shorter if $r_2 - r_1 \geq \frac{3}{4}\frac{4\pi(r_1+r_2)}{k\sqrt{n}} = \frac{3\pi(r_1+r_2)}{k\sqrt{n}}$. □

Lemma 7.3 *Suppose we have four concentric circles with radii $r_1 < r_2 < r_3 < r_4$ dividing the plane into five areas P, Q, R, S and T as in Figure 7.4. Suppose that P and T contain at least $K\sqrt{n}$ cities each, Q and S exactly $p\sqrt{n}$ cities each and R contains $\sqrt{n}$ cities. If at least $K\sqrt{n}$ edges in the optimal tour cross the ring R and $0 < p < \frac{K}{2}$, then one of the following inequalities is true.*

$$\begin{aligned} r_4 - r_2 &< \frac{3\pi(r_4 + r_2)}{\left(\frac{K}{2} - p\right)\sqrt{n}} \\ r_3 - r_1 &< \frac{3\pi(r_3 + r_1)}{\left(\frac{K}{2} - p\right)\sqrt{n}} \end{aligned}$$

Proof The ring R is crossed by at least $K\sqrt{n}$ edges. At most $2p\sqrt{n}$ of them can go between cities in Q and S since there are exactly $2p\sqrt{n}$ of cities in Q and S. Therefore $(K - 2p)\sqrt{n}$ of the edges must start in either P or T. Thus either $\left(\frac{K}{2} - p\right)\sqrt{n}$ edges cross both circle 2 and 4 or $\left(\frac{K}{2} - p\right)\sqrt{n}$ edges cross both circle 1 and 3. Applying Lemma 7.2 gives us the sought inequalities. □

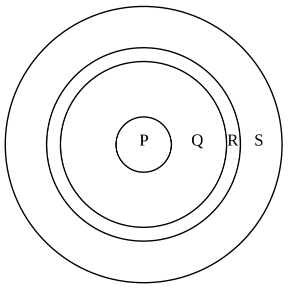

Figure 7.4: *Division of the plane into the areas P, Q, R, S, T.*

Lemma 7.4 *Suppose we have $0.8\sqrt{n}+1$ concentric circles with radii $r_0 < r_1 < \cdots < r_{0.8\sqrt{n}}$. Suppose that each pair of neighbouring circles (with radii r_i and r_{i+1}) encloses a ring with $\sqrt{n}$ cities and that at least $K\sqrt{n}$ edges in the optimal tour go across both the circles (i and $i+1$). If $K = 24$ then the quotient between the radius for the biggest and the smallest circle*

$$\frac{r_{0.8\sqrt{n}}}{r_0} < 1 + \sqrt{2}$$

PROOF Let $p = \lceil \frac{K}{2} \rceil - 1$. First look at the $2p+2$ innermost circles. By Lemma 7.3 we know that either $r_{p+1} - r_0 < \frac{3\pi(r_{p+1}+r_0)}{\left(\frac{K}{2}-p\right)\sqrt{n}}$ or $r_{2p+1} - r_p < \frac{3\pi(r_{2p+1}+r_p)}{\left(\frac{K}{2}-p\right)\sqrt{n}}$. In the first case we can use Lemma 7.3 again on rings $p+1$ to $2p+1$ and in the second case on rings 0 to p. This procedure can continue, each step giving a size bound for half of the rings. Finally we just have one or two rings left, which we bound using Lemma 7.2. If $K = 24$ we will get $p = 11$ and the bound

$$r_{2p+1} - r_0 < \frac{3\pi(r_{2p+1} + r_{2p+1})}{\sqrt{n}} \left(\frac{1}{\frac{24}{2} - 11} + \frac{1}{\frac{24}{2} - 5} + \frac{1}{\frac{24}{2} - 2} + 2 \cdot \frac{1}{24} \right) < \frac{25 r_{2p+1}}{\sqrt{n}}$$

In the same way we can bound $r_{2(2p+1)} - r_{2p+1}$ by $\frac{25 r_{2(2p+1)}}{\sqrt{n}}$, $r_{3(2p+1)} - r_{2(2p+1)}$ by $\frac{25 r_{3(2p+1)}}{\sqrt{n}}$ and so on. Define a_i as $r_{i(2p+1)}$ for $0 \le i \le \left\lfloor \frac{0.8\sqrt{n}}{2p+1} \right\rfloor$. We then have the recurrence relation

$$a_i - a_{i-1} < \frac{25 a_i}{\sqrt{n}} \Rightarrow a_i < \frac{1}{1 - \frac{25}{\sqrt{n}}} a_{i-1} \Rightarrow a_{\lfloor \frac{0.8\sqrt{n}}{2p+1} \rfloor} < \left(1 - \frac{25}{\sqrt{n}}\right)^{-\lfloor \frac{0.8\sqrt{n}}{2p+1} \rfloor} a_0$$

Thus

$$\begin{aligned} \frac{r_{0.8\sqrt{n}}}{r_0} &< \left(1 - \frac{25}{\sqrt{n}}\right)^{-\lfloor \frac{0.8\sqrt{n}}{2p+1} \rfloor - 1} = \exp\left(-\left\lfloor \frac{0.8\sqrt{n}}{2p+1} - 1 \right\rfloor \ln(1 - \frac{25}{\sqrt{n}})\right) \\ &< \exp\left(\frac{0.8\sqrt{n}}{2p+1} \frac{25}{\sqrt{n}}\right) = \exp \frac{0.8 \cdot 25}{23} < 1 + \sqrt{2} \end{aligned}$$

□

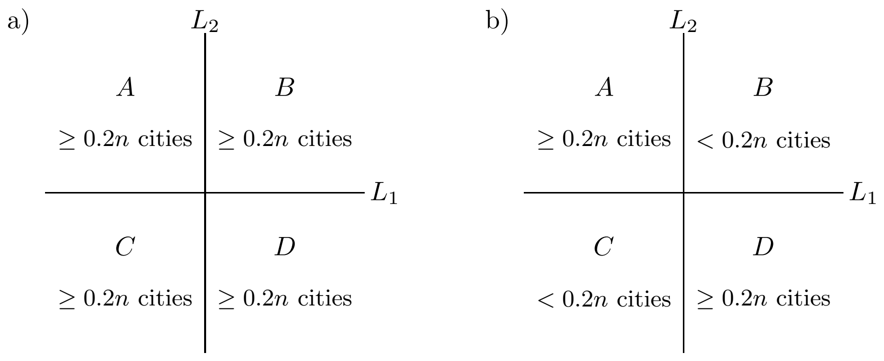

Figure 7.5: *Two partitions of n cities.*

Lemma 7.5 *Suppose we have a line L_1 in the plane and $n/2$ cities on each side of the line. Then there is a line L_2 orthogonal to L_1 dividing the plane into four areas A, B, C, D as in Figure 7.5, where A and D contain at least $n/5$ cities each and B and C either both contain at least $n/5$ cities (case a) or both contain less than $n/5$ cities (case b).*

Proof Place L_2 in such a way that C and D contain $n/4$ cities each. If A and B contain at least $n/5$ cities each, then we are done. Otherwise we suppose, without loss of generality, that it is B which contains less than $n/5$ cities. Now move the line L_2 slowly to the left, all the time counting how many cities are moved from A to B and from C to D. Stop when there are either $n/5$ cities in B (i.e. we are in case a in Figure 7.5) or less than $n/5$ cities in C (i.e. we are in case b), whichever occurs first. If the two cases occur at the same time, at least two cities must be on L_2, one on each side of the crossing point. Then we turn the two lines clockwise a little bit so the cities on L_2 are moved from B and C, leaving us in case b. □

Lemma 7.6 *In the complex plane the four circles $re^{i\theta}, r(1+\sqrt{2})e^{i\theta}, re^{i\theta}+c$ and $r(1+\sqrt{2})e^{i\theta}+c$ are given, where $c = \frac{r}{\sqrt{2}}(1+\sqrt{2}+i)$, r is a positive constant, and $0 \leq \theta < 2\pi$, see Figure 7.6. The ring segment D which is bounded by the circles $re^{i\theta}, r(1+\sqrt{2})e^{i\theta}$ and the lines te^{i0} and $te^{i\pi/4}$ for $t \geq r$ is contained in the circle $re^{i\theta}+c$. The ring segment B which is bounded by the circle $re^{i\theta}$ and the lines $te^{i\pi}$ and $te^{-i\pi/2}$ for $t \geq r$ has no point inside the circle $r(1+\sqrt{2})e^{i\theta}+c$. See Figure 7.7 for an illustration.*

Proof The intersection points of $re^{i\theta}$ and $re^{i\phi}+c$ are $re^{i\pi/4}$ (for $\phi = \pi$) and r (for $\phi = -\frac{3\pi}{4}$). The intersection points of $r(1+\sqrt{2})e^{i\theta}$ and $re^{i\phi}+c$ are $r(1+\sqrt{2})$ (for $\phi = -\frac{\pi}{4}$) and $r(1+\sqrt{2})e^{i\pi/4}$ (for $\phi = \frac{\pi}{2}$). Therefore D is contained in $re^{i\phi}+c$.

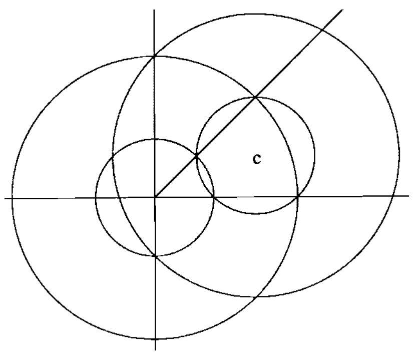

Figure 7.6: *The four circles of Lemma 7.6.*

The point in B which is closest to c is obviously $-ir$. But $-ir$ is on $r(1+\sqrt{2})e^{i\phi}+c$ for $\phi = -\frac{3\pi}{4}$. Thus no point in B can be inside the circle $r(1+\sqrt{2})e^{i\phi}+c$. □

Algorithm 7.2
INPUT: n cities in the plane.
OUTPUT: $0.8\sqrt{n}$ possible ring shaped separators, separating two areas in the plane. Every separator has $\sqrt{n}$ cities and the two areas it separates have at least $0.1n$ cities each. At least one of the separators is crossed by fewer than $24\sqrt{n}$ edges from the optimal TSP tour through the n cities.
ALGORITHM:

1. Find the two lines L_1 and L_2 as in Lemma 7.5. Let L_1 be the x-axis and L_2 the y-axis in a coordinate system for the plane.
2. Construct a circle with the center in the origin (the crossing point of L_1 and L_2). Make its radius r so big that the circle contains exactly $0.1n$ cities.
3. Construct $0.8\sqrt{n}$ concentric circles around the first circle so that every ring between two neighbouring circles contains exactly $\sqrt{n}$ cities.

   If, at any time, a city lies on a circle, we move the whole structure so that it does not lie on the circle and no other city is placed on a circle or on L_1 or L_2.
4. If the radius of the outermost circle is greater than $r(1+\sqrt{2})$ then return the $0.8\sqrt{n}$ rings between the circles.
5. Otherwise, count the cities in the four ring segments created by the innermost circle, the outermost circle and the lines L_1 and L_2. Cut the ring segment with the greatest number of cities in two symmetric halves by a line through the origin. Choose a new center c in the middle of the half with the greatest number of cities, at distance $r\sqrt{2+\sqrt{2}}$ from the origin.
6. Construct a new circle with the center in c and radius r' so that the circle contains exactly $0.1n$ cities.

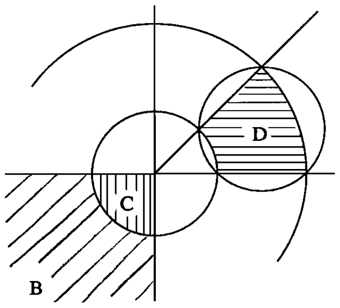

Figure 7.7: *Naming of the three areas B, C and D used in the correctness proof.*

7. Construct $0.8\sqrt{n}$ concentric circles around this circle in the same way as in step 3 of the algorithm.
8. Return the $0.8\sqrt{n}$ rings between the circles from step 6 and 7.

TIME: The heaviest part of the computation is to sort the n cities with respect to the y-coordinate (to find L_1), the x-coordinate (to find L_2), the distance to the origin (to find the circles in step 2 and 3) and the distance to c (to find the circles in step 6 and 7). These sortings use $O(n \log n)$ comparisons. The rest of the computation can also be done in $O(n \log n)$.

CORRECTNESS: The output from the algorithm is $0.8\sqrt{n}$ concentric rings with $\sqrt{n}$ cities in each ring. If the outermost radius is at least $1+\sqrt{2}$ times the innermost radius, then Lemma 7.4 says that there must be a ring with less than $24\sqrt{n}$ edges from the optimal tour. Thus the algorithm is correct if it returns in step 4.

The circle segment containing the most cities in step 5 must contain at least $\frac{1}{4}0.8n = 0.2n$ cities. The half containing the most cities must contain at least $0.1n$ cities. Without loss of generality we can assume that the new center is $c = r\sqrt{2+\sqrt{2}}\,e^{i\pi/8} = \frac{r}{\sqrt{2}}(1+\sqrt{2}+i)$. Lemma 7.6 says that the entire half is contained in a circle with radius r about c. Therefore the circle constructed in step 6 must have radius $r' \le r$.

By Lemma 7.6 we also know that the area B in Figure 7.7 is at least the distance $r(1+\sqrt{2})$ from c. If B contains more than $0.1n$ cities then the outermost circle constructed in step 7 must cross B and thus has radius $> r(1+\sqrt{2}) \ge r'(1+\sqrt{2})$.

The only thing left to show is that B contains more than $0.1n$ cities. The area C in Figure 7.7 can contain at most $0.1n$ cities because the circle $re^{i\theta}$ contains exactly $0.1n$ cities. We have already seen that the circle segment containing the most cities in step 5 contains at least $\frac{n}{5}$ cities. Therefore the opposite quadrant $B \cup C$ must contain at least $\frac{n}{5}$ cities, since L_1 and L_2 were chosen according to Lemma 7.5. Thus B contains at least $0.1n$ cities and the correctness is shown. □

Algorithm 7.3
INPUT: n cities in the plane.
OUTPUT: The minimal tour through the n cities.
ALGORITHM:

```
1. If $n < 100$, test all possible tours and return the shortest.
2. Otherwise, find $0.8\sqrt{n}$ possible separators by calling Algorithm 7.2.
3. For each separator $S$ separating $U$ from $V$, test all possible connections between the three areas with at most $24\sqrt{n}$ crossing edges. There are $\sqrt{n}$ cities in $S$ and thus there are at most $2\sqrt{n}$ edges between $S$ and $U \cup V$. For each such edge, test all combinations of the city in $S$, the city in $U \cup V$, the tour's next exit from $S$ and the tour's next exit from $U$ or $V$. For each possible crossing edge, test all combinations of the city in $U$, the city in $V$, the tour's next exit from $U$ and the tour's next exit from $V$.
   3.1 Check that the chosen combinations are consistent and create a single tour. Otherwise skip 3.2 to 3.5 and find the next combinations at once.
   3.2 Look at the cities in $S$ appearing in the combinations. They can be divided into pairs describing start points and end points for paths inside $S$. Call Algorithm 7.4 with the cities in $S$ and these pairs to find the shortest Hamiltonian paths between the cities in the pairs.
   3.3 In the same way as in step 3.2, compute the pairs for $U$ and call Algorithm 7.4 to compute the shortest paths in $U$.
   3.4 In the same way as in step 3.2, compute the pairs for $V$ and call Algorithm 7.4 to compute the shortest path in $V$.
   3.5 Combine the results of 3.2, 3.3 and 3.4, and compute the length of the resulting tour, which is optimal under the chosen combinations. Remember this tour if it is shorter than the former shortest tour found in step 3.
4. Return the shortest tour found in step 3.
```

Algorithm 7.4
INPUT: n cities in the plane and a collection of m pairs of cities.
OUTPUT: The shortest Hamiltonian paths with start points and end points given by the pairs.
ALGORITHM:

```
1. If $n < 100$, test all possible Hamiltonian paths and return the shortest solution.
2. Otherwise, find $0.8\sqrt{n}$ possible separators by calling Algorithm 7.2.
3. For each separator $S$ separating $U$ from $V$, test all possible connections between the three areas with at most $24\sqrt{n}$ crossing edges. There are at most $2\sqrt{n}$ edges between $S$ and $U \cup V$. For each such edge, test all combinations of the city in $S$, the city in $U \cup V$, the path's next exit from $S$ and the path's next exit from $U$ or $V$. For each possible crossing edge, test all combinations of the city in $U$, the city in $V$, the path's next exit from $U$ and the path's next exit from $V$. For each pair of cities in the
```

input, test the path's possible next exits from the two areas containing the cities.

3.1 Check that the chosen combinations are consistent and create m paths with the given endpoints. Otherwise skip 3.2 to 3.5 and find the next combinations at once.

3.2 Look at the cities in S appearing in the combinations and in the pairs. They can be divided into new pairs describing start points and end points for paths inside S. Call Algorithm 7.4 recursively with the cities in S and these new pairs to find the shortest Hamiltonian paths between the cities in the pairs.

3.3 In the same way as in step 3.2, compute the new pairs for U and call Algorithm 7.4 to compute the shortest paths in U.

3.4 In the same way as in step 3.2, compute the new pairs for V and call Algorithm 7.4 to compute the shortest path in V.

3.5 Combine the results of 3.2, 3.3 and 3.4, and compute the length of the resulting Hamiltonian paths, which is optimal under the chosen combinations. Remember this solution if it is shorter than the former shortest solution found in step 3.

4. Return the shortest solution found in step 3.

TIME: The algorithms are very similar — Algorithm 7.3 is mainly a special case of Algorithm 7.4 with $m = 0$. Therefore we only investigate the time of Algorithm 7.4.

Step 1 takes time $O(1)$ and step 2 takes $O(n \log n)$. Step 3 is a loop over all possible separators and connections between the three areas. For each of the at most $2\sqrt{n}$ edges from S to $U \cup V$ we specify

city in S	$\sqrt{n}$
city in $U \cup V$	n
next exit from S	$\sqrt{n}$
next exit from U or V	$24\sqrt{n} + 2\sqrt{n} + m$

For each of the at most $24\sqrt{n}$ edges from U to V we specify

city in U	n
city in V	n
next exit from U	$24\sqrt{n} + 2\sqrt{n} + m$
next exit from V	$24\sqrt{n} + 2\sqrt{n} + m$

For each of the m input pairs of cities we may specify

next exit from the area of the first city	$24\sqrt{n} + 2\sqrt{n}$
next exit from the area of the second city	$24\sqrt{n} + 2\sqrt{n}$

Thus the loop in step 3 is executed at most

$$0.8\sqrt{n}\big(1 + n^2(26\sqrt{n} + m)\big)^{2\sqrt{n}}\big(1 + n^2(26\sqrt{n} + m)^2\big)^{24\sqrt{n}}\big(1 + 26^2 n\big)^m$$

that is $2^{O((\sqrt{n}+m)\log(n+m))}$ times. The recursive calls of Algorithm 7.4 in steps 3.2 to 3.4 have at most $0.9n$ cities, because of the properties of the separation. At most $26\sqrt{n}$ pairs can be added in a recursive call. Thus m, the number of pairs, is bounded by

$$26\big(\sqrt{n} + \sqrt{0.9n} + \sqrt{0.9^2 n} + \ldots\big) = 26\sqrt{n}\frac{1}{1 - \sqrt{0.9}} < 508\sqrt{n}$$

The consistency check in step 3.1 and the computation of pairs in steps 3.2 to 3.4 cost $O(n)$. Thus the time $T(n)$ for the entire algorithm is

$$T(n) = 2^{O(\sqrt{n}\log n)}\big(O(n) + 2T(0.9n) + T(\sqrt{n})\big) = 2^{O(\sqrt{n}\log n)}$$

CORRECTNESS: We know that Algorithm 7.3 will return $0.8\sqrt{n}$ separators of which one is crossed by fewer than $24\sqrt{n}$ edges from the optimal TSP tour. Step 3 of the algorithm will test all separators, among them the sought separator. For every separator, all connections between the three areas are tested and the shortest solution is found. The shortest solution obtained from a separator which is crossed by more than $24\sqrt{n}$ edges from the optimal tour, can naturally not be optimal. Thus the shortest solution of all separators must be an optimal solution. Therefore the optimal tour will be tested and returned by the algorithm. $\square$

Chapter 8

How problems are related approximability-wise

8.1 Master and slave problems

Simon introduced the name "master and slave problems" for a pair of NPO problems where an algorithm may use the second problem, the slave problem, as a subroutine to solve the first problem, the master problem [101]. An example is the following algorithm for the minimum graph colouring problem which as a subroutine uses the maximum independent set problem.

