Designing (1 + ε)-certified algorithms for planar optimization problems with local feasibility properties

Venne, Jaime C.

Award date:
2022

Department of Mathematics and Computer Science
Algorithms, Geometry & Applications

Designing $(1+\epsilon)$-certified algorithms for planar optimization problems with local feasibility properties

Master's thesis

J.C. Venne

11 July 2022

Supervision:
Dr. B.M.P. Jansen
Dr. Benjamin Merlin Bumpus

Assessment committee:
Dr. B.M.P. Jansen
Dr. M. Roeloffzen
Dr. S.P. Luttik

Credits: 30

Abstract

Angelidakis, Awasthi, Blum, Chatziafratis, and Dan [2] recently obtained a $(1+\epsilon)$-certified algorithm (with $\epsilon > 0$) for the planar maximum weight independent set problem with integer polynomially-bounded weights. The algorithm runs in $|V(G)|^{\Omega(1/\epsilon)}$ time and we deem its underlying combinatorial properties as difficult to generalize to similar problems. Therefore, we explored opportunities to obtain an algorithm with an improved running time, which can also be easily generalized to similar problems.

We contribute by proposing a $(1+\epsilon)$-certified algorithm that runs in $W \cdot f(m/\epsilon) \cdot |V(G)|^{O(1)}$ time for each problem on connected planar graphs with integer weights that satisfies the conditions of a notion we introduce as *m-locally planar optimized*. Here W equals the sum of the input weights, and $m \geq 1$ is equal to 1 for the minimum weight vertex cover problem, 2 for the minimum weight dominating set problem, and a constant for each fixed H of the minimum weight H-Subgraph-Deletion problem. A linear-time reduction from the minimum weight vertex cover problem obtains our $(1+\epsilon)$-certified algorithm for the maximum weight independent set problem. Our algorithm is inspired by Baker's technique [4] for building polynomial time approximation schemes for problems on planar graphs. Starting from a trivial solution, our iterative algorithm repeatedly makes local improvements in induced subgraphs that have a treewidth of $O(m/\epsilon)$. When no further improvements of this form can be made, we use a stronger notion of the pigeonhole principle to prove that the resulting solution is optimal for a $(1+\epsilon)$-perturbation of the weight function.

Acknowledgements

This thesis serves as the final part of my master program Computer Science and Engineering at Eindhoven University of Technology.

I would like to express my sincerest gratitude to my supervisors, and *certified* geniuses, Dr. Bart M.P. Jansen and Dr. Benjamin Merlin Bumpus for the supervision and many helpful discussions during the course of my thesis. Furthermore, I would like to thank Dr. Bart M.P. Jansen, Dr. Marcel Roeloffzen, and Dr. Bas Luttik for taking the time to assess my thesis and defence. Moreover, I would like to thank my mentor Dr. Wouter Meulemans for his guidance during my master program. Finally, I am very thankful to Dr. George Fletcher and Dr. Natasha Stash by allowing me to pursue my passion by granting me admission to the master program with my atypical background.

Contents

Chapter 1

Introduction

Many combinatorial optimization problems are *NP-complete*, meaning that it is unlikely that an algorithm exists that could find an optimal solution efficiently for general input (see e.g. [22] for an overview). For over 50 years, researchers have employed various ways to deal with the intractability of many combinatorial optimization problems, including efficient algorithms for restricted input classes (see e.g. [31]), approximation algorithms (e.g. [34] gives an overview), and Downey and Fellows [19] even spawned a new branch of complexity theory, namely that of parameterized complexity. Consequently, new perspectives and ideas are frequently introduced to study the worst-case, and beyond the worst-case behavior of various algorithms on different input. One of these ideas comes from the observation that sometimes there exists a *unique* optimal solution i.e. this solution is better than all other solutions for an instance to some combinatorial optimization problem. The question of exactly how much better this unique optimal solution is, inspired Bilu and Linial [6] to come up with the notion of γ-*stability* with $\gamma \geq 1$. It measures the behavior of the unique optimal solution as we *perturb* the weights by at most γ. More specifically, suppose we have weights w that maps the objects that we optimize over (e.g. vertices of a graph) to positive values, then we could multiply each individual weight by a factor between 1 and γ to obtain new weights w' (we say that w' is a γ-*perturbation* of w). Now should there exists a unique optimal solution for some instance to a combinatorial optimization problem with weights w, and this solution happens to be invariant to all possible γ-perturbations of w, then we refer to the instance as γ-*stable* (see Definition 2.13). A powerful use case of γ-stability is that it might uncover the *true* optimal solution when our weights are obtained with possible error. To elaborate, say that some heuristic was used to obtain w, where the weights are based on arbitrary modeling decisions e.g. we might use the L2-norm, instead of L1-norm without it being strictly necessary. Should the instance be γ-stable for a sufficiently large γ, then for any modeling decision that yields a γ-perturbation of w, the structure of the optimal solution does not change. Therefore, the users get strong justification that the solution really is the optimal solution that goes beyond their chosen weight function. Moreover, we could loosely say that, the higher the value of γ of a γ-stable instance is, the more *obvious* the structure of the optimal solution becomes. Consequently, algorithms have been proposed to solve γ-stable instances exactly (see e.g. [24] for a survey). These range from polynomial time algorithms for fixed values of γ, to algorithms that grow exponentially with respect to a parameter related to γ.

In this thesis, we study a recent type of algorithm design that is inspired by γ-stability, namely that of γ-*certified algorithms* (see Definition 2.14). Certified algorithms were recently introduced by Makarychev and Makarychev [28] in 2018. They take as input some instance to a weighted combinatorial optimization problem using weights w, and return a γ-*approximate solution* S, together with a γ-perturbation w' of w such that S is guaranteed to be optimal if we use the weights w' (we say that w' *certifies* S). Therefore, a γ-certified algorithm behaves like a γ-*approximation algorithm*, meaning that the solution S is at most a factor of γ away from the weight of an optimal solution to the original instance (see Theorem 2.15). However, its behavior implies even stronger guarantees, because should the instance be γ-stable, then S is guaranteed to be the optimal solution for the instance using the original weights w. Now γ-certified algorithms are also powerful when the user knows that the weights exhibit some bounded error with respect to γ. This allows the user to justify that the problem could be modeled using the weights w', rather than w, while providing the same accuracy for the model. Finally, as a consequence of γ-certified algorithms behaving like stronger γ-approximation algorithms, a γ-certified algorithm could also behave like a *polynomial approximation scheme* (PTAS) (see Definition 2.4). More precisely, a $(1+\epsilon)$-certified algorithm, where $\epsilon > 0$ is determined by the user, is guaranteed to return a $(1+\epsilon)$-approximate solution and even solves $(1+\epsilon)$-stable instances exactly. We would like to investigate whether PTASs could be extended to $(1+\epsilon)$-certified algorithms. Arora, Lund, Motwani, Sudan, and Szegedy [3] proved that many combinatorial optimization problems on general graphs do not allow for PTASs. However, there do exist many PTASs when we restrict the input to planar graphs [16]. Planar graphs are interesting to study, because these pop-up in many real-life problems [8], including various geographic problems, installing fiber-optic cables underneath roads, and even have applications in image processing.

The work by Angelidakis, Awasthi, Blum, Chatziafratis, and Dan [2] designed such a $(1+\epsilon)$-certified algorithm for the planar *maximum weight independent set* (MWIS) problem with strictly positive integer polynomially-bounded weights. Given a weighted graph (G, w), the MWIS problem seeks a maximum weight $S \subseteq V(G)$ such that no vertices in S are adjacent (see Definition 2.6). Note that we leave out 'strictly positive' from now on, because certified algorithms require that the weights are strictly positive (see Definition 2.14). The running time of the algorithm is of the form $|V(G)|^{\Omega(1/\epsilon)}$ for a planar graph G. Furthermore, the algorithm depends on non-trivial mechanics that we deem as difficult to generalize to similar problems (see Section 1.2). Therefore, we sought out to investigate whether we could design a new $(1+\epsilon)$-certified algorithm for the planar MWIS problem that would simultaneously allow for a more efficient running time, while also exhibiting an underlying combinatorial structure that could be generalized to similar problems.

1.1 Our contributions

We obtained a $(1+\epsilon)$-certified algorithm for MWIS problem when the input is a connected planar graph G with integer weights. The algorithm has a running time of $W \cdot f(1/\epsilon)|V(G)|^{O(1)}$, where f is some computable function and W is the sum of the input weights. Therefore, we propose a different algorithm that improves the running time of the existing $(1+\epsilon)$-certified algorithm [2] of the form $|V(G)|^{\Omega(1/\epsilon)}$ to FPT time parameterized by $1/\epsilon$ (see Definition 2.3).

