Optimal Extended Formulations from Optimal Dynamic Programming Algorithms

Mateus de Oliveira Oliveira[1,2] , **Wim Van den Broeck**[2]
[1]Department of Computer and Systems Sciences, Stockholm University, Sweden
[2]Department of Informatics, University of Bergen, Norway
oliveira@dsv.su.se, wim.broeck@uib.no

Abstract

Vertex Subset Problems (VSPs) are a class of combinatorial optimization problems on graphs where the goal is to find a subset of vertices satisfying a predefined condition. Two prominent approaches for solving VSPs are *dynamic programming* over tree-like structures, such as tree decompositions or clique decompositions, and *linear programming*. In this work, we establish a sharp connection between both approaches by showing that if a vertex-subset problem Π admits a solution-preserving dynamic programming algorithm that produces tables of size at most $\alpha(k,n)$ when processing a tree decomposition of width at most k of an n-vertex graph G, then the polytope $P_\Pi(G)$ defined as the convex-hull of solutions of Π in G has extension complexity at most $O(\alpha(k,n)\cdot n)$. Additionally, this upper bound is optimal under the exponential time hypothesis (ETH).

On the one hand, our results imply that ETH-optimal solution-preserving dynamic programming algorithms for combinatorial problems yield optimal-size parameterized extended formulations for the solution polytopes associated with instances of these problems. On the other hand, unconditional lower bounds obtained in the realm of the theory of extended formulations yield unconditional lower bounds on the table complexity of solution-preserving dynamic programming algorithms.

1 Introduction

Computational problems arising in a wide variety of subfields of artificial intelligence can be formalized as vertex-subset problems, such as INDEPENDENT SET, DOMINATING SET, VERTEX COVER, etc. Two prominent algorithmic approaches to attack these problems are *dynamic programming* over tree-like structures, and linear programming. While the former approach is suitable to solve vertex subset problems on graphs of small width, such as treewidth [Marx, 2007] and cliquewidth [Courcelle *et al.*, 2000], the latter approach yields practical algorithms whenever the space of feasible solutions can be defined using a system of linear inequalities of moderate size [Hu and Laurent, 2019; Aprile *et al.*, 2017].

Vertex subset problems that are NP-hard do not admit efficient linear programming formulations assuming $P \neq NP$, and in some cases it can be even shown unconditionally that formulations for certain problems require exponential size [Yannakakis, 1988; Fiorini *et al.*, 2015; De Simone, 1990; Göös *et al.*, 2018a]. Nevertheless, recently there has been growing interest in the study of linear formulations of combinatorial polytopes parameterized by structural parameters of the input graph, such as treewidth and cliquewidth [Seif, 2019; Aboulker *et al.*, 2019; Buchanan and Butenko, 2015; Kolman *et al.*, 2020]. In this work, we contribute with this line of research by establishing a sharp connection between dynamic programming algorithms operating on tree decompositions and linear programming theory.

On the one hand, our results allow us to show that efficient solution-preserving dynamic programming algorithms parameterized by treewidth can be used to define small linear programming formulations. On the other hand, unconditional lower bounds obtained in the realm of the theory of extended formulations can be used to provide unconditional lower bounds on the table-complexity of solution-preserving dynamic programming algorithms expressible in the dynamic programming core model (DP-core model)[Baste *et al.*, 2022; de Oliveira Oliveira and Vadiee, 2023].

1.1 Our Results

Our main result (Theorem 14) states that if a vertex subset problem Π can be solved by a solution-preserving DP-core of table complexity $\alpha(k,n)$ when processing a width-k tree decomposition of an n-vertex graph G, then the solution polytope of Π on G has extension complexity $O(\alpha(k,n)\cdot n)$. Intuitively, problems that can be solved by solution-preserving DP-cores with small table complexity yield solution polytopes of small extension complexity. Conversely, lower bounds on the extension complexity of solution polytopes of a given problem Π yield lower bounds on the table complexity of solution-preserving DP-algorithms solving Π.

1.2 Proof Techniques

We formalize the notion of a dynamic programming algorithm parameterized by treewidth using an extension of the

DP-core model introduced in [Baste *et al.*, 2022] and further developed in [de Oliveira Oliveira and Vadiee, 2023]. The main conceptual addition to these models is an axiomatic definition of the notion of a solution-preserving dynamic programming algorithm. Intuitively, this notion formalizes dynamic programming algorithms satisfying two properties: first, any subset of vertices obtained by backtracking is a solution for the problem in question. Second, every solution should be retrievable by backtracking. It is worth noting that in general, dynamic programming algorithms may not be solution preserving. For instance, if one is simply interested in determining whether a solution exists then some solutions may be discarded during the dynamic programming process.

The bridge between dynamic programming and linear programming is made within the context of tree automata theory. More specifically, we define the notion of a T-shaped tree automaton, where T is a tree. Intuitively, these are automata accepting sets of terms over a given alphabet Σ, all of which have the same underlying tree-structure. The state set of such an automaton $\mathcal{A}$ is partitioned into a collection of cells, one cell Q_u per node u of T, and we define the width of $\mathcal{A}$ as the size of the largest cell. We then show that if $\mathcal{A}$ is a T-shaped tree automaton of width $\mathrm{w}(\mathcal{A})$, the polytope $\mathrm{P}(\mathcal{A})$ whose vertices correspond to terms accepted by $\mathcal{A}$ has extension complexity $\mathcal{O}(|T| \cdot (\mathrm{w}(\mathcal{A}) + |\Sigma|))$ (Corollary 6). Finally, we show that if $\mathtt{D}$ is a solution-preserving DP-core of table complexity $\alpha(k, n)$ solving a problem Π, then given an n-vertex graph G together with a tree decomposition $\mathcal{T}$ of width at most k and underlying tree structure T, one can construct a T-shaped tree automaton of width $\alpha(k, n)$ whose accepted terms encode solutions for Π in G (Theorem 13). By combining Theorem 13 with Corollary 6, we infer that the extension complexity of the solution polytope of Π on G is upper bounded by $O(\alpha(k, n) \cdot n)$.

1.3 Generalizations

Although for simplicity of exposition our main results will be stated in terms of vertex subset problems, it is worth noting that these results also generalize to edge subset problems, where the goal is to find a subset of edges satisfying a given condition (for instance, CUT$_\ell$, HAMILTONIANCYCLE), and to problems where feasible solutions are tuples of subsets of vertices, edges, or both. For instance for each fixed d, the d-COLORING problem is a problem where the goal is to partition the vertex set of the graph into d subsets, each of which is an independent set. Extensions for the cases mentioned above are dealt with in the full version of this paper.

