Complexity of Local Search for CSPs Parameterized by Constraint Difference

Aditya Anand* Vincent Cohen-Addad† Tommaso D'Orsi‡ Anupam Gupta§
Euiwoong Lee¶ Debmalya Panigrahi‖ Sijin Peng**

Abstract

In this paper, we study the parameterized complexity of local search, whose goal is to find a good nearby solution from the given current solution. Formally, given an optimization problem where the goal is to find the largest *feasible* subset S of a universe U, the new input consists of a *current solution* P (not necessarily feasible) as well as an ordinary input for the problem. Given the existence of a feasible solution S^*, the goal is to find a feasible solution as good as S^* in parameterized time $f(k) \cdot n^{O(1)}$, where k denotes the *distance* $|P \Delta S^*|$. This model generalizes numerous classical parameterized optimization problems whose parameter k is the minimum number of elements removed from U to make it feasible, which corresponds to the case $P = U$.

We apply this model to widely studied Constraint Satisfaction Problems (CSPs), where U is the set of constraints, and a subset U' of constraints is feasible if there is an assignment to the variables satisfying all constraints in U'. We give a complete characterization of the parameterized complexity of all boolean-alphabet *symmetric* CSPs, where the predicate's acceptance depends on the number of true literals.

*University of Michigan, Ann Arbor, USA. adanand@umich.edu.

†Google Research, NYC, USA. vcohenad@gmail.com.

‡Bocconi University, Italy. tomasso.dorsi@unibocconi.it.

§New York University, USA. anupam.g@nyu.edu. Supported in part by NSF awards CCF-2224718 and CCF-2422926.

¶University of Michigan, Ann Arbor, USA. euiwoong@umich.edu. Supported in part by NSF award CCF-2236669 and by Google, Inc.

‖Duke University, USA. debmalya@cs.duke.edu. Supported in part by NSF awards CCF-1955703 and CCF-2329230.

**Massachusetts Institute of Technology, USA. peng-sj21@mails.tsinghua.edu.cn.

Contents

1 Introduction

The classical theory of NP-hardness sets limits on the polynomial-time tractability of a vast array of algorithmic problems. This raises the question: *what additional information about the problem instance can one use to help overcome complexity barriers?* A very successful line of research that aims to address this broad question is that of *fixed parameter tractable* (FPT) algorithms. The idea is to identify a parameter that has a small value in *typical* instances, and design an algorithm whose running time is exponential in the value of the parameter but polynomial in the overall size of the instance.

This philosophy has been widely applied to the following general class of optimization problems: let U be a universe and let $\mathcal{S} \subseteq 2^U$ be an (often implicitly given) collection of *feasible* subsets of U, and the goal is to find $S \in \mathcal{S}$ that maximizes $|S|$, or equivalently to find the smallest T that puts $U \setminus T \in \mathcal{S}$. For instance, given a graph $G = (V, E)$, if $U = E$ and $F \in \mathcal{S}$ if (V, F) is 2-colorable, the problem corresponds to the famous MaxCut/MinUncut pair. Or given $G = (V, E)$, if $U = V$ and $\mathcal{S}$ denotes the collection of all independent sets, then we have Independent Set/Vertex Cover. In this class, there are many situations where typical (or at least interesting) instances have the optimal value for the minimization version much smaller than $|U|$. (I.e., the universe U is already close to feasible.) Consequently, numerous results in the FPT literature study a problem from the above class - we let the parameter k be the optimal value for the minimization version, and give an algorithm of running time $f(k)n^{O(1)}$ (better than the naive running time of $n^{O(k)}$).

Can we extend the applicability of this framework? We can interpret the algorithms in the above paragraph as one-step *local search*, where we are given a current solution (not necessarily feasible) *close* to some feasible solution S^* with the optimal value, and the algorithm finds a feasible solution as good as S^*. This motivates us to ask: *if the algorithm is provided* **any** *current solution $P \subseteq U$ (as a solution for the max version) and there is a feasible solution $S^* \subseteq U$ close to P, can the algorithm efficiently recover a solution as good as S^*?* Formally, we define the parameter k as the distance $|P \Delta S^*|$, and our goal is to design an algorithm whose runtime is arbitrary in k but polynomial in the size of the problem instance.[1]

We apply this model to Constraint Satisfaction Problems (CSPs), one of the most important classes of optimization problems in the theory of computation. Given a predicate $R \subseteq \{0,1\}^r$, MaxCSP(R) is the computational problem whose input consists of the set of variables V and the set of constraints $\mathcal{C}$. Given an assignment $\alpha : V \to \{0,1\}$, each constraint $C \in \mathcal{C}$ is satisfied when the values of the r specified literals (variables or their negations) belong to R. (See Section 2 for formal definitions.) The goal is to find an assignment that satisfies as many constraints as possible. Generalizing the MaxCut example above, we apply MaxCSP(R) to our model where $U = \mathcal{C}$ and a subset of constraints $\mathcal{C}' \subseteq \mathcal{C}$ is feasible if there exists an assignment α that satisfies all of $\mathcal{C}'$ (i.e., $\mathcal{C}'$ is *satisfiable*); so the current solution $\mathcal{P}$ is a set of constraints and we assume that it is close to some set of satisfiable constraints. It yields the following computational task.

Problem: ImproveMaxCSP(R)
Input: A CSP(R) instance $I(V, \mathcal{C})$, an integer $k \in \mathbb{N}$, and $\mathcal{P} \subseteq \mathcal{C}$.
Parameter: k
Promise: There exists an assignment α_0 with the property $|\mathcal{C}_I(\alpha_0) \Delta \mathcal{P}| \leq k$, where $\mathcal{C}_I(\alpha_0)$ is the set of clauses α_0 satisfies, and Δ represents symmetric difference.

[1] Actually, since $|S \Delta S^*| = |(U \setminus S) \Delta (U \setminus S^*)|$, this parameter remains the same for both maximization and minimization versions.

Output: A *good* assignment α, where the word *good* simply means $|\mathcal{C}_I(\alpha)| \geq |\mathcal{C}_I(\alpha_0)|$. (We do not require α to satisfy $|\mathcal{C}_I(\alpha)\Delta\mathcal{P}| \leq k$.)

Note that, given an instance $(I(V,\mathcal{C}), k, \mathcal{P})$, the problem is essentially equivalent to find α such that $|\mathcal{C}_I(\alpha)|$ is at least $\max_{\beta:|\mathcal{C}_I(\beta)\Delta\mathcal{P}|\leq k} |\mathcal{C}_I(\beta)|$; in words, find a feasible solution as good as any feasible k-neighbor of $\mathcal{P}$. (Formally, we still will keep the fixed promised assignment α_0 as part of the instance and define a good assignment based on α_0.)

We also observe that if the decision version of CSP(R) (i.e., finding an assignment that satisfies every constraint) can be solved in polynomial time, an easy $|\mathcal{C}|^k \cdot \mathrm{poly}(|V|, |\mathcal{C}|)$-time algorithm for ImproveMaxCSP(R) follows by exhaustively guessing $(\mathcal{C}_I(\alpha_0)\Delta\mathcal{P})$ and solving the decision version with constraints $\mathcal{C}_I(\alpha_0)$; otherwise it is NP-hard even when $k = 0$ (just letting $\mathcal{P} = \mathcal{C}$), which concludes that ImproveMaxCSP(R) is in **XP** if and only if the decision version is in **P**. Also, if MaxCSP(R) is NP-hard, there is no $\mathrm{poly}(|V|, |\mathcal{C}|, k)$-time algorithm for ImproveMaxCSP(R).

In this paper, we provide a complete characterization of the parameterized complexity of ImproveMaxCSP(R), where R is restricted to a *symmetric* predicate, whose acceptance only depends on the number of true literals. This still includes many well-known CSPs including rLIN (which generalizes MaxCut), rSAT, rAND, rNAE, and rAE, where AE and NAE abbreviate All-Equal and Not-All-Equal respectively. Namely, the main contribution of this paper is the following theorem:

Theorem 1.1 (Main Theorem (Informal))**.** *For any symmetric predicate R,* ImproveMaxCSP*(R) is either FPT or W[1]-hard.*

It turns out that among nontrivial predicates, rAND for any $r \geq 1$ and 2AE (generalizing MaxCut) are the only ones in FPT. We design parameterized algorithms for them extending the previous algorithms for MinCSP. While many predicates inherit their hardness result from the hardness of MinCSP, we also prove hardness for the problems whose MinCSP is tractable, namely rAE for $r \geq 3$ and $\leq$ 1-out-of-rSAT for $r \geq 2$. See Section 2.3 for our formal characterization.

In the context of local search, a natural special case of ImproveMaxCSP(R) is when the input $\mathcal{P}$ is restricted to be $\mathcal{C}_I(\beta)$ for some assignment $\beta : V \to \{0,1\}$. Theorem 1.1 addresses this special case as well, in the sense that our W[1]-hard reductions indeed have $\mathcal{P} = \mathcal{C}_I(\beta)$ for some assignment β.

1.1 Related Work and Discussion

The parameterized complexity of local search, whose goal is to improve the current solution to a strictly better solution nearby, was indeed studied in the parameterized complexity literature. Fellows, Fomin, Lokshtanov, Rosamond, Saurabh, and Villanger [FFL+12] study various problems like r-Center, Vertex Cover, Odd Cycle Transversal, Max-Cut, and Max-Bisection on special classes of sparse graphs like planar, minor-free, and bounded-degree graphs, and Szeider [Sze11] considers a similar framework for SAT and MaxSAT.

This (strict) local search model can be relaxed to another model called *permissive* local search [MS11]. Here we are given a solution S, and the goal is to find a solution S' which has lower cost (which may differ from S in more than k vertices), or correctly conclude that every solution which differs from S in at most k vertices has higher cost. This model has been studied for a few problems - including a variant of stable marriage [MS11], Vertex Cover [GKO+12] and also in the context of Min Ones CSPs [KM12], extending Marx's complete classification of the parameterized complexity of CSPs based on if there is a solution with exactly k ones [Mar05].

One crucial difference between our model and all previous models on CSPs is how the distance between the two solutions is defined. The previous papers define it as the number of variables with different values in two solutions, but our definition concerns the set of constraints whose satisfaction changes between two solutions. In fact, recall that our definition of a solution is an arbitrary set $\mathcal{P}$ of constraints not necessarily feasible (i.e., corresponding to a variable assignment), which allows $\mathcal{P} = \mathcal{C}$ and captures algorithms parameterized by the number of unsatisfied constraints. In that sense, our model is slightly broader than traditional local search where one maintains a feasible solution and tries to strictly improve in every step.

Fixed parameter tractability of CSPs parameterized by the number of unsatisfied constraints has been an important and active research area [BEM16, KKPW21, KKPW22, KKPW23], where a complete classification for Boolean relations was obtained very recently. Their definition of CSPs is strictly more general than ours, where each constraint language contains multiple predicates, even with different arities. It is an exciting research direction to extend our framework for every CSP.

Our negative results can be interpreted as demonstrating the importance of *near-perfect instances* for classical FPT algorithms. For instance, we show ImproveMaxCSP(2SAT) is $W[1]$-hard while MinCSP(2SAT) is known to admit an FPT algorithm [RO08, RRS11, CPPW13, NRRS12], so the fact that the optimal solution indeed satisfies all but k constraints affects the parameterized complexity.

There are several natural research directions based on this paper. We believe that the following directions will be interesting to study further:

1. The most immediate one is to extend our characterization for all CSPs. One notable set of boolean relations whose complexity is still open in our model is $\{x = y, x \wedge \neg y\}$.

2. A natural variant of our model is when the solution is represented as an assignment to the variables, which is guaranteed to have a Hamming distance at most k to a strictly better solution. For Min Ones CSP, where the goal is to find a satisfying assignment with the minimum number of true variables, this variant has been studied previously [KM12].

3. Other than CSPs, one framework that captures numerous results in the FPT literature is covering (resp. packing) problems, especially on graphs, where we want to select the smallest (resp. largest) subset $S \subseteq U$ subject to some covering (resp. packing) constraints. Local search naturally extends to this setting (i.e., we are given $S' \subseteq U$ with $|S' \Delta S| \leq k$), and it will be interesting to see if classical FPT results on graph covering/packing problems extend to this model, including Multicut [BDT11, MR11], Min Bisection [CLP+14], k-Cut [KT11], Disjoint Paths [RS95].