In each step the algorithm finds an approximate solution of the maximum independent set problem, colours the found independent set by a new colour and removes the independent set from the graph. This procedure continues until the graph is empty. The performance ratio for the colouring algorithm is at most $\log n$ times the performance ratio for the used Max Ind Set approximation algorithm where n is the number of nodes in the graph, thus

$$\mathcal{R}[\text{Min Graph Colouring}] \leq \log n \cdot \mathcal{R}[\text{Max Ind Set}].$$

The same master-slave relation can be obtained by other problems, for example Min Clique Partition and Max Clique, see further [101].

8.2 Problems in the plane

Many of the NP-complete optimization problems on graphs which are mentioned in Appendix B are still NP-complete when the input graph is restricted to be planar. The approximability, however, changes a lot for most of the problems. This is because planar graphs always can be divided into large disconnected components by removing just a small number of nodes or edges. The separation can often be used to solve problems with divide-and-conquer techniques. This leads to polynomial time asymptotic approximation schemes for the problems. Several problems can even be solved by polynomial time approximation schemes. As we will see there are still some planar problems which are relatively hard to approximate and do not seem to be in PTAS.

Some NP-complete problems are polynomially solvable when restricted to planar graphs. PO for example contains the maximum planar clique problem [70] as well as the maximum planar cut problem [100].

In 1979 Lipton and Tarjan stated the planar separator theorem which asserts that any planar graph of n nodes can be divided into components of roughly equal size by removing $O(\sqrt{n})$ nodes [72, 73]. The planar separator theorem was later improved somewhat by Djidjev and is formulated as follows.

Proposition 8.1 (Djidjev [25])
Let G be any n-node planar graph. The nodes of G can be partitioned into three sets A, B, C such that no edge joins a node in A with a node in B, $|A| \leq 2n/3$, $|B| \leq 2n/3$, and $|C| \leq \sqrt{6n}$.

Other separation results have been obtained by Miller [79] and Rao [93]. The planar separator theorem can be applied to a large number of planar problems, for example maximum induced subgraph problems. This is problems where one wants to find the largest induced subgraph which satisfies some property Q. If we for example choose Q as node independence we will get the maximum independent set problem. The following proposition specifies sufficient properties for problems to obtain good approximability behaviour using a divide-and-conquer algorithm based on the planar separator theorem.

Proposition 8.2 (Chiba, Nishizeki and Saito [82])
If P is a maximum induced subgraph problem on the planar graph G with respect to property Q, which is

i) hereditary*, that is every subgraph of G satisfies Q whenever G satisfies Q,*

ii) determined by the components*, that is G satisfies Q whenever every connected component of G satisfies Q,*

iii) recognizable in linear time*,*

then P can be approximated within

$$1 - O\left(\frac{1}{\sqrt{\log \log n}}\right)$$

in time $O(n \log n)$ where n is the number of nodes in G. Thus $P \in \text{PTAS}^{\infty}$.

Example of graph properties satisfying this are independence, bipartiteness, forest and outerplanarity. Other examples of NP-complete planar problems which can be approximated in the same way are minimum planar node cover [82], maximum planar three dimensional matching (see [28]) and the planar version of MAX 3SAT, where literals and clauses are placed in the plane (see [70]).

Using that every planar graph is m-outerplanar for some m Baker has constructed an algorithm which approximates maximum planar independent set within $1+1/k$ in time $O(8^k kn)$ for any positive integer k [6]. Thus her algorithm realizes both an absolute performance ratio and an asymptotic performance ratio. If we choose $k = 1/\varepsilon$ the algorithm places maximum planar independent set in Ptas and if we choose $k = \log\log n$ the algorithm solves maximum planar independent set within $1 + 1/\log\log n$ in time $O(n(\log n)^3 \log\log n)$.

The same method can be used to place the planar versions of the problems Min Node Cover, Min Dominating Set, Min Edge Dominating Set, Max Triangle Packing, and Max H-matching in Ptas [6].

Separation techniques do not seem to be useful for all planar problems. There are problems where it is not enough to look at separate components, but where you have to look at the whole input structure, for example the Euclidean travelling salesperson problem and the minimum Euclidean Steiner tree problem. These problems are in Apx but probably not in Ptas. Therefore it may be possible to show them to be $\overline{\text{Max SNP}}$-hard. As yet only the travelling salesperson problem with distances one and two and minimum Steiner tree with distances one and two have been shown to be $\overline{\text{Max SNP}}$-hard [89, 14].

An even harder problem to approximate is the minimum edge colouring problem, which cannot be approximated within $4/3 - \varepsilon$ for any $\varepsilon > 0$ provided that P $\neq$ NP [82]. The reason for this is that it is NP-hard to decide whether a given cubic graph can be edge coloured with three colours, in spite of the fact that chromatic index of a cubic graph is either three or four [43].

8.3 Summary of how all problems are related

In this section all problems mentioned in the thesis and defined in Appendix B are listed, ordered by approximability. Every box in the list below corresponds to an approximability class. Each problem is only included in the box which best corresponds to its approximability. For example the Max 3Sat problem is just included in the box with Max SNP-complete problems and not the boxes with $\overline{\text{Max SNP}}$-complete problems and Apx problems because the Max SNP-complete problems form a smaller class. For the definitions of the problems we refer to Appendix B.

Figure 8.3 gives a survey of how the approximability classes are included in each other. See Figure 4.1 for a survey of the relations between classes defined by logical formulas.

8.3.1 Problems with constant absolute error guarantee

Constant absolute error guarantee
an additive constant from the optimum

Min Edge Colouring
Min Deg Spanning Tree
Min Planar Record Packing

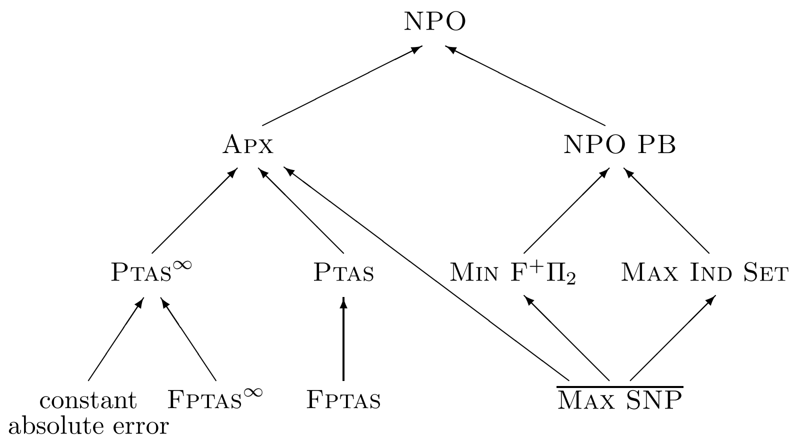

Figure 8.1: *Inclusions of approximability classes.*

8.3.2 Problems with approximation schemes

Fptas
within every ε in time polynomial in $1/\varepsilon$

Min m-processor Scheduling
Min m-processor Scheduling Speed Factors
Min Uniform m-processor Scheduling
Max k-Multicom Flow
Max Knapsack
Max Integer k-choice Knapsack

Fptas$^\infty$
asymptotically within every ε
in time polynomial in $1/\varepsilon$

Min Bin Packing

Ptas
within every ε

Min Planar Node Cover
Min Planar Dominating Set
Min Planar Edge Dominating Set
Max Planar H-matching
Max Planar Independent Set
Max Integer m-dimensional Knapsack
Max Weighted Sat with Small Bound

PTAS^∞
asymptotically within every ε
Min Planar Ind Dom Set
Max Planar k-colourable IS
Max Planar 3DM
Max Planar 3Sat
Max Planar kSat

8.3.3 Problems which can be approximated within a constant

Max SNP-**complete problems**
within a constant but not within every ε
Max 2Sat
Max 2Sat $-B$
Max 3Sat
Max 3Sat $-B$
Max kSat
Max kSat $-B$
Max Triangle Packing $-B$
Max 3DM $-B$
Max kDM $-B$
Max 3SP $-B$
Max kSP $-B$
Max Ind Set $-B$
Max k-constraint Sat
Max Not-all-equal 3Sat
Max Cut
Max Cut $-B$
Max k-cut
Max Directed Cut
Max k-colourable ES

$\overline{\text{Max SNP}}$-**complete problems**
P-equivalent to Max 3Sat
within a constant but not within every ε
Min Node Cover $-B$
Min Dominating Set $-B$
Min kSC $-B$
Max H-matching $-B$
Max Induced H-matching $-B$
Min Ind Dom Set $-B$

Max NP $\cap$ $\overline{\text{Max SNP}}$**-hard**
P-reduction from Max 3Sat
within a constant but not within every ε

Max Sat
Max Weighted Sat
Max G kSat

$\overline{\text{Max SNP}}$**-hard** $\cap$ Apx
P-reduction from Max 3Sat
within a constant but not within every ε

Min Node Cover
Min k-hypernode Cover
Max 3DM
Max kDM
Max 3SP
Max kSP
Max Triangle Packing
Max H-matching
Max Induced H-matching
Max CIS $-B$
Min $(1,2)$Tsp
Min $(1,2)$Tsp $-B$
Min $(1,2)$ Steiner Tree
Shortest Common Supersequence
Min kSC

Apx
within a constant, not known to be $\overline{\text{Max SNP}}$*-hard*

Min Cut
Min Weighted k-center
Min L-balanced k-center
Min L-balanced weighted k-center
Min L-balanced ρ-dominating Set
Min $(k, \mathcal{F})$-partition $-d$
Max CES $-B$
Min ΔTsp
Min $k\Delta$Tsp
Min ETsp
Min Steiner Tree
Min Euclidean Steiner Tree
Min Rectilinear Steiner Tree
Min Scp
Min 3D Packing
Max Weighted Sat with Bound

APX − PTAS
within a constant but not within every ε
MIN EDGE COLOURING
MIN k-CLUSTERING
MIN k-CENTER
MIN k-SUPPLIER
MIN k-SWITCHING NET
MIN BOTTLENECK Δ WSP

8.3.4 Problems not known to be approximable within a constant

MIN $F^+\Pi_2$**-complete**
within $O(\log n)$ but not within every ε
MIN DOMINATING SET
MIN SC
MIN HITTING SET

MIN GRAPH COLOURING **class**
P- or r-equivalent to MIN GRAPH COLOURING
not within every ε
MIN GRAPH COLOURING
MIN CLIQUE PARTITION
MIN CLIQUE COVER
MIN CBSC
(MIN CONSISTENT DFA)

NPO − APX
not within a constant
MIN IND DOM SET
MIN k-CLUSTERING SUM
MAX L-BALANCED k-CLUSTERING SUM
MIN NODE DISJOINT CYCLE COVER
MIN DIRECTED NODE DISJOINT CYCLE COVER
MIN EDGE DISJOINT CYCLE COVER
MIN DIRECTED EDGE DISJOINT CYCLE COVER
LONGEST PATH
MIN $0-1$ ASSIGNMENT
MIN QUADRATIC $0-1$ ASSIGNMENT
MIN 3DNF SAT

Max Ind Set class
P-equivalent to Max Ind Set
not within n^ε

Max Ind Set
Max Clique
Max SP
Max 2-Hyperclique
Max 2 Ones Neg
Max CIS
Max k-colourable IS
Max Complete Bipartite Subgraph
(Longest Common Subsequence)

NPO PB-complete
P-reduction from any NPO *problem with polynomially bounded optimum, not within* n^ε

Longest Computation
LP with Forbidden Pairs
Max PB $0-1$ Programming
Max Induced Connected Chordal Subgraph
LIP
Max CICS
Max # Sat
Max Dones
Max Ones

NPO-complete
A- and P-reductions from any NPO *problem*

Min Tsp
Min $0-1$ Programming
Min Weighted Sat
Min Weighted 3Sat

Other problems
not known to be approximable within a constant

Min Feedback Node Set
Min Feedback Edge Set
Min Balanced Cut
Min Ratio-cut
Min ρ-dominating Set
Max CES
Min Interval Graph Completion
Min EC
Min Test Collection
Min S/T 1-processor Scheduling
Min $\equiv$Deletion

Appendix A

Proofs of several reductions

This appendix contains proofs of some reductions which are mentioned in Chapter 8 and Appendix B. Some of the reductions have been claimed in works by other authors, but the proofs have not been published previously.

Theorem A.1 Min SC $\leq_L^p$ Min Dominating Set *and* Min Dominating Set $\leq_L^p$ Min SC.

Proof Let f_1 be the following reduction from Min SC to Min Dominating Set. An input $C = \{S_1, \ldots, S_n\}$ with $X = \bigcup_{S \in C} S = \{x_1, \ldots, x_m\}$ is reduced to an undirected graph $G = \langle V, E\rangle$ where $V = \{1, 2, \ldots, n, x_1, x_2, \ldots, x_m\}$ and $E = \{(i,j) : 1 \leq i < j \leq n\} \cup \{(i,x) : 1 \leq i \leq n \wedge x \in S_i\}$.

Given a dominating set $V' \subseteq V$ of G we can transform it back to a set cover in the following way. Note that the nodes $1, \ldots, n$ are all connected to each other by edges but there are no edges at all between the nodes $x_1, \ldots, x_m$. For every node $x_j \in V'$ we can substitute it in V' by any one of the nodes $\{i : (i, x_j) \in E\}$ and we still have a dominating set of at most the same size. Continue this until no node $x_1, \ldots, x_m$ is in V'. Now we can construct a set covering of X consisting of $\{S_i \in C : i \in V'\}$. It is clear that this is a set cover of size $|V'|$.

It is easy to see that from every set cover $C' \subseteq C$ we can construct a dominating set $V' \subseteq V$ of G of exactly the same size by including the nodes $\{i : S_i \in C'\}$. Thus $opt(f_1(C)) \leq opt(C)$ and we have shown that f_1 is an L-reduction with $\alpha = \beta = 1$.

Now we describe a reduction f_2 in the other direction, from Min Dominating Set to Min SC. An input graph $G = \langle V, E\rangle$ with $V = \{1, \ldots, n\}$ is reduced to a family C of subsets of the set $X = \{1, \ldots, n\}$ in the following manner. Let $C = \{S_1, \ldots, S_n\}$ where, for $i \in [1..n]$, $S_i = \{i\} \cup \{j \in [1..n] : (i,j) \in E\}$.

An element $i \in X$ can be covered either by including S_i, corresponding to including the node i in the dominating set, or by including one of the sets S_j such that $(i,j) \in E$, corresponding to including node j in the dominating set. Thus the minimum dominating set $V' \subseteq V$ gives us the minimum set cover $C' \subseteq C$ (which has the same size) and every set cover of C gives us a dominating set of G of the same size. Once again we have obtained an L-reduction with $\alpha = \beta = 1$.

If the degree of the input graph is bounded by B we observe that f_2 constructs sets of size at most $B + 1$ and each element in X occurs in at most $B+1$ sets. Thus f_2 is an L-reduction from MIN DOMINATING SET $-B$ to MIN $(B+1)$SC $-(B+1)$.

□

The two reductions f_1 and f_2 come from Paz and Moran [90], but they only showed that f_1 and f_2 are non-constructive measure preserving reductions, see Definition 3.13.

Theorem A.2 MIN IND DOM SET $-B$ *is* $\overline{\text{MAX SNP}}$*-complete.*

The minimum bounded independent dominating set problem takes a graph with node degrees bounded by a constant B. The problem is to find a subset of the nodes which is both independent and dominating and as small as possible.
PROOF We show that MIN IND DOM SET $-B \in \overline{\text{MAX SNP}}$ by giving a formulation of the complement problem, that is finding the maximum subset of nodes whose complement set is an independent dominating set.

$$\begin{aligned} opt(\langle V,E\rangle) \;=\; & \max_S |\{(u,v_1,\ldots,v_B,w_{11},\ldots,w_{1B},\;\ldots\;,w_{B1},\ldots,w_{BB}) : \\ & u \notin S \wedge (N(1)\wedge\cdots\wedge N(B)) \wedge (v_1 \in S \vee \cdots \vee v_B \in S) \wedge \\ & \wedge(M(1)\wedge\cdots\wedge M(B))\}| \end{aligned}$$

where $N(k) = v_k \in S \Rightarrow (w_{k1} \notin S \wedge \cdots \wedge w_{kB} \notin S)$ and $M(k) = (u,v_k) \in E \wedge (v_k,w_{k1}) \in E \wedge \cdots \wedge (v_k,w_{kB}) \in E \wedge v_k \neq v_{k+1} \wedge \cdots \wedge v_k \neq v_B \wedge w_{k1} \neq w_{k2} \wedge \cdots \wedge w_{k(B-1)} \neq w_{kB}$ insures that there are edges between v_k and u and w_{ki}.

This formulation is only valid if every node has degree exactly B, but it can be extended to work for lesser degrees too.

A solution of the complement problem of size s gives a solution of the original problem of size $|V| - s$. The size $m(\langle V,E\rangle, V') = |V'|$ of any solution V' of the original problem can easily be seen to be bounded in the following way.

$$\frac{|V|}{B+1} \le |V'| \le |V|$$

We can assume that there are no isolated nodes in the graph. This is no restriction since isolated nodes are always in a dominating set.

From this we immediately see that any polynomial time algorithm which finds an independent dominating set will approximate MIN IND DOM SET $-B$ within $B + 1$.

Now we have an L-reduction from MIN IND DOM SET $-B$ to the complement problem which is in MAX Σ_0. Thus MIN IND DOM SET $-B \in \overline{\text{MAX }\Sigma_0} = \overline{\text{MAX SNP}}$.

We will prove that MIN IND DOM SET $-B$ is $\overline{\text{MAX SNP}}$-hard by L-reducing from MAX 3SAT $-B$.

Given an instance $\langle U, C\rangle$ of Max 3Sat $-B$, we construct an undirected graph $f(\langle U, C\rangle) = \langle V, E\rangle$ as follows. For each variable $x_i \in U$ there is a three-clique consisting of three nodes labelled x_i, $\overline{x}_i$ and a_i. Moreover, for each clause $c_j \in C$ there is one node b_j and edges between b_j and x_i whenever the literal x_i is in clause c_j and between b_j and $\overline{x}_i$ whenever $\overline{x}_i$ is in clause c_j.

First, observe that for each i, exactly one of a_i, x_i and $\overline{x}_i$ must be in the independent dominating set in order to dominate a_i. Given an independent dominating set V', for each i such that $a_i \in V'$ we remove a_i from V', insert x_i and remove any b_j such that $(b_j, x_i) \in E$. The resulting set is still an independent dominating set, not larger that the original independent dominating set, and for every i it contains either x_i or $\overline{x}_i$.