Our $(1+\epsilon)$-certified algorithm follows from a linear-time reduction from a $(1+\epsilon)$-certified algorithm that we obtained for the planar *minimum weight vertex cover* (MWVC) problem with the same input conditions and running time. The MWVC problem seeks to find a minimum weight $S \subseteq V(G)$ such that each edge has at least one endpoint in S (see Definition 2.5). Furthermore, we extended this algorithm to also work for the planar *minimum weight dominating set* (MWDOM) problem. Given a weighted graph (G, w), the task is to find a minimum weight $S \subseteq V(G)$ such that each vertex is in S (inclusive) or has a neighbor in S (see Definition 2.9). We also made the algorithm work for the H-Subgraph-Deletion (H-S-Deletion) problem for any fixed connected graph H. Here the task is to find a minimum weight $S \subseteq V(G)$ such that its removal does not contain H as a subgraph (see Definition 2.8). Observe that H-S-Deletion problem is more general and contains the MWVC problem when H is a single edge. Given the similarities between the $(1+\epsilon)$-certified algorithms for each of these problems, we propose a general $(1+\epsilon)$-certified algorithm under the same input conditions that encompasses these problems, and more. We do so by introducing the notion of *m-locally planar optimized* (m-LPO for short) problems with integer $m \geq 1$. Problems that satisfy the conditions of m-LPO problems seek to find a minimum weight solution $S \subseteq V(G)$. Moreover, they are typically *union-closed* i.e. taking the union of any two solutions yields a feasible solution. In addition, they often have a *local structure*, meaning that checking if any solution is feasible can be achieved by testing a condition for every vertex by looking at its constant size closed-neighborhood. Note that we dedicate Chapter 3 entirely to m-LPO problems, so all details can be found there. We claim that each problem that satisfies the conditions of m-LPO for a constant m admits a $(1+\epsilon)$-certified algorithm for connected planar graphs with integer weights that runs in FPT time parameterized by $1/\epsilon$. We present this main result in our Meta-Theorem 5.9.

Theorem 5.9 (Meta-Theorem). *Given any vertex-optimization problem* Π *that is m-LPO and an instance* (G, w) *to* Π *such that* G *is a connected planar graph and* $w : V(G) \to \mathbb{Z}_{>0}$*, then Algorithm 3 invoked on* (G, w) *and* $\epsilon > 0$ *is a* $(1+\epsilon)$*-certified algorithm that runs in time* $W \cdot f(m/\epsilon) \cdot |V(G)|^{O(1)}$ *for some computable function* f *where* $W := \Sigma_{u \in V(G)} w(u)$.

Since for each of the above described problems there exists a constant m such that the problem is m-LPO (with the exception of the MWIS problem), we obtain Corollary 6.14.

Corollary 6.14. Algorithm 3 is a $(1+\epsilon)$-certified algorithm for the following optimization problems when the input (G, w) consists of a connected planar graph and $w : V(G) \to \mathbb{Z}_{>0}$:

- MWVC problem with a running time of $W \cdot 2^{O(1/\epsilon)}|V(G)|^{O(1)}$;
- For each fixed connected graph H, the H-S-Deletion problem can be solved in time $W \cdot 2^{O(1/\epsilon^c)}|V(G)|^{O(1)}$ for a constant $c \geq 1$;
- MWDOM problem with a running time of $W \cdot 2^{O(1/\epsilon)}|V(G)|^{O(1)}$;
- MWIS problem implicitly, by using Algorithm 6, with a running time of $W \cdot 2^{O(1/\epsilon)}|V(G)|^{O(1)}$,

with $W := \Sigma_{u \in V(G)} w(u)$.

The running time of our algorithm heavily relies on the notion of *treewidth* (see Definition 2.10). Treewidth measures how close a graph is to a *tree* (i.e. an acyclic graph), because it has been shown that many problems can be solved efficiently on trees [13, page 151]. Furthermore, the basic workings of our algorithm are inspired by Baker's technique [4] to obtain PTASs for problems on planar graphs (we formally introduce Baker in Section 2.3). Our algorithm starts from a trivial solution that is assumed to exist for a m-LPO problem, and proceeds to iteratively make local improvements in induced subgraphs with a treewidth of $O(m/\epsilon)$. When we can no longer find any improvements of this form, we apply a stronger notion of the pigeonhole principle to prove that the resulting solution is optimal for a $(1+\epsilon)$-perturbation of the weight function. Observe that since we use integer weights, we can find at most W improvements, hence its dependency in the running time. Furthermore, in order to find these local improvements, we employed existing FPT time algorithms parameterized by treewidth for each of the problems of Corollary 6.14. More specifically, those for the MWVC, MWIS, and MWDOM problems come from a book on Parameterized Algorithms [13, page 176], while the one for the H-S-Deletion problem comes from the work by Cygan, Marx, Pilipczuk, and Pilipczuk [14]. Note that for presentation purposes we have limited our presentation to connected planar graphs. We can easily generalize this technique to general planar graphs (see Section 7.1).

In conclusion, we have successfully improved the running time of the existing $(1+\epsilon)$-certified algorithm for the planar MWIS problem with integer polynomially-bounded weights, while also proposing a framework to obtain $(1+\epsilon)$-certified algorithms under the same input conditions for similar problems.

1.2 Related work

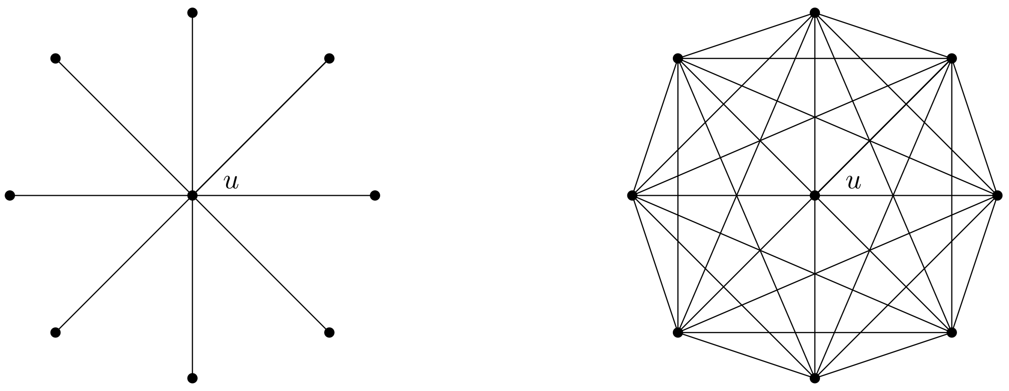
Most existing work studies γ-stability and γ-certified algorithms for graph partition problems (e.g. [2, 30]), but also for clustering problems (e.g. [10, 28]). A survey of most existing work for stability can be found in the work by Hazan, Papandreou, and Tarlow [24]. However, we would like to put some emphasis on one study in particular, namely the $(1+\epsilon)$-certified algorithm for the planar MWIS problem obtained by Angelidakis, Awasthi, Blum, Chatziafratis, and Dan [2]. This algorithm makes extensive use of the *Sherali-Adams* (SA) hierarchy of convex relaxations. In essence, SA is a hierarchy of convex relaxations of a integer program such that each round of SA gets the program closer to the integer hull. This allows us to obtain an optimal solution by solving the resulting linear program. Given the scope of this thesis, we do not introduce the SA hierarchy formally here, but instead refer the reader to a survey by Chlamtac and Tulsiani [9] for a complete introduction. Furthermore, we recommend the book by Conforti, Cornuéjols, and Zambelli [11], should the reader wish to read more about integer programming. The $(1+\epsilon)$-certified algorithm relies on Theorem 18 of the original paper [2] that follows as a corollary from Theorem 1.2 of the work by Bienstock and Ozbay [5]. Should we wish to generalize the $(1+\epsilon)$-certified algorithm to a new problem Π, then the scenario below must work.

