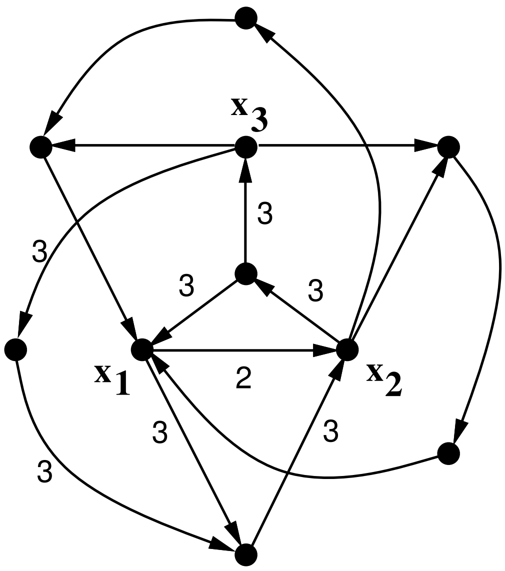
Università degli Studi di Roma "La Sapienza"

Dottorato di Ricerca in Informatica

Reductions and (Non-)Approximability



Luca Trevisan

IX-97-7

Luca Trevisan

Reductions and (Non-)Approximability

Dottorato di Ricerca in Informatica

IX-97-7

Università degli Studi di Roma "La Sapienza"

Author's address:
Luca Trevisan
Dipartimento di Scienze dell'Informazione
Università degli Studi di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
e-mail: trevisan@dsi.uniroma1.it
www: http://www.dsi.uniroma1.it/~trevisan

Abstract

In this thesis we study approximation-preserving reductions among combinatorial optimization problems, and we use them to prove *completeness results* in *approximation classes*, as well as stronger *non-approximability results* and the existence of improved *approximation algorithms* for specific problems.

The first part of the thesis deals with the way of correctly *defining* the notion of approximation preserving reducibility. Our emphasis here is on completeness results in approximation classes. Several different approximation-preserving reducibilities have been defined in the last decade. We prove that none of them allows to prove certain natural completeness results, such as the fact that the Maximum Satisfiability Problem is complete for the class of problems approximable with a constant factor. We also show that the L-reducibility, the most widely used reducibility so far, has very bad closure properties: it does not even preserve constant factor approximability. These negative results are proved under complexity-theoretic assumptions. We propose a new reducibility (the AP-reducibility) that overcomes all the limitations of previous definitions, and we prove several completeness results using the AP-reducibility. By making use of the PTAS-reducibility (a generalization of the AP-reducibility) we show how to improve the running time of approximation schemes for "planar" restriction of satisfiability problems.

We then turn to the problem of *finding* approximation-preserving reductions. Our emphasis here is on explicit non-approximability results. Our main result is a technique based on Linear Programming to find *automatically* approximation-preserving reductions by *local replacement* among a large class of problems, including Maximum Satisfiability, Maximum Cut, and problems expressing the computations of PCP verifiers. The reductions found by our method are *optimal* among local-replacement ones in terms of the way approximation is preserved. Our method works for several problems and yields the stronger known non-approximability results for Maximum Cut, Maximum Directed Cut and other related problems. It can also be used to find improved approximation algorithms.

The non-approximability results obtained in this way hold for the weighted versions of the studied problems. We show that for a large family of problems the unweighted version is precisely as hard to approximate as the weighted one. Finally, we present an improved non-approximability result for the Vertex Cover problem in bounded degree graphs.

Acknowledgments

I wish to acknowledge my great debt with my advisor Pierluigi Crescenzi, for having been advisor, coauthor, colleague and friend, for teaching me all that I know about doing research and for always pointing out interesting and stimulating problems to me. Thanks to Daniel Bovet, who revealed to me the existence of complexity theory, and who introduced Pierluigi to me.

During a short but exciting visit to IBM Research, Madhu Sudan taught me a lot about probabilistically checkable proofs, semidefinite programming, hanging out in Manhattan, and, mainly, how to get a deep insight into the problems at hand.

I am also grateful to José Rolim, who hosted me during a visit to the University of Geneva, and to José Balcázar, who hosted me during a visit to the Technical University of Catalonia.

The results surveyed in this dissertation arose from joint works with Pierluigi, Madhu, Andrea Clementi, Viggo Kann, Riccardo Silvestri, Greg Sorkin and David Williamson [50, 46, 42, 49, 132, 51]; I also borrowed a couple of theorems from [131].

I thank Giorgio Ausiello for serving with Pierluigi and Riccardo on my dissertation committee.

Mihir Bellare and Alessandro Panconesi were the external referees of a preliminary version of this thesis. I thank them for their comments.

My work benefited remarkably by helpful and clarifying discussions with several other people, including Paola Alimonti, Marco Cesati, Richard Chang, Josep Díaz, Ray Greenlaw, Sanjeev Khanna, Hoong Chuin Lau, Wolfgang Merkle, Chiara Petrioli, and Fatos Xhafa. I am especially grateful to Ray and Fatos for reading a preliminary version of this thesis and suggesting several corrections.

My fellow grad students at the Department of Computer Science of the University "La Sapienza" made me spend the last three years in a friendly and warm working environment. I have to apologize to them for my laziness.

In these years many other friends helped me develop as a person and a computer scientist, and I regret to remark how little did they succeed. I will not try to list their names, since no such list would be complete, but I gratefully thank all of them, and I beg them not to give up.

My parents are certainly the most important contributors of this thesis. I am grateful to them for their love and care.

Contents

Chapter 1

Introduction

Once more, we have decreased the number of open questions in the field – without, alas, increasing much the number of answers!
(C.H. Papadimitriou and M. Yannakakis [123])

In a *combinatorial optimization* problem, one is given an *instance* to which a *finite* set of *feasible* solutions is associated. A properly defined *measure* (or *objective*) function attributes an integer cost to any such solution, and the goal is to find a feasible solution of optimum (maximum or minimum) cost. For example, in the MIN TSP (Minimum Traveling Salesperson Problem) we are given a set of "cities" $\{c_1, \ldots, c_n\}$ and, for any pair of cities (c_i, c_j), their "distance" $d_{i,j}$. The goal is to find a minimum length tour that starts at any city, reaches all cities exactly once, and returns at the starting city.

Hundreds of natural problems from Computer Science, Operations Research and Engineering (but also Theoretical Physics, Discrete Mathematics, Molecular Biology, ...) are formulated in this way. The search space being finite suggests a simple algorithm for solving such problems: go through all feasible solutions and return the best one. However the search space is usually exponentially large in the size of the input: in the MIN TSP there are $n!$ feasible solutions for an instance with n cities. Nevertheless, for some problems, clever algorithms can be found that efficiently[1] compute an optimum solution.

Some of these algorithms were among the best early achievements of algorithmics: Kruskal's algorithm for finding a minimum cost spanning tree [108], Dijkstra's algorithm for finding shortest paths in a graph [54], Edmond and Karp's version of Ford and Fulkerson's algorithm for maximum flow [65, 55], and so on.

On the other hand, for the majority of interesting problems, including the MIN TSP, there seemed to be no way to efficiently find optimum solutions. Eventually, the theory of NP-completeness [44, 97, 111] came up giving an explanation to this phenomenon and providing a unified framework to deal with the *intractability* of optimization problems. In a short while, hundreds of NP-completeness results were discovered (see [70] for a survey of the field as it was in 1979), and it is estimated that now a few thousands NP-complete problems are known, most of them being (decision versions of) optimization problems.

Since NP-hard optimization problems are unlikely to be efficiently solvable to optimality, algorithm designers are forced to settle for more modest goals. A major flow of research deals with algorithms that, even if running asymptotically in exponential time, require a feasible amount of time for small instances. Branch and bound as well as cutting plane techniques have proved quite

[1] As usual, we say that an algorithm is *efficient* if it runs in time bounded by a polynomial in the size of the input.

successful in this regard. Unfortunately, this approach fails when it is necessary to solve very large-scale instances (e.g. the Min TSP on instances with one million cities).

Indeed, in many practical applications, fast heuristic algorithms finding "good enough" non-optimum solutions can be valuable. An heuristic is especially well-behaved if it returns near-optimal solutions *for any instance*. For example, an algorithm that solved Min TSP finding tours that are at most 1% longer than the optimum for any graph, would be as good as an optimum algorithm in most practical situations.

When there exists some constant $r > 1$ such that the cost of the solution found by the heuristic is always guaranteed to be within the multiplicative factor r from the optimum, then the heuristic is called an *r-approximate algorithm* (and r is called the *performance ratio* of the algorithm).

The design and analysis of approximation algorithms for combinatorial optimization problems is currently a large and growing field of theoretical computer science. The ultimate goal of this area is to find, for any interesting problem, the *best possible approximation algorithm*, that is, the r-approximate algorithm with the smallest value of r[2].

Such an endeavour has two facets: on the one hand algorithmic research focuses on techniques giving better and better approximation guarantees; on the other hand, complexity theory is interested in understanding which approximation factors are *intractable* to achieve and in how to interpret *non-approximability* results using notions of completeness in complexity classes.

The two areas are richly interconnected, and it happens that techniques and results developed to prove hardness results end up giving improved approximation algorithms. We list a few recent examples of such interaction (in chronological order).

1. Feige and Lovász [64] use semidefinite programming relaxations of optimization problems to investigate the power of two-provers one-round probabilistic proof systems. Probabilistic proof systems are at the core of almost all non-approximability results of the literature. Feige and Goemans [60] used the same relaxations to give the best known approximation algorithms for Max DiCut[3] and Max 2Sat.

2. The best known approximation algorithm for Max 3Sat uses a reduction to Max 2Sat, and then applies the previously mentioned Feige and Goemans' algorithm. Such reduction has been found using a general technique that has been developed with a focus on the hardness of approximation (see Section 4.3).

3. Arora [7], in a recent breakthrough result, showed that the Min TSP is r-approximable for any $r > 1$ in the important special case when the cities lie in the plane. His first approach with the problem was to try to prove hardness of approximation. He claims that a combinatorial result proved to understand the failure of standard non-approximability proof-techniques led him in a natural way to the algorithm.

4. A reduction technique introduced to study the relations between the approximation classes Max SNP and Max NP has been used to give an efficient approximation scheme for the "planar" restriction of Max Sat (see Section 3.3).

This thesis mainly focuses on the hardness of approximation, even if some results have algorithmic relevance. Before describing our results, we will try to sketch a short history of the field, in order to put such results in context.

[2] Of course, it is desirable that the approximation algorithm be as efficient as possible as well.

[3] The definitions of all the problems mentioned in the thesis can be found in Section 2.5.

1.1 History

1.1.1 1973–1986: Scattered Results

Approximation algorithms are a very natural way of coping with the hardness of optimization problems, and, even without using such terminology, they arose in the area of Operations Research. Already in 1966, a paper by Graham [77] addressed the issue of a worst-case analysis of the solutions provided by a heuristic algorithm for a scheduling problem.

The formal definition of approximation algorithm and performance ratio is due to a pioneering paper by Johnson [88]. In his paper, Johnson hinted at the relevance of non-approximability results, he presented a 2-approximate algorithm for Max Sat, a $(1 + \ln n)$-approximate algorithm for Min Set Cover, and noted the failure of most intuitive ways of approximating Max Clique and Min Node Coloring. Johnson's paper was extremely influential: even the notation introduced in it is (basically) still used. A few classes were introduced in his paper and shortly thereafter: APX (the class of problems that are r-approximable, *for some* $r > 1$), PTAS (the class of problems that are $(1+\varepsilon)$-approximable *for any* $\varepsilon > 0$), and FPTAS (the class of problems that are $(1+\varepsilon)$-approximable *for any* $\varepsilon > 0$, in time polynomial in the size of the input and in $1/\varepsilon$).

In a few years, the field developed rapidly on the algorithmic side, but only a few, definitely unsatisfying, hardness results came up.

Indeed, the unique known tool to prove intractability was NP-completeness theory, that deals with *languages* (i.e. decision problems). The complexity of *optimally* solving optimization problems can be investigated using NP-completeness theory by associating a language to any problem. Unfortunately, it turns out that the associated languages of two NP-hard problems are computationally equivalent regardless of the approximability properties of the two problems. A considerable research activity was directed towards overcoming this inconvenience by dealing with modified (restricted) versions of NP-completeness.

A successful attempt was the introduction of *strong* NP-completeness due to Garey and Johnson [69]. When a problem is strongly NP-hard (and a couple of technical conditions hold), then it is not in FPTAS, unless P = NP. Within a few year, proofs of strong NP-completeness (when possible) became standard in complexity papers.

The next goal was to find a restriction of NP-completeness that could prove non-inclusion in PTAS. One simple observation is the following. Assume that for some constant k, given an instance x of an optimization problem A, it is NP-hard to tell whether the optimum is larger than k or not. An r-approximate algorithm can be used to answer to such a question, provided $r < (k+1)/k$, and thus A cannot belong to PTAS. A few non-approximability results can be proved with this technique, but they are in general quite weak. An interesting exception is the k-center problem studied by Hochbaum and Shmoys [83]. Such problem can be 2-approximated, and the technique described above shows that performance ratios smaller than 2 are NP-hard. For several interesting problems, such as Max Clique, Min Vertex Cover, and Max 3Sat, this approach is useless, since for fixed k one can tell in polynomial time if the optimum is larger than k or not.

Sahni and Gonzalez [127] used a "gap explosion" technique to prove that Min TSP, is not in APX. This technique only works in a very few cases.

The most interesting result of this period, which is due to Garey and Johnson [68], is the fact that Min Node Coloring cannot be r-approximated for $r < 2$. Their result starts from the NP-completeness of 3-coloring (that already implies a $4/3 - \varepsilon$ hardness) and then they increase the gap using *graph product*, a technique whose power has been recently widely exploited. Graph product can also be used to establish the *self-improvability* of Max Clique. This puzzling property implies that either Max Clique is in PTAS or is not in APX. It was not clear which of the two cases holds.

1.1.2 1987–1990: Grouping Problems Together

An inspection of the early non-approximability results shows that almost all of them deal with the complexity of the decision problem associated with the optimization problem of interest. An important breakthrough in approximation theory was the realization that a different approach had to be taken: moving from the associated languages to the optimization problems themselves. Namely, *classes of optimization problems* had to be defined, and then one had to prove *completeness* or *hardness* results in such classes using *approximation-preserving* reductions.

The idea of defining notions of reducibility directly on optimization problems had already been conceived in the early 1980s in two pioneering papers. A paper by Paz and Moran [124] was perhaps the first one to introduce the idea of *approximation-preserving* reducibilities, but they did not find complete problems in approximation classes. Ausiello, D'Atri and Protasi [14] also define a notion of reducibility among optimization problems. Their reduction preserves a lot of structure of the problem (not only the approximability properties). Again, the issue of completeness was not addressed in their paper.

Krentel [107] was the first to prove completeness results in classes of optimization problems. The focus of his research was on the hardness of computing *optimum* solutions to optimization problems, and so the reduction he defined is not approximation-preserving. His results are partly surprising, since he proved that a problem in FPTAS (namely, the KNAPSACK problem) is harder than the MAX CLIQUE problem (more recently, it has been shown that MAX CLIQUE is very hard to approximate). A way of interpreting his result is that the hardness of an optimization problem is not related to the hardness of its approximate version. We will elaborate later on this point and on how it is related to the correct definition of approximation-preserving reducibility.

The seminal work of Orponen and Manilla [118] defined approximation-preserving reducibilities and contained completeness results in Min NPO[4]. For the first time, it was observed that a completeness result in an approximation class (under an approximation-preserving reducibility) implies a non-approximability result.

Crescenzi and Panconesi [47] extended this idea to other classes, and gave completeness results in Max NPO and APX. Completeness in APX is a particularly interesting notion. Indeed, an APX-complete problem cannot belong to PTAS unless P = NP, and thus the notion of APX-completeness could be a general tool similar to NP-completeness to prove non-membership in PTAS. Once the first APX-completeness result is established, one has only to use reductions to prove more completeness results. Unfortunately, the APX-complete problem described in [47] is artificial, and no natural complete problem had since been found for a long time. Interestingly, the APX-completeness of several natural problems has been recently established *building on a previously proved non-approximability result*. Thus, completeness followed from non-approximability instead of vice-versa. We will come back to this fact in the next section.

More recently, other completeness results were obtained by Berman and Schnitger [29], and Kann [93] in classes of problems with polynomially bounded measure function.

The most influential work of this period was by Papadimitriou and Yannakakis [123], where a radically new approach was taken. Instead of searching for complete problems in natural (computationally defined) approximation classes, they focused on natural problems and found a reasonable class where these problems are complete. Their starting point is Fagin's [58] logical characterization of NP. They obtained a class of APX optimization problems by introducing a notion of optimization in Fagin's characterization, and called this class Max NP. It captures several problems, including MAX SAT, but they could not find a complete problem. A restriction in the definition yields a subclass Max SNP, that still contains interesting problems such as MAX CUT and MAX 3SAT, and that turns out to have natural complete problems, including MAX CUT and MAX 3SAT themselves.

[4]See Section 2.1 for a definition of NPO, Max NPO, and Min NPO.

Other natural problems were shown to be Max SNP-hard, including MIN VERTEX COVER. This result does not answer the question of whether MAX SAT and MIN VERTEX COVER are in PTAS or not, but unifies all such questions to a single one: is Max SNP contained in PTAS? Clearly, if some Max SNP problem is not in PTAS, then no Max SNP-hard problem can be in PTAS either. Regarding the relevance of introducing new classes to capture the complexity of not-understood problems, we wish to quote two beautiful paragraphs from [123] and [121] about the nature of complexity classes and the reason why they are worth being defined.

> *In complexity, when we are faced with such a situation, in which a family of problems cannot be identified with the known complexity classes (...) one suspects that a* new complexity class *may be manifesting itself.*
>
> *[Since these new classes] contain (...) natural, important, well studied complete problems, in some sense they existed before their definition, they are* discovered, *not* invented. *They represent computational paradigms* that must be collapsed with known ones, *if progress is to be made on understanding the complexity of the problems in hand.*
>
> (C.H. Papadimitriou and M. Yannakakis)

The success of this approach was immediate and widespread. Other Max SNP-hard problems were soon discovered, including the restriction of MIN TSP to metric spaces [122], MIN STEINER TREE [30], SHORTEST COMMON SUPERSTRING [33], and MULTIWAY CUT [52].

In related research Berman and Schnitger [29] proved that if Max SNP problems do not have a randomized PTAS, then MAX CLIQUE is hard to approximate within n^c, for some $c > 0$. Their result is proved using *randomized graph product.*

The approach of syntactically defining approximation classes with natural complete problems was furtherly explored in a series of papers. Panconesi and Ranjan [119] proved that MAX CLIQUE is not in Max SNP and defined a class where MAX CLIQUE is complete. Kolaitis and Thakur performed a thorough investigation of syntactic classes [105] and extended the approach to minimization problems [106]. Further research exploited the expressive power of logical formalisms in this setting, considering also fixed points (see e.g. [21]).

1.1.3 1991–1993: The PCP Connection

The extended research activity spread after Papadimitriou and Yannakakis' work produced several results relating the hardness of problems one with the other, but not with standard complexity theoretic notions. The "collapse with known computational paradigms" was still to come, and it eventually came from one of the most unexpected directions: probabilistic proof-checking.

(Deterministic) proof checking is inherent in the standard definition of NP as the "class of languages with *short* and *easy to verify* membership proofs", where *short* stands for *polynomially sized* and *easy* stands for *doable in polynomial time.* The idea of extending this model allowing *randomness* and *interaction* was introduced independently and simultaneously by Babai [15] and by Goldwasser, Micali and Rackoff [76]. The class of languages admitting such randomized interactive proof systems was called IP. Ben-Or, Goldwasser, Kilian and Wigderson considered an extension of the model where the verifier interacts with two (or even more) provers (who do not communicate one with the other); the languages admitting such extravagant proof systems form the class MIP. Fortnow, Rompel and Sipser [66] gave an alternative characterization of MIP: a MIP language admits (exponentially long) proofs that can be checked by a polynomial time randomized verifier[5]. These new models were supposed to be only slightly more powerful than NP, until, in a major

[5]The FRS terminology was actually slightly different; they referred to *oracles* rather than to *proofs.* However there is no conceptual difference.

breakthrough, Lund, Fortnow, Karloff and Nisan [115] proved that IP contains the polynomial-time hierarchy. Shortly thereafter, Shamir [128] proved that IP is equal to PSPACE and Babai, Fortnow and Lund [17] proved that MIP equals NEXP. The latter result was particularly astonishing: it means that for any language admitting exponentially long membership proofs, such proofs can be verified in polynomial time with low error probability. The next logical step was to scale down the theorem and proving that any NP language has membership proofs checkable in polylogarithmic time. This was done by Babai, Fortnow, Levin and Szegedy [16]. Almost simultaneously, Feige, Goldwasser, Lovász, Safra and Szegedy [61] gave an improved proof checking system for NP (running in $O(\log n \log\log n)$ time) and discovered (invented?) the reduction that was to revolutionize the field of approximation. Namely, they showed how to encode the computation of a verifier into a graph, in such a way that there is a correspondence between cliques in the graph and alleged membership proofs, and between the size of the clique and the probability of the alleged proof to be accepted. Thus an approximation algorithm for Max Clique can be used to approximate the probability of acceptance of the verifier, and a gap between the probability of acceptance of correct and wrong proofs translates into a non-approximability result for Max Clique. Assuming that NP has no subexponential algorithms, it turns out that Max Clique is hard to approximate to within $2^{\log^{1-\varepsilon} n}$ for any $\varepsilon > 0$. The connection between proof checking and approximation was also pioneered in a paper by Condon [43].

After that, progress was fast. In a seminal paper, Arora and Safra [10] noted that to improve the FGLSS result it was not necessary to bound the running time of the verifier but only the number of *random bits* and the number of *query bits* that it uses. They introduced the notation $\mathsf{PCP}(r(n), q(n))$ that denotes the class of languages having membership proofs that can be probabilistically checked using $O(r(n))$ random bits and $O(q(n))$ query bits. Finally, they reduced the number of query bits by introducing and using the new idea of *recursive composition* of verifiers: a verifier checks a proof that a verifier accepts when checking the proof that a verifier accepts when ... Their main result was $\mathsf{NP} = \mathsf{PCP}(\log n, \sqrt{\log n})$, implying the NP-hardness of approximating Max Clique within $2^{\sqrt{\log n}}$.

The next step was the realization that the hardness of other approximation problems could be established using the PCP connection. Sudan and Szegedy observed that if $\mathsf{NP} = \mathsf{PCP}(\log n, 1)$, then $\mathsf{Max\ SNP}$ is not contained in PTAS (unless $\mathsf{P} = \mathsf{NP}$). In a short while, the $\mathsf{NP} = \mathsf{PCP}(\log n, 1)$ result was established by Arora, Lund, Motwani, Sudan and Szegedy [9]: this result is now known as the *PCP Theorem*. Besides the application to $\mathsf{Max\ SNP}$-hard problems, the PCP Theorem also implies that it is NP-hard to approximate Max Clique within n^c, for some $c > 0$.

Other optimization problems were then shown to be related to proof-checking, and a whole bunch of long-standing open questions were rapidly solved one after the other. In particular, Lund and Yannakakis [116] proved a $\frac{1}{4}\log n)$ lower bound for the approximability of Min Set Cover (almost matching Johnson's approximation algorithm) and an n^c lower bound for Min Node Coloring, for some $c > 0$. Their reductions are not from the verifier of the PCP Theorem, but from the slightly different model of two-provers one-round proof systems [109, 64] (Bellare [23] was perhaps the first to explore the connection between such proof systems and approximation problems).

1.1.4 Contemporary Results

The development of new results in this field unfolded at an amazing pace in the last three years. Even very recent surveys are slightly outdated [24, 94].

Improved non-approximability results are perhaps the most striking recent achievements. They have been obtained using more and more refined versions of the PCP Theorem, as well as finding better reductions. Recently, *tight* non-approximability results have been established for a series of important optimization problems.

On the other hand, the amazing power of PCP techniques has given new momentum to structural approximation theory.

Tight Non-approximability Results

All of the non-approximability results obtained from the first version of the PCP Theorem [9] have an existential flavour: for example there are positive constants c_1 and c_2 such that Max Sat cannot be approximated within $1 + c_1$ and Max Clique cannot be approximated within n^{c_2}, but there were no explicit estimations of the constants.

An optimized proof of the PCP Theorem was proved by Bellare, Goldwasser, Lund and Russel [26]. Their result implies that Max Clique is hard to approximate within $n^{1/30-\varepsilon}$ for any $\varepsilon > 0$ (from now on, ε will always denote some arbitrarily small positive constant). Progress on the constant was fast. The non-approximability result for Max Clique was obtained by developing a PCP verifier for NP that only queries 29 bits, and using a result of Zuckerman [136] (based on the randomized graph product technique of [29]) stating that if NP languages can be checked by querying q bits, then Max Clique is hard to approximate within $n^{1/(q+1)-\varepsilon}$. Subsequently, Feige and Kilian [62] observed that a more refined parameter than query complexity could be used. If a verifier uses f *free bits* then Max Clique is hard to approximate within $n^{1/(f+1)-\varepsilon}$. Roughly speaking, a verifier uses f free bits if, once it tosses its random coins and queries bits from the proof, there at most 2^f outcomes of its queries that make it accept: thus free bits are always no more than the query bits (after querying q bits, there cannot be more than 2^q possible answers) and are usually much less. This new idea implied an $n^{1/15-\varepsilon}$ hardness result for Max Clique. Yet another refinement was proposed by Bellare and Sudan [27], who introduced the notion of *amortized free bits*: a verifier uses $\bar{f}$ amortized free bits if it uses f free bits, accepts with probability one correct proofs, accepts with probability at most s wrong proofs, and $\bar{f} = f/\log(1/s)$. This notion of "amortization" tries to capture the fact that one is willing to use more free bits provided that this yields a considerable reduction of the error probability s. Indeed, if NP languages can be checked with $\bar{f}$ amortized free bits, then Max Clique is hard to approximate within $n^{1/(\bar{f}+1)-\varepsilon}$. Bellare and Sudan showed that $3 + \varepsilon$ amortized free bits are sufficient, and thus Max Clique is hard to approximate within $n^{1/4-\varepsilon}$. Bellare, Goldreich and Sudan [25] gave a system using $2+\varepsilon$ amortized free bits, then Håstad showed that $1 + \varepsilon$ are sufficient [79], and, eventually, that ε amortized free bits are sufficient [78]. Thus, for any $\varepsilon > 0$, it is hard to approximate Max Clique to within $n^{1-\varepsilon}$.

Similar continuous improvements were developed for Min Node Coloring. To make a long story short, the final (tight) result by Feige and Kilian [63] is that, for any $\varepsilon > 0$, it is hard to approximate Min Node Coloring to within $n^{1-\varepsilon}$. Their result heavily builds upon the ε amortized free bits verifier by Håstad.

The tight result for Min Set Cover is by Feige [59], who proved that it is hard to approximate this problem to within $(1 - \varepsilon) \ln n$, matching the positive result by Johnson [88].

The first explicit non-approximability result for Max Sat was established by Bellare, Goldwasser, Lund and Russel [26]. They gave a proof of the PCP Theorem with an explicit description of the computations performed by the verifier and an explicit bound on the probability of accepting wrong proofs (also known as the *soundness* probability); correct proofs are accepted with probability one. Once the computations of the verifier are encoded using small CNF formulae (as originally done in [9]), the non-approximability result implied for Max Sat can be computed as a function of the size of the CNF formulae and of the soundness probability. The hardness proved in [26] was 94/93. Further improvements were obtained using the same approach, but starting with a better version of the PCP Theorem (i.e. with simpler computation and/or with better soundness). Bellare and Sudan [27] proved a 73/72 hard factor, while Bellare, Goldreich and Sudan [25] arrived at 27/26: the current best result (perhaps tight) is $8/7 - \varepsilon$ by Håstad [80].

The Structural Revival

A new spate of results of structural flavour has recently come up stimulated by the PCP Theorem and its powerful corollaries.

Khanna, Motwani, Sudan, and Vazirani [102] use the PCP Theorem to prove that MAX 3SAT is complete for APX PB (the restriction of APX to problems with polynomially bounded objective function). More generally, they give a new reduction technique such that a completeness result can be obtained from a hardness of approximation result provided the problem has a property that they call "additivity". Their result closed a series of relevant open questions in structural approximation theory: including whether Max NP has complete problems, and whether MAX SAT has an approximation-preserving reduction to MAX 3SAT. The KMSV reduction technique transforms several *hardness* results into *completeness* results and definitely *collapses* the approach based on logical definability with standard complexity classes. Unfortunately, it only works inside polynomially bounded approximation classes.

In a different line of research, Chang, Gasarch and Lund [41, 40] use the PCP Theorem to investigate the *query complexity* of approximation problems. For example, Håstad's result implies that it is NP-hard to approximate MAX SAT within a 1.142 factor. It means that, by running once a 1.142-approximate algorithm for MAX SAT it is possible to answer to one NP-question. What about answering two or more questions? Chang, Gasarch and Lund explore this idea and give upper and lower bounds to the number of NP queries that can be encoded into one approximation problem. A consequence of their result is that it is strictly easier to compute a 2-approximation of the clique number of a graph than to compute the exact clique number. Even if the two problems are both NP-hard, the latter cannot be many-one reduced to the former, unless P = NP.

1.1.5 Further Reading

Essentially all the earliest non-approximability results (including the strong NP-completeness theory, the $2 - \varepsilon$ hardness result for MIN NODE COLORING, and the hardness of MIN TSP) are surveyed in the book of Garey and Johnson [70]. Good accounts on the first attempts at developing approximation-preserving reductions and completeness in approximation classes are a survey by Bruschi, Joseph and Young [37], Kann's PhD thesis [92], and a more recent survey by Ausiello, Crescenzi and Protasi [13]. Max SNP-completeness theory is treated very broadly in Papadimitriou's book on computational complexity [120]. In particular, several Max SNP-completeness proofs that are only sketched in [123] are fully spelled out (and sometimes simplified) in [120]. Interactive proof systems and probabilistically checkable proofs are a fascinating field, and a very complicated one: thus it is an ideal terrain for surveys. An enjoyable account on the meaning of the PCP Theorem and on its applications to approximation theory is given by Johnson [89]. The enterprising reader may want to have a look at the actual *proof* of the PCP Theorem: a full proof is contained in Sudan's thesis [130] and in Arora's one [5]. Arora and Lund [8] give a very clear account (with proofs) on how to use the PCP Theorem to prove non-approximability results. The latest non-approximability results are mentioned (with the exception of very recent ones) in a survey by Bellare [24]. The historical sections in the monograph by Bellare, Goldreich and Sudan [25] contain detailed and very complete accounts on the sequence of improvements that culminated in current non-approximability results.

1.2 The Results of This Thesis

We now give an overview of the original results presented in this thesis. We are concerned with two main questions about approximation-preserving reductions: what is the best way of *defin-*

ing the notion of approximation-preserving reducibility, and how to *find* approximation-preserving reductions that be the *tighter possible* between specific problems.

We prove several negative results about known approximation-preserving reducibilities, and we propose a new definition that overcomes these limitations and allows to prove all the completeness results that one would expect to hold (but that do not hold with respect to, say, the E-reducibility or the L-reducibility.) The main novelty of our new reducibility, that we call AP-reducibility, is the fact that it maps approximate solutions into approximate solutions but not optimum solutions into optimum solutions. We are able to prove that this feature is *necessary* in order to prove, for example, the APX-completeness of Max Sat.

We then consider approximation-preserving reductions between specific problems and reductions from PCP verifiers to these problems. Our goal is to show improved non-approximability results. Our main result is a technique that allows to use Linear Programming to find *automatically* reductions by *local replacement* between problems. Such computer-constructed reductions are, in a sense that can be made formal, the strongest possible. These reductions yield improved non-approximability results for the weighted versions of Max Cut and Max DiCut. We then prove that any non-approximability result that holds for the weighted version of Max Cut and Max DiCut (as well as for several other problems) also holds for the unweighted version. Finally, we show how to *sparsify* the graphs arising in the FGLSS reduction, and we use this result to prove that the hardness of approximation of Min Vertex Cover in graphs of bounded (but large) degree is basically the same as the one of the unrestricted Min Vertex Cover problem.

1.2.1 Structure in Approximation Classes

Reductions and Completeness

Research in structural approximation theory and in non-approximability produced several *hardness* results in the last few years. A lot of (sometimes artificial) classes were necessary to see them as *completeness* results. The work of Khanna, Motwani, Sudan and Vazirani [102] unified most of these classes; however we already mentioned the fact that their work does not extend beyond polynomially bounded classes. Additionally, in some classes one was forced to distinguish between maximization and minimization problems: there were Max NPO-complete and Min NPO-complete problems but not any NPO-complete problems.

Another source of complication was the amazing number of different approximation-preserving reducibilities that appeared in the literature. Some of them do not seem to be comparable, so a completeness result proved with respect to one reducibility may not hold with respect to a different one.

We develop a *systematic study* of the properties of known approximation-preserving reducibilities. We prove that the lack of unified completeness results in a few natural classes is due to *inherent limitations* of known reducibilities. In particular, we prove that

1. The L-reduction used by Papadimitriou and Yannakakis [123] does not preserve constant-factor approximability. More dramatically, the closure of Max SNP under L-reductions contains non-approximable problems. Both results are proved assuming $\mathsf{P} \neq \mathsf{NP} \cap co\text{-}\mathsf{NP}$.

2. Max Sat is not APX-complete under E-reductions (used by Khanna et al. [102]) nor under P-reductions (used by Crescenzi and Panconesi [47]), unless $\mathsf{P}^{\mathsf{NP}} = \mathsf{P}^{\mathsf{NP}[O(\log n)]}$ (a fairly unlikely event).

3. Max Sat is not even APX PB-complete under L-reductions (unless $\mathsf{P} = \mathsf{NP}$).

The first result contradicts a wide-spread belief. The way we prove point two indicates what the limitations of current reducibilities are: not only do they preserve approximability, but they also preserve the hardness of computing optimum solutions. This is, perhaps, an inheritage of the old approaches based on NP-completeness. As proved by Krentel [107], problems that are easier to approximate can be more difficult to solve optimally.

TECHNICAL CONTRIBUTIONS. *We use advanced tools from* structural complexity theory*, not used before in the analysis of approximation-preserving reductions.*

To overcome such limitations, we propose a definition of a new (kind of) reducibility that preserves approximation *but not optimal solvability*, and we call it the *AP-reducibility*. Under AP reductions we prove

1. The existence of natural NPO-complete problems, unifying the previous classes of Max NPO- and Min NPO-complete problems.
2. Similarly, the existence of natural NPO PB-complete problems.
3. The existence of natural, APX-, logAPX- and polyAPX-complete problems, overcoming the previous restriction to polynomially bounded classes.

TECHNICAL CONTRIBUTIONS. *The main novelty in the definition of the AP-reducibility is that it* knows *the approximation factor that it is expected to preserve: in a sense AP-reducibility is a* non-oblivious *reducibility.*

The APX- and polyAPX-completeness of natural problems is proved with thechniques that are based on the PCP Theorem. We introduce a reduction technique based on the probabilistic method that allows us to prove directly that MAX 3SAT is Max NP-complete.

The novelty of our approach is better explained in the special case of the reduction from MAX SAT to MAX 3SAT. Standard reduction techniques based on local replacement fail to reduce MAX SAT to MAX 3SAT due to the possible presence of large clauses, but large clauses are easy to satisfy using a simple randomized algorithm (that, in turn, performs poorly on small clauses). We then *combine* (in a probabilistic sense) a solution based on standard reductions and the one given by the randomized algorithm, and this mixed solution will be good for any combination of small and large clauses.

TECHNICAL CONTRIBUTIONS. *The idea of probabilistically combining different solutions has been used in the design of* approximation algorithms *(e.g. in [74]) but we use it for the first time to develop an* approximation-preserving reduction*.*

The Max NP-completeness of MAX 3SAT is an easy corollary of its APX-completeness, but our reduction has both a *structural* and an *algorithmic* relevance due to the fact that it does not use PCP machinery.

From a *structural* point of view, the Max NP-completeness of MAX 3SAT means that Max NP problems are no harder to approximate than Max SNP ones. The definition of Max NP (see Section 2.1 and [123]) involves a richer logical structure than Max SNP, and the PCP reduction does not clearly explain how come it is possible to get rid of this additional structure. This is due to the fact that PCP reductions are *global* in a very strong sense, and while reducing a logical problem to another, it is not clear which variables are mapped to which, which constraint is mapped to which, and so there is no clear relation between the structure of the source problem and the structure of the target problem. In contrast, our reduction gives a very simple explanation of the relation between Max NP and Max SNP: roughly speaking, the only difference between the two classes is that, in the logical definition of Max NP, problems with arbitrarily long disjunctions are admitted. But long disjunctions are easy to satisfy (in a probabilistic sense), and so they cannot make a problem harder.

An approximation-preserving reduction from a problem A to a problem B can yield an *approximation scheme* in the following case. Assume that problem B admits an approximation scheme when restricted to some subset $\mathcal{I}_B$ of its instances, and that it is possible to find a set $\mathcal{I}_A$ of instances of A such that any such instance is mapped by the reduction into an instance of $\mathcal{I}_B$. Then, the reduction together with the approximation scheme gives an approximation scheme for the instances of $\mathcal{I}_A$. Our reduction indeed gives an approximation scheme for *planar* restrictions of MAX SAT and MAX GSAT-B that works in linear time (as opposed to a previous scheme running in time $n^{O(1/\varepsilon)}$). Such a result would have been impossible using PCP reductions for at least two reasons:

1. It is very difficult to characterize the outcome of a PCP reduction, and so to understand which set of instances will be mapped into which one.

2. PCP reductions always generate instances that are hard to approximate, regardless of the simple structure of the source instance.

The above considerations suggest that a PCP-free proof that MAX 3SAT is APX-complete would have very interesting structural and algorithmic consequences. We show that, unfortunately, at least a weaker version of the PCP Theorem is necessary in order to prove the APX-completeness of MAX 3SAT.

Query Complexity and Its Applications

We then turn to the analysis of properties of complete problems in approximation classes. We concentrate on query complexity, which has already been used to measure the hardness of approximation problems, but we adapt it for the first time to the hardness of *computing approximate solutions* (as opposed to previous studies dedicated to the hardness of *approximately evaluating the optimum*).

We give a characterization of APX-completeness and polyAPX-completeness in terms of query complexity and combinatorial properties. This shows that query complexity is not an *effect* of the hardness of approximation, but is basically the *same phenomenon*, observed from a different perspective. This is another "collapse of computing paradigms". It also explains an already mentioned paradoxical phenomenon: in complexity theory the hardness of a problem (modulo standard conjectures such as $\mathsf{P} \neq \mathsf{NP}$) is usually established as a corollary of a completeness result in a complexity class. In APX, things went the other way around: completeness of natural problems, such as MAX SAT, has been established relying on the PCP Theorem, that is, relying on the hardness of approximation. Our characterization of APX-completeness tells us that for APX problems enjoying an "additivity" property (including MAX SAT) hardness of approximation is equivalent to APX-completeness.

Our query complexity results yield a series of consequences about the approximability properties of natural problems (under complexity-theoretic assumptions).

1. The MIN BIN PACKING problem, the MIN EDGE COLORING problem, and other problems are APX-intermediate, i.e. they are not in PTAS (unless $\mathsf{P} = \mathsf{NP}$, this was well known) nor APX-complete (unless the polynomial hierarchy collapses, this is our result).

2. The MIN SET COVER, the MIN DOMINATING SET, and the MIN HITTING SET problems are not self-improvable unless the polynomial hierarchy collapses.

3. The problem of computing an optimum solution for the MAX CLIQUE problem cannot be (many-one) reduced to the problem of 2-approximating MAX CLIQUE (unless the polynomial hierarchy collapses).

We would like to make a few remarks about our results. Regarding the first one, this is the first "intermediateness" result proved for natural problems in standard complexity classes. If $\mathsf{NP} = co\text{-}\mathsf{NP}$ then the problems listed in (1) above are APX-complete, so our result cannot be proved under the $\mathsf{P} \neq \mathsf{NP}$ hypothesis. There is still some gap between the necessary and the sufficient condition. Under our (very weak) definition of self-improvability, MIN SET COVER, MIN DOMINATING SET and MIN HITTING SET are self-improvable if $\mathsf{NP} = co\text{-}\mathsf{NP}$. Again, there is room for improvement between the necessary and the sufficient condition. We don't known how to prove our results (2) without using query complexity, and it is strange that such an esoteric complexity-theoretic concept has something to do with a basic algorithmic property like self-improvability. The third result solves an open question raised by Chang [40].

TECHNICAL CONTRIBUTIONS. *We extend the query complexity approach to deal with the problem of* computing *an approximate solution for an optimization problem. Previous results [41, 40] dealt with the problem of approximately* evaluating *the optimum. We do that by relating to* language classes*.*

1.2.2 Improved Non-approximability Results

The now standard way of proving non-approximability results is to start from (an optimized version of) the PCP Theorem, then show how to encode the computation of the verifier into an instance of the problem at hand, in such a way that the optimum of such an instance reflects the probability of acceptance of the verifier. The FGLSS reduction [61] and the reduction in [9] are paradigmatic in this respect.

Finding the Best Possible Gadgets

Constraint satisfaction problems are generalizations of the MAX SAT problem. The instance is a set of "clauses" over boolean variables: such clauses are not necessarily disjunctions of literals, but can be arbitrary boolean functions (different problems are defined using different sets of allowable boolean functions). The goal is to maximize the number of satisfied clauses or to minimize the number of contradicted clauses. Besides MAX SAT and its restrictions, several natural problems fit this framework, including MAX CUT and MAX DICUT.

Hardness results for maximization constraint satisfaction problems use a reduction by *local replacement* from the verifier of the PCP Theorem. This is a standard approach for MAX SAT (essentially dating back to [9]) but has been introduced only recently for other problems, by Bellare, Goldreich and Sudan [25]. To capture different reductions in a unified framework they formalize the notion of *gadget* that is used in local replacement reductions. Informally speaking, a gadget is a combinatorial structure that converts a constraint of an optimization problem into constraints of another problem.

Even if the notion of a gadget has been formalized only recently, it arose in NP-completeness proofs already in the 70s. An example of a gadget matching the formal definition of [25] appears in the 1976 proof that MAX 2SAT is NP-complete [71].

Bellare, Goldreich and Sudan also associate a *cost* to gadgets: the less costly is the gadget, the better is the reduction (and the implied non-approximability result). It follows that an improved construction of a gadget yields an improved non-approximability result. For this reason, it is desirable to have *lower bounds* on the cost of certain gadgets, to understand if it is possible to improve known construction or if they are the best possible. The issue of lower bounds for gadgets' costs was raised in [25].

