Fixed-Parameter Tractable Certified Algorithms for Covering and Dominating in Planar Graphs and Beyond

Benjamin Merlin Bumpus
University of Florida, Gainesville, FL, USA

Bart M. P. Jansen
Eindhoven University of Technology, The Netherlands

Jaime Venne
Eindhoven University of Technology, The Netherlands

Abstract

For a positive real $\gamma \geq 1$, a γ-certified algorithm for a vertex-weighted graph optimization problem is an algorithm that, given a weighted graph (G, w), outputs a re-weighting of the graph obtained by scaling each weight individually with a factor between 1 and γ, along with a solution which is optimal for the perturbed weight function. Here we provide $(1+\varepsilon)$-certified algorithms for Dominating Set and H-Subgraph-Free-Deletion which, for any $\varepsilon > 0$, run in time $f(1/\varepsilon) \cdot n^{\mathcal{O}(1)}$ on minor-closed classes of graphs of bounded local tree-width with polynomially-bounded weights. We obtain our algorithms as corollaries of a more general result establishing FPT-time certified algorithms for problems admitting, at an intuitive level, certain "local solution-improvement properties". These results improve – in terms of generality, running time and parameter dependence – on Angelidakis, Awasthi, Blum, Chatziafratis and Dan's XP-time $(1+\varepsilon)$-certified algorithm for Independent Set on planar graphs (ESA2019). Furthermore, our methods are also conceptually simpler: our algorithm is based on elementary local re-optimizations inspired by Baker's technique, as opposed to the heavy machinery of the Sherali-Adams hierarchy required in previous work.

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

Keywords and phrases fixed-parameter tractability, certified algorithms

Digital Object Identifier 10.4230/LIPIcs.SWAT.2024.19

Funding Authors Bumpus and Jansen received funding by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 803421, ReduceSearch).

Acknowledgements We are grateful to the anonymous referees for pointing out Theorem 2.5 and making suggestions that improved the presentation of the paper.

1 Introduction

In many algorithmic contexts there is no tolerance for uncertainty. For instance, when lives are at stake (e.g. kidney exchanges [6, 15]), the difference between an approximate solution and a truly optimal one is staggering. However, finding exact optima only makes sense if the *objective function* which we are optimizing is known to accurately model the optimization problem at hand (and often this is not the case in e.g. clustering or vertex-optimization problems [13]). Indeed, if the objective function is only an *approximate* model, then there is no use in finding a true optimum relative to this objective function: after all, how could one tell whether the returned solution is "truly" optimal or if it is instead optimal simply due to the error, or noise in the objective function?

Thus it is clear that, if we are optimizing an objective function which is subject to a certain degree γ of error, then it only makes sense to find *optimal* solutions when the inputs are *stable* under γ-perturbations: i.e. stable under small variations in the objective by factors which are at most our error γ. The precise formulation of the notion of γ-stability (which follows) is due to Bilu and Linial [3] and is a necessary prerequisite to the notion of *certified algorithms*, the focus of this paper.

▶ **Definition 1.1** (γ-perturbation). *For any $\gamma \in \mathbb{R}_{\geq 1}$ and set S, a* γ-perturbation *of a function $w\colon S \to \mathbb{R}$ is a function $w'\colon S \to \mathbb{R}$ satisfying $w(v) \leq w'(v) \leq \gamma \cdot w(v)$ for all $v \in S$.*

▶ **Definition 1.2** (γ-stable). *For any $\gamma \in \mathbb{R}_{\geq 1}$, a* γ-stable instance $(G, w\colon V(G) \to \mathbb{R})$ *of a vertex-minimization problem* Π *is an instance admitting a unique optimal solution S which remains optimal (though not necessarily unique) even under γ-perturbations of (G, w).*

Determining whether an instance is γ-stable or not can be a challenging computational task [13]. However, this is often beside the point: if we do not know whether the objective function we are optimizing has error or not, then it is enough to find a γ-approximate solution with the extra guarantee that the returned solution is optimal whenever the instance is γ-stable. Certified algorithms [3, 12, 13, 14] satisfy these requirements and more.

▶ **Definition 1.3** (Certified algorithm). *A* γ-certified solution *to an instance $(G, w\colon V(G) \to \mathbb{N})$ of a weighted vertex-optimization problem Π is a pair $(S, w'\colon V(G) \to \mathbb{R}_+)$ where w' is a γ-perturbation of w and S is an optimal solution on (G, w'). A* γ-certified algorithm *for Π is one mapping instances of Π to γ-certified solutions.*

Note that every γ-certified algorithm also serves as a factor-γ approximation algorithm [13, Thm. 5.11] for the problem, while the converse is false in general. For example, a γ-approximation for the Dominating Set problem may output a solution that fails to be inclusion-minimal, but this can never be the output of a γ-certified algorithm since there is no γ-perturbation for which such a solution is optimal.

Contributions. This paper is a foray into merging certified algorithms with parameterized complexity: here we develop FPT-time $(1+\varepsilon)$-certified algorithms for *vertex-optimization problems* (Definition 2.2) parameterized by $1/\varepsilon$. Specifically we provide *certified* algorithms for H-Subgraph-Free-Deletion (for connected H) and Dominating Set which run in polynomial time on minor-closed classes of bounded local tree-width, which are exactly the apex-minor free graphs (Section 2). These results improve – in terms of generality, running time and parameter dependence – on Angelidakis, Awasthi, Blum, Chatziafratis and Dan's XP-time $(1+\varepsilon)$-certified algorithm for Independent Set on planar graphs [1] which inspired the present paper.

Our results (Corollary 3.6) are obtained as by-products of our main theorem (Theorem 1.7). They draw inspiration from Baker's celebrated technique [2] and they establish FPT-time *certified* algorithms for any problem Π on such graph classes provided Π satisfies certain "local solution-improvement properties". The rest of this section will lead up to the formal statement of our main theorem by explaining precisely what these properties consist of.

The "local" nature of the "solution-improvement properties" mentioned above has to do with the operation of *m-stitching.* Intuitively, this operation consists of amending a given solution S_1 by "stitching" onto it a small, local portion of another solution S_2. In the following definition, $N_G^m[J]$ denotes the closed m-neighborhood of vertex set J (see Section 2).

▶ **Definition 1.4** (m-stitch operation)**.** *For an integer $m \geq 0$ and vertex sets $J, S_1, S_2 \subseteq V(G)$ of a graph G, we define the m-*stitch *of S_2* onto *S_1* along *J as:*

$$S_2 \oplus^m_{G,J} S_1 := (S_1 \setminus J) \cup (S_2 \cap N^m_G[J]).$$

Naturally we refer to vertex-optimization problems whose set of feasible solutions is closed under the m-stitch operator as *m-stitchable.*

▶ **Definition 1.5** (m-stitchable)**.** *A vertex-optimization problem Π is m-*stitchable *if, for any feasible solutions S_1 and S_2 to Π on a graph G and any vertex set $J \subseteq V(G)$, we have that $S_2 \oplus^m_{G,J} S_1$ is a feasible solution to Π on G.*

While the stitching operation seems natural, we are not aware of earlier work exploiting this idea. Our main theorem requires as a subroutine an algorithm for the following computational task for minimization problems. Roughly speaking, algorithms for the task below should be thought of as "local optimization" routines which improve any given solution S to produce solutions which are at least as good as any m-stitch onto S.

Π-m-Stitching **Parameter:** $\mathbf{tw}(G[N^m_G[J]])$
Input: an instance $(G, w \colon V(G) \to \mathbb{N})$ to an m-stitchable vertex-optimization problem Π along with a solution S and a vertex set $J \subseteq V(G)$.
Task: find a feasible solution S' to Π on G, such that for all other feasible solutions S^*, we have $w(S') \leq w(S^* \oplus^m_{G,J} S)$.

Notice that Π-m-Stitching is parameterized by the tree-width of the closed distance-m neighborhood of J; this restricts the exponential dependency of this local optimization task in terms of the tree-width of the closed m-neighborhood of J.