Our problem Π takes as input some weighted graph (G, w). The problem is characterized by a set of linear constraints imposed onto G and stored in some matrix A. Next, we copy all vertices of G to a new graph G' and create an edge in G' for every pair of vertices in each individual constraint of A. We proceed to measure the complexity of G', rather than G by using the notion of treewidth (see Definition 2.10). We show the dependencies between G, A, and G' in the diagram below.

$$\begin{tikzcd}
\text{our approach complexity} \arrow[d] & & \\
G \arrow[rr, "\text{constraints of } \Pi \text{ for } G"'] \arrow[dr, "\text{copies vertices to } G'"'] & & A \arrow[dl, "\text{determines connectivity of } G'"] \\
& G' & \\
& \text{SA-based approach complexity} \arrow[u] &
\end{tikzcd}$$

It can be shown that this approach works for the planar MWIS problem, but not for e.g. the planar MWDOM problem. Here, graph G' can become too complex to obtain the same running times. We show an example in Figure 1.1 of how the graph G' can get a treewidth in the order of $|V(G)|$ for the MWDOM problem.

Hence, we deem its underlying combinatorial properties as difficult to generalize to similar problems. Therefore we focus on a different approach to obtain $(1+\epsilon)$-certified algorithms for combinatorial optimization problems on planar graphs. As mentioned previously, $(1+\epsilon)$-certified algorithms behave like stronger PTASs (see Theorem 2.15). Consequently, we would like to investigate whether PTASs can be extended to $(1+\epsilon)$-certified algorithms. Moreover, it is easy to see that the running time of a $(1+\epsilon)$-certified algorithm can only be as efficient as the tight lower-bound on the running time of a PTAS for the same problem. We refer to a PTAS as an *efficient* PTAS (EPTAS for short) when ϵ does not depend on the size of the input in the running time analysis (see Definition 2.4). Therefore, a $(1+\epsilon)$-certified algorithm could only exhibit FPT running time parameterized by $1/\epsilon$ if the problem admits an EPTAS. We proceed to discuss existing PTASs for combinatorial optimization problems on planar graphs.



(a) Graph G. (b) Graph G'.

Figure 1.1: Example of how G' becomes a complete graph when G is a star-graph for the MWDOM problem.

All the way back in the year of 1979, Lipton and Tarjan developed the first framework to obtain PTASs for problems on planar graphs [26, 27]. Their approach is based on planar separators, which recursively splits the graph into multiple pieces until each piece is of size $O(1/\epsilon)$. These individual pieces are then solved brute-force and all resulting solutions are merged to obtain an approximate solution. This approach leads to a PTAS for some problems e.g. the independent set and vertex cover problems, but not for the dominating set problem [23]. The technique has two important limitations, namely Demaine and Hajiaghayi [17] argue that separator-based approaches do not handle weighted variants of problems as well as other decompositions do. Second, it is subject to very large hidden constants in the size of the $O(1/\epsilon)$ pieces that are solved brute-force [18].

Now both of these limitations are addressed in Baker's technique to obtain PTASs for problems on planar graphs [4, 17]. Baker's technique was first announced in 1983 and published in 1994 by an author under the same name [4]. In brief, Baker's technique decomposes the planar graph into subproblems such that each subproblem exhibits a bounded treewidth of $O(1/\epsilon)$. It solves the problem exactly on each of these subproblems and proceeds to merge the solutions. The technique works on *local* problems, meaning that some solution is feasible if, and only if, some condition in the closed neighborhood of each vertex is satisfied [18]. The technique obtains EPTASs for e.g. the planar MWIS problem, the planar MWVC problem, and the planar MWDOM problem. This technique does not obtain PTASs for problems that exhibit more of a *global* structure such as the planar *feedback vertex set* (FVS) problem. More specifically, the FVS problem seeks a minimum size vertex set such that its removal yields an acyclic graph.

Now to arrive roughly at the state of the art for PTASs for problems on planar graphs, Demaine and MohammadTaghi [17] generalized and unified the Lipton-Tarjan separator approach with Baker's technique using results from bidimensionality theory (see [15] for an overview). This framework also yields EPTASs for global problems, including the planar FVS problem. More research based on bidimensionality theory has been conducted to encompass more problems that allow for EPTASs (e.g. [20, 21]). There is one important limitation to obtain $(1+\epsilon)$-certified algorithms from these EPTASs, namely it is not trivial to extend the framework to weighted problems, which is required for certified algorithms. This is posed as an open question in an overview of the applications of bidimensionality theory by Demaine and Hajiaghayi [15].

Given that these frameworks to obtain EPTASs begin with Baker's technique and Baker's technique appears to handle weighted cases well, it is natural for us to investigate if Baker's technique can be generalized to $(1+\epsilon)$-certified algorithms that run in FPT time parameterized by $1/\epsilon$. We dedicate Section 2.3 to Baker's technique to further elaborate upon its algorithmic details and we also show how it works for the planar MWIS problem.

1.3 Organization

We organized our work as follows. In Chapter 2 we give an overview of the preliminaries that our work depends on. We proceed to formally define m-LPO problems in Chapter 3. Furthermore, in Chapter 4 we elaborate upon the $(1+\epsilon)$-certified algorithm for a general m-LPO problem and proceed to prove its correctness and running time in Chapter 5. Moreover, in Chapter 6 we formally prove that for both the H-S-Deletion problem and the MWDOM problem, there exists an integer m such that the problem is m-LPO. In addition, we provide a $(1+\epsilon)$-certified algorithm for the planar MWIS problem by using a reduction from the MWVC problem. Finally, in Chapter 7 we reflect on our work and discuss possible extensions to the algorithm, and also elaborate upon limitations and future work.

Chapter 2

Preliminaries

In this chapter we provide important theory and notation. We specifically elaborate upon results of graph theory (see Section 2.1), formally introduce γ-stability and γ-certified algorithms (see Section 2.2), and finally introduce Baker's technique in general and present it for the MWIS problem (see Section 2.3).

2.1 Graph theory

We give a brief overview of important results and notation of graph theory. For any notation that is not explicitly defined here, we refer the reader to a book on graph theory by Bondy and Murty [32].

2.1.1 Basic results and notation

When we mention a graph G, we always refer to an *undirected graph*, meaning that all edges are bidirectional and the graph does not contain self-loops or parallel edges. Moreover, we denote the *vertex set* of a graph as $V(G)$ and the *edge set* as $E(G)$. We often associate the vertices of G with a *weight function* $w : V(G) \to \mathbb{R}$, yielding a *weighted undirected graph* as a pair (G, w). We refer to a weight function w as positive if $w : V(G) \to \mathbb{R}_{>0}$, and sometimes we restrict the range of the weight function to only positive integers i.e. $w : V(G) \to \mathbb{Z}_{>0}$. We continue to define the *shortest path distance* for a graph G as a function $d_G(u, v)$ that maps any two vertices $u, v \in V(G)$ to the minimum number of edges on a path between u and v. Should no such path exist (i.e. u and v are in different connected components of G), then the distance is set to infinity. Note that the weights are not used to compute the shortest path distance. Furthermore, we define the *diameter* of a connected graph G as $diam(G) := \max\{d_G(u, v) \mid u, v \in V(G)\}$, or in other words, the maximum shortest path distance between any pair of vertices in $V(G)$. We also mention the *eccentricity* of a vertex u in a graph G, which we define as the maximum shortest path distance from u to any other vertex of G. Moreover, let $X \subseteq V(G)$ be a vertex set, then we define its *induced subgraph* $G[X]$ as a new graph that is obtained after deleting all vertices of $V(G) \setminus X$ from G. We make a distinction between an induced subgraph of G and a *subgraph* of G, where the latter is a subgraph of G if, and only if, it can be obtained by deleting both vertices *and* edges from G. Furthermore, we say that a graph H is *isomorphic* to G if there exists a bijection f between vertex sets $V(H)$ and $V(G)$ such that for any pair of vertices $u, v \in V(G)$ we have $\{u, v\} \in E(G)$ if, and only if, $\{f(u), f(v)\} \in E(H)$. We also look at *graph-separators* of a graph G. Let $u, v \in V(G)$ such that $\{u, v\} \notin E(G)$ (non-adjacent), then we define a *(u, v)-separator* as a set $S \subseteq V(G) \setminus \{u, v\}$ such that u and v exist in different connected components of $G[V(G) \setminus S]$. We refer to S as a minimal (u, v)-separator if no proper subset of S is a (u, v)-separator. Observe that should H be a *clique* (all vertices of H are pairwise adjacent), then all vertices of $V(H)$ are pairwise adjacent and hence no separator exists in H. Furthermore, we make extensive use of the *open* and *closed m-neighborhood* of a subgraph $X \subseteq V(G)$, which we define formally in Definition 2.1.