4. In the approximation algorithms literature, there are numerous studies about CSPs on *structured instances*, most notably dense and expanding instances (see [ALS25] and references therein). It would be interesting to see whether these structures, if suitably parameterized, help the problems become more tractable in our model.

2 Our Results

2.1 Constraint Satisfaction Problems

In this section we recall basic definitions of constraint satisfaction problems.

For integer $r \in \mathbb{N}^+$, a *boolean relation* R is a subset of $\mathbb{F}_2^r$. The integer r is called the *arity* of R. For a boolean relation R of arity r and $b \in \mathbb{F}_2^r$, define the boolean relation $R \oplus b := \{\alpha \oplus b \mid \alpha \in R\}$,

where $\oplus$ is the addition operator over $\mathbb{F}_2^r$. When considering R as a clause in a CSP instance, $\oplus$ operator can be seen as variable negations. For a relation R with index $[r]$ and $S \subseteq [r]$, define the projection relation $\pi_S(R) = \{\alpha_S \mid \alpha \in R\}$, where α_S means extracting the value on coordinates in S.

A boolean relation R of arity r is *symmetric* if for all $(x_1, \cdots, x_r) \in \mathbb{F}_2^r$ and any permutation $p : [r] \to [r]$, $(x_1, \cdots, x_r) \in R \iff (x_{p(1)}, \cdots, x_{p(r)}) \in R$. Equivalently, R is symmetric if there exists $S \subseteq \{0, 1, \cdots, r\}$ such that $(x_1, \cdots, x_r) \in R \iff \sum_{i=1}^{r} x_i \in S$ (where the addition is performed in $\mathbb{Z}$). We define $\text{SymRel}(r, S)$ to be such R.

A *boolean constraint language* is a set of boolean relations. Two constraint languages are *equivalent* if they contain exactly the same set of relations. A *clause* C over a boolean constraint language Γ is a pair (V_C, R_C), where $R_C \in \Gamma$ and $V_C = (v_{C,1}, \ldots, v_{C,r})$ is an r-tuple of variables where r is the arity of R_C. We refer to V_C as the *scope* of the clause C. An assignment $\alpha : V_C \to \{0, 1\}$ *satisfies* clause (V_C, R_C) if $(\alpha(v_{C,1}), \ldots, \alpha(v_{C,r})) \in R_C$, and α *violates* the clause otherwise.

In this paper, we only consider a restricted set of boolean constraint languages, where the language Γ includes all possible negation patterns $R \oplus b$ for some symmetric predicate R of arity r and $b \in \mathbb{F}_2^r$. We use the notation $\text{SymLang-N}(R) = \{R \oplus b \mid b \in \mathbb{F}_2^r\}$ for such Γ, where SymLang-N stands for "symmetric language with negation". We further define $\text{SymLang-N}(r, S)$ to be $\text{SymLang-N}(\text{SymRel}(r, S))$.

For a boolean constraint language Γ, a $\text{CSP}(\Gamma)$ instance I is a pair $(V_I, \mathcal{C}_I)$, where V_I is the set of variables, and $\mathcal{C}_I$ is the set of clauses over the boolean constraint language Γ, all of which have scope inside V_I. We say a variable v is incident to a clause C if v belongs to the scope of C.

For an assignment $\alpha : V_I \to \{0, 1\}$, define $\mathcal{C}_I(\alpha)$ to be the set of clauses α satisfies in I. We further define $cost_I(\alpha) = |\mathcal{C} \backslash \mathcal{C}_I(\alpha)|$, the number of unsatisfied clauses, to be the *cost* of the assignment. We define the *size* of an instance I as $\max(|V_I|, |\mathcal{C}_I|)$ and denote it by n_I. We ignore the subscripts when the instance is clear in the context.

Now we give formal definitions for the problems we consider. We first define $\text{MinCSP}(\Gamma)$, and then define our central problem $\text{ImproveMaxCSP}(\Gamma)$. In this paper, our main goal is to characterize the parameterized complexity of $\text{ImproveMaxCSP}(\Gamma)$ when $\Gamma = \text{SymLang-N}(r, S)$ for some $r \in \mathbb{N}^+$ and $S \subseteq \{0, 1, \cdots, r\}$.

Problem: $\text{MinCSP}(\Gamma)$
Input: A $\text{CSP}(\Gamma)$ instance $I(V, \mathcal{C})$, an integer $k \in \mathbb{N}$.
Parameter: k.
Output: An assignment α with $cost(\alpha) \leq k$ if it exists, otherwise report that such an assignment does not exist.

Problem: $\text{ImproveMaxCSP}(\Gamma)$
Input: A $\text{CSP}(\Gamma)$ instance $I(V, \mathcal{C})$, an integer $k \in \mathbb{N}$, and $\mathcal{P} \subseteq \mathcal{C}$
Parameter: k
Promise: There exists an assignment $\alpha_0 : V_I \to \{0, 1\}$ such that $|\mathcal{C}_I(\alpha_0) \Delta \mathcal{P}| \leq k$.
Output: A *good* assignment α, where the word *good* simply means $|\mathcal{C}_I(\alpha)| \geq |\mathcal{C}_I(\alpha_0)|$. (We do not require the output α to satisfy $|\mathcal{C}_I(\alpha) \Delta \mathcal{P}| \leq k$.)

From the definition, $\text{MinCSP}(\Gamma)$ is a special case of $\text{ImproveMaxCSP}(\Gamma)$ by setting $\mathcal{P}$ to be the constraint set $\mathcal{C}$.

A boolean constraint language Γ is *trivial* if $\Gamma = \varnothing$ or $\Gamma = \{\{0,1\}^r\}$ for some $r \in \mathbb{N}^+$ and *nontrivial* otherwise. When Γ is trivial, MinCSP(Γ) and ImproveMaxCSP(Γ) can be solved by trivial algorithms. So in the following we only consider nontrivial boolean constraint languages.

2.2 Abbreviations for Special CSPs

For simplicity, we abbreviate boolean constraint languages mentioned in the paper. Here we introduce the following simple but useful claim in identifying equivalence between boolean constraint languages.

Claim 2.1. SymLang-N(r, S) *and* SymLang-N$(r, \{r - x \mid x \in S\})$ *are equivalent boolean constraint languages.*

Proof. From SymRel$(r, S) + (1, 1, \cdots, 1) =$ SymRel$(r, \{r - x \mid x \in S\})$, where $(1, 1, \cdots, 1)$ is the element in $\mathbb{F}_2^r$ that has value 1 in each coordinate, we have SymRel$(r, S) + b =$ SymRel$(r, \{r - x \mid x \in S\}) + (b + (1, 1, \cdots, 1))$ for any $b \in \mathbb{F}_2^r$. □

We will use the following abbreviations in this paper.

- rAND = SymLang-N$(r, \{0\})$ = SymLang-N$(r, \{r\})$.
- rSAT = SymLang-N$(r, \{1, 2, \cdots, r\})$ = SymLang-N$(r, \{0, 1, \cdots, r-1\})$.
- rAE (All Equal) = SymLang-N$(r, \{0, r\})$.
- $\leq$ 1-out-of-rSAT = SymLang-N$(r, \{0, 1\})$ = SymLang-N$(r, \{r-1, r\})$.

We also use abbreviations for the corresponding MinCSP and ImproveMaxCSP problems. For example, ImproveMaxCSP(rAND) corresponds to the problem ImproveMaxCSP(SymLang-N$(r, \{0\})$).

2.3 Our Characterization

Having defined our framework, we first provide the formal version of Theorem 1.1:

Theorem 2.2 (Main Theorem (Formal))**.** *For any $r \in \mathbb{N}^+$ and $S \subseteq \{0, 1, \cdots, r\}$, the problem* ImproveMaxCSP(SymLang-N(r, S)) *is either FPT or W[1]-hard.*

As MinCSP(Γ) is a special case of ImproveMaxCSP(Γ), the W[1]-hardness of the former implies that of the latter. The recent characterization of parameterized tractability of MinCSP(Γ) [KKPW23] implies the following theorem, which is proved in Section 7.

Theorem 2.3. *For nontrivial $\Gamma =$ SymLang-N(r, S) for some $r \in \mathbb{N}^+$ and $S \subseteq \{0, 1, \cdots, r\}$ that is not equivalent to rAND, rAE or $\leq$ 1-out-of-rSAT,* ImproveMaxCSP(Γ) *is W[1]-hard.*

It remains to study the remaining problems.

Remark 2.4. *We say that an algorithm A* solves ImproveMaxCSP(Γ)*, if for any instance that satisfies the promise, A outputs a good assignment. However, the run time is measured on all possible instances. In other words, A runs in $f(k)n^{O(1)}$ time if it terminates and outputs some assignment in $f(k)n^{O(1)}$ time no matter whether the input instance $(I, k, \mathcal{P})$ satisfies the promise. Clearly, only the run time on instances satisfying the promise is critical, as we can always set a run time limit to the algorithm to rule out instances that require too much time to indicate their violation of promise.*

For ImproveMaxCSP(rAND), a color-coding combined with the LP rounding for Densest Subhypergraph yields the following result proved in Section 3.

Theorem 2.5. *There exists an algorithm that solves* ImproveMaxCSP*(rAND) in $2^{O(k \log k)} n^{O(1)}$ time for all $r \geq 1$.*

For rAE, its tractability depends on r. When $r = 2$, we use the celebrated *unbreakability* framework [KT11, CCH+16, CLP+14, LSS22] to design a parameterized algorithm in Section 4.

Theorem 2.6. *There exists an algorithm that solves* ImproveMaxCSP*(2AE) in time $2^{O(k^2)} n^{O(1)}$.*

However, for rAE with $r \geq 3$, we show nontrivial reductions from Paired Minimum st-Cut to show the following hardness in Section 5.

Theorem 2.7. ImproveMaxCSP(rAE) *is W[1]-hard for all $r \geq 3$.*

Finally, for $\leq$ 1-out-of-rSAT, we prove that even the first nontrivial case $r = 2$, which is equivalent to 2SAT, is already W[1]-hard (Section 6). Even though MinCSP(2SAT) is in FPT, this result follows from the (slightly informal) fact that *Vertex Cover and Independent Set are equivalent in our model*.

Theorem 2.8. ImproveMaxCSP($\leq$ 1-out-of-rSAT) *is W[1]-hard for any $r \geq 2$.*

It is easy to see that the other theorems imply Theorem 2.2 in a straightforward way.

3 FPT Algorithm for rAND

The main goal of this section is to prove Theorem 2.5. During the course of the algorithm, we create instances involving AND constraints of different arities. Hence, we will show a slightly stronger result:

Theorem 3.1. *There is an algorithm that solves* ImproveMaxCSP*($\cup_{r' \leq r} r'$AND) in $2^{O(k \log k)} n^{O(1)}$ time for all $r \geq 1$.*

Since $\cup_{r' \leq r} r'$AND $\supseteq$ rAND, Theorem 3.1 immediately shows that for any $r \geq 1$, the problem ImproveMaxCSP(rAND) is FPT.

3.1 Preliminaries

In this section we recall basic definitions about hypergraphs. A *hypergraph* $H(V, E)$ is a generalization of graph where an edge can contain multiple vertices. For $e \in E$ and $v \in V$, say v is incident to e and e is incident to v if e contains v. For a hypergraph H and $V_0 \subseteq V$, we denote $H(V_0)$ as the *induced subhypergraph* of V_0 in H, and denote $E(H(V_0))$ to be the edge set of the induced subhypergraph.

In the main algorithm in this section (provided in Theorem 3.5), we need to solve the following problem:

Problem: Maximum Induced Subhypergraph with Vertex Weights.
Input: A hypergraph $H(V, E)$ and a weight function $w : V \to \mathbb{Z}$.
Output: A vertex subset $V_0 \subseteq V$ that maximizes $|E(H(V_0))| - \sum_{v \in V_0} w(v)$.

The problem is closely related to Densest Subgraph (See [LMFB24] for a survey) and has a polynomial-time algorithm based on similar algorithms for Densest Subgraph. One algorithm is to observe that $|E(H(V_0))| - \sum_{v\in V_0} w(v)$ is supermodular and run a submodular minimization algorithm.[2] The below is an LP-based algorithm specific to our setting.

Lemma 3.2. *There exists a polynomial-time exact algorithm that solves* Maximum Induced Subhypergraph with Vertex Weights.