Finally, we can state our main result (Theorem 1.7) which, sweeping some details under the rug, can be thought of as a way of turning any algorithm for Π-m-Stitching into an FPT-time *certified* algorithm for Π whenever we can quickly guess at least one feasible solution (Definition 1.6).

▶ **Definition 1.6** (Guessable)**.** *We say that a vertex-optimization problem Π is* guessable *if there is an algorithm that outputs a feasible solution (with no requirement for optimality) in polynomial-time.*

▶ **Theorem 1.7** (main)**.** *Let $\mathcal{G}$ be a minor-closed graph class whose local tree-width is bounded above by a linear function of the form $g \colon r \mapsto \lambda r$ (where $r \in \mathbb{N}$) for some given, fixed $\lambda \in \mathbb{R}$. If Π is a vertex-minimization problem such that:*

- *Π is guessable and m-stitchable for some $m \in \mathbb{N}$, and*
- *there exists an algorithm $\mathcal{A}$ which solves Π-m-Stitching in time $f(t) \cdot |V(G)|^{\mathcal{O}(1)}$, where $t = \mathbf{tw}(G[N^m_G[J]])$ and f is some computable function;*

then, for each $\varepsilon > 0$ there is a $(1+\varepsilon)$-certified algorithm for Π which runs in time $f(\lambda m/\varepsilon) \cdot |V(G)|^{\mathcal{O}(1)}$ on any input $(G, w \colon V(G) \to \mathbb{N})$ with $G \in \mathcal{G}$ and polynomially-bounded weights.

We note that Theorem 1.7 also applies to the *complementary* maximization problem (see Section 5 for the formal definition) of any minimization problem Π as above. This observation will furthermore allow us to obtain a $2^{\mathcal{O}(1/\varepsilon)} \cdot n^{\mathcal{O}(1)}$-time certified algorithm for the *maximum independent set* problem (with polynomially bounded integer weights), which improves on the algorithm with running time $n^{\mathcal{O}(1/\varepsilon)}$ by Angelidakis, Awasthi, Blum, Chatziafratis and Dan [1]. Apart from being more efficient and more general, our algorithm is also conceptually simpler. It relies on repeated improvement of a solution in bounded-tree-width subgraphs, rather than the technical machinery of the Sherali-Adams hierarchy employed in earlier work.

Organization. After establishing some preliminary background and notation in Section 2, we will show in Section 3 how to apply our main theorem to obtain certified algorithms for H-Subgraph-Free-Deletion and Dominating Set. The main theorem itself (Theorem 1.7) is instead proved later on in Section 4. We discuss our algorithmic results and their application to complementary maximization problems in Section 5, which is also where we pose open questions as an invitation to further work.

2 Preliminaries

We follow the convention that zero is a natural number. We only consider finite, simple, and undirected graphs, which consist of a vertex set $V(G)$ and edge set $E(G) \subseteq \binom{V(G)}{2}$. For $m \in \mathbb{N}$, the *closed* m-neighborhood $N_G^m[X]$ of a vertex subset $X \subseteq V(G)$ in G is defined inductively as $N_G^m[X] := N_G[N_G^{m-1}[X]]$ where $N_G^1[X] = N_G[X] = \{y \in V(G) \mid \exists x \in X \colon \{x, y\} \in E(G)\} \cup X$. The *open* m-neighborhood $N_G^m(X)$ is defined as $N_G^m(X) := N_G^m[X] \setminus X$. The *tree-width* [8] of a graph G is denoted $\mathbf{tw}(G)$. The *diameter* of a connected graph G, which is defined as the maximum number of edges on any shortest path, is denoted by $\mathsf{diam}(G)$.

Throughout this paper we will always assume that weight functions are polynomially bounded in the size of the graph; i.e. we always consider weight functions of the form $w \colon V(G) \to \{0, \dots, |V(G)|^{\mathcal{O}(1)}\}$. This restriction is crucial to obtaining polynomial-time algorithms for the vertex-optimization problems (defined below) considered in this paper.

▶ **Definition 2.1.** *A* vertex-subset property $\mathcal{P}$ *assigns to each graph G the subset $\mathcal{P}(G) \subseteq 2^{V(G)}$ of vertex sets that satisfy property $\mathcal{P}$ on G. We say that a set $S \subseteq V(G)$ is* feasible *for $\mathcal{P}$ on G when $S \in \mathcal{P}(G)$.*

▶ **Definition 2.2** (vertex-optimization)**.** *A* vertex-optimization problem Π *is any pair of the form* $(\mathcal{P}, \mathsf{goal})$ *consisting of a vertex-subset property $\mathcal{P}$ and a function* $\mathsf{goal} \in \{\min, \max\}$*. The task of Π is to find some vertex subset $\hat{S} \in \mathcal{P}(G)$ such that $w(\hat{S}) = \mathsf{goal}_{S \in \mathcal{P}(G)} w(S)$. We call Π a* vertex-minimization problem *if* $\mathsf{goal} = \min$ *and a* vertex-maximization problem *otherwise.*

Our main algorithmic theorems concern algorithms running in minor-closed classes of (linearly) bounded local tree-width. We recall these notions below (where $d(x, y)$ denotes the usual shortest-paths distance metric on graphs).

▶ **Definition 2.3** (local tree-width)**.** *Given a graph G, the* local tree-width of G *is the map*

$$\mathbf{loctw}^G : \mathbb{N} \to \mathbb{N} \quad \textit{where} \quad \mathbf{loctw}^G : \delta \mapsto \max_{x \in V(G)} \mathbf{tw}\big(G[\{y \in V(G) : d(x, y) \leq \delta\}]\big).$$

▶ **Definition 2.4** (graphs of bounded local tree-width)**.** *A graph class $\mathcal{C}$ has* bounded local tree-width *if there is a function $f \colon \mathbb{N} \to \mathbb{R}$ such that $\mathbf{loctw}^G(r) \leq f(r)$ for all $(G, r) \in \mathcal{C} \times \mathbb{N}$. Furthermore, if there is a $\lambda \in \mathbb{R}$ such that the function f above can be defined as $f \colon r \mapsto \lambda r$, then we say that $\mathcal{C}$ has λ-linear local tree-width.*

An *apex graph* is a graph that can be made planar by removing a single vertex. Eppstein [9] proved that a minor-closed class of graphs has bounded local tree-width if and only if it excludes an *apex* graph as a minor. Demaine and Hajiaghayi [7, Theorem 4.1] proved that any apex-minor-free graph has *linear* local tree-width, thereby leading to the following equivalence.

▶ **Theorem 2.5** ([7, 9])**.** *A minor-closed graph class $\mathcal{C}$ has bounded local tree-width if and only if it has λ-linear local tree-width for some $\lambda \in \mathbb{R}$.*

For any graph-theoretic notation not defined here, we refer the reader to Diestel's textbook [8]; similarly for standard notation in parameterized complexity theory see Cygan et al.'s textbook [4].

3 Applications of Theorem 1.7

Here we will apply Theorem 1.7 to obtain FPT-time certified algorithms for Dominating Set and H-Subgraph-Free-Deletion. We recall the definitions of these problems below.

> H-Subgraph-Free-Deletion (for a fixed connected graph H)
> **Input:** a vertex-weighted graph $(G, w\colon V(G) \to \mathbb{N})$.
> **Task:** find a minimum-weight subset $X \subseteq V(G)$ such that no subgraph of $G - X$ is isomorphic to H.

> Dominating Set
> **Input:** a vertex-weighted graph $(G, w\colon V(G) \to \mathbb{N})$.
> **Task:** find a minimum-weight subset $X \subseteq V(G)$ such that $V(G) = N_G[X]$.

To apply our main theorem to these problems we need to show that they are guessable (which is trivially true: $V(G)$ is feasible solution), m-stitchable for some appropriate choices of m, and that there are FPT-time algorithms for the relevant stitching problems parameterized by tree-width. We begin with stitchability.

