Courcelle's Theorem for Lipschitz Continuity

Tatsuya Gima
Hokkaido University, Sapporo, Japan

Soh Kumabe
CyberAgent, Tokyo, Japan

Yuichi Yoshida
National Institute of Informatics, Tokyo, Japan

Abstract

Lipschitz continuity of algorithms, introduced by Kumabe and Yoshida (FOCS'23), measures the stability of an algorithm against small input perturbations. Algorithms with small Lipschitz continuity are desirable, as they ensure reliable decision-making and reproducible scientific research. Several studies have proposed Lipschitz continuous algorithms for various combinatorial optimization problems, but these algorithms are problem-specific, requiring a separate design for each problem.

To address this issue, we provide the first algorithmic meta-theorem in the field of Lipschitz continuous algorithms. Our result can be seen as a Lipschitz continuous analogue of Courcelle's theorem, which offers Lipschitz continuous algorithms for problems on bounded-treewidth graphs. Specifically, we consider the problem of finding a vertex set in a graph that maximizes or minimizes the total weight, subject to constraints expressed in monadic second-order logic (MSO_2). We show that for any $\varepsilon > 0$, there exists a $(1 \pm \varepsilon)$-approximation algorithm for the problem with a polylogarithmic Lipschitz constant on bounded treewidth graphs. On such graphs, our result outperforms most existing Lipschitz continuous algorithms in terms of approximability and/or Lipschitz continuity. Further, we provide similar results for problems on bounded-clique-width graphs subject to constraints expressed in MSO_1. Additionally, we construct a Lipschitz continuous version of Baker's decomposition using our meta-theorem as a subroutine.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability; Theory of computation → Approximation algorithms analysis

Keywords and phrases Fixed-Parameter Tractability, Algorithmic Meta-Theorem, Lipschitz Continuity

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.11

Related Version *Full Version*: http://arxiv.org/abs/2506.21118

Funding *Tatsuya Gima*: JSPS KAKENHI Grant Numbers JP24K23847, JP25K03077
Yuichi Yoshida: JSPS KAKENHI Grant Number JP24K02903

1 Introduction

Lipschitz continuity of algorithms, introduced by Kumabe and Yoshida [14], is a measure of an algorithm's stability in response to errors or small perturbations in the input for weighted optimization problems. Roughly speaking, it is the maximum ratio of the (weighted) Hamming distance between the outputs of an algorithm for two different weights to the ℓ_1 distance between those weights (see Section 2.4 for the precise definition). It is desirable for algorithms to have small Lipschitz constants, as large constants can undermine the reliability of decision-making and the reproducibility of scientific research.

Since its introduction, Lipschitz continuous algorithms have been proposed for various optimization problems [14, 16]. However, we need to design a different algorithm and analyze its Lipschitz continuity for each problem, which can be impractical. To address this limitation, we present an algorithmic meta-theorem for Lipschitz continuous algorithms, which can be seen as Lipschitz continuity analogue of celebrated Courcelle's theorem [3].

To present our results, we first introduce some notation. Let $G = (V, E)$ be a graph of n vertices with treewidth tw, $\varphi(X)$ be an MSO_2 formula with a free vertex set variable X (see Section 2.3 for details about logic), and $w \in \mathbb{R}^V_{\geq 0}$ be a weight vector. We consider the problem of finding a vertex subset $X \subseteq V$ such that $G \models \varphi(X)$ and $w(X)$ is maximized, which we call the *MSO_2 maximization problem.* We also consider the problem of finding X such that $w(X)$ is minimized, which we call the *MSO_2 minimization problem.* Our main results are the following.

▶ **Theorem 1.** *For any $\varepsilon \in (0, 1]$, there is a $(1-\varepsilon)$-approximation algorithm for the MSO_2 maximization problem with Lipschitz constant $O\left((f(\mathrm{tw}, |\varphi|) + \log \varepsilon^{-1} + \log\log n)\varepsilon^{-1}\log^2 n\right)$, where f is some computable function. The time complexity is bounded by $O\left(f(\mathrm{tw}, |\varphi|)n\right)$.*

▶ **Theorem 2.** *For any $\varepsilon \in (0, 1]$, there is a $(1+\varepsilon)$-approximation algorithm for the MSO_2 minimization problem with Lipschitz constant $O\left((f(\mathrm{tw}, |\varphi|) + \log \varepsilon^{-1} + \log\log n)\varepsilon^{-1}\log^2 n\right)$, where f is some computable function. The time complexity is bounded by $O\left(f(\mathrm{tw}, |\varphi|)n\right)$.*

We note that the trivial upper bound on Lipschitz constant is n [16]; therefore the bounds in the theorems above are significantly smaller for fixed tw and φ. We also remark that our meta-theorems yield randomized approximation algorithms. This is necessary since, for most problems, it is known that exact or deterministic algorithms cannot be Lipschitz continuous [14].

When the treewidth of the input graph is bounded by a constant, Theorems 1 and 2 provide algorithms that outperform existing algorithms in terms of approximability and/or Lipschitz continuity:

- For the minimum weight vertex cover problem, Theorem 2 yields an algorithm with a better approximation ratio than the previous 2-approximation algorithm with Lipschitz constant 4 [16], at the polylogarithmic sacrifice of the Lipschitz constant.
- For the minimum weight feedback vertex set problem, Theorem 2 outperforms the previous $O(\log n)$-approximation algorithm with Lipschitz constant $O(\sqrt{n}\log^{3/2} n)$ [16] in terms of both approximability and Lipschitz continuity.
- For the maximum weight matching problem[1], Theorem 1 yields an algorithm with a better approximation ratio than the previous $\left(\frac{1}{8} - \varepsilon\right)$-approximation algorithm with Lipschitz constant $O(\varepsilon^{-1})$ [14], at the polylogarithmic sacrifice of the Lipschitz constant.
- For the shortest path problem, Theorem 2 slightly improves the Lipschitz continuity compared to the previous $(1-\varepsilon)$-approximation algorithm with Lipschitz constant $O(\varepsilon^{-1}\log^3 n)$ [14], without losing approximability.

For a fixed φ, by explicitly specifying the transitions in the dynamic programming within the algorithm, we can provide more precise bounds on the function f that appears in Theorems 1 and 2. In particular, considering the case that φ is an MSO_2 formula representing the independent set constraint, we have the following.

▶ **Theorem 3.** *For any $\varepsilon \in (0, 1]$, there is a $(1-\varepsilon)$-approximation algorithm for the maximum weight independent set problem with Lipschitz constant $O\left((\mathrm{tw} + \log \varepsilon^{-1} + \log\log n)\varepsilon^{-1}\log^2 n\right)$. The time complexity is bounded by $2^{O(\mathrm{tw})}n$.*

This result is surprising as the dependence of tw on the Lipschitz constant is subexponential and, even more remarkably, linear. We note that through similar arguments, analogous results hold for several other problems, such as the minimum weight vertex cover problem and the minimum weight dominating set problem.

---

[1] Although the output of this problem is an edge set rather than a vertex set, this problem can be expressed by an MSO_2 formula on a new graph $G' = (V \cup E, \{(v, e) \colon e \text{ is incident to } v\})$

We further demonstrate that Theorems 1 and 2 lead to Lipschitz continuous version of Baker's technique [1]. As a representative example, we consider the maximum weight independent set problem on planar graphs, where a vertex subset X is an *independent set* of a graph G if no two vertices in X are adjacent in G. We prove the following.

▶ **Theorem 4.** *For any $\varepsilon \in (0,1]$, there is a $(1-\varepsilon)$-approximation algorithm for the maximum weight independent set problem on planar graphs with Lipschitz constant $O\left((\varepsilon^{-1} + \log\log n)\varepsilon^{-1}\log^2 n\right)$. The time complexity is bounded by $2^{O(\varepsilon^{-1})}n$.*

Using similar algorithms, Lipschitz continuous PTASes can be obtained for many problems, including the minimum weight vertex cover problem and the minimum weight dominating set problem.

