Approximating Max-Cut under Graph-MSO Constraints

Martin Koutecký[a,1], Jon Lee[b,2], Viswanath Nagarajan[b,3], Xiangkun Shen[b]

[a] *Technion – Israel Institute of Technology, Faculty of IE&M, Haifa, Israel*
[b] *IOE Dept., University of Michigan, Ann Arbor, MI 48109, USA*

Abstract

We consider the max-cut and max-k-cut problems under graph-based constraints. Our approach can handle any constraint specified using monadic second-order (MSO) logic on graphs of constant treewidth. We give a $\frac{1}{2}$-approximation algorithm for this class of problems.

Keywords: max cut, approximation algorithm, monadic second-order logic, treewidth, dynamic program
2000 MSC: 68W25, 68W05, 68R10

1. Introduction

This paper considers the classic max-cut problem under a class of graph-based constraints. The max-cut problem is a fundamental combinatorial-optimization problem which has many practical applications (see [1, 2, 3, 4]) as well as strong theoretical results (see [5, 6]). There have also been a number of papers on designing approximation algorithms for *constrained* max-cut problems (see [7, 8, 9, 10, 11]).

In this paper, we are interested in constraints that are specified by an auxiliary constraint graph. Our main result is a $\frac{1}{2}$-approximation algorithm for max-cut under any graph constraint that can be expressed in monadic second order logic (MSO) (see [12]). This is closely related to a recent result by a subset of the authors; see [13]. The contribution of this paper is in generalizing the class of constraints handled in [13], making the algorithm design more systematic, and extending the result to the max-k-cut setting with k instead of just 2 parts.

In particular, [13] gave a $\frac{1}{2}$-approximation algorithm for max-cut under any graph constraint $\mathcal{S}_G$ that has a specific type of dynamic program for optimizing linear objectives. In order to apply this result, one also has to design such a dynamic program separately for each constraint $\mathcal{S}_G$, which requires additional constraint-specific work. Indeed, [13] also gave constraint-specific dynamic programs for various graph constraints such as independent set, vertex cover, dominating set and connectivity, all on bounded-treewidth graphs.

In this paper, we bypass the need for constraint-specific dynamic programs by utilizing the language and results from monadic second-order logic. We show that any MSO constraint on a bounded-treewidth graph (defined formally in §2) admits a dynamic program that satisfies the assumptions needed in [13]. Therefore, we immediately obtain $\frac{1}{2}$-approximation algorithms for max-cut under any MSO graph constraint. We note that MSO constraints capture all the specific graph constraints in [13], and much more.

We also extend these results to the setting of max-k-cut, where we seek to partition the vertices into k parts $\{U_i\}_{i=1}^k$ so as to maximize the weight of edges crossing the partition. In the constrained version, we additionally require each part U_i to satisfy some MSO graph property. We obtain a $\frac{1}{2}$-approximation algorithm even in this setting (k is fixed). This result is a significant generalization over [13] even for $k = 2$, which corresponds to the usual max-cut problem: we now handle constraints on both sides of the cut.

2. Preliminaries

A k-partition of vertex set V is a function $h : V \to [k]$, where the k parts are $U_\alpha = \{v \in V : h(v) = \alpha\}$ for $\alpha \in [k]$. Note that $\cup_{\alpha=1}^k U_\alpha = V$ and $U_1, \cdots, U_k$ are disjoint. When we want to refer to the k parts directly, we also use $\{U_\alpha\}_{\alpha=1}^k$ to denote the k-partition.

Definition 1 (GCMC). The input to the *graph-constrained max-cut* (GCMC) problem consists of (i) an n-vertex graph $G = (V, E)$ with a graph property which implicitly specifies a collection $\mathcal{S}_G$ of vertex k-partitions, and (ii) symmetric edge-weights $c : \binom{V}{2} \to$

Email addresses: koutecky@kam.mff.cuni.cz (Martin Koutecký), jonxlee@umich.edu (Jon Lee), viswa@umich.edu (Viswanath Nagarajan), xkshen@umich.edu (Xiangkun Shen)

[1] Supported by a Technion postdoc grant.
[2] Supported in part by ONR grant N00014-17-1-2296. Part of this work was done while J. Lee was visiting the Simons Institute for the Theory of Computing (which was partially supported by the DIMACS/Simons Collaboration on Bridging Continuous and Discrete Optimization through NSF grant CCF-1740425).
[3] Supported in part by NSF CAREER grant CCF-1750127.

$\mathbb{R}_+$. The GCMC problem is to find a k-partition in $\mathcal{S}_G$ with the maximum weight of crossing edges:

$$\max_{h \in \mathcal{S}_G} \sum_{\substack{\{u,v\} \in \binom{V}{2} \\ h(u) \neq h(v)}} c(u,v). \tag{1}$$

Tree Decomposition. Given an undirected graph $G = (V, E)$, a tree decomposition consists of a tree $\mathcal{T} = (I, F)$ and a collection of vertex subsets $\{X_i \subseteq V\}_{i \in I}$ such that:

- for each $v \in V$, the nodes $\{i \in I : v \in X_i\}$ are connected in $\mathcal{T}$, and
- for each edge $(u, v) \in E$, there is some node $i \in I$ with $u, v \in X_i$.

The width of such a tree decomposition is $\max_{i \in I}(|X_i| - 1)$, and the treewidth of G is the smallest width of any tree decomposition for G.

We work with "rooted" tree decompositions, also specifying a root node $r \in I$. The depth d of such a tree decomposition is the length of the longest root-leaf path in $\mathcal{T}$. The depth of any node $i \in I$ is the length of the $r - i$ path in $\mathcal{T}$. For any $i \in I$, the set V_i denotes all the vertices at or below node i, that is

$$V_i := \cup_{k \in \mathcal{T}_i} X_k,$$

where $\mathcal{T}_i = \{k \in I : k \text{ in subtree of } \mathcal{T} \text{ rooted at } i\}$.

The following result provides a convenient representation of $\mathcal{T}$.

Theorem 2.1 (Balanced Tree Decomposition; see [14])**.** *Let $G = (V, E)$ be a graph with tree decomposition $(\mathcal{T} = (I, F), \{X_i | i \in I\})$ of treewidth k. Then G has a rooted tree decomposition $(\mathcal{T}' = (I', F'), \{X'_i | i \in I'\})$ where $\mathcal{T}'$ is a binary tree of depth $2\lceil \log_{\frac{5}{4}}(2|V|) \rceil$ and treewidth at most $3k + 2$. Moreover, for all $i \in I$, there is an $i' \in I'$ such that $X_i \subseteq X'_{i'}$. The tree decomposition $\mathcal{T}'$ can be found in $O(|V|)$ time.*

Definition 2 (CSP instance). A Constraint Satisfaction Problem (CSP) instance $J = (N, C)$ consists of:

- a set N of *boolean variables*, and
- a set C of *constraints*, where each constraint $C_U \in C$ is a $|U|$-ary relation $C_U \subseteq \{0,1\}^U$ on some subset $U \subseteq N$.

For a vector $x \in \{0,1\}^N$ and a subset R of variables, we denote by $x|_R$ the *restriction of x to R*. A vector $z \in \{0,1\}^N$ *satisfies constraint* $C_U \in C$ if $z|_U \in C_U$. We say that $z \in \{0,1\}^N$ is *a feasible assignment* for the CSP instance J if z satisfies every constraint $C \in C$. Let $\text{Feas}(J)$ be the set of all feasible assignments of J. Finally, $\|C\| = \sum_{C_U \in C} |C_U|$ denotes the *length of C*.

Definition 3 (Constraint graph). The constraint graph of J, denoted $G(J)$, is defined as $G(J) = (N, F)$ where $F = \{\{u, v\} \mid \exists C_U \in C \text{ s.t. } \{u, v\} \subseteq U\}$.