2 Preliminaries

2.1 Basic Definitions

We denote by $\mathbb{N}$ the set of natural numbers and by $\mathbb{N}_+$ the set of positive natural numbers. For each $n \in \mathbb{N}$, we let $[n] = \{1, \ldots, n\}$. In particular, $[0] = \emptyset$. Given a set S, the set of subsets of S is denoted by $\mathcal{P}(S)$, and the set of finite subsets of S is denoted by $\mathcal{P}_{\mathtt{fin}}(S)$. Given a function $f : S \to R$, and $S' \subseteq S$, we let $f(S') = \{f(s) \mid s \in S'\}$ be the image of S' under f.

We define a *graph* as a triple $G = (V, E, \rho)$, where $V \subset \mathbb{N}$ is a finite set of *vertices*, $E \subset \mathbb{N}$ is a finite set of *edges* and $\rho \subset E \times V$ is an incidence relation. For an edge $e \in E$, we let $\mathtt{endpts}(e) = \{v \in V | (e, v) \in \rho\}$ denote the set of vertices incident to e. We may write V_G, E_G and ρ_G to denote sets V, E and ρ. We define the *empty graph* as the graph $(\emptyset, \emptyset, \emptyset)$ with neither vertices, nor edges. We let GRAPHS denote the set of all graphs.

An *isomorphism* from a graph G to a graph H is a pair of functions $\phi = (\phi_1, \phi_2)$ where $\phi_1 : V_G \to V_H$ is a bijection from the vertex set of G to the vertex set of H and $\phi_2 : E_G \to E_H$ is a bijection from the edge set of G to the edge set of H such that for each vertex $v \in V_G$ and each edge $e \in E_G$, we have that $(e, v) \in \rho_G$ if and only if $(\phi_2(e), \phi_1(v)) \in \rho_H$. If there exists such a bijection, we say that graphs G and H are *isomorphic* and denote this by $G \sim H$.

2.2 Vertex Subset Problems

Definition 1 (Vertex Subset Problem)**.** *A* vertex subset problem *is a subset* $\Pi \subset$ GRAPHS $\times \mathcal{P}_{\mathtt{fin}}(\mathbb{N})$ *where the following conditions are satisfied for each pair* $(G, X) \in \Pi$*:*

1. $X \subseteq V_G$*, and*
2. *for each graph* H *isomorphic to* G*, and each isomorphism* $\phi = (\phi_1, \phi_2)$ *from* G *to* H*,* $(H, \phi_1(X)) \in \Pi$*.*

Given a graph G, we say that a subset $X \subseteq V_G$ is a *solution for* Π in G if $(G, X) \in \Pi$. The set of solutions for Π in G is denoted by

$$\mathrm{Sol}_\Pi(G) = \{X \ : \ (G, X) \in \Pi\}. \tag{1}$$

For example, for each $\ell \in \mathbb{N}_+$, INDEPENDENTSET$_\ell$ denotes the vertex subset problem consisting of all pairs (G, X) where G is a graph and X is an independent set of size at least ℓ in G, that is, a subset of vertices where no pair of vertices is connected by an edge. Other prominent examples of vertex subset problems are VERTEXCOVER$_\ell$, DOMINATINGSET$_\ell$ and CLIQUE$_\ell$.

2.3 Solution Polytopes

Let $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ be a set of variables. A vector over $\mathcal{X}$ is a function $v : \mathcal{X} \to \mathbb{R}$. We say that a vector $v : \mathcal{X} \to \mathbb{R}$ satisfies an inequality $\sum_i \alpha_i x_i \leq b$ if the inequality holds whenever each variable x_i is replaced by the value $v(x_i)$. That is to say, if $\sum_i \alpha_i v(x_i) \leq b$. A vector v is a convex combination of a set of vectors $U = \{v_1, \ldots, v_m\}$ if $v = \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_m v_m$ for some non-negative real numbers α_i such that $\sum_i \alpha_i = 1$. The *convex hull* of U is the set $\mathrm{conv}(U)$ of all convex combinations of vectors in U. We note that $\mathrm{conv}(U)$ is a polytope, and as such can be defined as the set of solutions of a system of linear inequalities.

Let G be an n-vertex graph, and let $X \subseteq V_G$. We let $\hat{X} : \mathcal{X} \to \mathbb{R}$ denote the vector over variables $\mathcal{X} = \{x_v \ : \ v \in V_G\}$ where for each $v \in V_G$, $\hat{X}(x_v) = 1$ if and only if $v \in X$. Given a vertex subset problem Π and a graph G, we let $P_\Pi(G)$ be the polytope defined as the convex hull of vectors associated with solutions of Π in G. More specifically, $P_\Pi(G) = \mathrm{conv}(\{\hat{X} \ : \ X \in \mathrm{Sol}_\Pi(G)\})$.

2.4 Addressed Trees

An *addressed tree* is a four-tuple $T = (N, \mathcal{F}, r, \gamma)$ denoting a rooted tree with nodes N, root $r \in N$, arcs $\mathcal{F} \subseteq N \times N$ directed from the leaves towards the root, and arc-labeling function $\gamma : \mathcal{F} \to \mathbb{N}$ that labels the arcs of T with numbers in such a way that for each $u \in N$, the arcs in the set $In(u) = \{(v, u) \mid (v, u) \in \mathcal{F}\}$ are injectively labeled with numbers from $\{1, \ldots, |In(u)|\}$. For each arc $(v, u) \in In(u)$, we say that v is the i-th child of u if $\gamma(v, u) = i$. We may write $N(T)$ to denote the set of nodes of T, and $r(T)$ to denote the root of T.

3 Dynamic Programming Cores for Tree Decompositions

The formalism we use to express dynamic programming algorithms operates on edge-introducing tree decompositions. For a matter of consistency with the notation in the remainder of the paper, our definition of tree decompositions uses a slightly distinct notation than the one typically used in the literature.

We define an *edge-introducing* tree decomposition of a graph G as a triple $\mathcal{T} = (T, B, \xi)$ where T is an addressed tree, $\xi : E(G) \to N(T)$ is an *injective* function, and $B : N(T) \to \mathcal{P}(V_G)$ is a function satisfying the following conditions:

1. For each vertex $v \in V(G)$, there is a node $u \in N(T)$ such that $v \in B(u)$;
2. for each edge $e \in E(G)$, $\texttt{endpts}(e) \subseteq B(\xi(e))$;
3. for each $v \in V(G)$, the set $\{u \in N(T) \mid v \in B(u)\}$ induces a connected sub-tree of T.