As a lower bound result, we prove that the Lipschitz constant of any $(1-\varepsilon)$-approximation algorithm for some MSO_2 maximization problem is large on general graphs, which justifies considering the MSO_2 maximization problem on a restricted graph class. Specifically, we consider the *max ones problem* [6, 9], where the instance is a 3CNF formula over a variable set X and a weight function $w : X \to \mathbb{R}_{\geq 0}$, and the goal is to find a satisfying assignment $\sigma : X \to \{0,1\}$ that maximizes the weight $\sum_{x\in\sigma^{-1}(1)} w(x)$. It is easy to see that there exists a fixed MSO_2 formula $\varphi(X)$ such that the max ones problem can be reduced to a problem on bipartite graphs, where the task is to find a vertex set X that maximizes the weight $\sum_{x\in X} w(x)$ subject to φ. We show the following.

▶ **Theorem 5.** *There exist $\varepsilon, \delta > 0$ such that any $(1-\varepsilon)$-approximation algorithm for the max ones problem has Lipschitz constant $\Omega(n^\delta)$, where n is the number of variables.*

We note that it is possible to construct an algorithm for the max ones problem with a Lipschitz constant similar to that in Theorem 3, where tw denotes the treewidth of the "incidence graph" of the 3CNF instance (see Section 4.2 for details). Since tw is upper bounded by n for any instance, Theorem 5 implies that the dependency of the Lipschitz constant on tw cannot be improved to $\mathrm{tw}^{o(1)}$.

We also prove analogous meta-theorems when parameterizing by clique-width. Let an *MSO_1 maximization (resp., minimization) problem* be the variant of an MSO_2 problem in which the formula φ is restricted to MSO_1. Denote by cw the clique-width of the graph G. We then obtain the following.

▶ **Theorem 6.** *For any $\varepsilon \in (0,1]$, there is a $(1-\varepsilon)$-approximation algorithm for the MSO_1 maximization problem with Lipschitz constant $O\left((f(\mathrm{cw}, |\varphi|) + \log\varepsilon^{-1} + \log\log n)\varepsilon^{-1}\log^2 n\right)$, where f is some computable function. The time complexity is bounded by $O\left(f(\mathrm{cw}, |\varphi|)n\right)$.*

▶ **Theorem 7.** *For any $\varepsilon \in (0,1]$, there is a $(1+\varepsilon)$-approximation algorithm for the MSO_1 minimization problem with Lipschitz constant $O\left((f(\mathrm{cw}, |\varphi|) + \log\varepsilon^{-1} + \log\log n)\varepsilon^{-1}\log^2 n\right)$, where f is some computable function. The time complexity is bounded by $O\left(f(\mathrm{cw}, |\varphi|)n\right)$.*

We note that the functions f in Theorems 6 and 7 are much larger than those in Theorems 1 and 2. In particular, the bounds for Theorems 3 and 4 do not follow from Theorem 6.

1.1 Technical Overview

Now we provide a technical overview of our framework. Since the arguments for clique-width are similar, we focus here on the treewidth results and omit the clique-width case.

For simplicity, here we consider the maximum weight independent set problem on a full binary tree (a rooted tree in which every vertex has exactly 0 or 2 children). This corresponds to the case where $\varphi(X) = \forall x \forall y((x \in X \wedge y \in X) \to \neg\mathsf{adj}(x,y))$. Let $w \in \mathbb{R}_{\geq 0}^V$ be the weight vector.

If we do not care about Lipschitzness, this problem can exactly be solved by the following algorithm. For each vertex $v \in V$, define $\mathsf{DP}[v][0]$ to be an independent set in the subtree rooted at v that does not include v with the maximum weight. Similarly, define $\mathsf{DP}[v][1]$ to be an independent set in the subtree rooted at v with the maximum weight. If v is a leaf, then $\mathsf{DP}[v][0] = \emptyset$ and $\mathsf{DP}[v][1] = \{v\}$. Otherwise, let u_1 and u_2 be the two children of v. We have $\mathsf{DP}[v][0] = \mathsf{DP}[u_1][1] \cup \mathsf{DP}[u_2][1]$, and $\mathsf{DP}[v][1]$ is the one with the larger weight of $X_{v,0} := \mathsf{DP}[u_1][0] \cup \mathsf{DP}[u_2][0] \cup \{v\}$ and $X_{v,1} := \mathsf{DP}[u_1][1] \cup \mathsf{DP}[u_2][1]$. By performing this dynamic programming in a bottom-up manner, the problem can be solved exactly.

However, this algorithm is not Lipschitz continuous. This is because when we compute $\mathsf{DP}[v][1]$, which of $X_{v,0}$ or $X_{v,1}$ has a larger weight is affected by small changes in the weights. To address this issue, we use the exponential mechanism [7]. Specifically, for some constant $c > 0$, we select $X_{v,i}$ with probability proportional to $\exp(c \cdot w(X_{v,i}))$. This approach makes the algorithm Lipschitz continuous, with only a slight sacrifice in the approximation ratio. Specifically, we can prove that by appropriately choosing c for a given $\varepsilon' \in (0, 1]$, the increase in the Lipschitz constant can be bounded by $\tilde{O}(\varepsilon'^{-1})$ by reducing the approximation guarantee by a factor of $1 - \varepsilon'$ at each vertex where the exponential mechanism is applied.

While this approach makes the algorithm Lipschitz continuous, the Lipschitz constant is still too large when the height h of the tree is large. Specifically, to achieve an approximation guarantee of $1 - \varepsilon$, we need to set $\varepsilon' = \frac{\varepsilon}{h}$, leading to a Lipschitz constant of $h \cdot \tilde{O}(\varepsilon'^{-1}) = \tilde{O}(\varepsilon^{-1}h^2)$. This is larger than the trivial bound n when $h = O(n)$. We resolve this issue by using the fact that any tree has a tree decomposition of width 5 and height $O(\log n)$ [2]. By performing dynamic programming with the exponential mechanism on this tree decomposition, we can obtain $(1-\varepsilon)$-approximation algorithm with Lipschitz constant $\tilde{O}(\varepsilon^{-1})$. This argument can be naturally extended to the case where G is a bounded treewidth graph. By following the proof of Courcelle's Theorem [3, 5], we further extend this argument to the case where φ is a general MSO_2 formula with a free vertex variable.

We prove Theorem 5 by leveraging a lower bound on a related notion of sensitivity for the maximum cut problem [8], where sensitivity measures the change in the output with respect to the Hamming distance under edge deletions [21]. Specifically, we reduce the maximum cut problem, where stability is defined with respect to edge deletions, to the max ones problem, where stability is measured with respect to weight changes.

1.2 Related Work

Lipschitz continuity of discrete algorithms is defined by Kumabe and Yoshida [14], and they provided Lipschitz continuous approximation algorithms for the minimum spanning tree, shortest path, and maximum weight matching problems. In a subsequent work [16], they further provided such algorithms for the minimum weight vertex cover, minimum weight set cover, and minimum weight feedback vertex set problems. In another work [15], they defined Lipschitz continuity for allocations in cooperative games and provided Lipschitz continuous allocation schemes for the matching game and the minimum spanning tree game.

A variant known as *pointwise Lipschitz continuity* has also been studied, which is defined using the unweighted Hamming distance instead of the weighted Hamming distance. Kumabe and Yoshida [14] defined pointwise Lipschitz continuity and provided pointwise Lipschitz continuous algorithms for the minimum spanning tree and maximum weight bipartite matching problems. Liu et al. [18] proposed the *proximal gradient method* as a general technique for solving LP relaxations stably. Using this, they provided pointwise Lipschitz continuous approximation algorithms for the minimum vertex (S, T)-cut, densest subgraph, maximum weight $(b\text{-})$matching, and packing integer programming problems.

(Average) sensitivity, introduced by Varma and Yoshida [21] with preliminary ideas by Murai and Yoshida [19], is a notion similar to Lipschitz continuity. While Lipschitz continuity evaluates an algorithm's stability against unit changes in the input weights, (average) sensitivity evaluates an algorithm's stability against (random) deletions of elements from the input. Algorithms with small (average) sensitivity have been studied for several problems, such as maximum matching [21, 23], minimum cut [21], knapsack problem [13], Euclidean k-means [22], spectral clustering [20], and dynamic programming problems [12]. Recently, Fleming and Yoshida [8] constructed a PCP framework to prove the sensitivity lower bound for the constraint satisfaction problem.

