Maximum And- vs. Even-SAT

Tamio-Vesa Nakajima
Department of Computer Science, University of Oxford, UK

Stanislav Živný
Department of Computer Science, University of Oxford, UK

Abstract

A multiset of literals, called a clause, is *strongly satisfied* by an assignment if *no* literal evaluates to false. Finding an assignment that maximises the number of strongly satisfied clauses is NP-hard. We present a simple algorithm that finds, given a multiset of clauses that admits an assignment that strongly satisfies ρ of the clauses, an assignment in which at least ρ of the clauses are *weakly satisfied*, in the sense that an *even* number of literals evaluate to false.

In particular, this implies an efficient algorithm for finding an undirected cut of value ρ in a graph G given that a directed cut of value ρ in G is promised to exist. A similar argument also gives an efficient algorithm for finding an acyclic subgraph of G with ρ edges under the same promise.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases approximation, promise constraint satisfaction, max and, max even, max cut, max dicut, max acyclic

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2025.3

Category APPROX

Related Version *Full version*: https://arxiv.org/abs/2409.07837

Funding *Tamio-Vesa Nakajima*: UKRI EP/X024431/1, Clarendon Fund Scholarship
Stanislav Živný: UKRI EP/X024431/1

Acknowledgements We thank anonymous reviewers of an earlier version of this paper for useful comments and a simplification of our algorithm. We also thank the anonymous reviewers of APPROX 2025 for feedback and suggestions for changes.

1 Introduction

The Maximum Cut problem in *undirected* graphs (Max-Cut) is a fundamental problem, seeking a partition of the vertex set into two parts while maximising the number of edges going across. While Max-Cut is NP-hard [10], a random assignment leads to a $\frac{1}{2}$-approximation algorithm. In their influential work, Goemans and Williamson gave the first improvement and presented an SDP-based α_{GW}-approximation algorithm [7], where $\alpha_{\text{GW}} \approx 0.878$. Under Khot's Unique Games Conjecture (UGC) [12], this approximation factor is optimal [13, 16]. The current best known inapproximability result not relying on UGC is $\frac{16}{17} \approx 0.941$ [19] (obtained by a gadget from Håstad's optimal inapproximability result [8]).

The Maximum Cut problem in *directed* graphs (Max-DiCut) is a closely related and well-studied NP-hard problem, seeking a partition of the vertex set into two parts while maximising the number of edges going across in the prescribed direction. A random assignment leads to a $\frac{1}{4}$-approximation algorithm. In the first improvement over the random assignment, Goemans and Williamson presented an SDP-based 0.796-approximation algorithm [7]. By considering a stronger SDP formulation (with triangle inequalities), Feige and Goemans later presented a 0.859-approximation algorithm for Max-DiCut [5], building on the work of Feige and Lovász [6]. Follow-up works by Matuura and Matsui [15] and by Lewin, Livnat, and Zwick [14] further improved the approximation factor, obtaining a 0.874-approximation

algorithm [14]. On the hardness side, the best inapproximability factor under NP-hardness is $\frac{12}{13} \approx 0.923$ [19] (again, via a gadget from a result in [8]). In recent work, Brakensiek, Huang, Potechin, and Zwick gave an 0.8746-approximation algorithm for Max-DiCut, also showing that it is UGC-hard to achieve a 0.875-approximation [4].

Our results

Consider the following *promise* variant of the two discussed problems:

▶ **Definition 1** (Max-DiCut-Cut). *Given a directed graph that has a directed cut of value ρ, efficiently find an undirected cut (i.e., ignoring edge directions) of value at least ρ.*

It turns out that the Max-DiCut-Cut admits an efficient algorithm, as will follow from our more general result (cf. Theorem 3 below).

We represent the Boolean *true* value by $+1$ and the *false* value by -1. A *literal* sx is a variable x ($s = 1$, positive sign) or its negation $-x$ ($s = -1$, negative sign). A *clause* $C = \{s_1x_1, \ldots, s_kx_k\}$ is a multiset of literals. An assignment of $+1$s and -1s to the variables of a clause C *strongly satisfies* C if no literal evaluates to false, and *weakly satisfies* C if an even number of literals evaluates to false.

We now define a natural variant of the Boolean satisfiability problem.