Additionally, a great deal of trickery is required in the construction (discovery?) of these combinatorial objects, and the work has to be done, essentially from scratch, every time a new PCP

Problem	Hard factor	Best approximation
Max 2Sat	1.047 [25, 80]	1.074 [60]
Max 3Sat	1.142 [80]	**1.249** (1.299 [74, 75, 60])
Max Sat	1.142 [80]	1.319 [74, 75]
Max Cut	**1.062** (1.055 [25, 80])	1.139 [75]
Max DiCut	**1.083** (1.055 [25, 80])	1.169 [60]
Max 2CSP	**1.111** (1.047 [25, 80])	1.169 [60]
Max 3CSP	$2-\varepsilon$ [80]	**2.721** (4 [131], 7 [3])
Min Vertex Cover	1.166 [80]	2 [73]

Table 1.1: Positive and negative approximability results for some constraint satisfaction problems. We report in bold our results, and in brackets the numbers that could be achievable without using the results of this thesis.

verifier is developed that uses a different kind of test. For example, the verifier described in the first and the second version of [25] is different from the verifier described in the third and the fourth version. New gadgets had to be developed to prove hardness results using the new verifier.

We completely resolve such problems by giving a linear-programming (LP) based technique for *automatically* finding *provably optimum* gadgets. Thus, our method solves, at once, the two mayor inconveniences of the "gadget method": the fact that it is difficult to find such objects at all (no matter how good they are) and the additional difficulty of finding them as good as possible. To find the best possible gadget implementing a certain reduction we just feed some data into a program and we wait for the answer.

Our technique proves that the known gadgets for Max 2Sat and Max 3Sat (introduced in [25]) are the best possible. The gadget for Max Cut is not optimum, and we find a computer-constructed optimum one, thus giving an improved non-approximability result for Max Cut. We also present optimum gadgets (and improved hardness) for Max DiCut and Max 2CSP. Our results were developed when the best PCP verifier on the market was the BGS one [25]. Using their verifier and our reduction, we proved that Max Cut (respectively, Max DiCut) was hard to approximate within 61/60 (respectively, 45/44): for both problems, the hardness established in [25] was 72/71. Using Håstad's verifier [80], we prove hardness within $17/16-\varepsilon$ (respectively, $13/12-\varepsilon$). Using Håstad verifier in combination with the BGS gadgets would have yielded hardness within $19/18-\varepsilon$ for both problems.

Obtaining better reductions between problems can also yield improved approximation algorithms for some problems. We illustrate the point by constructing a gadget reducing Max 3Sat to Max 2Sat. It is remarkable that the gadget that we obtain is *twice better* than the gadget of the Max 2Sat NP-completeness result of [71]. Using this new reduction in combination with a technique of Goemans and Williamson [75] and the 1.074-approximation algorithm for Max 2Sat due to Feige and Goemans [60], we obtain a 1.249-approximation algorithm for Max 3Sat. The best result that could be obtained previously, by combining the techniques of [74, 75] and the bound of [60], was 1.299. (see also the 1.301-approximation algorithm in the paper of Ono, Hirata and Asano [12]). Similarly, we construct a gadget reducing Max 3CSP to Max 2Sat and, using Feige and Goemans' algorithm, we obtain a 2.721-approximate algorithm for Max 3CSP. The best previous result for this problem was 4 [131].

Our results, and some related ones, are summarized in Table 1.1.

Finally, the LP formulation allows us to prove lower bounds by hand using dual analysis. We demonstrate this technique in a couple of cases where an automatic analysis is not possible (since we give lower bounds for *infinitely many problems*). Our lower bounds give new insight into the free bit parameter and into the nature of local reductions between Max Sat and Max 3Sat.

Technical Contributions. *The key step in our constructions is a combinatorial lemma stating that optimum gadgets exist having certain nice properties. The proof exploits the idea of using* substitutions *of variables to transform any gadget into a nice one, without increasing its cost. The next observation is that the search space formed by the "nice" gadgets can be described using linear inequalities and the cost of the gadget is a linear function. Thus, finding the best gadget turns out to be an LP problem, that can be automatically solved to optimality. The* duality *theory of linear programming can be used to prove the optimality of a gadget obtained in this way* without relying on the correctness *of the LP package used to do the actual computation.*

Removing Weights

The non-approximability results proved via gadget reductions hold for (polynomially bounded) *weighted* problems. Alternatively, one can say that repetition of clauses or multiple edges are admitted. The best algorithm for such problems work equally well in the non-weighted and the weighted version (even if the weights are not polynomially bounded).

For the Max Cut problem in simple graphs, Bellare, Goldreich and Sudan [25] prove a hardness result using a reduction form the weighted version of the problem. Their "weight removing" reduction makes the hardness result worse: applying it to our hardness result for Max Cut would yield, roughly, a 1.02 hardness for simple Max Cut. Explicit non-approximability results for Max Sat without repetition of clauses were not known.

A reasonable conjecture that can be drawn from this situation is that non-weighted versions of Max Cut, Max Sat etc. are easier to approximate than the respective weighted version. The fact that known algorithms perform equally well in the two cases can be explained by observing that all of them use linear or semidefinite programming. Maybe new algorithmic techniques can provide improved approximation in the non-weighted case.

We prove that for several constraint satisfaction problems (including Max Cut, Max DiCut, Max 2Sat, Max E3Sat, etc.) the weighted version is exactly as hard to approximate as the non-weighted one. More formally, an r-approximate algorithm for the non-weighted version yields a $(r + o(1))$-approximate algorithm for the weighted version.

On the one hand, our results give an explanation of the behaviour of known algorithms, explaining that having an equally good approximation in the weighted and non-weighted case is intrinsic to the problems and not due to a particular algorithmic technique. On the other hand, we give the best known non-approximability results for the non-weighted version of such problems. Any future non-approximability result for the weighted version will immediately apply to the non-weighted version as well.

Technical Contributions. *For the* Max Cut *problem, our reduction works by substituting a set of nodes in place of a node and the edge set of a bipartite expander in place of a weighted edge. The transformation of a cut of the non-weighted graph into a cut of the weighted graph uses the* probabilistic method*; the proof that the two solutions have roughly the same performance ratio exploits the* mixing *property of bipartite expanders. The generalization to other constraint satisfaction problems requires the introduction (and the explicit construction) of* mixing sets of tuples.

The Vertex Cover Problem

The first explicit non-approximability result for Min Vertex Cover is due to Bellare, Goldreich and Sudan [25]: they use the FGLSS reduction combined with the standard reduction from

MAX CLIQUE to MIN VERTEX COVER. They proved a 17/16 hardness result improved to $7/6 - \varepsilon$ by Håstad [80].

We show that MIN VERTEX COVER can be seen as a minimization constraint satisfaction problem: it is equivalent to a problem known as MIN SAT (this result has been independently discovered in [117] and in [82]). This equivalence provides an improved approximation algorithm and a new hardness result for MIN SAT.

As for the previously mentioned maximization problems, we are interested in comparing weighted and non-weighted versions of MIN VERTEX COVER and MIN SAT. In this case a simpler proof will suffice to show that they are equally hard to approximate.

An important restricted version on MIN VERTEX COVER is that to bounded-degree graphs. This problem is also APX-complete even if the degree is bounded by three. The bounded degree restriction of MIN VERTEX COVER is also used for proving APX-completeness of other problems, including shortest common superstring and MIN STEINER TREE [33, 30]. The APX-completeness proof for MIN VERTEX COVER-B uses a reduction from MAX 3SAT. The hardness factor becomes much worse in the reduction (roughly 200 times).

We present an alternative reduction that produces graphs of large (although bounded by some constant) degree but effects the hardness factor in only a negligible way. In particular, for some constant B, it is NP-hard to approximate MIN VERTEX COVER-B within 1.16. Note that this hardness result is stronger than what is known for MAX 3SAT.

TECHNICAL CONTRIBUTIONS. *We give a new characterization of the graphs arising from the FGLSS reduction. It was commonly understood that such gra[hs are not sparse since they contain large cliques. We prove the key point they actually have large* bipartite complete subgraphs*, and, indeed, they can be characterized as the* union *of bipartite complete graphs. Our reduction works by substituting* bipartite bounded-degree expanders *in place of the bipartite complete graphs.*

Chapter 2

Background

> **Symbol,** *n. Something that is supposed to typify or stand for something else. (...) We cannot stop making them, but we can give them a name to conceal our helplessness.*
>
> (A. Bierce [31])

We assume the reader to be familiar with the basic concepts of computational complexity theory. For the definitions of most of the complexity classes used in this thesis we refer the reader to one of the books on the subject (see, for example, [70, 20, 36, 120]). In this chapter we will review the definitions of a few approximation-preserving reducibilities, some facts about PCP's and query complexity, and we will define the problems that are mentioned throughout the thesis. But, to begin with, we start with the definitions of optimization problem and approximation algorithm.

2.1 Optimization and Approximation

In this section, we provide some standard definitions in the field of optimization and approximation theory. For a string x, we denote by $|x|$ the length of its binary encoding[1]. For an integer n we denote by $[n]$ the set $\{1, \ldots, n\}$.

Definition 2.1 (NPO problems) *An* NP optimization problem A *is a fourtuple* $(I, \mathsf{sol}, \mathsf{m}, \mathrm{type})$ *such that*

1. *I is the set of* instances *of A and I is recognizable in polynomial time.*
2. *Given an instance x of I, $\mathsf{sol}(x)$ denotes the set of* feasible solutions *of x. These solutions are short, that is, a polynomial p exists such that, for any $y \in \mathsf{sol}(x)$, $|y| \leq p(|x|)$. Moreover, for any x and for any y with $|y| \leq p(|x|)$, it is decidable in polynomial time whether $y \in \mathsf{sol}(x)$.*
3. *Given an instance x and a feasible solution y of x, $\mathsf{m}(x, y)$ denotes the positive integer* measure *of y (often also called the value of y). The function m is computable in polynomial time and is also called the* objective *function.*
4. type $\in \{\max, \min\}$.

[1] Sometimes, for an instance x of an optimization problem, $|x|$ will not be defined as the length of its binary encoding, but as another relevant parameter. For example, if x is a graph, then $|x|$ will usually be the number of nodes. These special cases will be specified each time.

The goal of an NP optimization problem with respect to an instance x is to find an *optimum solution*, that is, a feasible solution y such that $\mathsf{m}(x,y) = \text{type}\{\mathsf{m}(x,y') : y' \in \mathsf{sol}(x)\}$. In the following opt will denote the function mapping an instance x to the measure of an optimum solution. For an instance x, we also denote by $\mathsf{sol}^*(x)$ the set of optimum solutions for x,

The *class* NPO is the set of all NP optimization problems. Max NPO is the set of maximization NPO problems and Min NPO is the set of minimization NPO problems.

An NPO problem is said to be *polynomially bounded* if a polynomial q exists such that, for any instance x and for any solution y of x, $\mathsf{m}(x,y) \leq q(|x|)$. The *class* NPO PB is the set of all polynomially bounded NPO problems. Max PB is the set of all maximization problems in NPO PB and Min PB is the set of all minimization problems in NPO PB.

Definition 2.2 (Performance ratio) *Let A be an* NPO *problem. Given an instance x and a feasible solution y of x, we define the* performance ratio of y with respect to x *as*

$$R(x,y) = \max\left\{\frac{\mathsf{m}(x,y)}{\mathsf{opt}(x)}, \frac{\mathsf{opt}(x)}{\mathsf{m}(x,y)}\right\}$$

and the relative error of y with respect to x *as*

$$E(x,y) = \frac{|\mathsf{opt}(x) - \mathsf{m}(x,y)|}{\mathsf{opt}(x)}.$$

The performance ratio (respectively, relative error) is always a number greater than or equal to 1 (respectively, 0) and is as close to 1 (respectively, 0) as the value of the feasible solution is close to the optimum value. It is easy to see that, for any instance x and for any feasible solution y of x,

$$\frac{R(x,y)-1}{R(x,y)} \leq E(x,y) \leq R(x,y) - 1.$$

Definition 2.3 (Approximation algorithm) *Let A be an* NPO *problem and let T be an algorithm that, for any instance x of A such that $\mathsf{sol}(x) \neq \emptyset$, returns a feasible solution $T(x)$ in polynomial time. Given an arbitrary function $r : \mathcal{Z}^+ \to [1,\infty)$, we say that T is an* $r(n)$-approximate algorithm for A *if the performance ratio of the feasible solution $T(x)$ with respect to x verifies the following inequality:*

$$R(x, T(x)) \leq r(|x|).$$

Definition 2.4 (F-APX) *Given a class of functions F, an* NPO *problem A belongs to the* class F-APX *if an $r(n)$-approximate algorithm T for A exists, for some function $r \in F$.*

In particular, APX, logAPX, polyAPX, and expAPX will denote the classes F-APX with F equal to the set $O(1)$, to the set $O(\log n)$, to the set $O(n^{O(1)})$, and to the set $O(2^{n^{O(1)}})$, respectively. One could object that there is no difference between NPO and expAPX since the polynomial bound on the computation time of the objective function implies that any NPO problem is $h2^{n^k}$-approximable for some h and k. This is not true, since NPO problems exist for which it is even hard to find a feasible solution. We will see examples of such problems in Section 3.2.1 (e.g. MAX ONES SAT). For a class $\mathcal{C} \in \{\text{APX}, \text{logAPX}, \text{polyAPX}\}$, we denote by $\mathcal{C}$ PB the class $\mathcal{C} \cap$ NPO PB.

Definition 2.5 (APTAS) *An* NPO *problem A belongs to the* class APTAS *if an algorithm T and a constant k exist such that, for any $x \in I_A$ and for any $r > 1$, $R(x, T(x,r)) \leq r + k/\mathsf{opt}(x)$, and the time complexity of $T(x,r)$ is polynomial in $|x|$.*

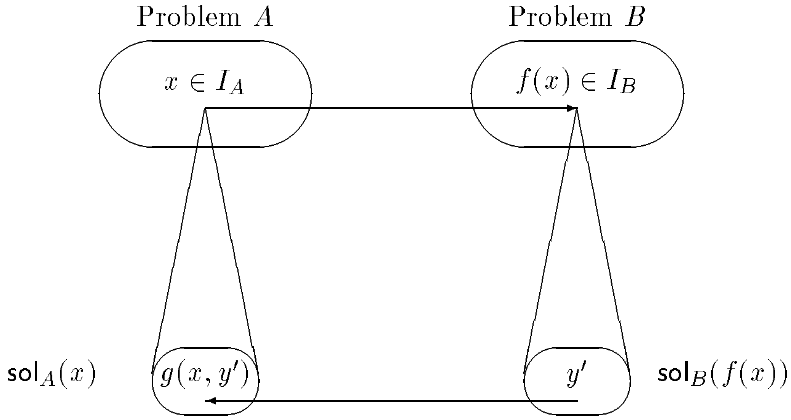


Figure 2.1: Reduction between two optimization problems

Definition 2.6 (PTAS) *An* NPO *problem A belongs to the* class PTAS *if an algorithm T exists such that, for any fixed rational $r > 1$, $T(\cdot, r)$ is an r-approximate algorithm for A.*

Definition 2.7 (FPTAS) *An* NPO *problem A belongs to the* class FPTAS *if an algorithm T exists such that, for any fixed $r > 1$, $T(\cdot, r)$ is an r-approximate algorithm for A whose running time is polynomial in $|x|$ and in $1/(r-1)$.*

Clearly, the following inclusions hold:

$$\mathsf{FPTAS} \subseteq \mathsf{PTAS} \subseteq \mathsf{APTAS} \subseteq \mathsf{APX} \subseteq \log\text{-}\mathsf{APX} \subseteq \text{poly-}\mathsf{APX} \subseteq \exp\text{-}\mathsf{APX} \subseteq \mathsf{NPO}.$$

It is also easy to see that these inclusions are strict if and only if $\mathsf{P} \neq \mathsf{NP}$.

We refer to [123] (see also [105]) for a formal definition of Max NP and Max SNP. We here give informal but equivalent definitions. We say that A is a *subproblem* of B if $I_A \subseteq I_B$, and for any $x \in I_A$ it holds $\mathsf{sol}_A(x) = \mathsf{sol}_B(x)$ and $\mathsf{m}_A(x, \cdot) = \mathsf{m}_B(x, \cdot)$.

Definition 2.8 (Max SNP) *An* NPO *problem is in the class* Max SNP *if a constant k exists such that A can be expressed as a subproblem of* Max k-CSP*.*

Definition 2.9 (Max NP) *An* NPO *problem is in the class* Max NP *if a constant B exists such that A can be expressed as a subproblem of* Max GSAT-B*.*

Theorem 2.10 (**[123]**) Max SNP $\subset$ Max NP $\subset$ APX PB.

2.2 Approximation-preserving Reducibilities

In general, in complexity theory, a *reduction* from a problem A to a problem B specifies some procedure to solve A by means of an algorithm solving B. In the context of approximation, the reduction should guarantee that an approximate solution of B can be used to obtain an approximate solution for A. To this aim, one needs a function f mapping an instance x of A to an instance x' of B, and also some other function g mapping back an approximate solution y' for x' into an approximate solution y for x (see Figure 2.1). As a matter of fact, most of the known reductions between NP-complete problems do explicitly define such a correspondence between the solutions.

We formalize this intuition by introducing the notion of *reduction template*.

Definition 2.11 (Reduction template) *Let A and B be two* NPO *problem, a* reduction template *(f, g) between them is an ordered 2-tuple of polynomial-time computable functions such that the following properties hold.*

1. *For any $x \in I_A$, $f(x) \in I_B$.*
2. *For any $x \in I_A$, if $\mathsf{sol}(x) \neq \emptyset$ then also $\mathsf{sol}(f(x)) \neq \emptyset$.*
3. *For any $y' \in \mathsf{sol}(f(x))$, $g(x, y') \in \mathsf{sol}(x)$.*

The second condition in the above definition may seem not natural. Its justification is as follows. Assume B is a problem such that a $x_{\text{void}} \in I_B$ exists such that $\mathsf{sol}(x_{\text{void}}) = \emptyset$ (for example, we can take $B =$ Linear Programming). Then, without condition 2, for any $A \in$ NPO, (f, g) would be a reduction template between A and B provided $f(x) = x_{\text{void}}$ for any $x \in I_A$. Indeed, in such case, condition (3) would be vacuosly true. Approximation-preserving reducibilities are defined by strengthening condition (3): without condition (2) there would be trivial reducibilities between any NPO problem and Linear Programming, so that Linear Programming (a problem solvable to optimality in polynomial time) would be NPO-complete.

Approximation preserving reducibilities are defined by imposing some relation between the performance ratios of y' and $g(x, y')$. The strictest notion of approximation preserving reducibilities is the S-reducibility (for *strict* reducibility) introduced by Orponen and Manilla [118].

Definition 2.12 (S-reducibility) *Let A and B be two* NPO *problems. A is said to be* S-reducible *to B if a reduction template (f, g) exists between A and B such that for any $x \in I_A$, any $y' \in \mathsf{sol}(f(x))$, it holds $E(x, g(x, y') \leq E(f(x), y')$.*

Some reductions in the literature are S-reductions, like the standard reduction between Max Clique and Max Independent Set. In Section 4.6.1 we shall see an S-reduction between Min Vertex Cover and Min Sat. It is straightforward to check that S-reducibility is transitive and that it preserves membership in PTAS and in APX.

The L-reducibility (for *linear* reducibility) introduced by Papadimitriou and Yannakakis [123] is often the most practical reducibility to use in order to show that a problem is at least as hard to approximate as another one.

Definition 2.13 (L-reducibility) *Let A and B be two* NPO *problems. A is said to be* L-reducible *to B, in symbols $A \leq_{\mathrm{L}} B$, if a reduction template (f, g) between A and B and two positive constants α and β exist such that:*

1. *For any $x \in I_A$, $\mathsf{opt}_B(f(x)) \leq \alpha \mathsf{opt}_A(x)$.*
2. *For any $x \in I_A$ and for any $y \in \mathsf{sol}_B(f(x))$,*

$$|\mathsf{opt}_A(x) - \mathsf{m}_A(x, g(x, y))| \leq \beta |\mathsf{opt}_B(f(x)) - \mathsf{m}_B(f(x), y)|$$

The fourtuple (f, g, α, β) is said to be an L-reduction from A to B.

Clearly, the L-reducibility preserves membership in PTAS. Indeed, if (f, g, α, β) is an L-reduction from A to B then, for any $x \in I_A$ and for any $y \in sol_B(f(x))$, we have that

$$E_A(x, g(x, y)) \leq \alpha\beta E_B(f(x), y),$$

so that if $B \in$ PTAS then $A \in$ PTAS [123]. The above inequality also implies that if A is a minimization problem and an r-approximate algorithm for B exists, then a $(1 + \alpha\beta(r - 1))$-approximate algorithm for A exists. L-reducibility preserves approximation in a somehow strange way, putting separate constraints on the optima and on the additive errors of the two problems. The E-reducibility (for *error* reducibility) introduced by Khanna et al. [102], instead, imposes a linear relation directly between the performance ratios.

Definition 2.14 (E-reducibility) *Let A and B be two* NPO *problems. A is said to be* E-reducible *to B, in symbols $A \leq_{\mathrm{E}} B$, if a reduction template (f, g) between A and B and a positive constant α exist such that, for any $x \in I_A$ and for any $y \in sol_B(f(x))$,*

$$R_A(x, g(x, y)) \leq 1 + \alpha(R_B(f(x), y) - 1).$$

The triple (f, g, α) is said to be an E-reduction from A to B.

Observe that, for any function r, an E-reduction maps $r(n)$-approximate solutions into $(1 + \alpha(r(n^h) - 1))$-approximate solutions where h is a constant depending only on the reduction. Hence, the E-reducibility not only preserves membership in PTAS but also membership in expAPX, polyAPX, logAPX, and APX.

Other reducibilities appearing in the literature are the P-reducibility (for PTAS-preserving reducibility) and the A-reducibility (for APX-preserving reducibility) defined by Crescenzi and Panconesi [123] (similar reducibilities are also defined in [118]).

Definition 2.15 (P-reducibility) *Let A and B be two* NPO *problems. A is said to be* P-reducible *to B, in symbols $A \leq_{\mathrm{P}} B$, if a reduction template (f, g) between A and B and a computable function $c : (1, \infty) \cap \mathcal{Q}^+ \to (1, \infty) \cap \mathcal{Q}^+$ exist such that, for any $x \in I_A$, for any $y \in sol_B(f(x))$, for any rational $r > 1$,*

$$R_B(f(x), y) \leq c(r) \textit{ implies } R_A(x, g(x, y)) \leq r\ .$$

The triple (f, g, c) is said to be a P-reduction from A to B.

Note that an E-reduction is also a P-reduction with $c(r) = 1 + (r - 1)/\alpha$, while an L-reduction is also a P-reduction with $c(r) = 1 + \frac{r-1}{\alpha\beta r}$.

Definition 2.16 (A-reducibility) *Let A and B be two* NPO *problems. A is said to be* A-reducible *to B, in symbols $A \leq_{\mathrm{A}} B$, if a reduction template (f, g) between A and B and a computable function $c : (1, \infty) \cap \mathcal{Q}^+ \to (1, \infty) \cap \mathcal{Q}^+$ exist such that, for any $x \in I_A$, for any $y \in sol_B(f(x))$, for any rational $r > 1$,*

$$R_A(x, g(x, y)) \leq c(r) \textit{ implies } R_B(f(x), y) \leq r\ .$$

The triple (f, g, c) is said to be a P-reduction from A to B.

The following results will be useful in the next chapter.

Theorem 2.17 ([102]) *Let $\mathcal{C} \in \{$APX, logAPX, polyAPX, expAPX$\}$, let A be a minimization problem in $\mathcal{C}$. Then there exists a maximization problem $B \in \mathcal{C}$ such that $A \leq_{\mathrm{E}} B$ and $A \leq_{\mathrm{L}} B$.*

Theorem 2.18 ([123]) MAX 3SAT *is* Max SNP*-complete under L-reductions.*

2.3 PCP and the Hardness of Approximation

In what follows, we summarize the main definitions and a few results from the theory of probabilistically checkable proofs. Our exposition follows [25].

A *promise problem* is a pair (Y, N) of disjoint sets of strings from some fixed alphabet Σ (we can assume without loss of generality $\Sigma = \{0, 1\}$). An algorithm *solves* a promise problem (Y, N) if for any input string $x \in Y$ the algorithm accepts, and for any input string $x \in N$ the algorithm rejects. The behaviour of the algorithm can be arbitrary when it receives in input a string from $\Sigma^* - (Y \cup N)$. A language L can be seen as the promise problem $(L, \Sigma^* - L)$.

A *verifier* is an oracle probabilistic polynomial-time Turing machine V. During its computation, V tosses random coins, reads its input and has oracle access to a string π called *proof*. In particular, let a be the sequence of oracle answers received by V during the course of its computation on input x and random string R. If V accepts in that particular circumstance, then we say that (x, R, a) is an accepting configuration for V. Let now x be an input and π be a proof. We denote by $\mathbf{ACC}[V^\pi(x)]$ the probability over its random tosses that V accepts x using π as an oracle. We also denote by $\mathbf{ACC}[V(x)]$ the maximum of $\mathbf{ACC}[V^\pi(x)]$ over all proofs π.

Several parameters determine the efficiency of the proof checking.

Definition 2.19 (PCP parameters) *Let (Y, N) be a promise problem, and let V be a verifier for (Y, N). Then we say that*

- V uses $r(n)$ random bits *(where $r : \mathcal{Z}^+ \to \mathcal{Z}^+$ is an integer function) if for any input x and for any proof π, V tosses at most $r(|x|)$ random coins;*
- V has query complexity q *(where q is an integer) if for any input x, any random string R, and any proof π, V reads at most q bits from π;*
- V has free bit complexity f *(where f is a real) if for any input x and any random string R, there are at most 2^f set of answers a such that (x, R, a) is an accepting configuration for V;*
- V has soundness s *(where $s \in [0, 1]$ is a real) if, for any $x \in N$, $\mathbf{ACC}[V(x)] \leq s$;*
- V has completeness c *(where $c \in [0, 1]$ is a real) if, for any $x \in Y$, $\mathbf{ACC}[V(x)] \geq c$.*

We note that the number of free bits cannot exceed the number of query bits.

Definition 2.20 (PCP classes) *Let (Y, N) be a promise problem, let $0 < s < c \leq 1$ be any constants, let $f > 0$ be a real, q be a positive integer and $r : \mathcal{Z}^+ \to \mathcal{Z}^+$, then we say that $(Y, N) \in \mathsf{FPCP}_{c,s}[r, f, q]$ if a verifier V exists for (Y, N) that uses $O(r(n))$ random bits, has query complexity q, free bit complexity f, soundness s and completeness c. $\mathsf{PCP}_{c,s}[r, q]$ is the union of $\mathsf{FPCP}_{c,s}[r, f, q]$ for all $f \leq q$.*

Using the above notation (introduced in [25] and slightly different from the one of [10]), we can state the PCP Theorem as follows.

Theorem 2.21 (PCP Theorem [10, 9]) *A constant q exists such that*

$$\mathsf{NP} \subseteq \mathsf{PCP}_{1,1/2}[\log, q] \ .$$

Sometimes we shall also use the short-hand notation $\mathsf{PCP}(r(n), q(n)) \stackrel{\text{def}}{=} \mathsf{PCP}_{2/3,1/3}[r(n), O(q(n))]$. Note that the constants $2/3$ and $1/3$ are arbitrarily chosen, that is, for any fixed c and s with $0 < s < c < 1$ it holds $\mathsf{PCP}(r(n), q(n)) = \mathsf{PCP}_{c,s}[r(n), O(q(n))]$.

The following theorem shows that the existence of efficient verifiers for any NP problem implies a non-approximability result for Min Vertex Cover.

Theorem 2.22 (Non-approximability of MIN VERTEX COVER **[61, 25])** *Let us assume that* $\mathsf{NP} \subseteq \mathsf{FPCP}_{c,s}[\log, f, q]$. *Then, for any* $\varepsilon > 0$, *it is* NP*-hard to approximate* MIN VERTEX COVER *to within*

$$1 + \frac{c-s}{2^f - c} - \varepsilon \ .$$

PROOF: [Sketch] Let φ be an instance of the SAT problem, and let us consider the behavior of the verifier claimed in the theorem with input φ and a proof π. Let $\mathbf{r} = 2^{O(\log n)}$ be the total (polynomial) number of possible random sequences accessed by the verifier. For any of these sequences R, there are at most 2^f different accepting configurations (x, R, a). We say that two configurations (x, R, a) and (x, R', a') are *consistent* if a proof π exists such that a (respectively, a') is the set of answers received during the computation $V^{\pi}(x, R)$ (respectively, $V^{\pi}(x, R')$). We construct a graph G_{φ} with a node for each accepting configurations (adding dummy configurations, we make sure that there are exactly $2^f\mathbf{r}$ nodes). Then we put an edge between u and v if and only if u and v are not consistent. It is possible to show (see [61]) that there is an independent set in G_{φ} with at least k nodes if and only if there exists a proof for φ that makes the verifier accept at least k times over $\mathbf{r}$ (i.e. with probability $k/\mathbf{r}$). Observe that a graph G_{φ} with n nodes has an independent set with k nodes if and only if it has a vertex cover with $n - k$ nodes. It follows that if φ is satisfiable then there exists a vertex cover in G_{φ} with at most $\mathbf{r}(2^f - c)$ nodes; otherwise any vertex cover in G_{φ} will have at least $\mathbf{r}(2^f - s)$ nodes. Thus, any approximation factor better than $(2^f - s)/(2^f - c)$ would be sufficient to decide the satisfiability of φ. •

In the following, the graphs G_{φ} arising from the above described construction will be called *FGLSS graphs.*

At the moment of writing this dissertation, the best PCP construction with few query and free bits is the following.

Theorem 2.23 (Håstad [80]) *For any constant* $\varepsilon > 0$, $\mathsf{NP} \subseteq \mathsf{FPCP}_{1-\varepsilon,.5+\varepsilon}[\log, 2, 3]$.

As a consequence, it is NP-hard to 1.166-approximate MIN VERTEX COVER.

2.4 Query Complexity

In this section we review a few definitions and results about complexity classes defined in terms of *bounded queries* to an NP oracle.

Definition 2.24 *Let* $q : \mathcal{Z}^+ \to \mathcal{Z}^+$. *A language* L *belongs to the* class $\mathsf{P}^{\mathsf{NP}[q(n)]}$ *if and only if it is decidable by a polynomial-time oracle Turing machine which asks at most* $q(n)$ *queries to an* NP*-complete oracle, where* n *is the input size. Moreover, for a class* Q *of functions, we let* $\mathsf{P}^{\mathsf{NP}[Q]} = \bigcup_{q \in Q} \mathsf{P}^{\mathsf{NP}[q(n)]}$.

The *class* QH (for Query Hierarchy) is equal to the union $\bigcup_{q \geq 1} \mathsf{P}^{\mathsf{NP}[q]}$. We also let $\mathsf{P}^{\mathsf{NP}} \stackrel{\text{def}}{=} \mathsf{P}^{\mathsf{NP}[n^{O(1)}]}$. The query hierarchy equals the *boolean hierarchy*, that has been the object of extensive studies (see [38] and the references therein). Here we are interested in separation results for query complexity classes. The following result is due to Kadin.

Theorem 2.25 ([90, 91]) *If a constant* q *exists such that*

$$\mathsf{QH} = \mathsf{P}^{\mathsf{NP}[q]},$$

then the polynomial-time hierarchy collapses.

Wagner extended the above result to the case of a non-constant number of queries.

Theorem 2.26 ([133]) *Let $q(n) \leq \log n$ be a polynomial-time computable function. If $\mathsf{P}^{\mathsf{NP}[q(n)]} \subseteq \mathsf{P}^{\mathsf{NP}[q(n)-1]}$ then the polynomial-time hierarchy collapses.*

It is also possible (and sometimes useful) to define classes of *functions* that are computable using bounded queries.

Definition 2.27 *A function $f : \mathcal{Z}^+ \to \mathcal{Z}^+$ belongs to the* class $\mathsf{PF}^{\mathsf{NP}[q(n)]}$ *if and only if it is computable by a polynomial-time oracle Turing machine which asks at most $q(n)$ queries to an* NP*-complete oracle, where n is the input size. Moreover, if Q is a class of functions, then $\mathsf{PF}^{\mathsf{NP}[Q]} = \bigcup_{q \in Q} \mathsf{PF}^{\mathsf{NP}[q(n)]}$.*

2.5 Definition of the Problems

Here we give the definitions of all the problems that are mentioned in the thesis. Following the format of the compendium of Crescenzi and Kann [45], we define for every problem the set of instances, the set of feasible solutions for a given instance and the objective function. We do not specify if the problem is a maximization or a minimization one, since it is apparent from the name.

2.5.1 Constraint Satisfaction

Definition 2.28 *A (k-ary)* constraint function *is a boolean function $f : \{0,1\}^k \to \{0,1\}$.*

When it is applied to variables $x_1, \ldots, x_k$ (see the following definitions) the function f is thought of as imposing the constraint $f(x_1, \ldots, x_k) = 1$.

Definition 2.29 *A* constraint family *$\mathcal{F}$ is a finite collection of constraint functions. The* arity *of $\mathcal{F}$ is the maximum arity of the functions in $\mathcal{F}$.*

Definition 2.30 *A* constraint *C over a variable set $x_1, \ldots, x_n$ is a pair $C = (f, (i_1, \ldots, i_k))$ where $f : \{0,1\}^k \to \{0,1\}$ is a constraint function and $i_j \in [n]$ for $j \in [k]$. Variable x_j is said to* occur *in C if $j \in \{i_1, \ldots, i_k\}$. The constraint C is said to be* satisfied *by an assignment $\vec{a} = a_1, \ldots, a_n$ to $x_1, \ldots, x_n$ if $C(a_1, \ldots, a_n) \stackrel{\text{def}}{=} f(a_{i_1}, \ldots, a_{i_k}) = 1$. We say that constraint C is* from *$\mathcal{F}$ if $f \in F$.*

Problem 1 (MAX WEIGHT $\mathcal{F}$, MAX $\mathcal{F}$, MAX SIMPLE $\mathcal{F}$) *For a function family $\mathcal{F}$, the* constraint satisfaction problem csp($\mathcal{F}$)*, which we shall also refer to as* MAX WEIGHT $\mathcal{F}$*, is the optimization problem defined as follows:*

INSTANCE: *A collection $\varphi = \{C_1, \ldots, C_m\}$ of constraints over variable set $X = \{x_1, \ldots, x_n\}$ and associated non-negative weights $w_1, \ldots, w_m$.*

SOLUTION: *A truth assignment for the variables in X.*

MEASURE: *Total weight of the constraints satisfied by the truth assignment.*

MAX SIMPLE $\mathcal{F}$ *is the restriction of* MAX WEIGHT $\mathcal{F}$ *where all weights are equal to 1.* MAX $\mathcal{F}$ *is the variation of* MAX SIMPLE $\mathcal{F}$ *where φ is allowed to be a* multiset *of clauses. Equivalently,* MAX $\mathcal{F}$ *is the variation of* MAX WEIGHT $\mathcal{F}$ *where the weights are written in unary.*

An assignment of values to $x_1, \ldots, x_n$ can be represented as a function $\tau : \{x_1, \ldots, x_n\}^n \to \{0,1\}$, with the meaning that x_i is assigned value $\tau(x_i)$, or as a sequence $a_1, \ldots, a_n$, with the meaning that x_i is assigned value a_i. The two conventions are clearly equivalent, and we will use in some cases the former and in some cases the latter depending on which one will be more convenient each time.

We now give a comprehensive list of all the constraints and constraint families used in this thesis.

Definition 2.31 *We define the following constraint families.*

- Parity check *is the constraint family* PC $= \{\mathrm{PC}_0, \mathrm{PC}_1\}$*, where, for* $i \in \{0,1\}$, PC_i *is defined as follows*

$$\mathrm{PC}_i(a_1, a_2, a_3) = \begin{cases} 1 & \textit{if } a_1 \oplus a_2 \oplus a_3 = i \\ 0 & \textit{otherwise.} \end{cases}$$

- Respect of monomial basis check *is the constraint family* RMBC $= \{\mathrm{RMBC}_{ij} : i, j \in \{0,1\}\}$*, where* RMBC $: \{0,1\}^4 \to \{0,1\}$ *with*

$$\mathrm{RMBC}_{ij}(a_1, a_2, a_3, a_4) = \begin{cases} 1 & \textit{if } a_1 = 0 \textit{ and } a_2 = a_3 \oplus i \\ 1 & \textit{if } a_1 = 1 \textit{ and } a_2 = a_4 \oplus j \\ 0 & \textit{otherwise.} \end{cases}$$

  RMBC_{00} *may be thought of as the test* $(a_3, a_4)[a_1] \stackrel{?}{=} a_2$, RMBC_{01} *as the test* $(a_3, \neg a_4)[a_1] \stackrel{?}{=} a_2$, RMBC_{10} *as the test* $(\neg a_3, a_4)[a_1] \stackrel{?}{=} a_2$*, and* RMBC_{11} *as the test* $(a_3, a_4)[a_1] \stackrel{?}{=} \neg a_2$.

- *For any* $k \geq 1$, Exactly-k-SAT *is the constraint family* EkSAT $= \{f : \{0,1\}^k \to \{0,1\} : |\{\vec{a} : f(\vec{a}) = 0\}| = 1\}$*, that is, the set of* k*-ary* disjunctive *constraints.*

- *For any* $k \geq 1$, k-SAT *is the constraint family* $k\textsc{Sat} = \bigcup_{l \in [k]} \mathrm{E}l\mathrm{SAT}$.

- SAT *is the constraint family* $\textsc{Sat} = \bigcup_{k \geq 1} \mathrm{E}k\mathrm{SAT}$

- *For any* $B \geq 1$, Generalized SAT-B *is the constraint family* GSAT-B *consisting of all boolean functions that admit a DNF representation where all the conjunctions have no more than* B *terms.*

- *For any* $k \geq 1$ k-Conjunctive SAT *is the constraint family* $k\textsc{ConjSAT} = \{f : \{0,1\}^h \to \{0,1\} : |\{\vec{a} : f(\vec{a}) = 1\}| = 1 \wedge h \leq k\}$*, that is the set of* conjunctive *constraints with no more than* k *literals.*

- CUT *is the constraint function* $\textsc{Cut} : \{0,1\}^2 \to \{0,1\}$ *with* $\textsc{Cut}(a_1, a_2) = a_1 \oplus a_2$.

- DICUT *is the constraint function* $\textsc{DiCut} : \{0,1\}^2 \to \{0,1\}$ *with* $\textsc{DiCut}(a_1, a_2) = \neg a_1 \wedge a_2$.

- *For any* $k \geq 1$, k-CSP *is the constraint family consisting of all* h*-ary boolean functions, for* $h \leq k$*, i.e.* k-CSP $= \{f : \{0,1\}^h \to \{0,1\} \ : \ h \leq k\}$.

The definition of Max Weight Cut, Max Cut, Max Weight DiCut, and Max DiCut as constraint satisfaction problems is equivalent to the standard definition as graph problems.

For constraint family $\mathcal{F}$, besides the Max Weight $\mathcal{F}$ and the Max $\mathcal{F}$ problem, we can also define the so-called max ones and min ones problems.

Problem 2 (Max Ones $\mathcal{F}$ **and** Min Ones $\mathcal{F}$) *For a constraint family* $\mathcal{F}$*, the* Max Ones $\mathcal{F}$ *problem (respectively, the* Min Ones $\mathcal{F}$ *problem) is defines as follows.*

Instance: *A collection* $\varphi = \{C_1, \ldots, C_m\}$ *of constraints from* $\mathcal{F}$ *over variable set* $X = \{x_1, \ldots, x_n\}$ *and associated non-negative weights* $w_1, \ldots, w_n$.

Solution: *A truth assignment* τ *for the variables in* X *that satisfies all the constraints in* φ.

Measure: *Total weight of the variables set to* 1 *by* τ.

2.5.2 Packing and Covering

Problem 3 (MIN BIN PACKING)

INSTANCE: *Finite set U of items, and a size $s(u) \in Q \cap (0,1]$ for each $u \in U$.*

SOLUTION: *A partition of U into disjoint sets $U_1, U_2, \ldots, U_m$ such that the sum of the sizes of the items in each U_i is at most 1.*

MEASURE: *The number of used bins, i.e., the number m of disjoint sets.*

Problem 4 (MIN ORDERED BIN PACKING)

INSTANCE: *Finite set U of items, a size $s(u) \in Q \cap (0,1]$ for each $u \in U$, and a partial order $\preceq$ on U.*

SOLUTION: *A partition of U into disjoint sets $U_1, U_2, \ldots, U_m$ such that the sum of the sizes of the items in each U_i is at most 1 and if $u \in U_i$ and $u' \in U_j$ with $u \preceq u'$, then $i \leq j$.*

MEASURE: *The number of used bins, i.e., the number m of disjoint sets.*

Problem 5 (MIN DEGREE SPANNING TREE)

INSTANCE: *Graph $G = (V, E)$.*

SOLUTION: *A spanning tree for G.*

MEASURE: *The maximum degree of the spanning tree.*

Problem 6 (MIN WEIGHT VERTEX COVER)

INSTANCE: *Graph $G = (V, E)$ and weights $\{w(u)\}_{u \in V}$.*

SOLUTION: *A vertex cover of G, i.e. a set $C \subseteq V$ such that C contains at least one endpoint of any edge of E.*

MEASURE: *Total weight of vertices of C, i.e. $\sum_{u \in C} w(u)$.*

The MIN VERTEX COVER problem is the restriction of the MIN WEIGHT VERTEX COVER problem where all the weights are 1. For a constant $B \geq 1$, MIN VERTEX COVER-B is the restriction of the MIN VERTEX COVER problem to graphs of maximum degree B.

Problem 7 (MIN SET COVER)

INSTANCE: *Family of subsets $S_1, \ldots, S_m$ of a set U such that $\bigcup_{i=1}^{m} S_i = U$.*

SOLUTION: *A cover of U, i.e. a subfamily $C = \{S_{i_1}, \ldots, S_{i_k}\} \subseteq \{S_1, \ldots, S_m\}$ such that $\bigcup_{j=1}^{k} S_{i_j} = U$.*

MEASURE: *The cardinality of the cover, i.e. k.*

Problem 8 (MIN HITTING SET)

INSTANCE: *Family of subsets $S_1, \ldots, S_m$ of a set U.*

SOLUTION: *A subset $H \subseteq U$ such that, for any $i = 1, \ldots, m$, $H \cap S_i \neq \emptyset$.*

MEASURE: *The cardinality of H.*

Problem 9 (Min Dominating Set)

Instance: *A graph $G = (V, E)$.*

Solution: *A subset $D \subseteq V$ such that for any $u \in V - D$ there exists a $v \in D$ such that u and v are adjacent in G.*

Measure: *The cardinality of D.*

2.5.3 Clique and Coloring

Problem 10 (Max Weight Clique)

Instance: *Graph $G = (V, E)$ and weights $\{w(u)\}_{u \in V}$.*

Solution: *A clique in G, i.e. a subset $V' \subseteq V$ such that every two vertices in V' are joined by an edge in E.*

Measure: *Total weight of the clique, i.e. $\sum_{u \in V'} w(u)$.*

The Max Clique problem is the restriction of the Max Weight Clique problem where all weights are equal to one.