1.3 Organization

The rest of this paper is organized as follows. In Section 2, we describe the necessary preliminaries on tree decomposition (Sections 2.1 and 2.2), logic (Section 2.3), and Lipschitz continuity (Section 2.4). In Section 3, we prove Theorem 1 by providing a Lipschitz continuous algorithm for the MSO_2 maximization problem. Since the algorithm and analysis for MSO_2 minimization are similar, we defer the proof of Theorem 2 to the full version. In Section 4, we give a more precise Lipschitzness analysis for specific formulas φ, which includes the proof of Theorem 3. Proofs of Theorems 4, 5, 6, and 7 are given in the full version.

2 Preliminaries

2.1 Tree Decomposition

Let G be a graph with n vertices. A pair $(\mathcal{B}, \mathcal{T})$ consisting of a family $\mathcal{B}$ of subsets (called *bags*) of $V(G)$ and a rooted tree $\mathcal{T}$ whose vertex set is $\mathcal{B}$ is a *(rooted) tree decomposition* of G if it satisfies the following three conditions.

- $\bigcup_{B\in\mathcal{B}} B = V(G)$.
- For each edge $e \in E(G)$, there is a bag $B \in \mathcal{B}$ such that $e \subseteq B$.
- For each vertex $v \in V(G)$, the set of bags $\{B \in \mathcal{B} \colon v \in B\}$ induces connected subgraph in T.

For a tree decomposition $(\mathcal{B}, \mathcal{T})$, we may refer to the *root node*, *leaf nodes*, and the *height* of $\mathcal{T}$ as those of $(\mathcal{B}, \mathcal{T})$, respectively. Moreover, $(\mathcal{B}, \mathcal{T})$ is binary if $\mathcal{T}$ is a binary tree. The *width* of a tree decomposition $(\mathcal{B}, \mathcal{T})$ is the maximum size of a bag in $\mathcal{B}$ minus one. The *treewidth* of G is the minimum possible width among all possible tree decompositions of G. It is known that the tree decomposition of G of width at most $2k + 1$ can be computed in $2^{O(k)}n$ time [10], where k is the treewidth of G. Moreover, any tree decomposition of G of width k can be transformed into a binary tree decomposition of width at most $3k + 2$ and height at most $O(\log n)$ [2]. Thus, we obtain the following lemma.

▶ **Lemma 8** ([2, 10])**.** *Let G be a graph with n vertices and k be the treewidth of G. Then, a binary tree decomposition of G with width $O(k)$ and height $O(\log n)$ can be computed in $2^{O(k)}n$ time.*

2.2 HR-algebra

We introduce the notions of HR-algebra[2], which is one of the algebraic definitions of tree decomposition. A *k-graph* is a tuple $G = \langle V, E, \mathsf{src}\rangle$ of a set of vertices V, a set of edges E, and a k-vector $\mathsf{src} \in (V \cup \{\mathtt{nil}\})^k$. We write the i-th element of a vector src by $\mathsf{src}(i)$. If $\mathsf{src}(i) = u$, we say that u is a *i-source*. We write the set $\{\mathsf{src}(i) : i \in [k]\} \setminus \{\mathtt{nil}\}$ by $\mathsf{src}(G)$.

Let G and H be k-graphs. The *parallel-composition* of G and H, denoted by $G \mathbin{/\!/} H$, is the graph generated by the following procedure. First, create the disjoint union of G and H, and identify the (non-$\mathtt{nil}$) vertices $\mathsf{src}_G(i)$ and $\mathsf{src}_H(i)$ for each $i \in [k]$. Finally, remove all the self-loops and multi-edges. Let B be a non-empty subset of $[k]$. The *source forgetting operation* fg_B is the function that maps a k-graph G to a k-graph G' such that $V_G = V_{G'}$, $E_G = E_{G'}$, and $\mathsf{src}_{G'}(i) = \mathtt{nil}$ if $i \in B$, and $\mathsf{src}_{G'}(i) = \mathsf{src}_G(i)$ otherwise.

▶ **Definition 9** (HR-algebra). *A* term *of a HR-algebra over k-graphs is either*

- *a constant symbol* $\mathtt{i}$, $\mathtt{ij}$, *or* $\emptyset$ *denoting a k-graph* $\langle\{v\}, \emptyset, \mathsf{src}\rangle$ *with* $\mathsf{src}(i) = v$, *a k-graph* $\langle\{u, v\}, \{\{u, v\}\}, \mathsf{src}\rangle$ *with* $\mathsf{src}(i) = u$ *and* $\mathsf{src}(j) = v$, *or the empty graph, respectively;*
- $t \mathbin{/\!/} s$ *for any terms t and s;*
- $\mathsf{fg}_B(t)$ *for any term t and any $B \subseteq [k]$.*

We may associate a term with the graphs represented by it. A term of an HR-algebra t can be decomposed into the parse tree in the usual way, called the *(rooted) HR-parse tree* of t. The *height* of a HR-parse tree is the maximum distance from the root to a leaf. It is known that tree decompositions are equivalent to HR-parse trees in the following sense.

▶ **Proposition 10** (see e.g. [4]). *The treewidth of a graph is at most k if and only if the graph can be denoted by a term of a HR-algebra over $(k + 1)$-graphs.*

Moreover, binary tree decomposition can be transformed into an HR-parse tree with approximately the same height.

▶ **Proposition 11** ($\star$). *Given a rooted binary tree decomposition $(\mathcal{B}, \mathcal{T})$ of G with width k and height h, we can compute a HR-parse tree over $(k + 1)$-graph denoting G with height $O(\log k + h)$.*

The proof of Proposition 11 is essentially the same as the proof of Proposition 10 but we give in the full version.

2.3 Monadic Second-Order Logic

We provide a slightly less formal definition of *monadic second-order (MSO) logic* over k-graphs for accessibility (see, e.g., [17] for more formal definition). An *atomic formula* is a formula of the form $s = t$, $\mathsf{adj}(s, t)$, $t \in X$, and Boolean constants `True` and `False`. Here, $\mathsf{adj}(s, t)$ is the predicate that represents whether s and t are adjacent. A *monadic second-order formula* over k-graphs is a formula built from atomic formulas, the usual Boolean connectives $\wedge, \vee, \neg, \rightarrow$, first-order quantifications $\exists x$ and $\forall x$ (quantifications over vertices), and second-order quantifications $\exists X$ and $\forall X$ (quantification over sets of vertices). The notation $\varphi(X)$ indicates that φ has exactly one free variable X. The *quantifier height* (or *quantifier rank*) of a formula φ, denoted by $|\varphi|$, is the maximum depth of quantifier nesting in φ (see e.g. [17] for a formal definition). The *counting monadic second-order (CMSO) logic*

[2] The acronym HR stands for hyperedge replacement [4].

is an expansion of MSO logic that have additional predicates $\mathsf{Card}_{m,r}(X)$ for a second-order variable X, meaning $|X| \equiv m \pmod r$. A $\mathsf{C}_r\mathsf{MSO}^q$-formula is a CMSO formula φ such that the quantifier height of φ is at most q and $r' \le r$ holds for any predicate $\mathsf{Card}_{m,r'}$ in φ. For a logical formula φ and a graph G, we write $G \models \varphi$ when graph G satisfies the property expressed by φ.

There are two variants of MSO logic: MSO_1 (or simply MSO), which is described above, allows quantification over vertices and sets of vertices only, and MSO_2 which allows quantification over edges and sets of edges as well. It is well known that, for any graph G and MSO_2-formula φ, there exists an MSO_1 formula φ' such that $G \models \varphi \iff G' \models \varphi'$, where G' is the *incidence graph*, obtained by subdividing each edge of G (with two colors meaning the original and the subdivision vertices) (see e.g. [11]). It is easy to see that the treewidth of the incidence graph is at most the treewidth of the original graph. Thus, since this paper focuses on graphs with bounded treewidth, we mainly consider MSO_1. However, all the theorems are held for MSO_2 and graphs of bounded treewidth.

We introduce some more notations. For a set V, two families $\mathcal{A} \subseteq 2^V$ and $\mathcal{B} \subseteq 2^V$ are *separated* if $A \cap B = \varnothing$ for all $A \in \mathcal{A}$, and $B \in \mathcal{B}$. We write $A \cup B$ by $A \uplus B$ if $A \cap B = \varnothing$, and $\mathcal{A} \boxtimes \mathcal{B} \coloneqq \{A \cup B : A \in \mathcal{A}, B \in \mathcal{B}\}$ if $\mathcal{A}$ and $\mathcal{B}$ are separated. For a formula $\varphi(X)$ and a graph G, the set of vertex sets satisfying φ is denoted by $\mathsf{sat}(G, \varphi)$, that is, $\mathsf{sat}(G, \varphi) = \{A : G \models \varphi(A)\}$.