▶ **Lemma 3.1.** *Dominating Set is 2-stitchable while H-Subgraph-Free-Deletion is* $\mathsf{diam}(H)$*-stitchable for any connected graph H.*

Proof. Consider any three vertex sets $J, S_1, S_2 \subseteq V(G)$.

First we consider Dominating Set. If S_1 and S_2 are dominating sets, then so is $S_2 \oplus^2_{G,J} S_1$: any vertex of $V(G) \setminus N_G[J]$ is dominated by $S_1 \setminus J$ while vertices of $N_G[J]$ are dominated by $S_2 \cap N^2_G[J]$. Note that we need to consider the 2-neighborhood of J, since there might be vertices in $N_G(J)$ that S_1 dominates from within J but that S_2 dominates from $N^2_G(J)$.

Now we turn our attention to H-Subgraph-Free-Deletion. Let $h\colon H \hookrightarrow G$ be an H-subgraph of G. If S_1 and S_2 are H-hitting sets and $h(H)$ is not hit by $S_1 \setminus J$, then $V(h(H) \cap J) \neq \emptyset$. Hence $h(H)$ lies entirely in $N_G^{\mathsf{diam}(H)}[J]$, since H is connected. But then $h(H)$ is hit by $S_2 \cap N_G^{\mathsf{diam}(H)}[J]$. Thus $S_2 \oplus_{G,J}^{\mathsf{diam}(H)} S_1$ is an H-hitting set. ◀

Next we give algorithms for H-Subgraph-Free-Deletion-Stitching (Lemma 3.2) and Dominating Set-Stitching (Lemma 3.3).

▶ **Lemma 3.2.** *Let H be a fixed connected graph and $m := \mathsf{diam}(H)$. Given any algorithm $\mathcal{A}$ which solves H-Subgraph-Free-Deletion on any vertex-weighted instance $(G, w\colon V(G) \to \mathbb{N})$ in time $f(\mathbf{tw}(G)) \cdot |V(G)|^c$ for some function f and constant c, the following algorithm solves H-Subgraph-Free-Deletion-m-Stitching in time $f(\mathbf{tw}(Q)) \cdot |V(G)|^c$ where $Q = G[N^m_G[J]]$.*

- ***Algorithm Stitch-H-Del***
- ***Input:*** *a vertex-weighted graph $(G, w\colon V(G) \to \mathbb{N})$, a vertex set $J \subseteq V(G)$, and a feasible solution S_1 on G, i.e., graph $G - S_1$ has no subgraph isomorphic to H.*
- ***Output:*** *a feasible solution S' on G, such that for all other feasible solutions S^*, we have $w(S') \leq w(S^* \oplus^m_{G,J} S_1)$.*

1. *Let $F = G[N^m_G[J] \setminus (S_1 \setminus J)]$.*
2. *Let S_2 be the output of the algorithm $\mathcal{A}$ on input $(F, w|_{V(F)})$.*
3. *Return $S_2 \oplus^m_{G,J} S_1$ if $w(S_2 \oplus^m_{G,J} S_1) < w(S_1)$ and S_1 otherwise.*

Proof. The running time is clearly dominated by that of $\mathcal{A}$. Notice, towards proving correctness, that $S_2 \oplus^m_{G,J} S_1$ is feasible: the set $S_2' := S_2 \cup (V(G) \setminus V(F)) \cup (S_1 \cap N^m_G(J))$ is an H-deletion set in G and thus, by the m-stitchability of H-Subgraph-Free-Deletion and definition of F, we find that $S_2' \oplus^m_{G,J} S_1 = S_2 \oplus^m_{G,J} S_1$ is an H-deletion set.

Now assume by way of contradiction that there is a feasible solution S_3 such that $w(S_2 \oplus^m_{G,J} S_1) > w(S_3 \oplus^m_{G,J} S_1)$. Then we have that:

$$\begin{aligned}
w|_{V(F)}(S_2) = w(S_2) = & & \text{(since } S_2 \subseteq V(F))\\
= w(S_2 \cap N^m_G[J]) & & \text{(since } V(F) \subseteq N^m_G[J])\\
= w(S_1 \setminus J) + w(S_2 \cap N^m_G[J]) - w(S_1 \setminus J) & & \\
= w\big((S_1 \setminus J) \cup (S_2 \cap N^m_G[J])\big) - w(S_1 \setminus J) & & \text{(since } V(F) \cap (S_1 \setminus J) = \emptyset)\\
= w(S_2 \oplus^m_{G,J} S_1) - w(S_1 \setminus J) & & \text{(by def. of stitch)}\\
> w(S_3 \oplus^m_{G,J} S_1) - w(S_1 \setminus J) & & \text{(by assumption on } S_3)\\
= w\big((S_1 \setminus J) \cup (S_3 \cap N^m_G[J])\big) - w(S_1 \setminus J) & & \text{(by def. of stitch)}\\
= w\big((S_3 \cap N^m_G[J]) \setminus (S_1 \setminus J)\big) & & (w(A \cup B) - w(A) = w(B \setminus A))\\
= w\big(S_3 \cap (N^m_G[J] \setminus (S_1 \setminus J))\big) & & ((A \cap B) \setminus C = A \cap (B \setminus C))\\
= w(S_3 \cap V(F)) & & \text{(by def. of } F)\\
= w|_{V(F)}(S_3 \cap V(F)) & &
\end{aligned}$$

which contradicts the fact that S_2 was optimal on $(F, w|_{V(F)})$ since $S_3 \cap V(F)$ is an H-deletion set on F (because the property of being an H-deletion set is closed under induced subgraphs). ◀

Since – in contrast to H-deletion sets – the property of being a dominating set is not closed under taking induced subgraphs, our algorithm for Dominating Set-2-Stitching will require slightly different ideas from those in Lemma 3.2. Indeed, rather than finding a solution that is locally optimal after the removal of $S_1 \setminus J$ (as we did in the previous lemma), we will instead find a minimum-weight set that dominates all vertices which are not already dominated by $S_1 \setminus J$.

▶ **Lemma 3.3.** *Given any algorithm $\mathcal{A}$ which solves Dominating Set on any vertex-weighted instance $(G, w\colon V(G) \to \mathbb{N})$ in time $f(\mathbf{tw}(G)) \cdot |V(G)|^c$ for some function f and constant c, the following algorithm solves Dominating Set-2-Stitching in time $f(\mathbf{tw}(Q)) \cdot |V(G)|^c$ where $Q = N^2_G[J]$.*

- ***Algorithm Stitch-Dom-Set***
- ***Input:** a vertex-weighted graph $(G, w\colon V(G) \to \mathbb{N})$, a vertex set $J \subseteq V(G)$, and a dominating set S_1 on G.*
- ***Output:** a dominating set S' in G, such that, for all other dominating sets S^*, we have $w(S') \le w(S^* \oplus^2_{G,J} S_1)$.*

1. *Define F to be the graph obtained from $G[N^2_G[J]]$ by adding a new vertex $\mathfrak{f}$ with $N_F(\mathfrak{f}) := N_G(N_G[J])$. (Vertex $\mathfrak{f}$ is adjacent to the vertices at distance exactly two from J in G.)*

2. *Define* $w_F : V(F) \to \mathbb{N}$ *as*

$$w_F \colon x \mapsto \begin{cases} 0 & \text{if } x \in (S_1 \setminus J) \cup \{\mathfrak{f}\} \\ w(x) & \text{otherwise.} \end{cases} \tag{1}$$

3. *Let* $S_2 = S_2' \setminus \{\mathfrak{f}\}$ *where* S_2' *is the output of algorithm* $\mathcal{A}$ *on input* (F, w_F)*.*
4. *Return* $S_2 \oplus^2_{G,J} S_1$ *if* $w(S_2 \oplus^2_{G,J} S_1) < w(S_1)$ *and* S_1 *otherwise.*

Proof. The proofs of the running-time bound and feasibility of $S_2 \oplus^2_{G,J} S_1$ are virtually identical to Lemma 3.2. Notice that we can assume that $S_1 \cap N^2_G(J) \subseteq S_2$ since, by its definition in the algorithm above, $w_F(S_1 \setminus J) = 0$. The rest of the proof will make use of the following auxiliary definition.