Now we can assign values to the variables (x_i is set to true iff the node $x_i \in V'$). This assignment will satisfy as many clauses as there are nodes b_j which are *not* in the independent dominating set, that is $|U| + |C| - |V'|$.

Thus, the optimal solution of an instance $\langle U, C\rangle$ of the Max 3Sat $-B$ problem corresponds to the optimal solution of the Min Ind Dom Set $-B$ problem $f(\langle U, C\rangle)$ in the following way (since $opt(\langle U, C\rangle) \geq |C|/2$ and we can assume that $|U| \leq |C|$).

$$\begin{aligned} opt(f(\langle U, C\rangle)) &= |U| + |C| - opt(\langle U, C\rangle) \leq 2|C| - opt(\langle U, C\rangle) \leq \\ &\leq 4opt(\langle U, C\rangle) - opt(\langle U, C\rangle) = 3opt(\langle U, C\rangle) \end{aligned}$$

Therefore the reduction f is an L-reduction with $\alpha = 3$ and $\beta = 1$. □

Theorem A.3 Max k-constraint Sat *is* Max Σ_0*-complete for* $k \geq 2$*.*

Proof We will show that Max k-constraint Sat $\in$ Max Σ_0 in a similar way as we showed that Max 3Sat $\in$ Max Σ_0 in Example 4.3.

First suppose that each conjunction is unique. Encode the input instance as $k+1$ relations $C_0, \ldots, C_k$ where C_i contains all conjunctions with exactly i negative literals. Let $c = (x_1, \ldots, x_k) \in C_i$ mean that the i first variables $x_1, \ldots, x_i$ occur negated in conjunction c and the $k - i$ remaining variables occur positive in conjunction c. If a conjunction contains less than k literals we duplicate the last variable in the encoding to fill up the k places in the conjunction. Now Max k-constraint Sat can be defined as follows.

$$\begin{aligned} opt(\langle C_0, \ldots, C_k\rangle) = \max_T |\, (x_1, \ldots, x_k) :& \\ &((x_1, \ldots, x_k) \in C_0 \wedge (x_1 \in T \wedge x_2 \in T \wedge \cdots \wedge x_k \in T)) \vee \\ \vee\ &((x_1, \ldots, x_k) \in C_1 \wedge (x_1 \notin T \wedge x_2 \in T \wedge \cdots \wedge x_k \in T)) \vee \\ \vee\ &\cdots \vee \\ \vee\ &((x_1, \ldots, x_k) \in C_k \wedge (x_1 \notin T \wedge x_2 \notin T \wedge \cdots \wedge x_k \notin T)) \,| \end{aligned}$$

If the same conjunction appears several times in the input we have to distinguish the different occurrences by introducing an extra variable in the x-vector. Say that the same conjunction appears at most m times in the input. Then

x_0 may take the values $[1..m]$, and $(x_0, x_1, \ldots, x_k) \in C$ if x_0 is less than or equal to the number of occurrences of the conjunction. It is easy to extend the formula in this way for any bounded number of occurrences of the same conjunction.

It is easy to reduce Max 2Sat to Max 2-constraint Sat. In addition to the ordinary variables we introduce a new variable s_i for each clause c_i in the Max 2Sat instance. A clause $c_i = (x_1 \vee x_2)$ corresponds to two conjunctions $(x_1 \wedge s_i)$ and $(x_2 \wedge \overline{s}_i)$. At most one of these two conjunctions can be satisfied at the same time. We can see that any solution of the constructed Max 2-constraint Sat instance gives us an equally large solution of the Max 2Sat instance with the same variable assignment. Moreover, for every solution of the Max 2Sat instance we can construct an equally large solution of the Max 2-constraint Sat instance by assigning values to the s_i variables such that for each satisfied clause, one of the two corresponding conjunctions is satisfied. Thus, this reduction is an L-reduction with $\alpha = \beta = 1$. □

Theorem A.4 Min Test Collection $\leq_L^p$ Min SC *with* $\alpha = \beta = 1$.

The input to the minimum test collection problem consists of a set C of subsets of some finite set X. The problem is to find a subset $C' \subseteq C$ of minimum cardinality such that for each pair of elements of X there is a $c \in C'$ such that exactly one of the elements in the pair is in c.

Proof Given an instance $\langle X, C\rangle$ of the minimum test collection problem we construct an instance $\{S_1, S_2, \ldots, S_{|C|}\}$ of the minimum set cover problem by

$$S_i = \{\{a, b\} : a \in X \wedge b \in X \wedge [(a \in c_i \wedge b \notin c_i) \vee (a \notin c_i \wedge b \in c_i)]\}$$

where $C = \{c_1, \ldots, c_{|C|}\}$.

A set cover now corresponds to a test collection of the same size. The reduction is an L-reduction with $\alpha = \beta = 1$. □

A property of Min Test Collection is that, in order to be able to separate all pairs, the size of the optimal solution must be at least logarithmic in the size of the element set X, that is, $opt(\langle X, C\rangle) \geq \log |X|$.

Appendix B

A list of optimization problems

This list contains every optimization problem which is mentioned in the thesis. It is structured in the same way as Garey's and Johnson's famous list of NP-complete problems [35].

The decision problem version of each optimization problem is NP-complete unless otherwise stated. We assume that P $\neq$ NP.

B.1 Graph theory

[1.1] **Minimum node cover** (MIN NODE COVER)

$\mathcal{I} = \{G = \langle V, E \rangle : G \text{ is a graph}\}$
$S(\langle V, E \rangle) = \{V' \subseteq V : \forall (v_1, v_2) \in E, v_1 \in V' \vee v_2 \in V'\}$
$m(\langle V, E \rangle, V') = |V'|$
$opt = \min$
APPROXIMABILITY:
MIN NODE COVER $\equiv_{sp}^{p}$ MAX CLIQUE $\equiv_{sp}^{p}$ MAX SP [4],
MIN NODE COVER $\in$ $\underline{\text{MIN } \Sigma_1}$ [61],
MIN NODE COVER is $\overline{\text{MAX SNP}}$-hard, see MIN NODE COVER $-B$.
The planar version of MIN NODE COVER $\in$ PTAS [6].
ALGORITHM: MIN NODE COVER can be approximated within
$2 - \dfrac{\log\log|V|}{2\log|V|}$ [8].

[1.2] **Minimum bounded node cover** (MIN NODE COVER $-B$)

This is the same problem as MIN NODE COVER but the degree of G is bounded by the constant B.
APPROXIMABILITY: MIN NODE COVER $-B$ is $\overline{\text{MAX SNP}}$-complete for $B \geq 7$ (MAX 3SAT $-B \leq_{L}^{p}$ MIN NODE COVER $-(B+1)$ with $\alpha = B+1$ and $\beta = 1$) [88].

ALGORITHM: MIN NODE COVER -3 can be approximated within $5/4$, MIN NODE COVER $-B$ can be approximated within $\dfrac{2B^2}{B^2+2B-1}$ for $B \geq 10$ [80]. There are results for $3 < B < 10$ too in [80].

[1.3] **Minimum k-hypernode cover** (MIN k-HYPERNODE COVER)
$\mathcal{I} = \{\langle A, E\rangle : A \text{ is a finite set, } E \subseteq A^k\}$
$S(\langle A, E\rangle) = \{A' \subseteq A : \forall (a_1, a_2, \ldots, a_k) \in E, a_1 \in A' \vee \cdots \vee a_k \in A'\}$
$m(\langle A, E\rangle, A') = |A'|$
$opt = \min$
APPROXIMABILITY: MIN NODE COVER $\leq^p_L$ MIN k-HYPERNODE COVER (if $k = 2$ this is the same problem as MIN NODE COVER),
MIN k-HYPERNODE COVER is MIN $F^+\Pi_1(k)$-complete [61].
ALGORITHM:
MIN k-HYPERNODE COVER can be approximated within k [61].

[1.4] **Minimum dominating set** (MIN DOMINATING SET)
$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a graph}\}$
$S(\langle V, E\rangle) = \{V' \subseteq V : \forall v_1 \in V - V' \; \exists v_2 \in V' : (v_1, v_2) \in E\}$
$m(\langle V, E\rangle, V') = |V'|$
$opt = \min$
APPROXIMABILITY: MIN DOMINATING SET $\equiv^p_L$ MIN SC with $\alpha = \beta = 1$, see Theorem A.1, MIN DOMINATING SET is MIN $F^+\Pi_2$-complete [60].
The planar version of MIN DOMINATING SET $\in$ PTAS [6].
ALGORITHM: MIN DOMINATING SET can be approximated within $O(\log |V|)$ by reduction to MIN SC, see MIN SC.

[1.5] **Minimum bounded dominating set** (MIN DOMINATING SET $-B$)
This is the same problem as MIN DOMINATING SET but the degree of G is bounded by the constant B.
APPROXIMABILITY: MIN DOMINATING SET $-B$ is $\overline{\text{MAX SNP}}$-complete for $B \geq 14$ (MIN NODE COVER $-B \leq^p_L$ MIN DOMINATING SET $-2B$) [88].
ALGORITHM: MIN DOMINATING SET $-B$ can be approximated within $\sum_{i=1}^{B+1} \frac{1}{i}$ by reduction to MIN SC, see MIN SC.

[1.6] **Minimum edge dominating set** (MIN EDGE DOMINATING SET)
$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a graph}\}$
$S(\langle V, E\rangle) = \{E' \subseteq E : \forall e_1 \in E - E' \; \exists e_2 \in E' : e_1 \cap e_2 \neq \emptyset\}$
$m(\langle V, E\rangle, E') = |E'|$
$opt = \min$
APPROXIMABILITY:
The planar version of MIN EDGE DOMINATING SET $\in$ PTAS [6].

[1.7] **Minimum independent dominating set** (MIN IND DOM SET)

$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a graph}\}$
$S(\langle V, E\rangle) = \{V' \subseteq V : (\forall v_1 \in V - V' \; \exists v_2 \in V' : (v_1, v_2) \in E) \wedge$
$\wedge (\forall v_1 \in V' \; \nexists v_2 \in V' : (v_1, v_2) \in E)\}$
$m(\langle V, E\rangle, V') = |V'|$
$opt = \min$
APPROXIMABILITY: MIN IND DOM SET $\notin$ APX [47].
The planar version of MIN IND DOM SET $\in$ PTAS$^\infty$, see Section 8.2.

[1.8] **Minimum bounded independent dominating set** (MIN IND DOM SET $-B$)

This is the same problem as MIN IND DOM SET but the degree of G is bounded by the constant B.
APPROXIMABILITY: MIN IND DOM SET $-B$ is $\overline{\text{MAX SNP}}$-complete, see Theorem A.2.

[1.9] **Minimum graph colouring** (MIN GRAPH COLOURING)

$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a graph}\}$
$S(\langle V, E\rangle) = \{F = \{C_1, C_2, \ldots, C_m\}$ a partition of V such that
$\forall i \in [1..m], C_i$ is an independent set$\}$
$m(\langle V, E\rangle, F) = |F|$
$opt = \min$
APPROXIMABILITY: There is a $c \in \mathbb{R}^+$ such that MIN GRAPH COLOURING cannot be approximated within $|V|^c$ [76], MIN CLIQUE PARTITION $\equiv^p$ MIN GRAPH COLOURING [90].
MIN Σ_1 $\not\ni$ MIN GRAPH COLOURING $\in$ MIN Π_1 [61]. Each algorithm which approximates MIN GRAPH COLOURING within $O\left(2^{(\log|V|)^{1-\delta}}\right)$ for arbitrarily small $\delta > 0$ can be transformed to an algorithm which approximates the size of the minimum colouring within $O\left(2^{(\log|V|)^{\varepsilon}}\right)$ for arbitrarily small $\varepsilon > 0$ [71].
If MAX IND SET can be approximated within $f(|V|)$ then MIN GRAPH COLOURING can be approximated within $\log|V| \cdot f(|V|)$, see Section 8.1 and [48, 101].
If A is an approximation algorithm for $\Pi =$MAX IND SET then there is an approximation algorithm A' for $\Pi' =$MIN GRAPH COLOURING such that

$$R_\Pi(G, A(G)) \cdot R_{\Pi'}(G, A'(G)) \leq \frac{5\,|V|}{(\log|V|)^2} \frac{opt_\Pi(G)}{opt_{\Pi'}} \leq \frac{5\,|V|}{(\log|V|)^2}$$

and vice versa [18].
ALGORITHM: MIN GRAPH COLOURING can be approximated within $O\left(|V| \dfrac{(\log\log|V|)^2}{(\log|V|)^3}\right)$ [38].
On a 3-colourable graph can MIN GRAPH COLOURING be approximated within $O(|V|^{0.4})$ [15].

[1.10] **Maximum k-colourable edge subgraph** (MAX k-COLOURABLE ES)

$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a graph}\}$
$S(\langle V, E\rangle) = \{E' \subseteq E : \exists f : V \to [1..k] : (v_1, v_2) \in E \Rightarrow f(v_1) \neq f(v_2)\}$
$m(\langle V, E\rangle, E') = |E'|$
$opt = \max$
For $k = 2$ Max k-colourable ES$\equiv^p$Max Cut.
Approximability: Max k-colourable ES is Max Σ_0-complete for $k \geq 2$ (Max Cut $\leq_L^p$ Max k-colourable ES) [88].
Algorithm:
Max k-colourable ES can be approximated within $\frac{k}{k-1}$ [106].

[1.11] **Maximum k-colourable induced subgraph** (Max k-colourable IS)

$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a graph}\}$
$S(\langle V, E\rangle) = \{V' \subseteq V : \exists f : V' \to [1..k] :$
$(v_1, v_2) \in E \wedge v_1 \in V' \wedge v_2 \in V' \Rightarrow f(v_1) \neq f(v_2)\}$
$m(\langle V, E\rangle, V') = |V'|$
$opt = \max$
For $k = 1$ Max k-colourable IS$\equiv^p$Max Ind Set.
Approximability:
Max k-colourable IS is Max $\mathrm{F}^-\Pi_1(2)$-complete for $k \geq 1$ [84].
The planar version of Max k-colourable IS $\in$ Ptas$^\infty$, see Section 8.2.
Algorithm: Reduce to Max Ind Set.

[1.12] **Minimum edge colouring** (Min Edge Colouring)

$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a graph}\}$
$S(\langle V, E\rangle) = \{F = \{C_1, C_2, \ldots, C_m\}$ a partition of E such that $\forall i \in [1..m], (e_1, e_2 \in C_i \Rightarrow e_1$ and e_2 are not incident to the same node)$\}$
$m(\langle V, E\rangle, F) = |F|$
$opt = \min$
Approximability: Min Edge Colouring cannot be approximated within $4/3 - \varepsilon$ for any $\varepsilon > 0$, Min Edge Colouring $\in$ Apx [82].
Algorithm: Min Edge Colouring can be approximated within 4/3. Min Edge Colouring can be approximated with an absolute error guarantee of 1 [82].

[1.13] **Minimum feedback node set** (Min Feedback Node Set)

$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a directed graph}\}$
$S(\langle V, E\rangle) = \{V' \subseteq V : \text{every directed cycle in } G \text{ has a node in } V'\}$
$m(\langle V, E\rangle, V') = |V'|$
$opt = \min$
Approximability: Min Feedback Node Set is $\overline{\text{Max SNP}}$-hard (Min Node Cover $\leq_L^p$ Min Feedback Node Set with $\alpha = \beta = 1$), Min Feedback Edge Set $\leq_L^p$ Min Feedback Node Set with $\alpha = \beta = 1$. Both the reductions are structure preserving [4].

[1.14] **Minimum feedback edge set** (Min Feedback Edge Set)
$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a directed graph}\}$
$S(\langle V, E\rangle) = \{E' \subseteq E : \text{every directed cycle in } G \text{ has an edge in } E'\}$
$m(\langle V, E\rangle, E') = |E'|$
$opt = \min$
Approximability: Min Feedback Edge Set is $\overline{\text{Max SNP}}$-hard (Min Node Cover $\leq_L^p$ Min Feedback Edge Set with $\alpha = \beta = 1$), Min Feedback Edge Set $\leq_L^p$ Min Feedback Node Set with $\alpha = \beta = 1$. Both the reductions are structure preserving [4].

[1.15] **Maximum triangle packing** (Max Triangle Packing)
$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a graph}\}$
$S(\langle V, E\rangle) = \{F = \{C_1, C_2, \ldots, C_m\}$ a collection of mutually disjoint subsets of V such that $\forall i \in [1..m], |C_i| = 3 \wedge G|_{C_i}$ is a triangle$\}$
$m(\langle V, E\rangle, F) = |F|$
$opt = \max$
Approximability: Max Triangle Packing $\in$ Apx, Max Triangle Packing is $\overline{\text{Max SNP}}$-hard (Max 3Sat $-B \leq_L^p$ Max Triangle Packing with $\alpha = 18B + 7$ and $\beta = 1$), see Section 5.6.
The planar version of Max Triangle Packing $\in$ Ptas [6].
Algorithm: Max Triangle Packing can be approximated within 3, see Section 5.6.

[1.16] **Maximum bounded triangle packing** (Max Triangle Packing $-B$)
This is the same problem as Max Triangle Packing but the degree of the graph is bounded by the constant B.
Approximability: Max Triangle Packing $-B$ is Max SNP-complete for $B \geq 4$ (Max 3Sat $-B \leq_L^p$ Max Triangle Packing -4 with $\alpha = 18B + 7$ and $\beta = 1$), see Section 5.6.
Algorithm: See Max Triangle Packing.

[1.17] **Maximum H-matching** (Max H-matching)
H is a fixed graph with at least three nodes in some connected component.
$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a graph}\}$
$S(\langle V, E\rangle) = \{D = \{C_1, C_2, \ldots, C_m\}$ a collection of mutually disjoint subsets of E such that $\forall i \in [1..m], G|_{C_i}$ and H are isomorphic$\}$
$m(\langle V, E\rangle, D) = |D|$
$opt = \max$
Approximability: Max H-matching $\in$ Apx, Max H-matching is $\overline{\text{Max SNP}}$-hard (Max 3DM $-B \leq_L^p$ Max H-matching) [53], see Section 5.7.

[1.18] **Maximum bounded H-matching** (Max H-matching $-B$)
This is the same problem as Max H-matching but the degree of the

graph G is bounded by the constant B.
APPROXIMABILITY: MAX H-MATCHING $-B \in$ APX, MAX H-MATCHING $-B$ is $\overline{\text{MAX SNP}}$-complete for any connected H with at least three nodes (MAX 3DM $-B \leq_L^p$ MAX H-MATCHING $-B$) [53]. MAX H-MATCHING $-B$ is $\overline{\text{MAX SNP}}$-hard for any H with at least three nodes in some connected component. See Section 5.7.

[1.19] **Maximum planar H-matching** (MAX PLANAR H-MATCHING)

This is the same problem as MAX H-MATCHING but the graph G must be planar.
APPROXIMABILITY: MAX PLANAR H-MATCHING $\notin$ FPTAS [10], MAX PLANAR H-MATCHING $\in$ PTAS [6], MAX PLANAR H-MATCHING cannot be approximated within $1+O(1/k^\alpha)$ where $k = opt(\langle V, E\rangle)$ for any $\alpha > 0$ [10].
ALGORITHM: MAX PLANAR H-MATCHING can be approximated within $1 + O\left(\frac{1}{\sqrt{\log opt(\langle V, E\rangle)}}\right)$ [10].