The *width* of $\mathcal{T}$ is defined as $\max_u |B(u)| - 1$. The treewidth of a graph G, denoted by $tw(G)$, is defined as the minimum width of an edge-introducing tree decomposition of G.

An edge-introducing tree decomposition $\mathcal{T}$ is said to be *nice* if each node $u \in N(T)$ has at most two children and the following conditions are satisfied: If u has two children u' and u'', then $B(u) = B(u') = B(u'')$. In this case, u is called a *join node*. If u has a single child u', then it either introduces a vertex v (meaning that $B(u) \setminus B(u') = \{v\}$), or it forgets a vertex v (meaning that $B(u') \setminus B(u) = \{v\}$), or it introduces an edge e (meaning that $B(u) = B(u')$ and $\xi(e) = u$). If u is a leaf node, or the root node, then $B(u) = \emptyset$. In this work, we assume that edge-introducing tree decompositions are nice. This assumption is without loss of generality, since any tree decomposition of width k of a graph G can be transformed into a nice edge-introducing tree decomposition of G of width at most k in time $O(k \cdot |N(T)|)$ [Kloks, 1994].

We formalize the notion of a dynamic programming algorithm operating on tree decompositions using the notion of a *dynamic programming core*. Our formalism is essentially equivalent in expressiveness as the notion of a DP-core introduced in [Baste *et al.*, 2022] in the sense that dynamic programming algorithms developed in either model can be easily converted to each other with minor adaptations.

Definition 2. *A DP-core is a* 6*-tuple* $\texttt{D}$ *whose components are specified as follows.*

1. Leaf *is a finite non-empty subset of* $\{0,1\}^*$.
2. $\text{IntroVertex} : \mathbb{N} \times \{0,1\}^* \to \mathcal{P}_{\mathit{fin}}(\{0,1\}^*)$
3. $\text{IntroEdge} : \mathbb{N} \times \mathbb{N} \times \{0,1\}^* \to \mathcal{P}_{\mathit{fin}}(\{0,1\}^*)$.
4. $\text{ForgetVertex} : \mathbb{N} \times \{0,1\}^* \to \mathcal{P}_{\mathit{fin}}(\{0,1\}^*)$.
5. $\text{Join} : \{0,1\}^* \times \{0,1\}^* \to \mathcal{P}_{\mathit{fin}}(\{0,1\}^*)$.
6. $\text{Final} : \{0,1\}^* \to \{0,1\}$.

For each $S \subseteq \{0,1\}^*$, we define $\text{IntroVertex}(v, S) = \bigcup_{\mathrm{w} \in S} \text{IntroVertex}(v, \mathrm{w})$. The extension of the remaining components of $\texttt{D}$ to subsets of strings is defined analogously.

Intuitively, a DP-core $\texttt{D}$ is a specification of a dynamic programming algorithm that operates on edge-introducing tree decompositions of graphs. The algorithm processes such a tree decomposition $\mathcal{T} = (T, B, \xi)$ from the leaves towards the root, assigning to each node $u \in N(T)$ a finite subset of strings $\Gamma(u)$ according to the following inductive process.

1. If u is a leaf node of T, $\Gamma(u) = \text{Leaf}$
2. If u is an internal node of T with a unique child u', then
   (a) $\Gamma(u) = \text{IntroVertex}(v, \Gamma(u'))$ if vertex v is introduced at u.
   (b) $\Gamma(u) = \text{ForgetVertex}(v, \Gamma(u'))$ if vertex v is forgotten at u.
   (c) $\Gamma(u) = \text{IntroEdge}(v, v', \Gamma(u'))$ if an edge connecting v and v' is introduced at u.
3. If u is a join node of T with children u' and u'' then $\Gamma(u) = \text{Join}(\Gamma(u'), \Gamma(u''))$.

We say that $\texttt{D}$ accepts the tree decomposition $\mathcal{T}$ if the set $\Gamma(r(T))$ associated with the root node of T has a final string. That is to say, $\mathrm{w} \in \Gamma(r(T))$ with $\text{Final}(\mathrm{w}) = 1$.

Definition 3. *We say that* $\texttt{D}$ *solves a vertex subset problem* Π *if for each graph* G*, and each tree decomposition* $\mathcal{T}$ *of* G*,* $\texttt{D}$ *accepts* $\mathcal{T}$ *if and only if* $\text{Sol}_\Pi(G) \neq \emptyset$.

We say that $\texttt{D}$ has *table complexity* $\alpha(k, n)$ if for each n-vertex graph G, each tree decomposition $\mathcal{T} = (T, B, \xi)$ of G of width at most k, and each node $u \in N(T)$, the set $\Gamma(u)$ has at most $\alpha(k, n)$ elements.

A DP-core for Independent Set. As an illustration, we define below a DP-core $\texttt{D}$ for the problem $\text{IndependentSet}_\ell$. It is enough to specify the action of each component of the core on strings that encode pairs of the form (S, c) where $S \subseteq \mathbb{N}$ and $c \in \mathbb{N}$. The particular encoding itself is not relevant, as long as it is fixed. We let $\texttt{D}$ be the DP-core whose components are defined as follows.

- $\text{Leaf} = \{(\emptyset, 0)\}$.
- $\text{IntroVertex}(v, (S, c)) = \{(S, c), (S \cup \{v\}, c + 1)\}$.
- $\text{IntroEdge}(v, v', (S, c)) = \begin{cases} \emptyset \text{ if } \{v, v'\} \subseteq S \\ \{(S, c)\} \text{ otherwise.} \end{cases}$
- $\text{ForgetVertex}(v, (S, c)) = \{(S \setminus \{v\}, c)\}$.

- $\mathrm{Join}((S_1, c_1), (S_2, c_2)) = \begin{cases} \emptyset \text{ if } S_1 \neq S_2 \\ \{(S_1, c_1 + c_2 - |S_1|)\} \text{ otw.} \end{cases}$
- $\mathrm{Final}((S, c)) = 1$ if and only if $c \geq \ell$.

The next proposition implies that the DP-core D specified above solves INDEPENDENTSET$_\ell$. Below, G_u means the subgraph of G induced by the set $\bigcup_{t \in N(T_u)} B(t)$, where T_u is the sub-tree of T rooted at u.