The following theorem is a key of the MSO model-checking algorithm [3, 4].

▶ **Theorem 12** ([4])**.** *Let r, q, k be positive integers. Let $\varphi(X)$ be a $\mathsf{C}_r\mathsf{MSO}^q$-formula over k-graphs with a second-order free variable X. Then, the following hold.*

1. *For any $B \subseteq [k]$, there exists a $\mathsf{C}_r\mathsf{MSO}^q$-formula $\psi(X)$ such that for any k-graph G, we have*
$$\mathsf{sat}(\mathsf{fg}_B(G), \varphi) = \mathsf{sat}(G, \psi).$$
2. *There exists a family of tuples $\{\theta_i(X), \psi_i(X)\}_{i \in [p]}$ of $\mathsf{C}_r\mathsf{MSO}^q$-formulas with a free variable X such that, for any k-graphs G and H,*
$$\mathsf{sat}(G \mathbin{/\!/} H, \varphi) = \biguplus_{i \in [p]} \{S \cup P : S \in \mathsf{sat}(G, \theta_i), P \in \mathsf{sat}(H, \psi_i)\}.$$

It is known that, for any r, q, k, up to logical equivalence, there are only finitely many different $\mathsf{C}_r\mathsf{MSO}^q$-formulas $\varphi(X)$ over k-graphs (see e.g., [17]). Thus, we can obtain a linear-time algorithm, based on dynamic programming over trees, for $\mathsf{C}_r\mathsf{MSO}^q$ model checking over bounded treewidth. Courcelle and Mosbah [5] adjusted Theorem 12 to address optimization, counting, and other problems. We use a slightly modified version of the theorem of Courcelle and Mosbah.

Let $\mathsf{nosrc}(X) \equiv \bigwedge_{i \in [k]} (\mathsf{src}(i) \notin X)$, and $\varphi{\restriction_S}(X)$ denote $\varphi(X \cup S) \wedge \mathsf{nosrc}(X)$ for a set $S \subseteq \mathsf{src}(G)$.

▶ **Observation 13.** *Let G be a k-graph, and $\varphi(X)$ be a formula with a free set variable X. Then, we have $\mathsf{sat}(G, \varphi) = \biguplus_{S \subseteq \mathsf{src}(G)} \mathsf{sat}(G, \varphi{\restriction_S}) \boxtimes \{S\}$.*

The proof of Corollary 14 is given in the full version.

▶ **Corollary 14** ($\star$)**.** *Let r, q, k be positive integers. Let $\varphi(X)$ be a $\mathsf{C}_r\mathsf{MSO}^q$-formula over k-graphs with a second-order free variables X. Then, for all $S \subseteq \mathsf{src}(G)$, the following hold.*

1. *For any $B \subseteq [k]$, there exists a $\mathsf{C}_r\mathsf{MSO}^q$-formula $\psi(X)$ such that for any k-graph G, we have*
$$\mathsf{sat}(\mathsf{fg}_B(G), \varphi{\restriction_S}) = \biguplus_{S' \subseteq B} \mathsf{sat}(G, \psi{\restriction_{S \cup S'}}) \boxtimes \{S'\}.$$

2. *There exists a family of tuples* $\{\theta_i\lceil_{S_i}(X), \psi_i\lceil_{S'_i}(X)\}_{i\in[p]}$ *of* $\mathsf{C}_r\mathsf{MSO}^q$*-formulas with a free variable* X *such that, for any* k*-graphs* G *and* H*,*

$$\mathsf{sat}(G \mathbin{/\!/} H, \varphi\lceil_S) = \biguplus_{i\in[p]} \mathsf{sat}(G, \theta_i\lceil_{S_i}) \boxtimes \mathsf{sat}(H, \psi_i\lceil_{S'_i}).$$

Moreover, $S_i \cup S'_i = S$ *for all* $i \in [p]$*.*

Then, we can design an algorithm for an (C)MSO maximization (minimization) problem over graphs of bounded treewidth. For simplicity, we assume that the given graph G has no sources, that is, $\mathsf{src}(G) = \varnothing$. Let t be a term denoting G_t and $\varphi(X)$ be the (C)MSO-formula describing the constraint of the problem. We recursively compute the value $\mathsf{opt}(t', \varphi'\lceil_S) = \max\{w(A) : G \models \varphi'\lceil_S(A)\}$ for any subterm t' of t and MSO-formula $\varphi'\lceil_S$ as follows, where $w \in \mathbb{R}^V_{\geq 0}$ is the given weight vector and $w(X) = \sum_{x\in X} w_x$ for any $X \subseteq V(G)$. If t' is of the form $\mathsf{fg}_B(t'')$, then $\mathsf{opt}(\mathsf{fg}_B(t''), \varphi'\lceil_S) = \max_{S'\subseteq B}\{\mathsf{opt}(t'', \psi\lceil_{S\cup S'}) + w(S')\}$, where, ψ is the formula obtained from Corollary 14. If t' is of the form $t_1 \mathbin{/\!/} t_2$, then $\mathsf{opt}(t', \varphi'\lceil_S) = \max_{i\in[p]}\{\mathsf{opt}(t_1, \theta\lceil_{S_i}) + \mathsf{opt}(t_2, \psi\lceil_{S'_i})\}$, where, $\{\theta_i\lceil_{S_i}(X), \psi_i\lceil_{S'_i}(X)\}_{i\in[p]}$ are the formulas obtained from Corollary 14. Then, $\mathsf{opt}(t, \varphi\lceil_\varnothing)$ is the maximum value for the problem since t has no sources. In the following, for simplicity, we may omit the $\lceil_S$ notation and consider only formulas φ such that $\mathsf{sat}(G, \varphi)$ does not contain any sources.

In the above setting, the graphs are considered without any special vertex sets (called as colors) or special vertices (called as labels). However, for k-graphs with a constant number of colors and/or labels, an appropriate HR-algebra and logics can be defined, and the similar results for Corollary 14 hold (see e.g. [4, 17]).

2.4 Lipschitz Continuity

We formally define Lipschitz continuity of algorithms. As this is sufficient for this work, we only consider algorithms for vertex-weighted graph problems. The definition can be naturally extended to other settings, such as edge-weighted problems. Let $G = (V, E)$ be a graph. For vertex sets $X, X' \subseteq V$ and weight vectors $w, w' \in \mathbb{R}^V_{\geq 0}$, we define the *weighted Hamming distance* between (X, w) and (X', w') by

$$d_{\mathrm{w}}((X, w), (X', w')) := \left\| \sum_{v\in X} \mathbf{1}_v w_v - \sum_{v\in X'} \mathbf{1}_v w'_v \right\|_1 = \sum_{v\in X\cap X'} |w_v - w'_v| + \sum_{v\in X\setminus X'} w_v + \sum_{v\in X'\setminus X} w'_v,$$

where $\mathbf{1}_v \in \{0, 1\}^V$ denotes the *characteristic vector* of v, that is, the vector $\mathbf{1}_{v,u} = 1$ holds if and only if $u = v$. For two probability distributions $\mathcal{X}$ and $\mathcal{X}'$ over subsets of V, we define

$$\mathsf{EM}\left((\mathcal{X}, w), (\mathcal{X}', w')\right) := \inf_{\mathcal{D}} \mathbb{E}_{(X,X')\sim\mathcal{D}}\left[d_{\mathrm{w}}((X, w), (X', w'))\right],$$

where the minimum is taken over *couplings* of $\mathcal{X}$ and $\mathcal{X}'$, that is, distributions over pairs of sets such that its marginal distributions on the first and second coordinates are equal to $\mathcal{X}$ and $\mathcal{X}'$, respectively. Consider an algorithm $\mathcal{A}$, that takes a graph $G = (V, E)$ and a weight vector $w \in \mathbb{R}^V_{\geq 0}$ as an input and outputs a vertex subset $X \subseteq V$. We denote the output distribution of $\mathcal{A}$ for weight w as $\mathcal{A}(w)$. The *Lipschitz constant* of $\mathcal{A}$ is defined by

$$\sup_{\substack{w,w'\in\mathbb{R}_{\geq 0},\\ w\neq w'}} \frac{\mathsf{EM}\left((\mathcal{A}(w), w), (\mathcal{A}(w'), w')\right)}{\|w - w'\|_1}.$$

For two random variables $\boldsymbol{X}$ and $\boldsymbol{X}'$, the *total variation distance* between them is given as:

$$\mathsf{TV}\left(\boldsymbol{X}, \boldsymbol{X}'\right) := \inf_{\mathcal{D}} \Pr_{(X, X') \sim \mathcal{D}} \left[X \neq X'\right],$$

where the minimum is taken over couplings between $\boldsymbol{X}$ and $\boldsymbol{X}'$, that is, distributions over pairs such that its marginal distributions on the first and the second coordinates are equal to $\boldsymbol{X}$ and $\boldsymbol{X}'$, respectively. For an element $u \in V$, we use $\mathbf{1}_u \in \mathbb{R}^V_{\geq 0}$ to denote the *characteristic vector* of u, that is, $\mathbf{1}_u(u) = 1$ and $\mathbf{1}_u(v) = 0$ for $v \in V \setminus \{u\}$. The following lemma indicates that, to bound the Lipschitz constant, it suffices to consider pairs of weight vectors that differ by one coordinate.