[1.20] **Maximum induced H-matching** (MAX INDUCED H-MATCHING)

H is a fixed graph with at least three nodes in some connected component.
$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a graph}\}$
$S(\langle V, E\rangle) = \{F = \{C_1, C_2, \ldots, C_m\}$ a collection of mutually disjoint subsets of V such that $\forall i \in [1..m], G|_{C_i}$ and H are isomorphic$\}$
$m(\langle V, E\rangle, F) = |F|$
$opt = \max$
APPROXIMABILITY: MAX INDUCED H-MATCHING $\in$ APX, MAX INDUCED H-MATCHING is $\overline{\text{MAX SNP}}$-hard (MAX 3DM $-B \leq_L^p$ MAX INDUCED H-MATCHING) [53], see Section 5.7.

[1.21] **Maximum bounded induced H-matching** (MAX INDUCED H-MATCHING $-B$)

This is the same problem as MAX INDUCED H-MATCHING but the degree of the graph G is bounded by the constant B.
APPROXIMABILITY: MAX H-MATCHING $-B \in$ APX, MAX INDUCED H-MATCHING $-B$ is $\overline{\text{MAX SNP}}$-complete for any connected H with at least three nodes (MAX 3DM $-B \leq_L^p$ MAX INDUCED H-MATCHING $-B$) [53]. MAX INDUCED H-MATCHING $-B$ is $\overline{\text{MAX SNP}}$-hard for any H with at least three nodes in some connected component. See Section 5.7.

[1.22] **Maximum cut** (MAX CUT)

$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a graph}\}$
$S(\langle V, E\rangle) = \{V' \subseteq V\}$
$m(\langle V, E\rangle, V') = |\{(v_1, v_2) \in E : v_1 \in V' \wedge v_2 \in V - V'\}|$
$opt = \max$

Approximability: Max Cut is Max Σ_0-complete (Max Not-all-equal 3Sat $\leq_L^p$ Max Cut) [88].
The planar version of Max Cut $\in$ PO, see Section 8.2.
Algorithm: Max Cut can be approximated within 2, see for example [88].

[1.23] **Maximum bounded cut** (Max Cut $-B$)

This is the same problem as Max Cut but the degree of G is bounded by the constant B.
Approximability: Max Cut $-B$ is Max Σ_0-complete [88].
Algorithm: See Max Cut.

[1.24] **Maximum directed cut** (Max Directed Cut)

$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a directed graph}\}$
$S(\langle V, E\rangle) = \{V' \subseteq V\}$
$m(\langle V, E\rangle, V') = |E'|$ where $E' = \{(v_1, v_2) \in E : v_1 \in V' \wedge v_2 \in V - V'\}$
$\cup \{(v_2, v_1) \in E : v_1 \in V' \wedge v_2 \in V - V'\}$
$opt = \max$
Approximability: Max Directed Cut is Max Σ_0-complete [88].
Algorithm: Max Directed Cut can be approximated within 4 [106].

[1.25] **Maximum k-cut** (Max k-cut)

$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a complete graph}, w : E \to \mathbb{N} \text{ edge weights bounded by a polynomial in } |V|\}$
$S(\langle V, E, w\rangle) = \{F = \{C_1, C_2, \ldots, C_k\} \text{ a partition of } V\}$

$$m(\langle V, E, w\rangle, F) = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \sum_{\substack{v_1 \in C_i \\ v_2 \in C_j}} w(\{v_1, v_2\})$$

$opt = \max$
Approximability: Max k-cut is Max SNP-complete.
Algorithm: Max k-cut can be approximated within k [96].

[1.26] **Minimum cut** (Min Cut)

$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a complete graph}, k \in [2..|V|], w : E \to \mathbb{N}\}$
$S(\langle V, E, k, w\rangle) = \{F = \{C_1, C_2, \ldots, C_k\} \text{ a partition of } V\}$

$$m(\langle V, E, k, w\rangle, F) = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \sum_{\substack{v_1 \in C_i \\ v_2 \in C_j}} w(\{v_1, v_2\})$$

$opt = \min$
Algorithm: Min Cut can be approximated within $2 - \frac{2}{k}$ [97].
For fixed k Min Cut can be solved exactly in time $O\left(|V|^{k^2}\right)$ [97].

[1.27] **Minimum balanced cut** (Min Balanced Cut)

$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a complete graph}, k \in [2..|V|], w : E \to \mathbb{N}\}$

$S(\langle V,E,k,w\rangle) = \{F = \{C_1, C_2, \dots, C_k\}$ a partition of V :
$\forall i \in [1..k]\ |C_i| = \frac{|V|}{k}\}$

$$m(\langle V,E,k,w\rangle, F) = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \sum_{\substack{v_1 \in C_i \\ v_2 \in C_j}} w(\{v_1, v_2\})$$

opt = min
ALGORITHM: MIN BALANCED CUT can be approximated within $\frac{k-1}{k} \cdot |V|$ [97].
For fixed k MIN BALANCED CUT can be solved exactly in time $O\left(|V|^{k^2}\right)$ [36].

[1.28] **Minimum ratio-cut** (MIN RATIO-CUT)

$\mathcal{I} = \{G = \langle V,E\rangle : G$ is a graph$\}$
$S(\langle V,E\rangle) = \{V' \subseteq V\}$

$$m(\langle V,E\rangle, V') = \frac{|\{(v_1,v_2) \in E : v_1 \in V' \wedge v_2 \in V - V'\}|}{|V'| \cdot |V - V'|}$$

opt = min
ALGORITHM:
MIN RATIO-CUT can be approximated within $O(\log |V|)$ [66, 67].

[1.29] **Minimum k-clustering** (MIN k-CLUSTERING)

$\mathcal{I} = \{G = \langle V,E\rangle$ a complete graph, $f : E \to \mathbb{N}$ edge weights satisfying the triangle inequality$\}$
$S(\langle V,E,f\rangle) = \{F = \{C_1, C_2, \dots, C_k\}$ a partition of $V\}$

$$m(\langle V,E,f\rangle, F) = \max_{\substack{i \in [1..k] \\ v_1,v_2 \in C_i}} f(\{v_1, v_2\})$$

opt = min
APPROXIMABILITY:
MIN k-CLUSTERING cannot be approximated within $2 - \varepsilon$ for $\varepsilon > 0$ [40].
ALGORITHM: MIN k-CLUSTERING can be approximated within 2 [40].

[1.30] **Minimum k-clustering sum** (MIN k-CLUSTERING SUM)

$\mathcal{I} = \{G = \langle V,E\rangle$ a complete graph, $f : E \to \mathbb{N}\}$
$S(\langle V,E,f\rangle) = \{F = \{C_1, C_2, \dots, C_k\}$ a partition of $V\}$

$$m(\langle V,E,f\rangle, F) = \sum_{i=1}^{k} \sum_{v_1,v_2 \in C_i} f(\{v_1, v_2\})$$

opt = min
APPROXIMABILITY: MIN k-CLUSTERING SUM $\notin$ APX[96].

[1.31] **Maximum L-balanced k-clustering sum** (MAX L-BALANCED k-CLUSTERING SUM)

$\mathcal{I} = \{G = \langle V,E\rangle$ a complete graph, $f : E \to \mathbb{N}, w : V \to \mathbb{N}\}$
$S(\langle V,E,f\rangle) = \{F = \{C_1, C_2, \dots, C_k\}$ a partition of V such that

$$\forall i \in [1..k], \sum_{i \in C_i} w(i) \le L\}$$

$m(\langle V,E,f\rangle, F) = \displaystyle\sum_{i=1}^{k} \sum_{v_1,v_2 \in C_i} f(\{v_1,v_2\})$
opt = max
APPROXIMABILITY: MAX L-BALANCED k-CLUSTERING SUM $\notin$ APX[96].

[1.32] **Minimum k-center** (MIN k-CENTER)
$\mathcal{I} = \{G = \langle V,E\rangle$ a complete graph, $f : E \to \mathbb{N}$ edge weights satisfying the triangle inequality$\}$
$S(\langle V,E,f\rangle) = \{C \subseteq V\}$
$m(\langle V,E,f\rangle, C) = \max\limits_{v \in V} \min\limits_{c \in C} f(\{v,c\})$
opt = min
APPROXIMABILITY:
MIN k-CENTER cannot be approximated within $2 - \varepsilon$ for $\varepsilon > 0$ [40].
ALGORITHM: MIN k-CENTER can be approximated within 2 [40].

[1.33] **Minimum weighted k-center** (MIN WEIGHTED k-CENTER)
$\mathcal{I} = \{G = \langle V,E\rangle$ a complete graph, $f : E \to \mathbb{N}$ edge weights satisfying the triangle inequality, $w : V \to \mathbb{N}\}$
$S(\langle V,E,f,w\rangle) = \{C \subseteq V\}$
$m(\langle V,E,f,w\rangle, C) = \max\limits_{v \in V} \min\limits_{c \in C} f(\{v,c\})$
opt = min
ALGORITHM:
MIN WEIGHTED k-CENTER can be approximated within 3 [40].

[1.34] **Minimum L-balanced k-center** (MIN L-BALANCED k-CENTER)
$\mathcal{I} = \{G = \langle V,E\rangle$ a complete graph, $f : E \to \mathbb{N}$ edge weights satisfying the triangle inequality$\}$
$S(\langle V,E,f\rangle) = \{F = \{C_1, C_2, \ldots, C_k\}$ a partition of $V, C = \{c_1, \ldots, c_k\}$: $\forall i \in [1..k], c_i \in C_i \wedge |C_i| \le L + 1\}$
$m(\langle V,E,f\rangle, \langle F,C\rangle) = \max\limits_{i \in [1..k]} \max\limits_{v \in C_i} f(\{v,c_i\})$
opt = min
ALGORITHM: MIN L-BALANCED k-CENTER can be approximated within 16 (10 without proof) [7].

[1.35] **Minimum L-balanced weighted k-center** (MIN L-BALANCED WEIGHTED k-CENTER)
$\mathcal{I} = \{G = \langle V,E\rangle$ a complete graph, $f : E \to \mathbb{N}$ edge weights satisfying the triangle inequality, $w : V \to \mathbb{N}\}$
$S(\langle V,E,f,w\rangle) = \{F = \{C_1, C_2, \ldots, C_m\}$ a partition of V,
$C = \{c_1, \ldots, c_m\} : \forall i \in [1..m], c_i \in C_i \wedge |C_i| \le L + 1 \wedge \sum\limits_{v \in C} w(v) \le k\}$
$m(\langle V,E,f,w\rangle, \langle F,C\rangle) = \max\limits_{i \in [1..k]} \max\limits_{v \in C_i} f(\{v,c_i\})$

$opt = \min$
Algorithm: Min L-balanced weighted k-center can be approximated within 21 [7].

[1.36] **Minimum k-supplier** (Min k-supplier)

$\mathcal{I} = \{G = \langle V_C \cup V_S, E\rangle$ a complete graph with $V_C \cap V_S = \emptyset, f : E \to \mathbb{N}$ edge weights satisfying the triangle inequality$\}$
$S(\langle V_C \cup V_S, E, f\rangle) = \{C \subseteq V_S\}$
$m(\langle V_C \cup V_S, E, f\rangle, C) = \max_{v \in V_C} \min_{c \in C} f(\{v, c\})$
$opt = \min$
Approximability:
Min k-supplier cannot be approximated within $3 - \varepsilon$ for $\varepsilon > 0$ [40].
Algorithm: Min k-supplier can be approximated within 3 [40].

[1.37] **Minimum k-switching network** (Min k-switching Net)

$\mathcal{I} = \{G = \langle V, E\rangle$ a complete graph, $f : E \to \mathbb{N}$ edge weights satisfying the triangle inequality$\}$
$S(\langle V, E, f\rangle) = \{F = \{C_1, C_2, \ldots, C_{2k}\}$ a partition of $V\}$
$m(\langle V, E, f\rangle, F) = \max_{i \in [1..k]} \max_{\substack{v_1 \in C_{2i-1} \\ v_2 \in C_{2i}}} f(\{v_1, v_2\})$
$opt = \min$
Approximability: Min k-switching Net cannot be approximated within $2 - \varepsilon$ for $\varepsilon > 0$ [40].
Algorithm:
Min k-switching Net can be approximated within 3 [40].

[1.38] **Minimum ρ-dominating set** (Min ρ-dominating Set)

$\mathcal{I} = \{G = \langle V, E\rangle$ a complete graph, $f : E \to \mathbb{N}$ edge weights satisfying the triangle inequality$\}$
$S(\langle V, E, f\rangle) = \{V' \subseteq V : \max_{v \in V - V'} \min_{c \in V'} f(\{v, c\}) \le \rho\}$
$m(\langle V, E, f\rangle, V') = |V'|$
$opt = \min$
Algorithm: Min ρ-dominating Set can be approximated within $\log |V| + 1$ [74].

[1.39] **Minimum L-balanced ρ-dominating set** (Min L-balanced ρ-dominating Set)

$\mathcal{I} = \{G = \langle V, E\rangle$ a complete graph, $f : E \to \mathbb{N}$ edge weights satisfying the triangle inequality$\}$
$S(\langle V, E, f\rangle) = \{F = \{C_1, C_2, \ldots, C_m\}$ a partition of $V, C = c_1, \ldots, c_m :$ $\forall i \in [1..m] c_i \in C_i \;\wedge\; |C_i| \le L + 1\}$
$m(\langle V, E, f\rangle, \langle F, C\rangle) = |C|$
$opt = \min$
Algorithm: Min L-balanced ρ-dominating Set can be approximated within $\log L + 1$ [7].

[1.40] **Minimum $(k,\mathcal{F})$-partition with diameter d** (Min $(k,\mathcal{F})$-partition $-d$)

$\mathcal{I} = \{G = \langle V, E\rangle$ a graph, $f : E \to \mathbb{N}$ edge weights satisfying the triangle inequality, $\mathcal{F}$ family of graphs such that $\forall H \in \mathcal{F}, \mathit{diam}(H) \le d$ and $\forall i \in [1..|V|], \mathcal{F}(i)$ constructs a member of $\mathcal{F}$ with i nodes in polynomial time$\}$
$S(\langle V, E, f, \mathcal{F}\rangle) = \{F = \{C_1, C_2, \ldots, C_k\}$ a partition of $V, g : F \to \mathcal{F}, h : E \to \mathbb{N}$ such that $\forall c \in F \exists$ edge subgraph of c isomorphic with $g(c)$ under the matching given by the function h, $h(e) = 0$ iff e is not in any subgraph$\}$
$m(\langle V, E, f, \mathcal{F}\rangle, \langle F, g, h\rangle) = \max\limits_{\substack{e\in E:\\ h(e)\neq 0}} f(e)$
$\mathit{opt} = \min$
Algorithm:
Min $(k,\mathcal{F})$-partition $-d$ can be approximated within $2d$ [40].

[1.41] **Minimum clique partition** (Min Clique Partition)

$\mathcal{I} = \{G = \langle V, E\rangle : G$ is a graph$\}$
$S(\langle V, E\rangle) = \{F = \{C_1, C_2, \ldots, C_m\}$ a partition of V such that $\forall i \in [1..m], C_i$ induces a clique$\}$
$m(\langle V, E\rangle, F) = |F|$
$\mathit{opt} = \min$
Approximability: Min Clique Partition $\equiv^p$ Min Graph Colouring [90].
Algorithm:
If Max Clique can be approximated within $f(|V|)$ then Min Clique Partition can be approximated within $O(\log|V|)\cdot f(|V|)$, see Section 8.1 and [101].

[1.42] **Minimum clique cover** (Min Clique Cover)

$\mathcal{I} = \{G = \langle V, E\rangle : G$ is a graph$\}$
$S(\langle V, E\rangle) = \{F = \{A_1, A_2, \ldots, A_m\}$ subsets of E such that $\forall i \in [1..m], A_i$ is a clique and $\forall e \in E \exists i \in [1..m] : e \in A_i\}$
$m(\langle V, E\rangle, F) = |F|$
$\mathit{opt} = \min$
Approximability: Min Clique Partition $\le_r^{1+\varepsilon}$ Min Clique Cover with a size amplification of $|V|^2/(\log|V|)^2$ [62, 101]. Min Clique Cover $\le_r^1$ Min Clique Partition with a size amplification of $|E|$ [62, 101].
Algorithm: If Max Clique can be approximated within $f(|V|)$ then Min Clique Cover can be approximated within $O(\log|V|)\cdot f(|V|)$ [101].

[1.43] **Minimum complete bipartite subgraph cover** (Min CBSC)

$\mathcal{I} = \{G = \langle V, E\rangle : G$ is a graph$\}$
$S(\langle V, E\rangle) = \{F = \{A_1, A_2, \ldots, A_m\}$ subsets of E such that $\forall i \in [1..m]$,

A_i is a complete bipartite graph and $\forall e \in E \exists i \in [1..m] : e \in A_i\}$
$m(\langle V, E\rangle, F) = |F|$
$opt = \min$
APPROXIMABILITY: MIN CLIQUE PARTITION $\leq_r^{1+\varepsilon}$ MIN CBSC with a size amplification of $|E|^2/\varepsilon^2$ [101]. MIN CBSC $\leq_r^1$ MIN CLIQUE PARTITION with a size amplification of $|E|$ [101].
ALGORITHM:
If MAX CLIQUE can be approximated within $f(|V|)$ then MIN CBSC can be approximated within $O(\log|V|) \cdot f(|V|)$ [101].

[1.44] **Minimum node disjoint cycle cover** (MIN NODE DISJOINT CYCLE COVER)

$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a graph}\}$
$S(\langle V, E\rangle) = \{F \text{ family of node disjoint cycles covering } V\}$
$m(\langle V, E\rangle, F) = |F|$
$opt = \min$
APPROXIMABILITY: MIN NODE DISJOINT CYCLE COVER $\notin$ APX[96].

[1.45] **Minimum directed node disjoint cycle cover** (MIN DIRECTED NODE DISJOINT CYCLE COVER)

This is the same problem as MIN NODE DISJOINT CYCLE COVER but on a directed graph G.
APPROXIMABILITY:
MIN DIRECTED NODE DISJOINT CYCLE COVER $\notin$ APX[96].

[1.46] **Minimum edge disjoint cycle cover** (MIN EDGE DISJOINT CYCLE COVER)

$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a graph}\}$
$S(\langle V, E\rangle) = \{F \text{ family of edge disjoint cycles covering } V\}$
$m(\langle V, E\rangle, F) = |F|$
$opt = \min$
APPROXIMABILITY:
MIN EDGE DISJOINT CYCLE COVER $\notin$ APX[96].

[1.47] **Minimum directed edge disjoint cycle cover** (MIN DIRECTED EDGE DISJOINT CYCLE COVER)

This is the same problem as MIN EDGE DISJOINT CYCLE COVER but on a directed graph G.
APPROXIMABILITY:
MIN DIRECTED EDGE DISJOINT CYCLE COVER $\notin$ APX[96].

[1.48] **Maximum clique** (MAX CLIQUE)

$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a graph}\}$
$S(\langle V, E\rangle) = \{V' \subseteq V : v_1, v_2 \in V' \wedge v_1 \neq v_2 \Rightarrow (v_1, v_2) \in E\}$
$m(\langle V, E\rangle, V') = |V'|$

$opt = \max$
APPROXIMABILITY: MAX CLIQUE $\equiv^p$ MAX IND SET, see section 4.11, MAX CLIQUE $\notin$ MAX Σ_1 [84], MAX CLIQUE is $\overline{\text{MAX NP}}$-hard [23], MAX CLIQUE is MAX $F^-\Pi_1(2)$-complete [84], MAX CLIQUE $\notin$ APX [3], there is a $c \in \mathbb{R}^+$ such that MAX CLIQUE cannot be approximated within $|V|^c$ [2].
The planar version of MAX CLIQUE $\in$ PO [70].
ALGORITHM: Reduce to MAX IND SET.