▶ **Definition 2** (Max-And-Even). *Given a multiset of clauses for which there is an assignment strongly satisfying ρ of the clauses, find an assignment that weakly satisfies ρ of the clauses.*

Coming back to Max-DiCut-Cut, a directed edge (u, v) is cut if the clause $\{-u, v\}$ is strongly satisfied, and the edge is cut (ignoring the direction) if the clause $\{-u, v\}$ is weakly satisfied. Thus, Max-And-Even is a generalisation of Max-DiCut-Cut.

Our main result is an algorithm for Max-And-Even, and thus also for Max-DiCut-Cut.

▶ **Theorem 3.** *There exists a polynomial-time algorithm for* Max-And-Even.

Theorem 3 has an interesting corollary for a related, and in some sense dual, problem.

▶ **Definition 4** (Cut-Orientation). *Given an undirected graph G that has a maximum cut of value ρ, orient the edges of G so that the resulting directed graph has a directed cut of maximum possible value.*

Cut-Orientation can be approximated with the ratio α_{GW}: Find a cut in G of size $\alpha_{\mathrm{GW}}\rho$ using the Goemans-Williamson algorithm [7], then orient all the edges from one side of the cut to the other. Interestingly, Theorem 3 shows that this approximation is UGC-optimal.

▶ **Corollary 5.** *It is* UGC*-hard to approximate* Cut-Orientation *with a factor better than α_{GW}.*

Proof. The observation is that the Max-DiCut-Cut problem gives a reduction from Max-Cut to Cut-Orientation: Given an instance of Max-Cut that has a cut of size ρ, if we could orient the graph to obtain a directed cut of size $\alpha\rho$, then by solving the resulting digraph as an instance of Max-DiCut-Cut we would find a cut of size $\alpha\rho$. Since it is UGC-hard to approximate Max-Cut with a factor better than α_{GW} [13, 16], the same is true for Cut-Orientation. ◀

Using ideas from the proof of Theorem 3, we give an efficient algorithm for another problem.

▶ **Definition 6** (Max-DiCut-Acyclic). *Given a directed graph G that has a directed cut of value ρ, efficiently find an acyclic subgraph of G with at least ρ edges.*

▶ **Theorem 7.** *There exists a polynomial-time algorithm for* Max-DiCut-Acyclic.

Related work

The motivation for our work comes from a systematic study of so-called promise constraint satisfaction problems (PCSPs) [3, 1]. These are problems concerned with homomorphisms between graphs and, more generally, relational structures. A homomorphism h from a graph G to a graph H, also known as an H-colouring of G [9], is a map from the vertex set of G to the vertex set of H that preserves edges; i.e., if (u, v) is an edge in G then $(h(u), h(v))$ must be an edge in H. For instance, if $H = K_3$ is a clique on 3 vertices, then homomorphisms from G to H are precisely 3-colourings of G. Equivalently, there is a homomorphism from G to H if G is a subgraph of a blow-up of H, where a blow-up of H replaces every vertex by an independent set and every edge by a complete bipartite graph. Homomorphisms between relational structures are defined analogously as maps from the universe of one structure to the universe of another structure so that all relations are preserved by a component-wise application of the map.

Our work is related to an optimisation variant of PCSPs. In particular, the problem Max-PCSP(G, H) asks: Given an input structure X which admits a G-colouring of value ρ, find an H-colouring of value at least ρ. For example, Max-PCSP(G, H) with bipartite G was classified [18] and the intractability of non-bipartite G was very recently established in [17]. This leaves open cases where G and H are not graphs but rather arbitrary relational structures. The simplest open case is Boolean binary structures, i.e., digraphs on 2 vertices. There are three interesting problems: Max-Cut, Max-DiCut, and Max-DiCut-Cut. Since Max-Cut and Max-DiCut are already well-understood, understanding the complexity of Max-DiCut-Cut was the first step in the ultimate goal of understanding all structures beyond (undirected) graphs. Thus, we view the importance of Theorem 3 as twofold. Firstly, as a non-trivial tractability result for a natural problem. Secondly, as initiating the study of Max-PCSP beyond graphs. Finally, Theorem 7 gives a tractability result for Max-PCSP(A, B), where $A = (\{0, 1\}, \{(0, 1)\})$ represents the cut problem in directed graphs and $B = (\mathbb{Q}, <)$ represents the graph acyclicity problem, thus identifying a natural tractable Max-PCSP with an infinite structure, following a well-established line of work on infinite-domain CSPs [2].