Proof. Consider the following LP:

$$\begin{array}{lll} \text{maximize} & \sum_{e\in E} x_e - \sum_{v\in V} w(v) y_v & \\ \text{subject to} & x_e \le y_v & \forall e \in E, v \in V \text{ s.t. } e \text{ is incident to } v \\ & x_e \in [0,1] & \forall e \in E \\ & y_v \in [0,1] & \forall v \in V \end{array}$$

Since we use $x_e \le y_v$ to mimic the constraint that "one should select a vertex before selecting incident edges", if the LP is integral (i.e. $x_e, y_v \in \{0,1\}$) then the set of vertices v with $y_v = 1$ in the optimal solution of the LP corresponds to the correct output to the Maximum Induced Subhypergraph with Vertex Weights problem.

The algorithm solves the LP, providing optimal fractional solution $(x_e^\star, y_v^\star)$, and rounds the solution to an integer solution in the following way: Arbitrarily select $p \in (0,1]$ and set $x_e^p = [x_e^\star \ge p], y_v^p = [y_v^\star \ge p]$, where $[P]$ denotes the Iverson bracket and equals to 1 if P is true otherwise 0 for some statement P. The solution is clearly an integral solution to the LP.

Now we show that any such p produces an integral solution that has the same value as that of $(x_e^\star, y_v^\star)$, denoted as $f^\star$. Define $f(p)$ to be the objective function value of the solution (x_e^p, y_v^p). Clearly we have $f(p) \le f^\star$. We further have

$$\begin{aligned} \int_0^1 f(p)\mathrm{d}p &= \int_0^1 \left(\sum_{e\in E} [x_e^\star \ge p] - \sum_{v\in V} w(v)[y_v^\star \ge p] \right) \mathrm{d}p \\ &= \sum_{e\in E} x_e^\star - \sum_{v\in V} w(v) y_v^\star = f^\star \end{aligned} \tag{1}$$

So the set $\{x \in (0,1] \mid f(x) < f^\star\}$ has zero measure. Further from the fact that $f(p)$ is a step function of finite number of "steps" and each "step" has nonzero length, such set must be empty, and $f(p) = f^\star$. □

Our FPT algorithms in Section 3 and Section 4 will use the technique of *color coding* and its derandomization given below.

Theorem 3.3 ([NSS95])**.** *Given a set U of size n and integers $0 \le a, b \le n$, one can in time $2^{O(\min(a,b)\log(a+b))} \cdot n \log n$ construct a family $\mathcal{F}$ of at most $2^{O(\min(a,b)\log(a+b))} \log n$ colorings $U \to \{0,1\}$ such that the following holds: for any sets $A, B \subseteq U$, $A \cap B = \varnothing$, $|A| \le a$, $|B| \le b$, there exists a coloring $\chi \in \mathcal{F}$ with $\chi(a) = 1$ for all $a \in A$ and $\chi(b) = 0$ for all $b \in B$.*

3.2 Algorithm with Variable Solution

Given an instance $(I, k, \mathcal{P})$, we say that $\mathcal{P}$ is *satisfiable* if some assignment satisfies all clauses in $\mathcal{P}$. We will give an algorithm that runs in $2^{O(k\log k)} n^{O(1)}$ time and solves ImproveMaxCSP($\cup_{r'\le r} r'$AND) when the instance satisfies an additional promise that $\mathcal{P}$ is satisfiable.

[2] We thank an anonymous reviewer for this observation.

When this additional promise is satisfied, it is easy to find an assignment satisfying all the clauses of $\mathcal{P}$ in polynomial time, so in the following we assume that the instance $(I, k, \mathcal{P})$ is equipped with such an assignment α. In the next section, we show how to transform any instance into instances where this additional promise is true.

Equipped with this additional assignment, we then restructure the instance in the following way: Define $k' := k + |\mathcal{P} \Delta C_I(\alpha)|$ and $\mathcal{P}' := C_I(\alpha)$, and replace $(I, k, \mathcal{P}, \alpha)$ with $(I, k', \mathcal{P}', \alpha)$, to keep $\mathcal{P}$ to be maximal in accordance with α. In other words, we reset $\mathcal{P}$ to be the set of clauses satisfied by α. If the instance satisfies the promise (that some good assignment is close to the $\mathcal{P}$), the good assignment satisfies at most $|\mathcal{P}| + k$ clauses, which implies that $|\mathcal{P} \Delta C_I(\alpha)| \leq k$, and $k' \leq 2k$. If $k' > 2k$, we return an arbitrary assignment (to handle cases where the promise is not met).

Now we show that, there exists an good assignment close to $\mathcal{P}$ whose Hamming distance to α is small.

Lemma 3.4. *For a* ImproveMaxCSP$(\cup_{r' \leq r} r'\text{AND})$ *instance* $(I, k, \mathcal{P}, \alpha)$*, there exists a good assignment* $\beta : V \to \{0, 1\}$ *with* $|C_I(\beta) \Delta \mathcal{P}| \leq k$ *and* $|\{v \in V \mid \alpha(v) \neq \beta(v)\}| \leq rk$*.*

Proof. Choose a good β with $|C_I(\beta) \Delta \mathcal{P}| \leq k$, and if there are multiple ones, choose the one with the smallest hamming distance to α.

For any v with $\alpha(v) \neq \beta(v)$, consider assignment β' that differs from β only on v. β' has smaller Hamming distance to α than β, so either β' is not good, or β' is good but does not satisfy the promise. Both cases require that some clause $C \in \mathcal{C}$ is satisfied in β but violated in β'. Since all clauses are AND clauses and $\beta'(v) = \alpha(v)$, such clause C is violated in α. Therefore, all variables contributing to the hamming distance are incident to some clauses in $C_I(\beta) \Delta \mathcal{P}$, and the statement follows from the fact that $|C_I(\beta) \Delta \mathcal{P}| \leq k$ and that the arity of all relations in $\cup_{r' \leq r} r'\text{AND}$ is no more than r. □

Now we use the (derandomized) color coding technique along with the algorithm of Lemma 3.2 for Maximum Induced Subhypergraph with Vertex Weights to solve our problem ImproveMaxCSP$(\cup_{r' \leq r} r'\text{AND})$.

Theorem 3.5. *There is an algorithm solving* ImproveMaxCSP$(\cup_{r' \leq r} r'\text{AND})$ *in* $2^{O(k \log k)} n^{O(1)}$ *time when the input instance satisfies an additional promise that* $\mathcal{P}$ *is satisfiable.*

Proof. First use Theorem 3.3 with $a = b = rk$ to obtain a family of at most $2^{O(k \log k)} \log n$ colorings with binary labels for variables. The algorithm then produces an assignment for each coloring and reports the best among them. In the following we fix a particular coloring.

Let L_1 be the set of variables with label 1. Construct a binary relation $\sim$ on L_1, where $u \sim v$ if $u = v$ or some clause $C \in \mathcal{P}$ is incident to both u and v. Since $\sim$ is reflexive and symmetric, its transitive closure is an equivalence relation. Denote $L_1 / \sim$ as the quotient set of the equivalence relation which includes all equivalence classes of the transitive closure. The intuition is that, the algorithm will produce the answer from α by flipping several variables with label 1, and variables in one equivalence class are considered a group and would be flipped or kept at the same time.

We then construct a hypergraph $H(V_H, E_H)$ with weight function $w : V_H \to \mathbb{Z}$ as follows:

- For each equivalence class in $L_1 / \sim$, introduce a vertex v in V_H, whose weight $w(v)$ equals to the number of clauses in $\mathcal{P}$ incident to any variable in the equivalence class. Intuitively, the weight measures that how many clauses we will lose if we flip this equivalence class.
- For each clause $C \in \mathcal{C}_I \backslash \mathcal{P}$, if C can be satisfied by flipping L_1 from α, introduce a hyperedge incident to all equivalence classes containing incident vertices of C. So the clause will be satisfied if we flip all equivalence classes incident to this hyperedge.

Finally, the algorithm uses the algorithm for Maximum Induced Subhypergraph with Vertex Weights proposed in Lemma 3.2 to find $V_0 \subseteq V_H$ that maximizes $|E(H(V_0))| - \sum_{v \in V_0} w(v)$, and returns the assignment produced by flipping in α all variables belonging to equivalence classes in V_0.

The algorithm runs in $2^{O(k \log k)} n^{O(1)}$ time since the whole process can be done in polynomial time except trying all $2^{O(k \log k)} n^{O(1)}$ colorings produced by Theorem 3.3. Now we show that the algorithm produces a good assignment for instances satisfying the promise that some good assignment is close to the $\mathcal{P}$. Since we finally choose the best assignment among all colorings, it is sufficient to show that there exists some coloring that produces a good assignment.

Fix any good assignment β with $|C_I(\beta) \Delta \mathcal{P}| \leq k$ and $|\{v \in V \mid \alpha(v) \neq \beta(v)\}| \leq rk$, where α is an assighnment such that $\mathcal{P} = \mathcal{C}_I(\alpha_0)$. According to Lemma 3.4 such β exists. Define $N(C_I(\beta)\Delta\mathcal{P})$ to be all variables incident to any clause in $C_I(\beta)\Delta\mathcal{P}$. Define $V_0 = \{v \in N(C_I(\beta)\Delta\mathcal{P}) \mid \alpha(v) = \beta(v)\}$ and $V_1 = N(C_I(\beta)\Delta\mathcal{P}) \backslash V_0$. Note that both $|V_0|, |V_1| \leq rk$.

According to Theorem 3.3, there exists a coloring assigning V_0 to 0 and V_1 to 1. We consider the behavior of the algorithm when trying this coloring.

We first claim that, there exists a set of equivalence classes in $L_1/\sim$ whose union is exactly V_1. This is equivalent to the statement that there is no $u \in V_1$ and $v \in L_1 \backslash V_1$ such that $u \sim v$. Suppose $u \sim v$ and $u \in V_1$, then some $C \in \mathcal{P}$ is incident to both of them. Since C is incident to $u \in V_1$ which satisfies $\alpha(u) \neq \beta(u)$, C is violated by β, so $C \in \mathcal{C}_I(\beta)\Delta\mathcal{P}$. Then $v \in N(C_I(\beta)\Delta\mathcal{P})$, which finishes the proof since it implies $v \notin L_1 \backslash V_1$.

The previous statement shows that, there exists some $V^\star \subseteq V_H$, such that by flipping from α all variables in equivalence classes corresponding to $V^\star$, one can produce β. We further show the following two claims:

- For any $V \subseteq V_H$, suppose α' is produced by flipping from α all variables in every equivalence class in V, then $cost_I(\alpha) - cost_I(\alpha') \leq |E(H(V))| - \sum_{v \in V} w(v)$. Notice that the latter is the objective of the Maximum Induced Subhypergraph with Vertex Weights problem. In other words, the cost of α' would not be underestimated. To see this, consider clauses in $C_I(\alpha)\Delta C_I(\alpha')$. Clauses satisfied in α but unsatisfied in α' are those belonging to $\mathcal{P}$ and incident to V, counted exactly by $\sum_{v \in V} w(v)$. Clauses satisfied in α' but unsatisfied in α are partially counted by $|E(H(V))|$, since it does not include satisfied clauses in α' violated by the assignment produced from flipping L_1 in α. However, all hyperedges in $E(H(V))$ correspond to clauses satisfied in α' but unsatisfied in α.

- $cost_I(\alpha) - cost_I(\beta) = |E(H(V^\star))| - \sum_{v \in V^\star} w(v)$, i.e. the cost of β is correctly estimated by the Maximum Induced Subhypergraph with Vertex Weights objective. We only need to show that $E(H(V^\star))$ contains all clauses satisfied in β but unsatisfied in α. Since the coloring colors V_0 with 0 and V_1 with 1, all clauses satisfied in β but unsatisfied in α have label-1 incident variables inside $V_1 \subseteq L_1$, so the clauses will introduce hyperedges in H whose scope is inside $V^\star$, so they are all counted by $|E(H(V^\star))|$.

The previous two claims imply that, $V^\star$ is one of the optimal solutions to the Maximum Induced Subhypergraph with Vertex Weights problem for the constructed hypergraph H, and any optimal solution of the Maximum Induced Subhypergraph with Vertex Weights problem corresponds to a good assignment to the CSP instance, which finishes the proof. □

3.3 From Clause Solution to Variable Solution for rAND

Now we show how to solve ImproveMaxCSP($\cup_{r' \leq r} r'$AND) using Theorem 3.5. We first define an auxiliary operation, which essentially fixes a given variable to a fixed boolean value $a \in \{0, 1\}$.