Problem 11 (Min Edge Coloring)

Instance: *Graph $G = (V, E)$.*

Solution: *A coloring of E, i.e., a partition of E into disjoint sets $E_1, E_2, \ldots, E_k$ such that, for $1 \le i \le k$, no two edges in E_i share a common endpoint in G.*

Measure: *Cardinality of the coloring, i.e., the number k of disjoint sets.*

Problem 12 (Min Node Coloring)

Instance: *Graph $G = (V, E)$.*

Solution: *A coloring of V, i.e., a partition of V into disjoint sets $V_1, V_2, \ldots, V_k$ such that, for $1 \le i \le k$, no two vertices in V_i are adjacent in G.*

Measure: *Cardinality of the coloring, i.e., the number k of disjoint sets.*

2.5.4 Mathematical Programming

Problem 13 (Linear Programming)

Instance: *Rational $m \times n$-matrix A, rational m-vector b, rational n-vector c.*

Solution: *A rational n-vector x such that $Ax \ge b$.*

Measure: $1 + \sum_{i=1}^{n} c_i x_i$.

Problem 14 (Max PB $0-1$ Programming **and** Min PB $0-1$ Programming)

Instance: *Integer $m \times n$-matrix A, integer m-vector b, binary n-vector c.*

Solution: *A binary n-vector x such that $Ax \ge b$.*

Measure: $1 + \sum_{i=1}^{n} c_i x_i$.

2.5.5 Edge Deletion

Problem 15 (Min Feedback Vertex Set)

Instance: *A directed graph* $G = (V, E)$.

Solution: *A set of vertices* $S \subseteq V$ *such that* $V - S$ *induces an acyclic subgraph.*

Measure: *Cardinality of* S.

Problem 16 (Min Clause Deletion 2SAT)

Instance: *A set* $\varphi = \{C_1, \ldots, C_m\}$ *of constraints from* 2SAT *over variable set* X.

Solution: *A subset* $S \subset \varphi$ *of constraints such that* $\varphi - S$ *is satisfiable.*

Measure: *Cardinality of* S.

Problem 17 (Min Edge Deletion Bipartite Subgraph)

Instance: *A graph* $G = (V, E)$.

Solution: *A set of edges* $E' \subseteq E$ *such that* $G' = (V, E - E')$ *is bipartite.*

Measure: *Cardinality of* E'.

2.5.6 Other Problems

Problem 18 (Min Steiner Tree)

Instance: *A complete graph* $G = (V, E)$ *with associated weights* $\{d(u, v)\}_{(u,v)\in E}$ *and a set of special vertices* $U \subseteq V$.

Solution: *A set of steiner vertices* $S \subseteq V - U$ *such that* $S \cup U$ *induce a connected subgraph* $G_{S,U}$ *of* G*, and a spanning tree of* $G_{S,U}$.

Measure: *Cost of the spanning tree.*

Problem 19 (Min Linear Arrangment)

Instance: *A graph* $G = (V, E)$ *with* n *vertices and associated weights* $\{w(u, v)\}_{(u,v)\in E}$.

Solution: *A mapping* $\pi : V \rightarrow [n]$.

Measure: $\sum_{(u,v)\in E} w(u, v)|\pi(u) - \pi(v)|$.

Chapter 3

Structure in Approximation Classes

There is a theory which states that if ever anyone discovers exactly what the Universe is for and why it is here, it will instantly disappear and be replaced by something even more bizarre and inexplicable.
There is another which states that this has already happened.

(D. Adams [1])

The first goal of this chapter is to define an approximation preserving reducibility that can be used for as many approximation classes as possible and such that all reductions that have appeared in the literature still hold. In spite of the fact that the L-reducibility has been the most widely used so far, we will give strong evidence that it cannot be used to obtain completeness results in "computationally defined" classes such as APX, logAPX, and polyAPX. Indeed, on the one hand we prove that the L-reducibility is *too weak* and is not approximation preserving (unless $\mathsf{P} = \mathsf{NP} \cap co\text{-}\mathsf{NP}$), on the other hand it is *too strict* since it does not allow to reduce some problems which are known to be easy to approximate to problems which are known to be hard to approximate. The strictness of the L-reducibility is shared by all the known reducibilities (unless $\mathsf{P}^{\mathsf{NP}} = \mathsf{P}^{\mathsf{NP}[O(\log n)]}$). We introduce a new kind of approximation-preserving reducibility that overcomes these limitations and allows to prove all approximation completeness results that have appeared in the literature, plus a few new ones.

In particular, we prove the existence of natural NPO-complete problems (previously only Max NPO-complete and Min NPO-complete problems were known), of natural NPO PB-complete problems (again, overcoming the dichotomy between maximization and minimization), of natural APX-complete problem (previously only natural APX PB-complete and artificial APX-complete problems were known), and natural logAPX-complete and polyAPX-complete problems (previous results were confined to logAPX PB and polyAPX PB, respectively).

All the above completeness results (with the exception of the NPO- and NPO PB-completeness) use the PCP Theorem in their proof. We show that a simpler, but non-trivial, completeness result can be obtained with a direct reduction. Namely, we give a simple proof of the Max NP-completeness of MAX 3SAT (no Max NP-complete problem was known before without the use of PCP techniques). Our reduction gives a satisfying answer to question of why Max SNP and Max NP problems are equally hard to approximate and has interesting algorithmic applications. In fact, we use it to improve the running time of previously known approximation schemes for "planar" instances of satisfiability problems.

Another group of results refers to new connections between the approximability properties and the query complexity of NP-hard optimization problems. In several recent papers the notion of query complexity (that is, the number of queries to an NP oracle needed to solve a given problem) has been shown to be a very useful tool for understanding the complexity of approximation problems. Such previous results include upper and lower bounds on the number of queries needed to approximate certain optimization problems (such as MAX SAT and MAX CLIQUE): these results deal with the complexity of approximating the value of the optimum solution and not with the complexity of computing approximate solutions. In this chapter, instead, the complexity of "constructive" approximation will be addressed by considering the languages that can be recognized by polynomial-time machines which have a function oracle that solves the approximation problem. We prove that completeness in approximation classes implies lower bounds on the query complexity. Indeed, we can say more: we prove that the two approaches are basically equivalent by giving *sufficient and necessary conditions for approximation completeness in terms of query-complexity hardness and combinatorial properties.* The importance of these results is twofold: they give new insights into the structure of complete problems for approximation classes and they reconcile the approach based on standard computation models with the approach based on the computation model for approximation proposed in [39]. As a final observation, our results can be seen as extensions of a result of [102] in which general sufficient (but not necessary) conditions for APX-completeness are proved.

We apply our query complexity results proving the existence of natural APX-intermediate problems, solving an open question of Chang [40] about the approximability of MAX CLIQUE, and proving that the MIN SET COVER problem is not self-improvable (all the results assume that the polynomial hierarchy does not collapse).

3.1 Reducibilities

In this section we shall define a new approximation preserving reducibility. We will justify our definition by emphasizing the disadvantages of previously known ones.

3.1.1 The L-reducibility

Theorem 3.1 *The following statements are equivalent:*

1. *Two problems $A \in$ Max NPO and $B \in$ Min NPO exist such that $A \notin$ APX, $B \in$ APX, and $A \leq_{\mathrm{L}} B$.*
2. *Two Max NPO problems A and B exist such that $A \notin$ APX, $B \in$ APX, and $A \leq_{\mathrm{L}} B$.*
3. *A polynomial-time recognizable set of satisfiable Boolean formulas exists for which no polynomial-time algorithm can compute a satisfying assignment for each of them.*

PROOF:

(1) $\Rightarrow$ (2). In this case, it suffices to L-reduce B to a maximization problem C in APX (see Theorem 2.17).

(2) $\Rightarrow$ (3). Assume that for any polynomial-time recognizable set of satisfiable Boolean formulas there is a polynomial-time algorithm computing a satisfying assignment for each formula in the set. Suppose that (f, g, α, β) is an L-reduction from a maximization problem A to a maximization problem B and that B is r-approximable for some $r > 1$. Let x be an instance of A and let y be a solution of $f(x)$ such that $\mathsf{opt}_B(f(x))/\mathsf{m}_B(f(x), y) \leq r$. For the sake of convenience, let $\mathsf{opt}_A = \mathsf{opt}_A(x)$, $\mathsf{m}_A = \mathsf{m}_A(x, g(x, y))$, $\mathsf{opt}_B = \mathsf{opt}_B(f(x))$, and $\mathsf{m}_B = \mathsf{m}_B(f(x), y)$. Let also $\mathsf{m}_x = \max\{\mathsf{m}_A, \mathsf{m}_B/\alpha\}$. Since $\mathsf{m}_A \leq \mathsf{opt}_A$ and $\mathsf{m}_B/\alpha \leq \mathsf{opt}_B/\alpha \leq \mathsf{opt}_A$, we have that $\mathsf{m}_x \leq \mathsf{opt}_A$. We now show that $\mathsf{opt}_A/\mathsf{m}_x \leq 1 + \alpha\beta(r-1)$, that is, m_x approximates the value of opt_A. Let $\gamma = \frac{\alpha r}{1+\alpha\beta(r-1)}$. There are two cases.

1. $\mathsf{opt}_B \leq \gamma \mathsf{opt}_A$. By the definition of the L-reducibility, $\mathsf{opt}_A - \mathsf{m}_A \leq \beta(\mathsf{opt}_B - \mathsf{m}_B)$. Since $\mathsf{opt}_B \leq \gamma \mathsf{opt}_A$ and $\mathsf{opt}_B/\mathsf{m}_B \leq r$, we have that

$$\frac{\mathsf{opt}_A - \mathsf{m}_A}{\mathsf{opt}_A} \leq \gamma\beta \frac{\mathsf{opt}_B - \mathsf{m}_B}{\mathsf{opt}_B} \leq \gamma\beta(1 - 1/r).$$

   Hence,

$$\frac{\mathsf{opt}_A}{\mathsf{m}_x} \leq \frac{\mathsf{opt}_A}{\mathsf{m}_A} \leq \frac{1}{1 - \gamma\beta\frac{r-1}{r}} = 1 + \alpha\beta(r-1)$$

   where the last equality is due to the definition of γ.

2. $\mathsf{opt}_B > \gamma \mathsf{opt}_A$. It holds that

$$\begin{aligned}
\frac{\mathsf{opt}_A}{\mathsf{m}_x} &\leq \frac{\mathsf{opt}_A}{\mathsf{m}_B/\alpha} \\
&< \frac{\alpha(\mathsf{opt}_B/\gamma)}{\mathsf{m}_B} \text{ (since } \mathsf{opt}_A < \mathsf{opt}_B/\gamma) \\
&\leq \frac{\alpha(\mathsf{opt}_B/\gamma)}{(\mathsf{opt}_B/r)} \text{ (since } \mathsf{m}_B \geq \mathsf{opt}_B/r) \\
&= \frac{\alpha r}{\gamma} \\
&= 1 + \alpha\beta(r-1).
\end{aligned}$$

Let us now consider the following non-deterministic polynomial-time algorithm.

begin {input: $x \in I_A$}
  compute m_x by using the r-approximate algorithm for B and the L-reduction from A to B;
  guess $y \in \mathsf{sol}_A(x)$;
  if $\mathsf{m}_A(x, y) \geq \mathsf{m}_x$ **then** accept **else** reject;
end;

By applying Cook's reduction [44] to the above algorithm, it easily follows that, for any $x \in I_A$, a satisfiable Boolean formula φ_x can be constructed in polynomial time in the length of x so that any satisfying assignment for φ_x encodes a solution of x whose measure is at least m_x. Moreover, the set $\{\varphi_x : x \in I_A\}$ is recognizable in polynomial time. By assumption, it is then possible to compute in polynomial time a satisfying assignment for φ_x and thus an approximate solution for x.

(3) $\Rightarrow$ (1). Assume that a polynomial-time recognizable set S of satisfiable Boolean formulas exists for which no polynomial-time algorithm can compute a satisfying assignment for each of them. Consider the following two NPO problems $A = (I_A, \mathsf{sol}_A, \mathsf{m}_A, \max)$ and $B = (I_B, \mathsf{sol}_B, \mathsf{m}_B, \min)$ where $I_A = I_B = S$, $\mathsf{sol}_A(x) = \mathsf{sol}_B(x) = \{y : y \text{ is a truth assignment to the variables of } x\}$,

$$\mathsf{m}_A(x, y) = \begin{cases} |x| & \text{if } y \text{ is a satisfying assignment for } x, \\ 1 & \text{otherwise,} \end{cases}$$

and

$$\mathsf{m}_B(x, y) = \begin{cases} |x| & \text{if } y \text{ is a satisfying assignment for } x, \\ 2|x| & \text{otherwise.} \end{cases}$$

Clearly, problem B is in APX, while if A is in APX then there is a polynomial-time algorithm that computes a satisfying assignment for each formula in S, contradicting the assumption. Moreover, it is easy to see that A L-reduces to B via $f \equiv \lambda x.x$, $g \equiv \lambda x \lambda y.y$, $\alpha = 1$, and $\beta = 1$.

•

Observe that in [113] it is shown that the third statement of the above theorem holds if and only if the γ-reducibility is different from the many-one reducibility. Moreover, in [87] it is shown that the latter hypothesis is somewhat intermediate between $\mathsf{P} \neq \mathsf{NP} \cap co\text{-}\mathsf{NP}$ and $\mathsf{P} \neq \mathsf{NP}$.

Corollary 3.2 *The L-reducibility does not preserve membership in* APX *unless* $\mathsf{P} = \mathsf{NP} \cap \text{co-}\mathsf{NP}$.

In other words, there is strong evidence that, even though the L-reducibility is suitable for proving completeness results within classes contained in APX (such as Max SNP [123]), this reducibility cannot be used to define the notion of completeness for classes beyond APX. Moreover, it cannot be blindly used to obtain positive results, that is, to prove the existence of approximation algorithms via reductions. Finally, it is possible to L-reduce the maximization problem B defined in the last part of the proof of the previous theorem to MAX 3SAT: this implies that the closure of Max SNP with respect to the L-reducibility is not included in APX, contrary to what is commonly believed (e.g. see [120, Page 314]).

3.1.2 The E-reducibility

Since the E-reducibility preserves membership in APX, we have that NPO problems should exist which are L-reducible to each other but not E-reducible. However, the following result shows that within the class APX the E-reducibility is just a generalization of the L-reducibility.

Theorem 3.3 *For any two* NPO *problems A and B, if $A \leq_{\mathrm{L}} B$ and $A \in$* APX*, then $A \leq_{\mathrm{E}} B$.*

PROOF: Let T be an r-approximate algorithm for A with r constant and let $(f_L, g_L, \alpha_L, \beta_L)$ be an L-reduction from A to B. Then, for any $x \in I_A$ and for any $y \in \mathsf{sol}_B(f_L(x))$, $E_A(x, g_L(x,y)) \leq \alpha_L \beta_L E_B(f_L(x), y)$. If A is a minimization problem then, for any $x \in I_A$ and for any $y \in \mathsf{sol}_B(f_L(x))$,

$$\begin{aligned} R_A(x, g_L(x,y)) &= 1 + E_A(x, g_L(x,y)) \\ &\leq 1 + \alpha_L \beta_L E_B(f_L(x), y) \\ &\leq 1 + \alpha_L \beta_L (R_B(f_L(x), y) - 1), \end{aligned}$$

and thus $(f_L, g_L, \alpha_L \beta_L)$ is an E-reduction from A to B. Otherwise (that is, A is a maximization problem) we distinguish the following two cases.

1. $E_B(f_L(x), y) \leq \frac{1}{2\alpha_L \beta_L}$: in this case we have that

$$\begin{aligned} R_A(x, g_L(x,y)) - 1 &= \frac{E_A(x, g_L(x,y))}{1 - E_A(x, g_L(x,y))} \\ &\leq \frac{\alpha_L \beta_L E_B(f_L(x), y)}{1 - \alpha_L \beta_L E_B(f_L(x), y)} \\ &\leq 2\alpha_L \beta_L (R_B(f_L(x), y) - 1). \end{aligned}$$

2. $E_B(f_L(x), y) > \frac{1}{2\alpha_L \beta_L}$: in this case we have that $R_B(f_L(x), y) - 1 \geq \frac{1}{2\alpha_L \beta_L}$ so that

$$R_A(x, T(x)) - 1 \leq r - 1 \leq 2\alpha_L \beta_L (r-1)(R_B(f_L(x), y) - 1)$$

where the first inequality is due to the fact that T is an r-approximation algorithm for A.

We can thus define a triple (f_E, g_E, α_E) as follows:

1. For any $x \in I_A$, $f_E(x) = f_L(x)$.
2. For any $x \in I_A$ and for any $y \in \mathsf{sol}_B(f_E(x))$,

$$g_E(x,y) = \begin{cases} g_L(x,y) & \text{if } \mathsf{m}_A(x, g_L(x,y)) \geq \mathsf{m}_A(x, T(x)), \\ T(x) & \text{otherwise.} \end{cases}$$

3. $\alpha_E = \max\{2\alpha_L\beta_L, 2\alpha_L\beta_L(r-1)\}$.

From the above discussion it follows that (f_E, g_E, α_E) is an E-reduction from A to B. •

Khanna et al. [102], have shown that MAX 3SAT is APX PB-complete with respect to the E-reducibility. Using Theorem 3.3 and the fact that MAX 3SAT is in Max SNP [123], lots of new completeness and hardness results can be inferred from the literature.

Corollary 3.4 *If an* NPO *problem A is* Max SNP*-hard with respect to the L-reducibility, then it is* APX PB*-hard with respect to the E-reducibility.*

Clearly, the converse of Theorem 3.3 does not hold since no problem in NPO − NPO PB can be L-reduced to a problem in NPO PB while any problem in PO can be E-reduced to any NPO problem. The L-reduction is stricter than the E-reduction also inside APX PB.

Theorem 3.5 MAX 3SAT *is not* APX PB*-complete with respect to the L-reducibility, unless* P = NP.

PROOF: Assume MAX 3SAT is APX PB-complete. Then there is an L-reduction (f, g, α, β) from MIN BIN PACKING to MAX 3SAT. Let x be an instance of MIN BIN PACKING such that $\mathsf{opt}(x) \leq 3$, then $\mathsf{opt}(f(x)) \leq 3\alpha$. It is well known that if φ is an instance of MAX 3SAT with m clauses, then $\mathsf{opt}(\varphi) \geq m/2$: it follows that $f(x)$ has at most 6α clauses and thus at most 18α variables. An optimum solution for $f(x)$ can be computed in $O(6\alpha 2^{18\alpha}) = O(1)$ time. From an optimum solution for $f(x)$ one gets an optimum solution for x, and this implies that MIN BIN PACKING is solvable in polynomial time when restricted to instances whose optimum is at most three. However, this restriction of MIN BIN PACKING is known to be NP-hard. •

The E-reducibility is still somewhat too strict, and it turns out that this is true for any previously known reducibility. By making use of either the P-reducibility, or the E-reducibility it does not seem possible to prove the APX-completeness of MAX 3SAT. The difficulty is mainly due to the fact that in these reducibilities function g maps optimum solutions into optimum solutions. The following definitions allow to formally capture the property of a reducibility to be optimum-preserving.

Definition 3.6 *Given an* NPO *problem A and a rational $r \geq 1$, A_r is a multi-valued partial function that, given an instance x of A, returns the set of feasible solutions y of x such that $R(x,y) \leq r$.*

Note that, in particular, $A_1(x) = \mathsf{sol}^*_A(x)$.

Definition 3.7 *Given an* NPO *problem A and a rational $r \geq 1$, a language L belongs to P^{A_r} if two polynomial-time computable functions f and g exist such that, for any x, $f(x)$ is an instance of A with $\mathsf{sol}(f(x)) \neq \emptyset$, and, for any $y \in A_r(f(x))$, $g(x,y) = 1$ if and only if $x \in L$. We also let $\mathsf{P}^A \stackrel{\text{def}}{=} \mathsf{P}^{A_1}$.*

P^{A_r} can be seen as the set of languages that can be solved by making *one query* to an "oracle" that gives r-approximate solutions for instances of A.

Definition 3.8 *Let $\leq$ be a preorder in a set $\mathcal{C}$, that is, a transitive reflexive binary relation. If $\mathcal{A} \subseteq \mathcal{C}$ and, for any $b \in \mathcal{A}$, $b \leq a$ with $a \in \mathcal{A}$, then a is said to be a* $\leq$-maximum *in $\mathcal{A}$.*

Definition 3.9 *Let $\leq$ be a preorder in* NPO*, then $\leq$ is said to be* optimum preserving *if $A \leq B$ implies $\mathsf{P}^A \subseteq \mathsf{P}^B$.*

It is easy to see that the L-reducibility, the E-reducibility, and the P-reducibility are optimum preserving preorders in NPO, and that a $\leq$-maximum in APX is indeed an APX-complete problem.

Proposition 3.10 *If an optimum preserving preorder $\leq$ exists in* NPO *such that a polynomially bounded problem A is a $\leq$-maximum in* APX *then $\mathsf{P}^{\mathsf{NP}} = \mathsf{P}^{\mathsf{NP}[O(\log n)]}$.*

Proof: Recall that from the results of [107] it follows that $\mathsf{P}^{\mathsf{NP}} \subseteq \mathsf{P}^{\mathrm{MK}}$, where MK stands for Max Knapsack, moreover it is well-known that Max Knapsack belongs to PTAS, and then to APX [86]. If A is a $\leq$-maximum in APX then Max Knapsack $\leq A$, so that $\mathsf{P}^{\mathsf{NP}} \subseteq \mathsf{P}^{\mathrm{MK}_1} \subseteq \mathsf{P}^{A}$ (since $\leq$ is optimum preserving). We will now show that $\mathsf{P}^{A} \subseteq \mathsf{P}^{\mathsf{NP}[O(\log n)]}$.

Let L be a language in P^A and let f and g be the functions witnessing that $L \in \mathsf{P}^A$. Since A is polynomially bounded and f is computable in polynomial time, then a polynomial q exists such that, for any x, $\mathsf{opt}(f(x)) \leq q(|x|)$. By applying the binary search technique, it is easy to see that, for any x, $\mathsf{opt}(f(x))$ is computable in polynomial time by asking at most $\lceil \log q(|x|) \rceil$ queries to an NP-complete oracle. Moreover, we have that, for any x, x belongs to L if and only if a solution y for $f(x)$ exists such that $\mathsf{m}(f(x), y) = \mathsf{opt}(f(x))$ and $g(x,y) = 1$. That is, given the value of $\mathsf{opt}(f(x))$, deciding whether $x \in L$ is an NP problem, and then $L \in \mathsf{P}^{\mathsf{NP}[\lceil \log q(n) \rceil + 1]}$.

In conclusion, $\mathsf{P}^{\mathsf{NP}} \subseteq \mathsf{P}^{A} \subseteq \mathsf{P}^{\mathsf{NP}[O(\log n)]}$.

•

Corollary 3.11 Max Cut, Max Sat, Min Δ-TSP, Min Vertex Cover *are not* APX*-complete with respect to the E-reducibility, nor with respect to the P-reducibility, unless $\mathsf{P}^{\mathsf{NP}} = \mathsf{P}^{\mathsf{NP}[O(\log n)]}$.*

The possibility that $\mathsf{P}^{\mathsf{NP}} = \mathsf{P}^{\mathsf{NP}[O(\log n)]}$ seems to be quite unlikely and, even if this event is currently not yet related to any other collapse in structural complexity theory, it is conjectured that it implies the collapse of the polynomial-time hierarchy [81, 133].

3.1.3 The AP-reducibility

The above mentioned drawback of the E-reducibility is due to the fact that an E-reduction preserves optimum values. Indeed, the linear relation between the performance ratios seems to be too restrictive. To overcome this problem it is not sufficient to express this relation by means of an implication, as in the P-reducibility. Intuitively, since the function g *does not know* which approximation is required, it must still map optimum solutions into optimum solutions.

The new approach that we shall take here consists of letting the functions f and g *depend on the performance ratio* the reduction is expected to preserve. There are at least two ways of looking at this new approach. On the one hand, a reduction of this kind is actually a *reduction scheme*, since a (possibly different) reduction is specified for any value of the performance ratio. On the other hand, it is *non-oblivious* in the sense that it *possess* and *uses* one more relevant piece of information.

There are two different constraints that have to be put on the computation time of f and g: we want the reduction to preserve membership in PTAS (so the computation time better be polynomial for fixed values of the performance ratio), and we also want the reduction to be efficient even when poor performance ratios are required (to preserve membership in logAPX and polyAPX), thus the computation time should not increase when the performance ratio decreases. Such considerations call for a generalization of the notion of reduction template.

Definition 3.12 (Extended reduction template) *Let A and B be two* NPO *problem, an extended* reduction template (f, g) *between them is an ordered 2-tuple of functions such that the following properties hold.*

1. *For any $x \in I_A$, and for any rational $r > 1$, $f(x, r) \in I_B$, and is computable in time $t_f(|x|, r)$. Moreover, if $\mathsf{sol}(x) \neq \emptyset$ then also $\mathsf{sol}(f(x, r)) \neq \emptyset$.*
2. *For any $y' \in \mathsf{sol}(f(x, r))$, $g(x, y', r) \in \mathsf{sol}(x)$ is computable in time $t_g(|x|, |y|, r)$.*
3. *For any fixed r, both $t_f(\cdot, r)$ and $t_g(\cdot, \cdot, r)$ are bounded by a polynomial.*
4. *For any fixed n and m, both $t_f(n, \cdot)$ and $t_g(n, m, \cdot)$ are non-increasing functions.*

What we are doing is to add one more parameter to the functions, and ask that they are polynomial-time computable for any fixed value of the parameter. Such parameter is going to play the role of the approximation factor that we want the reduction to preserve. According to the above definition, functions like $2^{1/(r-1)}n^h$ or $n^{1/(r-1)}$ are admissible bounds on the computation time of f and g, while this is not true for functions like n^r or 2^n.

The definition of the PTAS-reducibility (respectively, the AP-reducibility) is very similar to that of the P-reducibility (respectively, the E-reducibility), the main difference being the use of extended templates.

Definition 3.13 (PTAS-reducibility) *Let A and B be two* NPO *problems. A is said to be* PTAS-reducible *to B, in symbols $A \leq_{\text{PTAS}} B$, if an extended template (f, g) between A and B and a computable function $c : (1, \infty) \cap \mathcal{Q}^+ \to (1, \infty) \cap \mathcal{Q}^+$ exist such that, for any $x \in I_A$, for any $y \in \mathsf{sol}_B(f(x))$, for any rational $r > 1$,*

$$R_B(f(x, r), y) \leq c(r) \text{ implies } R_A(x, g(x, y, r)) \leq r \ .$$

The triple (f, g, c) is said to be a PTAS-reduction from A to B.

Definition 3.14 (AP-reducibility) *Let A and B be two* NPO *problems. A is said to be* AP-reducible *to B, in symbols $A \leq_{\text{AP}} B$, if an extended template (f, g) between A and B and a positive constant α exist such that, for any $x \in I_A$, for any rational $r > 1$, and for any $y \in \mathsf{sol}_B(f(x, r))$,*

$$R_B(f(x, r), y) \leq r \text{ implies } R_A(x, g(x, y, r)) \leq 1 + \alpha(r - 1).$$

The triple (f, g, α) is said to be an AP-reduction from A to B.

Sometimes we will call (f, g, α) an *α-AP-reduction* from A to B, and we shall write it $A \leq^{\alpha}_{\text{AP}} B$

Observe that, clearly, the AP-reducibility is a generalization of the E-reducibility. Moreover, it is easy to see that, contrary to the E-reducibility, any PTAS problem is AP-reducible to any NPO problem.

As far as we know, this reducibility is the strictest one appearing in the literature that allows to obtain natural APX-completeness results (for instance, the APX-completeness of Max Sat).

3.2 Completeness

3.2.1 NPO-complete problems

In this section, we will prove that there are natural problems that are complete for the classes NPO and NPO PB. Previously, completeness results have been obtained just for Max NPO, Min NPO, Max PB, and Min PB [47, 118, 29, 93]. One example of such a result is the following theorem.

Theorem 3.15 MIN ONES SAT *is* Min NPO*-complete and* MAX ONES SAT *is* Max NPO*-complete, even if only a subset* $\{v_1, \dots, v_s\}$ *of the variables has nonzero weight* $w(v_i) = 2^{s-i}$ *and any truth assignment satisfying the instance gives the value true to at least one* v_i*.*

We will construct AP-reductions from maximization problems to minimization problems and vice versa. Using these reductions we will show that a problem that is Max NPO-complete or Min NPO-complete in fact is complete for the whole of NPO, and that a problem that is Max PB-complete or Min PB-complete is complete for the whole of NPO PB.

Theorem 3.16 MIN ONES SAT *and* MAX ONES SAT *are* NPO*-complete.*

PROOF: In order to establish the NPO-completeness of MIN ONES SAT we just have to show that there is an AP-reduction from a Max NPO-complete problem to MIN ONES SAT. As the Max NPO-complete problem we will use the restricted version of MAX ONES SAT from Theorem 3.15.

Let x be an instance of MAX ONES SAT, i.e. a formula φ over variables $v_1, \dots, v_s$ with weights $w(v_i) = 2^{s-i}$ and some variables with weight zero. We will first give a simple reduction that preserves the approximability within the factor 2, and then adjust it to obtain an AP-reduction.

Let $f(x)$ be the formula $\varphi \wedge \alpha_1 \wedge \cdots \wedge \alpha_s$ where α_i is the conjunctive normal form of $(z_i \equiv (\overline{v}_1 \wedge \cdots \wedge \overline{v}_{i-1} \wedge v_i))$, where $z_1, \dots, z_s$ are new variables with weights $w(z_i) = 2^i$ and where all other variables (even the v-variables) have zero weight. If y is a satisfying assignment of $f(x)$, let $g(x, y)$ be the restriction of the assignment to the variables that occur in φ. This assignment clearly satisfies φ.

Note that exactly one of the z-variables is true in any satisfying assignment of $f(x)$. Indeed, if all z-variables were false, then all v-variables would be false and φ would not be satisfied. On the other hand, if both z_i and z_j were true with $j > i$, then v_i would be both true and false which is a contradiction. Hence,

$$\begin{aligned}
m(f(x), y) = 2^i \quad &\Leftrightarrow \quad z_i = 1 \\
&\Leftrightarrow \quad v_1 = v_2 = \cdots = v_{i-1} = 0, v_i = 1 \\
&\Leftrightarrow \quad 2^{s-i} \le \mathsf{m}(x, g(x, y)) < 2 \cdot 2^{s-i} \\
&\Leftrightarrow \quad \frac{2^s}{\mathsf{m}(f(x), y)} \le \mathsf{m}(x, g(x, y)) < 2\frac{2^s}{\mathsf{m}(f(x), y)}.
\end{aligned}$$

In particular this holds for the optimum solution. Thus the performance ratio for MAX ONES SAT is

$$R(x, g(x, y)) = \frac{\mathsf{opt}(x)}{\mathsf{m}(x, g(x, y))} < \frac{2\dfrac{2^s}{\mathsf{opt}(f(x))}}{\dfrac{2^s}{\mathsf{m}(f(x), y)}} = 2\frac{\mathsf{m}(f(x), y)}{\mathsf{opt}(f(x))} = 2R(f(x), y),$$

which means that the reduction preserves the approximability within 2.

Let us now extend the construction in order to obtain $R(x, g(x, y)) \le (1 + 2^{-k})R(f_k(x), y)$ for every nonnegative integer k. The reduction described above corresponds to $k = 0$.

For any $i \in \{1, \ldots, s\}$ and for any $(b_1, \ldots, b_{k(i)}) \in \{0, 1\}^{k(i)}$ where $k(i) = \min\{s - i, k\}$, we have a variable $z_{i,b_1,\ldots,b_{k(i)}}$. Let

$$f_k(x) = \varphi \wedge \bigwedge_{\substack{i \in \{1,\ldots,s\} \\ (b_1,\ldots,b_{k(i)}) \in \{0,1\}^{k(i)}}} \alpha_{i,b_1,\ldots,b_{k(i)}},$$

where $\alpha_{i,b_1,\ldots,b_{k(i)}}$ is the conjuctive normal form of

$$\left(z_{i,b_1,\ldots,b_{k(i)}} \equiv \left(\overline{v}_1 \wedge \cdots \wedge \overline{v}_{i-1} \wedge v_i \wedge (v_{i+1} = b_1) \wedge \cdots \wedge (v_{i+k(i)} = b_{k(i)})\right)\right).$$

Define $g(x, y)$ as above. Finally, define

$$w(z_{i,b_1,\ldots,b_{k(i)}}) = \left\lceil \frac{K \cdot 2^s}{w(v_i) + \sum_{j=1}^{k(i)} b_j w(v_{i+j})} \right\rceil = \left\lceil \frac{K \cdot 2^i}{1 + \sum_{j=1}^{k(i)} b_j 2^{-j}} \right\rceil$$

(by choosing K greater than 2^k we can disregard the effect of the ceiling operation in the following computations).

As in the previous reduction exactly one of the z-variables is true in any satisfying assignment of $f_k(x)$. If, in a solution y of $f_k(x)$, $z_{i,b_1,\ldots,b_{k(i)}} = 1$, then we have $\mathsf{m}(f_k(x), y) = w(z_{i,b_1,\ldots,b_{k(i)}})$ and we know that

$$\mathsf{m}(x, g(x, y)) \ge w(v_i) + \sum_{j=1}^{k(i)} b_j w(v_{i+j}) = 2^{s-i}(1 + \sum_{j=1}^{k(i)} b_j 2^{-j}).$$

On the other hand, if $k(i) = s - i$ then $\mathsf{m}(x, g(x, y)) = 2^{s-i}(1 + \sum_{j=1}^{k(i)} b_j 2^{-j})$, otherwise

$$\begin{aligned} \mathsf{m}(x, g(x, y)) &\le w(v_i) + \sum_{j=1}^{k} b_j w(v_{i+j}) + \sum_{j=k+i+1}^{s} w(v_j) \\ &< 2^{s-i}(1 + \sum_{j=1}^{k} b_j 2^{-j})(1 + 2^{-k}) . \end{aligned}$$

In both cases, we thus get

$$\frac{K \cdot 2^s}{\mathsf{m}(f_k(x), y)} \le \mathsf{m}(x, g(x, y)) < \frac{K \cdot 2^s}{\mathsf{m}(f_k(x), y)}(1 + 2^{-k})$$

and therefore $R(x, g(x, y)) < (1 + 2^{-k})R(f_k(x), y)$. Given any $r > 1$, if we choose k such that $2^{-k} \le (r - 1)/r$, e.g. $k = \lceil \log r - \log(r - 1) \rceil$, then $R(f_k(x), y) \le r$ implies $R(x, g(x, y)) < (1 + 2^{-k})R(f_k(x), y) \le r + r2^{-k} \le r + r - 1 = 1 + 2(r - 1)$. This is obviously an AP-reduction with $\alpha = 2$.

A very similar proof can be used to show that MAX ONES SAT is NPO-complete.

•

Corollary 3.17 *Any* Min NPO*-complete problem is* NPO*-complete and any* Max NPO*-complete problem is* NPO*-complete.*

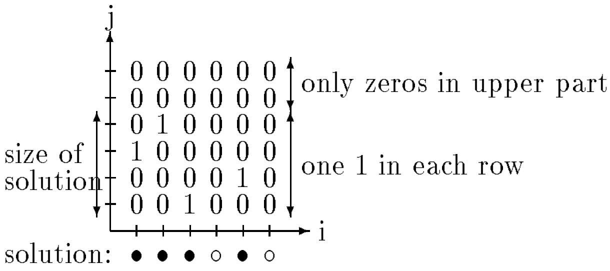


Figure 3.1: The idea of the reduction from MINIMUM/MAX PB $0-1$ PROGRAMMING to MAXIMUM/MIN PB $0-1$ PROGRAMMING. The variable $x_i^j = 1$ if and only if v_i is the jth one-variable in the solution. There is at most one 1 in each column and in each row.

3.2.2 NPO PB-complete problems

We can also show that there are natural complete problems for the class of polynomially bounded NPO problems.

Theorem 3.18 MAX PB $0-1$ PROGRAMMING *and* MIN PB $0-1$ PROGRAMMING *are* NPO PB-*complete.*

PROOF: MAX PB $0-1$ PROGRAMMING is known to be Max PB-complete [29] and MIN PB $0-1$ PROGRAMMING is known to be Min PB-complete [93]. Thus we just have to show that there are AP-reductions from MIN PB $0-1$ PROGRAMMING to MAX PB $0-1$ PROGRAMMING and from MAX PB $0-1$ PROGRAMMING to MIN PB $0-1$ PROGRAMMING.

Both reductions use exactly the same construction. Given a satisfying variable assignment, we define the *one-variables* to be the variables occurring in the objective function that have the value one. The objective value is the number of one-variables plus 1.

The objective value of a solution is encoded by introducing an order of the one-variables. The order is encoded by a squared number of $0-1$ variables, see Fig. 3.1. The idea is to invert the objective values, so that a solution without one-variables corresponds to an objective value of n of the constructed problem, and, in general, a solution with p one-variables corresponds to an objective value of $\left\lfloor \frac{n}{p+1} \right\rfloor$.

The reductions are constructed as follows. Given an instance of MIN PB $0-1$ PROGRAMMING or MAX PB $0-1$ PROGRAMMING, i.e. an objective function $1 + \sum_{k=1}^{m} v_k$ and some inequalities over variables $V = \{v_1, \ldots, v_m\} \cup U$, construct m^2 variables x_i^j, $1 \le i, j \le m$, and the following inequalities:

$$\forall i \in [1..m] \qquad \sum_{j=1}^{m} x_i^j \le 1 \qquad \text{(at most one 1 in each column)} \tag{3.1}$$

$$\forall j \in [1..m] \qquad \sum_{i=1}^{m} x_i^j \le 1 \qquad \text{(at most one 1 in each row)} \tag{3.2}$$

$$\forall j \in [1..m-1] \qquad \sum_{i=1}^{m} x_i^j - \sum_{i=1}^{m} x_i^{j+1} \ge 0 \qquad \text{(only zeros in upper part)} \tag{3.3}$$

Besides these inequalities we include all inequalities from the original problem, but we substitute each variable v_i with the sum $\sum_{k=1}^{m} x_i^k$. The variables in U (that do not occur in the objective function) are left intact.

The objective function is defined as

$$n - \sum_{p=1}^{m}\left(\left\lfloor\frac{n}{p}\right\rfloor - \left\lfloor\frac{n}{p+1}\right\rfloor\right)\sum_{i=1}^{m} x_i^p. \tag{3.4}$$

In order to express the objective function with only binary coefficients we have to introduce n new variables $y_1, \ldots, y_n$ where $y_j = 1 - \sum_{i=1}^{m} x_i^p$ for $\lfloor n/(p+1)\rfloor < j \leq \lfloor n/p \rfloor$ and $y_j = 1$ for $j \leq \lfloor n/(m+1)\rfloor$. The objective function then is $\sum_{j=1}^{n} y_j$. One can now verify that a solution of the original problem instance with s one-variables (i.e. with an objective value of $s+1$) will exactly correspond to a solution of the constructed problem instance with objective value $\lfloor n/(s+1)\rfloor$ and vice versa.

Suppose that the optimum solution to the original problem instance has M one-variables, then the performance ratio $(s+1)/(M+1)$ will correspond to the performance ratio

$$\frac{\left\lfloor\dfrac{n}{M+1}\right\rfloor}{\left\lfloor\dfrac{n}{s+1}\right\rfloor} = \frac{s+1}{M+1}\left(1 \pm \frac{m}{n}\right)$$

for the constructed problem, where $\frac{m}{n}$ is the relative error due to the floor operation. By choosing n large enough the relative error can be made arbitrarily small. Thus it is easy to see that the reduction is an AP-reduction. •

Corollary 3.19 *Any* Min PB*-complete problem is* NPO PB*-complete and any* Max PB*-complete problem is* NPO PB*-complete.*

3.2.3 APX- and FAPX-complete problems

In this section we will prove that Max 3Sat is APX-complete. To this aim, we first quote the following result from [102] that follows from the proof of the PCP theorem.

Theorem 3.20 ([9, 102]) *A constant $\varepsilon > 0$ and two polynomial-time computable functions f_s and g_s exist such that, for any Boolean formula φ in conjunctive normal form, $\psi = f_s(\varphi)$ is an instance of* Max 3Sat *such that, for any truth assignment τ satisfying at least $1/(1+\varepsilon)$ of the maximum number of satisfiable clauses of ψ, $g_s(\varphi, \tau)$ satisfies φ if and only if φ is satisfiable.*

Theorem 3.21 Max 3Sat *is* APX*-complete.*

Proof: From Theorem 2.17 it follows that it suffices to prove that any maximization problem in APX is AP-reducible to Max 3Sat.

Let A be a maximization in APX and let T_A be an r_A-approximate algorithm for A. Our goal is to define an AP-reduction (f, g, α) from A to Max 3Sat. First of all, we define

$$\alpha = 2(r_A \log r_A + r_A - 1)\frac{1+\varepsilon}{\varepsilon}$$

where ε is the constant of Theorem 3.20. In order to define functions f and g, for any r, we distinguish two cases.

1. $1 + \alpha(r-1) \geq r_A$. In this case, for any x, we simply define

$$g(x, \cdot, r) = T_A(x)$$

   and the AP-condition is clearly satisfied.

2. $r_n = 1 + \alpha(r-1) > r_A$. In this case, we have that

$$r = \frac{r_n - 1}{\alpha} + 1 = \frac{\varepsilon}{2(1+\varepsilon)} \frac{r_n - 1}{r_A \log r_A + r_A - 1} + 1 \leq \frac{\varepsilon}{2k(1+\varepsilon)} + 1 \tag{3.5}$$

where $k = \lceil \log_{r_n} r_A \rceil$ and the last inequality is due to the fact that

$$k \leq \frac{\log r_A}{\log r_n} + 1 \leq \frac{r_n \log r_A}{r_n - 1} + 1 \leq \frac{r_A \log r_A + r_A - 1}{r_n - 1}.$$

Let us partition the interval $[t_A(x), r_A t_A(x)]$ where $t_A(x) = m_A(x, T_A(x))$ into the following k subintervals:

$$[t_A(x), r_n t_A(x)], [r_n t_A(x), r_n^2 t_A(x)], \ldots, [r_n^{k-1} t_A(x), r_A t_A(x)].$$

Observe that $\mathsf{opt}_A(x) \leq r_A t_A(x) \leq r_n^k t_A(x)$, that is, the optimum value for the input x belongs to one of the above subintervals. For any $i = 0, \ldots, k-1$, let us consider the NP problem of deciding whether $\mathsf{opt}_A(x) \geq r_n^i t_A(x)$. Using the proof of Cook's Theorem [44], we can find a boolean formula φ_i such that given a truth assignment τ_i that satisfies φ_i we can compute in polynomial time a solution $y \in \mathsf{sol}_A(x)$ such that $\mathsf{m}_A(x, y) \geq r_n^i t_A(x)$. We then define the instance ψ of MAX 3SAT as follows:

$$\psi = f(x, r) = f_s(\varphi_0) \wedge \ldots \wedge f_s(\varphi_{k-1})$$

where f_s is the function deriving from Theorem 3.20. Without loss of generality, we may assume that each $f_s(\varphi_i)$ contains the same number m of clauses.