Definition 2.1 (open and closed m-neighborhood)**.** Let G be a graph, $m \geq 1$ be an integer, and D_G be a function that takes as input a vertex $u \in V(G)$ and a vertex set $X \subseteq V(G)$ and computes the shortest path distance from u to any vertex of X i.e. $D_G(u, X) := \min\{d_G(u, v) \mid v \in X\}$. We define the *closed m-neighborhood* as

$$N_G^m[X] := \{u \in V(G) \mid D_G(u, X) \leq m\}.$$

Furthermore, we define the *open (ℓ, m)-neighborhood* as:

$$N_G^{m,\ell}(X) := \{u \in V(G) \mid \ell \leq D_G(u, X) \leq m\},$$

and should $\ell = 1$, then it is conventional to write $N_G^m(X)$.

Finally, there exist different graph classes that restrict the structure of an undirected graph G. These restrictions could allow for more assumptions on the input and, consequently, opportunities to design more efficient algorithms for combinatorial optimization problems on graphs [31]. We often deal with *planar graphs*, which we define formally as graphs that admit an embedding without any crossing edges. Much more efficient algorithms can be obtained for planar graphs [8], however, the planar versions of each of the problems that we discuss remain NP-complete [1, 2, 14]. Furthermore, a *tree* is a connected acyclic graph that, consequently, has a simple structure. Finally, we also mention *k-outerplanar graphs*, which we define as follows. A graph G is 1-outerplanar, if it is planar, and admits a *1-outerplanar embedding* i.e. an embedding such that all vertices lie on the exterior face. Recursively, for $k \geq 2$ a graph G is k-outerplanar, if it is planar, and admits a *k-outerplanar embedding* i.e. an embedding such that when all vertices on the outer face are deleted, a $(k-1)$-outerplanar embedding on the resulting graph is obtained. We are interested in k-outerplanar graphs, because these are known to have a bounded treewidth (see Definition 2.10) of $3k - 1$ [7]. This is exploited in e.g. Baker's technique (see Section 2.3) to obtain the EPTAS running time (see Definition 2.4).

2.1.2 Optimization over vertex sets

There are many combinatorial optimization problems that can be modeled on a weighted graph (G, w) (see e.g. [13, pages 581-598]). Given that we only discuss combinatorial optimization problems that seek to minimize (or maximize) the weight of a set of vertices of some weighted graph (G, w), we formally introduce the concept of a *vertex-optimization problem* in Definition 2.2.

Definition 2.2 (Vertex-optimization problem)**.** A *vertex-optimization problem* Π is modeled on any instance (G, w) such that G is any undirected graph and $w : V(G) \to \mathbb{R}$. Furthermore, Π is specified by a pair (f, g), where:

- f is a function that maps any undirected graph G to subsets of vertices of $V(G)$ that are feasible solutions to Π in G;
- g is the goal function, which is either a minimization or a maximization objective.

The function f is often encoded implicitly. Our goal is to find an optimal solution $I \in f(G)$ to Π in (G, w), which satisfies:

$$w(I) = g\{\Sigma_{u \in S} w(u) \mid S \in f(G)\}.$$

As mentioned previously, all vertex-optimization problems (except for possible instances of H in the H-S-Deletion problem) that we discuss are NP-complete, meaning that no efficient algorithm exists that solves the problem exactly. However, they do allow *fixed parameter tractable*

(FPT) time algorithms, which given its importance in our work, we formally introduce. We define a *parameterized problem* as a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a fixed, finite alphabet (we could e.g. encode a graph as a string over Σ). For an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, variable k is called the *parameter.* This parameter measures some aspect of the instance which is typically much smaller than the size of the input e.g. the number of vertices in some vertex cover. We define FPT problems in Definition 2.3. For more information about FPT problems and subtleties regarding its running time, we refer the reader to a book on parameterized algorithms [13, page 13].

Definition 2.3 (FPT)**.** A parameterized problem L is *fixed-parameter tractable* (FPT) if there exists an algorithm that can decide whether $(x, k) \in L$ in time $f(k) \cdot |x|^{O(1)}$ for some computable function f that only depends on k.

Furthermore, we mention γ*-approximate solutions* with $\gamma \geq 1$. Let I be an optimal solution to some vertex-optimization problem Π. Should Π have a maximization objective, then the weight of a feasible solution S to Π is called a γ*-approximate* solution if $w(S) \geq \gamma^{-1}w(I)$. Similarly, should Π have a minimization objective, then $w(S) \leq \gamma w(I)$. Furthermore, we often mention algorithms that behave like a *polynomial time approximation scheme* (PTAS), since our $(1+\epsilon)$-certified algorithms are effectively a stronger version of a PTAS. We define these formally within the context of a vertex-optimization problem in Definition 2.4.

Definition 2.4 (PTAS)**.** Let Π be any vertex-optimization problem, A be an algorithm that takes as input any instance (G, w) to Π that admits an optimal solution I, together with a parameter $\epsilon > 0$. Then, we refer to algorithm A as a *polynomial time approximation scheme* (PTAS) for Π (possibly on restricted input) if in polynomial time, for any fixed ϵ, algorithm A does the following:

- should Π have a minimization objective, then A returns a solution S that is within a factor $(1+\epsilon)$ of being optimal i.e. $w(I) \leq w(S) \leq (1+\epsilon)w(I)$;
- should Π have a maximization objective, then A returns a solution S that is within a factor of $(1-\epsilon)$ of being optimal i.e. $(1-\epsilon)w(I) \leq w(S) \leq w(I)$.

A PTAS with a running time of the form $f(1/\epsilon) \cdot |V(G)|^{O(1)}$ (i.e. ϵ does not depend on $|V(G)|$) for some computable function f is referred to as an *efficient polynomial time approximation scheme* (EPTAS).

We proceed to list the vertex-optimization problems that we use in our work. Note that we introduce these in the setting of finding an optimal solution to the problem, rather than the more traditional decision-problem setting. First, we introduce the *minimum weight vertex cover* (MWVC) problem in Definition 2.5.

Definition 2.5 (MWVC)**.** Given a weighted graph (G, w), the objective of the *minimum weight vertex cover* (MWVC) problem is to find a minimum weight $S \subseteq V(G)$ such that for each $\{u, v\} \in E(G)$, it holds that $\{u, v\} \cap S \neq \emptyset$.

Next, we formally introduce the *maximum weight independent set* (MWIS) problem in Definition 2.6.

Definition 2.6 (MWIS)**.** Given a weighted graph (G, w), the objective of the *maximum weight independent set* (MWIS) problem is to find a maximum weight $S \subseteq V(G)$ such that for each $\{u, v\} \in E(G)$, it holds that $\{u, v\} \not\subseteq S$.

It is well known that an optimal solution to the MWIS problem is the complement of an optimal solution for the MWVC problem and vice versa. We prove the first direction in Proposition 2.7.

Proposition 2.7. Given a weighted graph (G, w) with positive weights and an optimal solution C^* to the MWVC problem in (G, w), then $S^* := V(G) \setminus C^*$ is an optimal solution to the MWIS problem in (G, w).

Proof. We prove this claim as follows. First, we show that $C \subseteq V(G)$ is a vertex cover of G if, and only if, $S := V(G) \setminus C$ is an independent set of G. We start with the first direction, given a vertex cover C then by definition we have that every edge $\{u, v\} \in E(G)$ satisfies that at $\{u, v\} \cap C \neq \emptyset$. Therefore, should both u and v be in C, then neither will be in S and hence S is an independent set of G. Similarly, should we have an independent set S, then we know that for each edge $\{u, v\} \in E(G)$ that it cannot happen that both u and v are in S. Therefore, in the complement at least u or v must be in C, yielding a vertex cover of G. Now this shows that in order to obtain a MWIS S^*, we could solve the following expression: $\max\{w(V(G) \setminus C) \mid C \text{ is a vertex cover of } G\}$, which is upper-bounded by the MWVC C^*. Our claim follows. □

We proceed to introduce the other vertex-optimization problems that we mention. First, a generalization of the MWVC problem is the *H-Subgraph-Deletion* (H-S-Deletion) problem, which we define formally in Definition 2.8.

Definition 2.8 (H-S-Deletion)**.** For each connected graph H, the *H-Subgraph-Deletion* (H-S-Deletion) problem is defined as follows: given a weighted graph (G, w), the objective is to find a minimum weight $S \subseteq V(G)$ such that the graph $G[V(G) \setminus S]$ does not include H as a subgraph.