Proposition 4. *Let G be a graph, $\mathcal{T} = (T, B, \xi)$ be a tree decomposition of G. Then, for each $u \in N(T)$, and each pair $(S, c) \in \mathcal{P}(\mathbb{N}) \times \mathbb{N}$, (S, c) belongs to $\Gamma(u)$ if and only if the graph G_u has an independent set I of size at least c such that $S = I \cap B(u)$.*

In particular, G has an independent set of size at least ℓ if and only for some $\ell' \geq \ell$, the pair $(\emptyset, \ell')$ belongs to $\Gamma(r(T))$, and this happens if and only if $\mathcal{T}$ is accepted by D. We also note that by Proposition 4, we have that if $\mathcal{T}$ has width k and G has n vertices, then $|\Gamma(u)| \leq 2^k \cdot |V_G|$. Therefore, the table complexity of D is upper bounded by $2^k \cdot n$. Although very simple, this the DP-core defined above is essentially optimal under SETH [Lokshtanov *et al.*, 2018].

4 The Polytope of a T-Shaped Language

Let Σ be a set of symbols. A term over Σ is a pair $\tau = (T, \lambda)$ where T is an addressed tree, and $\lambda : N(T) \to \Sigma$ is a function that labels each node of T with a symbol from Σ. We say that a term τ is T-shaped if $\tau = (T, \lambda)$ for some λ. Whenever clear from the context, we may write $\tau(u)$ to denote the label $\lambda(u)$. We let $\mathtt{Terms}(\Sigma)$ denote the set of all terms over Σ.

4.1 T-Shaped Tree Automata

Let T be an addressed tree. A T-shaped tree automaton is a tuple $\mathcal{A} = (\Sigma, Q, F, \Delta)$ where $Q = \bigcup_u Q_u$ is a set of states, partitioned into a collection of subsets $\{Q_u\}_{u \in N(T)}$, $F \subseteq Q_{r(T)}$ is a set of final states, and $\Delta = \bigcup_u \Delta_u$ is a set of transitions, partitioned into a collection of subsets $\{\Delta_u\}_{u \in N(T)}$ such that the following condition is satisfied: for each node $u \in N(T)$ with children $u_1, \ldots, u_s$,

$$\Delta_u \subseteq Q_{u_1} \times \cdots \times Q_{u_s} \times \Sigma \times Q_u.$$

Let $\tau = (T, \lambda)$ be a term over Σ. A trace for τ in $\mathcal{A}$ is a function $\rho : N(T) \to Q$ such that for each node $u \in N(T)$ with children $u_1, \ldots, u_s$, the tuple $(\rho(u_1), \ldots, \rho(u_s), \tau(u), \rho(u))$ is a transition in Δ_u. We say that ρ is an *accepting trace* if $\rho(r(T)) \in F$. We say that τ is accepted by $\mathcal{A}$ if there is an accepting trace ρ for τ in $\mathcal{A}$. The *language* of $\mathcal{A}$ is the set $\mathcal{L}(\mathcal{A})$ of all terms accepted by $\mathcal{A}$.

4.2 The Polytope $P(\mathcal{A})$

Let T be an addressed tree and Σ be a set of symbols. We let

$$\mathcal{X}(T, \Sigma) = \{x_{u,a} \mid u \in N(T), a \in \Sigma\}$$

denote the set of variables associated with T and Σ. Given a T-shaped term $\tau \in \mathtt{Terms}(\Sigma)$, we let $\hat{\tau} : \mathcal{X}(T, \Sigma) \to \{0, 1\}$ be the 0/1 vector defined by setting, for each $u \in N(T)$ and each symbol $a \in \Sigma$, $\hat{\tau}(x_{u,a}) = 1$ if $\tau(u) = a$ and $\hat{\tau}(x_{u,a}) = 0$ otherwise. Given a T-shaped tree automaton $\mathcal{A}$, we define the polytope associated with $\mathcal{A}$, denoted by $\mathrm{P}(\mathcal{A})$, as the convex hull of all vectors encoding terms accepted by $\mathcal{A}$:

$$\mathrm{P}(\mathcal{A}) = \mathrm{conv}(\{\hat{\tau} | \tau \in \mathcal{L}(\mathcal{A})\}).$$

4.3 An Extended Formulation for $P(\mathcal{A})$

A polytope P' is an extended formulation of a polytope P if P is a projection of P'. The next theorem, which is the main result of this section, states that polytopes associated with T-shaped tree automata have extended formulations that can be defined by systems of linear inequalities whose size is linear on the number of states plus the number of transitions of the automata.

Theorem 5. *Let $\mathcal{A} = (\Sigma, Q, F, \Delta)$ be a T-shaped tree automaton. The polytope $\mathrm{P}(\mathcal{A})$ has an extended formulation with $O(|Q| + |\Delta|)$ extension variables and $\mathcal{O}(|Q| + |T| \cdot |\Sigma|)$ inequalities.*

We define the *width* of a T-shaped tree automaton $\mathcal{A}$, denoted by $\mathrm{w}(\mathcal{A})$, as the maximum size of a state cell:

$$\mathrm{w}(\mathcal{A}) = \max_{u \in N(T)} |Q_u|.$$

Since a T-shaped tree automaton of width w can have at most $w \cdot |T|$ states, Theorem 5 immediately implies the following corollary:

Corollary 6. *Let $\mathcal{A} = (\Sigma, Q, F, \Delta)$ be a T-shaped tree automaton of width $\mathrm{w}(\mathcal{A})$. The polytope $\mathrm{P}(\mathcal{A})$ has extension complexity $\mathcal{O}(|T| \cdot (\mathrm{w}(\mathcal{A}) + |\Sigma|))$.*

In the remainder of this section we define an extended formulation for the polytope $P(\mathcal{A})$ satisfying the conditions of Theorem 5. We start by specifying the extension variables, which we split into two sets. The first set,

$$\mathcal{Y}(\mathcal{A}) = \{y_{u,q} | u \in N(T), q \in Q_u\},$$

has one variable $y_{u,q}$ for each node u in $N(T)$ and each state q in Q_u. We call the elements of $\mathcal{Y}(\mathcal{A})$ *state variables*. The second set,

$$\mathcal{Z}(\mathcal{A}) = \{z_{u,\delta} | u \in N(T), \delta \in \Delta_u\},$$

has one variable $z_{u,\delta}$ for each node u in $N(T)$ and each transition δ in Δ_u. We call the elements of $\mathcal{Z}(\mathcal{A})$ *transition variables*. We let $\mathrm{var}(\mathcal{A}) = \mathcal{X}(T, \Sigma) \cup \mathcal{Y}(\mathcal{A}) \cup \mathcal{Z}(\mathcal{A})$ be the set of variables associated with $\mathcal{A}$. In what follows, we will refer to the sets $\mathcal{X}(T, \Sigma)$, $\mathcal{Y}(\mathcal{A})$ and $\mathcal{Z}(\mathcal{A})$ simply as $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ respectively.