Let τ be any truth assignment to the variables of ψ whose performance ratio is at most r. Then

$$\mathsf{opt}(\psi) - \mathsf{m}(\psi, \tau) \leq \mathsf{opt}(\psi) \frac{r-1}{r} \leq km \frac{r-1}{r}.$$

On the other hand, if r_i denotes the performance ratio of τ with respect to $f_s(\varphi_i)$, we have that

$$\begin{aligned} \mathsf{opt}(\psi) - \mathsf{m}(\psi, \tau) &\geq \mathsf{opt}(f_s(\varphi_i)) - \mathsf{m}(f_s(\varphi_i), \tau) \\ &= \mathsf{opt}(f_s(\varphi_i)) \frac{r_i - 1}{r_i} \\ &\geq \frac{m}{2} \frac{r_i - 1}{r_i} \end{aligned}$$

where the last inequality is due to the fact that at least half clauses of a Boolean formula are always satisfiable.

We then have that, for $i = 0, \ldots, k-1$,

$$\frac{m}{2} \frac{r_i - 1}{r_i} \leq km \frac{r-1}{r},$$

that is,

$$1 - 2k\frac{r-1}{r} \leq \frac{1}{r_i}.$$

By making use of inequality 3.5 and by simple algebraic manipulations, we obtain that

$$r_i \leq 1 + \varepsilon.$$

From the observation preceding the theorem it follows that, for $i = 0, \ldots, k-1$, $\tau_i = g_s(\varphi_i, \tau)$ satisfies φ_i if and only if φ_i is satisfiable. Let j be the maximum i such that τ_i satisfies φ_i (observe that such a j must exist). Then

$$r_n^j t_A(x) \leq \mathsf{opt}_A(x) \leq r_n^{j+1} t_A(x)$$

and from τ_j we can derive a solution y whose measure is at least $r_n^j t_A(x)$, that is, y is r_n-approximate. In other words, the AP-condition turns out to be satisfied.

Since A was an arbitrary maximization problem in APX, it follows that Max 3Sat is APX-complete. •

We proved the APX-completeness of Max 3Sat from scratch to demonstrate the power of the AP-reducibility: the above proof is slightly easier then the APX PB-completeness proof appeared in [102]. Khanna et al. [102] partition the interval $[t_A(x), r_A t_A(x)]$ into a polynomial (instead that constant) number of subintervals, and then a few calculations are needed to show that an approximation for the Max 3Sat instance yields an approximation for problem A.

Alternatively, we can use the KMSV results as a black box, and then show how to remove the polynomially bounded restriction.

Theorem 3.22 *Let A be an $r(n)$-approximable NPO-problem. Then an NPO problem B exists with the following properties.*

1. *$A \leq_{\mathrm{AP}} B$.*
2. *B is $r(n)$-approximable.*
3. *For any $x \in I_B$, for any $y \in \mathsf{sol}(x)$, $\mathsf{m}(x, y) \leq |x| r(|x|)$.*

Proof: Assume A is a maximization problem. Let T_A be an $r(n)$-approximate algorithm for A; for an instance $x \in I_A$ let $t_A(x) = \mathsf{m}_A(x, T_A(x))$ be the measure of the $r(n)$-approximate solution found by T_A. We define a maximization problem B as follows: $I_B = I_A$, for any $x \in I_B$, we let $\mathsf{sol}_B(x) = \mathsf{sol}_A(x)$, for an instance x of size n and a feasible solution y, the measure is

$$\mathsf{m}_B(x, y) = \left\lfloor n \frac{\max\{\mathsf{m}_A(x, y), t_A(x)\}}{t_A(x)} \right\rfloor .$$

Note that for any $x \in I_B$, $y \in \mathsf{sol}_B(x)$, we have $\mathsf{m}_B(x, y) \leq n\frac{\max\{\mathsf{m}_A(x,y), t_A(x)\}}{t_A(x)} \leq nr(n)$. In particular, $\mathsf{opt}_B(x) = \left\lfloor n\frac{\mathsf{opt}_A(x)}{T_A(x)} \right\rfloor$. Besides being an $r(n)$-approximate algorithm for A, T_A is also an $r(n)$-approximate algorithm for B.

$$R_B(x, T_A(x)) = \frac{\left\lfloor n\frac{\mathsf{opt}_A(x)}{t_A(x)} \right\rfloor}{n} \leq \frac{\mathsf{opt}_A(x)}{t_A(x)} \leq r(n)$$

Consider any instance $x \in I_A$ and solution $y \in \mathsf{sol}_A(x)$ such that $\mathsf{m}_A(x,y) \geq t_A(x)$, we now prove that $R_A(x,y)$ cannot be much larger than $R_B(x,y)$.

$$\begin{aligned}
R_B(x,y) &= \frac{\mathsf{opt}_B(x)}{m_B(x)} \\
&= \frac{\left\lfloor n\frac{\mathsf{opt}_A(x)}{t_A(x)}\right\rfloor}{\left\lfloor n\frac{\mathsf{m}_A(x,y)}{t_A(x)}\right\rfloor} \\
&\geq \frac{n\frac{\mathsf{opt}_A(x,y)}{t_A(x)} - 1}{n\frac{\mathsf{m}_A(x)}{t_A(x)}} \\
&= \frac{\mathsf{opt}_A(x,y)}{\mathsf{m}_A(x)} - \frac{t_A(x)}{n\mathsf{m}_A(x)} \\
&\geq R_A(x,y) - \frac{1}{n}
\end{aligned}$$

If A is a minimization problem, we define B in a similar way, except for the measure function.

$$\mathsf{m}_B(x,y) = \left\lceil nr(n)\frac{\min\{\mathsf{m}_A(x,y), t_A(x)\}}{t_A(x)}\right\rceil \;.$$

We have that for an instance x of size n it holds $m_B(x,y) \leq |x|r(|x|)$ and $\mathsf{opt}_B(x) = \left\lceil nr(n)\frac{\mathsf{opt}_A(x)}{t_A(x)}\right\rceil$. As before, T_A is an $r(n)$-approximate algorithm for B, since

$$R_B(x, T_A(x)) = \frac{nr(n)}{\left\lceil nr(n)\frac{\mathsf{opt}_A(x)}{t_A(x)}\right\rceil} \leq \frac{t_A(x)}{\mathsf{opt}_A(x)} \leq r(n)$$

Let now $x \in I_A$ and let $y \in \mathsf{sol}_A(x)$ be a solution such that $m_A(x,y) \leq t_A(x)$, then we have

$$\begin{aligned}
R_B(x,y) &= \frac{m_B(x)}{\mathsf{opt}_B(x)} \\
&= \frac{\left\lceil nr(n)\frac{\mathsf{m}_A(x,y)}{T_A(x)}\right\rceil}{\left\lceil nr(n)\frac{\mathsf{opt}(x)}{T_A(x)}\right\rceil} \\
&\geq \frac{nr(n)\frac{\mathsf{m}_A(x)}{t_A(x)}}{nr(n)\frac{\mathsf{opt}_A(x)}{t_A(x)} - 1} \\
&= \frac{\mathsf{m}_A(x)}{\mathsf{opt}_A(x,y)}\left(\frac{1}{1 - \frac{t_A(x)}{nr(n)\mathsf{opt}_A(x,y)}}\right) \\
&\geq R_A(x,y)\left(\frac{1}{1 - \frac{1}{nr(n)}}\right) \\
&= R_A(x,y)\left(1 + \frac{1}{nr(n) - 1}\right) \\
&\geq R_A(x,y) + \frac{1}{n-1}
\end{aligned}$$

We now define the AP-reduction between A and B.

1. For any $r > 1$, $f(x, r) = x$.

2. For any $x \in I_A = I_B$, for any $y \in \mathsf{sol}_B(x)$ and for any $r > 1$,

$$g(x, y, r) = \begin{cases} y' & \text{if } r > 1 + 1/(n-1), \\ y^* & \text{otherwise} \end{cases}$$

   where $y^* \in \mathsf{sol}_B^*(x)$, while $y' = y$ if $R_A(x, y) \leq R_A(x, T_A(x))$, and $y' = T_A(x)$ otherwise.

3. $\alpha = 2$.

Observe that, according to the definition of the AP-reducibility, the running time of g can be exponential in $1/(r-1)$. If $r \leq 1 + 1/(n-1)$ then g has enough time to compute y^* so that, in this case, the AP-conditions are satisfied.

Assume now that $r > 1 + 1/(n-1)$, and let x be an instance and y be a solution such that $R_B(x, y) \leq r$. Let also y' be as defined above. Then,

$$R_B(x, y') \leq r + \frac{1}{n-1} \leq r + (r-1) = 1 + 2(r-1) \ .$$

Even in this cases, the AP conditions are satisfied, and this concludes the proof. •

Corollary 3.23 *Let $\mathcal{C} \in \{\mathsf{APX}, \log\mathsf{APX}, \mathrm{poly}\mathsf{APX}\}$, let A be $\mathcal{C}$ PB-complete under E-reductions. Then A is $\mathcal{C}$-complete under AP-reductions.*

Khanna et al. [102] proved that Max Clique and Min Node Coloring are polyAPX PB-complete under E-reductions. From the reduction technique of Khanna et al. [102] and the recent hardness result for Min Set Cover by Raz and Safra [126] (see also a similar independent result by Arora and Sudan [11]), it follows that Min Set Cover, Min Dominating Set and Min Hitting Set are logAPX PB-complete under E-reductions.

Using Corollary 3.23, we have that Max Clique and Min Node Coloring are polyAPX-complete under AP-reductions and that Min Set Cover, Min Dominating Set, and Min Hitting Set are logAPX-complete under AP-reductions.

3.3 A PCP-free Completeness Result

In this section we shall prove that Max 3Sat is $\mathsf{Max\ NP}$-complete without relying on (consequences of) the PCP Theorem. Our reduction will turn out to be efficient enough to have algorithmic applications.

3.3.1 Shrinking Disjunctions

We illustrate our reduction technique by presenting an approximation-preserving reduction from Max Sat to Max kSat. The actual value of k depends on the approximation factor that one wants to preserve.

Lemma 3.24 *Let $p \in (0, 1/2)$, $k \in \mathbf{Z}^+$, $r, r' \geq 1$ be such that*

$$r \geq \max\{1/(1-(1-p)^k)\ ,\ r'/(1-p)\}\ .$$

Then r-approximating Max Sat *is reducible to r'-approximating* Max kSat.

PROOF: Let φ be an instance of MAX SAT with m clauses, m_k be the number of clauses of φ with k or less literals, and φ_k be the instance of MAX kSAT containing those m_k clauses. Define $m_l = m - m_k$ and $\varphi_l = \varphi - \varphi_k$. Let τ_k be an r'-approximate solution for φ_k, let a be the number of clauses satisfied by τ_k. It is immediate to verify that

$$\mathsf{opt}(\varphi) \leq r'a + m_l \ . \tag{3.6}$$

Indeed, any assignment cannot satisfy more than $r'a$ clauses in φ_k (otherwise τ_k would not be r'-approximate) and more then m_l clauses in φ_l. We now define a random assignment τ_R over the variables of φ: for any variable x,

$$\mathbf{Pr}[\tau_R(x) \neq \tau_k(x)] = p \quad \mathbf{Pr}[\tau_R(x) = \tau_k(x)] = 1 - p \ .$$

If a variable x occurs in φ but not in φ_k, then we assign to it 1 of 0 with probability 1/2. Let us now estimate the average number of clauses of φ that are satisfied by τ_R. Any literal is true with probability at least p, thus the probability that a clause in φ_l is contradicted is at most $(1-p)^{k+1}$. On the other hand, if a clause is satisfied by τ_k, then there is a probability at least $(1-p)$ that it is still satisfied by τ_R. We can thus infer a lower bound on the average number of clauses of φ that are satisfied by τ_R.

$$\mathbf{E}[\mathsf{m}(\varphi, \tau_R)] \geq (1-p)a + (1 - (1-p)^{k+1})m_l \geq (1/r)\mathsf{opt}(\varphi)$$

where the last inequality is due to the bound (3.6). Using the method of conditional expectation [4] (see also [135]) we can find in linear time an assignment τ such that $\mathsf{m}(\varphi, \tau) \geq \mathsf{m}(\varphi, \tau_R)$. The performance ratio of τ is at least r. •

Lemma 3.25 *For any constant $B \in \mathbf{Z}^+$, $r > 1$, r-approximating* MAX GSAT-B *is reducible to r'-approximating* MAX Bk-CSP *provided a constant $p \in (0, 1/2)$ exists such that*

$$r \geq \max\left\{\frac{1}{1-(1-p^B)^{k+1}} \ , \ \frac{r'}{(1-p)^B}\right\}$$

PROOF: Identical to the proof of Lemma 3.24, with "literal" substituted by "B-ary conjunct". A MAX GSAT-B instance where all clauses are the disjunction of at most k conjuncts is clearly an instance of MAX Bk-CSP. We give details of the computation of the performance ratio of τ_R. If all literals are satisfied with probability at least p, then a B-ary conjunct is contradicted with probability at most $(1 - p^B)$. If a conjunct was satisfied by τ, then it is satisfied by τ_R with probability at least $(1-p)^B$. The average number of clauses satisfied by τ_R is thus

$$\mathbf{E}[\mathsf{m}(\varphi, \tau_R)] \geq (1-p)^B a + (1 - (1-p^B)^{k+1})m_l \geq (1/r)\mathsf{opt}(\varphi) \ .$$

Derandomization can still be done using conditional expectation. •

3.3.2 The Max NP-completeness Result

Theorem 3.26 MAX 3SAT *is* Max NP*-complete under PTAS-reductions.*

PROOF: For any B, we give a PTAS-reduction from MAX GSAT-B to MAX 3SAT: this reduction will be the composition of the reduction from MAX GSAT-B to MAX kB-CSP given in Lemma 3.25 and of the standard reduction from MAX kB-CSP to MAX 3SAT [123]. In the first reduction, k is not fixed but depends on the approximation factor that we want to preserve. This is not in contradiction with the definition of PTAS-reduction. For any k, the MAX k-CSP problem is in

Max SNP, and thus a PTAS-reduction (f_k, g_k, c_k) exists from MAX k-CSP to MAX 3SAT. An inspection of the results of [123] ensures that actually a single triple of functions (f, g, c) exist such that for any k $(f(\cdot,\cdot,k), g(\cdot,\cdot,\cdot,k), c(\cdot,k))$ is a PTAS-reduction. Let φ be an instance of MAX GSAT-B, $r > 1$. We compute values k, r', p such that

$$r \geq \max\left\{\frac{1}{1-(1-p^B)^{k+1}}, \frac{r'}{(1-p)^B}\right\}.$$

We then compute φ_k (the subset of φ containing clauses that are disjunctions of at most k terms) that is an instance of MAX kB-CSP. From an r'-approximate solution for φ_k we are able to reconstruct an r-approximate solution for φ. The former problem can in turn be reduced to find a $c(r', kB)$-approximate solution for the instance $f(\varphi_k, r', kB)$ of MAX 3SAT. •

We will now see that, in order to prove the APX-completeness of MAX 3SAT, at least a weaker version of the PCP Theorem is necessary. In the following we say that an algorithm T is a *non-constructive PTAS* for a Max NPO problem A if the following properties hold.

1. For an instance x of A and a rational $r > 1$, $a = T(x, r)$ is a real number with the property that $\mathsf{opt}(x)/r \leq a \leq \mathsf{opt}(x)$.

2. For fixed $r > 1$, the running time of $T(x, r)$ is polynomial in $|x|$.

Non-constructive approximation for minimization problems is defined similarly. Note that if a problem is in PTAS then it admits a non-constructive PTAS, but the converse is not necessarily true (see [50]).

Lemma 3.27 $\mathsf{PCP}(\log n, 1) = \mathsf{P}$ *if and only if* MAX 3SAT *admits a non-constructive PTAS.*

PROOF: The "if" part is a restating of the standard reduction from PCP verifiers to MAX 3SAT (see [9]).

The other direction is more interesting. We use ideas from [50]. For any $1/2 \leq s < c \leq 1$ let GAP 3SAT$_{c,s}$ be the following promise problem: given a 3SAT formula φ with m clauses, reject if $\mathsf{opt}(\varphi) \leq sm$ and accept if $\mathsf{opt}(\varphi) \geq cm$. This problem is in $\mathsf{PCP}_{c,s}[\log, 3]$ (a proof is an assignment, the verifier picks a random clause and accepts iff the clause is satisfied by the assignment) and thus in $\mathsf{PCP}(\log, 1) = \mathsf{P}$. Let $T_{c,s}$ denote a polynomial-time Turing machine that on input φ accepts whenever $\mathsf{opt}(\varphi) \geq cm$, rejects whenever $\mathsf{opt}(\varphi) \leq sm$, and whose behaviour is undefined otherwise. Let $r > 1$ be fixed, we describe a non-constructive r-approximate algorithm for MAX 3SAT. Let φ be a 3SAT formula with m variables; its optimum lies somewhere between $m/2$ and m. We divide this interval into $k = \lceil 2/\log r \rceil$ subintervals, where the ith interval (for $i = 0, \ldots, k-1$) is $[m\alpha_i, m\alpha_{i+1})$, with $\alpha_i = .5r^{i/2}$. For any i, we run $T_{\alpha_{i+1},\alpha_i}(\varphi)$. Let j be the largest index such that $T_{\alpha_{j+1},\alpha_j}(\varphi)$ accepts. From the definition of $T_{c,s}$ it follows that $\mathsf{opt}(\varphi) \geq m\alpha_j$ (since $T_{\alpha_{j+1},\alpha_j}(\varphi)$ accepts) and that $\mathsf{opt}(\varphi) \leq \mathrm{m}alpha_{j+2}$ (since $T_{\alpha_{j+2},\alpha_{j+1}}(\varphi)$ rejects). It follows that $m\alpha_j$ is a non-constructive r-approximate solution for φ. •

The following result states that APX-complete problems are unlikely to have non-constructive PTAS's[1].

[1] Recall that the NP-complete problem PARTITION is defined as follows: given a set U of items and a size function $s : U \to Q \cap (0, 1]$, does there exists a subset $U' \subseteq U$ such that

$$\sum_{u \in U'} s(u) = \sum_{u \notin U'} s(u)?$$

Lemma 3.28 *If an* APX*-complete problem A admits a non-constructive PTAS, then* NP = co-NP.

PROOF: Let (f, g, c) be a PTAS-reduction from MIN BIN PACKING to A; let $r = c(1.4)$. Let I be an instance of MIN BIN PACKING such that the total size of the elements is 2; it is easy to see that the optimum packing of such instance will use either 2 or 3 bins. Distinguishing between those two cases is NP-hard (it is a restatement of the PARTITION problem). Let $x = f(I, 1.4)$. From the definition of PTAS-reduction it follows that if y is an r-approximate solution for x, then $g(I, y, 1.4)$ is a 1.4-approximate solution for I (indeed, an optimum solution). An NP algorithm can compute a non-constructive r-approximate solution v for x, then guess a solution y of measure $\mathsf{m}(x, y) \geq v$, compute the solution $P = g(I, y, 1.4)$ and accept if and only if $\mathsf{m}(I, P) = 3$. It is easy to see that this is an NP algorithm for the complement of the PARTITION problem. •

The above two lemmas imply the following theorem.

Theorem 3.29 *If* MAX 3SAT *is* APX*-complete under PTAS-reductions, then* $\mathsf{PCP}(\log n, 1) \not\subseteq \mathsf{P}$ *unless* NP = co-NP.

The conclusion of Theorem 3.29 is weaker than the PCP Theorem in two ways: the verifier has not completeness 1 (i.e. it has *two-sided* error) and the existence of intractable problems in $\mathsf{PCP}(\log n, 1)$ assumes NP $\neq$ *co*-NP instead that P $\neq$ NP. None of these variations appear to make the PCP Theorem easier to be proved.

3.3.3 Algorithmic Results

In this section we will consider the *planar* restrictions of MAX SAT and MAX GSAT-B. We will use our reduction technique to develop linear-time approximation schemes for both problems. The approximation schemes we describe are a composition of the reductions of Section 3.3.1 and of the linear time PTAS's for MAX kSAT and MAX k-CSP that are implied by the techniques of [19, 85, 101].

We recall the definition of *planar instance* of a constraint satisfaction problem.

Definition 3.30 (Incidence Graph) *Let $\mathcal{F}$ be a (possibly infinite) constraint family. Let φ be an instance of* MAX WEIGHT $\mathcal{F}$ *over variable set X. The* incidence graph *of φ, denoted $I_\varphi = (V, E)$ is defined as follows:*

- *V has a v-vertex for each variable $x \in X$ and an f-vertex for any constraint C of φ.*
- *For each constraint C of φ and each variable x occurring in C there is an edge between the vertex for C and the vertex for x.*

For a constraint family $\mathcal{F}$, MAX PLANAR $\mathcal{F}$ is the restriction of MAX $\mathcal{F}$ to instances whose incidence graph is planar. We will be particularly interested in MAX PLANAR k-CSP, MAX PLANAR SAT, and MAX PLANAR GSAT-B.

Khanna and Motwani [101] prove that the latter two problems have a PTAS that computes $(1+\varepsilon)$-approximate solutions in time is $O(n^{1/\varepsilon})$. We will improve the running time to $O(n)$ (the constant hidden in the O-notation will depend on ε and, in the MAX PLANAR GSAT-B case, also on B). We use the fact that the techniques of Khanna and Motwani yield linear-time PTAS's when specialized to MAX kSAT and MAX k-CSP. We give a sketch of this latter result. We first give some definitions; we follow the exposition of [101].

Definition 3.31 (Tree Decomposition) *A* tree decomposition *of a graph $G = (V, E)$ is a 2-tuple $(\{X(i) : i \in I\}, T = (I, F))$ such that:*

- *T is a tree.*
- *For any $u \in V$, an $i \in I$ exists such that $u \in X(i)$.*
- *For any $(u, v) \in E$, an $i \in I$ exists such that $u, v \in X(i)$,*
- *For any $v \in V$, the set $\{i \in I : u \in X(i)\}$ induces a subtree of T.*

Definition 3.32 (Treewidth) *The* width *of a tree-decomposition $(X, (I, F))$ of a graph G is $\max\{|X(i)| - 1 : i \in I\}$. The* treewidth *of a graph is the minimum width over all its tree-decompositions.*

Definition 3.33 (p-Outerplanar Graphs) *A graph is* 1-outerplanar *(or simply* outerplanar *if it admits a planar embedding such that all vertices lies in the outer face. A graph is* p-outerplanar *($p \geq 2$) if it admits a planar embedding such that deleting all the vertices of the outer face yields a $(p-1)$-outerplanar graph.*

Theorem 3.34 ([34]) *A p-outerplanar graph has treewidth at most $3p - 1$.*

Theorem 3.35 ([35]) *For fixed h, given a graph of treewidth at most h, a tree-decomposition of optimum width can be found in linear time.*

Theorem 3.36 ([99]) *For fixed k and h,* Max k-CSP *restricted to instances whose incidence graph has treewidth at most h can be solved in linear time.*

Proof:[Sketch] Given an instance φ of Max k-CSP with m constraints over n variables whose incidence graph G has treewidth at most h we first find an optimum tree-decomposition of G. Then we apply divide and conquer: the root vertex of the tree-decomposition is a set of $h+1$ nodes of G that disconnect G into two components. If some of them are f-nodes that we replace them with the v-nodes corresponding to the variables occurring in them. This gives a separator S with at most $k(h+1)$ v-nodes. We try all the possible assignments to the variables of S, for any such assignment we delete the variables of S from the incidence graph (thus disconnecting it) and then we recur on the connected components of the incidence graph. Removing the root from the tree-decomposition of the incidence graph of φ gives tree-decompositions for all the connected components. The running time is given by the recursion

$$T(m, n) = 2^{k(h+1)}(T(m', n') + T(m'', n''))\ T(1, 1) = O(1) ,$$

where $n = n' + n''$ and $m = m' + m''$. The recursion solves as $T(m, n) = O(2^{k(h+1)}(m + n)) = O(m + n)$. •

Theorem 3.37 *For fixed $\varepsilon > 0$,* Max Planar k-CSP *admits an $(1 + \varepsilon)$-approximate linear time algorithm,*

Proof:[Sketch] Follows from Theorem 3.36 using the construction outlined in the first part of the proof of [101, Theorem 1] (see also [19, Theorem 1]). •

Hunt et al. [85] proved the following related result.

Theorem 3.38 *For fixed $\varepsilon > 0$,* Max Planar k-CSP *restricted to instances whose incidence graph is planar admits an $(1 + \varepsilon)$-approximate* NC *algorithm,*

Theorem 3.39 *For any fixed $\varepsilon > 0$ and $B \in \mathcal{Z}^+$,* Max Planar GSAT-B *admits a linear time and an* NC *$(1 + \varepsilon)$-approximate algorithm.*

PROOF: Let k, p, ε' be constants (depending only on ε and B) such that

$$1+\varepsilon \geq \max\left\{\frac{1}{1-(1-p^B)^{k+1}}\ ,\ \frac{1+\varepsilon'}{(1-p)^B}\right\}\ .$$

Let φ be a planar instance of MAX GSAT-B. The construction of Lemma 3.25 reduces the problem of $(1+\varepsilon)$-approximating φ to the problem of $(1+\varepsilon')$ approximating an instance φ_k of MAX kB-CSP obtained from φ by removing all the clauses with more than k conjuncts. The reduction runs in linear time, it doesn't increase the size of the instance and preserves planarity of the incidence graph. A $(1+\varepsilon')$-approximate solution for φ' can be found in linear time using Theorem 3.37. Alternatively, the reduction can be done in NC using k-wise independent distribution to do the derandomization, and a $(1+\varepsilon')$-approximate solution for φ' can be found in NC using Theorem 3.38. •

3.4 Query Complexity

The query-complexity of the "non-constructive" approximation of several NP-hard optimization problems has been studied and classified using classes of functions $\mathsf{PF}^{\mathsf{NP}[\cdot]}$ [40, 41]. However, this approach cannot be applied to analyze the complexity of "constructing" approximate solutions. To overcome this limitation, we use a novel approach that basically consists of considering how helpful is an approximation algorithm for a given optimization problem to solve decision problems. To this aim, for an NPO problem A, we consider the *approximate query hierarchy* of A, that is, the set of languages that can be solved using one query to an r-approximate algorithm for A, for some fixed $r > 1$.

Definition 3.40 *The* class $\mathsf{AQH}(A)$ *is equal to the union* $\bigcup_{r>1} \mathsf{P}^{A_r}$.

The following result states that an approximation algorithm for an APX problem does not help more than a constant number of queries to an NP-complete problem. It is worth observing that, in general, an approximate solution, even though not very helpful, cannot be computed with less than a logarithmic number of queries [39].

Proposition 3.41 *For any problem A in* APX, $\mathsf{AQH}(A) \subseteq \mathsf{QH}$.

PROOF: Assume that A is a maximization problem (the proof for minimization problems is similar). Let T be an r-approximate algorithm for A, for some $r > 1$, and let $L \in \mathsf{P}^{A_\rho}$ for some $\rho > 1$. Two polynomial-time computable functions f and g then exist witnessing this latter fact. For any x, let $m = \mathsf{m}(f(x), T(f(x)))$, so that $m \leq \mathsf{opt}(f(x)) \leq rm$. We can then partition the interval $[m, rm]$ into $\lfloor \log_\rho r \rfloor + 1$ subintervals

$$[m, \rho m), [\rho m, \rho^2 m), \ldots, [\rho^{\lfloor \log_\rho r \rfloor - 1} m, \rho^{\lfloor \log_\rho r \rfloor} m], [\rho^{\lfloor \log_\rho r \rfloor} m, rm],$$

and start looking for the subinterval containing the optimum value (a similar technique has been used in [40, 41]). This can clearly be done using $\lfloor \log_\rho r \rfloor + 1$ queries to an NP-complete oracle. One more query is sufficient to know whether a feasible solution y exists whose value lies in that interval and such that $g(x, y) = 1$. Since y is ρ-approximate, it follows that L can be decided using $\lfloor \log_\rho r \rfloor + 2$ queries, that is, $L \in \mathsf{QH}$. •

The next proposition, instead, states that any language L in the query hierarchy can be decided using just one query to A_r where A is APX-complete and r depends on the level of the query hierarchy L belongs to. In order to prove this proposition, we need the following technical result.

Lemma 3.42 *For any* APX*-complete problem A and for any k, two polynomial-time computable functions f and g and a constant r exist such that, for any k-tuple $(x_1, \ldots, x_k)$ of instances of* PARTITION, *$x = f(x_1, \ldots, x_k)$ is an instance of A and if y is a solution of x whose performance ratio is smaller than r then $g(x, y) = (b_1, \ldots, b_k)$ where $b_i \in \{0, 1\}$ and $b_i = 1$ if and only if x_i is a yes-instance.*

PROOF: Let $x_i = (U_i, s_i)$ be an instance of PARTITION for $i = 1, \ldots, k$. Without loss of generality, we can assume that the U_is are pairwise disjoint and that, for any i, $\sum_{u \in U_i} s_i(u) = 2$. Let $w = (U, s, \preceq)$ be an instance of MIN ORDERED BIN PACKING defined as follows (a similar construction has been used in [125]).

1. $U = \bigcup_{i=1}^{k} U_i \cup \{v_1, \ldots, v_{k-1}\}$ where the v_is are new items.
2. For any $u \in U_i$, $s(u) = s_i(u)$ and $s(v_i) = 1$ for $i = 1, \ldots, k-1$.
3. For any $i < j \le k$, for any $u \in U_i$, and for any $u' \in U_j$, $u \preceq v_i \preceq u'$.

Any solution of w must be formed by a sequence of packings of $U_1, \ldots, U_k$ such that, for any i, the bins used for U_i are separated by the bins used for U_{i+1} by means of one bin which is completely filled by v_i. In particular, the packings of the U_is in any optimum solution must use either two or three bins: two bins are used if and only if x_i is a yes-instance. The optimum measure thus is at most $4k-1$ so that any $(1 + 1/(4k))$-approximate solution is an optimum solution.

Since MIN ORDERED BIN PACKING belongs to APX [134] and A is APX-complete, then an AP-reduction (f_1, g_1, α) exists from MIN ORDERED BIN PACKING to A. We can then define $x = f(x_1, \ldots, x_k) = f_1(w, 1 + 1/(4\alpha k))$ and $r = 1 + 1/(4\alpha k)$. For any r-approximate solution y of x, the fourth property of the AP-reducibility implies that $z = g_1(x, y, 1 + 1/(4\alpha k))$ is a $(1 + 1/(4k))$-approximate solution of w and thus an optimum solution of w. From z, we can easily derive the right answers to the k queries $x_1, \ldots, x_k$. •

We are now able to prove the following result.

Proposition 3.43 *For any* APX*-complete problem A,* QH $\subseteq$ AQH(A).

PROOF: Let $L \in$ QH, then $L \in \mathsf{P}^{\mathsf{NP}[h]}$ for some h. It is well known (see, for instance, [22]) that L can be reduced to the problem of answering $k = 2^{h-1}$ non-adaptive queries to NP. More formally, two polynomial-time computable functions t_1 and t_2 exist such that, for any x, $t_1(x) = (x_1, \ldots, x_k)$, where $x_1, \ldots, x_k$ are k instances of the PARTITION problem, and for any $(b_1, \ldots, b_k) \in \{0, 1\}^k$, $t_2(x, b_1, \ldots, b_k) \in \{0, 1\}$. Moreover, if, for any j, $b_j = 1$ if and only if x_j is a yes-instance, then $t_2(x, b_1, \ldots, b_k) = 1$ if and only if $x \in L$.

Let now f, g and r be the two functions and the constant of Lemma 3.42 applied to problem A and constant k. For any x, $x' = f(t_1(x))$ is an instance of A such that if y is an r-approximate solution for x', then $t_2(g(x', y)) = 1$ if and only if $x \in L$. Thus, $L \in \mathsf{P}^{A_r}$. •

By combining Propositions 3.41 and 3.43, we thus have the following theorem that characterizes the approximation query hierarchy of the hardest problems in APX.

Theorem 3.44 *For any* APX*-complete problem A,* AQH(A) = QH.

An alternative approach to relate query complexity and approximability is to define bounded query classes of *multivalued functions* and then "place" approximation problems into such classes. This approach has been introduced by Chang [39]. We shall give a full characterization of problems complete for polyAPX and for APX, respectively, in terms of hardness of the corresponding approximation problems in Chang's classes and of suitably defined combinatorial properties.

Definition 3.45 ([39]) $\mathsf{NPF}^{\mathsf{NP}[q(n)]}$ *is the class of partial multi-valued functions computable by nondeterministic polynomial-time Turing machines which ask at most $q(n)$ queries to an* NP *oracle in the* entire *computation tree.*[2]

In order to talk about hardness with respect to these classes we will use the following reducibility which is an extension of both metric reducibility [90] and one-query reducibility [48] and has been introduced in [39].

Definition 3.46 (MV reducibility) *Let F and G be two partial multi-valued functions. We say that F many-one reduces to G (in symbols, $F \leq_{\mathrm{mv}} G$) if two polynomial-time algorithms t_1 and t_2 exist such that, for any x in the domain of F, $t_1(x)$ is in the domain of G and, for any $y \in G(t_1(x))$, $t_2(x, y) \in F(x)$.*

The combinatorial property used to characterize polyAPX-complete problems is the well-known self-improvability (see, for instance, [119]).

Definition 3.47 (Self-improvability) *A problem A is* self-improvable *if two algorithms t_1 and t_2 exist such that, for any instance x of A and for any two rationals $r_1, r_2 > 1$, $x' = t_1(x, r_1, r_2)$ is an instance of A and, for any $y' \in A_{r_2}(x')$, $y = t_2(x, y', r_1, r_2) \in A_{r_1}(x)$. Moreover, for any fixed r_1 and r_2, the running time of t_1 and t_2 is polynomial.*

We are now ready to state our first characterization theorem.

Theorem 3.48 *A* polyAPX *problem A is* polyAPX*-complete if and only if it is self-improvable and A_{r_0} is $\mathsf{NPF}^{\mathsf{NP}[\log\log n + O(1)]}$-hard for some $r_0 > 1$.*

PROOF: Let A be a polyAPX-complete problem. Since MAX CLIQUE is self-improvable [70] and poly-APX-complete [102] and since the equivalence with respect to the AP-reducibility preserves the self-improvability property (see [119]), we have that A is self-improvable. It is then sufficient to prove that A_2 is hard for $\mathsf{NPF}^{\mathsf{NP}[\log\log n + O(1)]}$.

From the poly-APX-completeness of A we have that MAX CLIQUE $\leq_{\mathrm{AP}} A$: let α be the constant of this reduction. From Theorem 12 of [39] we have that any function F in $\mathsf{NPF}^{\mathsf{NP}[\log\log n + O(1)]}$ many-one reduces to MAX CLIQUE$_{1+\alpha}$. From the definition of AP-reducibility, we also have that MAX CLIQUE$_{1+\alpha} \leq_{\mathrm{mv}} A_2$ so that F many-one reduces to A_2.

Conversely, let A be a polyAPX self-improvable problem such that, for some r_0, A_{r_0} is $\mathsf{NPF}^{\mathsf{NP}[\log\log n + O(1)]}$-hard. We will show that, for any problem B in poly-APX, B is AP-reducible to A. To this aim, we introduce the following partial multi-valued function `multisat`: given in input a sequence $(\varphi_1, \ldots, \varphi_m)$ of instances of the satisfiability problem with $m \leq \log |(\varphi_1, \ldots, \varphi_m)|$ and such that, for any i, if φ_{i+1} is satisfiable then φ_i is satisfiable, a possible output is a satisfying truth-assignment for φ_{i^*} where $i^* = \max\{i : \varphi_i \text{ is satisfiable}\}$. From the proof of Theorem 12 of [39] it follows that this function is $\mathsf{NPF}^{\mathsf{NP}[\log\log n + O(1)]}$-complete.

By making use of techniques similar to those of the proof of Proposition 3.41, it is easy to see that, since B is in poly-APX, two algorithms t_1^B and t_2^B exist such that, for any fixed $r > 1$, $t_1^B(\cdot, r)$ and $t_2^B(\cdot, \cdot, r)$ form a many-one reduction from B_r to `multisat`. Moreover, since A_{r_0} is $\mathsf{NPF}^{\mathsf{NP}[\log\log n + O(1)]}$-hard, then a many-one reduction (t_1^M, t_2^M) exists from `multisat` to A_{r_0}. Finally, let t_1^A and t_2^A be the functions witnessing the self-improvability of A.

The AP-reduction from B to A can then be derived as follows:

[2] We say that a multi-valued partial function F is computable by a nondeterministic Turing machine N if, for any x in the domain of F, an halting computation path of $N(x)$ exists and any halting computation path of $N(x)$ outputs a value of $F(x)$.

$$\begin{tikzcd}
x, r \arrow[r, "t_1^B(x,r)"] & x' \arrow[r, "t_1^M(x')"] & x'' \arrow[r, "t_1^A(x'',r_0,r)"] & x''' \arrow[d] \\
y & y' \arrow[l, "t_2^B(x,y',r)"'] & y'' \arrow[l, "t_2^M(x',y'')"'] & y''' \arrow[l, "t_2^A(x'',y''',r_0,r)"']
\end{tikzcd}$$

It is easy to see that if y''' is an r-approximate solution for the instance x''' of A, then y is an r-approximate solution of the instance x of B. That is, B is AP-reducible to A with $\alpha = 1$. •

The above theorem cannot be proved without the dependency of both f and g on r in the definition of AP-reducibility. Indeed, it is possible to prove that if only g has this property then, unless the polynomial-time hierarchy collapses, a self-improvable problem A exists such that A_2 is $\mathsf{NPF}^{\mathsf{NP}[\log\log n + O(1)]}$-hard and A is not polyAPX-complete.

In order to characterize APX-complete problems, we have to define a different combinatorial property. Intuitively, this property states that it is possible to merge several instances into one instance in an approximation preserving fashion. More specifically, we ask that the problem of $(1+\varepsilon)$-approximating k instances of a problem A can be reduced to $(1+\varepsilon/O(k))$-approximating a single instance of A.

Definition 3.49 (Linear Additivity) *An* NPO *problem A is* linearly additive *if a constant β and two algorithms t_1 and t_2 exist such that, for any rational $r > 1$ and for any sequence $x_1, \ldots, x_k$ of instances of A, $x' = t_1(x_1, \ldots, x_k, r)$ is an instance of A and, for any $y' \in A_{1+(r-1)\beta/k}(x')$, $t_2(x_1, \ldots, x_k, y', r) = y_1, \ldots, y_k$ where each y_i is an r-approximate solution of x_i. Moreover, the running time of t_1 and t_2 is polynomial for every fixed r.*

Theorem 3.50 *An* APX *problem A is* APX*-complete if and only if it is linearly additive and a constant r_0 exists such that A_{r_0} is* $\mathsf{NPF}^{\mathsf{NP}[1]}$*-hard.*

Proof: Let A be an r_A-approximable APX-complete problem. From the proof of Proposition 3.43 a constant r_0 exists such that A_{r_0} is hard for $\mathsf{NPF}^{\mathsf{NP}[1]}$. In order to prove the linear additivity, fix any $r > 1$ and let $x_1, \ldots, x_k$ be instances of A. Without loss of generality, we can assume $r < r_A$ (otherwise the k instances can be r-approximated by using the r_A-approximate algorithm). For any $i = 1, \ldots, k$ the problem of finding an r-approximate solution y_i for x_i is reducible to the problem of constructively solving a set of $\lceil \log_r r_A \rceil$ instances of Partition. Observe that $\lceil \log_r r_A \rceil \leq c/(r-1)$ for a certain constant c depending on r_A. Moreover, we claim that a constant γ exists such that constructively solving $kc/(r-1)$ instances of Partition is reducible to $(1+\gamma(r-1)/kc)$-approximating a single instance of A (indeed, this can be shown along the lines of the proof of Proposition 3.43). That is, A is linearly additive with $\beta = \gamma/c$.

Conversely, let A be a linearly additive APX problem such that A_{r_0} is $\mathsf{NPF}^{\mathsf{NP}[1]}$-hard for some r_0 and let B be an r_B-approximable problem. Given an instance x of B, for any $r > 1$ we can reduce the problem of finding an r-approximate solution for x to the problem of constructively solving $c/(r-1)$ instances of Partition, for a proper constant c not depending on r. Each of these questions is reducible to A_{r_0}, since any NP problem can be constructively solved by an $\mathsf{NPF}^{\mathsf{NP}[1]}$ function. From linear additivity, it follows that r_0-approximating $c/(r-1)$ instances of A is reducible to $(1+\beta(r_0-1)(r-1)/c)$-approximating a single instance of A. This is an AP-reduction from B to A with $\alpha = c/(\beta(r_0-1))$. •

Note that linear additivity plays for APX more or less the same role of self-improvability for polyAPX. These two properties are, in a certain sense, one the opposite of the other: while the usefulness of APX-complete approximation problems to solve decision problems depends on the performance ratio and does not depend on the size of the instance, the usefulness of polyAPX-complete approximation problems depends on the size of the instance and does not depend on the performance ratio. Indeed, it is possible to prove that no APX-complete problem can be self-improvable (unless P = NP) and that no polyAPX-complete problem can be linearly additive (unless the polynomial-time hierarchy collapses).

3.5 Applications to Natural Problems

3.5.1 Natural APX-intermediate Problems

The following result shows that Proposition 3.41 can be strengthened when restricting to APTAS problems.

Proposition 3.51 *Let* $A \in \mathsf{APTAS}$. *Then a constant* h *exists such that* $\mathsf{AQH}(A) \subseteq \mathsf{P}^{\mathsf{NP}[h]}$.