Definition 4 (Treewidth of CSP). The treewidth $\text{tw}(J)$ of a CSP instance J is defined as the treewidth of its constraint graph $\text{tw}(G(J))$.

Definition 5 (CSP extension). Let $J = (N, C)$ be a CSP instance. We say that $J' = (N', C')$ with $N \subseteq N'$ is an extension of J if $\text{Feas}(J) = \{z|_N \mid z \in \text{Feas}(J')\}$.

Monadic Second Order Logic. We briefly introduce MSO over graphs. In *first-order logic* (FO) we have variables for individual vertices/edges (denoted $x, y, \ldots$), equality for variables, quantifiers $\forall, \exists$ ranging over variables, and the standard Boolean connectives $\neg, \wedge, \vee, \implies$. MSO is the extension of FO by quantification over sets (denoted $X, Y, \ldots$). Graph MSO has the binary relational symbol $\text{edge}(x, y)$ encoding edges, and traditionally comes in two flavours, MSO_1 and MSO_2, differing by the objects we are allowed to quantify over: in MSO_1 these are the vertices and vertex sets, while in MSO_2 we can additionally quantify over edges and edge sets. For example, 3-colorability can be expressed in MSO_1 as follows:

$$\begin{aligned} \exists X_1, X_2, X_3 \quad & [\, \forall x \, (x \in X_1 \vee x \in X_2 \vee x \in X_3) \\ & \wedge \bigwedge\nolimits_{i=1,2,3} \forall x, y \, (x \notin X_i \vee y \notin X_i \\ & \vee \neg \, \text{edge}(x, y)) \,] \end{aligned}$$

We remark that MSO_2 can express properties that are not MSO_1 definable. As an example, consider Hamiltonicity on graph $G = (V, E)$; an equivalent description of a Hamiltonian cycle is that it is a connected 2-factor of a graph:

$$\begin{aligned} \varphi_{\text{ham}} &\equiv \exists F \subseteq E : \varphi_{\text{2-factor}}(F) \wedge \varphi_{\text{connected}}(F) \\ \varphi_{\text{2-factor}}(F) &\equiv (\forall v \in V : \exists e, f \in F : (e \neq f) \\ &\quad \wedge (v \in e) \wedge (v \in f)) \wedge \neg(\exists v \in V : \\ &\quad \exists e, f, g \in F : (e \neq f \neq g) \wedge (v \in e) \\ &\quad \wedge (v \in f) \wedge (v \in g)) \\ \varphi_{\text{connected}}(F) &\equiv \neg[\exists U, W \subseteq V : (U \cap W = \emptyset) \\ &\quad \wedge (U \cup W = V) \wedge \neg(\exists \{u, v\} \in F : \\ &\quad u \in U \wedge v \in W)] \ . \end{aligned}$$

We use φ to denote an MSO formula and $G = (V, E)$ for the underlying graph. For a formula φ, we denote by $|\varphi|$ the *size* (number of symbols) of φ.

In order to express constraints on k-vertex-partitions via MSO, we use MSO formulas φ with k free variables $\{U_\alpha\}_{\alpha=1}^k$ where (i) the U_α are enforced to form a partition of the vertex-set V, and (ii) each U_α satisfies some individual MSO constraint φ_α. Because k is constant, the size of the resulting MSO formula is a constant as long as each of the MSO constraints φ_α has constant size.

Connecting CSP and MSO. Consider an MSO formula φ with k free variables on graph G (as above). For a vector $t \in \{0,1\}^{V \times [k]}$, we write $G, t \models \varphi$ if

and only if φ is satisfied by solution $U_\alpha = \{v \in V : t((v,\alpha)) = 1\}$ for $\alpha \in [k]$.

Definition 6 ($CSP_\varphi(G)$ instance). Let G be a graph and φ be an MSO_2-formula with k free variables. By $CSP_\varphi(G)$ we denote the CSP instance (N, C) with $N = \{t((v,\alpha)) \mid v \in V(G), \alpha \in [k]\}$ and with a single constraint $\{t \mid G, t \models \varphi\}$.

Observe that $\mathrm{Feas}(CSP_\varphi(G))$ corresponds to the set of feasible assignments of φ on G. Also, the treewidth of $CSP_\varphi(G)$ is $|V|k$ which is unbounded. The following result shows that there is an equivalent CSP extension that has constant treewidth.

Theorem 2.2 ([15, Theorem 25][4]). *Let $G = (V, E)$ be a graph with $\mathrm{tw}(G) = \tau$ and φ be an MSO_2-formula with k free variables. Then $CSP_\varphi(G)$ has a CSP extension J with $\mathrm{tw}(J) \le f(|\varphi|, \tau)$ and $\|C_J\| \le f(|\varphi|, \tau) \cdot |V|$.*

3. Dynamic Program for CSP

In this section we demonstrate that every CSP of bounded treewidth admits a dynamic program that satisfies the assumptions required in [13].

Consider a CSP instance $J = (V, C)$ with a constraint graph $G = (V, E)$ of bounded treewidth. Let $(\mathcal{T} = (I, F), \{X_i | i \in I\})$ denote a balanced tree decomposition of G (from Theorem 2.1). In what follows, we denote the vertex set $V = [n] = \{1, 2, \cdots, n\}$. Let λ be a symbol denoting an unassigned value. For any $W \subseteq V$, define the *set of configurations of* W as:

$$\begin{aligned}\mathcal{K}(W) =& \{(z_1, \ldots, z_n) \in \{0,1,\lambda\}^V \mid \\ & \forall C_U \in C : (U \subseteq W \implies z|_U \in C_U), \\ & \forall i \notin W : z_i = \lambda,\ \forall j \in W : z_j \in \{0,1\}\}\end{aligned}$$

Let $k \in \mathcal{K}(W)$ be a configuration and $v \in V$. Because k is a vector, $k(v)$ refers to the v-th element of k.

Definition 7 (State Operations). Let $U, W \subseteq V$. Let $k \in \mathcal{K}(U)$ and $p \in \mathcal{K}(W)$.

- Configurations k and p are said to be *consistent* if, for each $v \in V$, either $k(v) = p(v)$ or at least one of $k(v), p(v)$ is λ.
- If configurations k and p are consistent, define $[p \cup k](v) = \begin{cases} p(v), & \text{if } k(v) = \lambda; \\ k(v), & \text{otherwise.} \end{cases}$
- Define $[k \cap W](v) = \begin{cases} k(v), & \text{if } v \in W; \\ \lambda, & \text{otherwise.} \end{cases}$

We start by defining some useful parameters for the dynamic program.

[4] To be precise, [15, Theorem 25] speaks of MSO_1 over σ_2-structures, which is equivalent to MSO_2 over graphs; cf. the discussion in [15, Section 2.1].

Definition 8. For each node $i \in I$ with children nodes $\{j, j'\}$, we associate the following:

1. state space $\Sigma_i = \mathcal{K}(X_i)$.
2. for each $\sigma \in \Sigma_i$, there is a collection of partial solutions
$$\mathcal{H}_{i,\sigma} := \{k \in \mathcal{K}(V_i) \mid k \cap X_i = \sigma\}.$$
3. for each $\sigma \in \Sigma_i$, there is a collection of valid combinations of children states
$$\begin{aligned}\mathcal{F}_{i,\sigma} =& \{(\sigma_j, \sigma_{j'}) \in \Sigma_j \times \Sigma_{j'} \mid (\sigma_j \cap X_i) = \\ & (\sigma \cap X_j) \text{ and } (\sigma_{j'} \cap X_i) = (\sigma \cap X_{j'})\}.\end{aligned}$$

In words, (a) Σ_i is just the set of configurations for the vertices X_i in node i, (b) $\mathcal{H}_{i,\sigma}$ are those configurations for the vertices V_i (in the subtree rooted at i) that are consistent with σ, (c) $\mathcal{F}_{i,\sigma}$ are those pairs of states at the children $\{j, j'\}$ that agree with σ on the intersections $X_i \cap X_j$ and $X_i \cap X_{j'}$ respectively.