[1.49] **Maximum clique in a bounded graph** (MAX CLIQUE $-B$)

This is the same problem as MAX CLIQUE but the degree of G is bounded by the constant B. The corresponding decision problem is polynomially solvable.
APPROXIMABILITY: MAX CLIQUE $-B \in$ PO,
MAX CLIQUE $-B \notin$ MAX Σ_1 (see Theorem 4.19).

[1.50] **Maximum k-hyperclique** (MAX k-HYPERCLIQUE)

$\mathcal{I} = \{\langle A, E\rangle : A \text{ is a finite set}, E \subseteq 2^A : e \in E \Rightarrow 1 \le |e| \le k\}$
$S(\langle A, E\rangle) = \{A' \subseteq A : S \subseteq 2^{A'} \wedge 1 \le |e| \le k \Rightarrow S \in E\}$
$m(\langle A, E\rangle, A') = |A'|$
$opt = \max$
APPROXIMABILITY: MAX CLIQUE $\equiv^p_P$ MAX 2-HYPERCLIQUE,
MAX k-HYPERCLIQUE $\equiv^p_P$ MAX k ONES NEG,
MAX k-HYPERCLIQUE is MAX $F^-\Pi_1(k)$-complete [84].
ALGORITHM: Reduce to MAX IND SET.

[1.51] **Maximum complete bipartite subgraph** (MAX COMPLETE BIPARTITE SUBGRAPH)

$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a graph}\}$
$$S(\langle V, E\rangle) = \{V_1, V_2 \subset V : \forall v_1 \in V_1 \cup V_2 \forall v_2 \in V_1 \cup V_2, (v_1, v_2) \in E \Rightarrow (v_1 \in V_1 \wedge v_2 \in V_2) \vee (v_1 \in V_2 \wedge v_2 \in V_1)\}$$
$m(\langle V, E\rangle, \langle V_1, V_2\rangle) = |V_1 \cup V_2|$
$opt = \min$
APPROXIMABILITY: MAX IND SET $\le^2_r$ MAX COMPLETE BIPARTITE SUBGRAPH without size amplification [101]. MAX COMPLETE BIPARTITE SUBGRAPH $\le^p_L$ MAX IND SET with $\alpha = \beta = 1$ and without size amplification [101].

[1.52] **Maximum independent set** (MAX IND SET)

$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a graph}\}$
$S(\langle V, E\rangle) = \{V' \subseteq V : v_1, v_2 \in V' \wedge v_1 \ne v_2 \Rightarrow (v_1, v_2) \notin E\}$
$m(\langle V, E\rangle, V') = |V'|$
$opt = \max$
APPROXIMABILITY: See MAX CLIQUE and Section 4.11.
The planar version of MAX IND SET $\in$ PTAS [6].

ALGORITHM: MAX IND SET can be approximated within $O\left(\frac{n}{(\log n)^2}\right)$ where $n = |V|$ [18].

[1.53] **Maximum bounded independent set** (MAX IND SET $-B$)

This is the same problem as MAX IND SET but the degree of G is bounded by the constant B.
APPROXIMABILITY: MAX IND SET $-B$ is MAX Σ_0-complete
(MAX 3SAT $-B \leq_L^p$ MAX IND SET$-(B+1)$ with $\alpha = \beta = 1$) [88].
ALGORITHM:
MAX IND SET $-B$ can trivially be approximated within $B+1$.

[1.54] **Longest induced path in a graph** (LIP)

$\mathcal{I} = \{G = \langle V, E\rangle : G$ is a graph$\}$
$S(\langle V, E\rangle) = \{V' \subseteq V : G|_{V'}$ is a simple path$\}$
$m(\langle V, E\rangle, V') = |V'|$
$opt = \max$
APPROXIMABILITY:
LIP is NPO PB-complete (LP WITH FORBIDDEN PAIRS $\leq_P^p$LIP) [12].

[1.55] **Longest path** (LONGEST PATH)

$\mathcal{I} = \{G = \langle V, E\rangle : G$ is a graph$\}$
$S(\langle V, E\rangle) = \{(v_1, \ldots, v_m)$ a sequence of m different nodes in V such that $\forall i \in [1..m-1](v_i, v_{i+1}) \in E\}$
$m(\langle V, E\rangle, (v_1, \ldots, v_m)) = m$
$opt = \max$
APPROXIMABILITY: LONGEST PATH $\notin$ APX [2].
ALGORITHM: For Hamiltonian graphs LONGEST PATH can be approximated within $O(|V| / \log |V|)$ [81].

[1.56] **Longest path with forbidden pairs** (LP WITH FORBIDDEN PAIRS)

$\mathcal{I} = \{G = \langle V, E\rangle : G$ is a graph$, P \subset V \times V\}$
$S(\langle V, E, P\rangle) = \{(v_1, \ldots, v_m)$ a sequence of m different nodes in V such that $\forall i \in [1..m-1]\,((v_i, v_{i+1}) \in E \wedge \forall j \in [i+1..m](v_i, v_j) \notin P)\}$
$m(\langle V, E, P\rangle, (v_1, \ldots, v_m)) = m$
$opt = \max$
APPROXIMABILITY: LP WITH FORBIDDEN PAIRS is
NPO PB-complete (LONGEST COMPUTATION $\leq_P^p$LP WITH FORBIDDEN PAIRS) [12].

[1.57] **Maximum induced connected chordal subgraph** (MAX INDUCED CONNECTED CHORDAL SUBGRAPH)

$\mathcal{I} = \{G = \langle V, E\rangle : G$ is a graph$\}$
$S(\langle V, E\rangle) = \{V' \subseteq V : G|_{V'}$ is connected and chordal$\}$

$m(\langle V, E\rangle, V') = |V'|$
$opt = \max$
APPROXIMABILITY: MAX INDUCED CONNECTED CHORDAL SUBGRAPH is NPO PB-complete (LP WITH FORBIDDEN PAIRS $\leq_P^p$MAX INDUCED CONNECTED CHORDAL SUBGRAPH) [12].

[1.58] **Maximum common induced subgraph** (MAX CIS)
$\mathcal{I} = \{G_1 = \langle V_1, E_1\rangle, G_2 = \langle V_2, E_2\rangle \text{ graphs}\}$
$S(\langle G_1, G_2\rangle) = \{V_1{}' \subseteq V_1, V_2{}' \subseteq V_2, f : V_1{}' \to V_2{}'$ bijective function such that $G_1\,|_{V_1{}'}$ and $G_2\,|_{V_2{}'}$ are f-isomorphic, that is
$$\forall v_1, v_2 \in V_1, (v_1, v_2) \in E_1 \Leftrightarrow (f(v_1), f(v_2)) \in E_2\}$$
$m(\langle G_1, G_2\rangle, \langle V_1{}', V_2{}'\rangle) = |V_1{}'| = |V_2{}'|$
$opt = \max$
APPROXIMABILITY: MAX CIS is as hard to approximate as MAX CLIQUE (MAX CLIQUE $\leq_L^p$ MAX CIS with $\alpha = \beta = 1$ and without size amplification, MAX CIS$\leq_L^p$MAX CLIQUE with $\alpha = \beta = 1$ and a size amplification of $|V_1|+|V_2|$ with respect to the nodes), see Section 6.2.2 and Section 4.11.
ALGORITHM: See Section 6.2.2.

[1.59] **Maximum bounded common induced subgraph** (MAX CIS $-B$)

This is the same problem as MAX CIS but the degree of the graphs G_1 and G_2 is bounded by the constant B.
APPROXIMABILITY: MAX CIS $-B \in$ APX, MAX CIS $-B$ is $\overline{\text{MAX SNP}}$-hard for $B \geq 25$ (MAX 3SAT $-6 \leq_L^p$ MAX CIS $-B$ with $\alpha = 43$ and $\beta = 1$), see Section 6.2.1.
ALGORITHM: MAX CIS $-B$ can be approximated within $B + 1$, see Section 6.2.1.

[1.60] **Maximum common edge subgraph** (MAX CES)
$\mathcal{I} = \{G_1 = \langle V_1, E_1\rangle, G_2 = \langle V_2, E_2\rangle \text{ graphs}\}$
$S(\langle G_1, G_2\rangle) = \{E_1{}' \subseteq E_1, E_2{}' \subseteq E_2, f : V_1{}' \to V_2{}'$ bijective function from the nodes in the subgraph $G_1\,|_{E_1{}'}$ to the nodes in $G_2\,|_{E_2{}'}$: $G_1\,|_{E_1{}'}$ and $G_2\,|_{E_2{}'}$ are f-isomorphic, that is
$\forall v_1, v_2 \in V_1{}', (v_1, v_2) \in E_1{}' \Leftrightarrow (f(v_1), f(v_2)) \in E_2{}'\}$
$m(\langle G_1, G_2\rangle, \langle E_1{}', E_2{}'\rangle) = |E_1{}'| = |E_2{}'|$
$opt = \max$
APPROXIMABILITY: MAX CES is not harder to approximate than MAX CLIQUE (MAX CES $\leq_L^p$ MAX CLIQUE with $\alpha = \beta = 1$), see Section 6.3.2.
ALGORITHM: See Section 6.3.2.

[1.61] **Maximum bounded common edge subgraph** (MAX CES $-B$)
This is the same problem as MAX CES but the degree of the graphs G_1 and G_2 is bounded by the constant B.
APPROXIMABILITY: MAX CES $-B \in$ APX,
MAX CES $-B$ is not harder to approximate than MAX CIS $-B$ (MAX

CES $-B \le_L^p$ Max CIS $-(2B+3)$ with $\alpha = 4B^2 + 10B + 7$ and $\beta = 1$), see Section 6.3.1.
Algorithm: Max CES $-B$ can be approximated within $B+1$, see Section 6.3.1.

[1.62] **Maximum common induced connected subgraph** (Max CICS)
This is the same problem as Max CIS but the only valid solutions are connected subgraphs.
Approximability: Max CICS is NPO PB-complete (LIP $\le_L^p$ Max CICS with $\alpha = \beta = 1$ [55]), see Theorem 6.9.

[1.63] **Maximum connected component** (Max Connected Component)

$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a graph}\}$
$S(\langle V, E\rangle) = \{V' \subseteq V : G|_{V'} \text{ is connected}\}$
$m(\langle V, E\rangle, V') = |V'|$
$opt = \max$
Approximability: Max Connected Component $\in$ PO,
Max $\Pi_1 \not\ni$ Max Connected Component $\in$ Max Π_2 [61].
Algorithm: Max Connected Component can be solved exactly in time $O(|E|)$, see for example [1].

[1.64] **Minimum interval graph completion** (Min Interval Graph Completion)
An interval graph is a graph whose nodes can be mapped to distinct intervals in the real line such that two nodes in the graph have an edge between them iff their corresponding intervals overlap.
$\mathcal{I} = \{G = \langle V, E\rangle : G \text{ is a graph}\}$
$S(\langle V, E\rangle) = \{G' = \langle V, E'\rangle : E \subseteq E', l : V \to \mathbb{Z}, r : V \to \mathbb{Z} \text{ such that}$
$\{v_1, v_2\} \in E' \Leftrightarrow l(v_1) \le l(v_2) \le r(v_1) \vee l(v_2) \le l(v_1) \le r(v_2)\}$
$m(\langle V, E\rangle, \langle E', l, r\rangle) = |E'|$
$opt = \min$
Algorithm: Min Interval Graph Completion can be approximated within $O\left(\log^2 |V|\right)$ [94].

B.2 Network design

[2.1] **Travelling salesperson problem** (Min Tsp)
$\mathcal{I} = \{\langle G, s\rangle : G = \langle V, E\rangle \text{ is a complete graph}, s : E \to \mathbb{N}\}$
$S(\langle G, s\rangle) = \{\text{permutation } \pi : [1..|V|] \to [1..|V|]\}$

$$m(\langle G, s\rangle, \pi) = s\left(\{v_{\pi(|V|)}, v_{\pi(1)}\}\right) + \sum_{i=1}^{|V|-1} s\left(\{v_{\pi(i)}, v_{\pi(i+1)}\}\right)$$

$opt = \min$
Approximability: Min Tsp $\notin$ Apx [35],
Min Tsp is NPO-complete [83].

[2.2] **TSP with triangle inequality** (Min ΔTsp)

This is the same problem as Min Tsp but the distance function s satisfies the triangle inequality, that is for all $i, j, k \in [1..|V|]$, $s(v_i, v_k) \le s(v_i, v_j) + s(v_j, v_k)$ if $i \ne j, i \ne k$ and $j \ne k$.
Approximability:
Min ΔTsp $\in$ Apx, Min ΔTsp is $\overline{\text{Max SNP}}$-hard, see Min $(1,2)$Tsp.
Algorithm:
Min ΔTsp can be approximated within $\frac{3}{2}$ in time $O(|V|^{5/2})$ [21].

[2.3] **Travelling k-salesperson problem with triangle inequality** (Min $k\Delta$Tsp)

This is the same problem as Min ΔTsp but we look for a collection of k subtours, each containing the start node, such that each node is in at least one subtour.
Approximability: $\mathcal{R}$[Min $k\Delta$Tsp] $\le \mathcal{R}$[Min ΔTsp]$+1-1/k$ [31].
Algorithm: Min $k\Delta$Tsp can be approximated within $\frac{5}{2} - \frac{1}{k}$ [31].

[2.4] **TSP with distances one and two** (Min $(1,2)$Tsp)

This is the same problem as Min Tsp but with the distance function $s : E \to \{1, 2\}$. Thus s satisfies the triangle inequality.
Approximability: Min $(1,2)$Tsp $\in$ Apx, Min $(1,2)$Tsp is $\overline{\text{Max SNP}}$-hard (Max 3Sat $-B \le_L^p$ Min $(1,2)$Tsp) [89].
Algorithm: Min $(1,2)$Tsp can be approximated within $\frac{7}{6}$ [89].

[2.5] **Bounded travelling salesperson problem with distances one and two** (Min $(1,2)$Tsp $-B$)

This is the same problem as Min $(1,2)$Tsp but the number of 1-edges from each node is bounded by the constant B.
Approximability: Min $(1,2)$Tsp $-B \in$ Apx, Min $(1,2)$Tsp $-B$ is $\overline{\text{Max SNP}}$-hard (Max 3Sat $-B \le_L^p$ Min $(1,2)$Tsp $-B$) [89].
Algorithm: See Min $(1,2)$Tsp.

[2.6] **Euclidean travelling salesperson problem** (Min ETsp)

$\mathcal{I} = \{C \subseteq \mathbb{Z}^k$ set of coordinates (integer k-vectors)$\}$
The length of an edge between two points (x_1, y_1) and (x_2, y_2) is the discretized Euclidean length $\left\lceil \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \right\rceil$.
$S(C) = \{$permutation $\pi : [1..|C|] \to [1..|C|]\}$
$$m(C, \pi) = \| c_{\pi(|C|)} - c_{\pi(1)} \| + \sum_{i=1}^{|C|-1} \| c_{\pi(i)} - c_{\pi(i+1)} \|$$
$opt = \min$
Approximability: Fptas $\not\ni$ Min ETsp $\in$ Apx [65].
Algorithm: See Min ΔTsp.

[2.7] **Stacker crane problem** (Min Scp)

$\mathcal{I} = \{G = \langle V, E\rangle$ is a complete graph, A a set of arcs where each arc is an ordered pair of cities, $s : A \cup E \to \mathbb{N}$ edge weights$\}$
$S(\langle G, A, s\rangle) = \{T = (i_1, \ldots, i_m)$ minimum length tour, possibly containing repeated nodes, such that $\forall (v_1, v_2) \in A \exists k : i_k = v_1 \wedge i_{k+1} = v_2\}$
$m(\langle G, A, s\rangle, T) =$ (length of the tour T)
$opt = \min$
ALGORITHM:
MIN BOTTLENECK Δ WSP can be approximated within 9/5 [31, 65].

[2.8] **Bottleneck wandering salesperson problem with triangle inequality** (MIN BOTTLENECK Δ WSP)

$\mathcal{I} = \{G = \langle V, E\rangle$ is a complete graph, $u, v \in V$, $s : E \to \mathbb{N}$ edge weights satisfying the triangle inequality$\}$
$S(\langle G, u, v, s\rangle) = \{$permutation $\pi : [1..|V|] \to [1..|V|]$ such that $v_{\pi(1)} = u$ and $v_{\pi(|V|)} = v\}$
$m(\langle G, u, v, s\rangle, \pi) = \max\limits_{i \in [1..|V|-1]} s\left(\{v_{\pi(i)}, v_{\pi(i+1)}\}\right)$
$opt = \min$
APPROXIMABILITY: MIN BOTTLENECK Δ WSP cannot be approximated within $2 - \varepsilon$ for $\varepsilon > 0$ [40].
ALGORITHM:
MIN BOTTLENECK Δ WSP can be approximated within 2 [40].

[2.9] **Minimum Steiner tree** (MIN STEINER TREE)

$\mathcal{I} = \{\langle G, s, S\rangle : G = \langle V, E\rangle$ is a complete graph, $s : E \to \mathbb{N}, S \subseteq V\}$
$S(\langle G, s, S\rangle) = \{Q \subseteq V - S$ set of Steiner nodes$\}$
$m(\langle G, s, S\rangle, Q) = \textit{mst-length}(C \cup Q)$ where *mst-length* is the length of the minimum spanning tree.
A minimum spanning tree for n points can be found in time $O(n^2)$, see for example [87].
$opt = \min$
APPROXIMABILITY: MIN STEINER TREE $\in$ APX.
ALGORITHM:
MIN STEINER TREE can be approximated within 16/9 [11].

[2.10] **Minimum Steiner tree with distances one and two** (MIN $(1,2)$ STEINER TREE)

This is the same problem as MIN STEINER TREE but with the distance function $s : E \to \{1, 2\}$. Thus s satisfies the triangle inequality.
APPROXIMABILITY: MIN $(1,2)$ STEINER TREE $\in$ APX,
MIN $(1,2)$ STEINER TREE is $\overline{\text{MAX SNP}}$-hard (MIN NODE COVER $-B$ $\leq_L^p$ MIN $(1,2)$ STEINER TREE with $\alpha = 2B, \beta = 1$) [14].
ALGORITHM:
MIN $(1,2)$ STEINER TREE can be approximated within 4/3 [14].

[2.11] **Minimum Euclidean Steiner tree** (MIN EUCLIDEAN STEINER TREE)

$\mathcal{I} = \{C \subseteq \mathbb{Z}^2$ set of integer coordinates in the plane$\}$
$S(C) = \{Q \subseteq \mathbb{Z}^2$ set of *Steiner points*, integer coordinates in the plane$\}$
$m(C, Q) = \textit{mst-length}(C \cup Q)$
where *mst-length* is the length of the minimum spanning tree. The length of an edge between two points (x_1, y_1) and (x_2, y_2) is the discretized Euclidean length $\left\lceil \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \right\rceil$.
$\textit{opt} = \min$
Approximability: Min Euclidean Steiner Tree $\in$ Apx.
Algorithm: Min Euclidean Steiner Tree can be approximated within $\frac{2}{\sqrt{3}} - \varepsilon$ for some $\varepsilon > 0$ [27]. The minimum spanning tree for C approximates Min Euclidean Steiner Tree within $\frac{2}{\sqrt{3}}$ [26].

[2.12] **Minimum rectilinear Steiner tree** (Min Rectilinear Steiner Tree)
$\mathcal{I} = \{C \subseteq \mathbb{Z}^2$ set of integer coordinates in the plane$\}$
$S(C) = \{Q \subseteq \mathbb{Z}^2$ set of *Steiner points*, integer coordinates in the plane$\}$
$m(C, Q) = \textit{mst-length}(C \cup Q)$ where *mst-length* is the length of the minimum spanning tree. The length of an edge between two points (x_1, y_1) and (x_2, y_2) in the rectilinear metric is $|x_1 - x_2| + |y_1 - y_2|$.
$\textit{opt} = \min$
Approximability: Min Rectilinear Steiner Tree $\in$ Apx.
Algorithm: Min Rectilinear Steiner Tree can be approximated within 97/72 [11].