Next, we formally introduce the *minimum weight dominating set* (MWDOM) problem in Definition 2.9.

Definition 2.9 (WDOM)**.** Given a weighted graph (G, w), the objective of the *minimum weight dominating set* (MWDOM) problem is to find a minimum weight $S \subseteq V(G)$ such that for each $u \in V(G)$, we have that $N^1_G[u] \cap S \neq \emptyset$.

In algorithm design for optimization problems on graphs we often exploit the *treewidth* of a graph G. Intuitively, treewidth measures how close a graph is to a *tree* (i.e. an acyclic connected graph), because it has been observed that optimization problems on graphs are often easier to solve on trees [13, page 151]. Therefore, a tree has a treewidth of 1, and the further it is removed from a tree, the larger the treewidth becomes. The most famous result of exploiting treewidth to design algorithms is *Courcelle's theorem*, which was introduced by an author under the same name [12]. Its statement implies that for any vertex-optimization problem that can be expressed in *monadic second-order logic* (see [38] for an overview), an FPT time algorithm (see Definition 2.3) parameterized by treewidth exists that can solve the problem exactly. We define the notion of treewidth formally in Definition 2.10.

Definition 2.10 (treewidth [13])**.** A *tree-decomposition* of a graph G is a family $\{X_i \mid i \in I\}$ of subsets of $V(G)$ (called *bags*), together with a tree T with $V(T) = I$, satisfying the following properties:

- $\bigcup_{i \in I} X_i = V(G)$;
- every edge of G has both its ends in X_i for some $i \in I$;
- for all $v \in V(G)$, the set of nodes $\{i \in I \mid v \in X_i\}$ induces a subtree of T.

The *width* of a tree-decomposition is defined as $\max\{|X_i| - 1 : i \in I\}$. Now the *treewidth* of G is the minimum width of any tree-decomposition of G.

Furthermore, we need to know a tree decomposition such that its width is close to the treewidth to effectively use algorithms that exploit treewidth. Since we only use planar graphs as input to our algorithms, it suffices to provide an algorithm from literature that does so for planar graphs. Proposition 2.11 states that a tree decomposition with a width in the order of a known bound k of the treewidth of a planar graph can be found efficiently. Therefore, and also for the sake of presentation, we do not discuss the computation and requirements of tree decompositions in any future algorithms or running time analyses.

Proposition 2.11. [25] There exists an algorithm that takes as input a planar graph G with a bounded treewidth of $k \geq 1$, and in time $O(k^2 \log k \cdot |V(G)|)$ outputs a tree decomposition of G of width $O(k)$.

We proceed to present Theorem 2.12, which provides an FPT time algorithm parameterized by treewidth for each of the introduced vertex-optimization problems. Note that its tree decomposition computation is taken into consideration for each of the algorithms (i.e. we take $O(k)$ instead of k). The running time of the algorithm for the H-S-Deletion problem is designed for the unweighted case [14], but the analysis does not depend on it. Hence, it easily generalizes to the non-negative weighted case as well.

Theorem 2.12. Given any weighted planar graph (G, w) with non-negative weights and a bounded treewidth of k, then we can optimally solve the

- $MWVC$ problem in $2^{O(k)}k^{O(1)} \cdot |V(G)|^{O(1)}$ [13, page 176];
- $MWIS$ problem in $2^{O(k)}k^{O(1)} \cdot |V(G)|^{O(1)}$ [13, page 176];
- H-S-Deletion problem in time $2^{O(k^{\mu(H)})}|V(G)|^{O(1)}$, where $\mu(H) = 1$ if H is a clique and $\mu(H)$ is the maximum size of minimal separator in H, otherwise [14];
- $WDOM$ problem in $2^{O(k)}k^{O(1)} \cdot |V(G)|^{O(1)}$ [13, page 176].

Note that the algorithms from [13, page 176] do not exhibit large hidden constants in $O(k)$.

2.2 Stability and certified algorithms

In this section we formally introduce the notions of stability and certified algorithms.

2.2.1 Stability

The concept of γ-*stability* was first introduced by Yonatan Bilu and Nathan Linial [6]. However, we adopt a more refined version described in a recent book chapter by Makarychev, Makarychev, and Vijayaraghavan [29]. Let (G, w) be a graph with positive weights and $\gamma \geq 1$, then we define a γ-*perturbation* w' of w as a new weight function for $V(G)$ such that for all $u \in V(G)$ it holds that $w(u) \leq w'(u) \leq \gamma w(u)$. Equivalently, we can increase the weight of each individual vertex $u \in V(G)$ by at most $(\gamma - 1)w(u)$. Using the definition of γ-perturbations, we can formally define γ-stability in Definition 2.13.

Definition 2.13 (γ-stable)**.** Let Π be a vertex-optimization problem that admits a unique optimal solution I for instance (G, w), where w is a positive weight function. Then (G, w) is referred to as γ-*stable* if I remains the optimal solution for all possible (G, w') such that w' is a γ-perturbation of w.

Testing whether an instance to some combinatorial optimization problem is γ-stable is often a difficult task [29]. However, it has been observed that in practice it is not uncommon that instances are γ-stable and, consequently, this is the initial motivation to study γ-stable instances [6]. Sometimes it does happen that some local part of an optimal solution changes when the weights are perturbed by at most γ, meaning that the instance would almost be γ-stable [30]. Therefore, a more relaxed (and practical) notion of stability was introduced , namely *weak-stability* [30]. We do not introduce weak-stability formally here, but its practicality deserves a mention. We refer the reader to book chapter by Makarychev, Makarychev, and Vijayaraghavan [29, page 96] for a formal introduction.

2.2.2 Certified algorithms

The high-level idea of a γ-certified algorithm with $\gamma \geq 1$ is that given any instance (G, w) of some family of instances $\mathcal{F}$ for which some algorithm A is a γ-certified algorithm, A returns a solution S of G in Π together with a γ-perturbation w' of w such that S is optimal for (G, w'). In our case $\mathcal{F}$ is equal to all connected planar graphs with positive integer weights. We formally define γ-certified algorithms in Definition 2.14.

Definition 2.14 (γ-certified algorithm [29])**.** Let Π be a vertex-optimization problem and $\mathcal{F}$ be a family of instances that Π admits such that each instance exhibits strictly positive weights. We define a *γ-certified algorithm* for Π restricted to family $\mathcal{F}$ as an algorithm that takes as input any $(G, w) \in \mathcal{F}$ and outputs a solution S to Π in G together with a γ-perturbation w' of w such that S is an optimal solution for (G, w'). We say that w' certifies S.

Informally, we take as input some graph G with strictly positive weights w such that G has the characteristics of some graph family $\mathcal{F}$ (e.g. $\mathcal{F}$ could be characterized by all graphs of bounded degree of 3, that are planar, or all graphs without any odd cycles). Then, a $(1+\epsilon)$-certified algorithm utilizes some procedure with respect to w to obtain a γ-perturbation w' of w, together with a solution S of vertex-optimization problem Π such that S is an optimal solution for the instance (G, w'). Furthermore, these procedures could be made more explicit by using one of the two techniques to design $(1+\epsilon)$-certified algorithms proposed in the recent book chapter by Makarychev, Makarychev, and Vijayaraghavan [29, page 101]. We do not introduce these two techniques here, because we did not follow them to obtain our $(1+\epsilon)$-certified algorithms. Therefore, we expect that the possibilities for these procedures are much more rich. Furthermore, γ-certified algorithms are one of the techniques to exploit γ-stable instances. More specifically, should (G, w) be γ-stable, then A correctly finds the optimal solution, but is not explicitly aware of this, meaning that w' does not have to be equal to w. Given that these results are essential for certified algorithms, we prove these in formally in Theorem 2.15.

Theorem 2.15 ([29], Theorem 5.11)**.** Consider a γ-certified algorithm A for some vertex-optimization problem Π on a weighted graph (G, w) with $w : V(G) \to \mathbb{R}_{>0}$, then:

- A finds a γ-approximate solution for (G, w).
- If the instance is γ-stable, then A returns the optimal solution.