Theorem 3.1 (Dynamic Program for CSP). *Let $(\mathcal{T} = (I, F), \{X_i | i \in I\})$ be a tree decomposition of a CSP instance (V, C) of bounded treewidth. Then Σ_i, $\mathcal{F}_{i,\sigma}$ and $\mathcal{H}_{i,\sigma}$ from Definition 8 satisfy the conditions:*

1. (bounded state space) *Σ_i and $\mathcal{F}_{i,\sigma}$ are all bounded by constant, that is,* $\max_i |\Sigma_i| = O(1)$ *and* $\max_{i,\sigma} |\mathcal{F}_{i,\sigma}| = O(1)$.
2. (required state) *For each $i \in I$ and $\sigma \in \Sigma_i$, the intersection with X_i of every vector in $\mathcal{H}_{i,\sigma}$ is the same, in particular $h \cap X_i = \sigma$ for all $h \in \mathcal{H}_{i,\sigma}$.*
3. *By condition 2, for any leaf $\ell \in I$ and $\sigma \in \Sigma_\ell$, we have $\mathcal{H}_{\ell,\sigma} = \{\sigma\}$ or $\emptyset$.*
4. (subproblem) *For each non-leaf node $i \in I$ with children $\{j, j'\}$ and $\sigma \in \Sigma_i$,*
$$\begin{aligned}\mathcal{H}_{i,\sigma} =& \Big\{\sigma \cup h_j \cup h_{j'} \mid h_j \in \mathcal{H}_{j,w_j}, \\ & h_{j'} \in \mathcal{H}_{j',w_{j'}}, (w_j, w_{j'}) \in \mathcal{F}_{i,\sigma}\Big\}.\end{aligned}$$
5. (feasible subsets) *At the root node r, we have* $\mathrm{Feas}(V, C) = \bigcup_{\sigma \in \Sigma_r} \mathcal{H}_{r,\sigma}$.

Proof. Let $q = O(1)$ denote the treewidth of $\mathcal{T}$. We now prove each of the claimed properties.

Bounded state space. Because $|X_i| \le q + 1$, we have $|\Sigma_i| = |\mathcal{K}(X_i)| \le 3^{q+1} = O(1)$ and $|\mathcal{F}_{i,\sigma}| \le |\Sigma_j \times \Sigma_{j'}| \le (3^{q+1})^2 = O(1)$.

Required state. This holds immediately by definition of $\mathcal{H}_{i,\sigma}$ in Definition 8.

Subproblem. We first prove the "$\subseteq$" inclusion of the statement. Consider any $h \in \mathcal{H}_{i,\sigma} \subseteq \mathcal{K}(V_i)$. Let $h_j = h \cap V_j$, $w_j = h \cap X_j$ and analogously for j'. Observe that for $U \subset W \subseteq V$ we have that $k \in \mathcal{K}(W) \implies k \cap U \in \mathcal{K}(U)$. By this observation, $h_j \in \mathcal{K}(V_j)$. Moreover, $h_j \cap X_j = h \cap X_j = w_j$, which implies $h_j \in \mathcal{H}_{j,w_j}$. Again, the same applies for j' and we have

$h_{j'} \in \mathcal{H}_{j',w_{j'}}$. Finally, note that $w_j \cap X_i = h \cap X_j \cap X_i = (h \cap X_i) \cap X_j = \sigma \cap X_j$ and similarly $w_{j'} \cap X_i = \sigma \cap X_{j'}$. So we have $(w_j, w_{j'}) \in \mathcal{F}_{i,\sigma}$.

Now, we prove the "$\supseteq$" inclusion of the statement. Consider any two partial solutions $h_j \in \mathcal{H}_{j,w_j}$ and $h_{j'} \in \mathcal{H}_{j',w_{j'}}$ with $(w_j, w_{j'}) \in \mathcal{F}_{i,\sigma}$. Note that h_j and σ (similarly $h_{j'}$ and σ) are consistent by definition of $\mathcal{F}_{i,\sigma}$. We now claim that h_j and $h_{j'}$ are also consistent: take any $v \in V$ with both $h_{j'}(v), h_{j'}(v) \neq \lambda$, then we must have $v \in V_j \cap V_{j'} \subseteq X_i \cap X_j \cap X_{j'}$ as X_i is a vertex separator, and so $h_j(v) = \sigma(v) = h_{j'}(v)$ by definition of $\mathcal{F}_{i,\sigma}$. Because σ, h_j and $h_{j'}$ are mutually consistent, $h = \sigma \cup h_j \cup h_{j'}$ is well-defined. It is clear from the above arguments that $h \cap X_i = \sigma$. In order to show $h \in \mathcal{H}_{i,\sigma}$ we now only need $h \in \mathcal{K}(V_i)$, that is, h does not violate any constraint that is contained in V_i. For contradiction assume that that there is such a violated constraint C_S with $S \subseteq V_i$. Then S induces a clique in the constraint graph G and thus there must exist a node k among the descendants of i such that $S \subseteq V_k$. But k cannot be in the subtree rooted in j or j', because then C_S would have been violated already in h_j or $h_{j'}$, and also it cannot be that $i = k$, because then C_S would be violated in σ, a contradiction.

Feasible subsets. Clearly, the set $\mathrm{Feas}(V, C)$ of feasible CSP solutions is equal to $\mathcal{K}(V)$. Because $\mathcal{H}_{r,\sigma}$ is those $k \in \mathcal{K}(V)$ with $k \cap X_r = \sigma$, the claim follows. □

We note that Theorem 3.1 proves Assumption 1 in [13]. To clarify the comparison, Assumption 1 is:

Assumption 1 (Assumption 1 in [13])**.** *Let* $(\mathcal{T} = (I, F), \{X_i | i \in I\})$ *be any tree decomposition. Then there exist* Σ_i, $\mathcal{F}_{i,\sigma}$ *and* $\mathcal{H}_{i,\sigma}$ *(see Definition 8) that satisfy the following conditions:*

1. (bounded state space) Σ_i *and* $\mathcal{F}_{i,\sigma}$ *are all bounded by constant, that is,* $\max_i |\Sigma_i| = O(1)$ *and* $\max_{i,\sigma} |\mathcal{F}_{i,\sigma}| = O(1)$.
2. (required state) *For each* $i \in I$ *and* $\sigma \in \Sigma_i$*, the intersection with* X_i *of every set in* $\mathcal{H}_{i,\sigma}$ *is the same, denoted* $X_{i,\sigma}$*, that is* $S \cap X_i = X_{i,\sigma}$ *for all* $S \in \mathcal{H}_{i,\sigma}$.
3. *By condition 2, for any leaf* $\ell \in I$ *and* $\sigma \in \Sigma_\ell$*, we have* $\mathcal{H}_{\ell,\sigma} = \{X_{\ell,\sigma}\}$ *or* $\emptyset$.
4. (subproblem) *For each non-leaf node* $i \in I$ *with children* $\{j, j'\}$ *and* $\sigma \in \Sigma_i$*,*

$$\mathcal{H}_{i,\sigma} = \Big\{ X_{i,\sigma} \cup S_j \cup S_{j'} \ : \ S_j \in \mathcal{H}_{j,w_j}, \ S_{j'} \in \mathcal{H}_{j',w_{j'}}, \ (w_j, w_{j'}) \in \mathcal{F}_{i,\sigma} \Big\}.$$

5. (feasible subsets) *At the root node* r*, we have* $\mathcal{S}_G = \bigcup_{\sigma \in \Sigma_r} \mathcal{H}_{r,\sigma}$.

Assumption 1 is used in the main result of [13], which is restated below.

Theorem 3.2 (Theorem 4 in [13])**.** *Consider any instance of the* GCMC *problem on a bounded-treewidth graph* G*. If the graph constraint* $\mathcal{S}_G$ *satisfies Assumption 1 then we obtain a* $\frac{1}{2}$*-approximation algorithm.*

We will use this result in Section 4, but we will modify its proof slightly in Section 5 for max-k-cut.