[2.13] **Minimum degree spanning tree** (Min Deg Spanning Tree)
$\mathcal{I} = \{\langle G, s\rangle : G = \langle V, E\rangle$ is a complete graph, $s : E \to \mathbb{N}\}$
$S(\langle G, s\rangle) = \{E' \subseteq E : G|_{E'}$ is a spanning tree of $G\}$
$m(\langle G, s\rangle, E') = \deg(G|_{E'})$ the maximum degree of the induced graph.
$\textit{opt} = \min$
Algorithm: Min Deg Spanning Tree can be approximated with an absolute error guarantee of 1, that is one can in polynomial time find a spanning tree of G with maximum degree at most $\textit{opt}(\langle G, s\rangle) + 1$ [32].

B.3 Sets and partitions

[3.1] **Maximum three dimensional matching** (Max 3DM)
$\mathcal{I} = \{T \subseteq X \times Y \times Z : X \cap Y = Y \cap Z = Z \cap X = \emptyset\}$
$S(T) = \{M \subseteq T :$ no two triples in T agree in any coordinate$\}$
$m(T, M) = |M|$
$\textit{opt} = \max$
Approximability: Max 3DM $\in$ Apx, Max 3DM is $\overline{\text{Max SNP}}$-hard (Max 3Sat $-B \leq_L^p$ Max 3DM with $\alpha = 18B + 7$ and $\beta = 1$) [52], Max 3DM $\notin$ Max Σ_1 [84]. See Section 5.2.
The planar version of Max 3DM $\in$ Ptas$^\infty$, see Section 8.2.

Algorithm:
Max 3DM can be approximated within 3 [52], see Section 5.2.

[3.2] **Maximum bounded three dimensional matching** (Max 3DM $-B$)

This is the same problem as Max 3DM but the number of occurrences of any element in X, Y or Z is bounded by the constant B.
Approximability: Max 3DM $-B$ is Max SNP-complete for $B \geq 3$ (Max 3Sat $-B \leq_L^p$ Max 3DM -3 with $\alpha = 18B + 7$ and $\beta = 1$) [52, 54], see Section 5.2 and Theorem 4.26.
Algorithm: See Max 3DM.

[3.3] **Maximum k-dimensional matching** (Max kDM)

$\mathcal{I} = \{T \subseteq X_1 \times X_2 \times \cdots \times X_k : 1 \leq i < j \leq k \Rightarrow X_i \cap X_j = \emptyset\}$
$S(T) = \{M \subseteq T :$ no two k-tuples in T agree in any coordinate$\}$
$m(T, M) = |M|$
$opt = \max$
If $k = 2$ Max kDM is the same problem as maximum bipartite matching.
Approximability: Max kDM $\in$ Apx, Max kDM is $\overline{\text{Max SNP}}$-hard (Max 3DM $\leq_L^p$ Max kDM with $\alpha = \beta = 1$, see Section 5.5), Max kDM $\notin$ Max Σ_1 [84].
Algorithm: Max kDM can be approximated within k, see Section 5.5.

[3.4] **Maximum bounded k-dimensional matching** (Max kDM $-B$)

This is the same problem as Max kDM but the number of any element in the k-tuples is bounded by the constant B.
Approximability:
Max kDM $-B$ is Max SNP-complete for $B \geq 3$, see Section 5.5.
Algorithm: See Max kDM.

[3.5] **Maximum bipartite matching** (Max 2DM)

The corresponding decision problem is polynomially solvable.
$\mathcal{I} = \{T \subseteq X \times Y : X \cap Y = \emptyset\}$
$S(T) = \{M \subseteq T :$ no two tuples in T agree in any coordinate$\}$
$m(T, M) = |M|$
$opt = \max$
Approximability: Max 2DM $\in$ PO, Max 2DM $\notin$ Max Σ_1 [84].
Algorithm:
Max 2DM can be solved exactly in time $O\left(|T| \sqrt{|X| + |Y|}\right)$ [44].

[3.6] **Maximum non-bipartite matching** (Max Matching)

The corresponding decision problem is polynomially solvable.
$\mathcal{I} = \{G = \langle V, E\rangle : G$ is a graph$\}$
$S(\langle V, E\rangle) = \{E' \subseteq E :$ no two edges in E' are connected$\}$
$m(\langle V, E\rangle, E') = |E'|$

$opt = \max$
APPROXIMABILITY: MAX MATCHING $\in$ PO,
MAX MATCHING $\notin$ MAX Σ_1 [84].
ALGORITHM:
MAX MATCHING can be solved exactly in time $O\left(|E|\sqrt{|V|}\right)$ [78].

[3.7] **Maximum set packing** (MAX SP)

$\mathcal{I} = \{\langle X, C\rangle : X$ is a finite set, $C \subseteq 2^X$ is a collection of subsets of $X\}$
$S(\langle X, C\rangle) = \{C' \subseteq C : S_1, S_2 \in C' \wedge S_1 \neq S_2 \Rightarrow S_1 \cap S_2 = \emptyset\}$
$m(\langle X, C\rangle, C') = |C'|$
$opt = \max$
APPROXIMABILITY: MAX SP $\equiv^p_{sp}$ MAX CLIQUE [4],
MAX SP $\equiv^p_L$ MAX CLIQUE with $\alpha = \beta = 1$ (see Section 5.4),
MAX SP is MAX $\mathrm{F}^-\Pi_1(2)$-complete [84]. See MAX CLIQUE and Section 4.11.

[3.8] **Maximum three-set packing** (MAX 3SP)

This is the same problem as MAX SP but C consists only of sets of size exactly three.
APPROXIMABILITY: MAX 3SP $\in$ APX, MAX 3SP is $\overline{\text{MAX SNP}}$-hard (MAX 3DM $\leq^p_L$ MAX 3SP with $\alpha = \beta = 1$) [52], MAX 3SP $\notin$ MAX Σ_1 [84]. See Section 5.3.
ALGORITHM:
MAX 3SP can be approximated within 3 [52], see Section 5.3.

[3.9] **Maximum bounded three-set packing** (MAX 3SP $-B$)

This is the same problem as MAX 3SP but the number of occurrences in C of any element in X is bounded by the constant B.
APPROXIMABILITY: MAX 3SP $-B$ is MAX SNP-complete for $B \geq 3$ (MAX 3DM $-B \leq^p_L$ MAX 3SP $-B$ with $\alpha = \beta = 1$) [52, 54], see Section 5.3 and Theorem 4.26. MAX 3SP $-B \notin$ MAX Σ_1 (see Theorem 4.19).
ALGORITHM: See MAX 3SP.

[3.10] **Maximum k-set packing** (MAX kSP)

This is the same problem as MAX SP but C consists only of sets of size exactly k.
APPROXIMABILITY: MAX kSP $\in$ APX, MAX kSP is $\overline{\text{MAX SNP}}$-hard for all $k \geq 3$ (MAX 3SP $\leq^p_L$ MAX kSP with $\alpha = \beta = 1$), see Section 5.4.
MAX kSP $\notin$ MAX Σ_1 for all $k \geq 2$ [84].
If $k = 2$ MAX kSP is the same problem as MAX MATCHING.
ALGORITHM:
MAX kSP can be approximated within k [84], see Section 5.4.

[3.11] **Maximum bounded k-set packing** (MAX kSP $-B$)

This is the same problem as MAX kSP but the number of occurrences in C of any element in X is bounded by the constant B.
APPROXIMABILITY: MAX kSP $-B$ is MAX SNP-complete for $B \geq 3$ (MAX 3SP $-B \leq_L^p$ MAX kSP $-B$ with $\alpha = \beta = 1$) [52, 54], see Section 5.4.
ALGORITHM: See MAX kSP.

[3.12] **Minimum set cover** (MIN SC)
$\mathcal{I} = \{\langle X, C\rangle : X$ is a finite set, $C \subseteq 2^X$ is a collection of subsets of $X\}$
$S(\langle X, C\rangle) = \{C' \subseteq C : \bigcup\limits_{S \in C'} S = \bigcup\limits_{S \in C} S\}$
$m(\langle X, C\rangle, C') = |C'|$
$opt = \min$
APPROXIMABILITY:
MIN SC $\equiv_L^p$ MIN DOMINATING SET with $\alpha = \beta = 1$, see Theorem A.1, MIN SC $\equiv^p$ MIN HITTING SET [4], MIN SC is MIN $\mathrm{F}^+\Pi_2$-complete [60].
ALGORITHM: MIN SC can be approximated within $1 + \ln |X|$ [48].

[3.13] **Minimum k-set cover** (MIN kSC)
This is the same problem as MIN SC but the number of elements in every set in C is bounded by the constant k.
APPROXIMABILITY: MIN kSC is $\overline{\text{MAX SNP}}$-hard, see MIN kSC $-B$.
ALGORITHM: MIN kSC can be approximated within $\sum\limits_{i=1}^{k} \frac{1}{i}$ [48].

[3.14] **Minimum bounded k-set cover** (MIN kSC $-B$)
This is the same problem as MIN kSC but the number of occurrences of each element in C is bounded by B.
APPROXIMABILITY: MIN kSC $-B$ is $\overline{\text{MAX SNP}}$-complete for $k \geq 15$ and $B \geq 15$ (MIN DOMINATING SET $-B \leq_L^p$ MIN $(B+1)$SC $-(B+1)$ where $\alpha = \beta = 1$), see Theorem A.1 and [88].
ALGORITHM:
MIN kSC $-B$ can be approximated within $\min(B, \sum\limits_{i=1}^{k} \frac{1}{i})$ [39].

[3.15] **Minimum exact cover** (MIN EC)
$\mathcal{I} = \{\langle X, C\rangle : X$ is a finite set, $C \subseteq 2^X$ is a collection of subsets of $X\}$
$S(\langle X, C\rangle) = \{C' \subseteq C : \bigcup\limits_{S \in C'} S = \bigcup\limits_{S \in C} S\}$
$m(\langle X, C\rangle, C') = \sum\limits_{S \in C'} |S|$
$opt = \min$
Note that the only difference between MIN SC and MIN EC is the definition of the objective function.
APPROXIMABILITY: MIN SC $\leq_L^p$ MIN EC with $\alpha = (k+1)|X|, \beta = 1/(k|X|)$ [76].
ALGORITHM: MIN EC can be approximated within $1 + \ln |X|$ [48].

[3.16] **Minimum test collection** (Min Test Collection)

$\mathcal{I} = \{\langle X, C\rangle : X$ is a finite set, $C \subseteq 2^X$ is a collection of subsets of $X\}$
$S(\langle X, C\rangle) = \{C' \subseteq C : \forall x_1, x_2 \in X \exists S \in C' :$
$x_1 = x_2 \vee (x_1 \in S \wedge x_2 \notin S) \vee (x_1 \notin S \wedge x_2 \in S)\}$
$m(\langle X, C\rangle, C') = |C'|$
$opt = \min$
Approximability:
Min Test Collection $\leq_L^p$ Min SC with $\alpha = \beta = 1$, see Theorem A.4.

[3.17] **Minimum hitting set** (Min Hitting Set)

$\mathcal{I} = \{\langle X, C\rangle : X$ is a finite set, $C \subseteq 2^X$ is a collection of subsets of $X\}$
$S(\langle X, C\rangle) = \{X' \subseteq X : S \in C' \Rightarrow S \cap X \neq \emptyset\}$
$m(\langle X, C\rangle, X') = |X'|$
$opt = \min$
Approximability: Min Hitting Set $\equiv^p$ Min SC [4],
Min Hitting Set is Min $\mathrm{F}^+\Pi_2$-complete [60].
Algorithm: Min Hitting Set can be approximated within $1 + \ln|X|$ by reduction to Min SC, see Min SC.

B.4 Storage and retrieval

[4.1] **Bin packing** (Min Bin Packing)

$\mathcal{I} = \{\langle U, c, s\rangle : U$ is a finite set, $c \in \mathbb{Z}^+, s : U \to [0..c]\}$
$S(\langle U, c, s\rangle) = \{L = \{B_1, B_2, \ldots, B_m\}$ a partition of U such that
$\forall i \in [1..m], \sum_{x \in B_i} s(x) \leq c\}$
$m(\langle U, c, s\rangle, L) = |L|$
$opt = \min$
Approximability: Ptas $\not\ni$ Min Bin Packing $\in$ Apx, Min Bin Packing $\in$ Fptas^∞, see Section 4.4.
Algorithm: Min Bin Packing can be approximated within
$1 + \varepsilon$ in time polynomial in $1/\varepsilon$ where $\varepsilon = O\left(\frac{\log^2(opt(\langle U, c, s\rangle))}{opt(\langle U, c, s\rangle)}\right)$ [56].

[4.2] **Minimum height three dimensional packing** (Min 3D Packing)

$\mathcal{I} = \{w \in \mathbb{Z}^+$ width, $d \in \mathbb{Z}^+$ depth, $B \subset 2^{\mathbb{Z}^+ \times \mathbb{Z}^+ \times \mathbb{Z}^+}$ set of boxes (x_i, y_i, z_i) with width x_i, depth y_i and height $z_i\}$
$S(\langle w, d, B\rangle) = \{$packing P of the boxes B in a large box with width w, depth d and unbounded height; the boxes must be packed orthogonally and oriented$\}$
$m(\langle w, d, B\rangle, P) =$height of the packing P
$opt = \min$
Approximability: Min 3D Packing $\in$ Apx [69].
Algorithm: $\mathcal{R}^\infty$[Min 3D Packing]≤ 3.25 [69].

There are lots of variants of packing problems. A survey of approximation results of packing problems can be found in [22].

[4.3] **Shortest common supersequence** (SHORTEST COMMON SUPERSEQUENCE)

$\mathcal{I} = \{\Sigma$ finite alphabet, R finite set of strings from $\Sigma^*\}$
$S(\langle\Sigma, R\rangle) = \{w \in \Sigma^* : x \in R \Rightarrow (x$ is a subsequence of w, i.e. one can get x by taking away letters from $w\}$
$m(\langle\Sigma, R\rangle, w) = |w|$
$opt = \min$
APPROXIMABILITY: SHORTEST COMMON SUPERSEQUENCE $\in$ APX, SHORTEST COMMON SUPERSEQUENCE is $\overline{\text{MAX SNP}}$-hard (MIN $(1,2)$TSP $-B \leq^p_L$ SHORTEST COMMON SUPERSEQUENCE with $\alpha = 2B + 3$ and $\beta = 1$) [17].
ALGORITHM: SHORTEST COMMON SUPERSEQUENCE can be approximated within 3 [17].

[4.4] **Longest common subsequence** (LONGEST COMMON SUBSEQUENCE)

$\mathcal{I} = \{\Sigma$ finite alphabet, R finite set of strings from $\Sigma^*\}$
$S(\langle\Sigma, R\rangle) = \{w \in \Sigma^* : x \in R \Rightarrow (w$ is a subsequence of x, i.e. one can get w by taking away letters from $x\}$
$m(\langle\Sigma, R\rangle, w) = |w|$
$opt = \max$
APPROXIMABILITY:
MAX IND SET $\leq^p_L$ LONGEST COMMON SUBSEQUENCE for a Σ of the same size as the set of nodes in the MAX IND SET problem [12, 77].
The LONGEST COMMON SUBSEQUENCE problem is still NP-complete for alphabets of size two, but it is an open problem whether it is hard to approximate for constant size alphabets.

B.5 Sequencing and scheduling

[5.1] **Minimum storage-time single-processor scheduling** (MIN S/T 1-PROCESSOR SCHEDULING)

$\mathcal{I} = \{G = \langle V, E\rangle : G$ is a directed acyclic graph (DAG), $l : V \to \mathbb{N}$, $w : E \to \mathbb{N}\}$
where the nodes in G corresponds to the tasks to be scheduled, $l(v)$ is the execution time of task v and $w(v_1, v_2)$ is the storage required to save the intermediate results generated by task v_1 until it is consumed by task v_2.
$S(\langle V, E, l, w\rangle) = \{\text{permutation } \pi : [1..|V|] \to [1..|V|]\}$

$$m(\langle V, E, l, w\rangle, \pi) = \sum_{\substack{i,j \in [1..|V|] \\ (v_{\pi(i)}, v_{\pi(j)}) \in E}} w(v_{\pi(i)}, v_{\pi(j)}) \sum_{k=\min(i,j)}^{\max(i,j)} l(v_{\pi(k)})$$

$opt = \min$
ALGORITHM: MIN S/T 1-PROCESSOR SCHEDULING can be approximated within $O\left(\log|V| \log \sum\limits_{v\in V} l(v)\right)$ [94].

[5.2] **Minimum m-processor scheduling makespan** (MIN m-PROCESSOR SCHEDULING)

m is the number of processors.
$\mathcal{I} = \{T \text{ set of tasks}, l : T \times [1..m] \to \mathbb{N} \text{ execution times}\}$
$S(\langle T, l\rangle) = \{f : T \to [1..m]\}$
$m(\langle T, l\rangle, f) = \max\limits_{i\in[1..m]} \sum\limits_{\substack{t\in T:\\ f(t)=i}} l(t, i)$
$opt = \min$
APPROXIMABILITY: MIN m-PROCESSOR SCHEDULING $\in$ FPTAS [45].
If m is included in the input instance MIN m-PROCESSOR SCHEDULING cannot be approximated within $3/2 - \varepsilon$ for $\varepsilon > 0$ [68].
ALGORITHM: For all m MIN m-PROCESSOR SCHEDULING can be approximated within 2 [68].

[5.3] **Minimum m-processor scheduling makespan with speed factors** (MIN m-PROCESSOR SCHEDULING SPEED FACTORS)

$\mathcal{I} = \{T \text{ set of tasks}, l : T \to \mathbb{N} \text{ execution times}, s : [1..m] \to \mathbb{Q} \text{ speed factors such that } s(1) = 1 \text{ and } \forall i, s(i) \geq 1\}$
$S(\langle T, l, s\rangle) = \{f : T \to [1..m]\}$
$m(\langle T, l, s\rangle, f) = \max\limits_{i\in[1..m]} \sum\limits_{\substack{t\in T:\\ f(t)=i}} l(t)/s(i)$
$opt = \min$
APPROXIMABILITY:
MIN m-PROCESSOR SCHEDULING SPEED FACTORS $\in$ FPTAS [45]. If m is included in the input instance FPTAS $\not\ni$ MIN m-PROCESSOR SCHEDULING $\in$ PTAS [42].

[5.4] **Minimum uniform m-processor scheduling makespan** (MIN UNIFORM m-PROCESSOR SCHEDULING)

$\mathcal{I} = \{T \text{ set of tasks}, l : T \to \mathbb{N} \text{ execution times}\}$.
$S(\langle T, l\rangle) = \{f : T \to [1..m]\}$
$m(\langle T, l\rangle, f) = \max\limits_{i\in[1..m]} \sum\limits_{\substack{t\in T:\\ f(t)=i}} l(t)$
$opt = \min$
APPROXIMABILITY:
MIN UNIFORM m-PROCESSOR SCHEDULING $\in$ FPTAS [95], if m is included in the input instance then FPTAS $\not\ni$ MIN UNIFORM m-PROCESSOR SCHEDULING $\in$ PTAS [41].
ALGORITHM:
For all m and $\varepsilon > 0$ MIN UNIFORM m-PROCESSOR SCHEDULING can be

approximated within $1+\varepsilon$ in time $O\left((n/\varepsilon)^{1/\varepsilon^2}\right)$ [41].