PROOF: Let A be a minimization problem in APTAS (the proof for maximization problem is very similar). By definition, a constant k and an algorithm T exist such that, for any instance x and for any rational $r > 1$,

$$\mathsf{m}(x, T(x,r)) \leq r \cdot \mathsf{opt}(x) + k.$$

We will now prove that a constant h exists such that, for any $r > 1$, a function $l_r \in \mathsf{PF}^{\mathsf{NP}[h-1]}$ exists such that, for any instance x of the problem A,

$$\mathsf{opt}(x) \leq l_r(x) \leq r \cdot \mathsf{opt}(x).$$

Intuitively, functions l_r form a non-constructive approximation scheme that is computable by a constant number of queries to an NP-complete oracle. Given an instance x, we can check whether $\mathsf{sol}(x) = \emptyset$ by means of a single query to an NP oracle, so that we can restrict ourselves to instances such that $\mathsf{sol}(x) \neq \emptyset$ (and thus $\mathsf{opt}(x) \geq 1$). Note that, for these instances, $T(\cdot, 2)$ is a $(k+2)$-approximate algorithm for A. Let us fix an $r > 1$, let $\varepsilon = r - 1$, $y = T(x, 1+\varepsilon/2)$ and $a = m(x, T(x,2))$. We have to distinguish two cases.

1. $a \geq 2k(k+2)/\varepsilon$: in this case, $\mathsf{opt}(x) \geq 2k/\varepsilon$, that is, $\mathsf{opt}(x)\varepsilon/2 \geq k$. Then

$$\begin{aligned} \mathsf{m}(x,y) &\leq \mathsf{opt}(x)(1+\varepsilon/2) + k \\ &\leq \mathsf{opt}(x)(1+\varepsilon/2) + \mathsf{opt}(x)\varepsilon/2 \\ &= \mathsf{opt}(x)(1+\varepsilon) = r\mathsf{opt}(x), \end{aligned}$$

   that is, y is an r-approximate solution for x, and we can set $l_r(x) = \mathsf{m}(x,y)$ (in this case l_r has been computed by only one query).

2. $a < 2k(k+2)/\varepsilon$: in this case, $\mathsf{opt}(x) < 2k(k+2)/\varepsilon$. Then,

$$\mathsf{opt}(x) \leq \mathsf{m}(x,y) \leq \mathsf{opt}(x) + \mathsf{opt}(x)\varepsilon/2 + k < \mathsf{opt}(x) + k(k+2) + k.$$

   Clearly, $\lceil \log k(k+3) \rceil$ queries to NP are sufficient to find the optimum value $\mathsf{opt}(x)$ by means of a binary search technique: in this case $l_r(x) = \mathsf{opt}(x)$ has been computed by $\lceil \log k(k+3) \rceil + 1$ queries.

Let now L be a language in $\mathsf{AQH}(A)$, then $L \in \mathsf{P}^{A_r}$ for some $r > 1$. Let f and g be the functions witnessing that $L \in \mathsf{P}^{A_r}$. Observe that, for any x, $x \in L$ if and only if a solution y for $f(x)$ exists such that $\mathsf{m}(f(x), y) \geq l_r(f(x))$ and $g(f(x), y) = 1$: that is, given $l_r(f(x))$, deciding whether $x \in L$ is an NP problem. Since $l_r(f(x))$ is computable by means of at most $\lceil \log k(k+3) \rceil + 1$ queries to NP, we have that $L \in \mathsf{P}^{\mathsf{NP}[h]}$ where $h = \lceil \log k(k+3) \rceil + 2$. •

The following result states the existence of natural intermediate problems in APX.

Theorem 3.52 *If the polynomial-time hierarchy does not collapse, then* Min Bin Packing, Min Degree Spanning Tree, *and* Min Edge Coloring *are* APX*-intermediate.*

Proof: From Proposition 3.51 and from the fact that Min Bin Packing is in APTAS [96], it follows that $\mathsf{AQH}(\text{Min Bin Packing}) \subseteq \mathsf{P}^{\mathsf{NP}[h]}$ for a given h. If Min Bin Packing is APX-complete, then from Proposition 3.43 it follows that $\mathsf{QH} \subseteq \mathsf{P}^{\mathsf{NP}[h]}$. From Theorem 2.25 we thus have the collapse of the polynomial-time hierarchy. The proofs for Min Degree Spanning Tree and Min Edge Coloring are identical and use the results of [84, 67]. •

Observe that the previous result does not seem to be obtainable by using the hypothesis $\mathsf{P} \neq \mathsf{NP}$, as shown by the following theorem.

Theorem 3.53 *If* $\mathsf{NP} = \text{co-}\mathsf{NP}$ *then* Min Bin Packing *is* APX*-complete.*

Proof: Assume $\mathsf{NP} = co\text{-}\mathsf{NP}$, we will present an AP reduction from Max Sat to Min Bin Packing. Since $\mathsf{NP} = co\text{-}\mathsf{NP}$ a nondeterministic polynomial time Turing machine M exists that, given in input an instance φ of Max Sat, has an accepting computation and all accepting computations halt with an optimum solution for φ written on the tape. Indeed, M guesses an integer k, an assignment τ such that $\mathsf{m}(\varphi, \tau) = k$ and a proof of the fact that $\mathsf{opt}(\varphi) \leq k$. From the proof of Cook's theorem it follows that, given φ, we can find in polynomial time a formula φ' such that φ' is satisfiable and that given any satisfying assignment for φ' we can find in polynomial time an optimum solution for φ. By combining this construction with the NP-completeness proof of the Min Bin Packing problem, we obtain two polynomial-time computable functions t_1 and t_2 such that, for any instance φ of Max Sat, $t_1(\varphi) = x_\varphi$ is an instance of Min Bin Packing such that $\mathsf{opt}(x_\varphi) = 2$ and, for any optimum solution y of x_φ, $t_2(x_\varphi, y)$ is an optimum solution of φ. Observe that, by construction, an r-approximate solution for x_φ is indeed an optimum solution provided that $r < 3/2$. Let T be a $4/3$-approximate algorithm for Max Sat [135, 75]. The reduction from Max Sat to Min Bin Packing is defined as follows: $f(\varphi, r) = t_1(\varphi)$;

$$g(\varphi, y, r) = \begin{cases} T(\varphi) & \text{if } r \geq 4/3, \\ t_2(t_1(\varphi), y) & \text{otherwise.} \end{cases}$$

It is immediate to verify that the above is an AP-reduction with $\alpha = 1$. •

3.5.2 Min Set Cover is not Self-improvable

It is an interesting question to find a characterizing combinatorial property of log-APX-complete problems. Indeed, we have not been able to establish this characterization: at present, we can only state that it cannot be based on the self-improvability property as shown by the following result.

Theorem 3.54 *No* logAPX*-complete problem can be self-improvable unless the polynomial-time hierarchy collapses.*

Proof: Let us consider the optimization problem Max Number of Satisfiable Formulas (in short, MNSF) defined as follows.

INSTANCE: Set of m boolean formulas $\varphi_1, \ldots, \varphi_m$ in 3CNF, such that φ_1 is a tautology and $m \leq \log n$, where n is the size of the input instance.

SOLUTION: Truth assignment τ to the variables of $\varphi_1, \ldots, \varphi_m$.

MEASURE: The number of satisfied formulas, i.e., $|\{i : \varphi_i \text{ is satisfied by } \tau\}|$.

Clearly, MNSF is in $\log\mathsf{APX}$, since the measure of any assignment τ is at least 1, and the optimum value is always smaller than $\log n$, where n is the size of the input. We will show that, for any $r < 2$, MNSF_r is hard for $\mathsf{NPF}^{\mathsf{NP}[\log\log\log n-1]}$.

Given $\log\log n$ queries to an NP-complete language (of size polynomial in n) $x_1, \ldots, x_{\log\log n}$, we can construct an instance $\Phi = \varphi_1, \ldots, \varphi_m$ of MNSF where φ_1 is a tautology and, for $i \geq 1$, the formulas $\varphi_{2^i} = \ldots = \varphi_{2^{i+1}-1}$ are satisfiable if and only if at least i instances among $x_1, \ldots, x_{\log\log n}$ are yes-instances (these formulas can be easily constructed using the standard proof of Cook's theorem). Note that $m = 2^{\log\log n+1} - 1$ and, by adding dummy clauses to some formulas, we can achieve the bound $m \leq \log|\varphi_1, \ldots, \varphi_m|$. Moreover, from an r-approximate solution for Φ we can decide how many instances in $x_1, \ldots, x_{\log\log n}$ are yes-instances, and we can also recover solutions for such instances. That is, any function in $\mathsf{NPF}^{\mathsf{NP}[\log\log\log n-1]}$ is many-one reducible to MNSF_r.

Let A be a self-improvable $\log\mathsf{APX}$-complete problem. Then, for any function $F \in \mathsf{NPF}^{\mathsf{NP}[\log\log\log n-1]}$, $F \leq_{\mathrm{mv}} \mathrm{MNSF}_{1.5} \leq_{\mathrm{mv}} A_{1+1/2\alpha} \leq_{\mathrm{mv}} A_{2^{16}}$ where α is the constant in the AP-reduction from MNSF to A and where the last reduction is due to the self-improvability of A. Thus, for any x, computing $F(x)$ is reducible to finding a 2^{16}-approximate solution for an instance x' with $|x'| \leq |x|^c$ for a certain constant c. Since $A \in \log\mathsf{APX}$, it is possible to find in polynomial time a $(k\log|x'|)$-approximate solution y for x' where k is a constant. From y, by means of binary search techniques, we can find a 2^{16}-approximate solution for x' using $\lceil\log\lceil\log_{2^{16}}(k\log|x'|)\rceil\rceil \leq \log\lceil\log\log|x|^{kc}\rceil - 3 \leq \log\log\log|x| - 2$ adaptive queries to NP where the last inequality surely holds for sufficiently large $|x|$. Thus,

$$\mathsf{NPF}^{\mathsf{NP}[\log\log\log n-1]} \subseteq \mathsf{NPF}^{\mathsf{NP}[\log\log\log n-2]}$$

which, from Theorem 2.26, implies the collapse of the polynomial-time hierarchy. •

It follows that, if the polynomial-time hierarchy does not collapse, MIN SET COVER and the other $\log\mathsf{APX}$-complete problems are not self-improvable. We can also prove that if $\mathsf{NP} = co\text{-}\mathsf{NP}$ then MIN SET COVER *is* self-improvable. The proof of this fact is very similar to the proof of Theorem 3.53 and is omitted.

3.5.3 Optimum vs. Approximate MAX CLIQUE Solutions

The following lemma is the analog of Proposition 3.41 within $\mathsf{NPO\ PB}$ and can be proved similarly by binary search techniques.

Lemma 3.55 *For any* $\mathsf{NPO\ PB}$ *problem A and for any $r > 1$,* $\mathsf{P}^{A_r} \subseteq \mathsf{P}^{\mathsf{NP}[\log\log n+O(1)]}$.

From this lemma, from the fact that $\mathsf{P}^{\mathsf{NP}[\log n]}$ is contained in $\mathsf{P}^{\mathrm{MC}_1}$ where MC stands for MAX CLIQUE [107], and from the fact that if a constant k exists such that

$$\mathsf{P}^{\mathsf{NP}[\log\log n+k]} = \mathsf{P}^{\mathsf{NP}[\log n]},$$

then the polynomial-time hierarchy collapses (see Theorem 2.26), it follows the next result that solves an open question posed in [40]. Informally, this result states that it is not possible to reduce the problem of finding a maximum clique to the problem of finding a 2-approximate clique (unless the polynomial-time hierarchy collapses).

Theorem 3.56 *If* $\mathsf{P}^{\mathrm{MC}_1} \subseteq \mathsf{P}^{\mathrm{MC}_2}$ *then the polynomial-time hierarchy collapses.*

3.6 Attributions and Related Works

The starting point of the results of this chapter has been a preliminary version of [102], where the E-reducibility was defined and completeness results in polynomially bounded approximation classes were proved. The idea of "scaling weights" to avoid the restriction to polynomially bounded classes was also suggested.

In a joint work with Pierluigi Crescenzi [50] we restricted our attention to APX, and we proved that the APX PB-completeness of Max 3Sat could not be extended to an APX-completeness result using known definitions of approximation preserving reducibility (Proposition 3.10). Motivated by this result, we introduced the PTAS-reducibility and we used it to prove the APX-completeness of Max 3Sat. The latter result was obtained by combining a less general version of Theorem 3.22, the results of Khanna et al. and a result of Crescenzi and Panconesi [47]. The proof of Theorem 3.21 and the version of Theorem 3.22 given in this chapter are new.

The comparison of different previous definitions of approximation-preserving reducibilities, the definition of AP-reducibility and the completeness results in NPO and NPO PB are from a joint paper with Pierluigi Crescenzi, Viggo Kann and Riccardo Silvestri [46].

The results of Section 3.3 have been obtained jointly with Pierluigi Crescenzi [51]. Lemma 3.28 has been proved in [50] using the PCP Theorem. The PCP-free proof given here is from [51]. Polynomial time approximation schemes for a large variety of planar constraint satisfaction problems have been developed by Khanna and Motwani [101]. Hunt et al. [85] developed NC approximation schemes for planar constraint satisfaction problems were each constraint depends on at most a constant number of variables. We give the first NC approximation schemes for constraint satisfaction problems over an infinite constraint family, as well as the first linear-time algorithms for this kind of problems. The proof of Theorem 3.36 is folklore and has been explained to me by Sanjeev Khanna.

The study of query complexity as a yardstick for the hardness of approximation problems was initiated by Chang, Gasarch and Lund [41, 40]. In [46] we extended their approach allowing to measure the complexity of generating (instead of evaluating) approximate solutions. The classes $\mathsf{NPF}^{\mathsf{NP}[\cdot]}$ were introduced in a survey paper by Chang [39], as a means of restating the results of a preliminary version of [46].

Chapter 4

Tight Reductions for Satisfiability Problems

"Forty-two" said Deep Thought, with infinity majesty and calm.(...) "It was a tough assignment," said Deep Thought mildly.
"Forty-two!" yelled Loonquawl. " Is that all you've got to show for seven and a half million years' work?"
"I checked it very thoroughly," said the computer.
(D. Adams [2])

This chapter introduces a series of general methods for developing stronger approximation-preserving reductions and thus infer improved non-approximability results. Our attention is focussed on *constraint satisfaction problems* (see Section 2.1). Recall that, besides Max Sat and its restrictions/generalizations, several other problems admit such kind of a formalization. Non-approximability results for those problems are proved using local-replacement reductions from the verifier of the PCP Theorem. The notion of "gadget", formalized by Bellare, Goldreich and Sudan [25], captures the way such reductions work.

We introduce a general technique that allows to find *automatically* (i.e. by running a computer program) the *best possible* gadget reducing a computation of a verifier to a given constraint satisfaction problem. The improved reductions found in this way give improved non-approximability results for Max Cut, Max DiCut and Max 2CSP. We also show how to make a *positive use* of gadgets, presenting improved approximation algorithms for Max 3Sat and Max 3CSP based on gadget reductions to Max 2Sat.

The non-approximability results that we prove (as well as those for Max Sat, Max 2Sat, and other problems [25, 80]) hold only for the weighted version. We give a technique to extend those results to the non-weighted version.

The Min Vertex Cover problem does not seemingly fit this framework of constraint satisfaction problems. We prove that it is equivalent to Min Sat, the problem of minimizing the number of satisfied clauses in a CNF boolean formula. Using different techniques, we prove that Min Vertex Cover and Min Sat are equivalent to their non-weighted versions. We also prove a non-approximability result for Min Vertex Cover on bounded-degree graphs that essentially matches the best known non-approximability result for general graphs.

Our reductions from weighted constraint satisfaction problems to their non-weighted version and from Min Vertex Cover to its bounded-degree restriction make use of combinatorial objects related to *expander graphs*. Randomized and deterministic constructions of those objects are presented in a separate section.

4.1 Gadgets and Linear Programming

4.1.1 Definitions

Definition 4.1 (gadget) *For $\alpha \in \mathcal{R}^+$, a constraint function $f : \{0,1\}^k \to \{0,1\}$, and a constraint family $\mathcal{F}$: an* α-gadget *reducing f to $\mathcal{F}$ is a finite collection of real weights $w_j \geq 0$, and associated constraints C_j from $\mathcal{F}$ over* primary variables $x_1, \ldots, x_k$ *and* auxiliary variables $y_1, \ldots, y_n$*, with the property that, for boolean assignments $\vec{a}$ to $x_1, \ldots, x_k$ and $\vec{b}$ to $y_1, \ldots, y_n$, the following are satisfied:*

$$(\forall \vec{a} : f(\vec{a}) = 1)\ (\forall \vec{b}) \ :\ \sum_j w_j C_j(\vec{a}, \vec{b}) \ \leq\ \alpha \tag{4.1}$$

$$(\forall \vec{a} : f(\vec{a}) = 1)\ (\exists \vec{b}) \ :\ \sum_j w_j C_j(\vec{a}, \vec{b}) \ =\ \alpha, \tag{4.2}$$

$$(\forall \vec{a} : f(\vec{a}) = 0)\ (\forall \vec{b}) \ :\ \sum_j w_j C_j(\vec{a}, \vec{b}) \ \leq\ \alpha - 1. \tag{4.3}$$

The gadget is strict *if, in addition,*

$$(\forall \vec{a} : f(\vec{a}) = 0)\ (\exists \vec{b}) \ :\ \sum_j w_j C_j(\vec{a}, \vec{b}) \ =\ \alpha - 1. \tag{4.4}$$

We say that the function $\vec{b} = \vec{b}(\vec{a})$ is a witness *for the gadget if equation (4.1) (and, for a strict gadget, equation (4.4)) is satisfied by $\vec{b}(\vec{a})$. For a given witness function $\vec{b}$, the function $b_i(\vec{a}) = \vec{b}(\vec{a})_i$.*

We observe that because the weights may all be re-scaled, what is significant is the ratio of the right-hand sides of equations (4.1–4.4). A "strong" gadget is one with a small α; in particular, if $\alpha' > \alpha$, any α-gadget is also automatically (after re-scaling) an α'-gadget. (But strictness is not maintained.) In the extreme, a 1-gadget would do a perfect reduction.

Because of the natural association between a gadget and the corresponding function $\sum w_j C_j$, a gadget may be thought of as an instance of a (weighted) constraint satisfaction problem. Viewed as a constraint satisfaction problem, a gadget has the property that when restricted to any satisfying assignment $\vec{a}$ to $x_1, \ldots, x_k$ its maximum is exactly α, and when restricted to any unsatisfying assignment its maximum is at most $\alpha - 1$ (exactly $\alpha - 1$ if the gadget is strict).

Our interest in gadget construction, and in the PC and RMBC families (that, contrary to the other one, do not yield natural constraint satisfaction problems) originally arose from the following result by Bellare, Goldreich and Sudan.

Lemma 4.2 *[25] For any family $\mathcal{F}$, if there exist α-gadgets reducing all the elements of* PC *to $\mathcal{F}$ and β-gadgets reducing all the elements of* RMBC *to $\mathcal{F}$, then for any $\gamma > 0$,* Max $\mathcal{F}$ *is hard to approximate to within $1 + \frac{.15}{.6\alpha + .4\beta - .15} - \gamma$.*

The current best starting point for non-approximability results is by now the following result by Håstad.

Lemma 4.3 *[80] For any family $\mathcal{F}$, if there exists an α_0-gadget reducing* PC_0 *to $\mathcal{F}$ and an α_1-gadget reducing* PC_1 *to $\mathcal{F}$, then for any $\gamma > 0$,* Max $\mathcal{F}$ *is hard to approximate to within $1 + \frac{1}{\alpha_0 + \alpha_1 - 1} - \gamma$.*

Gadget constructions reducing RMBC members to natural problems have no more applications, and will not be described in details, but we shall mention them for the sake of completeness.

4.1.2 The Basic Procedure

In this section we shall illustrate our technique by constructing a gadget reducing PC_0 to 2SAT.

The key aspect of our method is to limit the number of auxiliary variables, by showing that duplicates (in a sense to be clarified) can be eliminated by means of proper *substitutions*. In general this is possible if the target of the reduction be a family with a property that we name by analogy with the terminology for matroids.

Definition 4.4 *A constraint family $\mathcal{F}$ is* hereditary *if for any $f_i(x_1, \ldots, x_{n_i}) \in \mathcal{F}$, and any two indices $j, j' \in [n_i]$, the function f_i when restricted to $x_j \equiv x_{j'}$ and considered as a function of $n_i - 1$ variables, is identical (up to the order of the arguments) to some other function $f_{i'} \in \mathcal{F} \cup \{0, 1\}$, where $n_{i'} = n_i - 1$ (and 0 and 1 denote the constant functions).*

Lemma 4.5 *If an α-gadget Γ reducing f to a hereditary family $\mathcal{F}$ has a witness function for which two auxiliary variables are identical (i.e. $b_{j'}(\cdot) \equiv b_j(\cdot)$), or if an auxiliary variable is identical to a primary variable ($b_{j'}(\vec{a}) \equiv a_j$) then there is an α-gadget Γ' using one fewer auxiliary variable. If Γ is strict, so is Γ'.*

PROOF: We define a new gadget Γ' obtained from Γ by replacing each occurrence of $x_{j'}$ by x_j and argue that Γ' is an α'-gadget reducing f to $\mathcal{F}$ for some $\alpha' \leq \alpha$.

For any constraint C of Γ, define $\mathrm{red}(C)$ as follows. If $x_{j'}$ does not occur in C, then $\mathrm{red}(C) = C$. Otherwise, we tentatively define $\mathrm{red}(C)$ as the constraint obtained from C by replacing the occurrence of $x_{j'}$ by an occurrence of x_j. If C did not originally involve x_j, then $\mathrm{red}(C)$ is a valid constraint from $\mathcal{F}$. If C did involve x_j already, then $\mathrm{red}(C)$ contains two occurrences of x_j, which is not allowed by our definition. However, the hereditary property of $\mathcal{F}$ yields either an equivalent constraint $C' \in \mathcal{F}$ or else the constant function 0 or 1. In this case we reset $\mathrm{red}(C)$ to C' or the appropriate constant.

If $\Gamma = (C_1, \ldots, C_m, w_1, \ldots, w_m)$, then define a new gadget $\Gamma' = (\mathrm{red}(C_1), \ldots, \mathrm{red}(C_m), w_1, \ldots, w_m)$. Correspondingly, let $\vec{b'}(\vec{a})$ be identical to $b(\vec{a})$ but with $b_{j'}$ eliminated. Γ' has one fewer auxiliary variable ($x_{j'}$ never occurs in Γ').

By construction, $\Gamma'(\vec{a}, \vec{b'}(\vec{a})) \equiv \Gamma(\vec{a}, \vec{b}(\vec{a}))$, so Γ' satisfies the gadget-defining equation (2). (Similarly, for strict gadgets Γ, Γ' satisfies (4)). Also, the range of the universal quantification for Γ' is smaller than that for Γ, therefore Γ' satisfies inequalities (1) and (3). If constants are produced, subtracting them from both sides of the gadget-defining inequalities (1–4) produces an $(\alpha - w)$-gadget, where w is the total weight on clauses replaced by 1's. •

If the constraint family allows the complementation of any variable (as for example 2SAT but not CUT or DICUT), then the number of auxiliary variables may be approximately halved: we need only consider witness functions whose value is 1 at least as often as it is 0.

Definition 4.6 *A family $\mathcal{F}$ is* complementation-closed *if it is hereditary and, for any $f_i(x_1, \ldots, x_{n_i}) \in \mathcal{F}$, and any two indices $j, j' \in [n_i]$, the function f_i when restricted to $x_j \equiv \neg x_{j'}$ and considered as a function of $n_i - 1$ variables, is identical (up to the order of the arguments) to some other function $f_{i'} \in \mathcal{F} \cup \{0, 1\}$, where $n_{i'} = n_i - 1$ (and 0 and 1 denote the constant functions).*

The above property guarantees that we can make substitutions even if two columns in the (extended) witness are one the complement of the other. We need one more definition before stating our main result of this section.

Definition 4.7 *For a constraint f, call two variables $a_{j'}$ and a_j* distinct *if there exists an assignment $\vec{a}$, satisfying f, for which $a_{j'} \neq a_j$.*

Lemma 4.8 *Suppose f is a constraint on k variables, with s satisfying assignments and $k' \leq k$ variables distinct even under complementation. If there is an α-gadget reducing f to a hereditary (respectively, complementation-closed) constraint family $\mathcal{F}$, then there is an α-gadget with at most $2^s - k'$ (respectively, $2^{s-1} - k'$) auxiliary variables. If there is a strict α-gadget reducing f to a hereditary (respectively, complementation-closed) constraint family $\mathcal{F}$, then there is a strict α-gadget with at most $2^k - k'$ (respectively, $2^{k-1} - k'$) auxiliary variables.*

In some cases (e.g. 2SAT but not CUT), there is also no need to consider witness functions which are identically 0 or identically 1. (Clauses in which their corresponding auxiliary variables appear can be replaced by shorter clauses eliminating those variables.)

The previous discussion means that if we are looking for an α-gadget reducing PC_0 to 2SAT with the minimum value of α, then we can restrict our search to gadgets over at most 7 variables. Over 7 variables, $2 \cdot 7 + 4 \cdot \binom{7}{2}$ 2SAT constraints can be defined; call them $C_1, \ldots, C_{98}$. A gadget over 7 variables can thus be identified with the vector $(w_1, \ldots, w_{98})$ of the weights of the constraints. Since in [25] it is shown that an 11-gadget exists reducing PC_0 to 2SAT, it follows that in an optimum gadget no constraint will have a weight larger than 11. If we were allowed only integer weights over the constraints, then an optimum gadget could be found by exhaustive search over a space of 12^{98} elements. Allowing real weights over the constraints makes the search space infinite, yet we can use linear programming to find an optimum gadget quite efficiently. Consider the following linear program with $64 + 4 + 64 = 132$ constraints over 99 variables:

$$
\begin{array}{lll}
\text{minimize} & \alpha & \\
\text{subject to} & & \\
& \sum_j w_j C_j(\vec{a}, \vec{b}) \leq \alpha & (\forall \vec{a} : \mathrm{PC}_0(\vec{a}) = 1)\ (\forall \vec{b} \in \{0,1\}^4) \\
& \sum_j w_j C_j(\vec{a}, b(\vec{a})) = \alpha & (\forall \vec{a} : \mathrm{PC}_0(\vec{a}) = 1) \\
& \sum_j w_j C_j(\vec{a}, \vec{b}) \leq \alpha - 1 & (\forall \vec{a} : \mathrm{PC}_0(\vec{a}) = 0)\ (\forall \vec{b} \in \{0,1\}^4) \\
& \alpha \geq 0 & \\
& w_j \geq 0 & (\forall j \in [98])
\end{array}
\tag{LP1}
$$

In general, any "duplicated" variables can be eliminated from an α-gadget to give a "simplified" α'-gadget ($\alpha' \leq \alpha$), and the α'-gadget can be associated with its (fixed-length) vector of weights.

Theorem 4.9 *The weight vector associated with any simplified α-gadget provides a feasible solution to the associated LP, and conversely any feasible solution to the LP is a weight vector which describes an α-gadget. An optimal LP solution yields an optimal α-gadget (one where α is as small as possible).*

In particular, (LP1) has optimal solution $\alpha = 11$, proving the optimality of the [25] gadget.

4.2 Non-approximability Results Using Gadgets

We give gadgets reducing elements of PC to CUT, DICUT, and 2CSP. Our gadgets, combined with Lemma 4.3 imply improved non-approximability results for such problems.

4.2.1 The MAX CUT problem

We begin by showing an improved hardness result for the MAX WEIGHT CUT problem. It is not difficult to see that no gadget per Definition 4.1 can reduce PC_0 to CUT: For any setting of the variables which satisfies equation (4.1), the complementary setting has the opposite parity (so that it must be subject to inequality (4.1)), but the values of all the CUT constraints are unchanged (so the gadget's value is still α, violating (4.1)). Following [25], we generalize the definition.

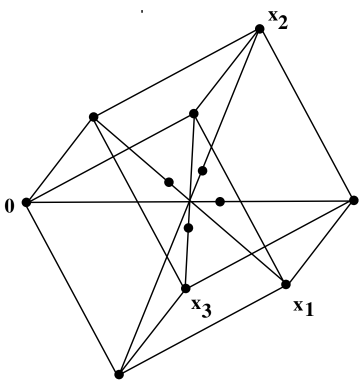


Figure 4.1: 8-gadget reducing PC_0 to CUT. Every edge has weight .5. The auxiliary variable which is always 0 is labelled 0.

Definition 4.10 *A* gadget with auxiliary constant 0 *is a gadget as previously defined, except that (4.1–4.4) are only required to hold when* $y_1 = 0$.

To get a hardness result for MAX CUT, we first need the following lemma, which is a very minor modification of Lemma 4.3.

Lemma 4.11 *[80] For the constraint family* CUT, *if there exists an* α_0*-gadget with constant 0 reducing* PC_0 *to* CUT *and an* α_1*-gadget with constant 0 reducing* PC_1 *to* CUT, *then for any* $\gamma > 0$, MAX CUT *is* NP*-hard to approximate to within* $1 + \frac{1}{\alpha_1+\alpha_2-1} - \gamma$.

Notice that the CUT constraint family is hereditary, since identifying the two variables in a CUT constraint yields the constant function 0. Thus by Lemma 4.8, if there is an α-gadget with constant 0 reducing a member of PC to CUT, then there is an α-gadget with at most 13 auxiliary variables (16 variables in all). Only $\binom{16}{2} = 120$ CUT constraints are possible on 16 variables. Since we only need to consider the cases when $y_1 = 0$, we can construct a linear program as above with $2^{16-1} + 4 = 32,772$ constraints to find the optimal α-gadgets with constant 0 reducing members of PC to CUT.

Lemma 4.12 *There exists an* 8*-gadget with constant 0 reducing* PC_0 *to* CUT, *and it is optimal and strict.*

We show the resulting gadget in Figure 4.1 as a graph. Each constraint of non-zero weight is shown as an edge; note that some auxiliary variables do not appear in any positive weight constraint, and thus are omitted from the graph. All non-zero weight constraints have weight .5. The auxiliary variable $y_1 = 0$ is shown as node "0" in the graph.

Lemma 4.13 *There exists a* 9*-gadget with constant 0 reducing* PC_1 *to* CUT, *and it is optimal and strict.*

The two lemmas imply the following theorem.

Theorem 4.14 *For every* $\gamma > 0$, MAX CUT *is hard to approximate to within* $17/16 - \gamma$.

The RMBC **corner** Finding RMBC gadgets was somehow more tricky. We discuss this point since it led to an idea that can find applications for other reductions. Indeed, it turned out that we couldn't exactly apply the technique above to find an optimal gadget reducing, say, RMBC_{00}

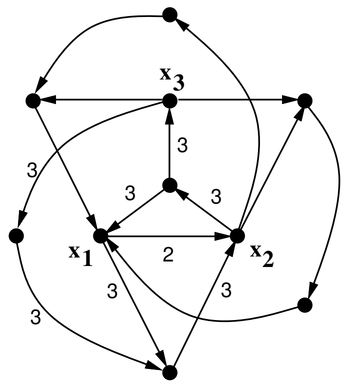


Figure 4.2: 8-gadget reducing PC_0 to DICUT. Edges have weight 1 except when marked otherwise.

to CUT. Since there are 8 satisfying assignments to the 4 variables of the $RMBC_{00}$ constraint, by Lemma 4.8, we would need to consider $2^8 - 4 = 252$ auxiliary variables, leading to a linear program with $2^{252} + 8$ constraints, which is somewhat beyond the capacity of current linear programming packages[1]. To overcome this difficulty, we observed that for the $RMBC_{00}$ function when $a_1 = 0$, the value of a_4 is irrelevant, and when $a_1 = 1$, the value of a_3 is irrelevant. Thus we considered only restricted witness functions for which $\vec{b}(0, a_2, a_3, 0) = \vec{b}(0, a_2, a_3, 1)$ and $\vec{b}(1, a_2, 0, a_4) = \vec{b}(1, a_2, 1, a_4)$. It is not *a priori* obvious that a gadget with a witness function of this form exists, but we assumed for that moment that such a gadget did exist. Following the proof of Lemma 4.8, we noted that the size of the domain of a restricted witness function is now 4 (instead of 8). Thus if an α-gadget with constant 0 and the restricted witness function existed, an α-gadget with constant 0 and at most 2^4 auxiliary variables existed as well. Noting that we could identify auxiliary variables identical to a_1 or a_2, we considered gadgets with at most 14 auxiliary variables. We then created a linear program with $\binom{18}{2} = 153$ variables and $2^{18-1} + 8 = 131,080$ constraints. The result of the linear program was that there exists an 8-gadget with constant 0 reducing $RMBC_{00}$ to CUT, and that it is strict. Notice that since we used a restricted witness function, the linear program did not prove that this gadget is optimal. However, its optimality can be proved, and this will be discussed in Section 4.4.

4.2.2 The MAX DICUT problem

In the case of the DICUT constraints, we do not need gadgets with a constant. By arguments similar to those above, if there exists an α-gadget reducing an element of PC to DICUT, there exists an α-gadget with 13 auxiliary variables. This leads to linear programs with $16 \cdot 15$ variables (one for each possible DICUT constraint, corresponding to a directed edge) and $2^{16} + 4 = 65,540$ linear constraints. The solution to the linear programs gives the following.

Lemma 4.15 *There exists a* 6.5*-gadget reducing* PC_0 *to* DICUT*, and it is optimal and strict.*

We show the resulting gadget in Figure 4.2.

Lemma 4.16 *There exists a* 6.5*-gadget reducing* PC_1 *to* DICUT*, and it is optimal and strict.*

The two lemmas imply the following theorem.

Theorem 4.17 *For every* $\gamma > 0$*,* MAX DICUT *is hard to approximate to within* $13/12 - \gamma$*.*

[1] Not even seven and a half million years would have been sufficient, let alone the deadline for submitting this thesis, and the fact that Håstad was going to revolutionize the field in far less time, confining the RMBC family to a premature obsolescence.

The RMBC **corner** Using the idea of a restricted witness function as in the reduction from RMBC to MAX CUT, we also set up a linear program to find α-gadgets reducing elements of RMBC to DICUT given the restricted witness function. The linear program has $18 \cdot 17 = 306$ variables and $2^{18} + 8 = 262,152$ constraints. Once again, because we consider only restricted witness functions, the linear program does not necessarily produce an optimal gadget, but we prove that no better gadget exists in Section 4.4. Our results were that there exists a 6-gadget reducing RMBC_{00} to DICUT, and it is strict; there exist 6.5-gadgets reducing RMBC_{01} and RMBC_{10} to DICUT, and they are strict; there exists a 7-gadget reducing RMBC_{11} to DICUT, and it is strict.

4.2.3 The MAX 2CSP problem

In the case of the 2CSP family, as in the case of 2SAT, we need only consider 4 auxiliary variables, or 7 total variables. There are two non-constant functions on a single variable, and twelve non-constant functions on pairs of variables, so that there are $2 \cdot 7 + 12 \cdot \binom{7}{2} = 266$ functions to consider overall. We can again set up a linear program with a variable per function and $2^7 + 4 = 132$ linear constraints. We obtain the following.

Lemma 4.18 *There exist 5-gadgets reducing* PC_0 *and* PC_1 *to* 2CSP*, and they are optimal and strict.*

The gadget reducing PC_0 to 2CSP is the following:

$$\begin{array}{cccc} x_1 \wedge \neg y_1, & x_1 \wedge y_2, & \neg x_1 \wedge y_3, & \neg x_1 \wedge y_4, \\ x_2 \wedge \neg y_1, & \neg x_2 \wedge y_2, & x_2 \wedge y_3, & \neg x_2 \wedge y_4, \\ \neg x_3 \wedge y_1, & x_3 \wedge \neg y_2, & x_3 \wedge \neg y_3, & \neg x_3 \wedge \neg y_4. \end{array}$$

The gadget reducing PC_1 to 2CSP can be obtained from the above one by complementing all the occurrences of x_1.

Theorem 4.19 *For every* $\gamma > 0$*,* MAX 2CSP *is hard to approximate to within* $10/9 - \gamma$*.*

MAX WEIGHT 2CSP can be approximated within 1.169 [60]. The above theorem has implications for probabilistically checkable proofs. Using the well known reduction from constraint satisfaction problems to probabilistically checkable proofs, Theorem 4.19 implies the following theorem.

Theorem 4.20 *For any* $\varepsilon > 0$*, constants* c *and* s *exist such that* $\mathsf{NP} \subseteq \mathsf{PCP}_{c,s}[\log, 2]$ *and* $c/s > 10/9 - \varepsilon$*.*

The RMBC result was the following: for any element of RMBC, there exists a 5-gadget reducing it to 2CSP, and it is optimal and strict.

4.3 Approximation Algorithms Using Gadgets

In this section we show that we can use gadgets to improve approximation algorithms. In particular, we look at MAX WEIGHT 3SATand MAX WEIGHT 3CONJSAT. An improved approximation algorithm for the latter problem leads to improved results for probabilistically checkable proofs in which the verifier examines only 3 bits. Both of the improved approximation algorithms rely on reducing the problem to a MAX WEIGHT 2SAT instance using gadgets. We begin with some notation.

Definition 4.21 *A (β_1, β_2)-approximation algorithm for* MAX WEIGHT 2SAT *is an algorithm which receives as input an instance with unary clauses of total weight m_1 and binary clauses of total weight m_2, and two reals $u_1 \le m_1$ and $u_2 \le m_2$, and produces reals $s_1 \le u_1$ and $s_2 \le u_2$ and an assignment satisfying clauses of total weight at least $\beta_1 s_1 + \beta_2 s_2$. If there exists an optimum solution that satisfies unary clauses of weight no more than u_1 and binary clauses of weight no more than u_2, then there is the guarantee that no assignment satisfies clauses of total weight more than $s_1 + s_2$.*

That is, supplied with a pair of "upper bounds" u_1, u_2, a (β_1, β_2)-approximation algorithm produces a single upper bound of $s_1 + s_2$, along with an assignment demonstrating a lower bound of $\beta_1 s_1 + \beta_2 s_2$.

Lemma 4.22 *[60] There exists a polynomial-time* $(.976, .931)$ *approximation algorithm for* MAX WEIGHT 2SAT.

4.3.1 The MAX 3SAT problem

In this section we show how to derive an improved approximation algorithm for MAX WEIGHT 3SAT. By restricting techniques in [75] from MAX WEIGHT SAT to MAX WEIGHT 3SAT and using a 1.074-approximation algorithm for MAX WEIGHT 2SAT due to Feige and Goemans [60], one can obtain a 1.299-approximation algorithm for MAX WEIGHT 3SAT. The basic idea of [75] is to reduce each clause of length 3 to the three possible subclauses of length 2, give each new length 2 clause one-third the original weight, and then apply an approximation algorithm for MAX WEIGHT 2SAT. This approximation algorithm is then "balanced" with another approximation algorithm for MAX WEIGHT 3SAT to obtain the result. Here we show that by using a strict gadget to reduce 3SAT to MAX WEIGHT 2SAT, a good (β_1, β_2)-approximation algorithm for MAX WEIGHT 2SAT leads to a 1.249-approximation algorithm for MAX WEIGHT 3SAT.

Lemma 4.23 *If for every $f \in$ E3SAT there exists a strict α-gadget reducing f to* 2SAT*, there exists a (β_1, β_2)-approximation algorithm for* MAX WEIGHT 2SAT*, and $\alpha \ge \frac{(\beta_1 - \beta_2)}{2(1-\beta_2)} + 1$, then there exists a $(1/\rho)$-approximation algorithm for* MAX WEIGHT 3SAT *with*

$$\rho = \frac{1}{2} + \frac{(\beta_1 - 1/2)(3/8)}{(\alpha - 1)(1 - \beta_2) + (\beta_1 - \beta_2) + (3/8)}.$$

PROOF: Let φ be an instance of MAX WEIGHT 3SAT with length 1 clauses of total weight m_1, length 2 clauses of total weight m_2, and length 3 clauses of total weight m_3. We use the two algorithms listed below, getting the corresponding upper and lower bounds on the number of satisfiable clauses:

- Random: We set each variable to 1 with probability 1/2. This gives a solution of weight at least $m_1/2 + 3m_2/4 + 7m_3/8$.
- Semidefinite programming: We use the strict α-gadget to reduce every length 3 clause to length 2 clauses. This gives an instance of MAX WEIGHT 2SAT, and we apply the (β_1, β_2)-approximation algorithm with parameters $u_1 = m_1$ and $u_2 = m_2 + \alpha m_3$ to find an approximate solution to this problem. The approximation algorithm gives an upper bound $s_1 + s_2$ on the weight of any solution to the MAX WEIGHT 2SAT instance and an assignment of weight $\beta_1 s_1 + \beta_2 s_2$. When translated back to the MAX WEIGHT 3SAT instance, the assignment has weight at least $\beta_1 s_1 + \beta_2 s_2 - (\alpha - 1)m_3$. Furthermore, $s_1 \le m_1$, $s_2 \le m_2 + \alpha m_3$, and the maximum weight satisfiable in the MAX WEIGHT 3SAT instance is at most $s_1 + s_2 - (\alpha - 1)m_3$.