2 Algorithm from Theorem 3

We denote by $[k]$ the set $\{1, \dots, k\}$. For a statement φ, we let $[\varphi]$ be 1 if φ is true and 0 otherwise. For a clause $C = \{s_1 x_1, \dots, s_k x_k\}$, weak and strong satisfaction of C by an assignment c can be expressed as

$$[C \text{ is strongly satisfied}] = \frac{1}{2} + \frac{1}{2} \min_{1 \le i \le k} s_i c(x_i),$$
$$[C \text{ is weakly satisfied}] = \frac{1}{2} + \frac{1}{2} \prod_{1 \le i \le k} s_i c(x_i).$$

Thus, a clause C is strongly satisfied if and only if the minimum of all the literals in that clause is 1, and this minimum is -1 otherwise. Hence, we can cast Max-And over an instance with variable set V, and clauses $C_1, \dots, C_m$ with $C_i = \{s_1^i x_1^i, \dots, s_{k_i}^i x_{k_i}^i\}$, as the problem of finding a solution to

$$\max_{c: V \to D} \sum_{i=1}^{m} \frac{1}{2} + \frac{1}{2} \min_{1 \le j \le k_i} s_j^i c(x_j^i), \tag{1}$$

where we take $D = \{-1, +1\}$. One way to establish Theorem 3 would be to first solve this problem relaxed to $D = [-1, 1]$ using LP, and then round directly. We take a slightly different

(and simpler) route, which relies on the idea of "half-integrality" and will be also useful for proving Theorem 7: (1) can be solved exactly if we take $D = \{-1, 0, +1\}$.

▶ **Theorem 8.** *An optimal solution to* (1) *can be found in polynomial time for* $D = \{-1, 0, +1\}$.

Proof. First, find any optimal solution c^* to (1) with $D = [-1, 1]$ by LP. To do this, one can use a standard trick. For each clause $C_i = \{s_1^i x_1^i, \ldots, s_{k_i}^i x_{k_i}^i\}$ for $i \in [m]$, introduce a variable t_i and constraints $t_i \leq s_j^i c(x_j^i)$ for $j \in [k_i]$. Next, replace the objective function by $\sum_{i=1}^m \frac{1}{2} + \frac{1}{2} t_i$. Observe that in any feasible solution to this LP, we have $t_i \leq \min_{1 \leq j \leq k_i} s_j^i c(x_j^i)$. Furthermore, since the objective is an increasing function of $t_1, \ldots, t_m$, in any optimal solution to this LP we have $t_i = \min_{1 \leq j \leq k_i} s_j^i c(x_j^i)$. Hence the optimal solutions to this LP precisely correspond to the optimal solutions to (1).

We will shift the solution while keeping it optimal until the image of c^* is contained in $\{-1, 0, 1\}$. As a pre-processing step, flip signs of literals so that $c^*(x) \geq 0$ for $x \in V$. Our goal now is to have the image of c^* contained in $\{0, 1\}$. Suppose without loss of generality that $V = \{1, \ldots, n\}$ and $c^*(1) \leq \cdots \leq c^*(n)$. Define

$$F(c) = \sum_{i=1}^{m} \frac{1}{2} + \frac{1}{2} \min_{1 \leq j \leq k_i} s_j^i c(x_j^i).$$

Note that for every c for which $0 \leq c(1) \leq \cdots \leq c(n)$, $\arg\min_{1 \leq j \leq k_i} s_j^i c(x_j^i)$ is known. Indeed, if all s_j^i are positive then the minimum is attained at the smallest x_j^i; whereas if there is at least one negative s_j^i then the minimum is attained at the largest x_j^i among those with $s_j^i = -1$. (Here we compare variables by the natural ordering, since we have assumed the variables are natural numbers.) Let $j(i) = \arg\min_{1 \leq j \leq k_i} s_j^i c(x_j^i)$ be this minimum; then, for all c where $0 \leq c(1) \leq \cdots \leq c(n)$,