Given a transition $\delta = (q_1, \ldots, q_s, a, q)$ in Δ, we say that q is the *consequent state* of δ, while for each $i \in [s]$, q_i is an *antecedent state* of δ. We denote the consequent state q by $\mathtt{cnq}(\delta)$ and the set of antecedent states $\{q_1, \ldots, q_s\}$ by $\mathtt{ant}(\delta)$. We denote by $\mathtt{symb}(\delta)$ the symbol a of δ.

Let $z_{u,\delta}$ be a transition variable in $\mathcal{Z}$ for some $u \in N(T)$ and transition $\delta \in \Delta_u$. If $q = \mathtt{cnq}(\delta)$, we say that $z_{u,\delta}$ is an *incoming* transition variable of $y_{u,q}$. If u' is a child of u and $q' \in \mathtt{ant}(\delta)$, we say that $z_{u,\delta}$ is an *outgoing* transition variable of $y_{u',q'}$.

Definition 7. *Let $\mathcal{A} = (\Sigma, Q, F, \Delta)$ be a T-shaped tree automaton, $\tau = (T, \lambda)$ be a term in $\mathcal{L}(\mathcal{A})$, and $\rho : N(T) \to Q$ be a trace for τ in $\mathcal{A}$. We let $\hat{\rho}$ be the 0/1 vector over $\mathrm{var}(\mathcal{A})$ satisfying the following conditions:*

1. *For each $x_{u,a} \in \mathcal{X}$, $\hat{\rho}(x_{u,a}) = 1$ iff $\lambda(u) = a$.*
2. *For each $y_{u,q} \in \mathcal{Y}$, $\hat{\rho}(y_{u,q}) = 1$ iff $\rho(u) = q$.*
3. *For each $z_{u,\delta}$ in $\mathcal{Z}$, $\hat{\rho}(z_{u,\delta}) = 1$ iff $\delta = (\rho(u_1), \ldots, \rho(u_s), \lambda(u), \rho(u))$, where $u_1, \ldots, u_s$ are the children of u.*

Let τ be a term in $\mathcal{L}(\mathcal{A})$, and ρ be a trace for τ in $\mathcal{A}$. Then by restricting the vector $\hat{\rho}$ to the variables in $\mathcal{X}$, we obtain the vector $\hat{\tau}$ associated with τ. This implies that the polytope

$$\mathrm{P}_{ex}(\mathcal{A}) = \mathrm{conv}(\{\hat{\rho} | \rho \text{ is an accepting trace in } \mathcal{A}\}) \quad (2)$$

defined as the convex-hull of vectors corresponding to *accepting* traces in $\mathcal{A}$ is an extended formulation of the polytope $P(\mathcal{A})$. Note that $\mathcal{Y}$ has $|Q|$ variables, while the set $\mathcal{Z}$ has $|\Delta|$ variables. Therefore, besides the main variables in $\mathcal{X}$, we are using $|Q| + |\Delta|$ extension variables to define $\mathrm{P}_{ex}(\mathcal{A})$.

Next, we will show that $\mathrm{P}_{ex}(\mathcal{A})$ can be defined as the set of solutions of a system of linear inequalities $\mathfrak{L}_{\mathcal{A}}$ containing $O(|Q|+|T|\cdot|\Sigma|)$ linear constraints. The first set of constraints ensure that all variables in a solution should assume values between 0 and 1. More specifically, for each main variable $x_{u,a} \in \mathcal{X}$, each state variable $y_{u,q} \in \mathcal{Y}$, and each transition variable $z_{u,\delta} \in \mathcal{Z}$,

$$0 \leq x_{u,a} \leq 1 \quad 0 \leq y_{u,q} \leq 1 \quad 0 \leq z_{u,\delta} \leq 1 \quad (3)$$

Let $r = r(T)$ be the root of T. The following constraints ensure that in an integral solution, there is exactly one final state $q \in F$ for which variable $y_{r,q}$ is set to 1. Additionally, in such a solution, for each state $q \in Q_r \backslash F$ the variable $y_{r,q}$ is set to 0.

$$\sum_{q \in F} y_{r,q} = 1 \quad (4)$$

$$\forall q \in Q_r \backslash F : \; y_{r,q} = 0 \quad (5)$$

The next set of constraints ensure that for each node $u \in N(T)$, and each state $q \in Q_u$, for any integral solution that sets variable $y_{u,q}$ to 1 there is some transition $\delta \in \Delta_u$ with consequent q such that $z_{u,\delta}$ is set to 1.

$$\forall u \in N(T), \forall q \in Q_u : \sum_{\delta, \mathtt{cnq}(\delta) = q} z_{u,\delta} = y_{u,q} \quad (6)$$

The next set of constraints ensure that for each node $u \in N(T)$ distinct from the root r, and each state $q \in Q$, for any integral solution that sets variable $y_{u,q}$ to 1 there is some transition $z \in \Delta_{\mathtt{par}(u)}$ having q as an antecedent such that $z_{u,\delta}$ is set to 1. Here, $\mathtt{par}(u)$ denotes the parent of u.

$$\forall u \in N(T) \backslash \{r\}, \forall q \in Q_u : \sum_{\delta, q \in \mathtt{ant}(\delta)} z_{\mathtt{par}(u),\delta} = y_{u,q} \quad (7)$$

Finally, the next set of equations ensure that if an integral solution sets a main variable $x_{u,a} \in \mathcal{X}$ to 1, then there is some transition $\delta \in \Delta_u$ with $\mathtt{symb}(\delta) = a$ such that the transition variable $z_{u,\delta}$ is also set to 1.

$$\forall u \in N(T), \forall a \in \Sigma : \sum_{\delta, \mathtt{symb}(\delta) = a} z_{u,\delta} = x_{u,a} \quad (8)$$

Theorem 5 is a direct consequence of the following theorem, stating that the set of inequalities $\mathfrak{L}_{\mathcal{A}}$ defines the polytope $\mathrm{P}_{ex}(\mathcal{A})$.