▶ **Definition 3.4.** *Given a vertex subset* X *of a graph* H*, an* $\overline{X}$-dominating set *in* H *is a set* $S \subseteq V(H)$ *such that* $y \in N_H[S]$ *for all* $y \in V(H) \setminus X$*.*

We claim that S_2 is a minimum-weight $\overline{N_F(\mathfrak{f})}$-dominating set in $G[N^2_G[J]]$ with respect to weight function w. To see this, first of all note that S_2 is a $\overline{N_F(\mathfrak{f})}$-dominating set since it dominates every vertex in $F - \mathfrak{f} - N_F(\mathfrak{f}) = G[N^2_G[J]] - N_F(\mathfrak{f})$: the vertex $\mathfrak{f}$ that is removed from the dominating set S_2' in F only dominates vertices of $\{\mathfrak{f}\} \cup N_F(\mathfrak{f})$, so the rest is dominated by $S_2' \setminus \{\mathfrak{f}\} = S_2$. Now suppose by way of contradiction that there is an $\overline{N_F(\mathfrak{f})}$-dominating set D in $G[N^2_G[J]]$ with $w(D) < w(S_2)$. Then, since $w_F(\mathfrak{f}) = 0$, $w_F(D \cup \{\mathfrak{f}\}) = w(D) < w(S_2) = w_F(S_2 \cup \{\mathfrak{f}\})$ which contradicts the fact that $S_2 \cup \{\mathfrak{f}\}$ is a minimum dominating set on (F, w_F).

Now take any dominating set S_3 in G. Observe that $(S_3 \oplus^2_{G,J} S_1) \cap N^2_G[J]$ is an $\overline{N_F(\mathfrak{f})}$-dominating set in $G[N^2_G[J]]$ and thus, by what we just showed,

$$w\big((S_3 \oplus^2_{G,J} S_1) \cap N^2_G[J]\big) \geq w(S_2). \tag{2}$$

Using the fact that $S_1 \setminus J = (S_1 \setminus N^2_G[J]) \cup (S_1 \cap N^2_G(J))$, we thus have:

$$\begin{aligned}
w(S_3 \oplus^2_{G,J} S_1) &= w\big((S_1 \setminus J) \cup (S_3 \cap N^2_G[J])\big) && \text{(by def. of stitch)} \\
&= w(S_1 \setminus N^2_G[J]) + w\big((S_1 \cap N^2_G(J)) \cup (S_3 \cap N^2_G[J])\big) && \text{(by fact above)} \\
&= w(S_1 \setminus N^2_G[J]) + w\big((S_3 \oplus^2_{G,J} S_1) \cap N^2_G[J]\big) && \text{(by def. of stitch)} \\
&\geq w(S_1 \setminus N^2_G[J]) + w(S_2) && \text{(by Inequality (2))} \\
&= w(S_1 \setminus N^2_G[J]) + w\big((S_1 \cap N^2_G(J)) \cup S_2\big) && \text{(since } S_1 \cap N^2_G(J) \subseteq S_2) \\
&= w(S_1 \setminus N^2_G[J]) + w\big((S_1 \cap N^2_G(J)) \cup (S_2 \cap N^2_G[J])\big) && \text{(since } S_2 \subseteq V(F) \setminus \{\mathfrak{f}\}) \\
&= w\big((S_1 \setminus J) \cup (S_2 \cap N^2_G[J])\big) && \text{(since } N^2_G[J] \setminus N^2_G(J) = J) \\
&= w(S_2 \oplus^2_{G,J} S_1). && \text{(by def. of stitch)}
\end{aligned}$$

◀

From what we've seen so far in this section, we have that both Dominating Set and H-Subgraph-Free-Deletion are stitchable (by Lemma 3.1). Thus, since these problems lie in FPT parameterized by tree-width (as we recall for convenience in Theorem 3.5 below), we can conclude by Lemmas 3.2 and 3.3 that both H-Subgraph-Free-Deletion-Stitching and Dominating Set-Stitching also lie in FPT.

▶ **Theorem 3.5** ([4, 5]). *Given any weighted graph* $(G, w \colon V(G) \to \mathbb{N})$ *with tree-width at most* k*, we can solve:*

- *H-Subgraph-Free-Deletion in time* $2^{O(k)} \cdot |V(G)|^{O(1)}$ *when* H *is a clique [5],*

- *H-Subgraph-Free-Deletion in time $2^{O(k)^{\mu^*(H)} \log k} \cdot |V(G)|^{O(1)}$ when H is a connected graph that is not a clique [5], and*
- *Dominating Set in $2^{O(k)} \cdot |V(G)|^{O(1)}$ [4, page 176],*

where $\mu^*(H)$ for a connected graph H denotes the maximum, over all connected vertex sets $A \subseteq V(H)$ satisfying $N_H(N_H[A]) \neq \emptyset$, of the quantity $|N_H(A)|$.

Furthermore, since both H-Subgraph-Free-Deletion and Dominating Set are guessable, we can apply Theorem 3.5 to obtain (Corollary 3.6) polynomial time, *certified* algorithms for H-Subgraph-Free-Deletion and Dominating Set on minor-closed classes with bounded local tree-width.

▶ **Corollary 3.6.** *For any minor-closed graph class $\mathcal{C}$ of λ-linear local tree-width and each $\varepsilon > 0$ there are $(1+\varepsilon)$-certified algorithms solving Dominating Set and H-Subgraph-Free-Deletion whenever the input is of the form (G, w) with $G \in \mathcal{C}$ and $w \colon V(G) \to \mathbb{N}$ a polynomially-bounded weight function. Furthermore these algorithms respectively admit the following worst-case running-time bounds:*

- *$2^{O(\lambda/\varepsilon)} \cdot |V(G)|^{O(1)}$ in the case of H-Subgraph-Free-Deletion when H is a clique,*
- *$2^{O(k)^{\mu^*(H)} \log k} \cdot |V(G)|^{O(1)}$ (where $\mu^*(H)$ is the constant of Theorem 3.5 and k equals $\mathsf{diam}(H)\lambda/\varepsilon$) in the case of H-Subgraph-Free-Deletion for a connected graph H that is not a clique, and*
- *$2^{O(2\lambda/\varepsilon)} \cdot |V(G)|^{O(1)}$ in the case of Dominating Set.*

4 Proving Theorem 1.7

The proof of Theorem 1.7 takes inspiration from Baker's technique [2] for designing polynomial-time approximation schemes on planar graphs. It will occur in three steps: we will outline the algorithm in Section 4.2.1, show that it has the desired running time in Section 4.2.2, and prove its correctness in Section 4.2.3. However, before doing so we shall briefly establish a few useful definitions in Section 4.1 which will streamline the presentation of what follows.

4.1 Definitions for Theorem 1.7

Throughout we assume all graphs are *connected* unless stated otherwise and we denote any interval $\{a, a+1, \dots, b\}$ in $\mathbb{Z}$ as $[a, b]$; furthermore we denote by $\iota_{a,b}$ the obvious inclusion $\iota_{a,b} : [a, b] \hookrightarrow \mathbb{Z}$ given by $\iota_{a,b}(i) = i$ for $a \leq i \leq b$. Often, we shall refer to $\iota_{a,b}$ itself as an *interval*.