[5.5] **Maximum k-multicommodity flow** (MAX k-MULTICOM FLOW)
$\mathcal{I} = \{G = \langle V,E\rangle$ a graph, $u : E \to \mathbb{Z}^+$ edge capacities, $C \subset V \times V \times \mathbb{Z}^+ :$ $|C| = k$ commodities$\}$
where $(s,t,d) \in C$ is a commodity with source s, sink t and demand d.
$S(\langle G,u,C\rangle) = \{$flow of each commodity through each edge in $G\}$
$m = \min\limits_{(s,t,d)\in C} \dfrac{\text{the flow of the commodity } (s,t,d) \text{ from } s \text{ to } t}{d}$
$opt = \max$
APPROXIMABILITY: MAX k-MULTICOM FLOW $\in$ FPTAS [59, 66].
ALGORITHM: For each $\varepsilon > 0$ MAX k-MULTICOM FLOW can be approximated within $1+\varepsilon$ in time $O\left((|E|\,|V|\log^3|V|\,k^2\log k)/\varepsilon^2\right)$ [66].

B.6 Mathematical programming

[6.1] **Minimum $0-1$ programming** (MIN $0-1$ PROGRAMMING)
$\mathcal{I} = \{A \in \mathbb{Z}^{m\cdot n}$ integer $m \times n$-matrix, $b \in \mathbb{Z}^m$ integer m-vector, $c \in \mathbb{N}^n$ non-negative integer n-vector$\}$
$S(\langle A,b,c\rangle) = \{x \in \{0,1\}^n : Ax \geq b\}$
$m(\langle A,b,c\rangle, x) = c^Tx = \sum_{i=1}^{n} c_ix_i$
$opt = \min$
APPROXIMABILITY: MIN $0-1$ PROGRAMMING is NPO-complete (MIN WEIGHTED 3SAT $\leq_s^p$ MIN $0-1$ PROGRAMMING) [83].

[6.2] **Maximum bounded $0-1$ programming** (MAX PB $0-1$ PROGRAMMING)
$\mathcal{I} = \{A \in \mathbb{Z}^{m\cdot n}$ integer $m \times n$-matrix, $b \in \mathbb{Z}^m$ integer m-vector, $c \in \{0,1\}^n$ binary n-vector$\}$
$S(\langle A,b,c\rangle) = \{x \in \{0,1\}^n : Ax \leq b\}$
$m(\langle A,b,c\rangle, x) = c^Tx = \sum_{i=1}^{n} c_ix_i$
$opt = \max$
APPROXIMABILITY:
MAX PB $0-1$ PROGRAMMING is NPO PB-complete (LP WITH FORBIDDEN PAIRS $\leq_P^p$MAX PB $0-1$ PROGRAMMING) [12].

[6.3] **Minimum generalized $0-1$ assignment** (MIN $0-1$ ASSIGNMENT)
$\mathcal{I} = \{A \in \mathbb{Z}^{m\cdot n}$ integer $m \times n$-matrix, $b \in \mathbb{Z}^m$ integer m-vector, $C \in \{0,1\}^{m\cdot n}$ binary $m \times n$-matrix$\}$
$S(\langle A,C\rangle) = \{X \in \{0,1\}^{m\cdot n}$ binary $m \times n$-matrix such that $\forall i \in [1..m]$,
$\sum_{j=1}^{n} A_{i,j}X_{i,j} \leq b_i$ and there is exactly one 1 in each column of $X\}$

$m(\langle A,C\rangle, X) = \sum_{i=1}^{m}\sum_{j=1}^{n} C_{i,j}X_{i,j}$
$opt = \min$
APPROXIMABILITY: MIN $0-1$ ASSIGNMENT $\neq$ APX [96].

[6.4] **Minimum quadratic** $0-1$ **assignment** (MIN QUADRATIC $0-1$ ASSIGNMENT)

$\mathcal{I} = \{C \in \mathbb{N}^{n\cdot n}$ non-negative integer $n\times n$-matrix, $D \in \mathbb{N}^{m\cdot m}$ non-negative integer $m\times m$-matrix$\}$
$S(\langle C,D\rangle) = \{X \in \{0,1\}^{n\cdot m}$ binary $n\times m$-matrix such that there is at most one 1 in each row of X and exactly one 1 in each column of $X\}$
$m(\langle C,D\rangle, X) = \sum_{\substack{i,j=1\\ i\neq j}}^{n}\sum_{\substack{k,l=1\\ k\neq l}}^{m} C_{i,j}D_{k,l}X_{i,k}X_{j,l}$
$opt = \min$
APPROXIMABILITY: MIN QUADRATIC $0-1$ ASSIGNMENT $\notin$ APX [96].

[6.5] **Minimum planar record packing** (MIN PLANAR RECORD PACKING)

$\mathcal{I} = \{p = (p_1,\ldots,p_n)$ vector of n probabilities, $0 \le p_i \le 1\}$
$S(p) = \{C \subseteq \mathbb{Z}^2 : |C| = n$ set of integer coordinates in the plane$\}$
$m(p,C) = \sum_{i=1}^{n}\sum_{j=1}^{n} p_i p_j d(c_i,c_j)$
where $d(c_i,c_j)$ is the discretized Euclidean distance between the points c_i and c_j in C.
$opt = \min$
ALGORITHM: MIN PLANAR RECORD PACKING can be approximated with an absolute error guarantee of $\lfloor 4\sqrt{2}+8\sqrt{\pi}\rfloor$, that is one can in polynomial time find a solution with objective function value at most $opt(p) + \lfloor 4\sqrt{2}+8\sqrt{\pi}\rfloor$ [57].

[6.6] **Knapsack** (MAX KNAPSACK)

$\mathcal{I} = \{\langle U,s,v,b\rangle : U$ is a finite set, $s : U \to \mathbb{Z}^+, v : U \to \mathbb{Z}^+, b \in \mathbb{Z}^+\}$
$S(\langle U,s,v,b\rangle) = \{U' \subseteq U : \sum_{u\in U'} s(u) \le b\}$
$m(\langle U,s,v,b\rangle, U') = \sum_{u\in U'} v(u)$
$opt = \max$
APPROXIMABILITY: MAX KNAPSACK $\in$ FPTAS [46].

[6.7] **Integer m-dimensional knapsack** (MAX INTEGER m-DIMENSIONAL KNAPSACK)

$\mathcal{I} = \{A \in \mathbb{N}^{m\cdot n}$ non-negative integer $m\times n$-matrix, $b \in \mathbb{N}^m$ non-negative integer m-vector, $c \in \mathbb{N}^n$ non-negative integer n-vector$\}$
$S(\langle A,b,c\rangle) = \{x \in \mathbb{N}^n : Ax \le b\}$

$m(\langle A, b, c\rangle, x) = c^T x = \sum_{i=1} n c_i x_i$
opt = max
APPROXIMABILITY: For constant m MAX INTEGER m-DIMENSIONAL KNAPSACK $\in$ PTAS [20].
ALGORITHM: For each $\varepsilon > 0$ MAX INTEGER m-DIMENSIONAL KNAPSACK can be approximated within $1 + \varepsilon$ in time $O(n^{\lceil m/\varepsilon \rceil})$ [20].

[6.8] **Integer k-choice knapsack** (MAX INTEGER k-CHOICE KNAPSACK)

$\mathcal{I} = \{k \in \mathbb{Z}^+, A, C \in \mathbb{N}^{k \cdot n}$ non-negative integer $k \times n$-matrices, $b \in \mathbb{N}$ non-negative integer$\}$
$S(\langle k, A, C, b\rangle) = \{x \in \mathbb{N}^n, f : [1..n] \to [1..k]$ such that $\sum_{i=1}^{n} a_{i,f(i)} x_i \leq b\}$
$m(\langle k, A, C, b\rangle, \langle x, f\rangle) = \sum_{i=1} n c_{i,f(i)} x_i$
opt = max
APPROXIMABILITY: MAX INTEGER k-CHOICE KNAPSACK $\in$ FPTAS [20].

B.7 Logic

[7.1] **Maximum satisfiability** (MAX SAT)

$\mathcal{I} = \{\langle U, C\rangle : U$ finite set of variables, C set of disjunctive clauses of literals$\}$
A literal is a variable or a negated variable.
$S(\langle U, C\rangle) = \{C' \subseteq C :$ there is a truth assignment for U such that every clause in C' is satisfied$\}$
$m(\langle U, C\rangle, C') = |C'|$
opt = max
APPROXIMABILITY: MAX SAT $\in$ MAX Σ_1 [88], MAX SAT is $\overline{\text{MAX SNP}}$-hard [88], MAX SAT $\notin$ MAX Σ_0 [61].
ALGORITHM: MAX SAT can be approximated within 4/3 [110].

[7.2] **Maximum k-satisfiability** (MAX kSAT)

$\mathcal{I} = \{\langle U, C\rangle : U$ finite set of variables, C set of disjunctive clauses, each involving at most k literals, where $k \geq 2\}$
$S(\langle U, C\rangle) = \{C' \subseteq C :$ there is a truth assignment for U such that every clause in C' is satisfied$\}$
$m(\langle U, C\rangle, C') = |C'|$
opt = max
APPROXIMABILITY:
MAX kSAT is MAX Σ_0-complete for every constant $k \geq 2$ [88].
The planar version of MAX 3SAT $\in$ PTAS$^\infty$, see Section 8.2.
ALGORITHM: MAX kSAT can be approximated within $1/(1 - 2^{-k})$ if every clause consists of exactly k literals [48].

[7.3] **Maximum bounded k-satisfiability** (Max kSat $-B$)

This is the same problem as Max kSat but the total number of occurrences of each variable is bounded by the constant B.
Approximability: Max kSat $-B$ is Max Σ_0-complete for every constant $k \geq 3$ if $B \geq 6$ or if there are at most 4 occurrences of each literal (Max kSat $\leq_L^p$ Max kSat $-B$) [88]. There is a constant B such that Max 2Sat $-B$ is Max Σ_0-complete [88].
Algorithm: See Max kSat.

[7.4] **Minimum** 3DNF **satisfiability** (Min 3DNF Sat)

$\mathcal{I} = \{\langle U, C\rangle : U$ finite set of variables, C set of conjunctive clauses, each involving at most three literals$\}$
$S(\langle U, C\rangle) = \{C' \subseteq C :$ there is a truth assignment for U such that every clause in C' is satisfied$\}$
$m(\langle U, C\rangle, C') = |C'|$
$opt = \min$
Min 3DNF Sat is the optimization problem version of the NP-complete problem Non-tautology of 3dnf formulas [35].
Approximability:
Min 3DNF Sat $\notin$ Apx, Min 3DNF Sat is Min Σ_0-complete under P-reductions [61].

[7.5] **Maximum not-all-equal 3-satisfiability** (Max Not-all-equal 3-Sat)

$\mathcal{I} = \{\langle U, C\rangle : U$ finite set of variables, C set of disjunctive clauses, each involving at most 3 literals$\}$
$S(\langle U, C\rangle) = \{C' \subseteq C :$ there is a truth assignment for U such that each clause in C' has at least one true literal and at least one false literal$\}$
$m(\langle U, C\rangle, C') = |C'|$
$opt = \max$
Approximability: Max Not-all-equal 3Sat is Max Σ_0-complete (Max 2Sat $\leq_L^p$ Max Not-all-equal 3Sat with $\alpha = \beta = 1$) [88].

[7.6] **Maximum generalized k-satisfiability** (Max G kSat)

$\mathcal{I} = \{\langle U, C\rangle : U$ finite set of variables, C set of clauses, where each clause is a disjunction of conjunctions of literals and each conjunction consists of at most k literals$\}$
$S(\langle U, C\rangle) = \{C' \subseteq C :$ there is a truth assignment for U such that every clause in C' is satisfied$\}$
$m(\langle U, C\rangle, C') = |C'|$
$opt = \max$
Approximability: Max G kSat $\in$ Max Σ_1 for each $k > 0$ [88],
Max G 1Sat=Max Sat, $\exists k > 1$:Max G kSat is $\overline{\text{Max NP}}$-hard [88],
Max G kSat $\notin$ Max Σ_0 (see Max Sat).

[7.7] **Minimum weighted satisfiability** (Min Weighted Sat)

$\mathcal{I} = \{\langle U, F, w\rangle : U$ finite set of variables, F boolean formula, $w : U \to \mathbb{N}$ weight function$\}$
$S(\langle U, F, w\rangle) = \{U' \subseteq U : F$ is satisfied when the variables in U' are set to 1 and the variables in $U - U'$ are set to 0$\}$
$m(\langle U, F, w\rangle, U') = \sum_{v \in U'} w(v)$
$opt = \min$
APPROXIMABILITY: MIN WEIGHTED SAT is NPO-complete [83].

[7.8] **Minimum weighted 3-satisfiability** (MIN WEIGHTED 3SAT)

This is the same problem as MIN WEIGHTED SAT but the formula must be in 3CNF.
APPROXIMABILITY: MIN WEIGHTED 3SAT is NPO-complete [83].

[7.9] **Maximum weighted satisfiability** (MAX WEIGHTED SAT)

$\mathcal{I} = \{\langle U, C, w\rangle : U$ finite set of variables, C set of clauses, where each clause is a disjunction of conjunctions of literals, $w : C \to \mathbb{N}$ weight function$\}$
$S(\langle U, C, w\rangle) = \{C' \subseteq C :$ there is a truth assignment for U such that every clause in C' is satisfied$\}$
$m(\langle U, C, w\rangle, C') = \sum_{c \in C'} w(c)$
$opt = \max$
APPROXIMABILITY: MAX WEIGHTED SAT is $\overline{\text{MAX NP}}$-hard (MAX SAT $\leq_L^p$ MAX WEIGHTED SAT with $\alpha = \beta = 1$).
ALGORITHM:
MAX WEIGHTED SAT can be approximated within 4/3 [110].

[7.10] **Maximum weighted satisfiability with bound** (MAX WEIGHTED SAT WITH BOUND)

$\mathcal{I} = \{\langle U, F, B, w\rangle : U$ finite set of variables, F boolean formula, $B \in \mathbb{N}$, $w : U \to \mathbb{N}$ weight function such that $\sum_{u \in U} w(u) \leq 2B\}$
$S(\langle U, F, B, w\rangle) = \{U' \subseteq U\}$

$$m(\langle U, F, B, w\rangle, U') = \begin{cases} \sum_{v \in U'} w(v) & \text{if } F \text{ is satisfied when the variables in } U' \text{ are set to 1 and the variables in } U - U' \text{ are set to 0,} \\ B & \text{otherwise.} \end{cases}$$

$opt = \max$
APPROXIMABILITY:
MAX WEIGHTED SAT WITH BOUND is APX-complete with respect to the P-reduction [24]. See Section 4.6.

[7.11] **Maximum weighted satisfiability with small bound** (MAX WEIGHTED SAT WITH SMALL BOUND)

$\mathcal{I} = \{\langle U, F, B, w\rangle : U$ finite set of variables, F boolean formula, $B \in \mathbb{N}$, $w : U \to \mathbb{N}$ weight function such that $\sum_{u \in U} w(u) \leq (1 + 1/(|U| - 1))\ B\}$
$S(\langle U, F, B, w\rangle) = \{U' \subseteq U\}$

$$m(\langle U, F, B, w\rangle, U') = \begin{cases} \sum_{v \in U'} w(v) & \text{if } F \text{ is satisfied when the variables in } U' \text{ are set to 1 and the variables in } U - U' \text{ are set to 0,} \\ B & \text{otherwise.} \end{cases}$$

opt = max
Approximability:
Max Weighted Sat with Bound is Ptas-complete with respect to the F-reduction [24]. See Section 4.3.

[7.12] **Maximum number of satisfiable formulas** (Max # Sat)

$\mathcal{I} = \{\langle U, S\rangle : U$ finite set of variables, S set of 3CNF formulas$\}$
$S(\langle U, S\rangle) = \{U' \subseteq U\}$
$m(\langle U, S\rangle, U') = |\{F \in S : F$ is satisfied when the variables in U' are set to 1 and the variables in $U - U'$ are set to $0\}|$
opt = max
Approximability:
Max # Sat is Max Π_1-complete [84], Max # Sat is NPO PB-complete (LIP $\leq_L^p$ Max # Sat with $\alpha = \beta = 1$), see Theorem 4.32.

[7.13] **Max distinguished ones** (Max Dones)

$\mathcal{I} = \{\langle X, Z, C\rangle : X$ and Z finite set of variables, C set of disjunctive clauses, each involving at most 3 literals$\}$
$S(\langle X, Z, C\rangle) = \{\langle X', Z'\rangle : X' \subseteq X \wedge Z' \subseteq Z \wedge$ every clause in C is satisfied when the variables in X' and Z' are set to 1 and the variables in $X - X'$ and $Z - Z'$ are set to $0\}$
$m(\langle X, Z, C\rangle, \langle X', Z'\rangle) = |Z'|$
opt = max
Approximability: Max Dones is Max Π_1-complete [84], Max Dones is NPO PB-complete (Max # Sat $\leq_L^p$ Max Dones with $\alpha = \beta = 1$ [84]), see Theorem 4.32.

[7.14] **Max ones** (Max Ones)

$\mathcal{I} = \{\langle U, F\rangle : U$ finite set of variables, F boolean formula in 3CNF$\}$
$S(\langle U, F\rangle) = \{U' \subseteq U : F$ is satisfied when the variables in U' are set to 1 and the variables in $U - U'$ are set to $0\}$
$m(\langle U, F\rangle, U') = |U'|$
opt = max
Approximability: Max Ones is Max Π_1-complete [84], Max Ones is NPO PB-complete (Max Dones $\leq_P^p$ Max Ones [84]), see Theorem 4.32.

[7.15] **Max k ones with negated variables** (MAX k ONES NEG)

$\mathcal{I} = \{\langle U, F\rangle : U$ finite set of variables, F boolean formula in kCNF where every variable appears negated$\}$
$S(\langle U, F\rangle) = \{U' \subseteq U : F$ is satisfied when the variables in U' are set to 1 and the variables in $U - U'$ are set to 0$\}$
$m(\langle U, F\rangle, U') = |U'|$
$opt = \max$
APPROXIMABILITY: MAX k ONES NEG is MAX $\mathrm{F}^{-}\Pi_1(k)$-complete [84].

[7.16] **Minimum equivalence deletion** (MIN $\equiv$DELETION)

$\mathcal{I} = \{\langle U, C\rangle : U$ finite set of variables, C set of pairs of literals$\}$
$S(\langle U, C\rangle) = \{C' \subseteq C :$ there is a truth assignment for U such that $l_1 \equiv l_2$ for every pair $(l_1, l_2) \in C'\}$
$m(\langle U, C\rangle, C') = |C| - |C'|$
$opt = \min$
ALGORITHM:
MIN $\equiv$DELETION can be approximated within $O(\log^3 |C|)$ [59].

[7.17] **Maximum k-constraint satisfaction** (MAX k-CONSTRAINT SAT)

$\mathcal{I} = \{\langle U, C\rangle : U$ finite set of variables, C vector of conjunctions, each involving at most k literals, where $k \geq 2\}$
$S(\langle U, C\rangle) = \{C' \subseteq C :$ there is a truth assignment for U such that every conjunction in C' is satisfied$\}$
$m(\langle U, C\rangle, C') = |C'|$
$opt = \max$
APPROXIMABILITY: MAX k-CONSTRAINT SAT is MAX Σ_0-complete for every constant $k \geq 2$ when the same conjunction appears at most a constant number of times in the input (MAX 2SAT $\leq_L^p$ MAX 2-CONSTRAINT SAT with $\alpha = \beta = 1$), see Theorem A.3.
ALGORITHM:
MAX k-CONSTRAINT SAT can be approximated within 2^k [12].
If MAX 2-CONSTRAINT SAT cannot be approximated within $o(1)$, then MAX k-CONSTRAINT SAT cannot be approximated within $2^{o(k)}$ when $k \leq \log |C|$ [12].