▶ **Lemma 15** ([14])**.** *Suppose that there exist some $c > 0$ and $L > 0$ such that*

$$\mathsf{EM}\left((\mathcal{A}(G, w), w), (\mathcal{A}(G, w + \delta \mathbf{1}_u), w + \delta \mathbf{1}_u)\right) \leq \delta L$$

holds for all $w \in \mathbb{R}^V_{\geq 0}$, $u \in V$ and $0 < \delta \leq c$. Then, $\mathcal{A}$ is L-Lipschitz.

In our algorithm, we use the following procedures *softmax* and *softmin*. These procedures are derived from the *exponential mechanism* in the literature of *differential privacy* [7] and frequently appear in the literature on Lipschitz continuity [14, 16]. Here, we organize them into a more convenient form for our use. Let $p \in \mathbb{Z}_{\geq 1}$, $x_1, \ldots, x_p \in \mathbb{R}_{\geq 0}$, and $\epsilon \in (0, 1]$. The *softmax* of $x_1, \ldots, x_p$ is taken as follows. If $\max_{i \in [p]} x_i = 0$, we set $\mathsf{softargmax}^{\epsilon}_{i \in [p]} x_i$ to be an arbitrary index i with $x_i = 0$ and $\mathsf{softmax}^{\epsilon}_{i \in [p]} x_i = 0$. Assume otherwise. First, we sample $\boldsymbol{c}$ uniformly from $\left[\frac{2\epsilon^{-1} \log(2p\epsilon^{-1})}{\max_{i \in [p]} x_i}, \frac{4\epsilon^{-1} \log(2p\epsilon^{-1})}{\max_{i \in [p]} x_i}\right]$. Let $\boldsymbol{i}^*$ be a probability distribution over $[p]$ such that

$$\Pr\left[\boldsymbol{i}^* = i\right] = \frac{\exp(\boldsymbol{c} x_i)}{\sum_{i' \in [p]} \exp(\boldsymbol{c} x_{i'})}.$$

holds for all $i \in [p]$. Then, we define $\mathsf{softargmax}^{\epsilon}_{i \in [p]} x_i := \boldsymbol{i}^*$ and $\mathsf{softmax}^{\epsilon}_{i \in [p]} x_i := x_{\boldsymbol{i}^*}$. We have the following. The proofs of Lemmas 16 and 17 are given in the full version.

▶ **Lemma 16.** *We have $\mathbb{E}\left[\mathsf{softmax}^{\epsilon}_{i \in [p]} x_i\right] \geq (1 - \epsilon) \max_{i \in [p]} x_i$.*

▶ **Lemma 17.** *Let $\delta > 0$ and assume $\delta \leq \frac{\max_{i \in [p]} x_i}{4\epsilon^{-1} \log(2p\epsilon^{-1})}$. Let $x'_1, \ldots, x'_p \in \mathbb{R}_{\geq 0}$ be numbers such that $x_i \leq x'_i \leq x_i + \delta$ holds for all $i \in [p]$. Then, $\mathsf{TV}\left(\mathsf{softargmax}^{\epsilon}_{i \in [p]} x_i, \mathsf{softargmax}^{\epsilon}_{i \in [p]} x'_i\right) \leq \frac{10\epsilon^{-1} \log(2p\epsilon^{-1}) \delta}{\max_{i \in [p]} x_i}$.*

Similarly, the *softmin* of $x_1, \ldots, x_p$ is taken as follows. If $\min_{i \in [p]} x_i = 0$, we set $\mathsf{softargmin}^{\epsilon}_{i \in [p]} x_i$ be an arbitrary index i with $x_i = 0$ and $\mathsf{softmin}^{\epsilon}_{i \in [p]} x_i = 0$. Assume otherwise. First, we sample $\boldsymbol{c}$ uniformly from $\left[\frac{2\epsilon^{-1} \log(2p\epsilon^{-1})}{\min_{i \in [p]} x_i}, \frac{4\epsilon^{-1} \log(2p\epsilon^{-1})}{\min_{i \in [p]} x_i}\right]$. Let $\boldsymbol{i}^*$ be a probability distribution over $[p]$ such that

$$\Pr\left[\boldsymbol{i}^* = i\right] = \frac{\exp(-\boldsymbol{c} x_i)}{\sum_{i' \in [p]} \exp(-\boldsymbol{c} x_{i'})}.$$

holds for all $i \in [p]$. Then, we define $\mathsf{softargmin}^{\epsilon}_{i \in [p]} x_i := \boldsymbol{i}^*$ and $\mathsf{softmin}^{\epsilon}_{i \in [p]} x_i := x_{\boldsymbol{i}^*}$. We have the following. The proofs of Lemmas 18 and 19 are given in the full version.

▶ **Lemma 18.** *We have $\mathbb{E}\left[\mathsf{softmin}^{\epsilon}_{i \in [p]} x_i\right] \leq (1 + \epsilon) \min_{i \in [p]} x_i$.*

▶ **Lemma 19.** *Let $\delta > 0$ and assume $\delta \leq \frac{\min_{i \in [p]} x_i}{4\epsilon^{-1} \log(2p\epsilon^{-1})}$. Let $x'_1, \ldots, x'_p \in \mathbb{R}_{\geq 0}$ be numbers such that $x_i \leq x'_i \leq x_i + \delta$ holds for all $i \in [p]$. Then, $\mathsf{TV}\left(\mathsf{softargmin}^{\epsilon}_{i \in [p]} x_i, \mathsf{softargmin}^{\epsilon}_{i \in [p]} x'_i\right) \leq \frac{10\epsilon^{-1} \log(2p\epsilon^{-1}) \delta}{\min_{i \in [p]} x_i}$.*

3 Lipschitz Continuous Algorithms for MSO_2 Optimization Problems

In this section, we prove Theorems 1–3. Since the proofs are similar to that for Theorem 1, we provide most of the proof of Theorem 2 in the full version. In Section 3.1, we define the notations and give a brief overview of the algorithm. Section 3.2 handles the base case, where the graph consists of a single vertex. Sections 3.3 and 3.4 analyze the impact of the parallel composition and the forget operations, respectively, on approximability and the Lipschitz constant. In Section 3.5, we put everything together to prove Theorem 1. Finally, in Section 4.1, we provide a more refined analysis for the special case of the maximum weight independent set problem to prove Theorem 3.