$$F(c) = \sum_{i=1}^{m} \frac{1}{2} + \frac{1}{2} s_{j(i)}^i c(x_{j(i)}^i),$$

In other words, F is an affine function in terms of c! Define $c^*(0) = 0, c^*(n+1) = 1$. While c^* has an image that is not 0 or 1, do the following. Take $1 \leq a \leq b \leq n$ so that $0 = c^*(0) = \cdots = c^*(a-1) < c^*(a) = \cdots = c^*(b) < c^*(b+1) \leq \cdots \leq c^*(n+1) = 1$. Such a, b exist, or else $c^*(x) \in \{0, 1\}$ for all $x \in [n]$. If the sum of the coefficients of $c^*(a), \ldots, c^*(b)$ in F (seen as an affine function) is negative, then we could decrease all of these values together (since this maintains the fact that $0 \leq c^*(1) \leq \cdots \leq c^*(n)$) and improve our solution, which is impossible. If the sum of the coefficients is positive, then we could improve our solution by increasing the values of $c^*(a), \ldots, c^*(b)$, which is impossible. So we can assume the sum of the coefficients is 0, in which case we can set the values $c^*(a), \ldots, c^*(b)$ to anything we want in the interval $[0, c^*(b+1)]$ without changing the value of the solution. So set $c^*(a) = \cdots = c^*(b) = 0$. Continue this procedure until there are no variables set to anything other than 0 or 1. ◀

We now prove Theorem 3, restated below for the reader's convenience.

▶ **Theorem 3.** *There exists a polynomial-time algorithm for* Max-And-Even.

Proof. Without loss of generality, we can assume that each variable appears in each clause at most once. Indeed, if a clause C contains a variable x and its negation $-x$ then C cannot be strongly satisfied but could be weakly satisfied, so running our algorithm on the

instance without C causes no issues. Similarly, if a literal appears twice in a clause then both occurrences can be removed, as this does not affect weak satisfiability (but could improve strong satisfiability).

Suppose we are given an instance of Max-And-Even, namely an expression of form

$$\max_{c:V\to\{\pm 1\}} \sum_{i=1}^{m} \frac{1}{2} + \frac{1}{2} \min_{1\le j\le k_i} s_j^i c(x_j^i).$$

Suppose that the value of this expression is ρ. Our goal is to find a function c^* such that that number of weakly satisfied clauses is at least ρ. Recalling that a clause $\{s_1x_1, \dots, s_kx_k\}$ is weakly satisfied by c if and only if $\frac{1}{2} + \frac{1}{2}\prod_i s_i c(x_i) = 1$, and that this expression is 0 otherwise, we note that we must find $c^* : V \to \{-1, +1\}$ such that

$$\rho \le \sum_{i=1}^{m} \frac{1}{2} + \frac{1}{2} \prod_{1\le j\le k_i} s_j^i c^*(x_j^i). \tag{2}$$

Using Theorem 8, we can efficiently find a function $\tilde{c} : V \to \{-1, 0, +1\}$ such that

$$\rho \le \sum_{i=1}^{m} \frac{1}{2} + \frac{1}{2} \min_{1\le j\le k_i} s_j^i \tilde{c}(x_j^i).$$

We claim that the following choice of c^* makes (2) hold in expectation: if $\tilde{c}(v) = \pm 1$ then set $c^*(v) = \tilde{c}(v)$. Otherwise set $c^*(v)$ equal to $+1$ or -1 uniformly and independently at random. Indeed, by linearity of expectation it suffices to show that, for any clause $\{s_1x_1, \dots, s_kx_k\}$, we have

$$\frac{1}{2} + \frac{1}{2} \min_{1\le i\le k} s_i \tilde{c}(x_i) \le \mathbb{E}\left[\frac{1}{2} + \frac{1}{2} \prod_{1\le i\le k} s_i c^*(x_i).\right]$$

In particular, let us consider the value of the minimum on the left. If it is -1 there is nothing left to prove. If it is $+1$, then $\tilde{c}(x_i) \neq 0$ and hence $c^*(x_i) = \tilde{c}(x_i)$ for all x_i, and the bound holds with equality. Finally, if the minimum is 0, then there exists some x_i such that $\tilde{c}(x_i) = 0$; hence $c^*(x_i)$ is set to $+1$ or -1 equiprobably. It is easy then to see that the product on the right hand side is either $+1$ or -1 equiprobably, whence the conclusion. (This depends, essentially, on the fact that every variable appears in each clause at most once.)