B.8 Automata and language theory

[8.1] **Minimum consistent deterministic finite automata** (MIN CONSISTENT DFA)

$\mathcal{I} = \{\langle P, N\rangle : P, N$ sets of binary strings$\}$
$S(\langle P, N\rangle) = \{A = \langle Q, \{0,1\}, \delta, q_0, F\rangle$ a deterministic finite automata accepting all strings in P and rejecting all strings in $N\}$
$m(\langle P, N\rangle, A) = |Q|$ number of states in the automata.
APPROXIMABILITY: MIN GRAPH COLOURING $\leq_r^{1+\varepsilon}$ MIN CONSISTENT DFA with a size amplification of $|V|/\varepsilon$ [101],

For each $\varepsilon > 0$ MIN CONSISTENT DFA cannot be approximated within $(|P| + |N|)^{1/14-\varepsilon}$ [92].

[8.2] **Longest computation** (LONGEST COMPUTATION)
$\mathcal{I} = \{\langle M, x\rangle : M$ nondeterministic Turing machine, x binary string$\}$
$S(\langle M, x\rangle) = \{c$ guess string produced by M on input $x\}$
$m(\langle M, x\rangle, c) = \min(|x|, |c|)$ where $|c|$ denotes the length of the guess string c
$opt = \max$
APPROXIMABILITY: LONGEST COMPUTATION is NPO PB-complete [12].

I would like to express my sincere gratitude to those, friends and colleagues, who in various ways helped me throughout this study:

Stefan Arnborg and *Germund Dahlquist* who interested me in the field of theoretical computer science and helped me to get started.

Johan Håstad, my supervisor, for drawing my attention to the area of complexity and especially approximation, always giving me new ideas and suggestions, and for his continuous encouragement and invaluable support. I have many times made use of his great knowledge and broad network of contacts.

Mikael Goldmann, for careful reading of the whole of my thesis and many suggestions for improvements.

The rest of the TD project group at NADA, *Lars Langemyr*, *Jens Lagergren*, *Joachim Hollman*, *Henry Feng*, *Kalev Kask* and *Ulf Berggren* for always being willing to discuss questions related to and unrelated to this work.

Magnus Halldórsson for constantly informing me about what in the approximation area is currently going on in the world.

All the colleagues at NADA, in particular the computer group for maintaining the computers.

Fysikalen's writer's group *Björn-Olof Skytt*, *Bengt Erlandsson*, *Johan Groth*, *Johan Karlander*, *Pontus Lindblad*, *Mats Löfdahl*, *Håkan Huss* and *Kimmo Eriksson* for continuously giving me writing practice and a good laugh.

My wife *Linda* and my family *Gerd*, *Henrik*, *Kimmo* and *Jonna Eriksson* for always being accessible for talks, questions, help, proofreading, music and much more.

The faculty of computer science at KTH "sektion D" for my graduate position "doktorandtjänst", which made this work possible.

Thanks, all of you!

Index

Problem index

General index

Bibliography

[1] A. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Mass., 1974.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proc. of 33rd Annual IEEE Sympos. on Foundations of Computer Science*, pages 14–23, 1992.

[3] S. Arora and S. Safra. Probabilistic checking of proofs; a new characterization of NP. In *Proc. of 33rd Annual IEEE Sympos. on Foundations of Computer Science*, pages 2–13, 1992.

[4] G. Ausiello, A. D'Atri, and M. Protasi. Structure preserving reductions among convex optimization problems. *Journal of Computer and System Sciences*, 21:136–153, 1980.

[5] G. Ausiello, A. Marchetti-Spaccamela, and M. Protasi. Toward a unified approach for the classification of NP-complete optimization problems. *Theoretical Computer Science*, 12:83–96, 1980.

[6] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. In *Proc. of 24th Annual IEEE Sympos. on Foundations of Computer Science*, pages 265–273, 1983.

[7] J. Bar-Ilan and D. Peleg. Approximation algorithms for selecting network centers. In *Algorithms and Data structures*, pages 343–354. Springer-Verlag, 1991. Lecture Notes in Computer Science 519.

[8] R. Bar-Yehuda and S. Even. *A local-ratio theorem for approximating the weighted vertex cover problem*, volume 25 of *Annals of Discrete Mathematics*, pages 27–46. Elsevier science publishing company, Amsterdam, 1985.

[9] H. Barrow and R. Burstall. Subgraph isomorphism, matching relational structures and maximal cliques. *Information Processing Letters*, 4:83–84, 1976.

[10] F. Berman, D. Johnson, T. Leighton, P. W. Shor, and L. Snyder. Generalized planar matching. *J. Algorithms*, 11:153–184, 1990.

[11] P. Berman and V. Ramaiyer. Improved approximations for the Steiner tree problem. In *Proc. Third Annual ACM-SIAM Symp. on Discrete Algorithms*, 1992.

[12] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. In *Proc. 6th Annual Symposium on Theoretical Aspects of Computer Science*, pages 256–268. Springer-Verlag, 1989. Lecture Notes in Computer Science 349.

[13] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. *Inform. and Comput.*, 96:77–94, 1992.

[14] M. Bern and P. Plassmann. The Steiner problem with edge lengths 1 and 2. *Information Processing Letters*, 32:171–176, 1989.

[15] A. Blum. An $O(n^{0.4})$-approximation algorithm for 3-coloring. In *Proc. Twenty first Annual ACM symp. on Theory of Comp.*, pages 535–542. ACM, 1989.

[16] A. Blum. Interactive protocols and clique lower bounds. Technical report, MIT, Laboratory for Computer Science, 1991.

[17] A. Blum, T. Jiang, M. Li, J. Tromp, and M. Yannakakis. Linear approximation of shortest superstrings. In *Proc. Twenty third Annual ACM symp. on Theory of Comp.*, pages 328–336. ACM, 1991.

[18] R. Boppana and M. M. Halldórsson. Approximating maximum independent sets by excluding subgraphs. *Bit*, 32(2):180–196, 1992.

[19] D. Bruschi, D. Joseph, and P. Young. A structural overview of NP optimization problems. In *Proc. Optimal Algorithms*, pages 205–231. Springer-Verlag, 1989. Lecture Notes in Computer Science 401.

[20] A. K. Chandra, D. S. Hirschberg, and C. K. Wong. Approximate algorithms for some generalized knapsack problems. *Theoretical Computer Science*, 3:293–304, 1976.

[21] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, 1976.

[22] E. G. Coffman, Jr, M. R. Garey, and D. S. Johnson. *Approximation Algorithms for Bin-Packing – An Updated Survey*, pages 49–106. SpV, New York, 1984.

[23] P. Crescenzi, C. Fiorini, and R. Silvestri. A note on the approximation of the MAX CLIQUE problem. *Information Processing Letters*, 40:1–5, 1991.

[24] P. Crescenzi and A. Panconesi. Completeness in approximation classes. *Inform. and Comput.*, 93(2):241–262, 1991.

[25] H. N. Djidjev. On the problem of partitioning planar graphs. *SIAM J. Alg. Disc. Meth.*, 3(2):229–240, 1982.

[26] D. Z. Du and F. K. Hwang. An approach for proving lower bounds: solution of Gilbert-Pollak's conjecture on Steiner ratio. In *Proc. of 31st Annual IEEE Sympos. on Foundations of Computer Science*, pages 76–85, 1991.

[27] D. Z. Du, Y. Zhang, and Q. Feng. On better heuristic for Euclidean Steiner minimum trees. In *Proc. of 32nd Annual IEEE Sympos. on Foundations of Computer Science*, pages 431–439, 1991.

[28] M. E. Dyer and A. M. Frieze. Planar 3DM is NP-complete. *J. Algorithms*, 11:174–184, 1986.

[29] R. Fagin. Generalized first-order spectra, and polynomial-time recognizable sets. In R. Karp, editor, *Complexity and Computations*. AMS, 1974.

[30] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proc. of 32nd Annual IEEE Sympos. on Foundations of Computer Science*, pages 2–12, 1991.

[31] G. N. Frederickson, M. S. Hecht, and C. E. Kim. Approximation algorithms for some routing problems. *SIAM Journal on Computing*, 7:178–193, 1978.

[32] M. Fürer and B. Raghavachari. Approximating the minimum degree spanning tree to within one from the optimal degree. In *Proc. Third Annual ACM-SIAM Symp. on Discrete Algorithms*, 1992.

[33] M. R. Garey and D. S. Johnson. *Approximation algorithms for combinatorial problems: an annotated bibliography*, pages 41–52. Academic Press, New York, 1976.

[34] M. R. Garey and D. S. Johnson. Strong NP completeness results: Motivations, examples and implications. *Journal of the ACM*, 25:499–508, 1978.

[35] M. R. Garey and D. S. Johnson. *Computers and Intractability: a guide to the theory of NP-completeness*. W. H. Freeman and Company, San Francisco, 1979.

[36] O. Goldschmidt and D. S. Hochbaum. Polynomial algorithm for the k-cut problem. In *Proc. of 29th Annual IEEE Sympos. on Foundations of Computer Science*, pages 444–451, 1988.

[37] M. M. Halldórsson. Approximating the independent set and vertex cover problems. Manuscript, 1992.

[38] M. M. Halldórsson. A still better performance guarantee for approximate graph coloring. *Information Processing Letters*, 45(1):19–23, 1993.

[39] D. S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11(3):555–556, 1982.

[40] D. S. Hochbaum and D. B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM*, 33(3):533–550, 1986.

[41] D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: theoretical and practical results. *Journal of the ACM*, 34(1):144–162, 1987.

[42] D. S. Hochbaum and D. B. Shmoys. A polynomial approximation scheme for machine scheduling on uniform processors: using the dual approach. *SIAM Journal on Computing*, 17(3):539–551, 1988.

[43] U. Holyer. The NP-completeness of edge-coloring. *SIAM Journal on Computing*, 10:718–720, 1981.

[44] J. E. Hopcroft and R. M. Karp. A $n^{5/2}$ algorithm for maximum matching in bipartite graphs. *SIAM Journal on Computing*, 2:225–231, 1973.

[45] E. Horowitz and S. K. Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the ACM*, 23(2):317–327, 1976.

[46] O. H. Ibarra and C. E. Kim. Fast approximation for the knapsack and sum of subset problems. *Journal of the ACM*, 22(4):463–468, 1975.

[47] R. W. Irving. On approximating the minimum independent dominating set. *Information Processing Letters*, 37:197–200, 1991.

[48] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[49] D. S. Johnson. Local optimization and the traveling salesman problem. In *Proc. ICALP-90*, pages 446–461. Springer-Verlag, 1990. Lecture Notes in Computer Science 443.

[50] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37:79–100, 1988.

[51] V. Kann. An algorithm solving the Euclidean travelling salesperson problem. Technical Report TRITA-NA-9102, ISSN 0348-2952, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, 1991.

[52] V. Kann. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters*, 37:27–35, 1991.

[53] V. Kann. Maximum bounded H-matching is MAX SNP-complete. Manuscript, submitted for publication, 1991.

[54] V. Kann. Which definition of MAX SNP is the best? Manuscript, submitted for publication, 1991.

[55] V. Kann. On the approximability of the maximum common subgraph problem. In *Proc. 9th Annual Symposium on Theoretical Aspects of Computer Science*, pages 377–388. Springer-Verlag, 1992. Lecture Notes in Computer Science 577.

[56] N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin packing problem. In *Proc. of 23rd Annual IEEE Sympos. on Foundations of Computer Science*, pages 312–320, 1982.

[57] R. M. Karp, A. C. McKellar, and C. K. Wong. Near-optimal solutions to a 2-dimensional placement problem. *SIAM Journal on Computing*, 4(3):271–286, 1975.

[58] D. Kavadias, L. M. Kirousis, and P. Spirakis. The complexity of the reliable connectivity problem. *Information Processing Letters*, 39:245–252, 1991.

[59] P. Klein, A. Agrawal, R. Ravi, and S. Rao. Approximation through multicommodity flow. In *Proc. of 31st Annual IEEE Sympos. on Foundations of Computer Science*, pages 726–737, 1990.

[60] P. G. Kolaitis and M. N. Thakur. Approximation properties of NP minimization classes. In *Proc. 6th Annual Conf. on Structures in Computer Science*, pages 353–366, 1991.

[61] P. G. Kolaitis and M. N. Thakur. Logical definability of NP optimization problems. Technical Report UCSC-CRL-93-10, Board of Studies in Computer and Information Sciences, University of California at Santa Cruz, 1993. To be published in Information and Computation.

[62] L. T. Kou, L. J. Stockmeyer, and C. K. Wong. Covering edges by cliques with regard to keyword conflicts and intersection graphs. *Communications of the ACM*, 21(2):135–139, 1978.

[63] M. W. Krentel. The complexity of optimization problems. *Journal of Computer and System Sciences*, 36:490–509, 1988.

[64] M. W. Krentel. Structure in locally optimal solutions. In *Proc. of 30th Annual IEEE Sympos. on Foundations of Computer Science*, pages 216–221, 1989.

[65] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The Traveling Salesman Problem.* John Wiley & Sons, 1985.

[66] T. Leighton, F. Makedon, S. Plotkin, C. Stein, E. Tardos, and S. Tragoudas. Fast approximation algorithms for multicommodity flow problems. In *Proc. Twenty third Annual ACM symp. on Theory of Comp.*, pages 101–111, 1991.

[67] T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *Proc. of 29th Annual IEEE Sympos. on Foundations of Computer Science*, pages 422–431, 1988.

[68] J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. In *Proc. of 28th Annual IEEE Sympos. on Foundations of Computer Science*, pages 217–224, 1987.

[69] K. Li and K. Cheng. On three-dimensional packing. *SIAM Journal on Computing*, 19(5):847–867, 1990.

[70] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

[71] N. Linial and U. Vazirani. Graph products and chromatic numbers. In *Proc. of 30th Annual IEEE Sympos. on Foundations of Computer Science*, pages 124–128, 1989.

[72] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math.*, 36:177–189, 1979.

[73] R. J. Lipton and R. E. Tarjan. Applications of a planar separator theorem. *SIAM Journal on Computing*, 9(3):615–627, 1980.

[74] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975.

[75] L. Lovász and M. D. Plummer. *Matching Theory*, volume 29 of *Annals of Discrete Mathematics.* Elsevier science publishing company, Amsterdam, 1986.

[76] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. In *Proc. Twenty fifth Annual ACM symp. on Theory of Comp.*, pages 286–293, 1993.

[77] D. Maier. The complexity of some problems on subsequences and supersequences. *Journal of the ACM*, 25:322–336, 1978.

[78] S. Micali and V. V. Vazirani. An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In *Proc. of 21st Annual IEEE Sympos. on Foundations of Computer Science*, pages 17–27, 1980.

[79] G. L. Miller. Finding small simple cycle separators for 2-connected planar graphs. *Journal of Computer and System Sciences*, 32:265–279, 1986.

[80] B. Monien and E. Speckenmeyer. Some further approximation algorithms for the vertex cover problem. In *Proc. CAAP '83*, pages 341–349. Springer-Verlag, 1983. Lecture Notes in Computer Science 159.

[81] R. Motwani and G. D. S. Ramkumar. On finding long paths in (weakly) Hamiltonian graphs. Manuscript, 1991.

[82] T. Nishizeki and N. Chiba. *Planar Graphs: Theory and Algorithms*, volume 32 of *Annals of Discrete Mathematics.* Elsevier science publishing company, Amsterdam, 1988.

[83] P. Orponen and H. Mannila. On approximation preserving reductions: Complete problems and robust measures. Technical Report C-1987-28, Department of Computer Science, University of Helsinki, 1987.

[84] A. Panconesi and D. Ranjan. Quantifiers and approximation. In *Proc. Twenty second Annual ACM symp. on Theory of Comp.*, pages 446–456. ACM, 1990.

[85] C. H. Papadimitriou. The Euclidean travelling salesman problem is NP-complete. *Theoretical Computer Science*, 4:237–244, 1977.

[86] C. H. Papadimitriou, 1990. Private communication.

[87] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization, Algorithms and Complexity.* Prentice Hall, 1982.

[88] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.

[89] C. H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 1992. To appear.

[90] A. Paz and S. Moran. Non deterministic polynomial optimization problems and their approximations. *Theoretical Computer Science*, 15:251–277, 1981.

[91] J. G. Peters and L. Rudolph. Parallel approximation schemes for subset sum and knapsack problems. *Acta Informatica*, 24:417–432, 1987.

[92] L. Pitt and M. K. Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. In *Proc. Twenty first Annual ACM symp. on Theory of Comp.*, pages 421–432. ACM, 1989.

[93] S. Rao. Finding near optimal separators in planar graphs. In *Proc. of 28th Annual IEEE Sympos. on Foundations of Computer Science*, pages 225–236, 1987.

[94] R. Ravi, A. Agrawal, and P. Klein. Ordering problems approximated: single-processor scheduling and interval graph completion. In *Automata, Languages and Programming*, pages 751–762. Springer-Verlag, 1991. Lecture Notes in Computer Science 510.

[95] S. K. Sahni. Algorithms for scheduling independent tasks. *Journal of the ACM*, 23(1):116–127, 1976.

[96] S. K. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23(3):555–565, 1976.

[97] H. Saran and V. Vazirani. Finding k-cuts within twice the optimal. In *Proc. of 32nd Annual IEEE Sympos. on Foundations of Computer Science*, pages 743–751, 1991.

[98] M. Serna. Approximating linear programming is log-space complete for P. *Information Processing Letters*, 37:233–236, 1991.

[99] M. Serna and P. Spirakis. The approximability of problems complete for p. In *Proc. Optimal Algorithms*, pages 193–204. Springer-Verlag, 1989. Lecture Notes in Computer Science 401.

[100] W. Shih, S. Wu, and Y. S. Kuo. Unifying maximum cut and minimum cut of a planar graph. *IEEE Trans. on Computers*, 39(5):694–697, 1990.

[101] H. U. Simon. On approximate solutions for combinatorial optimization problems. *SIAM J. Disc. Math.*, 3(2):294–310, 1990.

[102] W. D. Smith, 1990. Unpublished result.

[103] L. J. Stockmeyer. The polynomial time hierarchy. *Theoretical Computer Science*, 3:1–22, 1976.

[104] L. J. Stockmeyer. On approximation algorithms for #P. *SIAM Journal on Computing*, 14(4):849–861, 1985.

[105] K. J. Supowit, E. M. Reingold, and D. A. Plaisted. The traveling salesman problem and minimum matching in the unit square. *SIAM Journal on Computing*, 12(1):144–156, 1983.

[106] P. Vitanyi. How well can a graph be n-colored? *Annals of Discrete Mathematics*, 21, 1981.

[107] M. Yannakakis. The effect of a connectivity requirement on the complexity of maximum subgraph problems. *Journal of the ACM*, 26:618–630, 1979.

[108] M. Yannakakis. Edge deletion problems. *SIAM Journal on Computing*, 10:297–309, 1981.

[109] M. Yannakakis. The analysis of local search problems and their heuristics. In *Proc. 7th Annual Symposium on Theoretical Aspects of Computer Science*, pages 298–311. Springer-Verlag, 1990. Lecture Notes in Computer Science 415.

[110] M. Yannakakis. On the approximation of maximum satisfiability. In *Proc. Third Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 1–9, 1992.

[111] X. Yao. Finding approximate solutions to NP-hard problems by neural networks is hard. *Information Processing Letters*, 41:93–98, 1992.