3.1 Definitions

For a graph G, a weight vector w, and an MSO_2 formula φ, we define

$$\mathsf{opt}_w[G,\varphi] = \max_{S\subseteq V(G), G\models\varphi(S)} w(S). \tag{1}$$

for maximization problems and

$$\mathsf{opt}_w[G,\varphi] = \min_{S\subseteq V(G), G\models\varphi(S)} w(S). \tag{2}$$

for minimization problems. If $\mathsf{sat}(G,\varphi)=\emptyset$, we define $\mathsf{opt}_w[G,\varphi]=\mathtt{nil}$. For graphs G and MSO_2 formulas φ with $\mathsf{sat}(G,\varphi)\neq\emptyset$, our algorithm recursively computes the vertex set $\mathsf{DP}_w[G,\varphi]$ that approximately achieves the maximum in Equation (1) or minimum in Equation (2). When it is clear from the context, we omit the subscript w and simply write $\mathsf{opt}[G,\varphi]$ and $\mathsf{DP}[G,\varphi]$. We perform the dynamic programming algorithm using the formulas defined in Corollary 14 over the parse tree of a term obtained from Lemma 8 and Proposition 11. Specifically, at every stage of the algorithm, it is guaranteed that each $X\in\mathsf{sat}(G,\varphi)$ satisfies $X\subseteq V(G)\setminus\mathsf{src}(G)$. Furthermore, our algorithm is randomized. Thus, $\mathsf{DP}[G,\varphi]$ can be considered as a probability distribution over vertex sets in $\mathsf{sat}(G,\varphi)$.

To bound the approximation ratio, for a weight vector $w\in\mathbb{R}^V_{\geq 0}$, we bound the ratio between $\mathbb{E}\left[w(\mathsf{DP}[G,\varphi])\right]$ and $\mathsf{opt}[G,\varphi]$. To bound the Lipschitz constant, for a weight vector $w\in\mathbb{R}^V_{\geq 0}$, $u\in V$, and $\delta>0$, we bound $\mathsf{EM}\left(\mathsf{DP}_w[G,\varphi],\mathsf{DP}_{w+\delta\mathbf{1}_u}[G,\varphi]\right)$. From now on, we will concentrate on maximizing problems. The algorithm for minimization problems is similar to that for maximization problems, while the detail of the analysis is slightly differ. We discuss the minimization version in the full version.

3.2 Base Case

Here we consider the base case. Let G be a graph with a single vertex v and no edges, where v is a source of G, and φ be an MSO_2 formula such that $\mathsf{sat}(G,\varphi)$ is nonempty and contains no set containing v. In particular, we have $\mathsf{sat}(G,\varphi)=\{\emptyset\}$. We set $\mathsf{DP}[G,\varphi]:=\emptyset$. Since $w(\emptyset)=0$, it is clear that $w\left(\mathsf{DP}[G,\varphi]\right)=\mathsf{opt}[G,\varphi]=0$. Furthermore, it is obvious that

$$\mathsf{EM}\left(\mathsf{DP}_w[G,\varphi],\mathsf{DP}_{w+\delta\mathbf{1}_u}[G,\varphi]\right)=\mathsf{EM}(\emptyset,\emptyset)=0.$$

3.3 Parallel Composition

Next, we consider the parallel composition. Let

$$\mathsf{sat}(G /\!/ H,\varphi) = \biguplus_{i\in[p]} \mathsf{sat}(G,\theta_i)\boxtimes\mathsf{sat}(H,\psi_i). \tag{3}$$

Assume we have already computed $\mathsf{DP}[G, \theta_i]$ and $\mathsf{DP}[H, \psi_i]$ for each $i \in [p]$. We compute $\mathsf{DP}[G \mathbin{/\!/} H, \varphi]$. For each $i \in [p]$, we denote $\mathsf{opt}_i := \mathsf{opt}[G, \theta_i] + \mathsf{opt}[H, \psi_i]$. By definition, we have $\max_{i \in [p]} \mathsf{opt}_i = \mathsf{opt}[G \mathbin{/\!/} H, \varphi]$. Then, we take $\boldsymbol{i}^* = \mathsf{softargmax}^{\varepsilon}_{i \in [p]} \mathsf{opt}_i$ and define $\mathsf{DP}[G \mathbin{/\!/} H, \varphi] := \mathsf{DP}[G, \theta_{\boldsymbol{i}^*}] \cup \mathsf{DP}[H, \psi_{\boldsymbol{i}^*}]$. First we analyze the approximation ratio.

▶ **Lemma 20.** *Let $0 < \alpha \leq 1$ and suppose $\mathbb{E}[w(\mathsf{DP}[G, \theta_i])] \geq \alpha \mathsf{opt}[G, \theta_i]$ and $\mathbb{E}[w(\mathsf{DP}[H, \psi_i])] \geq \alpha \mathsf{opt}[H, \psi_i]$ hold for all $i \in [p]$. Then, we have $\mathbb{E}[w(\mathsf{DP}[G \mathbin{/\!/} H, \varphi])] \geq (1 - \varepsilon)\alpha \mathsf{opt}[G \mathbin{/\!/} H, \varphi]$.*

Proof. We have

$$\begin{aligned}&\mathbb{E}[w(\mathsf{DP}[G \mathbin{/\!/} H, \varphi])] = \mathbb{E}[w(\mathsf{DP}[G, \theta_{\boldsymbol{i}^*}] \cup \mathsf{DP}[H, \psi_{\boldsymbol{i}^*}])] \geq \alpha \mathbb{E}[\mathsf{opt}[G, \theta_{\boldsymbol{i}^*}] + \mathsf{opt}[H, \psi_{\boldsymbol{i}^*}]] \\ &= \alpha \mathbb{E}\left[\mathsf{softmax}^{\varepsilon}_{i \in [p]} \mathsf{opt}_i\right] \geq (1 - \varepsilon)\alpha \max_{i \in [p]} \mathsf{opt}_i = (1 - \varepsilon)\alpha \mathsf{opt}[G \mathbin{/\!/} H, \varphi].\end{aligned}$$ ◀

Now we analyze the Lipschitz constant. We denote the variable $\boldsymbol{i}^*$ corresponding to the weight w and $w + \delta \mathbf{1}_u$ by $\boldsymbol{i}^*_w$ and $\boldsymbol{i}^*_{w+\delta \mathbf{1}_u}$, respectively. We note that $\mathsf{DP}_w[G, \theta_i]$ and $\mathsf{DP}_{w+\delta \mathbf{1}_u}[G, \theta_i]$ are the same as a distribution unless $u \in V(G) \setminus \mathsf{src}_G$. The same also holds for H. Since $V(G) \setminus \mathsf{src}_G$ and $V(H) \setminus \mathsf{src}_H$ are disjoint, without loss of generality, we can assume the $\mathsf{DP}_w[H, \psi_i]$ and $\mathsf{DP}_{w+\delta \mathbf{1}_u}[H, \psi_i]$ are the same as a distribution. We have the following.

▶ **Lemma 21.** *Let $\beta \in \mathbb{R}_{\geq 0}$ and suppose $\mathsf{EM}(\mathsf{DP}_w[G, \theta_i], \mathsf{DP}_{w+\delta \mathbf{1}_u}[G, \theta_i]) \leq \beta$ holds for all $i \in [p]$. Then, we have $\mathsf{EM}(\mathsf{DP}_w[G \mathbin{/\!/} H, \varphi], \mathsf{DP}_{w+\delta \mathbf{1}_u}[G \mathbin{/\!/} H, \varphi]) \leq 30\varepsilon^{-1} \log(2p\varepsilon^{-1})\delta + \beta$.*

Proof. We have

$$\begin{aligned}&\mathsf{EM}(\mathsf{DP}_w[G \mathbin{/\!/} H, \varphi], \mathsf{DP}_{w+\delta \mathbf{1}_u}[G \mathbin{/\!/} H, \varphi]) \\ &\leq \mathsf{TV}(\boldsymbol{i}^*_w, \boldsymbol{i}^*_{w+\delta \mathbf{1}_u}) \left(\mathsf{opt}_w[G \mathbin{/\!/} H, \varphi] + \mathsf{opt}_{w+\delta \mathbf{1}_u}[G \mathbin{/\!/} H, \varphi]\right) + \max_{i \in [p]} \left(\mathsf{EM}(\mathsf{DP}_w[G, \theta_i], \mathsf{DP}_{w+\delta \mathbf{1}_u}[G, \theta_i])\right) \\ &\leq \frac{10\varepsilon^{-1} \log(2p\varepsilon^{-1})\delta}{\mathsf{opt}_w[G \mathbin{/\!/} H, \varphi]} \left(\mathsf{opt}_w[G \mathbin{/\!/} H, \varphi] + \mathsf{opt}_{w+\delta \mathbf{1}_u}[G \mathbin{/\!/} H, \varphi]\right) + \beta \leq 30\varepsilon^{-1} \log(2p\varepsilon^{-1})\delta + \beta,\end{aligned}$$

where the last inequality is from $\varepsilon \leq 1$ and $\delta \leq \mathsf{opt}_w[G \mathbin{/\!/} H, \varphi]$. ◀

3.4 Forget

Here we consider forget operation. Let $B \subseteq \mathsf{src}_G$ and

$$\mathsf{sat}(\mathsf{fg}_B(G), \varphi) = \biguplus_{i \in [p]} \mathsf{sat}(G, \varphi_i) \boxtimes \{S_i\}. \tag{4}$$

Assume we have already computed $\mathsf{DP}[G, \varphi_i]$ for each $i \in [p]$. We compute $\mathsf{DP}[\mathsf{fg}_B(G), \varphi]$.