Definition 3.6 (Value Assignment)**.** *For a* ImproveMaxCSP($\cup_{r' \le r} r'$AND) *instance* $(I(V, \mathcal{C}), k, \mathcal{P})$*, a variable* $v \in V$ *and a boolean value* $a \in \{0, 1\}$*, to "assign* v *to* a *in* $(I, k, \mathcal{P})$*", we mean producing a* ImproveMaxCSP($\cup_{r' \le r} r'$AND) *instance* $(I'(V', \mathcal{C}'), k', \mathcal{P}')$ *in the following way:*

- $V' = V \backslash v$*. Initially,* $k' = k$*.*
- *For all clauses* $C(V_C, R_C) \in \mathcal{C}$*, define* $C' = (V'_C = V_C \backslash \{v\}, R'_C = \{r_{V_C \backslash \{v\}} \mid r \in R, r(v) = a\})$ *if* $v \in V_C$ *and otherwise* $C' = C$*. If* $R' = \varnothing$ *and* $C \in \mathcal{P}$ *or* $R' = \{0, 1\}^{r-1}$ *and* $C \notin \mathcal{P}$*, decrease* k' *by* 1*. If* R' *is not trivial, add* C' *to* $\mathcal{C}'$*, and add* C' *to* $\mathcal{P}'$ *if* $C \in \mathcal{P}$*.*

Notice that the instance $(I', k', \mathcal{P}')$ *may not satisfy the promise that there exists a good assignment* α_0 *with* $|\mathcal{C}_{I'}(\alpha_0) \Delta \mathcal{P}'| \le k'$*.*

The following claims are straightforward from the definition.

Claim 3.7. *Suppose instance* $(I', k', \mathcal{P}')$ *is produced from assigning* v *to* a *in* $(I, k, \mathcal{P})$*, and there exists an assignment* α *with the least cost for* I *which satisfies* $\alpha(v) = a$ *and* $|C_I(\alpha) \Delta \mathcal{P}| \le k$*, then* $\alpha_{V'}$ *has the least cost for* I' *and* $|C_{I'}(\alpha_{V'}) \Delta \mathcal{P}'| \le k'$*.*

Claim 3.8. *Suppose instance* $(I', k', \mathcal{P}')$ *is produced from assigning* v *to* a *in* $(I, k, \mathcal{P})$*, then for all assignments* $\alpha, \beta : V_I \to \{0, 1\}$ *with* $\alpha(v) = \beta(v) = a$*,* $cost_I(\alpha) - cost_{I'}(\alpha_{V'}) = cost_I(\beta) - cost_{I'}(\beta_{V'})$*.*

Now we show the algorithm, which branches on variables involved in conflicts in the solution to produce instances in which some assignment satisfies the solution. Notice that since we use Definition 3.6 to do branching, some ImproveMaxCSP($\cup_{r' \le r} r'$AND) instances which do not satisfy the promise that some good solution is close to the current solution may be created.

Theorem 3.9. *Suppose there exists an algorithm* A *that runs in* $2^{O(k \log k)} n^{O(1)}$ *time, solves* ImproveMaxCSP($\cup_{r' \le r} r'$AND) *when the input instance to* A *satisfies an additional promise that* $\mathcal{P}$ *is satisfiable. Then there is an algorithm that solves* ImproveMaxCSP($\cup_{r' \le r} r'$AND) *in* $2^{O(k \log k)} n^{O(1)}$ *time (with calls to algorithm* A*).*

Proof. Given a ImproveMaxCSP($\cup_{r' \le r} r'$AND) instance $(I, k, \mathcal{P})$, consider the following algorithm:

- If $k \le 0$, return any assignment (to handle instances violating the promise).
- Otherwise, if there exists $v \in V$ and $C_1, C_2 \in \mathcal{P}$ such that C_1 contains literal v while C_2 contains literal $\neg v$, branch on variable v: Produce two instances $(I_0, k_0, \mathcal{P}_0)$ and $(I_1, k_1, \mathcal{P}_1)$ by assigning v to 0 respectively 1, and solve two instances, producing assignments α_0 and α_1. Extend two assignments to V with $\alpha_0(v) = 0$ and $\alpha_1(v) = 1$ and return the assignment with smaller cost in I.
- Otherwise, run algorithm A to solve the instance $(I, k, \mathcal{P})$.

We do induction on k to prove the correctness of the algorithm. In the base case, when $k < 0$, the instance cannot satisfy the promise. Now suppose the statement holds for all integer less than k.

- If there exists $v \in V$ and $C_1, C_2 \in \mathcal{P}$ such that C_1 contains literal v while C_2 contains literal $\neg v$, then any assignment for I would falsify one of C_1 and C_2. So the two produced instances $(I_0, k_0, \mathcal{P}_0)$ and $(I_1, k_1, \mathcal{P}_1)$ have $k_0 < k, k_1 < k$, and the algorithm would finally output some assignments for them from the induction hypothesis. Further, if $(I, k, \mathcal{P})$ satisfies the promise, there exists a good α such that $|C_I(\alpha) \Delta \mathcal{P}| \le k$. Without loss of generality suppose $\alpha(v) = 0$, then from Claim 3.7 $(I_0, k_0, \mathcal{P}_0)$ satisfies the promise, and the algorithm would produce the good assignment for $(I_0, k_0, \mathcal{P}_0)$. Further according to Claim 3.8, extending the assignment with $\alpha(v) = 0$ results in a good assignment for $(I, k, \mathcal{P})$.

- Otherwise, for all $v \in V$, at most one of the literals v and $\neg v$ appears in $\mathcal{P}$. Assigning variables to satisfy the appeared literal, we end up with an assignment satisfying all clauses in $\mathcal{P}$. So the correctness of the algorithm follows from A.

The runtime of the algorithm follows the recursion

$$T(k) = \max\left(2 \max_{0 \le i \le k-1} T(i) + n^{O(1)}, 2^{O(k \log k)} n^{O(1)}\right) \tag{2}$$

since the instance size never increases within the recursion. $T(k) \le 2^{O(k \log k)} n^{O(1)}$ satisfies the recursion. □

Proof of Theorem 3.1. Combining the algorithms in Theorem 3.5 and Theorem 3.9, we get the algorithm that solves ImproveMaxCSP($\cup_{r' \le r} r'$AND) in $2^{O(k \log k)} n^{O(1)}$ time. □

4 FPT Algorithms for 2AE

In this section we prove Theorem 2.6.

4.1 Preliminaries

We recall useful definitions about graphs. Let $G = (V, E)$ be a multigraph, where parallel edges and loops are allowed. For two disjoint vertex sets $A, B \subseteq V$, we denote by $E(A, B)$ the set of edges between A and B. When (A, B) is a partition, then $E(A, B)$ is the set of edges in the corresponding cut. For a vertex subset $V' \subseteq V$, we denote by $G(V')$ the subgraph of G induced by V'. We write $E(G(V'))$ for the set of edges in $G(V')$. For a set $E' \subseteq E$, we denote by $G \setminus E'$ the multigraph $(V, E \setminus E')$. For $S \subseteq V$, we let $N(S) := \{v \in V \setminus S : \exists u \in S \text{ such that } (u, v) \in E\}$ be the open neighborhood of S. We say two cuts (A, B) and (X, Y) are k-close if $|E(A, B) \Delta E(X, Y)| \le k$. When the graph is clear in the context, we define $n = \max(|V|, |E|)$ to be the size of the graph.

Notice that we can see clauses as edges connecting variables, assigning boolean values to variables as partitioning the variable set into two sets, and the equality and non-equality constraints correspond to uncut and cut constraints respectively. This motivates us to introduce the following graph problem. Given a graph $G = (V, E)$, an edge type function $t : E \to \{0, 1\}$ and a partition (A, B) of V, define $C(A, B) = (t^{-1}(1) \cap E(A, B)) \cup (t^{-1}(0) \cap (E \setminus E(A, B)))$ to be the set of edges satisfied by (A, B), where edges of type 1 need to be separated, while edges of type 0 should be inside some vertex set.

Problem: ImproveMaxCut-Uncut
Input: A connected multigraph $G(V, E)$, an integer k, $\mathcal{P} \subseteq E$, and an edge type function $t : E \to \{0, 1\}$.
Parameter: k
Promise: There exists some (A, B) that maximizes $|C(A, B)|$ and satisfies $|C(A, B) \Delta \mathcal{P}| \le k$.
Output: Any partition (A, B) of vertices that maximizes $|C(A, B)|$. (Notice that we do not require $|C(A, B) \Delta \mathcal{P}| \le k$ for the output.)

Clearly, if the given graph for ImproveMaxCut-Uncut is not connected, we can solve each connected component separately, so connectivity of G does not lose any generality for the problem. The following claim is straightforward.

Claim 4.1. ImproveMaxCSP(2AE) *and* ImproveMaxCut-Uncut *are equivalent.*

In the following, we present the algorithm for ImproveMaxCut-Uncut.

4.2 From Edge Solution to Vertex Solution

We first apply a pre-processing algorithm to the given ImproveMaxCut-Uncut instance, intending to make it in accordance with some partition of the vertex set.

Lemma 4.2. *There exists an algorithm that given a* ImproveMaxCut-Uncut *instance, runs in* $4^k n^{O(1)}$ *time and finds a bipartition* (A, B) *of the vertex set* V *such that there exists a* good *partition* $(A^\star, B^\star)$ *which satisfies* $|C(A^\star, B^\star)\Delta\mathcal{P}| \leq k$ *and* $|C(A^\star, B^\star)\Delta C(A, B)| \leq 3k$.

Proof. Fix any $(A^\star, B^\star)$ satisfying $|C(A^\star, B^\star)\Delta\mathcal{P}| \leq k$. We first find the largest subset $\mathcal{P}_1 \subseteq \mathcal{P}$ that can be simutaneously satisfied by some bipartition using a $4^k n^{O(1)}$-time algorithm for MinCSP(2AE) [DJO+23]: Since $C(A^\star, B^\star) \cap \mathcal{P}$ is one possible solution, this means that $|\mathcal{P}_1| \geq |C(A^\star, B^\star) \cap \mathcal{P}| \geq \max(\mathcal{P}, |C(A^\star, B^\star)|) - k$. Let (A, B) be any bipartition satisfying $\mathcal{P}_1$. Consider $C(A, B)$: It contains all edges in $\mathcal{P}_1$ and perhaps more. However, since the size cannot be larger than $|C(A^\star, B^\star)|$, there are at most k edges in $C(A, B) \setminus \mathcal{P}_1$. Now we have

$$|C(A, B)\Delta C(A^\star, B^\star)| \leq |\mathcal{P}\Delta C(A^\star, B^\star)| + |\mathcal{P} \setminus \mathcal{P}_1| + |C(A, B) \setminus \mathcal{P}_1| \leq 3k.$$

This completes the proof. □

From Lemma 4.2, we can set $\mathcal{P}$ to be such $C(A, B)$ and triple k to change the original ImproveMaxCut-Uncut instance to an equivalent ImproveMaxCut-Uncut instance, whose $\mathcal{P}$ is given by some $C(A, B)$ for a vertex partition (A, B).

4.3 Algorithms for Instances with Vertex Solution

In the rest of this section, we prove the following theorem:

Theorem 4.3. *There exists an algorithm that, given a* ImproveMaxCut-Uncut *instance* $(G(V, E), k, \mathcal{P}, t)$ *where* $\mathcal{P} = C(A, B)$ *for some partition* (A, B) *of* V*, finds a partition* (X, Y) *that maximizes* $|C(X, Y)|$ *in time* $2^{O(k^2)} n^{O(1)}$.

We first show how this theorem implies the main theorem.

Proof of Theorem 2.6. According to Claim 4.1, we only need to find such algorithm for ImproveMaxCut-Uncut. For a ImproveMaxCut-Uncut instance, we first apply the algorithm stated in Lemma 4.2 to produce an instance equivalent to the input whose $\mathcal{P}$ is given by some $C(A, B)$ for a vertex partition. Then we apply the algorithm stated in Theorem 4.3 to solve the new instance. The runtime is $4^k n^{O(1)} + 2^{O(k^2)} n^{O(1)} = 2^{O(k^2)} n^{O(1)}$. □

Recall that we are given a vertex partition (A, B) as the solution, while there exists a good partition (X, Y) with $|C(A, B)\Delta C(X, Y)| \leq k$. Let $AX := A \cap X$ and define AY, BX, BY similarly.