Proof. Let I be an optimal solution in (G, w) for Π. Moreover, by definition of a γ-certified algorithm, A returns a solution S together with a γ-perturbation w' of w such that S is optimal for (G, w'). First, we prove that A always returns a γ-approximate solution S by applying case distinction on the goal functions that Π could have i.e. minimization and maximization objectives. Let us start by assuming that Π has a maximization objective, then we observe the following:

$$\begin{aligned} w(S) &\geq \frac{1}{\gamma} w'(S) && \text{since } w(u) \leq w'(u) \leq \gamma w(u) \text{ for all } u \in V(G) \\ &\geq \frac{1}{\gamma} w'(I) && \text{because } S \text{ is optimal for } (G, w') \\ &\geq \frac{1}{\gamma} w(I) && \text{since } w(u) \leq w'(u) \text{ for all } u \in V(G). \end{aligned}$$

We conclude that S is a γ-approximate solution for the maximization objective. We continue to the case where Π has a minimization objective and find that

$$\begin{aligned} w(S) &\leq w'(S) && \text{since } w(u) \leq w'(u) \text{ for all } u \in V(G) \\ &\leq w'(I) && \text{because } S \text{ is optimal for } (G, w') \\ &\leq \gamma w(I) && \text{since } w(u) \leq w'(u) \leq \gamma w(u) \text{ for all } u \in V(G). \end{aligned}$$

We conclude that S is a γ-approximate solution for both the maximization and minimization objectives of problem Π.

We continue to prove the second property. Let us assume that (G, w) is γ-stable. By assumption, I is the unique optimal solution for all possible γ-perturbations of w. Since w' is one of such γ-perturbations, we must have that I is an optimal solution for both (G, w) and (G, w'). We conclude that both properties hold and so our claim must be true. □

We have introduced the most important properties of γ-stability and γ-certified algorithms for our purposes. For any supplementary material we refer the reader to the book chapter by Makarychev, Makarychev, and Vijayaraghavan [29].

2.3 Baker's technique

We introduce Baker's technique [4] by elaborating upon its general approach and by providing an example for the MWIS problem on connected planar graphs. Baker's technique exploits the (outer)planarity of a graph to design PTASs (see Definition 2.4) for various optimization problems on planar graphs. Examples of applications of Baker's technique include the MWIS problem, MWVC problem, and the MWDOM problem. The technique is named after Brenda Baker, who first announced it during a conference in 1983, and later published it in 1994 in the *Journal of the ACM* [4].

On a high level Baker's technique selects sets of vertices by using a periodic function with $O(1/\epsilon)$ distinct offsets (i.e. distinct positions to start the cycle). Should we have a maximization objective, then for each set it removes these vertices and solves the problem optimally on the remaining graph. If we have a minimization objective, then it creates subproblems by considering every vertex in the selected set exactly twice, while considering the remaining vertices once. This implies that we *under-count* in the maximization objective, and *over-count* in the minimization objective. The key correctness property comes from the famous *pigeonhole principle*, where we claim that at least one of the ways we the periodic function selects the vertices, we over-count (or under-count) at most ϵ times the weight of an optimal solution I. This implies that at least one of the solutions has an error of at most $\epsilon w(I)$. Recall that the pigeonhole principle states that '*Suppose that n items are put into m containers, with $n > m$, then at least one container must contain more than one item*'. The statement is trivial and dates back as far as 1624 [35], but can yield very powerful results. Both Baker and our technique make extensive use of the pigeonhole principle. Finally, Baker's technique also ensures that both the remaining graph and each subproblem have a treewidth of $O(1/\epsilon)$, meaning that we can use an FPT time algorithm parameterized by the treewidth to solve the problems exactly.

2.3.1 Example MWIS problem

We formally introduce Baker's technique for the MWIS problem (see Definition 2.6) on connected planar graphs. Our presentation is inspired by a lecture on graph theory [37], but slightly adapted and generalized to the weighted case of the independent set problem i.e. the MWIS problem. The pseudocode of the algorithm can be found in Algorithm 1.

Algorithm 1 Baker's technique for `MWIS` problem on connected planar graphs

```
    Input connected planar graph $G$, weight function $w : V(G) \to \mathbb{R}_{\geq 0}$ , error parameter $\epsilon > 0$
    Output $(1 + \epsilon)$-approximate independent set $S$ of $G$
1: $k \leftarrow \lceil \frac{1}{\epsilon} \rceil$
2: Let $r$ be an arbitrary vertex of $G$.
3: for $i \leftarrow 0, \dots, k-1$ do
4:     $V_i \leftarrow \{v \in V(G) \mid d(r, v) \pmod{k} \equiv i\}$
5:     $G_i \leftarrow G[V(G) \setminus V_i]$
6:     Let $S_i$ be an optimal solution to the MWIS problem for $(G_i, w)$ by the algorithm obtained
        from Theorem 2.12.
7: end for
8: $S \leftarrow \max\{w(S_i) \mid i \in \{0, \dots, k-1\}\}$
9: return $S$
```

The algorithm takes as input a weighted connected planar graph (G, w) together with a user-defined error parameter $\epsilon > 0$. It starts with setting $k \leftarrow \lceil \frac{1}{\epsilon} \rceil$, which determines the worst-case error guarantee of our solution. More specifically, the higher the value of k, the fewer vertices we remove, the better the approximation ratio of our independent set becomes. Next, we select an arbitrary vertex $r \in V(G)$ and continue to a for-loop with $i \in \{0, \dots, k-1\}$. During each iteration we select a set of vertices $V_i \subseteq V(G)$ that includes all vertices $v \in V(G)$ such that the shortest path distance from r to v modular k is congruent to i. This is our periodic function with offset i. We show the construction of each V_i in Figure 2.1.

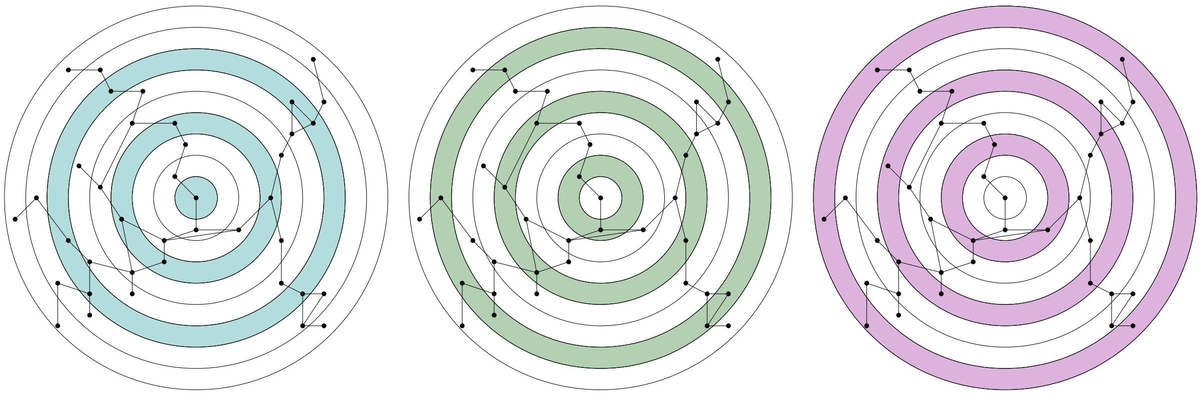

(a) V_0 (blue areas). (b) V_1 (green areas). (c) V_2 (purple areas).

Figure 2.1: Example of how the periodic function selects all vertices from a distance 0, 3, 6 (a), 1, 4, 7 (b), and 2, 5, 8 (c) away from vertex $r \in V(G)$ that lives isolated in the center, for some graph G and $k = 3$.

Next we construct the remaining graph G_i obtained from G by removing all vertices of V_i. We proceed to solve the MWIS problem exactly on (G_i, w) to obtain an independent set S_i using the FPT-time algorithm parameterized by the treewidth stated in Theorem 2.12. The fundamental reason that Baker's technique works is because each G_i has a bounded treewidth of $3k - 1$ [4]. The proof is out of scope for our purposes, but the key argument (loosely) comes from the way G_i is constructed. More specifically, each connected component becomes a k-outerplanar graph, which is known to have a treewidth of at most $3k - 1$ [7]. Consequently, we can solve the MWIS problem on G_i in FPT time parameterized by $O(1/\epsilon)$ since $k = O(1/\epsilon)$. We illustrate the construction of G_i and the computation of S_i in Figure 2.2.