4. The Max-Cut Setting

Here, we consider the GCMC problem when $k = 2$ and there is a constraint $\mathcal{S}_G$ for only one side of the cut. We show that the above dynamic-program structure can be combined with [13] to obtain a $\frac{1}{2}$-approximation algorithm.

Formally, there is an MSO formula φ with one free variable defined on graph $G = (V, E)$ of bounded treewidth. The feasible vertex subsets $\mathcal{S}_G$ are those $S \subseteq V$ that satisfy φ. There is also a symmetric weight function $c : \binom{V}{2} \to \mathbb{R}_+$. We are interested in the following problem (GCMC_I).

$$\max_{S \in \mathcal{S}_G} \sum_{u \in S, v \notin S} c(u, v). \tag{2}$$

We note that this is precisely the setting of [13].

Theorem 4.1. *There is a* $\frac{1}{2}$*-approximation algorithm for* GCMC_I *when the constraint* $\mathcal{S}_G$ *is given by any* MSO *formula on a bounded-treewidth graph.*

Proof. The proof uses Theorem 3.2 from [13] as a black-box. Note that the constraint $\mathcal{S}_G$ corresponds to feasible assignments to $CSP_\varphi(G)$ as in Definition 6. Consider the CSP extension ϑ obtained after applying Theorem 2.2 to $CSP_\varphi(G)$. Then ϑ has variables $V' \supseteq V$ and bounded treewidth. We obtain an extended weight function $c : \binom{V'}{2} \to \mathbb{R}_+$ from c by setting $c'(u, v) = c(u, v)$ if $u, v \in V$ and $c'(u, v) = 0$ otherwise. We now consider a new instance of GCMC_I on vertices V' and constraint ϑ. Due to the bounded-treewidth property of ϑ, we can apply Theorem 3.1 which proves that Assumption 1 is satisfied by the dynamic program in Definition 8. Combined with Theorem 3.2, we obtain the claimed result. □

5. The Max-k-Cut Setting

In this section, we generalize the setting to any constant k, i.e. problem (1). Recall the formal definition from §2. Here the graph property $\mathcal{S}_G$ is expressed as an MSO formula with k free variables on graph G. Our main result is the following:

Theorem 5.1. *There is a* $\frac{1}{2}$*-approximation algorithm for any* GCMC *instance with constant* k *when the constraint* $\mathcal{S}_G$ *is given by any* MSO *formula on a bounded-treewidth graph.*

Omitted proofs in this section are in Appendix A.

Remark 1. The complexity of Theorem 5.1 in terms of the treewidth τ, length $|\varphi|$ of φ, depth d of a tree decomposition of G, and maximum degree r of a tree

decomposition of G, is s^{dr}, where s is the number of states of the dynamic program, namely $f(|\varphi|, \tau)$ for f from Theorem 2.2. From the perspective of parameterized complexity [16] our algorithm is an *XP algorithm* parameterized by τ, i.e., it has runtime $n^{g(\tau)}$ for some computable function g.

Let $G = (V, E)$ be the input graph (assumed to have bounded treewidth) and φ be any MSO formula with k free variables. Recall the CSP instance $CSP_\varphi(G)$ on variables $\{y(v, \alpha) : v \in V, \alpha \in [k]\}$ from Definition 6. Feasible solutions to $CSP_\varphi(G)$ correspond to feasible k-partitions in $\mathcal{S}_G$. Now consider the CSP extension ϑ obtained after applying Theorem 2.2 to $CSP_\varphi(G)$. Note that ϑ is defined on variables $V' \supseteq \{(v, \alpha) : v \in V, \alpha \in [k]\}$ and has bounded treewidth. Let $\mathcal{T}$ denote the tree decomposition for ϑ. Below we utilize the dynamic program from Definition 8 applied to ϑ: recall the quantities Σ_i, $\mathcal{F}_{i,\sigma}$ etc. We will also refer to the variables in V' as vertices, especially when referring to the tree decomposition $\mathcal{T}$; note that these are different from the vertices V in the original graph G.

Claim 1. *Let $\{U_\alpha\}_{\alpha=1}^k$ be a k-partition satisfying $\mathcal{S}_G$. There is a collection of states $\{b[i] \in \Sigma_i\}_{i \in I}$ such that:*

- *for each node $i \in I$ with children j and j', $(b[j], b[j']) \in \mathcal{F}_{i,b[i]}$,*
- *for each leaf ℓ we have $\mathcal{H}_{\ell,b[\ell]} \neq \emptyset$, and*
- *$U_\alpha = \{v \in V : b_{\mathcal{T}}((v, \alpha)) = 1\}$ for all $\alpha \in [k]$, where $b_{\mathcal{T}} = \bigcup_{i \in I} b[i]$.*

Moreover, for any vertex $(v, \alpha) \in V'$, if $\overline{v\alpha} \in I$ denotes the highest node in $\mathcal{T}$ containing (v, α) then we have: $v \in U_\alpha$ if and only if $b[\overline{v\alpha}]((v, \alpha)) = 1$.

Proof. By definition of CSP ϑ, we know that it has some feasible solution $t \in \{0, 1\}^{V'}$ where $U_\alpha = \{v \in V : t((v, \alpha)) = 1\}$ for all $\alpha \in [k]$. Now, using Theorem 3.1(5) we have $t \in \bigcup_{\sigma \in \Sigma_r} \mathcal{H}_{r,\sigma}$. The rest of the proof is identical to Claim 1 in [13]. See the full proof in Appendix A. □

LP relaxation for Max-k-Cut. We start with some additional notation related to the tree decomposition $\mathcal{T}$ (from Theorem 2.1) and the dynamic program for CSP (from Theorem 3.1).

- For any node $i \in I$, T_i is the set consisting of (1) all nodes N on the $r-i$ path in $\mathcal{T}$, and (2) children of all nodes in $N \setminus \{i\}$.
- $\mathcal{P}$ is the collection of all node subsets J such that $J \subseteq T_{\ell_1} \cup T_{\ell_2}$ for some pair of leaf-nodes ℓ_1, ℓ_2.
- $s[i] \in \Sigma_i$ denotes a state at node i. Moreover, for any subset of nodes $N \subseteq I$, we use the shorthand $s[N] := \{s[k] : k \in N\}$.
- $a[i] \in \Sigma_i$ denotes a state at node i chosen by the algorithm. Similar to $s[N]$, for any subset $N \subseteq I$ of nodes, $a[N] := \{a[k] : k \in N\}$.
- $\overline{v\alpha} \in I$ denotes the highest tree-decomposition node containing vertex $(v, \alpha) \in V'$.

The LP that we use here is a generalization of that in [13]. The variables are $y(s[N])$ for all $\{s[k] \in \Sigma_k\}_{k \in N}$ and $N \in \mathcal{P}$. Variable $y(s[N])$ corresponds to the probability of the joint event that the solution (in $\mathcal{S}_G$) "induces" state $s[k]$ at each node $k \in N$. Variable $z_{uv\alpha}$ corresponds to the probability that edge $(u, v) \in E$ is cut by part α of the k-partition.