▶ **Definition 4.1** (m-boundary of an interval)**.** *Given any integer $m \geq 1$, we define the* left and right m-boundaries *of any interval $\iota_{a,b}$ to respectively be the intervals*

$$\delta_m^L(\iota_{a,b}) : [a-m, a-1] \hookrightarrow \mathbb{Z} \quad \text{and} \quad \delta_m^R(\iota_{a,b}) : [b+1, b+m] \hookrightarrow \mathbb{Z}.$$

We define the closed m-boundary *of $\iota_{a,b}$ as $\delta_m[\iota_{a,b}] : [a-m, a] \cup [a, b] \cup [b, b+m] \hookrightarrow \mathbb{Z}$ while the* open m-boundary *of $\iota_{a,b}$ is defined as $\delta_m(\iota_{a,b}) := \delta_m^L(\iota_{a,b}) \cup \delta_m^R(\iota_{a,b})$.*

Recall that, given any vertex v in a graph G, the *eccentricity* of v in G is the maximum length of a shortest path from v to any other vertex. Here we will denote this as $\hat{\epsilon}(v, G)$ or simply as $\hat{\epsilon}(v)$ if G is understood from context.

Ripples

When one drops a stone in a pond, an outward-radiating rippling ring of waves forms where the stone hit the surface of the water. In analogy to this phenomenon, we shall now define an r-*ripple*[1] in a graph as the sets of vertices (the waves, as it were) at fixed distances from some given vertex r.

▶ **Definition 4.2** (r-ripple)**.** *Given a vertex r in a graph G, we call the function*

$$\rho_r : \mathbb{Z} \to 2^{V(G)} \quad \textit{where} \quad \rho_r : i \mapsto \{x \in V(G) : d(r,x) = i\}$$

the r-ripple *in G. The vertex-subsets that make up a ripple will be referred to as* waves*: for any integer i, we define the i-th wave in ρ_r to be the set $\rho_r(i)$.*

In Definition 4.2 above, we call the vertex r the *center* of the ripple. If the center of the ripple is understood from context, then we simply denote the ripple as ρ. Notice that the i-th wave of a ripple will always be empty if i is negative or if it is greater than the eccentricity $\hat{\epsilon}(r)$ of the center of the ripple; one should think of such as "dummy" indices. Our choice to represent ripples as functions with domain $\mathbb{Z}$ is simply for notational convenience; indeed, one could instead restrict these functions to simply view any r-ripple as a function with domain $\{0, \dots, \hat{\epsilon}(r)\}$.

▶ **Definition 4.3** ((a,b)-subripple; see also Figure (1))**.** *Let ρ be an r-ripple in a graph G and $\iota_{a,b} : [a,b] \hookrightarrow \mathbb{Z}$ be an interval. We define the* (a,b)-subripple *in ρ to be the function $\rho_{a,b} : [a,b] \to 2^{V(G)}$ defined as the composite $\rho_{a,b} := \rho \circ \iota_{a,b}$. The* width *of a finite subripple is the number of waves it consists of (e.g. the width of an (a,b)-ripple is $|b-a+1|$).*

For any graph G and (a,b)-subripple of an r-ripple ρ in G, the graph $G[\bigcup_{a \le i \le b} \rho(i)]$ is a subgraph of the graph G' obtained from G by contracting all vertices v with $d_G(r,v) < a$ into r. As $\bigcup_{a \le i \le b} \rho(i)$ is contained in $N_{G'}^{b-a+1}[r]$, the tree-width of $G[\bigcup_{a \le i \le b} \rho(i)]$ is bounded in terms of the local tree-width of G'. Whenever G comes from a minor-closed graph class $\mathcal{C}$ of bounded local tree-width, we have $G' \in \mathcal{C}$ which ensures a bound on its local tree-width. This yields the following observation.

▶ **Observation 4.4** ([11])**.** *Let $\mathcal{C}$ be a minor-closed class of graphs which has λ-bounded linear local tree-width. If ρ is an r-ripple in a graph G belonging to $\mathcal{C}$, then the tree-width of any (a,b)-subripple of ρ is upper-bounded by* $\mathbf{tw}(G[\bigcup_{a \le i \le b} \rho(i)]) \le \lambda(|b-a+1|)$.

We note that one can of course use composition to generalize the notion of m-boundaries from intervals (Definition 4.1) to (sub)ripples. Indeed, we overload the notation so that, for example, the left m-boundary of any (a,b)-subripple $\rho_{a,b}$ is denoted $\delta_m^L(\rho_{a,b})$ and it is defined as the composite $\rho \circ \delta_m^L(\iota_{a,b})$. One can similarly define right, open and closed m-boundaries of any (a,b)-subripple.

In the rest of this section we shall make two further definitions related to simple constructions with ripples: *modular slices* of a ripple (Definition 4.5) and the *difference of a ripple and a modular slice* (Definition 4.6). Since both of these concepts are very easy to grasp visually, we defer their formal definitions and instead define them first "by picture" in Figure (1) below. The notation $S_2^{\mathrm{mod}\ 4}(i)$ in Figure (1) denotes the *i-th modular slice* (an evenly spaced sequence of subripples with a given start-index i) and the *difference* $\rho \ominus S_2^{\mathrm{mod}\ 4}(i)$ is simply the sequence of subripples that is "left-over" from ρ after we remove the modular slice $S_2^{\mathrm{mod}\ 4}(i)$.

[1] This is sometimes referred to as a "*layering*" in the literature.

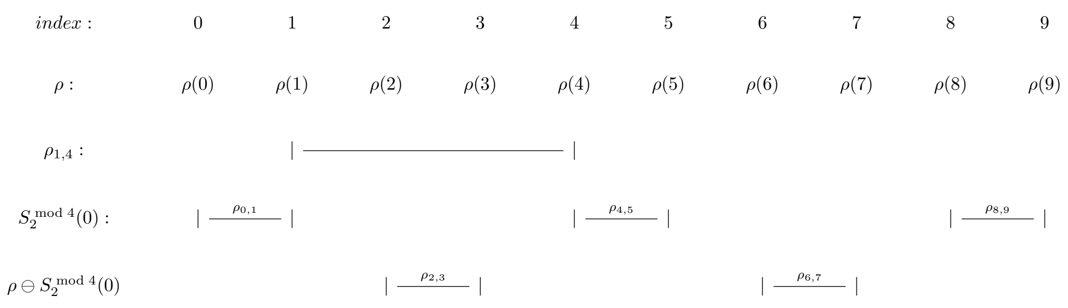

Figure 1 Illustration of subripples, modular slices, and remainders.

▶ **Definition 4.5** (modular slices). *Let ρ be an r-ripple in a graph G. For any integers s and k with $1 \leq s \leq k$ and any integer $i \in \{0, 1, \ldots, k-1\}$ define the set*

$$\mathbb{Z}_s^k(i) := \bigcup_{j \in \mathbb{Z}} \{j \cdot k + i, j \cdot k + i + 1, \ldots, j \cdot k + i + s - 1\}$$

and consider its obvious inclusion $\iota_{s,k,i} \colon \mathbb{Z}_s^k \hookrightarrow \mathbb{Z}$. We call the map $S_s^{\text{mod } k}(i) \colon \mathbb{Z}_s^k \to 2^{V(G)}$ defined as the composite $S_s^{\text{mod } k} := \rho \circ \iota_{s,k,i}$ the i-th modular k-slice of width s in ρ.

▶ **Definition 4.6** (Remainder). *Let s, k, and i be integers with $1 \leq s \leq k$ and $0 \leq i \leq k-1$. Let ρ be an r-ripple in a graph G and $S_s^{\text{mod } k}(i) \colon \mathbb{Z}_s^k \hookrightarrow 2^{V(G)}$ be a modular k-slice of width s in ρ. Letting $\overline{\iota_{s,k,i}} \colon \mathbb{Z} \setminus \mathbb{Z}_s^k \hookrightarrow \mathbb{Z}$ be the obvious inclusion of $\mathbb{Z} \setminus \mathbb{Z}_s^k$ into $\mathbb{Z}$, we define the* remainder *of $S_s^{\text{mod } k}(i)$ in ρ, denoted as $\rho \ominus S_s^{\text{mod } k}(i)$, to be the composite $\rho \circ \overline{\iota_{s,k,i}}$.*

To ease legibility and conciseness, throughout this document, we shall treat any subripple (resp. modular slice or difference thereof) as the union of all of its constituent waves whenever performing set-theoretic operations. For example, for any subset X of $V(G)$ and any (a, b)-subripple $\rho_{a,b}$, we shall simply write $X \cap \rho_{a,b}$ instead of $X \cap (\bigcup_{a \leq i \leq b} \rho(i))$.