Our high-level approach follows the framework first introduced by Kawarabayashi and Thorup [KT11] for Minimum k-cut and its refinement by Chitnis et al. [CCH+16]. Let us define the following *terminal version* of the problem. We will fix k to be the parameter of the initial instance, which *remains invariant throughout the recursive calls to the terminal version.*

Problem: ImproveMaxCut-Uncut Terminal Version
Input: A connected multigraph $G(V, E)$, an integer $k' \leq k$, a $\mathcal{P} \subseteq E$, an edge type function $t : E \to \{0, 1\}$, a set of terminals $T \subseteq V$ with $|T| \leq 2k$ (Notice that it is k rather than k') and a *marked edge set* $M \subseteq E$.
Parameter: k'
Promise:

- $\mathcal{P} = C(A, B)$ for some partition (A, B);
- M is a matching allowing parallel edges: Any two edges in M are either completely disjoint or parallel copies of each other.
- There exists a good cut (X, Y) with $|C(A, B)\Delta C(X, Y)| \leq k'$ and $M \subseteq E(X, Y) \cap E(A, B)$.

Requirement: For *each* function $f : T \to \{X, Y\}$ and $0 \leq k'' \leq k$, output a cut $(X_{f,k''}, Y_{f,k''})$ satisfying all the following or output nothing:

1. consistent with f: $f(v) = X$ iff $v \in X_{f,k''}$ for all $v \in T$,
2. consistent with M: $M \subseteq E(A, B) \cap E(X_{f,k''}, Y_{f,k''})$,
3. k''-close to (A, B): $|C(A, B)\Delta C(X_{f,k''}, Y_{f,k''})| \leq k''$,

For any (X, Y) which is a good cut that satisfies $|C(A, B)\Delta C(X, Y)| \leq k''$ and $M \subseteq E(X, Y) \cap E(A, B)$ for some $k'' \leq k'$, the output $(X_{f^*,k''}, Y_{f^*,k''})$, where $f^\star$ is consistent to (X, Y), should additionally be a good cut (not necessarily equal to (X, Y)).

Clearly, ImproveMaxCut-Uncut is a special case with $M = T = \varnothing$ after doing the preprocessing in Lemma 4.2. In the following we introduce the algorithm solving ImproveMaxCut-Uncut Terminal Version.

Let $q := k^2 2^{2k+2}$. Before introducing the algorithm, we require one additional definition.

Definition 4.4 ((k, q)-cut)**.** Let $G = (V, E)$ be a multigraph and let $M \subseteq E$ be the set of marked edges which is a matching allowing parallel edges. A cut $L \subseteq V$ is called (k, q)-balanced (more simply a (k, q)-cut) if the following conditions are met.

- $|E(L, V \setminus L)| \leq k$.
- Both $G(L)$ and $G(V \setminus L)$ are connected (using both edges inside and outside M).
- Both $G(L)$ and $G(V \setminus L)$ contains at least q non-marked edges (edges outside M).

Our algorithm follows a simple win-win strategy. If there is no (k, q)-cut, a simple color-coding algorithm will solve the terminal problem. If there is a (k, q)-cut, by recursively solving a smaller subproblem, one can reduce the number of unmarked edges. Combining this with the algorithm to compute a (k, q)-cut, one can solve the terminal version in FPT time.

Now we present each of the three parts. We start by directly solving the case where there is no (k, q)-cut, using the (derandomized) coloring coding technique.

Claim 4.5. *One can solve* ImproveMaxCut-Uncut Terminal Version *in* $2^{O(k \log q)} n^{O(1)}$ *time when there is no* (k, q)*-cut in the instance.*

Proof. Given an ImproveMaxCut-Uncut Terminal Version instance $I = (G, k', \mathcal{P} = C(A,B), t, T, M)$, the algorithm considers every $f : T \to \{X, Y\}$ and $0 \le k'' \le k'$ and does the following. Assume that f and k'' is consistent with some good cut (X, Y), because apart from running time, the requirement only concerns for the correct f and k''.

Note that the set of edges $S = C(A,B)\Delta C(X,Y) = E(A,B)\Delta E(X,Y) = E(AX, BX) \cup E(AY, BY) \cup E(AX, AY) \cup E(BX, BY)$ forms a cut between $L := AX \cup BY$ and $R := AY \cup BX$, no matter how t assigns type to edges. Since G is connected and $|S| \le k''$, $G \setminus S$ has at most $k'' + 1$ connected components.

Assume towards a contradiction that there are two connected components L', R' of $G \setminus S$ such that $L' \subseteq L$ and $R' \subseteq R$ and both $G(L')$ and $G(R')$ have at least q non-marked edges. Then the minimum cut separating L' and R' in G will cut at most $k'' \le k$ edges where the two resulting parts are connected and contain L' and R' respectively, which contradicts the fact that there is no (k, q)-cut. [3]

Therefore, at least one of L and R (say R) has the property that every connected component of $G \setminus S$ contained in R has at most q non-marked edges. Since marked edges form a matching, the number of vertices in such a component can be at most $2q + 2$, so we have $|R| = O(qk)$.

Let $\{R_i\}_{i=1}^{\ell}$ be the connected components of $G(R)$. Define $a_i^+ = |C(A\Delta R_i, B\Delta R_i) \backslash C(A,B)|$ and $a_i^- = |C(A,B) \backslash C(A\Delta R_i, B\Delta R_i)|$. Notice that $|C(X,Y)| = |C(A,B)| + \sum_{i=1}^{\ell}(a_i^+ - a_i^-)$ and $|C(X,Y)\Delta C(A,B)| = \sum_{i=1}^{\ell}(a_i^+) + (a_i^-) \le k'$. Guessing the correct values of ℓ and a_i^+, a_i^- takes at most $k^{O(k)}$ time.

Let $N = N(R)$ be the open neighbor of R in G. Note that $|N| \le |E(R, V \backslash R)| = |C(A,B)\Delta C(X,Y)| \le k''$. Use Theorem 3.3 with $a = |R|, b = k$ to have a family of at most $2^{O(k \log q)} \log n$ colorings that color each vertex using $\{0, 1\}$. There exists a coloring χ that colors R with 1 and colors N with 0. For each $i \in [\ell]$, find all candidates $R_i' \subseteq V$ that is *identical* to R_i with respect to the guessed information about R_i. Formally, R_i' is a connected component of G after deleting all vertices of color 0, and in line with the guessing a_i^+ and a_i^-.

For the correct coloring χ, such R_i' exists since R_i satisfies the condition. Also, finding all such candidates R_i' is easy in polynomial time since one only needs to scan connected components after deleting color-0 vertices.

For any $R_1', \cdots, R_\ell'$ that are disjoint, since R_i' and $N(R_j')$ are disjoint for any $i, j \in [\ell]$, the set of edges newly cut/uncut by updating $A \leftarrow A\Delta R_i'$ and $B \leftarrow B\Delta R_i'$ are disjoint for all i. Therefore, letting $R' := \cup_i R_i'$ and letting $A' \leftarrow A\Delta R'$ and $B' \leftarrow B\Delta R'$ the resulting partition has $|C(A', B')| = |C(A,B)| + \sum_i (a_i^+ - a_i^-) = |C(X,Y)|$ and $|C(A', B')\Delta C(A,B)| = \sum_i (a_i^+ + a_i^-) \le k''$, so the solution is good and $k''-$close to (A, B).

We finally need the solution (A', B') to be consistent with both f and M. For any vertex $v \in T$, if v has color 0, then $v \in A$ if and only if $v \in X$ (recall $R = AY \cup BX$), otherwise by including v in R, one can change the belonging of v. So the consistency of f requires certain components to be included into or excluded from R_i'. For $(u, v) \in M$, if u and v has the same color then $M \in E(A,B) \cap E(X,Y)$ if and only if $M \in E(A,B)$ which is always satisfied, otherwise $M \in E(A,B) \cap E(X,Y)$ if and only if the label-1 vertex is not in R, so consistency of M requires certain components to be excluded from R_i'.

The consistency constraints are all in the form of "removing a component from the candidates" or "impose certain components to be selected by R_i'", which is easy to find out and handle in polynomial time.

We finally calculate the running time of the algorithm. The algorithm first tries all possible (f, k'') in $2^{O(k)}$ time, tries all colorings in $2^{O(k \log q)} n^{O(1)}$ time, and finally guesses the correct values

[3] We use the fact that any minimum st-cut in a connected graph have both sides connected.

of ℓ and a_i^+, a_i^- in $2^{O(k \log k)}$ time. All other processes are in polynomial time. So the total running time is $2^{O(k \log q)} n^{O(1)}$. □

Next we show how to compute a (k, q)-cut if it exists.

Claim 4.6. *There exists an algorithm that runs in time $2^{O(k \log q)} n^{O(1)}$ and returns a (k, q)-balanced cut for a connected graph if it exists.*

Proof. Suppose that $L \subseteq V$ is a (k, q)-cut and $R = V \setminus L$. We first show that there exists $E_L \subseteq E(L)$ that induces a connected subgraph and has at most $O(q)$ edges but at least q non-marked edges. Consider a procedure where we start from $V_L = \{v\}$ for an arbitrary vertex $v \in L$, and iteratively add one arbitrary vertex $u \in L \cap N(V_L)$ to V_L until $E(G(V_L))$ contains at least q non-marked edges. Since $E(G(V_L))$ is connected and increased by at least one while marked edges form a matching, $|V_L| \leq 2q + 2$. Then we choose $E_L \subseteq E(G(V_L))$ that contains at least q non-marked edges while (V_L, E_L) is connected and $|E_L| \leq O(q)$. Define V_R and E_R similarly.

Use Theorem 3.3 with $a = O(q)$ and $b = k$ to try at most $2^{O(k \log q)} \log n$ colorings of edges with 2 colors. One coloring χ colors $E(L, R)$ with 0 and $E_L \cup E_R$ with 1. After deleting all edges of color 0, there are two connected components V'_L and V'_R such that $E_L \subseteq E(G(V'_L))$ and $E_R \subseteq E(G(V'_R))$. Note that $V'_L \neq V'_R$ because L and R are separated by removing edges of color 0.

Then, trying every pair (U', V') of connected components after deleting 0-colored edges and computing the minimum cut separating U' and V' in the original graph will yield a cut of size at most k where each side is connected has at least q non-marked edges. We use the fact that in a connected graph, any minimum s-t cut has both of its sides connected. □

Finally, we prove that the existence of a (k, q)-cut, combined with a recursion, allows us to make progress by reducing the number of unmarked edges. Let $T(m)$ be the running time of our algorithm to solve the terminal version with m unmarked edges.

Claim 4.7. *Suppose there exists a (k, q)-cut, in time $T(\ell) + n^{O(1)}$, one can reduce the current instance to another* ImproveMaxCut-Uncut Terminal Version *instance with at least $\ell - q/2$ fewer unmarked edges for some $\ell \in [q, m - q]$.*

Proof. Let (L, R) be a (k, q)-cut. Without loss of generality, suppose that L has at most k terminals from T, and let $T_L \subseteq L$ be the set of vertices that have an edge to R. Letting $T' = (T \cap L) \cup T_L \subseteq L$ be the new set of terminals for L, we know that $|T'| \leq 2k$. Let ℓ be the number of unmarked edges in L and m be the total number of unmarked edges.

Recursively solve the terminal version with input $(G(L), (L \cap A, L \cap B), k', T', M \cap E(G(L)))$. For each $f' : T' \to \{X, Y\}$ and $0 \leq k'' \leq k'$, it returns a partition $(X_{f',k''}, Y_{f',k''})$ of L that is (1) consistent with f', (2) consistent with $M \cap E(G(L))$, and (3) k''-close to $(L \cap A, L \cap B)$ (or nothing).