In constraint (6), we use j and j' to denote the two children of node $i \in I$. We note that constraints (4)-(8) which utilize the dynamic-program structure, are identical to the constraints (4)-(8) in the LP from [13]. This allows us to essentially reuse many of the claims proved in [13], which are stated below.

$$\text{maximize} \quad \frac{1}{2} \sum_{\{u,v\} \in \binom{V}{2}} c_{uv} \sum_{\alpha=1}^{k} z_{uv\alpha} \tag{LP}$$

$$z_{uv\alpha} = \sum_{\substack{s[\overline{u\alpha}] \in \Sigma_{\overline{u\alpha}},\, s[\overline{v\alpha}] \in \Sigma_{\overline{v\alpha}} \\ s[\overline{u\alpha}]((u,\alpha)) \neq s[\overline{v\alpha}]((v,\alpha))}} y(s[\{\overline{u\alpha}, \overline{v\alpha}\}]),$$

$$\forall \{u, v\} \in \binom{V}{2}, \forall \alpha \in [k]; \tag{3}$$

$$y(s[N]) = \sum_{s[i] \in \Sigma_i} y(s[N \cup \{i\}]),$$

$$\forall s[k] \in \Sigma_k,\ \forall k \in N,\ \forall N \in \mathcal{P},\ \forall i \notin N : N \cup \{i\} \in \mathcal{P}; \tag{4}$$

$$\sum_{s[r] \in \Sigma_r} y(s[r]) = 1; \tag{5}$$

$$y(s[\{i, j, j'\}]) = 0,$$
$$\forall i \in I,\ \forall s[i] \in \Sigma_i,\ \forall (s[j], s[j']) \notin \mathcal{F}_{i,s[i]}; \tag{6}$$

$$y(s[\ell]) = 0,$$
$$\forall \text{ leaf } \ell \in I,\ \forall s[\ell] \in \Sigma_\ell : \mathcal{H}_{\ell,s[\ell]} = \emptyset; \tag{7}$$

$$0 \leq y(s[N]) \leq 1,$$
$$\forall N \in \mathcal{P},\ \forall s[k] \in \Sigma_k \text{ for } k \in N. \tag{8}$$

Claim 2. *Let y be feasible to* (LP)*. For any node $i \in I$ with children j, j' and $s[k] \in \Sigma_k$ for all $k \in T_i$,*

$$y(s[T_i]) = \sum_{s[j] \in \Sigma_j} \sum_{s[j'] \in \Sigma_{j'}} y(s[T_i \cup \{j, j'\}]).$$

Proof. This follows directly from Claim 2 in [13]. See the full proof in Appendix A. □

Lemma 1. (LP) *has a polynomial number of variables and constraints.*

Proof. The proof is identical to Lemma 2 in [13]. See the full proof in Appendix A. □

Lemma 2. (LP) *is a valid relaxation of* GCMC.

Proof. Let k-partition $\{U_\alpha\}_{\alpha=1}^k$ be any feasible solution to $\mathcal{S}_G$. Let $\{b[i]\}_{i \in I}$ denote the states given by

Claim 1 corresponding to $\{U_\alpha\}_{\alpha=1}^k$. For any subset $N \in \mathcal{P}$ of nodes, and for all $\{s[i] \in \Sigma_i\}_{i \in N}$, set

$$y(s[N]) = \begin{cases} 1, & \text{if } s[i] = b[i] \text{ for all } i \in N; \\ 0, & \text{otherwise.} \end{cases}$$

Clearly constraints (4) and (8) are satisfied. By the first property in Claim 1, constraint (6) is satisfied. And by the second property in Claim 1, constraint (7) is also satisfied. The last property in Claim 1 implies that $v \in U_\alpha \iff b[\overline{v\alpha}]((v,\alpha)) = 1$ for any vertex $v \in V$. So any edge $\{u, v\}$ is cut by U_α exactly when $b[\overline{u\alpha}]((u,\alpha)) \neq b[\overline{v\alpha}]((v,\alpha))$. Using the setting of variable $z_{uv\alpha}$ in (3) it follows that $z_{uv\alpha}$ is exactly the indicator of edge $\{u, v\}$ being cut by U_α. Finally, the objective value is exactly the total weight of edges cut by the k-partition $\{U_\alpha\}_{\alpha=1}^k$ where the coefficient $\frac{1}{2}$ comes from the fact that that summation counts each cut-edge twice. Thus (LP) is a valid relaxation. □

Rounding Algorithm. This is a top-down procedure, exactly as in [13]. We start with the root node $r \in I$. Here $\{y(s[r]) : s[r] \in \Sigma_r\}$ defines a probability distribution over the states of r. We sample a state $a[r] \in \Sigma_r$ from this distribution. Then we continue top-down: for any node $i \in I$, given the chosen states $a[k]$ at each $k \in T_i$, we sample states for both children of i *simultaneously* from their joint distribution given at node i. Our algorithm is formally described in Algorithm 1.

Input : Optimal solution of LP.
Output: A vertex partition of V in $\mathcal{S}_G$.
1 Sample a state $a[r]$ at the root node by distribution $y(s[r])$.
2 **Do** *process all nodes i in $\mathcal{T}$ in order of increasing depth* **:**
3 Sample states $a[j], a[j']$ for the children of node i by joint distribution

$$\Pr[a[j] = s[j] \text{ and } a[j'] = s[j']] = \frac{y(s[T_i \cup \{j, j'\}])}{y(s[T_i])}, \quad (9)$$

where $s[T_i] = a[T_i]$.
4 **end**
5 **Do** *process all nodes i in $\mathcal{T}$ in order of decreasing depth* **:**
6 $h_i = a[i] \cup h_j \cup h_{j'}$ where j, j' are the children of i.
7 **end**
8 Set $U_\alpha = \{v \in V : h_r((v,\alpha)) = 1\}$ for all $\alpha \in [k]$.
9 **return** k-partition $\{U_\alpha\}_{\alpha=1}^k$.

Algorithm 1: Rounding Algorithm for LP

Lemma 3. *The algorithm's solution $\{U_\alpha\}_{\alpha=1}^k$ is always feasible.*

Proof. Note that the distributions used in Step 1 and Step 3 are well-defined due to Claim 2; so the states $a[i]$s are well-defined. Moreover, by the choice of these distributions, for each node i, $y(a[T_i]) > 0$.

We now show that for any node $i \in I$ with children j, j' we have $(a[j], a[j']) \in \mathcal{F}_{i,a[i]}$. Indeed, at the iteration for node i (when $a[j]$ and $a[j']$ are set), using the conditional probability distribution (9) and constraint (6), we have $(a[j], a[j']) \in \mathcal{F}_{i,a[i]}$ with probability one.

We show by induction that for each node $i \in I$, $h_i \in \mathcal{H}_{i,a[i]}$. The base case is when i is a leaf. In this case, due to constraint (7) and the fact that $y(a[T_i]) > 0$ we know that $\mathcal{H}_{i,a[i]} \neq \emptyset$. So $h_i = a[i] \in \mathcal{H}_{i,a[i]}$ by Theorem 3.1(3). For the inductive step, consider node $i \in I$ with children j, j' where $h_j \in \mathcal{H}_{j,a[j]}$ and $h_{j'} \in \mathcal{H}_{j',a[j']}$. Moreover, from the property above, $(a[j], a[j']) \in \mathcal{F}_{i,a[i]}$. Now using Theorem 3.1(4) we have $h_i = a[i] \cup h_j \cup h_{j'} \in \mathcal{H}_{i,a[i]}$. Finally, using $h_r \in \mathcal{H}_{r,a[r]}$ at the root node and Theorem 3.1(5), it follows that $h_r \in \text{Feas}(\vartheta)$. Now let h' denote the restriction of h_r to the variables $\{(v,\alpha) : v \in V, \alpha \in [k]\}$. Then, using the CSP extension result (Theorem 2.2) we obtain that h' is feasible for $CSP_\varphi(G)$. In other words, the k-partition $\{U_\alpha\}_{\alpha=1}^k$ satisfies $\mathcal{S}_G$. □

Claim 3. *For any node i and states $s[k] \in \Sigma_k$ for all $k \in T_i$, the rounding algorithm satisfies* $\Pr[a[T_i] = s[T_i]] = y(s[T_i])$.