Pigeonhole arguments on ripples and their modular slices

We will conclude this preliminary section by proving two auxiliary lemmas (Lemmas 4.7 and 4.9) which will be of use to us in the proof of Theorem 1.7. Intuitively, the next lemma says that for any weighted vertex set X in a graph G, when considering the modular k-slice of width s of a ripple in G, there will be an offset i such that the vertices contained in its waves at offset i contribute at most an $\frac{s}{k}$ fraction of the total weight of X.

▶ **Lemma 4.7.** *Let ρ be an r-ripple in a weighted graph $(G, w \colon V(G) \to \mathbb{R}_+)$ and let $k \geq 1$ be an integer. For any vertex set $X \subseteq V(G)$ and integer $1 \leq s \leq k$, there exists an integer $i \in \{0, \ldots, k-1\}$ such that $S_s^{\text{mod } k}(i)$ satisfies $w(X \cap S_s^{\text{mod } k}(i)) \leq \frac{s}{k} w(X)$.*

Proof. Seeking a contradiction, assume no such index i exists and hence conclude, by summing over each index $0 \leq j \leq k-1$, that

$$\sum_{0 \leq j \leq k-1} w(X \cap S_s^{\text{mod } k}(j)) > \sum_{0 \leq j \leq k-1} \frac{s}{k} w(X) = s \cdot w(X) \frac{1}{k} = s \cdot w(X). \tag{3}$$

However, since each non-empty wave of the ripple ρ is counted by exactly s out of k of the modular k-slices in the sum above, we can contradict the strictness of Inequality (3) by verifying that $\sum_{0 \leq j \leq k-1} w(X \cap S_s^{\text{mod } k}(j)) = \sum_{j' \in \mathbb{Z}} s \cdot w(X \cap \rho(j')) = s \cdot w(X)$. ◀

Now consider, for example, the difference $\rho \ominus S_2^{\text{mod } 4}(0)$ shown in Figure 1 above. It is easy to see (by inspection of the figure) that, after taking the closed 1-boundary of every subripple in $\rho \ominus S_2^{\text{mod } 4}(0)$, the domains of the resulting subripples partition[2] the domain of ρ. In this case, we say simply that the extended subripples *partition* ρ. We can state this more generally as the following observation.

▶ **Observation 4.8.** *Let $0 \leq 2s \leq k$ be integers and ρ be an r-ripple in a graph G. If $S_{2s}^{\text{mod } k}$ is a modular k-slice of width $2s$ in ρ, then $\left(\delta^s[\rho_{a,b}]\right)_{\rho_{a,b} \in \rho \ominus S_{2s}^{\text{mod } k}(i)}$ partitions ρ for any i.*

Observation 4.8 together with a pigeon-hole-like argument similar to that of the proof of Lemma 4.7 yields the following lemma. It applies to two vertex subsets I and S in a weighted graph G, which will later correspond to the solution S found by our algorithm and a solution I that is purported to be better. The lemma applies when the weight of $S \setminus I$ within the waves of a ripple ρ, on vertices outside the i-th modular k-slice of a certain width $2s$, is strictly larger than the weight of $I \setminus S$. It guarantees the existence of a *single* subripple $\rho_{a,b}$ with the following special property: the weight of $S \setminus I$ inside the subripple $\rho_{a,b}$ is strictly larger than the weight of $I \setminus S$ inside the *extended* subripple $\rho_{a-s,\dots,b+s}$. We will later use this lemma to argue that under certain conditions, a local optimization step can strictly improve the solution.

▶ **Lemma 4.9.** *Let ρ be an r-ripple in a weighted graph $(G, w \colon V(G) \to \mathbb{R}_+)$ and $S_s^{\text{mod } k}$ be the modular k-slice of width $2s$ in ρ. If there are sets $S, I \subseteq V(G)$ and an index i such that*

$$w((S \setminus I) \cap (\rho \ominus S_{2s}^{\text{mod } k}(i))) > w(I \setminus S), \tag{4}$$

then there exists $\rho_{a,b} \in \rho \ominus S_{2s}^{\text{mod } k}(i)$ satisfying $w((S \setminus I) \cap \rho_{a,b}) > w((I \setminus S) \cap \delta_s[\rho_{a,b}])$.

Proof. Seeking a contradiction, assume no such subripple $\rho_{a,b}$ exists; i.e. assume that

$$w((S \setminus I) \cap \rho_{a,b}) \leq w((I \setminus S) \cap \delta_s[\rho_{a,b}]) \tag{5}$$

for all subripples $\rho_{a,b}$ in the difference $\rho \ominus S_{2s}^{\text{mod } k}(i)$. Then we have

$$\begin{aligned} w((S \setminus I) \cap (\rho \ominus S_{2s}^{\text{mod } k}(i)) &= \sum_{\rho_{a,b} \in \rho \ominus S_{2s}^{\text{mod } k}(i)} w((S \setminus I) \cap \rho_{a,b}) && \text{(by definition)} \\ &\leq \sum_{\rho_{a,b} \in \rho \ominus S_{2s}^{\text{mod } k}(i)} w((I \setminus S) \cap \delta_s[\rho_{a,b}]) && \text{(by Equation (5))} \\ &= w(I \setminus S) \end{aligned}$$

where the last equality holds because $\left(\delta^s[\rho_{a,b}]\right)_{\rho_{a,b} \in \rho \ominus S_s^{\text{mod } k}(i)}$ is a partition of ρ (by Observation 4.8). However, this contradicts Inequality (4) as desired. ◀

4.2 Proof of Theorem 1.7

We are now finally ready to prove Theorem 1.7: we will first describe (Section 4.2.1) the algorithm mentioned in the statement of Theorem 1.7; then we shall establish its running time guarantees (Section 4.2.2) and finally its correctness (Section 4.2.3). Using the existence of an algorithm for Π-m-Stitching, it will be easy to describe our $(1+\varepsilon)$-certified algorithms; the main challenge lies in the proof that the solution it outputs is optimal for a $(1+\varepsilon)$-perturbation of the input.

[2] Notice that, although it is not drawn, $\rho(0)$ is in the 1-boundary of the element $\rho_{-1,-2}$ of $\rho \ominus S_2^{\text{mod } 4}(0)$.

4.2.1 The algorithm

Throughout the rest of the proof of Theorem 1.7 we shall let Π be a vertex-optimization problem as given in the statement of Theorem 1.7, i.e. it satisfies the following:

1. Π is guessable and m-stitchable for some given constant $m \in \mathbb{N}$, and
2. there exists an algorithm $\mathcal{A}$ that solves Π-m-Stitching in time $f(t)\cdot|V(G)|^{\mathcal{O}(1)}$, where $t = \mathbf{tw}(G[N_G^m[J]])$ and f is some computable function.

In what follows, given any feasible solution S on an instance (G, w) of Π and any (a, b)-subripple $\rho_{a,b}$ of an r-ripple ρ, we denote by $\mathcal{A}((G, w), S, \rho_{a,b})$ the output of running the algorithm $\mathcal{A}$ for Π-m-Stitching on inputs (G, w), the vertex set J of $\rho_{a,b}$, and the solution S. The algorithm is defined as follows.