Theorem 8. *Let $\mathcal{A} = (\Sigma, Q, F, \Delta)$ be a T-shaped tree automaton. A vector μ belongs to $\mathrm{P}_{ex}(\mathcal{A})$ if and only if μ satisfy all inequalities in $\mathfrak{L}_{\mathcal{A}}$.*

The proof of Theorem 8 is carried out in detail in the full version of this work. The proof is split into three parts. In the first part, we show that for each trace ρ in $\mathcal{A}$, the 0/1- vector $\hat{\rho}$ satisfies the system of linear inequalities $\mathfrak{L}_{\mathcal{A}}$. This implies that the polytope $\mathrm{P}_{ex}(\mathcal{A})$ is contained in the polytope $P(\mathfrak{L}_{\mathcal{A}})$ defined by $\mathfrak{L}_{\mathcal{A}}$. In the second part, we show that if μ is an integral vector satisfying all inequalities of $\mathfrak{L}_{\mathcal{A}}$, then there is some trace ρ of $\mathcal{A}$ such that $\hat{\rho} = \mu$. This shows that at least the integral solutions of $\mathfrak{L}_{\mathcal{A}}$ are contained in $\mathrm{P}_{ex}(\mathcal{A})$. Finally, in the third part, we show that any vertex of the polytope defined by $\mathfrak{L}_{\mathcal{A}}$ is a $0/1$-vector. Therefore, since $0/1$-vectors cannot be convex combinations of other $0/1$ vectors, we have that the vertices of $P(\mathfrak{L}_{\mathcal{A}})$ are precisely the $0/1$-vectors satisfying all constraints in $\mathfrak{L}_{\mathcal{A}}$. This shows that $P(\mathfrak{L}_{\mathcal{A}}) = \mathrm{P}_{ex}(\mathcal{A})$, and concludes the proof of Theorem 8.

5 Witness Trees and Solution Polytopes

In this section, we let Π be a vertex subset problem, $\mathtt{D}$ be a DP-core solving Π, G be a graph and $\mathcal{T}$ be a tree decomposition of G.

Definition 9. *A $\mathtt{D}$-witness-tree for $\mathcal{T}$ is a function $W : N(T) \to \{0,1\}^*$ satisfying the following properties:*

1. $\mathrm{Final}(W(r(T))) = 1$,
2. $W(u) \in \mathrm{Leaf}$ *if u is a leaf node,*
3. *If u is a node with a single child u',*
   (a) $W(u) \in \mathrm{IntroVertex}(v, W(u'))$ *if vertex v is introduced at node u,*
   (b) $W(u) \in \mathrm{ForgetVertex}(v, W(u'))$ *if vertex v is forgotten at node u,*
   (c) $W(u) \in \mathrm{IntroEdge}(v, v', W(u'))$ *if an edge an edge e with $\mathtt{endpts}(e) = \{v, v'\}$ is introduced at node u.*
4. *If u is a node with children u' and u'', then $W(u) \in \mathrm{Join}(W(u'), W(u''))$.*

Intuitively, a $\mathtt{D}$-witness-tree is a certificate that the tree decomposition $\mathcal{T}$ is accepted by $\mathtt{D}$. One pertinent question is whether one can devise a way of extracting a solution for Π in G from a witness tree.

Definition 10. *An* abstract membership function *is any function of type $\mu : \mathbb{N} \times \{0,1\}^* \to \{0,1\}$. We say that $v \in \mathbb{N}$ is a μ-member of w if $\mu(v, \mathrm{w}) = 1$.*

Let $\nu[G, \mathcal{T}] : V_G \to N(T)$ be the function that assigns to each vertex $v \in V_G$, the child of the node where v is forgotten. We denote this node by $\nu[G, \mathcal{T}](v)$. This function is well defined, since in a nice edge-introducing tree decomposition, for each vertex v there is exactly one node u where v is forgotten. We can extract a subset of vertices from a witness tree as follows.

$$\mathbf{X}(G, \mathcal{T}, W, \mu) = \{v \in V_G \ : \ \mu(v, W(\nu[G, \mathcal{T}](v))) = 1\}.$$

Intuitively, $\mathbf{X}(G, \mathcal{T}, W, \mu)$ is the set of all vertices v of G that are μ-members of the string assigned by W to the child of the node of $\mathcal{T}$ where v is forgotten.

Definition 11. *We say that* $\mathtt{D}$ *is* solution-preserving *if there is an abstract membership function μ such that for each graph G and each nice edge-introducing tree decomposition $\mathcal{T}$ accepted by* $\mathtt{D}$*, the following conditions are satisfied:*

1. *For each witness tree W of $\mathcal{T}$, the set $\mathbf{X}(G, \mathcal{T}, W, \mu)$ belongs to $\mathrm{Sol}_\Pi(G)$.*
2. *For each solution $X \in \mathrm{Sol}_\Pi(G)$, there is a witness tree W such that $X = \mathbf{X}(G, \mathcal{T}, W, \mu)$.*

Intuitively, the first condition implies that if a solution for Π in G exists, then such a solution can be retrieved by backtracking: after inductively constructing the set $\Gamma(u)$ for each $u \in N(T)$, and determining that $\Gamma(r(T))$ is non-empty, we may construct a witness-tree W by backtracking and then apply the membership function to the string associated with the child of each forget node to extract a solution. On the other hand, Condition 2 ensures that each solution in $\mathrm{Sol}_\Pi(G)$ can be retrieved from some suitable witness-tree.

Definition 12. *Let G be a graph, $\mathcal{T} = (T, B, \xi)$ be a tree decomposition of G, and $X \subseteq V_G$. The characteristic tree of X is the T-shaped term $\chi[G, \mathcal{T}, X]$ where for each $u \in N(T)$, $\chi[G, \mathcal{T}, X](u) = 1$ if there is some $v \in X$ with $\nu[G, \mathcal{T}](v) = u$, and $\chi[G, \mathcal{T}, X](u) = 0$, otherwise.*

Given a problem Π, and a graph G, we let

$$\Phi_\Pi(G, \mathcal{T}) = \{\chi[G, \mathcal{T}, X] \ : \ X \in \mathrm{Sol}_\Pi(G)\}$$

be the set of characteristic trees associated with solutions of G. The next theorem establishes a connection between DP-cores deciding a problem Π and T-shaped tree automata accepting the set of characteristic trees of solutions of a graph G.

Theorem 13. *Let Π be a vertex subset problem and $\mathtt{D}$ be a solution-preserving DP-core of table-complexity $\alpha(k, n)$ solving Π. Let G be a graph and $\mathcal{T} = (T, B, \xi)$ be a tree decomposition of G of width k. Then, there is a T-shaped tree automaton $\mathcal{A}$ of width $\alpha(k, n)$ accepting the tree language $\Phi_\Pi(G, \mathcal{T})$.*

As a direct consequence of Corollary 6 and Theorem 13 we have our main result establishing an upper bound extension complexity in terms of the complexity of a DP-core.