Proof. The proof is identical to Claim 4 in [13]; see Appendix A. □

Lemma 4. *Consider any $u, v \in V$ and $\alpha \in [k]$ such that $\overline{u\alpha} \in T_{\overline{v\alpha}}$. Then the probability that edge (u, v) is cut by $U_\alpha = \{v \in V : h_r((v,\alpha)) = 1\}$ is $z_{uv\alpha}$.*

Proof. This follows directly from Lemma 4 in [13] by considering U_α as the solution. See Appendix A. □

Lemma 5. *Consider any $u, v \in V$ and $\alpha \in [k]$ such that $\overline{u\alpha} \notin T_{\overline{v\alpha}}$ and $\overline{v\alpha} \notin T_{\overline{u\alpha}}$. Then the probability that edge (u, v) is cut by $U_\alpha = \{v \in V : h_r((v,\alpha)) = 1\}$ is at least $z_{uv\alpha}/2$.*

Proof. This follows directly from Lemma 5 in [13] by considering U_α as the solution. See Appendix A. □

Lemma 6. *For any $u, v \in V$, the probability that edge (u, v) is cut by the k-partition $\{U_\alpha\}_{\alpha=1}^k$ is at least $\frac{1}{4}\sum_{\alpha=1}^k z_{uv\alpha}$.*

Proof. Edge (u, v) is cut by $\{U_\alpha\}_{\alpha=1}^k$ if and only if $u \in U_\alpha$ and $v \in U_\beta$ for some $\alpha \neq \beta$. Enumerating all partition parts and applying Lemmas 4 and 5, we get that the probability is at least $\frac{1}{4}\sum_{\alpha=1}^k z_{uv\alpha}$. The extra factor of $\frac{1}{2}$ is because any cut edge (u, v) is cut by the partition twice: by the parts containing u and v. □

From Lemmas 2, 3 and 6, we obtain Theorem 5.1.

6. Applications

We claim that MSO_2 is powerful enough to model various graph properties; to that end, consider the following formulae, meant to model that a set S is a vertex cover, an independent set, a dominating set, and a connected set, respectively:

$$\varphi_{\text{vc}}(S) \equiv \forall \{u,v\} \in E : (u \in S) \vee (v \in S)$$
$$\varphi_{\text{is}}(S) \equiv \forall \{u,v\} \in E : \neg((u \in S) \wedge (v \in S))$$
$$\varphi_{\text{ds}}(S) \equiv \forall v \in V : \exists u \in S : (v \notin S) \implies \{u,v\} \in E$$
$$\varphi_{\text{conn}}(S) \equiv \neg[\exists U, V \subseteq S : U \cap V = \emptyset \wedge U \cup V = S \wedge \neg(\exists \{u,v\} \in E : u \in U \wedge v \in V)]$$

We argue as follows: φ_{vc} is true if every edge has at least one endpoint in S; φ_{is} is true if every edge does not have both endpoints in S; φ_{ds} is true if for each vertex v not in S there is a neighbor u in S; finally, φ_{conn} is true if there does not exist a partition U, V of S with an edge going between U and V.

We also show how to handle the precedence constraint. Let G be a directed graph; we require S to satisfy that, for each arc $(u,v) \in E$, either $v \notin S$, or $u, v \in S$. This can be handled directly with CSP constraints: we have a binary variable for each vertex with the value 1 indicating that a vertex is selected for S; then, for each arc $(u,v) \in E$, we have a constraint $C_{(u,v)} = \{(1,0),(0,0),(1,1)\}$.

It is known [12] that many other properties are expressible in MSO_2, such as that S is k-colorable, k-connected (both for fixed $k \in \mathbb{N}$), planar, Hamiltonian, chordal, a tree, not containing a list of graphs as minors, etc. It is also known how to encode directed graphs into undirected graphs in an "MSO-friendly" way [12], which allows the expression of various properties of directed graphs. Our results also extend to so-called *counting* MSO, where we additionally have a predicate of the form $|X| = p \bmod q$ for a fixed integer $q \in \mathbb{N}$.

7. Conclusions

In this paper we obtained $\frac{1}{2}$-approximation algorithms for graph-MSO-constrained max-k-cut problems, where the constraint graph has bounded treewidth. This work generalizes the class of constraints handled in [13] and extends the result to the setting of max-k-cut. Getting an approximation ratio better than $\frac{1}{2}$ for any of these problems is an interesting question, even for a specific MSO-constraint. Regarding Remark 1, could our algorithm be improved to an FPT algorithm (runtime $g(\tau)n^{O(1)}$ for some function g)? If not, is there an FPT algorithm parameterized by the (more restrictive) tree-depth of G?

Acknowledgments

We thank Samuel Fiorini for raising the possibility of a systematic extension of our prior work [13], which eventually lead to this paper.

References

[1] K. C. Chang, D. H. Du, Efficient algorithms for layer assignment problem, IEEE Trans. on CAD of Integrated Circuits and Systems 6 (1) (1987) 67–78.
[2] F. Barahona, M. Grötschel, M. Jünger, G. Reinelt, An application of combinatorial optimization to statistical physics and circuit layout, Oper. Res. 36 (1988) 493–513.
[3] A. Jalali, N. Srebro, Clustering using max-norm constrained optimization, in: ICML, 2012.
[4] J. D. Lee, B. Recht, R. Salakhutdinov, N. Srebro, J. A. Tropp, Practical large-scale optimization for max-norm regularization, in: NIPS, 2010, pp. 1297–1305.
[5] M. X. Goemans, D. P. Williamson, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming, J.ACM 42 (1995) 1115–45.
[6] S. Khot, G. Kindler, E. Mossel, R. O'Donnell, Optimal inapproximability results for MAX-CUT and other 2-variable CSPs?, SIAM J. Comp. 37 (1) (2007) 319–357.
[7] A. A. Ageev, R. Hassin, M. Sviridenko, A 0.5-approximation algorithm for MAX DICUT with given sizes of parts, SIAM J. Discrete Math. 14 (2) (2001) 246–255.
[8] A. A. Ageev, M. Sviridenko, Approximation algorithms for maximum coverage and max cut with given sizes of parts, in: IPCO, Vol. 1610, 1999, pp. 17–30.
[9] C. Chekuri, J. Vondrák, R. Zenklusen, Submodular function maximization via the multilinear relaxation and contention resolution schemes, SIAM J. Comp. 43 (2014) 1831–79.
[10] M. Feldman, J. Naor, R. Schwartz, A unified continuous greedy algorithm for submodular maximization, in: FOCS, 2011, pp. 570–579.
[11] M. T. Hajiaghayi, G. Kortsarz, R. MacDavid, M. Purohit, K. K. Sarpatwar, Approximation algorithms for connected maximum cut and related problems, in: ESA, Vol. 9294, 2015, pp. 693–704.
[12] B. Courcelle, On the expression of graph properties in some fragments of monadic second-order logic., Descriptive complexity and finite models 31 (1996) 33–62.
[13] X. Shen, J. Lee, V. Nagarajan, Approximating graph-constrained max-cut, Math. Prog., Ser. B, online 2017.
[14] H. L. Bodlaender, NC-algorithms for graphs with small treewidth, in: Graph-Theoretic Concepts in Computer Science, WG 1988, Amsterdam, The Netherlands, June 15-17, 1988, Vol. 344 of Lec. Notes in Comp. Sci., 1988, pp. 1–10.
[15] D. Knop, M. Koutecký, T. Masařík, T. Toufar, Simplified algorithmic metatheorems beyond MSO: Treewidth and neighborhood diversity, in: WG, Vol. 10520, 2017, pp. 344–357.
[16] R. G. Downey, M. R. Fellows, Parameterized complexity, Springer Science & Business Media, 2012.