Thus the performance guarantee of the algorithm which takes the better of the two solutions above is at most $(1/\rho_1)$, where

$$\rho_1 \stackrel{\text{def}}{=} \min_{\substack{s_1 \le m_1 \\ s_2 \le m_2 + \alpha m_3}} \frac{\max\{m_1/2 + 3m_2/4 + 7m_3/8, \;\; \beta_1 s_1 + \beta_2 s_2 - (\alpha - 1)m_3\}}{s_1 + s_2 - (\alpha - 1)m_3}$$

We now define a sequence of simplifications which will help prove the bound.

$$\rho_2 \stackrel{\text{def}}{=} \min_{\substack{t_1 \le m_1 \\ t_2 \le m_2 + m_3}} \frac{1}{t_1 + t_2} \max\{ \begin{array}{l} m_1/2 + 3m_2/4 + 7m_3/8, \\ \beta_1 t_1 + \beta_2 t_2 - (1 - \beta_2)(\alpha - 1)m_3\} \end{array}$$

$$\rho_3 \stackrel{\text{def}}{=} \min_{\substack{t_1 \le m_1 \\ t_2 \le m_2 + m_3}} \frac{1}{t_1 + t_2} \max\{ \begin{array}{l} t_1/2 + 3t_2/4 + m_3/8, \\ t_1/2 + 7m_3/8, \\ \beta_1 t_1 + \beta_2 t_2 - (1 - \beta_2)(\alpha - 1)m_3\} \end{array}$$

$$\rho_4 \stackrel{\text{def}}{=} \min_{t_2 \le t} \frac{1}{t} \max\{ \begin{array}{l} t/2 + t_2/4 + m_3/8, \\ t/2 - t_2/2 + 7m_3/8, \\ \beta_1 t - (\beta_1 - \beta_2)t_2 - (1 - \beta_2)(\alpha - 1)m_3\} \end{array}$$

$$\rho_5 \stackrel{\text{def}}{=} \frac{1}{2} + \left(\frac{\frac{3}{8}(\beta_1 - \frac{1}{2})}{(1 - \beta_2)(\alpha - 1) + (\beta_1 - \beta_2) + \frac{3}{8}} \right)$$

To finish the proof of the lemma, we claim that

$$\rho_1 \ge \rho_2 \ge \cdots \ge \rho_5.$$

To see this, notice that the first inequality follows from the substitution of variables $t_1 = s_1$, $t_2 = s_2 - (\alpha - 1)m_3$. The second follows from the fact that setting m_1 to t_1 and m_2 to $\max\{0, t_2 - m_3\}$, only reduces the numerator. The third inequality follows from setting $t = t_1 + t_2$. The fourth is obtained by substituting a convex combination of the arguments instead of max and then simplifying. The convex combination takes a θ_1 fraction of the first argument, θ_2 of the second and θ_3 of the third, where

$$\theta_1 = \frac{\frac{2}{3}(1 - \beta_2)(\alpha - 1) + \frac{7}{6}(\beta_1 - \beta_2)}{(1 - \beta_2)(\alpha - 1) + (\beta_1 - \beta_2) + \frac{3}{8}},$$

$$\theta_2 = \frac{\frac{1}{3}(1 - \beta_2)(\alpha - 1) - \frac{1}{6}(\beta_1 - \beta_2)}{(1 - \beta_2)(\alpha - 1) + (\beta_1 - \beta_2) + \frac{3}{8}}$$

and

$$\theta_3 = \frac{\frac{3}{8}}{(1 - \beta_2)(\alpha - 1) + (\beta_1 - \beta_2) + \frac{3}{8}}.$$

Observe that $\theta_1 + \theta_2 + \theta_3 = 1$ and that the condition on α guarantees that $\theta_2 \ge 0$. •

Remark 4.24 *The analysis given in the proof of the above lemma is tight. In particular for an instance with m clauses such that*

$$m_3 \stackrel{\text{def}}{=} m \frac{\beta_1 - 1/2}{(1 - \beta_2)(\alpha - 1) + (\beta_1 - \beta_2) + 3/8},$$

$m_1 \stackrel{\text{def}}{=} m - m_3$*,* $m_2 \stackrel{\text{def}}{=} 0$*,* $s_1 = m_1$*, and* $s_2 = \alpha m_3$*, it is easy to see that* $\rho_1 = \rho_5$*.*

Lemma 4.25 *For every function $f \in$ E3SAT, there exists a strict 3.5-gadget reducing f to 2SAT.*

PROOF: Since 2SAT is complementation-closed, it is sufficient to only present the gadget reducing $(x_1 \vee x_2 \vee x_3)$ to 2SAT. The gadget is

$$x_1 \vee x_3, \quad \neg x_1 \vee \neg x_3,$$

$$x_1 \vee \neg y, \quad \neg x_1 \vee y,$$

$$x_3 \vee \neg y, \quad \neg x_3 \vee y,$$

$$x_2 \vee y,$$

where every clause except the last has weight 1/2, and the last clause has weight 1. •

We remark that a 7-gadget reducing $(x_1 \vee x_2 \vee x_3)$ to 2SAT was found by Garey, Johnson and Stockmayer in 1976 [71].

Combining Lemmas 4.22, 4.23 and 4.25 we get a 1.249-approximation algorithm.

Theorem 4.26 *There exists a polynomial-time 1.249-approximation algorithm for* MAX WEIGHT 3SAT.

4.3.2 The MAX 3CSP problem

We now turn to the MAX WEIGHT 3CSP problem. In order to approximate MAX 3CSP it is sufficient to approximate its restricted version MAX WEIGHT 3CONJSAT. This is a consequence of a general result not restricted to 3-ary constraints.

Theorem 4.27 *For any $k > 1$ and for any $r \geq 1$, if* MAX WEIGHT kCONJSAT *is r-approximable, then* MAX WEIGHT k-CSP *is r-approximable.*

PROOF: Without loss of generality, we can assume that any constraint involves the application of a boolean function of arity exactly k (indeed, any h-ary function, with $h < k$ can be seen as a k-ary function whose value is independent of the value of $k-h$ parameters). Let $(C_1, \dots, C_m, w_1, \dots, w_m)$ be an instance of MAX WEIGHT k-CSP over boolean variables $x_1, \dots, x_n$. For any $j \in [m]$, $C_j = f_j(x_{j_1}, \dots, x_{j_k})$ where $j_1, \dots, j_k \in [n]$ and $f_j : \{1, 0\}^k \to \{1, 0\}$. Consider now the set $S_{f_j} \subseteq \{1, 0\}^k$ of satisfying assignment for f_j. For any such assignment $b_1, \dots, b_k$ we can write a k-ary conjunctive clause that is satisfied only by that assignment, namely, the clause $l_{j_1} \wedge \dots \wedge l_{j_k}$ where l_{j_i} stands for x_{j_i} if $b_i = 1$, and for $\neg x_{j_i}$ otherwise. In this way, we can convert any constraint C_j into $|S_{f_j}|$ new conjunctive constraints. We give weight w_j to all such constraints. Let $D_1, \dots, D_{m'}$ be the union of such sets of new constraints: they constitute an instance of MAX WEIGHT kCONJSAT over variables $x_1, \dots, x_n$, and it should be clear that any assignment to $x_1, \dots, x_n$ has the same measure for $C_1, \dots, C_m$ and for $D_1, \dots, D_{m'}$, so the instances have the same optimum, and an r-approximate solution for $D_1, \dots, D_{m'}$ is also an an r-approximate solution for $C_1, \dots, C_m$. •

Approximation algorithms for MAX kCONJSAT also imply containment if P of certain PCP classes.

Theorem 4.28 **(**MAX kCONJSAT **vs PCP)** *If* MAX kCONJSAT *is r-approximable for some $r \geq 1$, then $\mathsf{PCP}_{c,s}[\log, k] \subseteq \mathsf{P}$ for any $c/s > r$.*

PROOF: Let $L \in \mathsf{PCP}_{c,s}[\log, k]$ and let V be a verifier witnessing it. Let x be any string of size n. Let l be the maximum length of a proof for x (note that l is polynomial in n). On input x, V tosses $\alpha \log n$ random bits (for some constant α) and adaptively reads q bits of the proof. We show that for any of its deterministic computations there is a 1-gadget reducing it to kCONJSAT.

Details follow. Let $\mathsf{r} = 2^{\alpha \log n}$ be the total (polynomial) number of possible sequences of coin tosses of V. For any of such strings $R \in \{0,1\}^{\alpha \log n}$, let us consider the behaviour of V with input x and random tosses R: clearly it is entirely determined by the outcomes of its queries, and can be represented as a binary tree of height k. Indeed, any internal vertex corresponds to a query, branching corresponds to the two possible outcomes of the query and leaves correspond to final accepting or rejecting states. Any root-leaf path corresponds to a set of queries (positions of the proof) $q_1, \ldots, q_k$ and to answers $a_1, \ldots, a_k$, and it is followed by the verifier if and only if $\pi[q_i] = a_i$ for any $i \in [k]$. Let us now associate a boolean variable x_i to any position $\pi[i]$ of the proof, for any $i \in [l]$. There is an immediate correspondence between proofs (i.e. strings in $\{0,1\}^l$) and assignments for such variables (i.e. strings in $\{0,1\}^l$), say by identifying 1 with 1, and 0 with 0. According to this intuition, we encode root-leaf paths leading to acceptance as conjunctive clauses in the expected way: a path involving queries $q_1, \ldots, q_k$ and answers $a_1, \ldots, a_k$ is converted into the conjunctive clause $l_{q_1} \wedge \ldots, l_{q_k}$ where l_{q_i} stands for x_{q_i} if $a_i = 1$, and stands for $\neg x_{q_i}$ otherwise. It should be clear that those clauses are inconsistent one with the other, and that an assignments satisfies one of those clauses iff the corresponding proof makes the verifier accept when it uses random string R. We can repeat such construction for any R, and it leads to an instance of Max kConjSAT such that i clauses are simultaneously satisfiable iff a proof exists that makes V accept x with probability i/r. It follows that the optimum of the resulting Max kConjSAT instance is at least $c\mathsf{r}$ if $x \in L$, and is at most $s\mathsf{r}$ otherwise. Any approximation better than c/s is sufficient to distinguish between the two cases. •

We shall now present an approximation algorithm for Max Weight 3ConjSAT. Our analysis is similar to that above.

Lemma 4.29 *If for every $f \in$ 3ConjSAT there exists a strict $(\alpha_1+\alpha_2)$-gadget reducing f to 2SAT composed of α_1 length 1 clauses and α_2 length 2 clauses, and there exists a (β_1, β_2)-approximation algorithm for Max Weight 2SAT, then there exists a $(1/\rho)$-approximate algorithm for Max Weight 3ConjSAT with*

$$\rho = \frac{\frac{1}{8}\beta_1}{\frac{1}{8} + (1-\alpha_1)(\beta_1 - \beta_2) + (1-\beta_2)(\alpha_1 + \alpha_2 - 1)}$$

provided $\alpha_1 + \alpha_2 > 1 + 1/8(1 - \beta_2)$.

Proof: Let φ be an instance of Max Weight 3ConjSAT with constraints of total weight m. As in the Max Weight 3SAT case, we use two algorithms and take the better of the two solutions:

- Random: We set every variable to 1 with probability half. The total weight of satisfied constraints is at least $m/8$.
- Semidefinite programming: We use the strict α-gadget to reduce any constraint to 2SAT clauses. This gives an instance of Max Weight 2SAT and we use the (β_1, β_2)-approximation algorithm with parameters $u_1 = \alpha_1 m$ and $u_2 = \alpha_2 m$. The algorithm returns an upper bound $s_1 + s_2$ on the total weight of satisfiable constraints in the Max Weight 2SAT instance, and an assignment of measure at least $\beta_1 s_1 + \beta_2 s_2$. When translated back to the Max Weight 3ConjSAT instance, the measure of the assignment is at least $\beta_1 s_1 + \beta_2 s_2 - (\alpha_1 + \alpha_2 - 1)m$. Furthermore, $s_1 \le \alpha_1 m$, $s_2 \le \alpha_2 m$, and the total weight of satisfiable constraints in the Max Weight 3ConjSAT instance is at most $s_1 + s_2 - (\alpha_1 + \alpha_2 - 1)m$.

Thus we get that the performance ratio of the algorithm which takes the better of the two solutions above is at most $1/\rho_1$ where ρ_1 is defined below.

$$\rho_1 \stackrel{\text{def}}{=} \min_{\substack{s_1 \le \alpha_1 m \\ s_2 \le \alpha_2 m}} \frac{\max\{m/8, \;\; \beta_1 s_1 + \beta_2 s_2 - (\alpha_1 + \alpha_2 - 1)m\}}{s_1 + s_2 - (\alpha_1 + \alpha_2 - 1)m}$$

We now define a sequence of simplifications which will help prove the bound.

$$\rho_2 \stackrel{\text{def}}{=} \min_{\substack{t_1 \leq \alpha_1 m \\ t_2 \leq (1-\alpha_1)m}} \frac{1}{t_1+t_2} \max\{m/8, \quad \beta_1 t_1 + \beta_2 t_2 - (1-\beta_2)(\alpha_1+\alpha_2-1)m\}$$

$$\rho_3 \stackrel{\text{def}}{=} \min_{\substack{t \leq m \\ t_2 \leq (1-\alpha_1)m}} \frac{1}{t} \max\{m/8, \quad \beta_1 t - (\beta_1-\beta_2)t_2 - (1-\beta_2)(\alpha_1+\alpha_2-1)m\}$$

$$\rho_4 \stackrel{\text{def}}{=} \min_{t \leq m} \frac{1}{t} \max\{m/8, \quad \beta_1 t - ((1-\alpha_1)(\beta_1-\beta_2) + (1-\beta_2)(\alpha_1+\alpha_2-1))m\}$$

$$\rho_5 \stackrel{\text{def}}{=} \frac{\frac{1}{8}\beta_1}{\frac{1}{8} + (1-\alpha_1)(\beta_1-\beta_2) + (1-\beta_2)(\alpha_1+\alpha_2-1)}$$

In order to prove the lemma, we claim that

$$\rho_1 \geq \rho_2 \geq \cdots \geq \rho_5.$$

To see this, observe that the first inequality follows from the substitution of variables $t_1 = s_1$ and $t_2 = s_2 - (\alpha_1+\alpha_2-1)m$. The second follows from setting $t = t_1 + t_2$. The third inequality follows from the fact that setting t_2 to $(1-\alpha_1)m$ only reduces the numerator. The fourth is obtained by substituting a convex combination of the arguments instead of max and then simplifying. •

Lemma 4.30 *For any $f \in$ 3ConjSAT there exists a 4-gadget reducing f to 2SAT. The gadget is composed of one length 1 clause and 3 length 2 clauses.*

Proof: Recall that 2SAT is complementation-closed, and thus it is sufficient to exhibit a gadget reducing $f(a_1, a_2, a_3) = a_1 \wedge a_2 \wedge a_3$ to 2SAT. Such gadget is

$$y, (\neg y \vee x_1), (\neg y \vee x_2), (\neg y \vee x_3),$$

where all clauses have weight 1. The variables x_1, x_2, x_3 are primary variables and y is an auxiliary variable. •

Theorem 4.31 *There exists a polynomial time 2.7214-approximate algorithm for* Max Weight 3ConjSAT.

Corollary 4.32 *There exists a polynomial time 2.7214-approximate algorithm for* Max Weight 3CSP.

Corollary 4.33 $\mathsf{PCP}_{c,s}[\log, 3] \subseteq \mathsf{P}$ *provided that $c/s > 2.7214$.*

4.4 Lower Bounds for Gadget Constructions

In this section we shall show that some of the gadget constructions mentioned in this paper and in [25] are optimal, and we shall prove lower bounds for some other gadget constructions.

The following result is useful to prove lower bounds for the RMBC family.

Lemma 4.34 *If there exists an α-gadget reducing an element of the* RMBC *family to a complementation-closed constraint family $\mathcal{F}$, then there exists an α-gadget reducing all elements of* PC *to $\mathcal{F}$.*

PROOF: If a family $\mathcal{F}$ is complementation-closed, then an α-gadget reducing an element of PC (respectively RMBC) to $\mathcal{F}$ can be modified (using complementations) and yield α gadgets reducing all elements of PC (respectively of RMBC) to $\mathcal{F}$. For this reason, we will restrict our analysis to PC_0 and RMBC_{00} gadgets. Note that, for any $a_1, a_2, a_3 \in \{0,1\}^3$, $\mathrm{PC}_0(a_1, a_2, a_3) = 1$ if and only if $\mathrm{RMBC}_{00}(a_1, a_2, a_3, \bar{a}_3) = 1$. Let Γ be an α gadget over primary variables $x_1, \ldots, x_4$ and auxiliary variables $y_1, \ldots, y_n$ reducing RMBC to 2SAT. Let Γ' be the gadget obtained from Γ by imposing $x_4 \equiv \bar{x}_3$: it is immediate to verify that Γ' is an α-gadget reducing PC_0 to $\mathcal{F}$. •

4.4.1 Reducing PC and RMBC to 2SAT

Theorem 4.35 *If Γ is an α-gadget reducing an element of* PC *to* 2SAT*, then $\alpha \geq 11$.*

PROOF: It suffices to consider PC_0. We prove that the optimum of (LP1) is at least 11. To this end, consider the dual program of (LP1). We have a variable $y_{\vec{a},\vec{b}}$ for any $\vec{a} \in \{0,1\}^3$ and any $\vec{b} \in \{0,1\}^4$, plus additional variables $\hat{y}_{\vec{a},b(\vec{a})}$ for any $\vec{a} : \mathrm{PC}(\vec{a}) = 1$. The formulation is

$$\begin{array}{ll}
\text{maximize} & \sum_{\vec{a},\vec{b}:\mathrm{PC}(\vec{a})=0} y_{\vec{a},\vec{b}} \\
\text{subject to} & \\
 & \sum_{\vec{a},\vec{b}} y_{\vec{a},\vec{b}} \leq 1 + \sum_{\vec{a}:\mathrm{PC}(\vec{a})=1} y_{\vec{a},b(\vec{a})} \\
\forall j \in [98] & \sum_{\vec{a},\vec{b}} y_{\vec{a},\vec{b}} C_j(\vec{a},\vec{b}) \geq \sum_{\vec{a}:\mathrm{PC}(\vec{a})=1} \hat{y}_{\vec{a},b(\vec{a})} C_j(\vec{a}, b(\vec{a})) \\
(\forall \vec{a} \in \{0,1\}^3)\ (\forall \vec{b} \in \{0,1\}^4) & y_{\vec{a},\vec{b}} \geq 0 \\
(\forall \vec{a} : \mathrm{PC}(\vec{a}) = 1) & \hat{y}_{\vec{a},b(\vec{a})} \geq 0
\end{array} \qquad \text{(DUAL1)}$$

There exists a feasible solution for (DUAL1) whose cost is 11. •

Corollary 4.36 *If Γ is an α-gadget reducing an element of* RMBC *to* 2SAT*, then $\alpha \geq 11$.*

4.4.2 Reducing PC and RMBC to SAT

Theorem 4.37 *If Γ is an α-gadget reducing an element of* PC *to* SAT*, then $\alpha \geq 4$.*

PROOF: As in the proof of Theorem 4.35 we give a feasible solution to the dual to obtain the lower bound. The linear program that finds the best gadget reducing PC_0 to SAT is similar to (LP1), the only difference being that a larger number of clauses are considered, namely, $N = \sum_{i=1}^{7} \binom{7}{i} 2^i$. The dual program is then

$$\begin{array}{ll}
\text{maximize} & \sum_{\vec{a},\vec{b}:\mathrm{PC}(\vec{a})=0} y_{\vec{a},\vec{b}} \\
\text{subject to} & \\
 & \sum_{\vec{a},\vec{b}} y_{\vec{a},\vec{b}} \leq 1 + \sum_{\vec{a}:\mathrm{PC}(\vec{a})=1} y_{\vec{a},b(\vec{a})} \\
\forall j \in [N] & \sum_{\vec{a},\vec{b}} y_{\vec{a},\vec{b}} C_j(\vec{a},\vec{b}) \geq \sum_{\vec{a}:\mathrm{PC}(\vec{a})=1} \hat{y}_{\vec{a},b(\vec{a})} C_j(\vec{a}, b(\vec{a})) \\
(\forall \vec{a} \in \{0,1\}^3)\ (\forall \vec{b} \in \{0,1\}^4) & y_{\vec{a},\vec{b}} \geq 0 \\
(\forall \vec{a} : \mathrm{PC}(\vec{a}) = 1) & \hat{y}_{\vec{a},b(\vec{a})} \geq 0
\end{array} \qquad \text{(DUAL2)}$$

Consider now the following assignment of values to the variables of (DUAL2) (the unspecified values have to be set to zero):

$$\begin{array}{l}
(\forall \vec{a} : \mathrm{PC}(\vec{a}) = 1)\ \hat{y}_{\vec{a},b(\vec{a})} = \frac{3}{4} \\
(\forall \vec{a} : \mathrm{PC}(\vec{a}) = 1)(\forall \vec{a}' : d(\vec{a},\vec{a}') = 1)\ y_{\vec{a}',b(\vec{a})} = \frac{1}{3}
\end{array}$$

where d is the Hamming distance between binary sequences. It is possible to show that this is a feasible solution for (DUAL2) and it is immediate to verify that its cost is 4. •

Corollary 4.38 *If Γ is an α-gadget reducing an element of* RMBC *to* SAT*, then $\alpha \geq 4$.*

4.4.3 Reducing PC and RMBC to CUT and DICUT

Theorem 4.39 *If Γ is an α-gadget reducing an element of* PC *to* CUT*, then $\alpha \geq 8$.*

Theorem 4.40 *If Γ is an α-gadget reducing an element of* RMBC *to* CUT*, then $\alpha \geq 8$.*

Theorem 4.41 *If Γ is an α-gadget reducing an element of* PC *to* DICUT*, then $\alpha \geq 6.5$.*

Theorem 4.42 *If Γ is an α-gadget reducing an element of* RMBC *to* DICUT*, then $\alpha \geq 6$.*

4.4.4 Reducing kSAT to lSAT

Let k and l be any integers such that $k > l \geq 3$. The standard reduction from EkSAT to lSAT can be seen as a $\lceil (k-2)/(l-2) \rceil$-gadget. In this section we shall show that this is asymptotically the best possible. Note that since lSAT is complementation-closed we can restrict ourselves to considering just one constraint function of EkSAT, say $f(a_1, \ldots, a_k) \equiv \bigvee_i a_i$.

Theorem 4.43 *For any $k > l > 2$, if Γ is an α-gadget reducing f to $lSAT$ then $\alpha \geq k/l$.*

PROOF: As shown in Section 4.1.2, we can write a linear program whose optimum gives the smallest α such that an α-gadget exists reducing f to lSAT. Let b be the witness function used to formulate such linear program. We can assume that b is 2^{2^k}-ary and we let $K = 2^{2^k}$. Let also N be the total number of constraints from lSAT that can be defined over $k + K$ variables. Assume some enumeration $C_1, \ldots, C_N$ of such constraints. The dual formulation is

$$\begin{array}{ll}
\text{maximize} & \sum_{\vec{a},\vec{b}:\mathrm{PC}(\vec{a})=0} y_{\vec{a},\vec{b}} \\
\text{subject to} & \\
 & \sum_{\vec{a},\vec{b}} y_{\vec{a},\vec{b}} \leq 1 + \sum_{\vec{a}:f(\vec{a})=1} y_{\vec{a},b(\vec{a})} \\
\forall j \in [N] & \sum_{\vec{a},\vec{b}} y_{\vec{a},\vec{b}} C_j(\vec{a},\vec{b}) \geq \sum_{\vec{a}:f(\vec{a})=1} \hat{y}_{\vec{a},b(\vec{a})} C_j(\vec{a}, b(\vec{a})) \\
(\forall \vec{a} \in \{0,1\}^k)\ (\forall \vec{b} \in \{0,1\}^K) & y_{\vec{a},\vec{b}} \geq 0 \\
(\forall \vec{a} : f(\vec{a}) = 1) & \hat{y}_{\vec{a},b(\vec{a})} \geq 0
\end{array} \tag{DUAL3}$$

Let $A_k \subset \{0,1\}^k$ be the set of binary k-ary strings with exactly one non-zero component (note that $|A_k| = k$). Let also $\vec{0}$ (respectively, $\vec{1}$) be the k-ary string all whose components are equal to 0 (respectively, 1). The following is a feasible solution for (DUAL3) whose cost is k/l. We only specify non-zero values.

$$\begin{array}{l}
\forall \vec{a} \in A_k\ \hat{y}_{\vec{a},b(\vec{a})} = 1/l \\
\forall \vec{a} \in A_k\ y_{\vec{0},b(\vec{a})} = 1/l \\
\forall \vec{a} \in A_k\ y_{\vec{1},b(\vec{a})} = 1/k\ .
\end{array}$$

•

It is very interesting to contrast the above result with Lemma 3.24. In force of the above lower bound, a gadget reduction cannot provide an approximation-preserving reduction from MAX SAT to MAX kSAT. More generally, there cannot be an approximation-preserving gadget reduction from MAX SAT to MAX $(\log n)$SAT. Such a reduction can instead be done with our "disjunction shrinker" technique and, in this case, we have a *tight* reduction that preserves the approximation threshold up to lower-order terms.

Theorem 4.44 *The approximation thresholds of* MAX SAT *and* MAX $k(n)$SAT *are equal, provided* $k(n)$ *is a monotone non-decreasing unbounded function.*

PROOF: We prove that for any r and any $r' < r$, the existence of an r'-approximate algorithm for MAX $k(n)$SAT implies the existence of an r-approximate algorithm for MAX SAT.

Let p and h be such that

$$r \geq \max\{1/(1-(1-p)^{h+1})\ ,\ r'/(1-p)\}\ .$$

Since $k(n)$ is non-decreasing and unbounded, there is some constant n_0 (depending only on h) such that $k(n) \geq h$ for any $n \geq n_0$. The approximation algorithm for MAX SAT uses the technique of Lemma 3.24 and an r'-approximation algorithm for MAX hSAT. It remains to be seen that MAX hSAT is r'-approximable. Indeed, the only instances of MAX hSAT that are not also instances of MAX $k(n)$SAT have length at most $n_0 = O(1)$. For those instances an optimum solution can be found in constant time. •

We conclude that our technique for shrinking disjunctions produces reductions that *beat the gadget bound*, that is, that are *provably better* than any possible reduction by local replacement that fits the framework of these sections. We don't know if there is any way of producing a simple reduction from, say, PC_0 to MAX CUT that beats the gadget bound. A positive answer would imply a furtherly improved non-approximability result for MAX CUT.

4.4.5 Free bits vs. MAX SAT

In [25] it has been shown that one can prove a hardness result for MIN VERTEX COVER using Lemma 4.2 and constructing appropriate gadgets. Indeed, the definition of gadget can be easily adapted to deal with the syntactic structure of MIN VERTEX COVER (even if we gave the definition only for maximization problems). However, a stronger hardness result is proved in [25] in a different way: they note that the general reduction defined in [61] from PCP verifiers to MAX CLIQUE can be also seen as a reduction from PCP verifiers to MIN VERTEX COVER, and the efficiency of such reduction depends only on the *free bit complexity* of the verifier. They then develop a very efficient verifier minimizing such parameter and get a quite strong hardness result for MIN VERTEX COVER. In the following, we will show that such verifier cannot be used to prove hardness results for MAX SAT. We first rephrase the definition of free bits in the context of constraint satisfaction.

Definition 4.45 (Free bits) *A boolean function* $f : \{0,1\}^k \to \{0,1\}$ *is said to have* free bit complexity c *if it has a DNF expression with at most* 2^c *clauses and such that the clauses are pairwise inconsistent.*

The following constraints arise in the construction of verifiers with low free bit complexity [79, 78].

Definition 4.46 (COMPLETE) *For a fixed* $s \geq 1$, *let* $f_1, \ldots, f_{2^{2^s}}$ *be an enumeration of the functions from* $\{0,1\}^s$ *onto* $\{0,1\}$; *then the* s-ary complete test *constraint* $\mathrm{COMPLETE}_s : \{0,1\}^{s+2^{2^s}} \to \{0,1\}$ *is the function*

$$\mathrm{COMPLETE}_s(a_1, \ldots, a_s, b_1, \ldots, b_{2^{2^s}}) = \begin{cases} 1 & \text{if } f_j(a_1, \ldots, a_s) = b_j\ \forall j \in [2^{2^s}] \\ 0 & \text{otherwise.} \end{cases}$$

Definition 4.47 (AND_k) *For any* $k \geq 1$, *let* AND_k *be the* k*-ary constraint such that* $\mathrm{AND}_k(a_1, \ldots, a_k) = 1$ *iff* $a_1 = \ldots = a_k = 1$.

Note that AND_k has free bit complexity 0.

Theorem 4.48 *For any $k \geq 2$ and for any $\alpha < k$, there exists no α-gadget reducing* AND_k *to* SAT.

PROOF: Let us fix $k \geq 2$. Note that AND_k has only one satisfying assignment, thus we can restrict ourselves to consider gadgets *without auxiliary variables*. Let $\vec{1} \in \{0,1\}^k$ be the k-ary string of all ones, let $N = \sum_{l=1}^{k} \binom{l}{i} 2^i$ be the number of SAT constraints that can be defined over k boolean variables $\{x_1, \ldots, x_k\}$, and let $C_1, \ldots, C_N$ be an enumeration of such constraints. The following is a relaxation of the linear program that finds the best gadget reducing AND_k to SAT.

$$
\begin{array}{lll}
\text{minimize} & \alpha & \\
\text{subject to} & & \\
& & \sum_{j=1}^{N} w_j C_j(\vec{1}) \geq \alpha \\
& (\forall \vec{a} \in \{0,1\}^k - \vec{1}) & \sum_{j=1}^{N} w_j C_j(\vec{a}) \leq \alpha - 1 \\
& & \alpha \geq 0 \\
& (\forall j \in [N]) & w_j \geq 0
\end{array}
$$

It is a relaxation because the constraint enforcing Property 1 in Definition 4.1 is missing. The dual of the above linear program is

$$
\begin{array}{lll}
\text{maximize} & \sum_{\vec{a} \in \{0,1\}^k - \vec{1}} y_{\vec{a}} & \\
\text{subject to} & & \\
& & \sum_{\vec{a} \in \{0,1\}^k - \vec{1}} y_{\vec{a}} \leq 1 + y_{\vec{1}} \\
& (\forall j \in [N]) & \sum_{\vec{a} \in \{0,1\}^k - \vec{1}} y_{\vec{a}} C_j(\vec{1}) \geq C_j(\vec{1}) y_{\vec{1}} \\
& (\forall \vec{a} \in \{0,1\}^k & y_{\vec{a}} \geq 0
\end{array}
$$

Let $S \subset \{0,1\}^k$ be the set of binary k-tuples containing *exactly one zero* (that is, $|S| = k$). Consider the dual solution such that $y_{\vec{a}} = 1$ for all $\vec{a} \in S$, $y_{\vec{1}} = k - 1$, and all other variables are equal to zero. Such solution has cost k and satisfies the first constraint. We will now show that it satisfies all the other constraints, that is, that for any $j \in [N]$, the following holds:

$$\sum_{\vec{a} \in S} C_j(\vec{a}) \geq (k-1) C_j(\vec{1}) \,. \tag{4.5}$$

If C_j is a disjunction of negated literals, then $C_j(\vec{1}) = 0$ and (4.5) is trivially satisfied. Otherwise, let i be an index such that x_i has a positive occurrence in C_j. There are at least $k - 1$ assignments in S satisfying C_j (namely, the ones that give value 1 to x_i), thus (4.5) is satisfied. •

Proposition 4.49 *Let f be a constraint function with k distinct variables and free bit complexity zero. If Γ is an α-gadget reducing f to* SAT, *then $\alpha \geq k$.*

PROOF: By hypothesis, f is expressible as the conjunction of k literals. Since SAT is complementation-closed, we can use substitutions to convert any α-gadget reducing f to SAT into an α-gadget reducing AND_k to SAT. •

Theorem 4.50 *For any $s \geq 1$, for any α, if there exists an α-gadget reducing* $\mathrm{COMPLETE}_s$ *to* SAT, *then $\alpha \geq 2^{2^s}$.*

PROOF: Observe that for any $s \geq 1$ and for any $(a_1, \ldots, a_s) \in \{0,1\}^s$, we have that $F_s(b_1, \ldots, b_{2^{2^s}}) = \mathrm{COMPLETE}_s(0, \ldots, 0, b_1, \ldots, b_{2^{2^s}})$ is a constraint with 2^{2^s} variables and free bit complexity zero. Using substitutions, any α-gadget reducing $\mathrm{COMPLETE}_s$ to SAT yields an α-gadget reducing F_s to SAT. The Theorem follows from Proposition 4.49. •

4.5 Removing Weights from Constraint Satisfaction Problems

4.5.1 Polynomially Bounded Weights vs. Unbounded Weights

In the following we will consider optimization problems whose goal is to find a set of objects satisfying a given property with the maximum (or minimum) weight sum. More formally, we say that an optimization problem A is a *subset problem* if it can be expressed in the following way.

- Any instance x of A is a triple (U, x', ω), where $U = \{u_1, \ldots, u_n\}$ is a set of objects, x' is a (possibly empty) string, and $\omega : U \to N$ assigns a weight $\omega(u)$ to any object $u \in U$.
- With any solution $y \in \mathsf{sol}(x)$ is associated a subset $S \subseteq U$, and the measure of y is equal to $\sum_{u \in S} \omega(u)$.

For example, consider the weighted version of the MAX SAT problem. An instance can be seen as a triple (C, ε, ω) where C is the set of clauses and $\omega : C \to N$ is the function that weights the clauses. A solution is a truth-assignment to the variables in C: with any solution we can then associate the subset $C' \subseteq C$ of satisfied clauses and the measure of the truth-assignment is the sum of the weights of the elements of C'.

A subset problem A is *nice* if, for any instance (U, x', ω), if y is a feasible solution, then, for any function $\omega' : U \to N$, y is a feasible solution for (U, x', ω') as well. Roughly speaking, this property says that the definition of feasible solution in A is independent of the weights. It is easy to see that, for instance, MIN WEIGHT VERTEX COVER, and any MAX WEIGHT $\mathcal{F}$ problem are nice.

Let A be a subset problem and p be a polynomial. We denote by A^p the restriction of A to instances x such that the sum of the weights is at most $p(|x|)$. Note that that for a constraint family $\mathcal{F}$, MAX $\mathcal{F} \equiv^1_{\mathrm{AP}}$ MAX WEIGHT $\mathcal{F}^{n^\varepsilon}$ for any $\varepsilon > 1$.

The proof of the following result uses a scaling technique similar to that used to prove Theorem 3.22.

Theorem 4.51 (Scaling Theorem) *Let A be a nice subset problem in* APX *and let $r > 1$. A polynomial p exists such that if A^p is r-approximable, then A is $(r + 1/n)$-approximable.*

PROOF: We assume that A is a maximization problem (the proof for minimization problems is similar). Let A be r_0-approximable, let T be an r_0-approximate polynomial-time algorithm for A, and let $t(x) = \mathsf{m}(x, T(x))$. For any instance x, we have that

$$t(x) \leq \mathsf{opt}(x) \leq r_0 t(x).$$

Let $x = (U, x', \omega)$ be an instance of size n. Note that $|U| \leq n$. We define a new *scaled down* weight function $\tilde{\omega}$, such that, for any $u \in U$,

$$\tilde{\omega}(u) = \left\lfloor \frac{\omega(u) n^2}{t(x)} \right\rfloor .$$

Note that, since A is nice, the set of feasible solutions for (U, x', ω) is equal to that of $(U, x', \tilde{\omega})$. Moreover, we may assume without loss of generality that $\omega(u) \leq r_0 t(x)$ for any $u \in U$. It follows that $\tilde{\omega}(u) \leq r_0 n^2$ for any $u \in U$. Thus, $\tilde{x} = (U, x', \tilde{\omega})$ is an instance of $A^{r_0 n^3}$.

From the definition of $\tilde{\omega}$ it follows that, for any feasible solution y,

$$\mathsf{m}(x, y) \geq \mathsf{m}(\tilde{x}, y) t(x)/n^2.$$

Moreover,

$$\begin{aligned}
\mathsf{m}(x,y) &= \sum_{u\in S}\omega(u) \le \sum_{u\in S}\frac{t(x)}{n^2}\left\lceil\frac{\omega(u)n^2}{t(x)}\right\rceil \\
&\le (t(x)/n^2)\sum_{u\in S}\left(\left\lfloor\frac{\omega(u)n^2}{t(x)}\right\rfloor+1\right) \\
&= (t(x)/n^2)(\mathsf{m}(\tilde{x},y)+|S|) \\
&\le (t(x)/n^2)(\mathsf{m}(\tilde{x},y)+n).
\end{aligned}$$

On the other hand, we may assume that $\mathsf{m}(x,y)\ge t(x)$, otherwise we replace y with the solution computed by the approximation algorithm T. Then,

$$\begin{aligned}
\frac{\mathsf{opt}(x)}{\mathsf{m}(x,y)} &\le \frac{(t(x)/n^2)(\mathsf{opt}(\tilde{x})+n)}{\mathsf{m}(x,y)} \\
&\le \frac{(t(x)/n^2)\mathsf{opt}(\tilde{x})}{(t(x)/n^2)\mathsf{m}(\tilde{x},y)}+\frac{t(x)}{n\cdot\mathsf{m}(x,y)} \\
&\le \frac{\mathsf{opt}(\tilde{x})}{\mathsf{m}(\tilde{x},y)}+\frac{1}{n}.
\end{aligned}$$

Thus if y is an r-approximate solution for $\tilde{x}$, then y is an $(r+\frac{1}{n})$-approximate solution for x. The theorem follows. •

Corollary 4.52 *Let A be a nice subset problem such that $A\in\mathsf{APX}$. Then a polynomial p exists such that A and A^p have the same approximability threshold.*

It is easy to see that the previous theorem can be extended to any nice subset problem that admits an n^c-approximate algorithm where c is a constant and n denotes the number of objects in the instance. For example, it is possible to prove that a polynomial p exists such that if MAX WEIGHT CLIQUE^p is approximable within $r(n)$ then MAX WEIGHT CLIQUE is approximable within $r(n)+1/n$, where n denotes the number of nodes of the input graph.

4.5.2 The MAX CUT Problem

In this section we prove how to reduce r-approximating the MAX WEIGHT CUT^p problem to $(r-o(1))$-approximating the MAX SIMPLE CUT problem, for any polynomial p. We first give a rough sketch of the reduction, and then we present it formally. Let $(G=(V,E),\omega)$ be a weighted graph; we define a graph $\hat{G}=(\hat{V},\hat{E})$ in the following way: for each node $u\in V$ there are N nodes $u^1,\ldots,u^N$ in $\hat{V}$, and for each edge $(u,v)\in E$ of weight w, $\hat{E}$ contains the set of edges $\{(u^i,v^j)|(i,j)\in S^w\}$, where S^w is a random set of w elements of $N\times N$. Any cut of cost c in G clearly yields a cut of cost c in $\hat{G}$. Consider now a cut $\hat{C}$ in $\hat{G}$. For any $u\in V$, let p_u be the ratio of nodes $u^1,\ldots,u^N$ in $\hat{C}$, and let $q_u=1-p_u$. Since the sets S^w are random, it follows that

$$\mathsf{m}(\hat{G},\hat{C})\sim\sum_{(u,v)\in E}\omega(u,v)(p_uq_v+p_vq_u).$$

Now define a random cut C in G such that the probability that $u\in V$ is equal to p_u (observe that this is also known as a random rounding). Clearly, the expected measure of this cut satisfies the following equality:

$$\begin{aligned}\mathbf{E}[\mathsf{m}(G,\omega,C)] &= \sum_{(u,v)\in E} \omega(u,v)(p_u q_v + p_v q_u) \\ &\sim \mathsf{m}(\hat{G},\hat{C}).\end{aligned}$$

We will now be more formal and we will show how to *derandomize* the reduction. We first define a kind of graphs that will play the role of the random sets S^w.

Definition 4.53 (Mixing Graphs) *A bipartite graph $G = (V_1, V_2, E)$ is said to be $(n, w, \delta, \varepsilon)$-mixing if $|V_1| = |V_2| = n$, and for any two subsets $A \subseteq V_1$, $B \subseteq V_2$ with at least δn nodes, the following holds*

$$\left| Cut(A,B) - w\frac{|A||B|}{n^2} \right| \leq \varepsilon w \frac{|A||B|}{n^2}$$

where $Cut(A,B)$ denotes the number of edges between nodes in A and nodes in B.

We note that a $(n, w, \delta, \varepsilon)$-mixing graph has at least $(1-\varepsilon)w$ and at most $(1+\varepsilon)w$ edges.

It is well known that expander graphs have good mixing properties. Indeed, the proof of the following theorem (deferred to Section 4.7) uses a known construction of expanders.

Theorem 4.54 (Construction of Mixing Graphs) *For any $\varepsilon, \delta > 0$, two constants c and n_0 exist such that for any $n > n_0$ and for any $w \geq cn$, an $(n, w, \varepsilon, \delta)$-mixing graph can be constructed in time polynomial in n.*

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

Theorem 4.55 *For any $\alpha > 1$,* Max Weight Cut $\leq^{\alpha}_{\mathrm{AP}}$ Max Simple Cut.

Proof: Let $(G = (V, E), \omega)$ be an instance of Max Weight Cut, let $r > 1$ be fixed. Without loss of generality (see Theorem 4.51) we can assume that a polynomial q exists such that $\omega(e) \leq q(|V|)$ for any edge $e \in E$.

Let ε, δ be such that $0 < \varepsilon \leq 1/2$, $0 < \delta < 1/2$, and

$$\alpha \geq \frac{1}{r-1}\left(\frac{(1+\varepsilon)}{(1-\varepsilon)(1-12\delta)} r - 1\right) .$$

Let c, n_0 be the constants of Theorem 4.54 relative to ε and δ. We assume that there exists an integer $N > n_0$ such that, for any edge e, $cN \leq \omega(e) \leq N^2$ (otherwise we can multiply each weight by $c^2 n_0^2 w_{\max}$, where $w_{\max}$ is the maximum weight, and then set $N = cn_0 w_{\max}$).