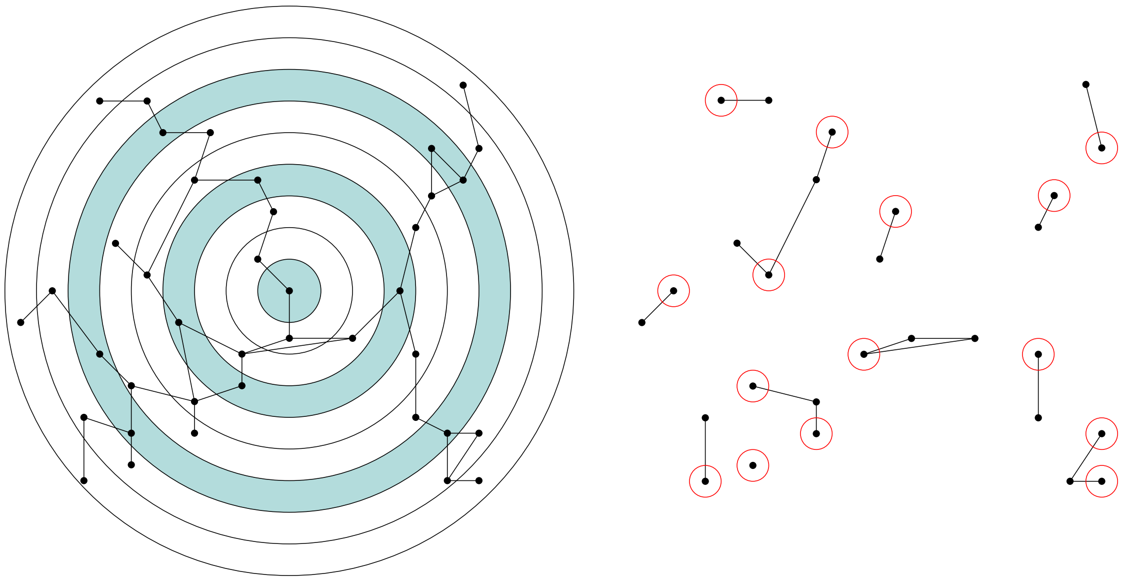

(a) V_0 (blue areas). (b) MWIS S_0 of G_0.

Figure 2.2: Example of how $G_0 := G[V(G) \setminus V_0]$ of a graph G is constructed with V_0 for $k = 3$, and in addition, how the MWIS S_0 is computed on G_0. Observe that $r \in V(G)$ lives isolated in the center. All vertex weights are 1.

After our for-loop terminates, we return an independent set S of maximum weight from all of the k computed independent sets i.e. $S := \max\{w(S_i) \mid i \in \{0, \dots, k-1\}\}$. We continue to formally prove that Algorithm 1 yields a PTAS for the MWIS problem on connected planar graphs. Let us first show a simple result of applying the pigeonhole principle over all V_i constructed by the algorithm in Lemma 2.16.

Lemma 2.16. Let (G, w) be a weighted connected planar graph and let I be an optimal MWIS of (G, w), then there exists an $i \in \{0, \dots, k-1\}$ such that the set $V_i := \{v \in V(G) \mid d(r, v) \pmod{k} \equiv i\}$ satisfies

$$w(I \cap V_i) \leq \frac{1}{k} w(I).$$

Proof. Let us, for the sake of contradiction, assume that for each $V_i = \{v \in V(G) \mid d(r, v) \pmod{k} \equiv i\}$ with $i \in \{0, \dots, k-1\}$ it holds that $w(I \cap V_i) > \frac{1}{k} w(I)$, then we have that:

$$\begin{aligned} \sum_{j=0}^{k-1} w(I \cap V_i) &> \sum_{j=0}^{k-1} \frac{1}{k} w(I) && \text{by assumption} \\ &= w(I) \end{aligned}$$

However, all V_i are pairwise disjoint and it is clear to see that its disjoint union is $V(G)$. Therefore, this would imply that $w(I) > w(I)$, which cannot happen. We have reached a contradiction and so our claim must be true. □

We need a formal statement that all connected components of G_i for all $i \in \{0, \dots, k-1\}$ have a treewidth of $O(k)$ to obtain the required PTAS running time. Again, this proof is beyond the scope of this thesis and we refer the reader to the original paper [4] for formal arguments as to why this holds. Note that the lemma is not directly proven in the original paper [4], but is well known to be the key reason Baker's technique exhibits the PTAS running time (for a similar result see [33, Theorem 7.6]).

Lemma 2.17. ([4], cf. [33, Theorem 7.6]) Given a connected planar graph G, then for any $r \in V(G)$ and any $i \in \{0, \dots, k-1\}$ we have that every connected component of $G_i := G[V(G) \setminus V_i]$ with $V_i = \{v \in V(G) \mid d(r, v) \pmod{k} \equiv i\}$ has a treewidth of at most $3k - 1$.

Now we can formally prove that Algorithm 1 is a PTAS for the MWIS problem on connected planar graphs in Theorem 2.18.

Theorem 2.18. Algorithm 1 is a PTAS for the MWIS problem on connected planar graphs and runs in time

$$2^{O(1/\epsilon)}(1/\epsilon)^{O(1)} \cdot |V(G)|^{O(1)}.$$

Proof. Let (G, w) be a connected weighted planar graph and I be an optimal MWIS of (G, w). We start by applying Lemma 2.16 on (G, w) and I to obtain a set V_i with $i \in \{0, \dots, k-1\}$ such that

$$w(V_i \cap I) \leq \frac{1}{k} w(I).$$

Algorithm 1 trivially finds this set V_i, because it loops over all possible V_i. We proceed to show that S_i is guaranteed to yield at most $\epsilon w(I)$ error.

$$w(S_i) \geq w(I \cap V(G_i))$$

since S_i is optimal for (G_i, w) (line 6) and $I \cap V(G_i)$ is an independent set of (G_i, w)

$$= w(I) - w(I \cap V_i)$$

by definition of $G_i = G[V(G) \setminus V_i]$

$$\geq w(I) - \frac{1}{k} w(I)$$

by assumption on the weight of the selected V_i

$$\geq w(I) - \epsilon w(I)$$

since $k := \left\lceil \frac{1}{\epsilon} \right\rceil$ implies that $\frac{1}{k} \leq \epsilon$

$$= (1 - \epsilon) w(I).$$

We conclude that Algorithm 1 indeed yields a solution S with an error of at most $\epsilon w(I)$ for the MWIS problem on connected planar graphs.

We proceed to prove that the running time of Algorithm 1 is polynomial for any fixed $\epsilon > 0$. It is clear to see the running time of the algorithm is dominated by line 6, which we compute k times. Now by Lemma 2.17, we know that the treewidth of each connected component of any G_i with $i \in \{0, \dots, k-1\}$ computed by the algorithm has a treewidth of at most $3k - 1$. Furthermore, by proposition 2.12, we know that an exact algorithm for the MWIS problem parameterized by treewidth k exists that runs in time

$$2^{O(k)} k^{O(1)} \cdot |V(G)|^{O(1)}.$$

We conclude that Algorithm 1 runs in time

$$\begin{aligned} |V(G)|^{O(1)} + k \cdot 2^{O(k)} k^{O(1)} \cdot |V(G)|^{O(1)} &= 2^{O(k)} k^{O(1)} \cdot |V(G)|^{O(1)} \\ &= 2^{O(1/\epsilon)} (1/\epsilon)^{O(1)} \cdot |V(G)|^{O(1)} \quad \text{since } O(k) = O(1/\epsilon). \end{aligned}$$

We have proven both the approximation ratio of solution S that algorithm 1 returns and that its running time is polynomial for any fixed $\epsilon > 0$. Hence, Algorithm 1 is a PTAS for the MWIS problem on connected planar graphs. □

We have introduced all the necessary preliminaries for the reader not to miss any context in any work of this thesis. In the next chapter we formally introduce what m-LPO problems are.

Chapter 3

Locally planar optimized problems

In this chapter we identify a class of vertex-optimization problems that we refer to as *m-locally planar optimized* problems or m-LPO for short (see Definition 3.3). Here m is a positive integer, whose value depends on the vertex-optimization problem in question. Each such problem Π takes as input some weighted graph (G, w) and allows to be modified locally along any induced subgraph G' of G by a property that we introduce as *m-stitching* (see Definition 3.2). This property effectively serves as the backbone of any m-LPO problem. Moreover, Π requires the existence of an efficient algorithm to improve a given solution to Π in G, by modifying it along a given induced subgraph G' of G, provided that an improvement of this form is possible. Here the efficiency is expressed in terms of the treewidth (see Definition 2.10) of the graph induced by vertex set $N_G^m[V(G')]$ (see Definition 2.1). In Chapter 5 we prove that all such problems allow for $(1+\epsilon)$-certified algorithms when the input graph is a connected planar graph and the vertex-weights are positive integers.