Appendix A. Omitted Proofs

Remain Proof of Claim 1. We define the states $b[i]$ in a top-down manner. We will also define an associated vector $t_i \in \mathcal{H}_{i,b[i]}$ at each node i. At the root, we set $b[r] = \sigma$ such that $t \in \mathcal{H}_{r,\sigma}$: this is well-defined because $t \in \bigcup_{\sigma \in \Sigma_r} \mathcal{H}_{r,\sigma}$. We also set $t_r = t$. Having set $b[i]$ and $t_i \in \mathcal{H}_{i,b[i]}$ for any node $i \in I$ with children $\{j, j'\}$, we use Theorem 3.1(4) to write $h_i = b[i] \cup h_j \cup h_{j'}$ where $h_j \in \mathcal{H}_{j,w_j}$, $h_{j'} \in \mathcal{H}_{j',w_{j'}}$ and $(w_j, w_{j'}) \in \mathcal{F}_{i,b[i]}$. Then we set $b[j] = w_j$, $t_j = h_j$ and

$b[j'] = w_{j'}$, $t_{j'} = h_{j'}$ for the children of node i. The first condition in the claim is immediate from the definition of states $b[i]$. By induction on the depth of node i, we obtain $t_i \in \mathcal{H}_{i,b[i]}$ for each node i. This implies that $\mathcal{H}_{\ell,b[\ell]} \neq \emptyset$ for each leaf ℓ, which proves the second condition; moreover, by Theorem 3.1(3) we have $t_\ell = b[\ell]$. Now, by definition of the vectors t_i, we obtain $t = t_r = \bigcup_{i\in I} b[i] = b_{\mathcal{T}}$ which, combined with $U_\alpha = \{v \in V : t((v,\alpha)) = 1\}$ for all $\alpha \in [k]$, proves the third condition in the claim.

Because $t = \bigcup_{i\in I} b[i]$, it is clear that if $b[\overline{v\alpha}]((v,\alpha)) = 1$ then $v \in U_\alpha$. In the other direction, suppose $b[\overline{v\alpha}]((v,\alpha)) \neq 1$: we will show $v \notin U_\alpha$. Since $\overline{v\alpha}$ is the highest node containing (v,α), it suffices to show that $t_{\overline{v\alpha}}((v,\alpha)) \neq 1$. But this follows directly from Theorem 3.1(2) because $t_{\overline{v\alpha}} \in \mathcal{H}_{\overline{v\alpha},b[\overline{v\alpha}]}$, $(v,\alpha) \in X_{\overline{v\alpha}}$ and $b[\overline{v\alpha}]((v,\alpha)) \neq 1$. □

Proof of Claim 2. Note that $T_i \cup \{j,j'\} \subseteq T_\ell$ for any leaf node ℓ in the subtree below i. So $T_i \cup \{j,j'\} \in \mathcal{P}$ and the variables $y(s[T_i \cup \{j,j'\}])$ are well-defined. The claim follows by two applications of (4). □

Proof of Lemma 1. There are $\binom{n}{2}\cdot k = O(kn^2)$ variables $z_{uv\alpha}$. Because the tree is binary, we have $|T_i| \leq 2d$ for any node i, where $d = O(\log n)$ is the depth of the tree decomposition. Moreover there are only $O(n^2)$ pairs of leaves as there are $O(n)$ leaf nodes. For each pair ℓ_1, ℓ_2 of leaves, we have $|T_{\ell_1} \cup T_{\ell_2}| \leq 4d$. Thus $|\mathcal{P}| \leq O(n^2)\cdot 2^{4d} = poly(n)$. By Theorem 3.1, we have $\max|\mathcal{H}_{i,\sigma}| = O(1)$, so the number of y-variables is at most $|\mathcal{P}|\cdot(\max|\mathcal{H}_{i,\sigma}|)^{4d} = poly(n)$. This shows that (LP) has polynomial size and can be solved optimally in polynomial time. Finally, it is clear that the rounding algorithm runs in polynomial time. □

Proof of Claim 3. We proceed by induction on the depth of node i. It is clearly true when $i = r$, i.e. $T_i = \{r\}$. Assuming the statement is true for node i, we will prove it for i's children. Let j, j' be the children nodes of i; note that $T_j = T_{j'} = T_i \cup \{j,j'\}$. Then using (9), we have

$$\Pr[a[T_j] = s[T_j] \mid a[T_i] = s[T_i]] = \frac{y(s[T_i \cup \{j,j'\}])}{y(s[T_i])}.$$

Combined with $\Pr[a[T_i] = s[T_i]] = y(s[T_i])$ we obtain $\Pr[a[T_j] = s[T_j]] = y(s[T_j])$ as desired. □

Proof of Lemma 4. Applying Claim 3 with node $i = \overline{v\alpha}$, for any $\{s[k] \in \Sigma_k : k \in T_{\overline{v\alpha}}\}$, we have $\Pr[a[T_{\overline{v\alpha}}] = s[T_{\overline{v\alpha}}]] = y(s[T_{\overline{v\alpha}}])$. Let $D_{u\alpha} = \{s[\overline{u\alpha}] \in \Sigma_{[\overline{u\alpha}]} \mid s[\overline{u\alpha}]((u,\alpha)) = 1\}$ and similarly $D_{v\alpha} = \{s[\overline{v\alpha}] \in \Sigma_{[\overline{v\alpha}]} \mid s[\overline{v\alpha}]((v,\alpha)) = 1\}$. Because $\overline{u\alpha} \in T_{\overline{v\alpha}}$,

$$\Pr[u \in U_\alpha, v \notin U_\alpha] = \sum_{s[\overline{u\alpha}]\in D_{u\alpha}} \sum_{s[\overline{v\alpha}]\notin D_{v\alpha}} \sum_{\substack{s[k]\in\Sigma_k \\ k\in T_{[\overline{v\alpha}]}\setminus\{\overline{u\alpha}\}\setminus\{\overline{v\alpha}\}}} y(s[\overline{v\alpha}])$$

$$= \sum_{s[\overline{u\alpha}]\in D_{u\alpha}} \sum_{s[\overline{v\alpha}]\notin D_{v\alpha}} y(s[\{\overline{u\alpha}, \overline{v\alpha}\}]).$$

The last equality above is by repeated application of LP constraint (4) where we use $T_{\overline{v\alpha}} \in \mathcal{P}$. Similarly,

$$\Pr[u \notin U_\alpha, v \in U_\alpha] = \sum_{s[\overline{u\alpha}]\notin D_{u\alpha}} \sum_{s[\overline{v\alpha}]\in D_{v\alpha}} y(s[\{\overline{u\alpha}, \overline{v\alpha}\}]),$$

which combined with constraint (3) implies $\Pr[|\{u,v\} \cap U_\alpha| = 1] = z_{uv\alpha}$. □

Proof of Lemma 5. We first state a useful observation.

Observation 1 (Observation 1 in [13])**.** *Let X, Y be two jointly distributed $\{0,1\}$ random variables. Then* $\Pr(X = 1)\Pr(Y = 0) + \Pr(X = 0)\Pr(Y = 1) \geq \frac{1}{2}[\Pr(X = 0, Y = 1) + \Pr(X = 1, Y = 0)]$.

Now we start to prove Lemma 5. In order to simplify notation, we define:

$$z^+_{uv\alpha} = \sum_{\substack{s[\overline{u\alpha}]\in\Sigma_{\overline{u\alpha}},\, s[\overline{v\alpha}]\in\Sigma_{\overline{v\alpha}} \\ s[\overline{u\alpha}]((u,\alpha))=1, s[\overline{v\alpha}]((v,\alpha))=0}} y(s[\{\overline{u\alpha}, \overline{v\alpha}\}]),$$

$$z^-_{uv\alpha} = \sum_{\substack{s[\overline{u\alpha}]\in\Sigma_{\overline{u\alpha}},\, s[\overline{v\alpha}]\in\Sigma_{\overline{v\alpha}} \\ s[\overline{u\alpha}]((u,\alpha))=0, s[\overline{v\alpha}]((v,\alpha))=1}} y(s[\{\overline{u\alpha}, \overline{v\alpha}\}]).$$

Note that $z_{uv} = z^+_{uv\alpha} + z^-_{uv\alpha}$.