We now derandomise the rounding scheme above; in other words, we will show how to deterministically set the variables so that the resulting value is at least as good as the expectation of the random variables. Equivalently, given a set of parity constraints on subsets of variables, the goal is to find an assignment that satisfies at least as many constraints as the random assignment. This problem can be derandomised easily using the method of conditional expectation — indeed these techniques have appeared before in e.g. [11], but we include them for completeness. Recall that, by assumption, each variable appears in each clause at most once. Thus, conditional on some subset of variables being fixed, we expect half of the remaining parity constraints that have at least one unfixed variable within them to be satisfied. Thus, we can derandomise by going through the variables one by one; when we set a variable x, we set it so that as many as possible of the parity constraints where x is the last unfixed variable remaining are satisfied. ◀

3 Algorithm from Theorem 7

We are now ready to prove Theorem 7, restated below.

▶ **Theorem 7.** *There exists a polynomial-time algorithm for* Max-DiCut-Acyclic.

Proof. Equivalently, given a digraph G that has a dicut with ρ edges, we will produce an ordering of the vertices of G such that at least ρ edges are oriented according to the ordering — removing all the edges that are not oriented in this way gives us the required subgraph. For some ordering σ, we say that an edge (x, y) is well ordered by σ if x comes before y in σ.

As a pre-processing step, remove all loops from the input digraph. This will never lower the size of the maximum dicut, nor will it lower the size of the outputted subgraph. Now compute a solution $\tilde{c}$ to (1) for $G = (V, E)$ over $D = \{-1, 0, 1\}$, using Theorem 8. By this, we mean a solution to the instance including the clause $\{-x, y\}$ once for each occurrence of the edge (x, y) in G. In other words, each edge (x, y) contributes $\frac{1}{2} + \frac{1}{2}\min(-c(x), c(y))$ to (1). The values of this expression over $D = \{-1, 0, 1\}$ are tabulated in Figure 1.

$c(x)$ \ $c(y)$	-1	0	1
-1	0	$\frac{1}{2}$	1
0	0	$\frac{1}{2}$	$\frac{1}{2}$
1	0	0	0

Figure 1 Value of $\frac{1}{2} + \frac{1}{2}\min(-c(x), c(y))$

Since the digraph admits a dicut with ρ edges, the value of $\tilde{c}$ is at least ρ. Partition the vertices of G into V_{-1}, V_0, V_1 according to their image through $\tilde{c}$. Let $E_{ij} = E \cap (V_i \times V_j)$, for $i, j \in \{-1, 0, 1\}$. Let π_{-1}, π_0, π_1 be arbitrary orderings of V_{-1}, V_0, V_1, and let π_0' be the reverse of π_0. We claim that one of the orderings $\sigma = (\pi_{-1}, \pi_0, \pi_1)$ or $\sigma' = (\pi_{-1}, \pi_0', \pi_1)$ can be returned.

To see why, note that every edge (x, y) with $\tilde{c}(x) < \tilde{c}(y)$ is well ordered by both σ and σ'. Furthermore, at least half the edges (x, y) with $\tilde{c}(x) = \tilde{c}(y) = 0$ are ordered correctly in one of σ or σ' (this is why removing loops is essential). But (cf. Figure 1 and the optimality of $\tilde{c}$),

$$\begin{aligned}\rho \leq \#E_{-1,1} + \frac{1}{2}\#E_{-1,0} + \frac{1}{2}\#E_{0,1} + \frac{1}{2}\#E_{0,0} &\leq \#E_{-1,1} + \#E_{-1,0} + \#E_{0,1} + \frac{1}{2}\#E_{0,0} \\ &= \#\{(x, y) \in E \mid \tilde{c}(x) < \tilde{c}(y)\} + \frac{1}{2}\#E_{0,0}.\end{aligned}$$

Hence, at least one of σ and σ' well orders at least ρ edges. ◀

4 Conclusions

As discussed briefly in Section 1, the main contribution of this paper is twofold. Firstly, we give a simple, efficient algorithm for a very natural computational problem. Secondly, we initiate the study of Max-PCSPs beyond graphs (which have been recently classified [17]) and beyond finite structures. A concrete question worthy of investigating is for which Max-PCSPs our method of rounding, relying on the idea of half-integrality, is applicable.