Theorem 14. *Let Π be a vertex subset problem and $\mathtt{D}$ be a solution-preserving DP-core of table-complexity $\alpha(k, n)$ solving Π. For each n-vertex graph G of treewidth k, the extension complexity of $P_\Pi(G)$ is at most $O(\alpha(k, n) \cdot n)$.*

6 Upper Bounds for Extended Formulations

A wide variety of vertex subset problems can be solved dynamic programming algorithms operating on tree decompositions. In many cases, such algorithms can be formalized using solution-preserving dynamic programming cores. In this section we show how upper bounds on the table-complexity of classical dynamic programming algorithms parameterized by treewidth can be translated into parameterized upper bounds on the extension complexity of polytopes associated with several well studied combinatorial problems on graphs. It turns out that for several of these problems [Lokshtanov *et al.*, 2011] the upper bounds are asymptotically optimal under the exponential time hypothesis (ETH) [Impagliazzo and Paturi, 2001] the strong exponential time hypothesis (SETH) [Impagliazzo and Paturi, 2001], or related conjectures [Calabro *et al.*, 2009; Carmosino *et al.*, 2016].

6.1 Vertex Problems

We say that a vertex subset problem Π has extension complexity $f(k, n)$ on n-vertex graphs of treewidth at most k if for each n-vertex graph G of treewidth at most k, the polytope $P_\Pi(G)$ has extension complexity at most $f(k, n)$. On the other hand, we say that Π has no extended formulation with $g(k, n)$ inequalities if no polytope P' that is an extended formulation of $P_\Pi(G)$ can be defined with $g(k, n)$ inequalities or less.

Theorem 15. *The problem* IndependentSet$_\ell$ *has extension complexity $2^k n^{O(1)}$ on n-vertex graphs of treewidth at most k. Under ETH, this problem has no extended formulation with $2^{o(k)} n^{O(1)}$ inequalities.*

Let DominatingSet$_\ell$ be the problem consisting of all pairs of the form (G, X) where G is a graph and X is a dominating set in G of size at most ℓ. That is, each vertex of G is either in X or connected to some vertex in X.

Theorem 16. *The problem* DominatingSet$_\ell$ *has extension complexity $3^k n^{O(1)}$ on n-vertex graphs of treewidth at most k. Under ETH, this problem has no extended formulation with $2^{o(k)} n^{O(1)}$ inequalities.*

Interestingly, the two results above show that our general conversion from solution-preserving DP-cores to extended formulations is optimal in general when it comes to the dependency on the table-complexity of the DP-algorithm.

Theorem 17. *Under ETH, the complexity $O(\alpha(k, n) \cdot n)$ in Theorem 14 cannot be improved to $\alpha(k, n)^{o(1)} \cdot n^{O(1)}$.*

6.2 Edge Problems

Although we have stated our main theorems in terms of vertex subset problems, analogous results also hold for edge subset problems.

Definition 18 (Edge Subset Problem)**.** *An* edge subset problem *is a subset $\Pi \subset$ Graphs $\times \mathcal{P}_{fin}(\mathbb{N})$ satisfying the following conditions:*

1. *$X \subseteq E_G$, and*
2. *for each graph H isomorphic to G, and each isomorphism $\phi = (\phi_1, \phi_2)$ from G to H, $(H, \phi_2(X)) \in \Pi$.*

Given an edge subset problem Π, and instance (G, X) of Π, the set of $\mathrm{Sol}_\Pi(G)$ of all solutions of G is defined as in Equation 1.

Let HAMILTONIANCYCLE be the edge subset problem consisting of all pairs of the form (G, X) where G is a graph and X is the subset of edges of some Hamiltonian cycle. This problem can be solved by a solution-preserving DP-core of table-complexity $2^{O(k \log k)} \cdot n^{O(1)}$ on n-vertex graphs of treewidth at most k.

Theorem 19. *The problem* HAMILTONIANCYCLE *has extension complexity* $2^{O(k \log k)} \cdot n^{O(1)}$ *on* n*-vertex graphs of treewidth at most* k*. Under ETH, this problem has no extended formulation with* $2^{o(k)} n^{O(1)}$ *inequalities.*

Interestingly, HAMILTONIANCYCLE has a (non-solution-preserving) DP-core whose table-complexity, $2^{O(k)} \cdot n^{O(1)}$, matches the conditional lower bound. Although we can still extract a linear system of inequalities of size $2^{O(k)} \cdot n^{O(1)}$ from the specification of this DP-core, not all Hamiltonian cycles in the corresponding graph are guaranteed to occur extremal solutions of this system of inequalities.

Let G be a graph and (A, B) be a partition of the vertex set of G. The cut-set of this partition is the set of all edges with one endpoint in A and another endpoint in B. We let $\textsc{Cut}_\ell$ be the edge problem consisting of all pairs of the form (G, X) where G is a graph and X is a cut-set in G of size at least ℓ.

Theorem 20. *The problem* $\textsc{Cut}_\ell$ *has extension complexity* $2^k n^{O(1)}$*. Under ETH, this problem has no extended formulation with* $2^{o(k)} n^{O(1)}$ *inequalities.*

6.3 Tuple Problems

Vertex and edge problems may be generalized straightforwardly to the setting where solutions are tuples containing sets of vertices, sets of edges, or both. For example, if a solution is a d-tuple $X = (X_1, X_2, \ldots, X_d)$ of subsets of vertices, then the vector corresponding to X is the $0/1$-matrix $\hat{X}$ with d rows and $|V_G|$ columns where for each $i \in [d]$ and $v \in V_G$, $\hat{X}_{iv} = 1$ if and only if vertex v belongs to the set X_i. The solution polytope of such a problem is the convex hull of all such matrices. One prominent example of tuple of vertex-subsets problem is the d-COLORING problem, consisting of all pairs of the form $(G, (X_1, \ldots, X_d))$ where G is a graph and $(X_1, \ldots, X_d)$ is a partition of the vertex set of G such that each X_i is an independent set in G.