Let $f^* : T' \to \{X, Y\}$ be the correct guessing with respect to the good partition (X, Y) and k^* be the correct closeness parameter $|C(A \cap L, B \cap L) \Delta C(X \cap L, Y \cap L)|$ in L. We update (X, Y) with $X \leftarrow (X \cap R) \cup X_{f^*,k^*}$ and $Y \leftarrow (Y \cap R) \cup Y_{f^*,k^*}$. Note that the new (X, Y) is still a good cut since $(X_{f^*,k^*}, Y_{f^*,k^*})$ is good within L and the interaction between L and R only depends on the terminals T', where the previous and new solutions agree. By the guarantee (2) and (3) in the above paragraph, the new (X, Y) is still k'-close to (A, B) and consistent with M.

Since L has at least $q = k^2 2^{2k+2}$ non-marked edges and there are at most $2^{2k}(k + 1)$ possible values for f' and k'', we have $|E(L) \setminus (\cup_{f',k''}(C(A \cap L, B \cap L) \Delta C(X_{f',k''}, Y_{f',k''}))| \geq \ell - q/2$. These edges do not belong to $C(A, B) \Delta C(X, Y) = E(A, B) \Delta E(X, Y)$ for sure. For each such edge e, perform the following operation.

- If $e \notin E(A,B)$, which means that e is not cut in the current solution, this means that e is not cut in the good solution too. Contract it.
- If $e \in E(A,B)$, mark it. To ensure that the set of marked edges M is a matching, if $e=(u,v), f=(v,w)$ are both in M with $u \neq w$, then we know that u,w will be in the same side in the good solution, so we can contract them.

By doing this, the number of unmarked edges is decreased by at least $\ell - q/2$. □

With the three main claims, we finish the proof of Theorem 4.3.

Proof of Theorem 4.3. From the definition of the terminal version, given graph G and the current solution (A,B), running the terminal version with $T=M=\varnothing$ and $k'=k$ we can find a good cut given the promise $|C(A,B)\Delta C(X,Y)| \leq k$ for some (possibly different) good cut (X,Y).

Let us analyze the running time. As defined previously, let $T(m)$ be the running time of our algorithm with m unmarked edges. Since the graph is connected and marked edges form a matching, we have $m \geq \frac{n}{2}$. Our algorithm first uses Claim 4.6 to find a (k,q)-cut in time $2^{O(k\log q)}n^{O(1)}$. If there is none, it uses Claim 4.5 to solve the terminal version in time $2^{O(k\log q)}n^{O(1)}$. Otherwise, it uses Claim 4.7 to reduce the number of unmarked edges by $\ell - q/2$ in time $T(\ell)+n^{O(1)}$, for some $\ell \in [q, m-q]$. Therefore, we have the following recurrence relation:

$$T(m) \leq 2^{O(k\log q)}n^{O(1)} + \max_{\ell\in[q,m-q]}\Big(T(\ell)+T(m-(\ell-q/2))+n^{O(1)}\Big).$$

Using $m \in \left[\frac{n}{2}, n^2\right]$, $T(m) \leq 2^{O(k\log q)}n^{O(1)}$ satisfies the above. □

5 Hardness Proof for rAE for $r \geq 3$

In this section, we prove Theorem 2.7. We start with a simple lemma addressing that hardness result for 3AE is enough.

Lemma 5.1. *If* ImproveMaxCSP(3AE) *is W[1]-hard, then for any integer* $r \geq 3$, ImproveMaxCSP(rAE) *is W[1]-hard.*

Proof. For each integer $r \geq 3$, we give a reduction from ImproveMaxCSP(rAE) to ImproveMaxCSP($(r+1)$AE), and the statement follows from induction. Given an ImproveMaxCSP(rAE) instance $(I,k,\mathcal{P})$, the ImproveMaxCSP($(r+1)$AE) instance $(I',k,\mathcal{P}')$ is constructed in the following way:

- Initially, $V'=V$.
- For each clause $C \in \mathcal{C}$, introduce a variable v_C to V'. Choose any variable v whose literal appears in C, and introduce a clause C' to $\mathcal{C}'$, which is satisfied when C is satisfied and additionally $v=v_C$. It is easy to see that C' is a $(r+1)$AE clause. If $C \in \mathcal{P}$ then add $\mathcal{C}'$ to $\mathcal{P}'$.

Since variable v_C is only incident to C', $v=v_C$ in the good assignment for any clause C. Then the equivalence of two instances is straightforward. □

Using this approach of adding dummy variables, it is straightforward to further show the following.

Claim 5.2. *For all integer $r \geq 3$, if* ImproveMaxCSP($\cup_{r' \leq r} r'$AE) *is W[1]-hard, then* ImproveMaxCSP(rAE) *is W[1]-hard.*

Now we introduce the auxiliary W[1]-hard problem we will use in our hardness reduction, which originates from [MR09] and serves as the source problem of all hardness reductions in [KKPW23].

Problem: Paired Minimum st-Cut.
Input: A directed acyclic graph $G = (V, E)$, vertices $s, t \in V$, an integer $l \in \mathbb{N}$, a partition $\mathcal{E}$ of E into pairs (sets of size 2), and a partition of E into a set $\mathcal{F}$ of $2l$ arc-disjoint st-paths.
Parameter: l.
Output: An st-Cut Z that touches (has non-empty intersection with) at most l pairs from $\mathcal{E}$ if exists, otherwise report that such cut does not exist.

Lemma 5.3 ([KKPW23])**.** Paired Minimum st-Cut *is W[1]-hard.*

We first present the proof for ImproveMaxCSP($\cup_{r' \leq 4} r'$AE), which is more intuitive.

Theorem 5.4. ImproveMaxCSP($\cup_{r' \leq 4} r'$AE) *is W[1]-hard.*

Proof. We give a reduction from Paired Minimum st-Cut, whose hardness is by Lemma 5.3, to ImproveMaxCSP($\cup_{r' \leq 4} r'$AE). Given an Paired Minimum st-Cut instance $(G, s, t, l, \mathcal{E}, \mathcal{F})$, we construct the following ImproveMaxCSP($\cup_{r' \leq 4} r'$AE) instance $(I, k, \mathcal{P})$:

- For each vertex $u \in V_G$, introduce a variable x_u in V_I.
- For each edge $(u, v) \in E$, introduce a 2AE clause $x_u = x_v$ in both $\mathcal{C}_I$ and $\mathcal{P}$. In the rest of this proof, we refer to such clauses as *edge clauses*.
- For each pair $((u_1, v_1), (u_2, v_2)) \in \mathcal{E}$, introduce a 4AE clause $x_{u_1} = \overline{x_{v_1}} = x_{u_2} = \overline{x_{v_2}}$ in $\mathcal{C}_I$.
- Further introduce $(l + 1)$ copies of the 2AE clause $x_s \neq x_t$ to $\mathcal{C}_I$. Set $k = 4l + 1$.

Now we prove the equivalence of the instances. Suppose the given Paired Minimum st-Cut instance has an st-cut Z touching l pairs, we construct the following assignment α for the ImproveMaxCSP($\cup_{r' \leq 4} r'$AE) instance: Assign x_u to 0 for $u \in V_G$ if u is in the s-side of Z, otherwise, when u is in the t-side, assign x_u to 1. α satisfies the following clauses:

- All edge clauses except for the $2l$ clauses corresponding to edges in Z;
- l 4AE clauses corresponding to the l edge pairs in Z (Notice that the tails of all edges in Z must both belong to s-side);
- $(l + 1)$ copies of the $x_s \neq x_t$ clause.

So $|C_I(\alpha) \Delta \mathcal{P}| \leq 4l + 1$ and $|C_I(\alpha)| - |\mathcal{P}| = 1$. Now we only need to check α is good.

Consider any assignment $\beta : V_I \to \{0, 1\}$ with $|C_I(\beta)| > |\mathcal{P}|$. Define $Z_\beta = \{(u, v) \in E \mid x_u \neq x_v\}$ and $\mathcal{E}_\beta = \{(e_1, e_2) \in \mathcal{E} \mid e_1 \in Z_\beta, e_2 \in Z_\beta\}$. We have the following:

- $|C_I(\beta)| \leq |E| - |Z_\beta| + |\mathcal{E}_\beta| + (l + 1)[x_s \neq x_t]$ (It is not equality since pairs in $\mathcal{E}_\beta$ may not satisfy the corresponding 4AE clauses). Since we further have $|E| = |\mathcal{P}| < |C_I(\beta)|$, we have $|\mathcal{E}_\beta| + (l + 1)[x_s \neq x_t] > |Z_\beta|$.

- $0 \leq |\mathcal{E}_\beta| \leq \lfloor \frac{|Z_\beta|}{2} \rfloor$, since $\mathcal{E}$ is a partition of E. Hence, $x_s \neq x_t$ must hold for such β. Further, we have $l + 1 > |Z_\beta| - |\mathcal{E}_\beta| \geq \lceil \frac{|Z_\beta|}{2} \rceil$, so $|Z_\beta| \leq 2l$.
- $x_s \neq x_t \Rightarrow |Z_\beta| \geq 2l$, since Z_β corresponds to an st-cut and $\mathcal{F}$ implies that the minimum st-cut have size $2l$. Therefore, β must satisfy $|Z_\beta| = 2l, |\mathcal{E}_\beta| = l$ and $x_s \neq x_t$, which corresponds to a solution for the Paired Minimum st-Cut instance. Further, in this instance, the tails of the $2l$ edges in Z_β must belong to s-side of Z_β, so that all the clauses corresponding to $\mathcal{E}_\beta$ are satisfied, and we have $|C_I(\beta)| - |\mathcal{P}| = 1$.

So any assignment β with $|C_I(\beta)| > |\mathcal{P}|$, if it exists, corresponds to a feasible solution to the given Paired Minimum st-Cut instance, and is a good assignment for the constructed Improve-MaxCSP($\cup_{r' \leq 4} r'$AE) instance. This shows the validity (satisfying the promise) of the Improve-MaxCSP($\cup_{r' \leq 4} r'$AE) instance.

Finally, if the given Paired Minimum st-Cut instance does not have a st-cut Z touching l pairs, following the previous argument, we cannot find β with $|C_I(\beta)| > |\mathcal{P}|$. Notice that in this case, if we set all variables to zero, the assignment exactly satisfies $\mathcal{P}$, which is a good solution now, so the instance is still valid.

In summary, by distinguishing the number of satisfied clauses in the good assignment and $|\mathcal{P}|$, we can determine whether an st-cut Z in the Paired Minimum st-Cut instance satisfying the condition exists. If it exists, we can turn the good assignment of the ImproveMaxCSP($\cup_{r' \leq 4} r'$AE) instance to an st-cut satisfying the conditions for the Paired Minimum st-Cut problem. □

To extend the proof in Theorem 5.4 to 3AE, we need to mimic the 4AE clause $x_{u_1} = \overline{x_{v_1}} = x_{u_2} = \overline{x_{v_2}}$ corresponding to edge pairs using 3AE clauses. The most straightforward way with some clause duplication suffices to prove the main hardness result.

Theorem 5.5. ImproveMaxCSP($\cup_{r' \leq 3} r'$AE) *is W[1]-hard.*

Proof. We give a reduction from Paired Minimum st-Cut, whose hardness is by Lemma 5.3, to ImproveMaxCSP($\cup_{r' \leq 3} r'$AE). Given an Paired Minimum st-Cut instance $(G, s, t, l, \mathcal{E}, \mathcal{F})$, we construct a ImproveMaxCSP($\cup_{r' \leq 3} r'$AE) instance $(I, k, \mathcal{P})$ in the following way:

- For each vertex $u \in V_G$ introduce a variable x_u in V_I.
- For each edge $(u, v) \in E$ introduce 5 copies of the 2AE clause $x_u = x_v$ in both $\mathcal{C}_I$ and $\mathcal{P}$. In the rest of this proof, we refer to such clauses as *edge clauses*.
- For each pair $((u_1, v_1), (u_2, v_2)) \in \mathcal{E}$ introduce 2 copies of each of the following 4 3AE clauses in $\mathcal{C}_I$: $x_{u_1} = x_{u_2} = \overline{x_{v_2}}, \overline{x_{v_1}} = x_{u_2} = \overline{x_{v_2}}, x_{u_1} = \overline{x_{v_1}} = x_{u_2}$ and $x_{u_1} = \overline{x_{v_1}} = \overline{x_{v_2}}$. These are all four 3AE clauses generated from removing one variable in the 4AE clause $x_{u_1} = x_{u_2} = \overline{x_{v_1}} = \overline{x_{v_2}}$.
- Further introduce $(2l + 1)$ copies of the 2AE clause $x_s \neq x_t$ to $\mathcal{C}_I$. Set $k = 20l + 1$.