In the following, $[N]$ denotes the set $\{1, \ldots, N\}$. We construct an unweighted graph $\hat{G} = (\hat{V}, \hat{E})$ as follows. For any $cN \leq w \leq N^2$ let $M_w = ([N], [N], E_w)$ be the $(N, w, \varepsilon, \delta)$-mixing graph whose existence is guaranteed by Theorem 4.54. We let $\hat{V}$ contain N copies of any vertex of V, and, for any edge e of weight $\omega(e)$ of G, we let $\hat{E}$ contain the edges of $M_{\omega(e)}$. More formally,

$$\hat{V} = \{v^i \ : \ v \in V \text{ and } i = 1, \ldots, N\}$$

and

$$\hat{E} = \{(u^i, v^j) \ : \ (u,v) \in E \text{ and } (i,j) \in E_{\omega(u,v)}\}.$$

Claim 4.56 *For any cut C in (G, ω) there exists a cut $\hat{C}$ in $\hat{G}$ such that*

$$\mathsf{m}(\hat{G},\hat{C}) \geq (1-\varepsilon)\mathsf{m}(G,\omega,C) .$$

PROOF: [Of Claim 4.56] Let $\hat{C} = \{u^i \ : \ u \in C \text{ and } i \in [N]\}$. Then,

$$\begin{aligned}
\mathsf{m}(\hat{G},\hat{C}) &= \sum_{(u,v)\in E, |\{u,v\}\cap C|=1} |E_{\omega(u,v)}| \\
&\geq \sum_{(u,v)\in E, |\{u,v\}\cap C|=1} (1-\varepsilon)\omega(u,v) \\
&= (1-\varepsilon)\mathsf{m}(G,\omega,C).
\end{aligned}$$

•

Claim 4.57 *Assume that, for any node u in G, we are given two rationals $p_u, q_u \geq 0$ such that $p_u + q_u = 1$, then we can find in polynomial time a cut C such that*

$$\mathsf{m}(G,\omega,C) \geq \sum_{(u,v)\in E} \omega(u,v)(p_u q_v + q_u p_v).$$

PROOF: [Of Claim 4.57] This is a folklore result. For the sake of completeness we sketch the proof: consider the *random* cut where each vertex u is chosen independently and with probability p_u. The expected measure of such a cut is clearly equal to $\sum_{(u,v)\in E} \omega(u,v)(p_u q_v + q_u p_v)$. Using the method of conditional probabilities (see e.g. [4]), a cut with measure not smaller than this expectation can be constructed in polynomial time. •

Claim 4.58 *Given a cut $\hat{C}$ in $\hat{G}$, we can construct in polynomial time a cut C for G such that $\mathsf{m}(G,\omega,C) \geq \mathsf{m}(\hat{G},\hat{C})(1-12\delta)/(1+\varepsilon)$.*

PROOF: [Of Claim 4.58] We assume that $\hat{C}$ is a local optimum, and thus

$$\mathsf{m}(\hat{G},\hat{C}) \geq 1/2 \sum_{e\in E} (1-\varepsilon)\omega(e) \geq 1/4 \sum_{e\in E} \omega(e).$$

We first transform $\hat{C}$ into a new cut $\hat{C}'$ such that, for any $u \in V$, at least δN copies of u belong to $\hat{C}'$ and at least δN copies belong to $\hat{V} - \hat{C}'$. Clearly $\mathsf{m}(\hat{G},\hat{C}')$ can be smaller than $\mathsf{m}(\hat{G},\hat{C})$. However, from the mixing property of each graph $M_{\omega(u,v)}$, it follows that

$$\begin{aligned}
\mathsf{m}(\hat{G},\hat{C}') &\geq \mathsf{m}(\hat{G},\hat{C}) - 2\sum_{e\in E}(1+\varepsilon)\delta\omega(e) \\
&\geq \mathsf{m}(\hat{G},\hat{C})(1-12\delta).
\end{aligned}$$

Let $p_u = |\{u^i : u^i \in \hat{C}'\}|/N$. Using the preceding claim we can construct a cut C such that

$$\mathsf{m}(G,\omega,C) \geq \sum_{(u,v)\in E} \omega(u,v)(p_u(1-p_v) + p_v(1-p_u)).$$

On the other hand, consider the measure of $\hat{C}'$: for any edge $(u,v) \in E$ we have to consider the edges connecting the Np_u copies of u in $\hat{C}'$ with the $N - Np_v$ copies of v not in $\hat{C}'$, plus the edges connecting the $N - Np_u$ copies of u not in $\hat{C}'$ with the Np_v copies of v in $\hat{C}'$. Recalling that such edges are chosen from a mixing graph, we have that

$$\begin{aligned}
\mathsf{m}(\hat{G},\hat{C}') &\leq \sum_{(u,v)\in E} \Big[\frac{(1+\varepsilon)\omega(u,v)}{N^2}(Np_u(N-Np_v)) \\
&\quad + \frac{(1+\varepsilon)\omega(u,v)}{N^2}(Np_v(N-Np_u))\Big] \\
&\leq (1+\varepsilon)\mathsf{m}(G,\omega,C).
\end{aligned}$$

•

From the above claims it follows that given a solution $\hat{C}$ for $\hat{G}$ whose performance ratio is at most r, we can construct in polynomial time a solution C whose performance ratio is

$$\begin{aligned}\frac{\mathsf{opt}(G,\omega)}{\mathsf{m}(G,\omega,C)} &\le \frac{\mathsf{opt}(\hat{G})/(1-\varepsilon)}{\mathsf{m}(\hat{G},\hat{C})(1-12\delta)/(1+\varepsilon)} \\ &\le \frac{1+\varepsilon}{(1-12\delta)(1-\varepsilon)} r \le 1+\alpha(r-1).\end{aligned}$$

We have thus shown that MAX WEIGHT CUT $\le_{\mathrm{AP}}^{\alpha}$ MAX SIMPLE CUT. •

Corollary 4.59 *For any $\gamma > 0$, it is* NP*-hard to approximate* MAX SIMPLE CUT *to within* $17/16-\gamma$.

4.5.3 Constraint Satisfaction Problems

In order to deal with general MAX $\mathcal{F}$ problems we have to introduce the notion of *mixing set* of tuples.

Definition 4.60 (Mixing Sets) *Let $n > 0$, $k > 0$ be integers; a set $S \subseteq [n]^k$ is k-ary $(n, w, \varepsilon, \delta)$-mixing if for any k sets $A_1, \ldots, A_k \subseteq [n]$, each with at least δn elements, the following holds:*

$$\left| |S \cap A_1 \times \cdots \times A_k| - w\frac{|A_1|\cdots|A_k|}{n^k} \right| \le \varepsilon w \frac{|A_1|\cdots|A_k|}{n^k}.$$

For any fixed k and for sufficiently large values of n and w, k-ary mixing sets are efficiently constructible. We defer to Section 4.7 the proof of the following theorem.

Theorem 4.61 (Construction of Mixing Sets) *For any integer $k \ge 2$ and for any two rationals $\varepsilon, \delta > 0$, two constants n_0 and c exist such that, for any $n \ge n_0$ and for any w with $cn \le w \le n^k$, a k-ary $(n, w, \varepsilon, \delta)$-mixing set is constructible in time polynomial in n.*

Theorem 4.62 *Let $\mathcal{F}$ be a constraint family without unary constraints. Then, for any $\alpha > 1$, it is the case that* MAX WEIGHT $\mathcal{F}$ $\le_{\mathrm{AP}}^{\alpha}$ MAX SIMPLE $\mathcal{F}$.

PROOF: The proof has the same structure of the proof of Theorem 4.55. We restrict ourselves to MAX WEIGHT $\mathcal{F}^p$, for a proper polynomial p. Let h be the largest arity of a constraint in $\mathcal{F}$. Let $\varepsilon, \delta \in (0, 1/2)$ be such that

$$\alpha \ge \frac{1}{r-1}\left(\frac{1+\varepsilon+2^h(1+\varepsilon-(1-\varepsilon)(1-\delta)^h)}{1-\varepsilon} r - 1\right) .$$

Let c, n_0 be constants such that Theorem 4.61 holds for any $k = 2, \ldots, h$. Let $\varphi = (C_1, \ldots, C_m, w_1, \ldots, w_m)$ be an instance of MAX WEIGHT $\mathcal{F}^p$ over variable set $X = \{x_1, \ldots, x_n\}$. We assume without loss of generality that the weights are polynomially bounded (otherwise we can apply Theorem 4.51) and that an integer $N > n_0$ exists such that for all $i \in [m]$ it holds $cN \le w_i \le N^2$ (otherwise we can multiply each weight by $c^2 n_0^2 w_{\max}$, where $w_{\max}$ is the maximum weight, and then set $N = cn_0 w_{\max}$). For any $cN \le w \le N^2$ and any $k \in [h]$ we let S_k^w be a k-ary $(N, w, \varepsilon, \delta)$-mixing set. We define an instance $\hat{\varphi}$ of MAX SIMPLE $\mathcal{F}$ as follows. The variable set of $\hat{\varphi}$ has N copies of any variable of X:

$$\hat{X} = \{x_i^j : i \in [n] \ , \ j \in [N]\} .$$

For a k-ary constraint $C_j = f(x_{i_1}, \ldots, x_{i_k})$ (with $f \in \mathcal{F}$) of weight w_j we define a constraint set $\mathrm{red}(C)$ as follows:

$$\mathrm{red}(C) = \{f(x_{i_1}^{a_1}, \ldots, x_{i_k}^{a_k}) : (a_1, \ldots, a_k) \in S_k^{w_j}\} .$$

$\hat{\varphi}$ is defined as the union of the $\mathrm{red}(C_j)$ sets

$$\hat{\varphi} = \bigcup_{j \in [m]} \mathrm{red}(C_j) .$$

We now claim that any assignment for φ can be transformed into an assignment for $\hat{\varphi}$ with roughly the same measure and vice versa.

Claim 4.63 *For any assignment τ for φ, an assignment $\hat{\tau}$ for $\hat{\varphi}$ exists such that*

$$\mathsf{m}(\hat{\varphi}, \hat{\tau}) \geq (1 - \varepsilon)\mathsf{m}(\varphi, \tau) .$$

PROOF: [Of Claim 4.63] We define $\hat{\tau}$ as follows: for any variable $x_i^j \in \hat{X}$, $\hat{\tau}(x_i^j) = \tau(x_i)$. We then have

$$\begin{aligned}
\mathsf{m}(\hat{\varphi}, \hat{\tau}) &= \sum_j |\{C \in \mathrm{red}(C_j) : \hat{\tau} \text{ satisfies } C\}| \\
&= \sum_{j : \tau \text{ satisfies } C_j} |\mathrm{red}(C_j)| \\
&\geq (1 - \varepsilon)\mathsf{m}(\varphi, \tau) .
\end{aligned}$$

Since $S_k^{w_j}$ is a k-ary $(N, w_j, \varepsilon, \delta)$-mixing set, it holds $|\mathrm{red}(C_j)| = |S_k^{w_j}| \geq (1 - \varepsilon)w_j$. •

Claim 4.64 *For any assignment $\hat{\tau}$ for $\hat{\varphi}$, an assignment τ for φ can be computed in polynomial time such that*

$$\mathsf{m}(\varphi, \tau) \geq \frac{1}{(1 + \varepsilon)(1 + 2^h(1 - (1 - \delta)^h))}\mathsf{m}(\hat{\varphi}, \hat{\tau}) .$$

PROOF: [Of Claim 4.64] Let $w_{\mathrm{tot}} = \sum_j w_j$. To begin with, we transform $\hat{\tau}$ into an assignment $\hat{\tau}'$ such that for any variable $x_i \in X$ at least δN and at most $(1 - \delta)N$ copies of x_i are set to 1 by $\hat{\tau}'$. To do that, for any $x_i \in X$ that does not fulfill the requested condition, we switch the value of some of its copies. Note that, for any $x_i \in X$, the value of at most δN of its copies is switched. This switching may be such that $\hat{\tau}'$ satisfies less constraints than $\hat{\tau}$ did. The difference between the measure of $\hat{\tau}$ and that of $\hat{\tau}'$ is bounded by the number of constraints where a switched variable occur. For each $j \in [m]$ we can prove (using the definition of mixing set and of $\mathrm{red}(C_j)$) that the number of constraints in $\mathrm{red}(C_j)$ that contain a switched variable is at most $(1 + \varepsilon)w_j - (1 - \varepsilon)(1 - \delta)^k w_j$, where k is the arity of C_j.

Summing over all $j \in [m]$ we derive that

$$\mathsf{m}(\hat{\varphi}, \hat{\tau}) - \mathsf{m}(\hat{\varphi}, \hat{\tau}') \leq w_{\mathrm{tot}}(1 + \varepsilon - (1 - \varepsilon)(1 - \delta)^h) .$$

For any variable $x_i \in X$, let

$$p_i = \frac{|\{j : \hat{\tau}'(x_i^j) = 1\}|}{N}$$

be the fraction of copies of x_i whose value is set to 1 by $\hat{\tau}'$. We define τ_R as a *random assignment* (more formally, a distribution over all the assignments) of X in the following way: $\mathbf{Pr}[\tau_R(x_i) = 1] = p_i$. A way of looking at this random assignment is the following: for any $i \in [n]$, a random value $j \in [N]$ is chosen uniformly at random and then the value $\tau_R(x_i)$ is set equal to $\hat{\tau}'(x^i_j)$. For a k-ary constraint $C_j = f(x_{i_1}, \ldots, x_{i_k})$ of φ of weight w_j, let $SAT_j \subseteq \{0,1\}^k$ be the set of *satisfying assignments* to the variables of C_j. From the mixing property of the set $S_k^{w_j}$ used to define $\mathrm{red}(C_j)$ it is easy to derive

$$\mathbf{Pr}[\tau_R \text{ satisfies } C_j] =$$

$$\begin{aligned}
& \sum_{(b_1,\ldots,b_k)\in SAT_j} \mathbf{Pr}[\tau_R(x_{i_1}) = b_1, \ldots, \tau_R(x_{i_k}) = b_k] \\
= & \sum_{(b_1,\ldots,b_k)\in SAT_j} \frac{|\{(a_1,\ldots,a_k) \in [N]^k : \hat{\tau}'(x^{a_1}_{i_1}) = b_1, \ldots, \hat{\tau}'(x^{a_k}_{i_k}) = b_k\}|}{N^k} \\
\geq & \sum_{(b_1,\ldots,b_k)\in SAT_j} \frac{|\{(a_1,\ldots,a_k) \in S^w_k : \hat{\tau}'(x^{a_1}_{i_1}) = b_1, \ldots, \hat{\tau}'(x^{a_k}_{i_k}) = b_k\}|}{(1+\varepsilon)w_j} \\
\geq & \frac{|\{C \in \mathrm{red}(C_j) : \hat{\tau}' \text{ satisfies } C\}|}{(1+\varepsilon)w_j} .
\end{aligned}$$

And thus we have that

$$\mathbf{E}[\mathsf{m}(\varphi, \tau_R)] \geq \mathsf{m}(\hat{\varphi}, \hat{\tau}')/(1+\varepsilon) .$$

Using the method of conditional expectation [4], given a probability distribution over the variable of φ of average measure m_R it is possible to find in polynomial time an assignment of measure at least m_R. Note that the random assignment such that each variable is 1 or 0 with probability 1/2 has avergae measure at least $2^{-h} w_{\mathrm{tot}}$. It follows that we can find in polynomial time an assignment τ whose measure is at least the average measure of τ_R and at least $2^{-h} w_{\mathrm{tot}}$. We now compare the measure of τ with that of $\hat{\tau}$

$$\begin{aligned}
\mathsf{m}(\varphi, \tau) & \geq \frac{1}{1+\varepsilon}(\mathsf{m}(\hat{\varphi}, \hat{\tau}) - w_{\mathrm{tot}}(1+\varepsilon-(1-\varepsilon)(1-\delta)^h)) \\
& \geq \frac{1}{1+\varepsilon}(\mathsf{m}(\hat{\varphi}, \hat{\tau}) - 2^h \mathsf{m}(\varphi, \tau)(1+\varepsilon-(1-\varepsilon)(1-\delta)^h))
\end{aligned}$$

$$\mathsf{m}(\varphi, \tau) \geq \frac{1}{(1+\varepsilon+2^h(1+\varepsilon-(1-\varepsilon)(1-\delta)^h))} \mathsf{m}(\hat{\varphi}, \hat{\tau})$$

•

From the above claims we have that given a solution $\hat{\tau}$ for $\hat{\varphi}$ whose performance ratio is at most r, we can construct in polynomial time a solution τ for φ whose performance ratio is at most

$$\begin{aligned}
\frac{\mathsf{opt}(\varphi)}{\mathsf{m}(\varphi, \tau)} & \leq \frac{\mathsf{opt}(\varphi)/(1-\varepsilon)}{\mathsf{m}(\hat{\varphi}, \hat{\tau})/((1+\varepsilon)(1+2^h(1-(1-\delta)^h)))} \\
& \leq \frac{(1+\varepsilon)(1+2^h(1-(1-\delta)^h))}{1-\varepsilon} r \leq 1 + \alpha(r-1) .
\end{aligned}$$

We have thus shown that MAX WEIGHT $\mathcal{F} \leq^{\alpha}_{\mathrm{AP}}$ MAX SIMPLE $\mathcal{F}$. •

Corollary 4.65 *The following problems have the same approximation threshold of their weighted version:* Max Simple DiCut, Max Simple EkSat *for any* $k \geq 2$, Max Simple 2SAT, Max Simple k-CSP *for any* $k \geq 2$.

Proof: The only non-trivial result is for Max Simple 2SAT (since 2SAT contains unary constraints). We use a result of Yannakakis [135] who proved that Max Weight 2SAT$\leq^1_{\mathrm{AP}}$ Max Weight E2SAT. Thus we have that, for any $\alpha > 1$,

$$\text{Max Weight 2SAT} \leq^1_{\mathrm{AP}} \text{Max Weight E2SAT} \leq^{\alpha}_{\mathrm{AP}} \text{Max Simple E2SAT} .$$

•

The following corollary contains a small sample of the results implied by Theorem 4.62.

Corollary 4.66 *For any* $\gamma > 0$, *it is* NP*-hard to approximate* Max Simple DiCut *to within* $13/12 - \gamma$, Max Simple E3Sat *(and thus* Max Simple 3Sat *and* Max Simple Sat*) within* $8/7 - \varepsilon$, Max Simple 2CSP *within* $10/9 - \gamma$.

4.6 The Vertex Cover Problem

4.6.1 Yet Another Constraint Satisfaction Problem

The Min Weight Vertex Cover and the Min Weight Sat problems are both nice ones. In this section, we will mainly deal with the unweighted and the polynomially bounded weighted version of these two problems.

The following two lemmas can be proved by using arguments similar to the reduction between Max Clique and zero-free bit PCP given in [25] and the reduction between zero-free bit PCP and Max Clique implicitly given in [61], respectively.[2]

Lemma 4.67 Min Vertex Cover$\leq^1_{\mathrm{AP}}$ Min Sat.

Proof: Let $G = (V, E)$ be a graph. Fix an arbitrary total order $\leq$ in V (e.g. lexicographic order). We define an instance of Min Sat as follows. There is a variable $x_{u,v}$ for any edge $(u, v) \in E$; there is a clause C_u for any vertex $u \in V$. C_u is defined as

$$C_u = \bigvee_{v:(u,v)\in E \wedge u \leq v} x_{u,v} \vee \bigvee_{v:(u,v)\in E \wedge u \geq v} \neg x_{u,v} .$$

Given an assigment τ for φ, $C = \{u : \tau \text{ satisfies } C_u\}$ is a vertex cover for G and has the same measure of τ. Indeed, if $(u, v) \in E$ then $x_{u,v}$ occurs in C_u and C_v once negated and once not, so at least one of the clauses is satisfied and so either $u \in C$ or $v \in C$.

Conversely, if C is a vertex cover then we define an assigment τ as follows. For any $u \notin C$ and any $(u, v) \in E$, $\tau(x_{u,v}) = 1$ if $u \geq v$, and $\tau(x_{u,v}) = 0$ if $u \leq v$. Since C is a vertex cover, this definition is not contradictory. All the other variables (if any) are set arbitrarily. τ contradicts all the clauses C_u such that $u \notin C$, and so its measure is at most the measure of C. •

Lemma 4.68 *For any polynomial* p, Min Weight Sat^p $\leq^1_{\mathrm{AP}}$ Min Weight Vertex Cover^p.

[2] From the proof of these two lemmas it is also possible to show that Min Sat is equivalent to its restricted version where each variable occurs exactly twice, once positively and once negatively. Note that a similar result is not known to hold for Max Sat.

PROOF: Let $(C_1, \ldots, C_m, w_1, \ldots, w_m$ be an instance of MIN WEIGHT SAT^p. We define a graph $G = (V, E)$ as follows. For any clause C_j of weight w_j, we have a node u_j of weight w_j. Two nodes u_j and u_h are connected if there is a variable x_i such that x_i occurs in C_j and $\neg x_i$ occurs in C_h.

Given a vertex cover C for G, we define the assigment τ as follows: if x_i (resp. $\neg x_i$) occurs in C_j and $u_j \notin C$, then $\tau(x_i) = 0$ (resp. $\tau(x_i) = 1$). This is always possible since if x_i occurs in C_j and $\neg x_i$ occurs in C_h then x_j and x_h are adjacent in G and at most one of them can be out of C. All the other variables (if any) are set arbitrarily. τ contradicts all clauses C_j such that $u_j \notin C$, and so its measure is no larger than the measure of C.

Conversely, given an assignment τ, we have that $C = \{u_j : \tau \text{ satisfies } C_j\}$ is a vertex cover for G and has the same measure of τ (see the proof of the previous lemma). •

4.6.2 Removing Weights

Lemma 4.69 *For any polynomial p,* MIN WEIGHT VERTEX COVER^p $\leq^1_{\mathrm{AP}}$ MIN VERTEX COVER.

PROOF: Let $G = (V, E)$ be a graph and $\omega : V \to N$ be a polynomially bounded weight function for its nodes. Let us define the graph $G^\omega = (V^\omega, E^\omega)$ as follows. For any node $u \in V$ such that $\omega(u) = w$, V^ω contains w distinct nodes $u^1, \ldots, u^w$. If $(u, v) \in E$ is an edge of G, then E^ω contains all the edges (u^i, v^j) for $i = 1, \ldots, \omega(u)$ and $j = 1, \ldots, \omega(v)$. We note that, given G and ω, G^ω can be constructed in polynomial time.

Let C be a solution for (G, ω) whose measure is c. Then there exists a solution for G^ω of size c. Indeed, we can simply consider the set $C^\omega = \{u^i \in V^\omega : u \in C \wedge i = 1, \ldots, \omega(u)\}$.

On the other hand, let C be a solution for G^ω of size c. From C, we can recover in polynomial time a solution C' for (G, ω) whose measure is at most c. To this aim, assume that a node $u \in V$ and two indices $i, j \in \{1, \ldots, \omega(u)\}$ exist such that $u^i \in C$ and $u^j \notin C$. Then, $C - \{u^i\}$ is still a feasible solution for G^ω. Indeed, u^i and u^j have the same neighborhood, and since $u^j \notin C$ it follows that all the nodes that are adjacent to u^i belong to C and thus there is no need of taking u^i in the cover. By an easy induction argument, it follows that there exists a subset C^ω of C such that for any $u \in U$ either all its copies belong to C^ω, or none does. C^ω can be transformed into a solution C' for (G, ω) of measure $|C^\omega| \leq c$. •

From the above three lemmas, we have the following result.

Theorem 4.70 *The approximability threshold of the four problems* MIN (WEIGHT) SAT *and* MIN (WEIGHT) VERTEX COVER *is the same.*

PROOF: From Corollary 4.52 it follows that a polynomial p exists such that MIN WEIGHT SAT^p and MIN SAT have the same approximability threshold and MIN WEIGHT VERTEX COVER^p and MIN VERTEX COVER have the same approximability threshold. The above three lemmas also imply that MIN WEIGHT SAT^p $\equiv^1_{\mathrm{AP}}$ MIN WEIGHT VERTEX COVER^p $\equiv^1_{\mathrm{AP}}$ MIN VERTEX COVER$\equiv^1_{\mathrm{AP}}$ MIN SAT. The theorem thus follows. •

4.6.3 Bounding the Degree

In this section we will show that an important ingredient in the proof of Theorem 2.22 is the fact that (in a sense to be made precise later) an FGLSS graph contains a lot of bipartite complete subgraphs. Our technical goal is to replace complete bipartite graphs with sparse bipartite graphs which preserve a sufficiently good "connectivity" property. In which follows we will define this particular kind of graphs and we will show its existence and how to generate them deterministically.

Definition 4.71 (Switcher) *Let ε be a positive number. A bipartite graph $G = (V_1, V_2, E)$ is an (n_1, n_2, ε)-switcher if the following holds:*

1. $|V_1| = n_1$, $|V_2| = n_2$;

2. *for any vertex cover* C *of* G, *either* $|V_1 - C| \leq \varepsilon|C|$ *or* $|V_2 - C| \leq \varepsilon|C|$.

Roughly speaking, a switcher is such that any of its vertex covers has to choose almost all the nodes in at least one component. It is worth noting that a bipartite complete graph over components of size n_1 and n_2 is an $(n_1, n_2, 0)$-switcher. As will be shown later, bipartite complete graphs are used in the proof of Theorem 2.22 because of their perfect switching properties.

We now state two results (whose proofs are in Section 4.7) about the existence and the explicit construction of bounded-degree graphs with good switching properties. For a real $k > 0$ we say that a graph is *everywhere k-sparse* if any subset of nodes induce a subgraph with average degree at most k.

Lemma 4.72 (Existence of switchers) *A constant* $c > 0$ *exists such that for any* $\varepsilon > 0$, *for any* $k > c(1/\varepsilon)\log(1/\varepsilon)$ *and for any* n_1, n_2, *a* $2k$*-everywhere sparse* (n_1, n_2, ε)*-switcher with at most* $k(n_1 + n_2)$ *edges exists.*

Lemma 4.73 (Deterministic construction of switchers) *A constant* $c > 0$ *exists such that, for any* $\varepsilon > 0$ *and any* n_1, n_2 *such that* $n_1 \geq n_2$, *an* (n_1, n_2, ε)*-switcher with maximum degree* $k \leq c(n_1 + n_2)/n_2\varepsilon^2$ *exists and is constructible in polynomial time.*

Theorem 4.74 (Non-approximability of MIN VERTEX COVER-B **)** *Let us assume that* $\mathsf{NP} \subseteq \mathsf{FPCP}_{c,s}[\log, f, q]$. *Then for any* $\varepsilon > 0$ *a constant* $B = O(q^4/\varepsilon^3)$ *exists such that it is* NP*-hard to approximate the* MIN VERTEX COVER *problem on graphs with maximum degree* B *to within*

$$1 + \frac{c - s}{2^f - c} - \varepsilon \ .$$

PROOF: Let φ be an instance of SAT, and let us consider the FGLSS graph $G_\varphi = (V_\varphi, E_\varphi)$. This graph has the following characterization. Let l be the length of the proof accessed by the verifier; for any $i = 1, \ldots, l$, let $\pi[i]$ be the i-th bit of the proof π, and let $U[i]$ (respectively, $Z[i]$) be the set of nodes of the graph corresponding to accepting configurations in which $\pi[i] = 1$ (respectively, $\pi[i] = 0$). Let also $\mathbf{u}[i] = |U[i]|$ and $\mathbf{z}[i] = |Z[i]|$. Finally, let $u_i^{(j)}$ (respectively, $z_i^{(j)}$) be the j-th element of $U[i]$ (respectively, of $Z[i]$) in lexicographic order. Given two configurations (vertices) u and v, there is an edge between u and v if and only if they are inconsistent, that is, if and only if for some i they read the same bit $\pi[i]$ and expect it to have different values. Then, we can characterize the edge set of G_φ as

$$E_\varphi = \bigcup_{i=1}^{l} \{(u_i^{(j)}, z_i^{(k)}) : (j,k) \in K_{\mathbf{u}[i],\mathbf{z}[i]}\} \ ,$$

where, for any n_1 and n_2, K_{n_1,n_2} is the edge set of the bipartite complete graph with vertex components $\{1, \ldots, n_1\}$ and $\{1, \ldots, n_2\}$. Moreover, any node u of V_φ belongs to at most q sets $U[i], Z[i]$. We can thus see E_φ as the union of bipartite complete graphs, i.e. graphs with the best possible switching properties. We shall now show that indeed constant degree switchers are sufficient. Without loss of generality, we assume that for any $i = 1, \ldots, l$, $\mathbf{u}[i] \geq \mathbf{z}[i]$ (otherwise, we can invert the value of the i-th bit of the proof in any configuration and then swap the values of $\mathbf{u}[i]$ and $\mathbf{z}[i]$). Let γ be a constant to be fixed later such that $1/\gamma = O(q/\varepsilon)$. Let I be the set of bits i such that $\mathbf{z}[i] \geq \gamma(\mathbf{z}[i] + \mathbf{u}[i])$. For any n_1 and for any n_2, let S_{n_1,n_2} be the set of edges of an (n_1, n_2, γ)-switcher (we assume that the vertex sets are $\{1, \ldots, n_1\}$ and $\{1, \ldots, n_2\}$). We define a graph $G'_\varphi = (V_\varphi, E'_\varphi)$ with the same vertex set of G_φ and with edge set

$$E'_\varphi = \bigcup_{i\in I}\{(u_i^{(j)}, z_i^{(k)}) : (j,k) \in S_{\mathbf{u}[i],\mathbf{z}[i]}\} \ .$$

Lemma 4.73 implies that, for any $i \in I$, we can construct a $(\mathbf{u}[i], \mathbf{z}[i], \gamma)$-switcher with degree bounded by

$$O((\mathbf{u}[i] + \mathbf{z}[i])/\mathbf{z}[i]\gamma^2) = O(1/\gamma^3) \ .$$

Since any node belongs to at most q sets $U[i]$, $Z[i]$, and we assumed that $\gamma = O(\varepsilon/q)$, it follows that G'_φ has degree bounded by $O(q^4/\varepsilon^3)$.

We shall now show how to convert any vertex cover for G'_φ into a "slightly larger" vertex cover for G_φ. We claim that from any vertex cover C' in G'_φ we can recover a vertex cover C in G_φ such that $|C| \leq |C'|(1 + q\gamma) + q\gamma n$. Indeed, let $C'' = \bigcup_{i \notin I} Z[i]$: from the definition of I, it follows that

$$|C''| \leq \sum_{i \notin I} \gamma(\mathbf{z}[i] + \mathbf{u}[i]) \leq q\gamma n \ .$$

Moreover, for any $i \in I$, let $C'[i] = C' \cap (U[i] \cup Z[i])$ be the set of nodes of C' that are used to cover the configurations where the i-th bit has been read. By adding at most $\gamma C'[i]$ nodes, we have a cover that comprises either all nodes of $U[i]$ or all nodes of $Z[i]$. Let $C[i]$ be this new set. Clearly, $C = C'' \cup \bigcup_{i=1}^{l} C[i]$ is a vertex cover for G_φ, and we have that

$$|C| - |C'| \leq q\gamma n + \sum_i |C[i]| - |C'[i]| \leq q\gamma n + q\gamma|C'|$$

where $n = \mathbf{r}2^f$. If φ is not satisfiable, then

$$\mathsf{opt}(G'_\varphi) \geq \frac{1}{1+q\gamma}\mathsf{opt}(G_\varphi) - \gamma 2^f q\mathbf{r} \geq \mathbf{r}(2^f - s)\left(\frac{1}{1+q\gamma} - q\gamma\frac{2^f}{2^f - s}\right) \ .$$

Furthermore, G'_φ is an edge-subgraph of G_φ, thus any vertex cover for G_φ is also a vertex cover for G'_φ. It follows that if φ is satisfiable, then

$$\mathsf{opt}(G'_\varphi) \leq \mathsf{opt}(G_\varphi) \leq \mathbf{r}(2^f - c) \ .$$

By letting γ be sufficiently small (but such that $1/\gamma = O(q/\varepsilon)$), the theorem follows. •

Using the same technique applied in the proof of Theorem 4.74 we can prove the following result. The main difference with respect to the proof of Theorem 4.74 is that this time we use sparse switchers whose existence is guaranteed by Lemma 4.72.

Theorem 4.75 *Let us assume that* $\mathsf{NP} \subseteq \mathsf{FPCP}_{c,s}[\log, f, q]$. *Then for any* $\varepsilon > 0$ *a constant* $k = O((q^2/\varepsilon)\log q/\varepsilon)$ *exists such that the* MIN VERTEX COVER *problem restricted to everywhere* k*-sparse graphs is not approximable within*

$$1 + \frac{c - s}{2^f - c} - \varepsilon$$

unless $\mathsf{NP} \subseteq \mathsf{P}/\text{poly}$.

PROOF: The proof is very similar to that of Theorem 4.74, the only difference being the use of better switchers whose existence is proved in Lemma 4.72. In particular, for any i, we use an everywhere k/q-sparse $(\mathbf{u}[i], \mathbf{z}[i], \gamma)$-switcher which exists provided that $k/q = O((1/\gamma)\log 1/\gamma)$. It follows that G'_φ is everywhere k-sparse where $k = O(q(1/\gamma)\log 1/\gamma)$, that is, $k = O((q^2/\varepsilon)\log q/\varepsilon)$. We can now repeat the same analysis as in the proof of Theorem 4.74. Since we are not able to explicitly construct such switchers, we assume that the reduction receives them as *polynomial size advice*. Thus, instead of a polynomial-time reduction we use a P/poly reduction, and this allows us to prove hardness results under the hypothesis that $\mathsf{NP} \not\subseteq \mathsf{P}$/poly. •

Recall that if $\mathsf{NP} \subseteq \mathsf{P}$/poly then the polynomial hierarchy collapses [98].

Our techniques also yield results regarding the approximability of the MIN VERTEX COVER problem on graphs having a non-linear number of edges.

An interesting consequence of Theorem 4.74 is the fact that any lower bound proved with the PCP technique for the MIN VERTEX COVER problem on general graphs extends *without any loss* to graphs with maximum degree bounded by *any* (thus even very slow) increasing function.

Corollary 4.76 (of Theorem 4.74) *Let $h : \mathcal{Z}^+ \to \mathcal{Z}^+$ be a computable function such that $\lim_n h(n) = \infty$, let $\mathsf{NP} \subseteq \mathsf{FPCP}_{c,s}[\log, f, q]$. Then for any $\varepsilon > 0$ the* MIN VERTEX COVER *problem restricted to graphs with maximum degree $h(n)$ is NP-hard to approximate within*

$$1 + \frac{c-s}{2^f - c} - \varepsilon\ .$$

PROOF: For any $\varepsilon > 0$, Theorem 4.74 implies that a constant B_ε exists such that the MIN VERTEX COVER problem is NP-hard to approximate within $1 - \varepsilon + (c-s)/(2^f - c)$ when restricted to graph with maximum degree B_ε. It is clear that the MIN VERTEX COVER problem restricted to graphs with maximum degree B_ε is reducible to the MIN VERTEX COVER problem restricted to graphs with maximum degree $h(n)$. Indeed, the two problems just differ on a finite number of instances. •

4.7 Mixing Graphs, Mixing Sets and Switchers

4.7.1 Construction of Mixing Graphs

We first recall the following folklore result about the mixing properties of expander graphs.

Lemma 4.77 (see e.g. [75]) *Let $G = (V, E)$ be a d-regular graph, let λ be an upper bound on the absolute values of all the eigenvalues, save the biggest one, of the adjacency matrix of G, let A and B be any two disjoint sets of nodes, then*

$$\left| Cut(A, B) - \frac{d}{2|V|}|A||B| \right| \leq \lambda\sqrt{|A||B|}.$$

As is well known, the largest eigenvalue of the adjacency matrix of a d-regular graph is equal to d, thus we need graphs with second largest eigenvalue much smaller than d. Ramanujan graphs meet our requirement.

Theorem 4.78 ([114], Theorem 4.4) *Let p and q be primes such that $p \equiv q \equiv 1 \pmod 4$ and $q > p$. Then there exists a graph $Y^{p,q}$ that is $p+1$-regular, has $q+1$ nodes, and such that the second largest eigenvalue of its adjacency matrix is at most $2\sqrt{p}$.*

Dirichlet's stronger version of the Prime Number Theorem helps understanding the density of primes of the form $4k+1$. Let $\pi_{b,c}(n)$ be the number of primes $p \leq n$ such that $p \equiv c \pmod b$, and let $\varphi(n)$ be the Euler function, then the following theorem holds.

Theorem 4.79 (Prime Numbers Theorem) *If b and c are co-prime, then*

$$\lim_{n\to\infty} \frac{\pi_{b,c}(n)}{n/\ln n} = \frac{1}{\varphi(b)}.$$

The above theorem is due to Dirichlet, Hadamard, and de la Vallèe Poussin, and a proof can be found in [53]. Let now $l_{b,c}(n)$ be the smallest prime $p \geq n$ such that $p \equiv c \pmod b$. From the above Theorem, it is an easy exercise to show that

$$\lim_{n\to\infty} \frac{l_{4,1}(n)}{n} = 1.$$

Said another way, for any ε a constant n_0 exists such that for any $n \geq n_0$ a prime $n \leq p \leq n(1+\varepsilon)$ exists such that $p \equiv 1 \pmod 4$.

We are now able to prove Theorem 4.54.

Proof: [Of Theorem 4.54] Let n_0 be such that for any $m \geq 2n_0 - 1$ a prime $p \equiv 1 \pmod 4$ exists such that $m \leq p \leq m(1+\varepsilon/2)$. Let $c = \max\{n_0, 32/(\varepsilon^2\delta^2)\}$.

Consider any $n \geq n_0$ and any w such that $cn \leq w \leq n^2$. If $w > n^2/(1+\varepsilon/2)$ then the bipartite graph $\mathrm{K}_{n,n}$ is clearly $(n, w, \varepsilon, \delta)$-mixing. Otherwise, the above conditions imply the existence of two primes $p \equiv q \equiv 1 \pmod 4$ such that

$$n-1 \leq q \leq (n-1)(1+\varepsilon/2) \leq n(1+\varepsilon/2) - 1,$$

$$2w/n - 1 \leq p \leq (2w/n-1)(1+\varepsilon/2) \leq 2w/n(1+\varepsilon/2) - 1.$$

That is,

$$\frac{2w}{n^2(1+\varepsilon/2)} \leq \frac{p+1}{q+1} \leq \frac{2w(1+\varepsilon/2)}{n^2}$$

and thus

$$\left|\frac{p+1}{2(q+1)} - \frac{w}{n^2}\right| \leq \frac{\varepsilon}{2}\frac{w}{n^2}. \tag{4.6}$$

Consider now the graph $Y^{p,q}$, let V_1, V_2 be any two disjoint sets of n vertices of $Y^{p,q}$. Let E be the set of edges of $Y^{p,q}$ that have an endpoint in V_1 and an endpoint in V_2, we shall prove that the bipartite graph $G = (V_1, V_2, E)$ is $(n, w, \varepsilon, \delta)$-mixing. Let $A \subseteq V_1$ and $B \subseteq V_2$ be any two sets of at least δn nodes, let $a = |A|$, $b = |B|$. Applying Lemma 4.77 we have that

$$\left|Cut(A,B) - \frac{p+1}{2(q+1)}ab\right| \leq 2\sqrt{pab}$$

and, combining with (4.6),

$$\begin{aligned}
\left|Cut(A,B) - \frac{w}{n^2}ab\right| &\leq \frac{\varepsilon}{2}\frac{w}{n^2}ab + 2\sqrt{pab} \\
&\leq \frac{\varepsilon}{2}\frac{w}{n^2}ab + 2\sqrt{\frac{2w}{n}ab} \\
&\leq \frac{w}{n^2}ab\left(\frac{\varepsilon}{2} + 2\sqrt{\frac{2n^3}{wab}}\right) \\
&\leq \frac{w}{n^2}ab\left(\frac{\varepsilon}{2} + 2\sqrt{\frac{2n^3}{cn\delta^2 n^2}}\right) \\
&\leq \frac{w}{n^2}ab\left(\frac{\varepsilon}{2} + \frac{\varepsilon}{2}\right).
\end{aligned}$$

•

4.7.2 Construction of Mixing Sets

Proof:[Of Theorem 4.61] We first prove the theorem assuming that $k = 2^h$ is a power of two. We proceed by induction on h. For $h = 1$ the theorem follows from Theorem 4.54. Assume that the theorem holds for $k = 2^h$ and consider the case $2k = 2^{h+1}$. Let $\varepsilon, \delta > 0$ be arbitrary positive constants, let $\bar{\varepsilon}, \bar{\delta}$ be greater than zero and such that

$$\frac{(1+\bar{\varepsilon})^3}{(1-\bar{\varepsilon})^2} - 1 \leq \varepsilon \ , \ 1 - \frac{(1-\bar{\varepsilon})^3}{(1+\bar{\varepsilon})^2} \leq \varepsilon \text{ and } \bar{\delta} \leq \frac{(1-\bar{\varepsilon})}{1+\bar{\varepsilon}}\delta^k.$$

The inductive hypothesis ensures the existence of constants c, n_0 such that for any $n, m > n_0$ and any $w > cm$, a 2^h-ary $(n, cn, \bar{\varepsilon}, \delta)$-mixing set and a binary $(m, w, \bar{\varepsilon}, \bar{\delta})$-mixing set exist. Let $n \geq n_0$ and $w \geq c^2(1+\bar{\varepsilon})n$ be integers, let S be a k-ary $(n, cn, \bar{\varepsilon}, \delta)$-mixing set with m elements and let B be a binary $(m, w, \bar{\varepsilon}, \bar{\delta})$-mixing set. Note that B exists since

$$w \geq c^2(1+\bar{\varepsilon})n \geq cm.$$

We can fix an arbitrary (say, lexicographic) order among the tuples of S, that is a bijection between S and $[m]$. Using this bijection, we can see the pairs of B as $2k$-tuples over $[n]$. Under this mapping, we are going to show that B is a $2k$-ary $(n, w, \varepsilon, \delta)$-mixing set.

Let $A_1, \ldots, A_{2k} \subseteq [n]$ be any family of $2k$ sets such that, for any $i = 1, \ldots, 2k$ $|A_i| \geq \delta n$. We define two sets T and U such that

$$T = S \cap (A_1 \times \cdots \times A_k) \text{ and } U = S \cap (A_{k+1} \times \cdots \times A_{2k}).$$

We first note that, from the mixing property of S, it follows that

$$(1-\bar{\varepsilon})cn \leq m \leq (1+\bar{\varepsilon})cn,$$

that

$$(1-\bar{\varepsilon})cn\frac{|A_1|\cdots|A_k|}{n^k} \leq |T| \leq (1+\bar{\varepsilon})cn\frac{|A_1|\cdots|A_k|}{n^k},$$

and that

$$(1-\bar{\varepsilon})cn\frac{|A_{k+1}|\cdots|A_{2k}|}{n^k} \leq |U| \leq (1+\bar{\varepsilon})cn\frac{|A_{k+1}|\cdots|A_{2k}|}{n^k}.$$

We observe that $|T|, |U| \geq \bar{\delta}m$ and thus, from the mixing property of B, it follows that

$$(1-\bar{\varepsilon})w\frac{|T||U|}{m^2} \leq |B \cap (T \times U)| \leq (1+\bar{\varepsilon})w\frac{|T||U|}{m^2}.$$

Putting all the pieces together we get

$$\begin{aligned}\frac{(1-\bar{\varepsilon})^3}{(1+\bar{\varepsilon})^2}w\frac{|A_1|\cdots|A_{2k}|}{n^{2k}} &\leq |B \cap (A_1 \times \cdots \times A_{2k})| \\ &\leq \frac{(1+\bar{\varepsilon})^3}{(1-\bar{\varepsilon})^2}w\frac{|A_1|\cdots|A_{2k}|}{n^{2k}}.\end{aligned}$$

To conclude the proof, we note that from a k-ary $(n, w, \varepsilon, \delta)$-mixing set S (where $w \leq n^{k-1}$) we can obtain a $(k-1)$-ary $(n, w, \varepsilon, \delta)$-mixing set S' as follows

$$S' = \{(a_1, \ldots, a_{k-1}) : \exists a_{k+1}.(a_1, \ldots, a_{k-1}, a_k) \in S\}.$$

•

4.7.3 Construction of Switchers

In order to construct switchers, it is useful to restate property (2) in Definition 4.71 in a different way. Let I be any independent set in G, let $A = V_1 \cap I$ and $B = V_2 \cap I$. Then property (2) is equivalent to asking that either

$$|A| \leq \varepsilon(|V_1| + |V_2| - |A| - |B|) ,$$

or

$$|B| \leq \varepsilon(|V_1| + |V_2| - |A| - |B|) .$$

If we consider the counterpositive version of the latter statement, we have that property (2) holds if and only if for any subset $A \subseteq V_1$ and for any subset $B \subseteq V_2$ such that

$$|A|, |B| > \varepsilon(n_1 + n_2 - (|A| + |B|))$$

there is at least one edge in E joining a node in A with a node in B. This turns out to be an expansion property: switchers are indeed a generalization of *OR dispersers*.