References

1 Libor Barto, Jakub Bulín, Andrei A. Krokhin, and Jakub Opršal. Algebraic approach to promise constraint satisfaction. *J. ACM*, 68(4):28:1–28:66, 2021. doi:10.1145/3457606.

2 Manuel Bodirsky. *Complexity of infinite-domain constraint satisfaction*, volume 52. Cambridge University Press, 2021.

3 Joshua Brakensiek and Venkatesan Guruswami. Promise Constraint Satisfaction: Algebraic Structure and a Symmetric Boolean Dichotomy. *SIAM J. Comput.*, 50(6):1663–1700, 2021. doi:10.1137/19M128212X.

4 Joshua Brakensiek, Neng Huang, Aaron Potechin, and Uri Zwick. Separating MAX 2-AND, MAX DI-CUT and MAX CUT. In *Proc. 64th IEEE Annual Symposium on Foundations of Computer Science (FOCS'23)*, pages 234–252. IEEE, 2023. doi:10.1109/FOCS57990.2023.00023.

5 Uriel Feige and Michel X. Goemans. Aproximating the Value of Two Prover Proof Systems, With Applications to MAX 2SAT and MAX DICUT. In *Proc. 3rd Israel Symposium on Theory of Computing and Systems (ISTCS'95)*, pages 182–189. IEEE Computer Society, 1995. doi:10.1109/ISTCS.1995.377033.

6 Uriel Feige and László Lovász. Two-prover one-round proof systems: Their power and their problems (extended abstract). In *Proc. 24th Annual ACM Symposium on Theory of Computing (STOC'92)*, pages 733–744. ACM, 1992. doi:10.1145/129712.129783.

7 Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995. doi:10.1145/227683.227684.

8 Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, jul 2001. doi:10.1145/502090.502098.

9 Pavol Hell and Jaroslav Nešetřil. On the Complexity of H-coloring. *J. Comb. Theory, Ser. B*, 48(1):92–110, 1990. doi:10.1016/0095-8956(90)90132-J.

10 Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations*, pages 85–103. Springer US, 1972. doi:10.1007/978-1-4684-2001-2_9.

11 Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The approximability of constraint satisfaction problems. *SIAM J. Comput.*, 30(6):1863–1920, 2000. doi:10.1137/S0097539799349948.

12 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proc. 34th Annual ACM Symposium on Theory of Computing (STOC'02)*, pages 767–775. ACM, 2002. doi:10.1145/509907.510017.

13 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal Inapproximability Results for MAX-CUT and Other 2-Variable CSPs? *SIAM J. Comput.*, 37(1):319–357, 2007. doi:10.1137/S0097539705447372.

14 Michael Lewin, Dror Livnat, and Uri Zwick. Improved Rounding Techniques for the MAX 2-SAT and MAX DI-CUT Problems. In *Proc. 9th International Conference on Integer Programming and Combinatorial Optimization (IPCO'02)*, volume 2337 of *Lecture Notes in Computer Science*, pages 67–82. Springer, 2002. doi:10.1007/3-540-47867-1_6.

15 Shiro Matuura and Tomomi Matsui. New approximation algorithms for MAX 2SAT and MAX DICUT. *Journal of the Operations Research Society of Japan*, 46(2):178–188, 2003. doi:10.15807/jorsj.46.178.

16 Elchanan Mossel, Ryan O'Donnell, and Krzysztof Oleszkiewicz. Noise stability of functions with low influences: invariance and optimality. *Ann. of Math. (2)*, 171(1):295–341, 2010. doi:10.4007/annals.2010.171.295.

17 Tamio-Vesa Nakajima and Stanislav Živný. A Dichotomy for Maximum PCSPs on Graphs, 2025. arXiv:2406.20069v3.

18 Tamio-Vesa Nakajima and Stanislav Živný. Maximum bipartite vs. triangle-free subgraph. In *Proc. 52nd International Colloquium on Automata, Languages, and Programming (ICALP'25)*, 2025. doi:10.4230/LIPIcs.ICALP.2025.121.

19 Luca Trevisan, Gregory B. Sorkin, Madhu Sudan, and David P. Williamson. Gadgets, approximation, and linear programming. *SIAM J. Comput.*, 29(6):2074–2097, 2000. doi:10.1137/S0097539797328847.