Now we prove the equivalence of the instances. Suppose the given Paired Minimum st-Cut instance has an st-cut Z touching l pairs, we construct exactly the same assignment α for the ImproveMaxCSP($\cup_{r' \leq 3} r'$AE) instance as we do in Theorem 5.4. α satisfies the following clauses:

- All edge clauses except for $10l$ clauses corresponding to edges in Z;
- $8l$ 3AE clauses corresponding to the l pairs in Z;
- $(2l + 1)$ copies of $x_s \neq x_t$ clause.

So $|C_I(\alpha)\Delta\mathcal{P}| \leq 20l+1$ and $|C_I(\alpha)| - |\mathcal{P}| = 1$. Now we only need to check α is good.

Consider any assignment $\beta : V_I \to \{0,1\}$ with $|C_I(\beta)| > |\mathcal{P}|$. Define $Z_\beta = \{(u,v) \in E \mid x_u \neq x_v\}, \mathcal{E}_\beta = \{(e_1,e_2) \in \mathcal{E} \mid e_1 \in Z_\beta, e_2 \in Z_\beta\}$ and $\mathcal{E}'_\beta = \{(e_1,e_2) \in \mathcal{E} \mid [e_1 \in Z_\beta] + [e_2 \in Z_\beta] = 1\}$. We have the following:

- $|C_I(\beta)| \leq 5|E| - 5|Z_\beta| + 8|\mathcal{E}_\beta| + 2|\mathcal{E}'_\beta| + (2l+1)[x_s \neq x_t]$, since one can easily check that exactly one of the 4 3AE clauses are satisfied for all pairs in $\mathcal{E}'_\beta$, and pairs in $\mathcal{E}_\beta$ can satisfy at most 4 3AE clauses. Since we further have $5|E| = |\mathcal{P}| < |C_I(\beta)|$, β satisfies $8|\mathcal{E}_\beta| + 2|\mathcal{E}'_\beta| + (2l+1)[x_s \neq x_t] > 5|Z_\beta|$.
- $|Z_\beta| = 2|\mathcal{E}_\beta| + |\mathcal{E}'_\beta|$, since $\mathcal{E}$ is a partition of E. Hence, $x_s \neq x_t$ must hold for such β. Further, we have $2l+1 > 2|\mathcal{E}_\beta| + 3|\mathcal{E}'_\beta|$, so $|Z_\beta| = 2|\mathcal{E}_\beta| + |\mathcal{E}'_\beta| \leq 2|\mathcal{E}_\beta| + 3|\mathcal{E}'_\beta| < 2l+1$.
- $x_s \neq x_t \Rightarrow |Z_\beta| \geq 2l$, since Z_β corresponds to an st-cut and $\mathcal{F}$ implies that the minimum st-cut have size $2l$. Therefore, $|Z_\beta| = 2l$ and $|\mathcal{E}'_\beta| = 0$, which means $|\mathcal{E}_\beta| = l$. Along with $x_s \neq x_t$, the assignment corresponds to a feasible solution for the Paired Minimum st-Cut instance. Further, in such instance, the tails of the $2l$ edges in Z_β must belong to s-side of Z_β, so all $8l$ clauses corresponding to $\mathcal{E}_\beta$ are satisfied, and we have $|C_I(\beta)| - |\mathcal{P}| = 1$.

So any assignment β with $|C_I(\beta)| > |\mathcal{P}|$, if it exists, corresponds to a feasible solution to the given Paired Minimum st-Cut instance, and is a good assignment for the constructed ImproveMaxCSP($\cup_{r' \leq 3} r'$AE) instance. This shows the validity (satisfying the promise) of the ImproveMaxCSP($\cup_{r' \leq 3} r'$AE) instance.

Finally, if the given Paired Minimum st-Cut instance does not have a st-cut Z touching l pairs, following the previous argument, we cannot find β with $|C_I(\beta)| > |\mathcal{P}|$. Notice that in this case, if we set all variables to zero, the assignment exactly satisfies $\mathcal{P}$, which is a good solution now, so the ImproveMaxCSP($\cup_{r' \leq 3} r'$AE) is still valid.

In summary, by distinguishing the number of satisfied clauses in the good assignment and $|\mathcal{P}|$, we can determine whether an st-cut Z in the Paired Minimum st-Cut instance satisfying the condition exists. If it exists, we can turn the good assignment of the ImproveMaxCSP($\cup_{r' \leq 3} r'$AE) instance to an st-cut satisfying the conditions for the Paired Minimum st-Cut problem. □

Proof of Theorem 2.7. From Theorem 5.5, ImproveMaxCSP($\cup_{r' \leq 3} r'$AE) is W[1]-hard. Along with Claim 5.2, ImproveMaxCSP(3AE) is W[1]-hard, and the statement follows from Lemma 5.1. □

6 Hardness Proof for $\leq$ 1-out-of-rSAT

In this section we prove the W[1]-hardness of ImproveMaxCSP($\leq$ 1-out-of-rSAT) for all $r \geq 2$. We first prove the case for $r = 2$, which is ImproveMaxCSP(2SAT), from the following problem called Multicolored Independent Set.

Problem: Multicolored Independent Set.
Input: An undirected graph $G = (V, E)$, an integer $l \in \mathbb{N}^+$, and a partition $(V_1, \cdots, V_l)$ of V into l non-empty sets.
Parameter: l.
Output: An independent set of G that contains exactly one vertex from each V_i if exists, otherwise report that such an independent set does not exist.

The reader can find the W[1]-hardness of Multicolored Independent Set in Corollary 13.8 of [CFK+15].

Lemma 6.1 ([CFK+15])**.** Multicolored Independent Set *is W[1]-hard.*

Theorem 6.2. ImproveMaxCSP*(2SAT) is W[1]-hard.*

Proof. We give a reduction from Multicolored Independent Set, whose hardness is given by Lemma 6.1, to ImproveMaxCSP(2SAT). Suppose that $G(V_1, \cdots, V_l; E)$ is the given Multicolored Independent Set instance. We assume there is no isolated vertex in G, otherwise we can simply include them in the independent set. We construct the following ImproveMaxCSP(2SAT) instance $(I, k, \mathcal{P})$:

For each vertex $u \in V$, introduce a variable x_u in I and a clause $\overline{x_u}$ in $\mathcal{C}$. For each edge $(u, v) \in E$, add 2 copies of the clause $x_u \vee x_v$ in $\mathcal{C}$ and $\mathcal{P}$. For each $i \in [l]$ and $u \neq v \in V_i$, add 2 copies of the clause $x_u \vee x_v$ in $\mathcal{C}$ and $\mathcal{P}$. Set k to be l.

We first show that, any optimal assignment satisfies $\mathcal{P}$. Assume towards contradiction that some α with the smallest cost have $x_u = x_v = 0$ for some clause $x_u \vee x_v$ in $\mathcal{P}$. By flipping $\alpha(x_u)$, at least two clauses change from unsatisfied to satisfied (since we introduce 2 copies for each binary clause, and $x_u \vee x_v$ becomes satisfied), and one unary clause changes from satisfied to unsatisfied. So the new assignment would satisfy strictly more clauses than α, contradicting with the selection of α.

Define $V_\alpha = \{u \mid x_u = 0\}$. Since the optimal α satisfies $\mathcal{P}$, from the construction of $\mathcal{P}$, V_α is a multicolored independent set, whose size is bounded by l. Since $|\mathcal{C}_I(\alpha) \Delta \mathcal{P}| = |\mathcal{C}_I(\alpha) \backslash \mathcal{P}| = |V_\alpha|$, the ImproveMaxCSP(2SAT) instance satisfies the promise, and the maximum multicolored independent set in G is exactly V_α for the optimal assignment α in the ImproveMaxCSP(2SAT) instance. This finishes the proof. □

Finally, by introducing dummy variables, one can easily reduce 2SAT to $\leq$ 1-out-of-rSAT, showing the main hardness result in this section.

Proof of Theorem 2.8. We give a reduction from ImproveMaxCSP(2SAT), whose hardness was proved in Theorem 6.2, to ImproveMaxCSP($\leq$ 1-out-of-rSAT) for $r \geq 3$. Given an instance $(I, k, \mathcal{P})$ of ImproveMaxCSP(2SAT), we would construct the instance $(I', k, \mathcal{P}')$ as follows:

- $V' = V \cup \{u_i \mid 1 \leq i \leq r-2\}$.
- For each binary clause $\mathcal{C} = (v_a \oplus p_{v_a}) \vee (v_b \oplus p_{v_b}) \in \mathcal{C}$ for $v_a \neq v_b \in V$ and $p_{v_a}, p_{v_b} \in \{0, 1\}$, introduce a clause $\wedge_{x \neq y \in \{u_i | 1 \leq i \leq r-2\} \cup \{v_a, v_b\}} (x \oplus p_x) \vee (y \oplus p_y)$ in $\mathcal{C}'$ (notice this clause is exactly $\leq$ 1-out-of-rSAT clause), where $p_{u_i} = 0$ for all $i \in [r-2]$. If $C \in \mathcal{P}$, add this clause to $\mathcal{P}'$. For unary clauses process similarly.

Since all u_i's appear positively in I', any good solution for $(I', k, \mathcal{P}')$ always sets u_i to 1. By assigning u_i's to 1 in instance $(I', k, \mathcal{P}')$ (see Definition 3.6), we get exactly the instance $(I, k, \mathcal{P})$. This proves the equivalence of the two instances. □

7 Hardness from MinCSPs

In this section, we prove Theorem 2.3.

7.1 Structural Characterization of Boolean Constraint Languages

We give the following definitions characterizing boolean constraint languages to introduce Lemma 7.8. All of these definitions are as in [KKPW23].

A boolean boolean relation R is *bijunctive* if it is expressible as a conjunction of 1- and 2-clauses (allowing negations), and R is *IHS-B-* (respectively *IHS-B+*) if it is expressible as a conjunction of negative clauses ($\neg x_1 \vee \ldots \vee \neg x_r$), positive 1-clauses ($x$), and implications ($x \to y$, negation not allowed) (respectively positive clauses, negative 1-clauses, and implications). Here, IHS-B is an abbreviation for *implicative hitting set, bounded*. A boolean constraint language Γ is bijunctive if every boolean relation $R \in \Gamma$ is bijunctive, and is IHS-B+ (respectively IHS-B-) if every relation in Γ is IHS-B+ (respectively IHS-B-). Finally, Γ is IHS-B if it is either IHS-B+ or IHS-B-. (Note that this is distinct from every boolean relation $R \in \Gamma$ being either IHS-B+ or IHS-B-.)

We say that an undirected graph is $2K_2$*-free* if it does not contain $2K_2$ as an induced subgraph. An directed graph is $2K_2$*-free* if its underlying undirected graph is $2K_2$-free.

Definition 7.1 (Gaifman Graph)**.** *The* Gaifman graph *of a boolean relation R of arity r is the undirected graph G_R with vertex set $V(G_R) = [r]$ and an edge (i, j) for all $1 \leq i < j \leq r$ for which there exist $a_i, a_j \in \{0, 1\}$ such that R contains no tuple $(t_1, \ldots, t_r)$ with $t_i = a_i$ and $t_j = a_j$. (In other words, there is an edge (i, j) in G_R if and only if constraint $R(x_1, \cdots, x_r)$ has no satisfying assignment α with $\alpha(x_i) = a_i$ and $\alpha(x_j) = a_j$ for some $a_i, a_j \in \{0, 1\}$.)*

Definition 7.2 (Arrow Graph)**.** *The* arrow graph *of a boolean boolean relation R of arity r is the directed graph H_R with vertex set $V(H_R) = [r]$ and an edge $i \to j$ for all $1 \leq i, j \leq r$ with $i \neq j$ such that R contains no tuple $(t_1, \ldots, t_r)$ with $t_i = 1$ and $t_j = 0$ but does contain two tuples t' and t'' with $t'_i = t'_j = 0$ and $t''_i = t''_j = 1$. (In other words, there is an edge (i, j) in H_R if and only if the constraint $R(x_1, \cdots, x_r)$ has no satisfying assignment α with $\alpha(x_i) = 1$ and $\alpha(x_j) = 0$ but has two satisfying assignments α_1 and α_2 with $\alpha_1(x_i) = \alpha_1(x_j) = 0$ and $\alpha_2(x_i) = \alpha_2(x_j) = 1$.)*

The following claims are straightforward from the definition and the symmetry of $\mathrm{SymRel}(r, S)$ relations.