- **Algorithm StitchAndCertify**
- **Input:** a (connected) vertex-weighted graph $(G, w\colon V(G) \to \mathbb{N})$ and $\varepsilon > 0$.
- **Output:** a vertex set $\hat{S} \subseteq V(G)$ and $(1+\varepsilon)$-perturbation w' of w such that $\hat{S}$ is an optimal solution for Π on (G, w').

```
1. Let $r \in V(G)$ be an arbitrary vertex and let $\rho$ be the $r$-ripple in $G$.
2. Let $\hat{S}$ be a feasible solution for $\Pi$ on $G$ (obtained by leveraging the guessability of $\Pi$; c.f. Definition 1.6).
3. Let $k = \lceil \frac{2m}{\varepsilon} \rceil + 2m$.
4. While there exists an $(a, b)$-subripple $\rho_{a,b}$ of width $k - 2m$ such that
       $w\big(\mathcal{A}((G, w), \hat{S}, \rho_{a,b})\big) < w(\hat{S})$          (6)
     - then replace $\hat{S}$ with $\mathcal{A}((G, w), S, \rho_{a,b})$.
5. Otherwise, return $(\hat{S},\ w' : V(G) \to \mathbb{R}_+)$ where $w'$ is defined as
       $w' : x \mapsto \begin{cases} w(x) & \text{if } x \in \hat{S} \\ (1+\varepsilon)w(x) & \text{otherwise.} \end{cases}$          (7)
```

4.2.2 Running time

Recall that the parameter for Π-Stitching is the tree-width of the closed m-neighborhood of the vertex set J along which we stitch. Each call to the algorithm $\mathcal{A}$ for Π-Stitching (Inequality 6) made inside StitchAndCertify runs on a subripple of width $k - 2m$. Hence the closed m-neighborhood of the subgraph along which we stitch is contained in $\delta^m[\rho_{a,b}]$, which is a subripple of width $k - 2m + 2m = k$. By Proposition 4.4 we know that $G[\delta^m[\rho_{a,b}]]$ has tree-width at most λk. These observations allow us to upper-bound the running time of each call to $\mathcal{A}$ by $f\big(\mathbf{tw}(G[\delta^m[\rho_{a,b}]])\big) \cdot |V(G)|^{\mathcal{O}(1)} \leq f\big(\lambda k\big) \cdot |V(G)|^{\mathcal{O}(1)}$.

Now, since all other lines of the algorithm clearly take polynomial time (recall that Π is guessable by Definition 1.6), the calls to $\mathcal{A}$ dominate the running time. Note that, for $W = \max_{x \in V(G)} w(x)$, the number of iterations in which we find a strictly better solution is bounded by $W \cdot n$: the weight of the initial solution is at most $W \cdot n$, all weights are integers, and the value cannot improve to below 0. Thus, since $k \in \mathcal{O}(m/\varepsilon)$ the entire algorithm runs in time at most $W \cdot f(\mathcal{O}(\lambda m/\varepsilon)) \cdot |V(G)|^{O(1)}$, as desired.

4.2.3 Proof of correctness

To prove that StitchAndCertify is indeed a $(1+\varepsilon)$-certified algorithm, we must show that the output $(\hat{S}, w'\colon V(G) \to \mathbb{R}_+)$ consists of an optimal solution $\hat{S}$ for Π on the instance (G, w') (which is clearly a $(1+\varepsilon)$-perturbation of the input (G, w)). It is easy to see that $\hat{S}$ is indeed a solution to Π, since it is initialized as a feasible solution and is only replaced by the output of $\mathcal{A}$, which is also a feasible solution by definition. Hence it suffices to prove optimality.

Assume, by way of contradiction, that $\hat{S}$ is not optimal. Then there is a solution I for Π on (G, w') such that

$$w'(\hat{S}) > w'(I) \implies w'(\hat{S} \setminus I) > w'(I \setminus \hat{S}) \implies w(\hat{S} \setminus I) > (1+\varepsilon)w(I \setminus \hat{S}) \tag{8}$$

(by the definition of w'; see Equation (7)). The remainder of this proof will rest on the following claim which states that not only is $w(\hat{S} \setminus I) > (1+\varepsilon)w(I \setminus \hat{S})$, but moreover, there exists an index of the modular slice whose intersection with $\hat{S} \setminus I$ has greater weight than that of $I \setminus \hat{S}$.

▷ Claim 4.10. For the given $\hat{S}$ and I, there exists an index $0 \leq i \leq k-1$ such that the preconditions of Lemma 4.9 are met for $s = m$; stating this explicitly, there is a choice of i such that $w((\hat{S} \setminus I) \cap (\rho \ominus S_{2m}^{\text{mod } k}(i))) > w(I \setminus \hat{S})$.

Claim 4.10 (whose proof we defer to the end of this section) enables us to apply Lemma 4.9 in order to find a "heavy" subripple; i.e. a subripple $\rho_{a,b} \in \rho \ominus S_{2m}^{\text{mod } k}(i)$ satisfying

$$w((\hat{S} \setminus I) \cap \rho_{a,b}) > w((I \setminus \hat{S}) \cap \delta_m[\rho_{a,b}]). \tag{9}$$

To aid the upcoming derivation, we now argue that the sets $A := \hat{S} \setminus \rho_{a,b}$, $B := (I \setminus \hat{S}) \cap \delta_m[\rho_{a,b}]$, and $C := I \cap \hat{S} \cap \rho_{a,b}$, form a partition of $D := (\hat{S} \setminus \rho_{a,b}) \cup (I \cap \delta_m[\rho_{a,b}])$. To see this, observe first that A, B, C are disjoint: $A \cap C = \emptyset$ since C lives inside $\rho_{a,b}$ but A outside; $A \cap B = \emptyset$ since A lives inside $\hat{S}$ but B outside; and $B \cap C = \emptyset$ since C lives inside $\hat{S}$ but B outside. To establish that A, B, C partition D, it therefore suffices to argue their union covers D. For this, the crucial insight is that those vertices of $I \cap \hat{S} \cap \delta_m[\rho_{a,b}]$ that are not contained in $I \cap \hat{S} \cap \rho_{a,b}$, belong to $\hat{S} \setminus \rho_{a,b}$ and therefore to A.

Using this property we now deduce

$$\begin{aligned}
w(\hat{S}) &= w\big(\hat{S} \setminus \rho_{a,b}\big) + w\big(\hat{S} \cap \rho_{a,b}\big) \\
&= w\big(\hat{S} \setminus \rho_{a,b}\big) + w\big((\hat{S} \setminus I) \cap \rho_{a,b}\big) + w\big(I \cap \hat{S} \cap \rho_{a,b}\big) \\
&> w\big(\hat{S} \setminus \rho_{a,b}\big) + w\big((I \setminus \hat{S}) \cap \delta_m[\rho_{a,b}]\big) + w\big(I \cap \hat{S} \cap \rho_{a,b}\big) && \text{(by Inequality (9))} \\
&= w\big((\hat{S} \setminus \rho_{a,b}) \cup (I \cap \delta_m[\rho_{a,b}])\big) && (A, B, C \text{ partition } D) \\
&\geq w\big((\hat{S} \setminus \rho_{a,b}) \cup (I \cap N_G^m[\rho_{a,b}])\big) && (\delta_m[\rho_{a,b}] \supseteq N_G^m[\rho_{a,b}]) \\
&= w(I \oplus_{G,\rho_{a,b}}^m \hat{S}) && \text{(by Definition 1.4).}
\end{aligned}$$

But then this means that the m-stitch of I onto $\hat{S}$ along the subripple $\rho_{a,b}$ of width $k-2m$ yields a solution (since Π is m-stitchable) whose weight under w is strictly better than $\hat{S}$. By definition of Π-m-Stitching, the output of $\mathcal{A}$ for the subripple $\rho_{a,b}$ is at least as good, thus satisfying Inequality (6) of StitchAndCertify. We conclude that the algorithm cannot possibly have terminated. Thus, since we have found our desired contradiction, all that remains to be done is to prove Claim 4.10.