We first sample $\boldsymbol{c} \in \mathbb{R}_{>0}$ uniformly from $\left[\frac{2\log(2p\varepsilon^{-1})}{\varepsilon \mathsf{opt}[G, \varphi, S]}, \frac{4\log(2p\varepsilon^{-1})}{\varepsilon \mathsf{opt}[G, \varphi, S]}\right]$. For each $i \in [p]$, we denote $\mathsf{opt}_i = \mathsf{opt}[G, \varphi_i] + w(S_i)$. By definition, we have $\max_{i \in [p]} \mathsf{opt}_i = \mathsf{opt}[\mathsf{fg}_B(G), \varphi]$. Then, we take $\boldsymbol{i}^* = \mathsf{softargmax}^{\varepsilon}_{i \in [p]} \mathsf{opt}_i$ and define $\mathsf{DP}[\mathsf{fg}_B(G), \varphi] := \mathsf{DP}[G, \varphi_{\boldsymbol{i}^*}] \cup S_{\boldsymbol{i}^*}$. First we analyze the approximation ratio.

▶ **Lemma 22.** *Let $0 < \alpha \leq 1$ and suppose $\mathbb{E}[w(\mathsf{DP}[G, \varphi_i])] \geq \alpha \mathsf{opt}[G, \varphi_i]$ holds for all $i \in [p]$. Then, we have $\mathsf{DP}[\mathsf{fg}_B(G), \varphi] \geq (1 - \varepsilon)\alpha \mathsf{opt}[\mathsf{fg}_B(S), \varphi]$.*

Proof. We have

$$\begin{aligned}&\mathbb{E}[w(\mathsf{DP}[G, \varphi])] = \mathbb{E}[w(\mathsf{DP}[G, \varphi_{\boldsymbol{i}^*}] \cup S_{\boldsymbol{i}^*})] \geq \mathbb{E}[\alpha \mathsf{opt}[G, \varphi_{\boldsymbol{i}^*}] + w(S_{\boldsymbol{i}^*})] \geq \alpha \mathbb{E}[\mathsf{opt}[G, \varphi_{\boldsymbol{i}^*}] + w(S_{\boldsymbol{i}^*})] \\ &= \alpha \mathbb{E}\left[\mathsf{softmax}^{\varepsilon}_{i \in [p]} \mathsf{opt}_i\right] \geq (1 - \varepsilon)\alpha \max_{i \in [p]} \mathsf{opt}_i = (1 - \varepsilon)\alpha \mathsf{opt}[\mathsf{fg}_B(G), \varphi].\end{aligned}$$ ◀

Now we analyze the Lipschitz constant.

▶ **Lemma 23.** *Let $\beta \in \mathbb{R}_{\geq 0}$ and suppose* $\mathsf{EM}\left(\mathsf{DP}_w[G,\varphi_i], \mathsf{DP}_{w+\delta\mathbf{1}_u}[G,\varphi_i]\right) \leq \beta$ *holds for all* $i \in [p]$*. Then, we have* $\mathsf{EM}\left(\mathsf{DP}_w[\mathsf{fg}_B(G),\varphi], \mathsf{DP}_{w+\delta\mathbf{1}_u}[\mathsf{fg}_B(G),\varphi]\right) \leq 31\varepsilon^{-1}\log(2p\varepsilon^{-1})\delta + \beta$.

Proof. We have

$$\begin{aligned}
&\mathsf{EM}\left(\mathsf{DP}_w[\mathsf{fg}_B(G),\varphi], \mathsf{DP}_{w+\delta\mathbf{1}_u}[\mathsf{fg}_B(G),\varphi]\right)\\
&\quad\leq \mathsf{TV}\left(\boldsymbol{i}^*_w, \boldsymbol{i}^*_{w+\delta\mathbf{1}_u}\right)\left(\mathsf{opt}_w[\mathsf{fg}_B(G),\varphi] + \mathsf{opt}_{w+\delta\mathbf{1}_u}[\mathsf{fg}_B(G),\varphi]\right)\\
&\qquad+\max_{i\in[p]}\left(\mathsf{EM}\left(\mathsf{DP}_w[G,\varphi_i]\cup S_i, \mathsf{DP}_{w+\delta\mathbf{1}_u}[G,\varphi_i]\cup S_i\right)\right)\\
&\quad\leq \frac{10\varepsilon^{-1}\log(2p\varepsilon^{-1})\delta}{\mathsf{opt}_w[\mathsf{fg}_B(G),\varphi]}\left(\mathsf{opt}_w[\mathsf{fg}_B(G),\varphi] + \mathsf{opt}_{w+\delta\mathbf{1}_u}[\mathsf{fg}_B(G),\varphi]\right)\\
&\qquad+\max_{i\in[p]}\left(\mathsf{EM}\left(\mathsf{DP}_w[G,\varphi_i], \mathsf{DP}_{w+\delta\mathbf{1}_u}[G,\varphi_i]\right)\right) + \max_{i\in[p]}\left[d_{\mathrm{w}}((S_i,w),(S_i,w+\delta\mathbf{1}_u))\right]\\
&\quad\leq 30\varepsilon^{-1}\log(2p\varepsilon^{-1})\delta + \beta + \delta \leq 31\varepsilon^{-1}\log(2p\varepsilon^{-1})\delta + \beta. \qquad ◀
\end{aligned}$$

3.5 Putting Together

Let $G = (V,E)$ be a graph with treewidth k. From Lemma 8 and Proposition 11, we can compute a HR-parse tree t over $(k+1)$-graph denoting G with height $O(\log k + \log n) \leq O(\log n)$. We perform the dynamic programming algorithm on this parse tree. We have the following.

▶ **Lemma 24.** *Let $\varepsilon \in (0,1]$ and h be the height of t. Our algorithm outputs a solution $X \in \mathsf{sat}(G,\varphi)$ such that $\mathbb{E}[w(X)] \geq (1-h\varepsilon)\mathsf{opt}[G,\varphi]$. The Lipschitz constant is bounded by $31h\varepsilon^{-1}\log(2p_{\max}\varepsilon^{-1})$, where $p_{\max}$ is the maximum of p among all update formulas (3) or (4) that the algorithm uses.*

Proof. The approximability bound is obtained by repeatedly applying Lemmas 20 and 22 and $(1-\varepsilon)^h \geq 1 - h\varepsilon$. The Lipschitzness bound is obtained by repeatedly applying Lemmas 21 and 23. ◀

Since $p_{\max}$ is bounded by a function of k and $|\varphi|$, we have the following.

Proof of Theorem 1. The claim follows by substituting ε, h, and k in Lemma 24 with $\frac{\varepsilon}{\Theta(\log n)}$, $O(\log n)$, and $3k+2$, respectively. ◀

4 Special Cases

4.1 Independent Set

From the proof of Theorem 1, $f(k,\varphi)$ in Theorem 1 is bounded by $\log(2p_{\max})$. Particularly, if $p_{\max}$ is bounded by $2^{O(k)}$, the Lipschitz constant depends linearly on k. Here, we prove Theorem 3 for the maximum weight independent set problem by showing that $p_{\max} \leq 2^{O(k)}$. Similar arguments can be applied to several other problems, such as the minimum weight vertex cover problem and the minimum weight dominating set problem. For a k-graph G and $S \subseteq \mathsf{src}(G)$, let

$$\varphi_S(X) \equiv \forall x(x \in \mathsf{src}(G) \to \neg x \in X) \wedge \forall x \forall y(((x \in X \vee x \in S) \wedge (y \in X \vee y \in S)) \to \neg\mathsf{adj}(x,y)).$$

In words, $\mathsf{sat}(G, \varphi_S)$ is the family of subsets X of V such that X is disjoint from $\mathsf{src}(G)$ and $X \cup S$ is an independent set of G. Then, we have

$$\mathsf{sat}(\mathsf{fg}_B(G), \varphi_S) = \biguplus_{S' \subseteq B} \mathsf{sat}(G, \varphi_{S \cup S'}) \boxtimes \{S'\}, \quad \mathsf{sat}(G \mathbin{/\!/} H, \varphi_S) = \mathsf{sat}(G, \varphi_S) \boxtimes \mathsf{sat}(H, \varphi_S).$$

Particularly, we have $p_{\max} \leq 2^{O(k)}$ and therefore Theorem 3 is proved.