Claim 7.3. *If a boolean relation R of arity r is bijunctive, then $R \oplus b$ is bijunctive for any $b \in \mathbb{F}_2^r$.*

Claim 7.4. *For any boolean relation R of arity r and $b \in \mathbb{F}_2^r$, $G_R \cong G_{R \oplus b}$ and $H_R \cong H_{R \oplus b}$, where $\cong$ stands for graph isomorphism relation.*

Claim 7.5. *For any r and $S \subseteq \{0, 1, \cdots, r\}$, $G_{\mathrm{SymRel}(r,S)}$ and the underlying undirected graph of $H_{\mathrm{SymRel}(r,S)}$ is either the complete graph or the empty graph.*

Combining Claim 7.4 and Claim 7.5, we immediately get the following corollary.

Corollary 7.6. *For any integer $r \in \mathbb{N}^+$, $S \subseteq \{0, 1, \cdots, r\}$ and $b \in \mathbb{F}_2^r$, $G_{\mathrm{SymRel}(r,S) \oplus b}$ and the underlying undirected graph of $H_{\mathrm{SymRel}(r,S) \oplus b}$ is either the complete graph or the empty graph.*

7.2 Main Theorem on MinCSP Hardness

We start with a simple claim that allows us to use hardness results of MinCSP to prove most hardness results for ImproveMaxCSP.

Claim 7.7. *If* MinCSP(Γ) *is W[1]-hard, then* ImproveMaxCSP(Γ) *is W[1]-hard.*

Proof. We present a reduction from MinCSP(Γ) to ImproveMaxCSP(Γ). For a given MinCSP(Γ) instance (I, k), we simply define $\mathcal{P} := \mathcal{C}_I$, and this would result in a ImproveMaxCSP(Γ) instance $(I, k, \mathcal{P})$. Notice that the instance $(I, k, \mathcal{P})$ may not satisfy the promise, and remind that we specify in Remark 2.4 the definition of the ImproveMaxCSP problem that some assignment should be provided for such instances.

Since $cost(\alpha) = |\mathcal{C}_I(\alpha)\Delta\mathcal{C}_I| = |\mathcal{C}_I(\alpha)\Delta\mathcal{P}|$, when (I, k) has an assignment of cost $\leq k$, $(I, k, \mathcal{P})$ satisfies the promise and solving $(I, k, \mathcal{P})$ would result in an optimal assignment. Otherwise, solving $(I, k, \mathcal{P})$ would result in an arbitrary assignment whose cost is larger than k, and we can conclude by checking its cost that I has no assignment of cost $\leq k$.

□

According to [KKPW23], the complexity of MinCSP for boolean constraint languages can be fully characterized.

Lemma 7.8 ([KKPW23])**.** *For a finite boolean constraint language* Γ*,* MinCSP(Γ) *is FPT if it satisfies one of the following conditions, and is W[1]-hard if none of the following holds.*

- *For all* $R \in \Gamma$*,* R *is bijunctive and* G_R *is* $2K_2$*-free.*
- *For all* $R \in \Gamma$*,* R *is IHS-B and* H_R *is* $2K_2$*-free.*

7.3 Case Distinction for Our Setting

Since complete and empty graphs are all $2K_2$-free, from Corollary 7.6, $2K_2$-freeness is always satisfied for SymLang-N(r, S). So we only need to check the bijunctive and IHS-B properties for them. Recall two constraint languages are equivalent if they contain the same set of relations.

Theorem 7.9. *When* $\Gamma =$ SymLang-N(r, S) *is nontrivial and IHS-B,* Γ *is equivalent to* rAND*.*

Proof. We show here proof for IHS-B- case, IHS-B+ case is similar. Since Γ is nontrivial and allows all negation patterns, there exists a relation $R \in \Gamma$ with IHS-B- definition ψ, say with scope $x_1, \cdots, x_r$, rejecting assignment 0^r. Since 0^r satisfies all negative clauses and all implications, there must be a positive 1-clause x_i in ψ for some variable x_i. Therefore, all elements $\alpha \in R$ have $\alpha_i = 1$.

Now choose any symmetric relation $R_0 \in \Gamma$. Such R_0 exists from the definition of Γ. Since $R_0 = R \oplus b$ for some $b \in \mathbb{F}_2^r$, all elements $\alpha_0 \in R_0$ have the same value in the i-th index. Without loss of generality suppose $\alpha_{0,i} = 0$. From the symmetry of R_0, we have $\alpha_{0,j} = 0$ for any $\alpha_0 \in R_0$ and $j \in [r]$. Therefore, $R_0 = \{0^r\} =$ SymRel$(r, \{0\})$ and Γ is rAND. □

Theorem 7.10. *When* $\Gamma =$ SymLang-N(r, S) *is nontrivial and bijunctive,* Γ *is either* $\leq$ 1-out-of-rSAT*,* rAE *or* rAND*.*

Proof. When $r = 1$, Γ is clearly bijunctive, and the only nontrivial one is 1AND. So in the following we consider $r \geq 2$.

Choose any symmetric relation $R \in \Gamma$. Such R exists from the definition of Γ. For a term ψ with scope X, let us say that R implies ψ if all elements in R satisfies ψ. Define $\pi_{ij}(R) = \pi_{\{i,j\}}(R) = \{\alpha_{\{i,j\}} \mid \alpha \in R\}$ for $i, j \in [r]$. By symmetry, $\pi_{ij}(R)$ are all the same for any $i \neq j$. Further, since R is bijunctive, $\wedge_{ij}\pi_{ij}(R)$ must defines R. So R is exactly determined by $\pi_{12}(R)$.

We have $\pi_{12}(R) = \pi_{21}(R)$, so $\pi_{12}(R)$ is symmetric. Write $\pi_{12}(R) =$ SymRel$(2, S')$. Let us enumerate the set of all possible S':

- $\{0, 1, 2\}$: $R = \{0, 1\}^r$ which is trivial.

- $\{0,1\}$: $R = \text{SymRel}(r, \{0,1\})$ whose corresponding language is $\leq$ 1-out-of-rSAT.
- $\{1,2\}$: $R = \text{SymRel}(r, \{r-1, r\})$ whose corresponding language is $\leq$ 1-out-of-rSAT.
- $\{0,2\}$: $R = \text{SymRel}(r, \{0, r\})$ whose corresponding language is rAE.
- $\{0\}$: $R = \text{SymRel}(r, \{0\})$ whose corresponding language is rAND.
- $\{1\}$: No such R exists for $r \geq 3$. For $r = 2, R = \text{SymRel}(2, \{1\})$ corresponds to 2AE.
- $\{2\}$: $R = \text{SymRel}(r, \{r\})$ whose corresponding language is rAND.
- $\varnothing$: $R = \varnothing$ which is trivial.

It is easy to check $\leq$ 1-out-of-rSAT, rAE and rAND are bijunctive. □

Now we prove Theorem 2.3.

Proof of Theorem 2.3. For nontrivial $\Gamma = \text{SymLang-N}(r, S)$ for some $r \in \mathbb{N}^+$ and $S \subseteq \{0, 1, \cdots, r\}$ that is not equivalent to rAND, rAE or $\leq$ 1-out-of-rSAT, from Theorem 7.9, Γ is not IHS-B; from Theorem 7.10, Γ is not bijunctive. So from Lemma 7.8, MinCSP(Γ) is W[1]-hard. Finally, according to Claim 7.7, ImproveMaxCSP(Γ) is W[1]-hard. □

References

[ALS25] Aditya Anand, Euiwoong Lee, and Amatya Sharma. Min-CSPs on complete instances. In *Proceedings of the 2025 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2025. 5

[BDT11] Nicolas Bousquet, Jean Daligault, and Stéphan Thomassé. Multicut is fpt. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 459–468, 2011. 5

[BEM16] Edouard Bonnet, László Egri, and Dániel Marx. Fixed-parameter approximability of boolean mincsps. In *24th European Symposium on Algorithms (ESA 2016)*, page 246, 2016. 5

[CCH+16] Rajesh Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, Marcin Pilipczuk, and Michał Pilipczuk. Designing fpt algorithms for cut problems using randomized contractions. *SIAM Journal on Computing*, 45(4):1171–1229, 2016. 8, 14

[CFK+15] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 5. Springer, 2015. 22

[CLP+14] Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. Minimum bisection is fixed parameter tractable. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 323–332, 2014. 5, 8

[CPPW13] Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. On multiway cut parameterized above lower bounds. *ACM Transactions on Computation Theory (TOCT)*, 5(1):1–11, 2013. 5

[DJO+23] Konrad K Dabrowski, Peter Jonsson, Sebastian Ordyniak, George Osipov, and Magnus Wahlström. Almost consistent systems of linear equations. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3179–3217. SIAM, 2023. 14

[FFL+12] Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Frances Rosamond, Saket Saurabh, and Yngve Villanger. Local search: Is brute-force avoidable? *Journal of Computer and System Sciences*, 78(3):707–719, 2012. In Commemoration of Amir Pnueli. 4

[GKO+12] Serge Gaspers, Eun Jung Kim, Sebastian Ordyniak, Saket Saurabh, and Stefan Szeider. Don't be strict in local search! In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 486–492, 2012. 4

[KKPW21] Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. Solving hard cut problems via flow-augmentation. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 149–168. SIAM, 2021. 5

[KKPW22] Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. Directed flow-augmentation. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 938–947, 2022. 5

[KKPW23] Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. Flow-augmentation iii: Complexity dichotomy for boolean csps parameterized by the number of unsatisfied constraints. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3218–3228. SIAM, 2023. 5, 7, 19, 23, 24

[KM12] Andrei Krokhin and Dániel Marx. On the hardness of losing weight. *ACM Transactions on Algorithms (TALG)*, 8(2):1–18, 2012. 4, 5

[KT11] Ken-ichi Kawarabayashi and Mikkel Thorup. The minimum k-way cut of bounded size is fixed-parameter tractable. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 160–169. IEEE, 2011. 5, 8, 14

[LMFB24] Tommaso Lanciano, Atsushi Miyauchi, Adriano Fazzone, and Francesco Bonchi. A survey on the densest subgraph problem and its variants. *ACM Computing Surveys*, 56(8):1–40, 2024. 9

[LSS22] Daniel Lokshtanov, Saket Saurabh, and Vaishali Surianarayanan. A parameterized approximation scheme for min k-cut. *SIAM Journal on Computing*, (0):FOCS20–205, 2022. 8

[Mar05] Dániel Marx. Parameterized complexity of constraint satisfaction problems. *computational complexity*, 14:153–183, 2005. 4

[MR09] Dániel Marx and Igor Razgon. Constant ratio fixed-parameter approximation of the edge multicut problem. *Information Processing Letters*, 109(20):1161–1166, 2009. 19

[MR11] Dániel Marx and Igor Razgon. Fixed-parameter tractability of multicut parameterized by the size of the cutset. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 469–478, 2011. 5

[MS11] Dániel Marx and Ildikó Schlotter. Stable assignment with couples: Parameterized complexity and local search. *Discrete Optimization*, 8(1):25–40, 2011. Parameterized Complexity of Discrete Optimization. 4

[NRRS12] NS Narayanaswamy, Venkatesh Raman, MS Ramanujan, and Saket Saurabh. Lp can be a cure for parameterized problems. In *29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2012. 5

[NSS95] Moni Naor, Leonard J Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 182–191. IEEE, 1995. 9

[RO08] Igor Razgon and Barry O'Sullivan. Almost 2-sat is fixed-parameter tractable. In *Automata, Languages and Programming: 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I 35*, pages 551–562. Springer, 2008. 5

[RRS11] Venkatesh Raman, MS Ramanujan, and Saket Saurabh. Paths, flowers and vertex cover. In *Algorithms–ESA 2011: 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings 19*, pages 382–393. Springer, 2011. 5

[RS95] Neil Robertson and Paul D Seymour. Graph minors. xiii. the disjoint paths problem. *Journal of combinatorial theory, Series B*, 63(1):65–110, 1995. 5

[Sze11] Stefan Szeider. The parameterized complexity of k-flip local search for sat and max sat. *Discrete Optimization*, 8(1):139–145, 2011. 4