Definition 4.80 (OR disperser [129]) *An (n_1, n_2, ε)-disperser is a bipartite graph $G = (V_1, V_2, E)$ such that $|V_1| = n_1$, $|V_2| = n_2$, and for any subsets $A \subseteq V_1$, $B \subseteq V_2$ such that $|A| \geq \varepsilon n_1$ and $|B| \geq \varepsilon n_2$, there is at least one edge having an endpoint in A and an endpoint in B.*

Proposition 4.81 *An $(n_1, n_2, \varepsilon/(1+\varepsilon))$-OR disperser is also an (n_1, n_2, ε)-switcher.*

PROOF: Let $G = (V_1, V_2, E)$ be an $(n_1, n_2, \varepsilon/(1+\varepsilon))$-OR disperser, and let $A \subseteq V_1$ and $B \subseteq V_2$ be such that

$$\begin{array}{rcl} |A| & > & \varepsilon(n_1 + n_2 - (|A| + |B|)) \\ |B| & > & \varepsilon(n_1 + n_2 - (|A| + |B|)) \end{array}$$

Since $|B| \leq n_2$, it follows that $|A| \geq \varepsilon(n_1 - |A|)$, that is,

$$|A| \geq \frac{\varepsilon}{1+\varepsilon} n_1 .$$

Similarly, we can show that

$$|B| \geq \frac{\varepsilon}{1+\varepsilon} n_2 .$$

Consequently, G is an (n_1, n_2, ε)-switcher. •

PROOF: [Of Lemma 4.72] It suffices to show the existence of a (n_1, n_2, γ)-OR disperser where $\gamma = \varepsilon/(1+\varepsilon) \geq \varepsilon/2$. We randomly construct a bipartite graph in the following way. Consider two vertex sets V_1 and V_2 where $|V_1| = n_1$, $|V_2| = n_2$, and $n_1 \geq n_2$. Then for any vertex u of V_1 we choose at random $\lceil (k-1)(n_1+n_2)/n_1 \rceil$ distinct elements of V_2 and we connect u to them. This construction ensures that

$$|E| \leq (k-1)(n_1 + n_2) + n_1 < k(n_1 + n_2) .$$

For any vertex pair $(v_1, v_2) \in V_1 \times V_2$, we have that

$$\mathbf{Pr}[(v_1, v_2) \in E] \geq (k-1)\frac{n_1+n_2}{n_1 n_2} \ .$$

We now provide an upper bound on the probability that the random graph $G = (V_1, V_2, E)$ is not an (n_1, n_2, γ)-OR disperser. This probability will be denoted as $\mathbf{Pr}$[no-disperser]. It is not hard to prove that

$$\mathbf{Pr}[\text{no-disperser}] \ \leq \ \binom{n_1}{\gamma n_1}\binom{n_2}{\gamma n_2}\left(1 - \frac{(k-1)(n_1+n_2)}{n_1 n_2}\right)^{\gamma n_1 \gamma n_2} \ .$$

Using Stirling's approximation for the binomial coefficient and the inequality $(1-x) \leq e^{-x}$ $(x \geq 0)$, we obtain the following inequalities

$$\begin{aligned}
\mathbf{Pr}[\text{no-disperser}] \ &\leq \ \left(\frac{en_1}{\gamma n_1}\right)^{\gamma n_1}\left(\frac{en_2}{\gamma n_2}\right)^{\gamma n_2}\left(1 - \frac{(k-1)(n_1+n_2)}{n_1 n_2}\right)^{\gamma n_1 \gamma n_2} \\
&\leq \ \left(\frac{e}{\gamma}\right)^{\gamma(n_1+n_2)} e^{-\gamma^2(k-1)(n_1+n_2)} \\
&= \ e^{(1-\log\gamma)\gamma(n_1+n_2) - \gamma^2(k-1)(n_1+n_2)} \ .
\end{aligned}$$

If we introduce the condition $\mathbf{Pr}$[no-disperser] $\leq e^{-b} < 1$, where b is a fixed positive constant then we have that

$$\gamma(n_1+n_2)(1 - \log\gamma - \gamma(k-1)) \leq -b \ .$$

That is,

$$k \geq 1 + \frac{1}{\gamma}\left(\log\frac{e}{\gamma} + \frac{b}{\gamma^2(n_1+n_2)}\right) = O\left(\frac{1}{\gamma}\log\frac{1}{\gamma}\right) = O\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right) \ .$$

To show that the resulting graphs are everywhere $2k$-sparse, we note that the nodes in the component V_1 have degree at most $k(n_1+n_2)/n_1 \leq 2k$. Thus, given any set $W = W_1 \cup W_2$ of nodes (where $W_i = V_i \cap W$ for $i = 1, 2$), it follows that the number of edges in the subgraph induced by W is at most $2k|W_1| < 2k|W|$. •

We shall now consider a deterministic construction that makes again use of Ramanujan graphs.

Proof: [Of Lemma 4.73 It is sufficient to show how to construct an (n_1, n_2, γ)-OR disperser where $\gamma = \varepsilon/(1+\varepsilon) \geq \varepsilon/2$. Let G be a d-regular Ramanujan expander with n nodes, where $n_1 + n_2 \leq n \leq 4(n_1+n_2)$, and $512(n_1+n_2)/n_2\gamma^2 < d \leq 2048(n_1+n_2)/n_2\gamma^2$ (such a graph exists and is constructible in polynomial time [114]). The second largest eigenvalue λ of G is at most $2\sqrt{d-1}$. From Lemma 4.77, we have that the number of edges connecting A and B is at least

$$d\frac{|A||B|}{2n} - \lambda\sqrt{|A||B|} \ .$$

Let us now consider any two sets A, B such that $|A| \geq \gamma n_1$ and $|B| \geq \gamma n_2$.

We have that

$$\begin{aligned}
d\frac{|A||B|}{2n} - \lambda\sqrt{|A||B|} \ &= \ \sqrt{|A||B|d}\left(\frac{\sqrt{|A||B|d}}{2n} - \frac{\lambda}{\sqrt{d}}\right) \\
&> \ \sqrt{|A||B|d}\left(\frac{\sqrt{\gamma n_1 \gamma 512(n_1+n_2)/\gamma^2}}{2n} - 2\right) \\
&\geq \ \sqrt{|A||B|d}\left(\frac{\sqrt{256(n_1+n_2)^2}}{2n} - 2\right) \\
&\geq \ 0 \ .
\end{aligned}$$

Let now V_1 and V_2 be two disjoint sets of nodes of G such that $|V_1| = n_1$ and $|V_2| = n_2$. Let G' be the bipartite subgraph of G induced by the components V_1 and V_2: clearly, G' is an (n_1, n_2, γ)-OR disperser (and thus an (n_1, n_2, ε)-switcher) and its maximum degree is $O((n_1 + n_2)/n_2\varepsilon^2)$. •

4.8 Attributions and Related Works

Gadgets arose in NP-completeness proofs starting with the original works of Cook and Karp [44, 97]. Bellare, Goldreich and Sudan [25] were the first to propose a concrete definition, that is also accompanied by a *cost*. Khanna and Sudan [103] gave a definition that is a bit more formal and that is identical to Definition 4.1 except that they do not consider weights. Constraint satisfaction problems are somehow implicit in [123]; definitions similar to ours have been given by Hunt et al. [85] and by Khanna and Sudan [103].

The use of linear programming to find optimum gadgets, the analysis of the MAX WEIGHT 3SAT and MAX WEIGHT 3CSP algorithms and the lower bounds using duality are from a joint paper with Greg Sorkin, Madhu Sudan and David Williamson [132]. Theorems 4.27 and 4.28 are from [131]. With reference to Theorem 4.20, the previous larger gap between the completeness and soundness achievable reading two bits was obtained starting from the hardness of MAX 2SAT. It was 74/73 in [25], and would be $22/21 - \varepsilon$ in light of Håstad's results. The previous best published analysis of a MAX 3SAT approximation algorithm gave a 1.301 approximation factor [12]; a slightly better bound of 1.2499 could be obtained adapting the techniques of Goemans and Williamson [75]. The previous best approximation factor achievable for MAX 3CSP was 4 [131]. The only previous lower bound on gadget construction is by Bellare, Goldreich and Sudan [25]. They prove that if an α-gadget exists reducing PC_0 to E3SAT, and the total weight of the clauses of the gadget is α, then $\alpha \geq 4$. They use probabilistic arguments.

Reductions of weighted problems to their non-weighted versions are from a joint paper with Pierluigi Crescenzi and Riccardo Silvestri [49]. There is no comparable previous result, except for MAX WEIGHT CUT: Bellare, Goldreich and Sudan [25] give a reduction where each weighted edge of (integer) weight w is replaced by w length-three disjoint paths. w auxiliary vertices are introduced. The reduction works as a 3-gadget and considerably worsens non-approximability results. There was no previous result for MAX WEIGHT SAT versus MAX SIMPLE SAT and MAX WEIGHT 2SAT versus MAX SIMPLE 2SAT. In the same paper [49] we also proved the equivalence between MIN SAT and MIN VERTEX COVER (also independently obtained in [117, 82]) and between them and their weighted versions.

The reduction from PCP verifiers with low free bit complexity to MIN VERTEX COVER in bounded degree graph is from a joint paper with Andrea Clementi [42]. Previous results include the Max SNP-hardness of MIN VERTEX COVER in graphs of maximum degree 4 [123] and 3 [28].

Chapter 5

Conclusions and Open Questions

Plan, *v.t. To bother about the best method of accomplishing an accidental result.*

(A. Bierce [31])

This thesis dealt with the hardness of combinatorial approximation problems. *Structural* results, *non-approximability* results and *algorithmic* results have been presented. Unifying themes were the use of new types of approximation-preserving reducibilities, designed to overcome inherent limitations of previously known ones. To prove completeness results in natural approximation classes we defined a new kind of *non-oblivious* reduction that *knows the approximation factor it is expected to preserve*. By making use of the generality of our new definition of reducibility we proved a completeness result without using PCP machinery. To understand the properties of complete problems in approximation classes and the limitations of previous reducibilities we used *query complexity* arguments, borrowing results and proof-techniques from Structural Complexity Theory.

We showed that *linear programming* could be used to find computer-constructed provably optimal *local replacement* reductions. We showed how to extend non-approximability results from weighted problems to their non-weighted versions by using in a novel way expander graphs and the probabilistic method. We gave a new logical formulation of the MIN VERTEX COVER problem and showed how to use expander graphs to prove a strong non-approximability result for its restriction to bounded-degree graphs.

Our PCP-free completeness result and some of our optimal local reductions yield the best known approximation algorithms for MAX PLANAR GSAT-B, MAX 3SAT, and MAX 3CSP.

So many questions are still to be settled in the field of approximation theory. In the following we try to list a few of them.

Structural Results

Nicer Characterization of Complete Problems. The characterizations of polyAPX and APX-complete problems given in Section 3.4 are not as natural as one would like them to be. In particular, for polyAPX-complete problems it would be nicer to have as conditions the $\mathsf{NPF}^{\mathsf{NP}[1]}$-hardness of A_{n^ε} for some $\varepsilon > 0$ and then a combinatorial property analogous to self-improvability. Even better, having only the hardness of A_2 and then use to the self-improvability to boost the gap, as is done in practice for MAX CLIQUE. Several technical difficulties prevented us from doing that, but we hope that they can be overcome. We also don't know if there is a nice characterization of APX PB-complete problems under E-reductions. Such a characterization would shed new light on the relation between the E-reducibility and the AP-reducibility.

logAPX-complete Problems. We still lack a characterization of logAPX-complete problems. It is also still to be completely understood why logAPX-complete problems are not self-improvable. Presumably, a PCP-free proof of the fact that MIN SET COVER is not self-improvable can be done.

PCP Theorem vs. the Hardness of MAX CLIQUE. Randomized graph product shows that if the PCP Theorem holds even with two-sided error verifiers then MAX CLIQUE is hard to approximate within n^{δ} for some $\delta > 0$. We don't know a reverse connection. Bellare, Goldreich and Sudan show that the hardness of approximating MAX CLIQUE is "equivalent" to the existence of PCP verifiers for NP with low amortized free bits complexity. Their reverse connection is tight with respect to the non-approximability factors. We would like to have a weaker reverse connection with respect to the query complexity parameter. Equivalently, we would like a simple and direct reduction stating that for any $r > 1$, a $\delta_r > 0$ exists such that approximating MAX SAT within r is reducible to approximating MAX CLIQUE within n^{δ_r}.

Minimization Problems

poly $\log(n)$-approximable problems. There is a large class of minimization problems that can be approximated within poly$(\log n)$ factors and that (at most) are only known to be Max SNP-hard. For example MIN EDGE DELETION BIPARTITE SUBGRAPH, that is $O(\log n)$-approximable using the approximate max-flow min-cut theorem [110, 72, 112], and MIN FEEDBACK VERTEX SET, MIN LINEAR ARRANGMENT, MIN CLAUSE DELETION 2SAT, that are $O(\log n \log\log n)$-approximable using the spreading metric approach [57, 56]. The MIN LINEAR ARRANGMENT problem is not even known to be Max SNP-hard. It is reasonable to assume that the latter problems are $O(\log n)$-approximable, perhaps using a refined analysis of the spreading metric approach, and that all the problems are hard to approximate within any fixed constant.

Coloring. Given a graph G and the promise that it is 3-colorable, it is possible to color it with $n^{0.2143}$ colors [95, 32]. On the other hand it is NP-hard to color it with less than 5 colors [100]. It is a widespread believing that it is hard to use even $O(\log n)$ colors, but it would be a good improvement even to prove hardness within any fixed constant. Arora [6] shows that "current techniques" do not suffice for that. He proves similar bounds for MAX CLIQUE and MIN VERTEX COVER. However Håstad's [78] ε free bit verifier (developed with exciting new techniques) beats Arora's bound for MAX CLIQUE. Perhaps the coloring bounds can be beaten as well.

Removing weights. Our expander-based technique to reduce weighted problems to non-weighted versions can be applied to maximization constraint satisfaction problems since they have the property that the optimum is a constant fraction of the number of constraints. For minimization constraint satisfaction problems, there is no such guarantee, and our approach fails. However, for MIN SAT and MIN VERTEX COVER a very simple combinatorial reduction works. We do not know whether, for any of the following problems, the weighted version is AP-equivalent to the unweighted one: MIN SET COVER, MIN LINEAR ARRANGMENT, MIN FEEDBACK VERTEX SET, MIN CLAUSE DELETION 2SAT.

Non-approximability Results to be Improved

Tight results to come. Håstad [80] proves that it is hard to approximate MAX 3SAT to within $8/7 - \varepsilon$; the best positive result (see Section 4.3) says that a 5/4 approximation is possible. A tight result is in sight, and it may well be that the right threshold is 8/7. A better algorithm (closing the gap) may involve ideas related to the approximation-preserving reductions of this thesis. For MIN VERTEX COVER, the best algorithm achieves a factor of 2 and Håstad's non-approximability result is $7/6 - \varepsilon$. A hardness result that uses Theorem 2.22 and a PCP verifier with perfect completeness cannot be better than $3/2 - \varepsilon$ (this is because a verifier with perfect completeness has to use at least $\log 3$ free bits, see [25] and the discussion in [24]). Current bets are that the right threshold is 2, and significantly new ideas are needed to achieve such a hardness results.

Non-approximability Result for Practical Problems. There is no known explicit non-approximability result for metric MIN TSP, MIN STEINER TREE and shortest common super-sequence. Gadget reductions cannot (seemingly) be used since such problems lack a clean logical definition. The APX-completeness of MIN STEINER TREE is proved using a reduction from MIN VERTEX COVER in bounded-degree graphs [30] (actually, sparse graphs are good enough). To have a reasonable bound for this problem, an explicit hardness result is needed for, say MIN VERTEX COVER-4 or MIN VERTEX COVER-10. It is not clear how to get for this problems a bound close to the one guaranteed by Theorem 2.22. To rule out the possibility of using alternative approaches, a *reverse reduction* from metric MIN STEINER TREE to MIN VERTEX COVER-B would be very interesting too.

Algorithmic Applications

Approximation Schemes. Our reduction technique described in Section 3.3 implies efficient PTAS's for the planar versions of MAX SAT and MAX GSAT-B. Khanna and Motwani [101] gave approximation schemes for a larger class of logically defined problems: we don't know if an improved reduction can yield an approximation scheme for other problems as well.

Unary Constraints. We would like to prove that the approximation threshold of MAX SIMPLE SAT is equal to that of MAX WEIGHT SAT. One obstacle to prove this result is the fact that SAT constraints can be arbitrarily large, but we can use Lemma 3.24 to bring the size of the disjunctions down to a constant. The most difficult part is how to get rid of *unary* constraints. A reduction from MAX WEIGHT SAT to MAX WEIGHT SAT without unary constraints (but also from MAX WEIGHT 3SAT to MAX WEIGHT 3SAT without unary constraints) would have algorithmic relevance since unary constraints are a bottleneck in the approximation analyses in [75] and in Section 4.3.

Bibliography

[1] ADAMS, D. *The Hitchhiker's Guide to the Galaxy.* Harmony Books, 1980.

[2] ADAMS, D. *The Restaurant at the End of the Universe.* Pan Books, 1980.

[3] ALIMONTI, P. Non-oblivious local search for graphs and hypergraphs coloring problems. In *Proceedings of the 21th Workshop on Graph-Theoretic Concepts in Computer Science* (1995), LNCS 1017, Springer-Verlag, pp. 167–180.

[4] ALON, N., AND SPENCER, J. *The Probabilistic Method.* Wiley Interscience, 1992.

[5] ARORA, S. *Probabilistic Checking of Proofs and Hardness of Approximation Problems.* PhD thesis, University of California at Berkeley, 1994.

[6] ARORA, S. Reductions, codes, PCP's and inapproximability. In *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science* (1995), pp. 404–413.

[7] ARORA, S. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science* (1996), pp. 2–11.

[8] ARORA, S., AND LUND, C. Hardness of approximations. In *Approximation Algorithms for NP-hard Problems* (1996), PWS Publishing.

[9] ARORA, S., LUND, C., MOTWANI, R., SUDAN, M., AND SZEGEDY, M. Proof verification and hardness of approximation problems. In *Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science* (1992), pp. 14–23.

[10] ARORA, S., AND SAFRA, S. Probabilistic checking of proofs; a new characterization of NP. In *Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science* (1992), pp. 2–13.

[11] ARORA, S., AND SUDAN, M. Improved low degree testing and its applications. In *Proceedings of the 29th ACM Symposium on Theory of Computing* (1997), pp. 485–495.

[12] ASANO, T., ONO, T., AND HIRATA, T. Approximation algorithms for the maximum satisfiability problem. In *Proceedings of the 5th Scandinavian Workshop on Algorithm Theory* (1996), LNCS 1097, Springer-Verlag, pp. 100–111.

[13] AUSIELLO, G., CRESCENZI, P., AND PROTASI, M. Approximate solution of NP optimization problems. *Theoretical Computer Science 150*, 1 (1995), 1–55.

[14] AUSIELLO, G., D'ATRI, A., AND PROTASI, M. Structure preserving reductions among convex optimization problems. *Journal of Computer and System Sciences 21* (1980), 136–153.

[15] Babai, L. Trading group theory for randomness. In *Proceedings of the 17th ACM Symposium on Theory of Computing* (1985), pp. 421–429. See also [18].

[16] Babai, L., Fortnow, L., Levin, L., and Szegedy, M. Checking computations in polylogarithmic time. In *Proceedings of the 23rd ACM Symposium on Theory of Computing* (1991), pp. 21–31.

[17] Babai, L., Fortnow, L., and Lund, C. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity 1* (1991), 3–40. Preliminary version in *Proc. of FOCS'90.*

[18] Babai, L., and Moran, S. Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences 36* (1988), 254–276.

[19] Baker, B. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM 41*, 1 (1994), 153–180.

[20] Balcázar, J., Díaz, J., and Gabarró, J. *Structural Complexity I.* Springer-Verlag, 1988.

[21] Beherendt, T., Compton, K., and Grädel, E. Optimization problems: expressibility, approximation properties and expected asymptotic growth of optimal solutions. In *Proceedings of the 6th Workshop of Computer Science and Logic* (1993), LNCS 702, Springer-Verlag, pp. 43–60.

[22] Beigel, R. Bounded queries to SAT and the boolean hierarchy. *Theoretical Computer Science 84* (1991), 199–223.

[23] Bellare, M. Interactive proofs and approximation: Reductions from two provers in one round. In *Proceedings of the 2nd IEEE Israel Symposium on Theory of Computing and Systems* (1993). Preliminary version in IBM Research Report RC17969 (1992).

[24] Bellare, M. Proof checking and approximation: Towards tight results. *Sigact News 27*, 1 (1996).

[25] Bellare, M., Goldreich, O., and Sudan, M. Free bits, PCP's and non-approximability – towards tight results (4th version). Tech. Rep. TR95-24, Electronic Colloquium on Computational Complexity, 1996. Preliminary version in *Proc. of FOCS'95.*

[26] Bellare, M., Goldwasser, S., Lund, C., and Russell, A. Efficient probabilistically checkable proofs and applications to approximation. In *Proceedings of the 25th ACM Symposium on Theory of Computing* (1993), pp. 294–304. See also the errata sheet in *Proc of STOC'94.*

[27] Bellare, M., and Sudan, M. Improved non-approximability results. In *Proceedings of the 26th ACM Symposium on Theory of Computing* (1994), pp. 184–193.

[28] Berman, P., and Fujito, T. On the approximation properties of the independent set problem in degree 3 graphs. In *Proceedings of the 4th Workshop on Algorithms and Data Structures* (1995), LNCS 955, Springer-Verlag, pp. 449–460.

[29] Berman, P., and Schnitger, G. On the complexity of approximating the independent set problem. *Information and Computation 96* (1992), 77–94. Preliminary version in *Proc. of STACS'89.*

[30] Bern, M., and Plassmann, P. The Steiner tree problem with edge lengths 1 and 2. *Information Processing Letters 32* (1989), 171–176.

[31] Bierce, A. *The Devil's Dictionary.* Neale Publishing, 1911.

[32] Blum, A., and Karger, D. An $\tilde{O}(n^{3/14})$ coloring algorithm for 3-colorable graphs. *Information Processing Letters 61*, 1 (1997), 49–53.

[33] Blum, A., Li, M., Tromp, J., and Yannakakis, M. Linear approximation of shortest superstrings. *Journal of the ACM 41*, 4 (1994), 630–647.

[34] Bodlaender, H. Some classes of graphs with bounded treewidth. *Bulletin of the EATCS 36* (1988), 116–126.

[35] Bodlaender, H. A linear time algorithm for finding tree-decompositions of small treewidth. In *Proceedings of the 25th ACM Symposium on Theory of Computing* (1993), pp. 226–234.

[36] Bovet, D., and Crescenzi, P. *Introduction to the Theory of Complexity.* Prentice Hall, 1993.

[37] Bruschi, D., Joseph, D., and Young, P. A structural overview of NP optimization problems. In *Proceedings of Optimal Algorithms* (1989), LNCS 401, Springer-Verlag.

[38] Cai, J., Gundermann, T., Hartmanis, J., Hemachandra, L., Swelson, V., Wagner, K., and Wechsung, G. The boolean hierarchy I: Structural properties. *SIAM Journal on Computing 17*, 6 (1988), 1232–1252.

[39] Chang, R. A machine model for NP-approximation problems and the revenge of the boolean hierarchy. *Bulletin of the EATCS 54* (1994).

[40] Chang, R. On the query complexity of clique size and maximum satisfiability. *Journal of Computer and System Sciences 53*, 2 (1996), 298–313. Preliminary version in *Proc. of Structures'94.*

[41] Chang, R., Gasarch, W. I., and Lund, C. On bounded queries and approximation. *SIAM Journal on Computing 26*, 1 (1997), 188–209. Preliminary version in *Proc. of FOCS'93.*

[42] Clementi, A., and Trevisan, L. Improved non-approximability results for vertex cover with density constraints. In *Proceedings of 2nd Computing and Combinatorics Conference* (1996), LNCS 1090, Springer-Verlag, pp. 333–342.

[43] Condon, A. The complexity of the max-word problem and the power of one-way interactive proof systems. *Computational Complexity 3* (1993), 292–305. Preliminary version in *Proc. of STACS91.*

[44] Cook, S. The complexity of theorem proving procedures. In *Proceedings of the 3rd ACM Symposium on Theory of Computing* (1971), pp. 151–158.

[45] Crescenzi, P., and Kann, V. A compendium of NP optimization problems. Tech. Rep. TR SI-95/02, Università di Roma "La Sapienza", Dipartimento di Scienze dell'Informazione, 1995. Updated on-line version is available at the URL `http://www.nada.kth.se/~viggo/problemlist/compendium.html`.

[46] Crescenzi, P., Kann, V., Silvestri, R., and Trevisan, L. Structure in approximation classes. In *Proceedings of the 1st Combinatorics and Computing Conference* (1995), LNCS 959, Springer-Verlag, pp. 539–548.

[47] Crescenzi, P., and Panconesi, A. Completeness in approximation classes. *Information and Computation 93* (1991), 241–262. Preliminary version in *Proc. of FCT'89.*

[48] Crescenzi, P., and Silvestri, R. Relative complexity of evaluating the optimum cost and constructing the optimum for maximization problems. *Information Processing Letters 33* (1990), 221–226.

[49] Crescenzi, P., Silvestri, R., and Trevisan, L. To weight or not to weight: Where is the question? In *Proceedings of the 4th IEEE Israel Symposium on Theory of Computing and Systems* (1996), pp. 68–77.

[50] Crescenzi, P., and Trevisan, L. On approximation scheme preserving reducibility and its applications. In *Proceedings of 14th Conference on Foundations of Software Technology and Theoretical Computer Science* (1994), LNCS 880, Springer-Verlag, pp. 330–341.

[51] Crescenzi, P., and Trevisan, L. MAX NP-completeness made easy. Manuscript, 1996.

[52] Dahlhaus, E., Johnson, D., Papadimitriou, C., Seymour, P., and Yannakakis, M. The complexity of multi-way cuts. *SIAM Journal on Computing 23* (1994), 864–894.

[53] Davenport, H. *Multiplicative Number Theory.* Springer-Verlag, 1980. Second Edition.

[54] Dijkstra, E. A note on two problems in connexion with graphs. *Numeriske Mathematik 1* (1959), 269–271.

[55] Edmonds, J., and Karp, R. Theoretical improvements in algorithmic efficiency for network flow problem. *Journal of the ACM 19*, 2 (1972), 248–264.

[56] Even, G., Naor, J., Rao, S., and Schieber, B. Divide-and conquer approximation algorithms via spreading metrics. In *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science* (1995), pp. 62–71.

[57] Even, G., Naor, J., Schieber, B., and Sudan, M. Approximating minimum feedback sets and multicuts in directed graphs. In *Proceedings of the 4th Conference on Integer Programming and Combinatorial Optimization* (1995), pp. 14–28.

[58] Fagin, R. Generalized first-order spectra and polynomial-time recognizable sets. In *SIAM-AMS Proceedings* (1974), pp. 43–73.

[59] Feige, U. A threshold of $\ln n$ for approximating set cover. In *Proceedings of the 28th ACM Symposium on Theory of Computing* (1996), pp. 314–318.

[60] Feige, U., and Goemans, M. Approximating the value of two provers proof systems, with applications to MAX 2SAT and MAX DICUT. In *Proceedings of the 3rd IEEE Israel Symposium on Theory of Computing and Systems* (1995), pp. 182–189.

[61] Feige, U., Goldwasser, S., Lovász, L., Safra, S., and Szegedy, M. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM 43*, 2 (1996), 268–292. Preliminary version in *Proc. of FOCS91.*

[62] Feige, U., and Kilian, J. Two prover protocols - low error at affordable rates. In *Proceedings of the 26th ACM Symposium on Theory of Computing* (1994), pp. 172–183.

[63] Feige, U., and Kilian, J. Zero knowledge and the chromatic number. In *Proceedings of the 11th IEEE Conference on Computational Complexity* (1996).

[64] Feige, U., and Lovász, L. Two-prover one round proof systems: Their power and their problems. In *Proceedings of the 24th ACM Symposium on Theory of Computing* (1992), pp. 733–741.

[65] Ford, L., and Fulkerson, D. *Flows in Networks*. Princeton University Press, 1962.

[66] Fortnow, L., Rompel, J., and Sipser, M. On the power of multi-prover interactive protocols. In *Proceedings of the 3th IEEE Conference on Structure in Complexity Theory* (1988), pp. 156–161.

[67] Fürer, M., and Raghavachari, B. Approximating the minimum degree spanning tree to within one from the optimal degree. In *Proceedings of the 3rd ACM-SIAM Symposium on Discrete Algorithms* (1992), pp. 317–324.

[68] Garey, M., and Johnson, D. The complexity of near-optimal graph coloring. *Journal of the ACM 23* (1976), 43–49.

[69] Garey, M., and Johnson, D. Strong NP-completeness results: motivation, examples, and implications. *Journal of the ACM 25* (1978), 499–508.

[70] Garey, M., and Johnson, D. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. Freeman, 1979.

[71] Garey, M. R., Johnson, D. S., and Stockmeyer, L. Some simplified NP-complete graph problems. *Theoretical Computer Science 1* (1976), 237–267.

[72] Garg, N., Vazirani, V., and Yannakakis, M. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM Journal on Computing 25*, 2 (1996), 235–251. Preliminary version in *Proc. of STOC'93*.

[73] Gavril, F. Manuscript cited in [70], 1974.

[74] Goemans, M., and Williamson, D. New 3/4-approximation algorithms for the maximum satisfiability problem. *SIAM Journal on Discrete Mathematics 7*, 4 (1994), 656–666. Preliminary version in *Proc. of IPCO'93*.

[75] Goemans, M., and Williamson, D. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM 42*, 6 (1995), 1115–1145. Preliminary version in *Proc. of STOC'94*.

[76] Goldwasser, S., Micali, S., and Rackoff, C. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing 18*, 1 (1989), 186–208. Preliminary version in *Proc of STOC'85*.

[77] Graham, R. Bounds for certain multiprocessing anomalies. *Bell System Technology Journal 45* (1966), 1563–1581.

[78] Håstad, J. Clique is hard to approximate within $n^{1-\varepsilon}$. In *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science* (1996), pp. 627–636.

[79] Håstad, J. Testing of the long code and hardness for clique. In *Proceedings of the 28th ACM Symposium on Theory of Computing* (1996), pp. 11–19.

[80] Håstad, J. Some optimal inapproximability results. In *Proceedings of the 29th ACM Symposium on Theory of Computing* (1997), pp. 1–10.

[81] HEMASPAANDRA, L. Complexity theory column 5: the not-ready-for-prime-time conjectures. *SIGACT News* (1994).

[82] HOCHBAUM, D. A framework for half integrality and 2-approximations with applications to feasible cut and minimum satisfiability. Manuscript, 1996.

[83] HOCHBAUM, D., AND SHMOYS, D. A best possible heuristic for the k-center problem. *Mathematics of Operations Research 10*, 2 (1985), 180–184.

[84] HOLYER, I. The NP-completeness of edge-coloring. *SIAM Journal on Computing 10* (1981), 718–720.

[85] HUNT III, H. B., MARATHE, M., RADHAKRISHNAN, V., RAVI, S., ROSENKRANTZ, D., AND STEARNS, R. Approximation schemes using L-reductions. In *Proceedings of 14th Conference on Foundations of Software Technology and Theoretical Computer Science* (1994), LNCS 880, Springer-Verlag, pp. 342–353.

[86] IBARRA, O. H., AND KIM, C. E. Fast approximation for the knapsack and sum of subset problems. *Journal of the ACM 22* (1975), 463–468.

[87] IMPAGLIAZZO, R., AND NAOR, M. Decision trees and downward closures. In *Proceedings of the 3th IEEE Conference on Structure in Complexity Theory* (1988), pp. 29–38.

[88] JOHNSON, D. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences 9* (1974), 256–278.

[89] JOHNSON, D. The NP-completeness column: An ongoing guide. *Journal of Algorithms 13* (1992), 502–524.

[90] KADIN, J. The polynomial time hierarchy collapses if the boolean hierarchy collapses. *SIAM Journal of Computing 17*, 6 (1988), 1263–1282.

[91] KADIN, J. Erratum: The polynomial time hierarchy collapses if the boolean hierarchy collapses. *SIAM Journal on Computing 20*, 2 (1990), 404.

[92] KANN, V. *On the Approximability of NP-Complete Optimization Problems.* PhD thesis, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, 1992.

[93] KANN, V. Polynomially bounded minimization problems which are hard to approximate. *Nordic Journal of Computing 1* (1994), 317–331. Preliminary version in *Proc. of ICALP'93.*

[94] KANN, V., AND PANCONESI, A. *Hardness of Approximation.* Annotated bibliographies in combinatorial optimization, 1996. To appear.

[95] KARGER, D., MOTWANI, R., AND SUDAN, M. Approximate graph coloring by semidefinite programming. In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science* (1994), pp. 2–13.

[96] KARMAKAR, N., AND KARP, R. An efficient approximation scheme for the one-dimensional bin packing problem. In *Proceedings of the 23th IEEE Symposium on Foundations of Computer Science* (1982), pp. 312–320.

[97] KARP, R. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, R. Miller and J. Thatcher, Eds. Plenum Press, 1972, pp. 85–103.

[98] Karp, R., and Lipton, R. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 21th IEEE Symposium on Foundations of Computer Science* (1980), pp. 302–309.

[99] Khanna, S. Personal Communication, 1996.

[100] Khanna, S., Linial, N., and Safra, S. On the hardness of approximating the chromatic number. In *Proceedings of the 2nd IEEE Israel Symposium on Theory of Computing and Systems* (1993), pp. 250–260.

[101] Khanna, S., and Motwani, R. Towards a syntactic characterization of PTAS. In *Proceedings of the 28th ACM Symposium on Theory of Computing* (1996), pp. 329–337.

[102] Khanna, S., Motwani, R., Sudan, M., and Vazirani, U. On syntactic versus computational views of approximability. In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science* (1994), pp. 819–830.

[103] Khanna, S., and Sudan, M. The optimization complexity of constraint satisfaction problems. Tech. Rep. TR96-028, Electronic Colloquium on Computational Complexity, 1996. See also [104].

[104] Khanna, S., Sudan, M., and Williamson, D. A complete classification of the approximability of maximization problems derived from boolean constraint satisfaction. In *Proceedings of the 29th ACM Symposium on Theory of Computing* (1997), pp. 11–20.

[105] Kolaitis, P., and Thakur, M. Logical definability of NP optimization problems. *Information and Computation 115*, 2 (1994), 321–353.

[106] Kolaitis, P., and Thakur, M. Approximation properties of NP minimization classes. *Journal of Computer and System Sciences 50* (1995), 391–411. Preliminary version in *Proc. of Structures91*.

[107] Krentel, M. The complexity of optimization problems. *Journal of Computer and System Sciences 36* (1988), 490–509.

[108] Kruskal, J. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the AMS 7* (1956), 48–50.

[109] Lapidot, D., and Shamir, A. Fully parallelized multi prover protocols for NEXPTIME. In *Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science* (1991), pp. 13–18.

[110] Leighton, F., and Rao, S. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *Proceedings of the 29th IEEE Symposium on Foundations of Computer Science* (1988), pp. 422–431.

[111] Levin, L. A. Universal search problems. *Problemi Peredachi Informatsii 9* (1973), 265–266.

[112] Linial, N., London, E., and Rabinovich, Y. The geometry of graphs and some of its algorithmic applications. *Combinatorica 15*, 2 (1995), 215–245.

[113] Long, T. On γ-reducibility versus polynomial time many-one reducibility. *Theoretical Computer Science 14* (1981), 91–101.

[114] Lubotzky, A., Phillips, R., and Sarnak, P. Ramanujan graphs. *Combinatorica 8* (1988), 261–277.

[115] Lund, C., Fortnow, L., Karloff, H., and Nisan, N. Algebraic methods for interactive proof systems. *Journal of the ACM 39*, 4 (1992), 859–868. Preliminary version in *Proc of FOCS'90.*

[116] Lund, C., and Yannakakis, M. On the hardness of approximating minimization problems. *Journal of the ACM 41* (1994), 960–981. Preliminary version in *Proc. of STOC'93.*

[117] Marathe, M., and Ravi, S. On approximation algorithms for the minimum satisfiability problem. *Information Processing Letters 58*, 1 (1996), 23–29.

[118] Orponen, P., and Mannila, H. On approximation preserving reductions: complete problems and robust measures. Tech. Rep. C-1987-28, Department of Computer Science, University of Helsinki, 1987.

[119] Panconesi, A., and Ranjan, D. Quantifiers and approximations. *Theoretical Computer Science 107* (1993), 145–163. Preliminary version in *Proc. of STOC'90.*

[120] Papadimitriou, C. *Computational Complexity.* Addison-Wesley, 1994.

[121] Papadimitriou, C., and Yannakakis, M. On limited nondeterminism and the complexity of the VC dimension. In *Proceedings of the 8th IEEE Conference on Structure in Complexity Theory* (1993), pp. 305–312.

[122] Papadimitriou, C., and Yannakakis, M. The travelling salesman problem with distances one and two. *Mathematics of Operations Research 18* (1993), 1–11.

[123] Papadimitriou, C. H., and Yannakakis, M. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences 43* (1991), 425–440. Preliminary version in *Proc. of STOC'88.*

[124] Paz, A., and Moran, S. Non deterministic polynomial optimization problems and their approximation. *Theoretical Computer Science 15* (1981), 251–277.

[125] Queyranne, M. Bounds for assembly line balancing heuristics. *Operations Research 33* (1985), 1353–1359.

[126] Raz, R., and Safra, S. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th ACM Symposium on Theory of Computing* (1997), pp. 475–484.

[127] Sahni, S., and Gonzalez, T. P-complete approximation problems. *Journal of the ACM 23* (1976), 555–565.

[128] Shamir, A. IP = PSPACE. *Journal of the ACM 39*, 4 (1992), 869–877. Preliminary version in *Proc of FOCS'90.*

[129] Sipser, M. Expanders, randomness or time vs. space. In *Proceedings of the 1st IEEE Conference on Structure in Complexity Theory* (1986), Springer Verlag.

[130] Sudan, M. *Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems.* ACM Distinguished Theses, LNCS 1001, Springer-Verlag, 1995.

[131] Trevisan, L. Positive linear programming, parallel approximation, and PCP's. In *Proceedings of the 4th European Symposium on Algorithms* (1996), LNCS 1136, Springer-Verlag, pp. 62–75.

[132] Trevisan, L., Sorkin, G., Sudan, M., and Williamson, D. Gadgets, approximation, and linear programming. In *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science* (1996), pp. 617–626.

[133] Wagner, K. Bounded query computations. In *Proceedings of 3rd Structure in Complexity Theory Conference* (1988), pp. 260–277.

[134] Wee, T., and Magazine, J. Assembly line balancing as generalized bin packing. *Operations Research Letters 1* (1982), 56–58.

[135] Yannakakis, M. On the approximation of maximum satisfiability. *Journal of Algorithms 17* (1994), 475–502. Preliminary version in *Proc. of SODA'92*.

[136] Zuckerman, D. NP-complete problems have a version that is hard to approximate. In *Proceedings of the 8th IEEE Conference on Structure in Complexity Theory* (1993), pp. 305–312.

Dottorato di Ricerca in Informatica
Collana della tesi
Collection of Theses

V-93-1 Marco Cadoli. *Two Methods for Tractable Reasoning in Artificial Intelligence: Language Restriction and Theory Approximation.* June 1993.

V-93-2 Fabrizio d'Amore. *Algorithms and Data Structures for Partitioning and Management of Sets of Hyperrectangles.* June 1993.

V-93-3 Miriam Di Ianni. *On the complexity of flow control problems in Store-and-Forward networks.* June 1993.

V-93-4 Carla Limongelli. *The Integration of Symbolic and Numeric Computation by p-adic Construction Methods.* June 1993.

V-93-5 Annalisa Massini. *High efficiency self-routing interconnection networks.* June 1993.

V-93-6 Paola Vocca. *Space-time trade-offs in directed graphs reachability problem.* June 1993.

VI-94-1 Roberto Baldoni. *Mutual Exclusion in Distributed Systems.* June 1994.

VI-94-2 Andrea Clementi. *On the Complexity of Cellular Automata.* June 1994.

VI-94-3 Paolo Giulio Franciosa. *Adaptive Spatial Data Handling.* June 1994.

VI-94-4 Andrea Schaerf. *Query Answering in Concept-Based Knowledge Representation Systems: Algorithms, Complexity, and Semantic Issues.* June 1994.

VI-94-5 Andrea Sterbini. *2-Thresholdness and its Implications: from the Synchronization with PVchunk to the Ibaraki-Peled Conjecture.* June 1994.

VII-95-1 Piera Barcaccia. *On the Complexity of Some Time Slot Assignment Problems in Switching Systems.* June 1995.

VII-95-2 Michele Boreale. *Process Algebraic Theories for Mobile Systems.* June 1995.

VII-95-3 Antonella Cresti. *Unconditionally Secure Key Distribution Protocols.* June 1995.

VII-95-4 Vincenzo Ferrucci. *Dimension-Independent Solid Modeling.* June 1995.

VII-95-5 Esteban Feuerstein. *On-line Paging of Structured Data and Multi-threaded Paging.* June 1995.

VII-95-6 Michele Flammini. *Compact Routing Models: Some Complexity Results and Extensions.* June 1995.

VII-95-7 Giuseppe Liotta. *Computing Proximity Drawings of Graphs.* June 1995.

VIII-96-1 Luca Cabibbo. *Querying and Updating Complex-Object Databases.* May 1996.

VIII-96-2 Diego Calvanese. *Unrestricted and Finite Model Reasoning in Class-Based Representation Formalisms.* May 1996.

VIII-96-3 Marco Cesati. *Structural Aspects of Parameterized Complexity.* May 1996.

VIII-96-4 Flavio Corradini. *Space, Time and Nondeterminism in Process Algebras.* May 1996.

VIII-96-5 Stefano Leonardi. *On-line Resource Management with Application to Routing and Scheduling.* May 1996.

VIII-96-6 Rosario Pugliese. *Semantic Theories for Asynchronous Languages.* May 1996.

IX-97-1 Paola Alimonti. *Local search and approximability of MAX SNP problems.* May 1997.

IX-97-2 Tiziana Calamoneri. *Does Cubicity Help to Solve Problems?.* May 1997.

IX-97-3 Paolo Di Blasio. *A Calculus for Concurrent Objects: Design and Control Flow Analysis.* May 1997.

IX-97-4 Bruno Errico. *Intelligent Agents and User Modelling.* May 1997.

IX-97-5 Roberta Mancini. *Modelling Interactive Computing by exploiting the Undo.* May 1997.

IX-97-6 Riccardo Rosati. *Autoepistemic Description Logics.* May 1997.

IX-97-7 Luca Trevisan. *Reductions and (Non-)Approximability.* May 1997.