4.2 Max Ones

Recall that in *max ones problem*, we given a 3CNF formula Φ over a variable set X and a weight function $w : X \to \mathbb{R}_{\geq 0}$, the goal is to find a satisfying assignment $\sigma : X \to \{0, 1\}$ that maximizes the weight $\sum_{x \in \sigma^{-1}(1)} w(x)$. We reduce the problem to a graph problem. Let $G_\Phi = (X \cup \bar{X} \cup \mathcal{C}, E)$ be the graph such that X is the variable set of Φ, $\bar{X} = \{\bar{x}_i : x_i \in X\}$, $\mathcal{C}$ is the clause set of Φ, $\{x_i, \bar{x}_i\} \in E$ for all $x_i \in X$, $\{C_j, x_i\} \in E$ iff clause C_j contains x_i as a positive literal, and $\{C_j, \bar{x}_i\} \in E$ iff clause C_j contains x_i as a negative literal. Let $w'(x) = w(x)$ if $x \in X$, and otherwise $w'(x) = 0$. Note that the treewidth of G_Φ is at most $2|X|$.

Here, the notations $a \in A \cup B$, $a \in A \cap B$, $a \in A \setminus B$, $\forall a \in A\ \psi$, and $\exists a \in A\ \psi$ are syntactic sugar defined in the usual sense. For a k-graph G and $S, D \subseteq \mathsf{src}(G)$, let

$$\varphi_{S,D}(A) \equiv \exists B \begin{bmatrix} & \forall a \in A \cup S\ \forall b \in B\ (a \in X \wedge b \in \bar{X} \wedge \neg \mathsf{adj}(a, b)) \\ \wedge & \forall x \in X \cup \bar{X}\ \forall y \in X \cup \bar{X}\ (\mathsf{adj}(x, y) \to (x \in A \cup S \cup B \vee y \in A \cup S \cup B) \\ \wedge & \forall s \in \mathsf{src}(G)\ (\neg s \in A) \wedge \forall c \in \mathcal{C} \setminus D\ \exists x \in A \cup S \cup B\ (\mathsf{adj}(x, c)) \end{bmatrix}.$$

The first and the second rows says that $A \cup S$ and B represent the sets $\sigma^{-1}(1)$ and $\sigma^{-1}(0)$ of an assignment σ, respectively. The third row says that A has no sources, and all clauses except in D are satisfied by the σ defined by $A \cup S$ and B. Then, the max ones problem is equivalent to find a vertex set Y maximizes the weight $\sum_{x \in Y} w'(x)$ and satisfies $G_\Phi \models \varphi_{\emptyset,\emptyset}(Y)$. Here, we have

$$\mathsf{sat}(\mathsf{fg}_B(G), \varphi_{S,D}) = \begin{cases} \biguplus_{S' \subseteq B} \mathsf{sat}(G, \varphi_{S \cup S', D}) \boxtimes \{S'\} & \text{if } D \cap B = \emptyset, \\ \emptyset & \text{otherwise} \end{cases}$$

$$\mathsf{sat}(G \mathbin{/\!/} H, \varphi_{S,D}) = \biguplus_{\substack{D_1 \cap D_2 = D \\ D_1, D_2 \subseteq \mathsf{src}(G)}} \mathsf{sat}(G, \varphi_{S,D_1}) \boxtimes \mathsf{sat}(H, \varphi_{S,D_2}).$$

Particularly, we have $p_{\max} \leq 2^{O(k)}$.

References

1 Brenda Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM (JACM)*, 41(1):153–180, 1994. doi:10.1145/174644.174650.

2 Hans Bodlaender and Torben Hagerup. Parallel algorithms with optimal speedup for bounded treewidth. *SIAM Journal on Computing*, 27(6):1725–1746, 1998. doi:10.1137/S0097539795289859.

3 Bruno Courcelle. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.

4 Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic – A Language-Theoretic Approach*, volume 138 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2012. doi:10.1017/CBO9780511977619.

5 Bruno Courcelle and Mohamed Mosbah. Monadic second-order evaluations on tree-decomposable graphs. *Theoretical Computer Science*, 109(1&2):49–82, 1993. doi:10.1016/0304-3975(93)90064-Z.

6 Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity classifications of Boolean constraint satisfaction problems.* SIAM, 2001.

7 Cynthia Dwork. Differential privacy. In *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 4052, pages 1–12, 2006. doi:10.1007/11787006_1.

8 Noah Fleming and Yuichi Yoshida. Sensitivity lower bounds for approximation algorithms. *arXiv preprint arXiv:2411.02744*, 2024. doi:10.48550/arXiv.2411.02744.

9 Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David Williamson. The approximability of constraint satisfaction problems. *SIAM Journal on Computing*, 30(6):1863–1920, 2001. doi:10.1137/S0097539799349948.

10 Tuukka Korhonen. Lower bounds on dynamic programming for maximum weight independent set. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 198 of *LIPIcs*, pages 87:1–87:14, 2021. doi:10.4230/LIPICS.ICALP.2021.87.

11 Stephan Kreutzer. Algorithmic meta-theorems. In *Finite and Algorithmic Model Theory*, volume 379 of *London Mathematical Society Lecture Note Series*, pages 177–270. Cambridge University Press, 2011.

12 Soh Kumabe and Yuichi Yoshida. Average sensitivity of dynamic programming. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1925–1961, 2022. doi:10.1137/1.9781611977073.77.

13 Soh Kumabe and Yuichi Yoshida. Average sensitivity of the knapsack problem. In *European Symposium on Algorithms (ESA)*, volume 244, pages 75:1–75:14, 2022. doi:10.4230/LIPICS.ESA.2022.75.

14 Soh Kumabe and Yuichi Yoshida. Lipschitz continuous algorithms for graph problems. In *Symposium on Foundations of Computer Science (FOCS)*, pages 762–797, 2023. doi:10.1109/FOCS57990.2023.00051.

15 Soh Kumabe and Yuichi Yoshida. Lipschitz continuous allocations for optimization games. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 297, pages 102:1–102:16, 2024. doi:10.4230/LIPICS.ICALP.2024.102.

16 Soh Kumabe and Yuichi Yoshida. Lipschitz continuous algorithms for covering problems. In *Symposium on Discrete Algorithms (SODA)*, pages 1057–1093, 2025. doi:10.1137/1.9781611978322.31.

17 Leonid Libkin. *Elements of Finite Model Theory.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004. doi:10.1007/978-3-662-07003-1.

18 Quanquan Liu, Grigoris Velegkas, Yuichi Yoshida, and Felix Zhou. Pointwise lipschitz continuous graph algorithms via proximal gradient analysis. *arXiv preprint*, 2024. doi:10.48550/arXiv.2405.08938.

19 Shogo Murai and Yuichi Yoshida. Sensitivity analysis of centralities on unweighted networks. In *The world wide web conference*, pages 1332–1342, 2019. doi:10.1145/3308558.3313422.

20 Pan Peng and Yuichi Yoshida. Average sensitivity of spectral clustering. In *ACM SIGKDD international conference on knowledge discovery & data mining (KDD)*, pages 1132–1140, 2020. doi:10.1145/3394486.3403166.

21 Nithin Varma and Yuichi Yoshida. Average sensitivity of graph algorithms. *SIAM Journal on Computing*, 52(4):1039–1081, 2023. doi:10.1137/21M1399592.

22 Yuichi Yoshida and Shinji Ito. Average sensitivity of euclidean k-clustering. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:32487–32498, 2022.

23 Yuichi Yoshida and Samson Zhou. Sensitivity analysis of the maximum matching problem. In *Innovations in Theoretical Computer Science Conference (ITCS)*, volume 185, pages 58:1–58:20, 2021. doi:10.4230/LIPIcs.ITCS.2021.58.