Theorem 21. *The problem* d-COLORING *has extension complexity* $d^k n^{O(1)}$ *on* n*-vertex graphs of treewidth at most* k*. Under ETH, this problem has no extended formulation with* $d^{o(k)} n^{O(1)}$ *inequalities.*

7 Lower Bounds from EC

Our main theorem states that solution-preserving dynamic programming algorithms with small table-complexity can be translated into extended formulations with a small number of inequalities. Interestingly, this translation allows us to obtain unconditional lower bounds on the table-complexity of solution-preserving DP-cores for a given problem from unconditional lower bounds obtained in the context of the theory of extended formulations. For instance, it has been shown in [Göös *et al.*, 2018b] that there is a suitable sequence of graphs $\{G_n\}_{n \in \mathbb{N}}$ and suitable $\ell(n) \in O(n)$ such that for each $n \in \mathbb{N}$, the graph G_n has n vertices and the solution polytope of the problem $\textsc{IndependentSet}_{\ell(n)}$ has extension complexity $2^{\Omega(n/\log n)}$ on G_n. This implies the following unconditional parameterized lower bound on the table-complexity of solution-preserving DP-cores solving the independent set problem.

Theorem 22. *The problem* $\textsc{IndependentSet}_\ell$ *has no solution-preserving DP-core of table-complexity* $2^{o(k/log k)} \cdot n^{O(1)}$ *on* n*-vertex graphs of treewidth at most* k*.*

8 Conclusion

In this work, we introduced a general framework to obtain parameterized upper bounds on the extension complexity of the solution polytopes of a wide variety of combinatorial problems in terms of the table-complexity of solution-preserving dynamic programming algorithms. In many cases, the extension complexity obtained using our method is the best one can get (under ETH).

Nevertheless, for some combinatorial problems, such as HAMILTONIANCYCLE and other connectivity problems, the fastest known dynamic programming algorithm parameterized by treewidth [Bodlaender *et al.*, 2015] is not solution preserving. In these cases, our result still yields a polytope associated with the convex-hull of all solutions that can be retrieved by backtracking from the DP tables constructed by the algorithm. The crucial difference is that this polytope does not depend only on the graph, but also on the input tree decomposition and on the particularities of the DP-core.

Another point that is worth mentioning is that our upper bounds on the extension complexity of polytopes in terms of the table-complexity of DP-cores does not depend on the actual computational complexity of constructing the tables. Only the table sizes matter. In this sense, it is not possible to generalize our results to arbitrary non-solution preserving DP-cores without making further assumptions. The reason is that any vertex subset problem has a trivial (non-solution preserving) DP-core of table-complexity 1. This DP-core non-deterministically guesses a solution, and when processing each node, keeps only the partial solution corresponding to the restriction of the guessed solution to vertices belonging to bags up to that node.

Acknowledgments

We acknowledge support from the Research Council of Norway (grant numbers 326537 and 288761).

References

[Aboulker *et al.*, 2019] Pierre Aboulker, Samuel Fiorini, Tony Huynh, Marco Macchia, and Johanna Seif. Extension complexity of the correlation polytope. *Operations Research Letters*, 47(1):47–51, 2019.

[Aprile *et al.*, 2017] Manuel Aprile, Yuri Faenza, Samuel Fiorini, Tony Huynh, and Marco Macchia. Extension complexity of stable set polytopes of bipartite graphs. In

Hans L. Bodlaender and Gerhard J. Woeginger, editors, *Graph-Theoretic Concepts in Computer Science*, pages 75–87, Cham, 2017. Springer International Publishing.

[Baste *et al.*, 2022] Julien Baste, Michael R. Fellows, Lars Jaffke, Tomáš Masařík, Mateus de Oliveira Oliveira, Geevarghese Philip, and Frances A. Rosamond. Diversity of solutions: An exploration through the lens of fixed-parameter tractability theory. *Artificial Intelligence*, 303:103644, 2022.

[Bodlaender *et al.*, 2015] Hans L Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Information and Computation*, 243:86–111, 2015.

[Buchanan and Butenko, 2015] Austin Buchanan and Sergiy Butenko. Tight extended formulations for independent set. *Manuscript available at Optimization-Online*, 2015.

[Calabro *et al.*, 2009] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In *Proc. of the 4th International Workshop on Parameterized and Exact Computation*, pages 75–85. Springer, 2009.

[Carmosino *et al.*, 2016] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proc. of the 7th ACM Conference on Innovations in Theoretical Computer Science (ITCS 2016)*, pages 261–270. ACM, 2016.

[Courcelle *et al.*, 2000] Bruno Courcelle, Johann A Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000.

[de Oliveira Oliveira and Vadiee, 2023] Mateus de Oliveira Oliveira and Farhad Vadiee. From width-based model checking to width-based automated theorem proving. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023*, pages 6297–6304. AAAI Press, 2023.

[De Simone, 1990] C. De Simone. The cut polytope and the boolean quadric polytope. *Discrete Math.*, 79(1):71–75, jan 1990.

[Fiorini *et al.*, 2015] Samuel Fiorini, Serge Massar, Sebastian Pokutta, Hans Raj Tiwary, and Ronald De Wolf. Exponential lower bounds for polytopes in combinatorial optimization. *Journal of the ACM (JACM)*, 62(2):17, 2015.

[Göös *et al.*, 2018a] Mika Göös, Rahul Jain, and Thomas Watson. Extension complexity of independent set polytopes. *SIAM Journal on Computing*, 47(1):241–269, 2018.

[Göös *et al.*, 2018b] Mika Göös, Rahul Jain, and Thomas Watson. Extension complexity of independent set polytopes. *SIAM J. Comput.*, 47(1):241–269, 2018.

[Hu and Laurent, 2019] Hao Hu and Monique Laurent. On the linear extension complexity of stable set polytopes for perfect graphs. *European Journal of Combinatorics*, 80:247–260, 2019. Special Issue in Memory of Michel Marie Deza.

[Impagliazzo and Paturi, 2001] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

[Kloks, 1994] Ton Kloks. *Treewidth: Computations and Approximations*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1 edition, 1994. Springer Book Archive.

[Kolman *et al.*, 2020] Petr Kolman, Martin Koutecký, and Hans Raj Tiwary. Extension complexity, mso logic, and treewidth. *Discrete Mathematics and Theoretical Computer Science*, 22(4), 2020. Distributed under a Creative Commons Attribution 4.0 International License. A preliminary version of this paper appeared at the 15th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2016).

[Lokshtanov *et al.*, 2011] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bull. EATCS*, 105:41–72, 2011.

[Lokshtanov *et al.*, 2018] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. *ACM Transactions on Algorithms (TALG)*, 14(2):1–30, 2018.

[Marx, 2007] Dániel Marx. Can you beat treewidth? In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 169–179. IEEE, 2007.

[Seif, 2019] Johanna Seif. Bounding techniques for extension complexity. Master's thesis, École normale supérieure de Lyon, 2019. Accessed: 2023-04-01.

[Yannakakis, 1988] Mihalis Yannakakis. Expressing combinatorial optimization problems by linear programs. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 223–228, 1988.