Proof of Claim 4.10. Applying Lemma 4.7 on (G, w) and ρ with $X = (\hat{S} \setminus I)$ we obtain an index i such that modular k-slice $S_{2m}^{\text{mod } k}(i)$ satisfies

$$w((\hat{S} \setminus I) \cap S_{2m}^{\text{mod } k}(i)) \leq \frac{2m}{k} w(\hat{S} \setminus I). \tag{10}$$

Now observe that, by the definition of $\ominus$ (Definition 4.6), we have

$$\begin{aligned} w((\hat{S} \setminus I) \cap (\rho \ominus S_{2m}^{\bmod k}(i))) &= w(\hat{S} \setminus I) - w((\hat{S} \setminus I) \cap S_{2m}^{\bmod k}(i)) \\ &\geq w(\hat{S} \setminus I) - \frac{2m}{k} w(\hat{S} \setminus I) && \text{(by Inequality (10))} \\ &= \frac{k-2m}{k} w(\hat{S} \setminus I) \\ &> \frac{k-2m}{k}(1+\varepsilon) w(I \setminus \hat{S}) && \text{(by Inequality (8)).} \end{aligned}$$

Notice that, since we defined k as $k = \lceil \frac{2m}{\varepsilon} \rceil + 2m$ (Line 3 of StitchAndCertify), we must have $\varepsilon \geq \frac{2m}{k-2m}$. Combining this observation with our derivation above, we obtain precisely the desired inequality of Claim 4.10 as follows.

$$\begin{aligned} w((\hat{S} \setminus I) \cap (\rho \ominus S_{2m}^{\bmod k}(i))) &> \frac{k-2m}{k}(1+\varepsilon) w(I \setminus \hat{S}) \qquad \text{(from the derivation above)} \\ &\geq \frac{k-2m}{k}\left(1 + \frac{2m}{k-2m}\right) w(I \setminus \hat{S}) \ = \ w(I \setminus \hat{S}). \end{aligned}$$

This concludes the proof of Claim 4.10 and hence also the proof of Theorem 1.7. ◁

5 Discussion

Our main theorem allows us to obtain FPT-time *certified* algorithms for vertex-minimization problems such as H-Subgraph-Free-Deletion and Dominating Set (Corollary 3.6 of Theorem 1.7). However, as mentioned in Section 1, our results also apply to the *complementary* maximization problems simply by virtue of being certified algorithms. Inspired by Makarychev and Makarychev's notation [13], we define the notion of the *complementary* problem as follows.

▶ **Definition 5.1.** *Fix a vertex-minimization (resp. maximization) problem* Π *as in Definition 2.2. The* complementary vertex-maximization *(resp.* minimization*) problem is obtained by equipping the vertex-subset property that encodes feasibility for* Π *with following maximization (resp. minimization) objective: for any given vertex-weighted instance* $(G, w \colon V(G) \to \mathbb{N})$ *find a set* $S \subseteq V(G)$ *such that* $w(V(G) \setminus S)$ *is maximum (resp. minimum) subject to the requirement that* S *be feasible with respect to* Π.

Makarychev and Makarychev [13] discuss many examples of complementary problems; perhaps the prototypical example pair is Vertex Cover and Independent Set: every minimum vertex cover S in a graph G corresponds to a maximum independent set $V(G) \setminus S$.

Notice that one can deduce [13, Theorem 5.11] that any certified algorithm $\mathcal{A}$ for some problem Π is also an approximation algorithm for the complementary problem to Π; this is recalled below for completeness. Furthermore, it is easy to show a polynomial-time equivalence between certified algorithms for a problem and its complementary problem.

▶ **Theorem 5.2** ([13]). *If* $\mathcal{A}$ *is a* γ*-certified algorithm for a vertex-minimization (resp. maximisation) problem* Π*, then* $\mathcal{A}$ *is a* γ*-approximation algorithm for both* Π *and its complementary vertex-maximization (resp. minimization) problem.*

Recalling that Vertex Cover is just K_2-Deletion, we find that Corollary 3.6 yields a polynomial-time $(1+\varepsilon)$-certified algorithm for Independent Set on minor-closed graph classes of bounded local tree-width. This improves – in terms of generality and running time – on the XP-time $(1+\varepsilon)$-certified algorithm for Independent Set on planar graphs which was due to Angelidakis, Awasthi, Blum, Chatziafratis and Dan [1].

Further questions

As we mentioned in Section 1, any certified algorithm $\mathcal{A}$ for a problem Π happens to also be an approximation algorithm for both Π and its complementary problem $\Pi^{\mathfrak{c}}$. Thus a natural direction for future work is to seek $(1+\varepsilon)$-certified algorithms for other problems that admit efficient polynomial-time approximation schemes. In contrast, by Theorem 5.2, whenever either Π or $\Pi^{\mathfrak{c}}$ do not admit any EPTAS, then the question that we just posed is clearly not a viable direction for further work. Thus for such problems such as WEIGHTED PLANAR FEEDBACK VERTEX SET (for which, for instance, bidimensional techniques [10] do not apply) even simply finding XP-time certified algorithms can be a fruitful direction of research.

Another interesting direction for future research concerns the range of weight values. Our running-time analysis crucially relies on the assumption that the weights are non-negative integers of value at most $n^{\mathcal{O}(1)}$: this property ensures that the local search terminates after $n^{\mathcal{O}(1)}$ improvements. The algorithm by Angelidakis et al. [1] also requires polynomially bounded weights. Is it possible to give FPT-time $(1+\varepsilon)$-certified algorithms on inputs whose weights are encoded in $n^{\mathcal{O}(1)}$ bits, but may have value $2^{\Omega(n)}$?

References

1 H. Angelidakis, P. Awasthi, A. Blum, V. Chatziafratis, and C. Dan. Bilu-linial stability, certified algorithms and the independent set problem. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*, volume 144 of *LIPIcs*, pages 7:1–7:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ESA.2019.7.

2 B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, January 1994. doi:10.1145/174644.174650.

3 Y. Bilu and N. Linial. Are stable instances easy? *Comb. Probab. Comput.*, 21(5):643–660, September 2012. doi:10.1017/S0963548312000193.

4 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

5 Marek Cygan, Dániel Marx, Marcin Pilipczuk, and Michał Pilipczuk. Hitting forbidden subgraphs in graphs of bounded treewidth. *Information and Computation*, 256:62–82, 2017. doi:10.1016/j.ic.2017.04.009.

6 M. Delorme, S. García, J. Gondzio, J. Kalcsics, D. Manlove, and W. Pettersson. New algorithms for hierarchical optimisation in kidney exchange programmes. *Technical report ERGO 20–005, Edinburgh Research Group in Optimization*, 2020. URL: https://optimization-online.org/2020/10/8058/.

7 Erik D. Demaine and Mohammad Taghi Hajiaghayi. Equivalence of local treewidth and linear local treewidth and its algorithmic applications. In J. Ian Munro, editor, *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004*, pages 840–849. SIAM, 2004. URL: http://dl.acm.org/citation.cfm?id=982792.982919.

8 R. Diestel. *Graph theory*. Springer, 2010. ISBN:9783642142789.

9 David Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27(3):275–291, 2000. doi:10.1007/S004530010020.

10 Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. Bidimensionality and EPTAS. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011*, pages 748–759. SIAM, 2011. doi:10.1137/1.9781611973082.59.

11 M. Grohe. Local tree-width, excluded minors, and approximation algorithms. *Combinatorica*, 23:613–632, 2000. doi:10.1007/s00493-003-0037-9.

12 T. Hazan, G. Papandreou, and D. Tarlow. *Bilu-Linial Stability*, pages 375–400. The MIT Press, 2016.

13 Konstantin Makarychev and Yury Makarychev. Perturbation resilience. In Tim Roughgarden, editor, *Beyond the Worst-Case Analysis of Algorithms*, pages 95–119. Cambridge University Press, 2020. doi:10.1017/9781108637435.008.

14 Konstantin Makarychev, Yury Makarychev, and Aravindan Vijayaraghavan. Bilu-linial stable instances of max cut and minimum multiway cut. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014*, pages 890–906. SIAM, 2014. doi:10.1137/1.9781611973402.67.

15 David Manlove. *Algorithmics of matching under preferences*, volume 2. World Scientific, 2013.