Let $D_{u\alpha} = \{s[\overline{u\alpha}] \in \Sigma_{[\overline{u\alpha}]} \mid s[\overline{u\alpha}]((u,\alpha)) = 1\}$ and $D_{v\alpha} = \{s[\overline{v\alpha}] \in \Sigma_{[\overline{v\alpha}]} \mid s[\overline{v\alpha}]((v,\alpha)) = 1\}$. Let i denote the least common ancestor of nodes $\overline{u\alpha}$ and $\overline{v\alpha}$, and $\{j,j'\}$ the two children of i. Note that $T_j = T_{j'} = T_i \cup \{j,j'\}$ and $T_{\overline{u\alpha}}, T_{\overline{v\alpha}} \supseteq T_j$. Because $\overline{u\alpha} \notin T_{\overline{v\alpha}}$ and $\overline{v\alpha} \notin T_{\overline{u\alpha}}$, both $\overline{u\alpha}$ and $\overline{v\alpha}$ are strictly below j and j' (respectively) in the tree decomposition.

For any choice of states $\{s[k] \in \Sigma_k\}_{k\in T_j}$ define:

$$z^+_{uv\alpha}(s[T_j]) = \sum_{s[\overline{u\alpha}]\in D_{u\alpha}} \sum_{s[\overline{v\alpha}]\notin D_{v\alpha}} \frac{y(s[T_j \cup \{\overline{u\alpha}, \overline{v\alpha}\}])}{y(s[T_j])},$$

and similarly $z^-_{uv\alpha}(s[T_j])$.

In the rest of the proof, we fix states $\{s[k] \in \Sigma_k\}_{k\in T_j}$ and *condition* on the event $\mathcal{E}$ that $a[T_j] = s[T_j]$. We will show $\Pr[|\{u,v\} \cap U_\alpha| = 1 \mid \mathcal{E}]$

$$\geq \frac{1}{2}\left(z^+_{uv\alpha}(s[T_j]) + z^-_{uv\alpha}(s[T_j])\right). \tag{A.1}$$

By taking expectation over the conditioning $\mathcal{E}$, this would imply Lemma 5.

We now define the following indicator random variables (conditioned on $\mathcal{E}$).

$$I_{u\alpha} = \begin{cases} 0 & \text{if } a[\overline{u\alpha}] \notin D_{u\alpha} \\ 1 & \text{if } a[\overline{u\alpha}] \in D_{u\alpha} \end{cases} \quad \text{and} \quad I_{v\alpha} = \begin{cases} 0 & \text{if } a[\overline{v\alpha}] \notin D_{v\alpha} \\ 1 & \text{if } a[\overline{v\alpha}] \in D_{v\alpha} \end{cases}$$

Observe that $I_{u\alpha}$ and $I_{v\alpha}$ (*conditioned* on $\mathcal{E}$) are independent because $\overline{u\alpha}, \overline{v\alpha} \notin T_j$, and $\overline{u\alpha}$ and $\overline{v\alpha}$ appear in distinct subtrees under node i. So,

$$\Pr[|\{u,v\} \cap U_\alpha| = 1 \mid \mathcal{E}]$$
$$= \Pr[I_{u\alpha} = 1]\cdot\Pr[I_{v\alpha} = 0] + \Pr[I_{u\alpha} = 0]\cdot\Pr[I_{v\alpha} = 1] \tag{A.2}$$

For any $s[k] \in \Sigma_k$ for $k \in T_{\overline{u\alpha}} \setminus T_j$, we have by Claim 3 and $T_j \subseteq T_{\overline{u\alpha}}$ that

$$\Pr[a[T_{\overline{u\alpha}}] = s[T_{\overline{u\alpha}}] \mid a[T_j] = s[T_j]] \;=\; \frac{y(s[T_{\overline{u\alpha}}])}{y(s[T_j])}.$$

Therefore $\Pr[I_{u\alpha} = 1]$ equals

$$\sum_{s[\overline{u\alpha}] \in D_{u\alpha}} \sum_{\substack{k \in T_{\overline{u\alpha}} \setminus T_j \setminus \{\overline{u\alpha}\} \\ s[k] \in \Sigma_k}} \frac{y(s[T_{\overline{u\alpha}}])}{y(s[T_j])} = \sum_{s[\overline{u\alpha}] \in D_{u\alpha}} \frac{y(s[T_j \cup \{\overline{u\alpha}\}])}{y(s[T_j])}.$$

The last equality follows by repeatedly using LP constraint (4) and the fact that $T_{\overline{u\alpha}} \in \mathcal{P}$. Furthermore, note that $T_j \cup \{\overline{u\alpha}, \overline{v\alpha}\} \in \mathcal{P}$; again by constraint (4),

$$\begin{aligned}
\Pr[I_{u\alpha} = 1] &= \sum_{s[\overline{u\alpha}] \in D_{u\alpha}} \frac{y(s[T_j \cup \{\overline{u\alpha}\}])}{y(s[T_j])} \\
&= \sum_{s[\overline{u\alpha}] \in D_{u\alpha}} \sum_{s[\overline{v\alpha}] \in \Sigma_{\overline{v\alpha}}} \frac{y(s[T_j \cup \{\overline{u\alpha}, \overline{v\alpha}\}])}{y(s[T_j])} \\
&= \sum_{s[\overline{u\alpha}] \in D_{u\alpha}} \sum_{s[\overline{v\alpha}] \in D_{v\alpha}} \frac{y(s[T_j \cup \{\overline{u\alpha}, \overline{v\alpha}\}])}{y(s[T_j])} + z^+_{uv\alpha}(s[T_j])
\end{aligned}$$

Similarly,

$$\begin{aligned}
\Pr[I_{v\alpha} = 1] &= \sum_{s[\overline{v\alpha}] \in D_{v\alpha}} \frac{y(s[T_j \cup \{\overline{v\alpha}\}])}{y(s[T_j])} \\
&= \sum_{s[\overline{u\alpha}] \in D_{u\alpha}} \sum_{s[\overline{v\alpha}] \in D_{v\alpha}} \frac{y(s[T_j \cup \{\overline{u\alpha}, \overline{v\alpha}\}])}{y(s[T_j])} + z^-_{uv\alpha}(s[T_j]).
\end{aligned}$$

$$\begin{aligned}
\Pr[I_{u\alpha} = 0] &= \sum_{s[\overline{u\alpha}] \notin D_{u\alpha}} \frac{y(s[T_j \cup \{\overline{u\alpha}\}])}{y(s[T_j])} \\
&= \sum_{s[\overline{u\alpha}] \notin D_{u\alpha}} \sum_{s[\overline{v\alpha}] \notin D_{v\alpha}} \frac{y(s[T_j \cup \{\overline{u\alpha}, \overline{v\alpha}\}])}{y(s[T_j])} + z^-_{uv\alpha}(s[T_j]).
\end{aligned}$$

Now define $\{0, 1\}$ random variables X and Y jointly distributed as:

	$Y = 0$	$Y = 1$
$X = 0$	$\Pr[I_{u\alpha} = 0] - z^-_{uv\alpha}(s[T_j])$	$z^-_{uv\alpha}(s[T_j])$
$X = 1$	$z^+_{uv\alpha}(s[T_j])$	$\Pr[I_{u\alpha} = 1] - z^+_{uv\alpha}(s[T_j])$

Note that $\Pr[X = 1] = \Pr[I_{u\alpha} = 1]$ and $\Pr[Y = 1] = \Pr[I_{u\alpha} = 1] - z^+_{uv\alpha}(s[T_j]) + z^-_{uv\alpha}(s[T_j]) = \Pr[I_{v\alpha} = 1]$. So, applying Observation 1 and using (A.2) we have $\Pr[|\{u, v\} \cap U_\alpha| = 1 \mid \mathcal{E}]$ is at least

$$\frac{1}{2}\left(\Pr[X = 0, Y = 1] + \Pr[X = 1, Y = 0]\right),$$

which implies (A.1). □