We start by introducing the notions of m-stitching and m-LPO problems formally in Section 3.1, then we apply this theory by showing that the MWVC problem is 1-LPO in Section 3.2.

3.1 Stitching and locally planar optimized problems

We begin with introducing the *m-stitch operation* of two vertex sets in Definition 3.1.

Definition 3.1 (m-stitch operation)**.** Given an integer $m \geq 0$, an induced subgraph G' of a graph G, and vertex sets $S_1, S_2 \subseteq V(G)$, then we define the *m-stitch* of S_2 onto S_1 along G' as the set

$$S_3 := (S_1 \setminus V(G')) \cup (S_2 \cap N_G^m[V(G')]).$$

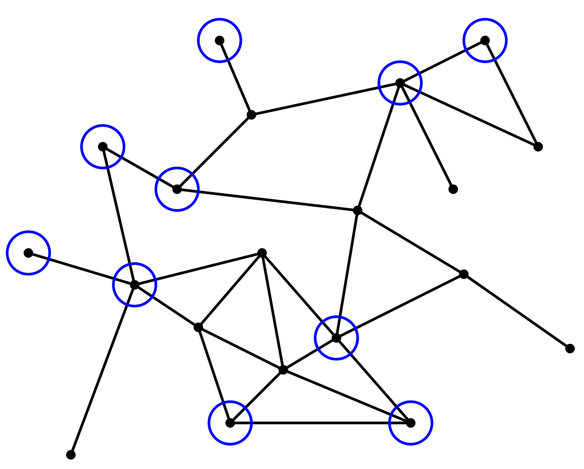
We refer to the *m-stitch operation* of replacing S_1 with S_3 obtained from the m-stitch of S_2 onto S_1 along G'.

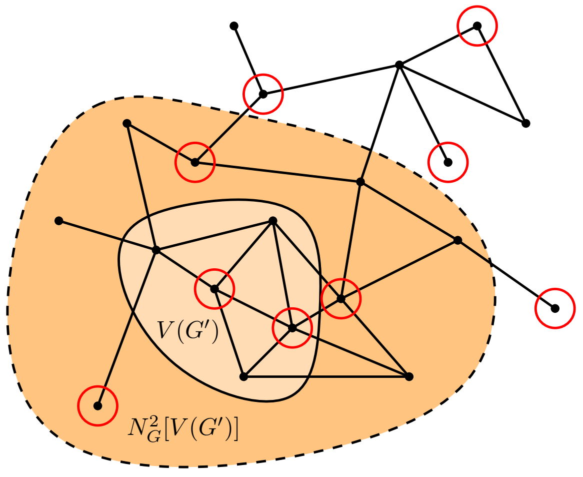
Let G be any graph, G' be any induced subgraph of G, and S_1 and S_2 be any vertex sets of $V(G)$, then intuitively, we take S_1 and proceed to *stitch* S_2 onto S_1. This stitching process occurs locally along $V(G')$ and its m-open neighborhood $N_G^m(V(G'))$, implying that S_1 and S_3 are equal in the remaining graph i.e. $(S_3 \setminus N_G^m[V(G')]) = (S_1 \setminus N_G^m[V(G')])$. We like to divide this operation into two steps, namely:

Step 1: Remove all vertices of $V(G')$ from S_1 to obtain $(S_1 \setminus V(G'))$.

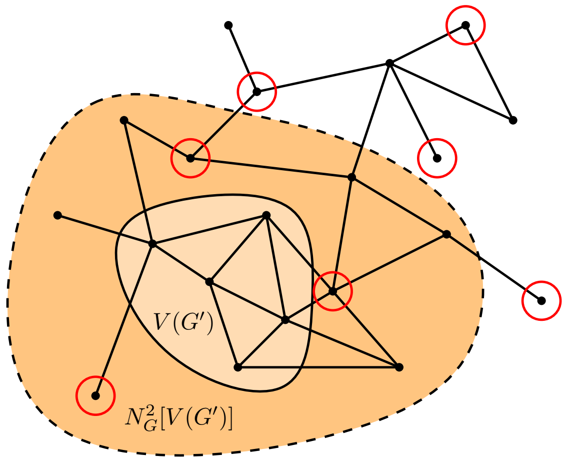
Step 2: Add all vertices of $(S_2 \cap N_G^m[V(G')])$ to $(S_1 \setminus V(G'))$ to obtain S_3.

We illustrate the m-stitch operation according to these steps in Figure 3.1 by showing the 2-stitch of $S_2 \subseteq V(G)$ onto $S_1 \subseteq V(G)$ along an induced subgraph G' of a graph G.

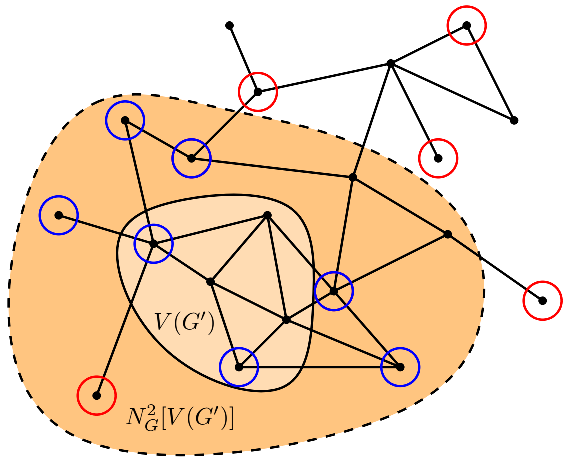

(a) Vertex set S_2.



(b) Vertex set S_1 along with $N_G^2[V(G')]$.



(c) Step 1: remove $V(G')$ from S_1.



(d) Step 2: add $(S_2 \cap N_G^2[V(G')])$ to $(S_1 \setminus V(G'))$.

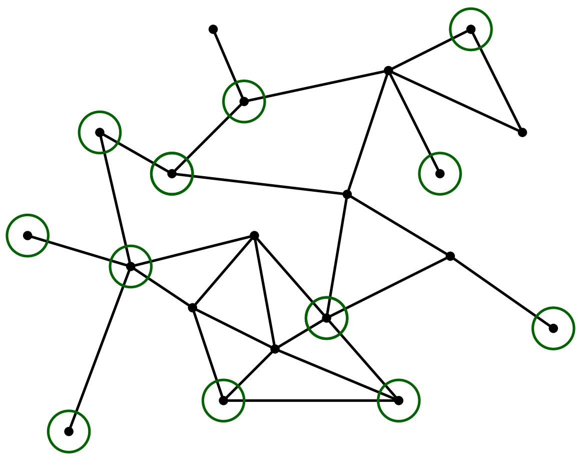

(e) Resulting vertex set S_3.

Figure 3.1: Example of a 2-stitch of S_2 onto S_1 along G' for some graph G.

We are interested in vertex-optimization problems (see Definition 2.2) that *allow for m-stitching*. This means that given some graph G to a vertex-optimization problem Π, the m-stitch operation of any two solutions to Π along any induced subgraph of G yields a feasible solution to Π in G. We define the notion of Π allowing for m-stitching in Definition 3.2.

Definition 3.2 (m-stitching)**.** We say that a vertex-optimization problem Π *allows for m-stitching* if for any graph G, the m-stitch operation of any feasible solution S_2 to Π in G onto any feasible solution S_1 to Π in G along any induced subgraph G' of G yields a feasible solution to Π in G.

We proceed to show the concept of m-stitching by using an example for the MWVC problem (see Definition 2.5).

Given any graph G, any two vertex covers S_1 and S_2 in G, and any induced subgraph G' of G, then we claim that the 1-stitch of S_2 onto S_1 along G' results in a vertex cover S_3 of G. We show this is Figure 3.2.

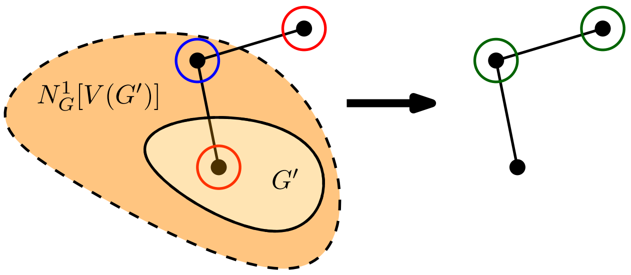


Figure 3.2: Example showing that a graph G together with an induced subgraph G' of G such that the 1-stitch S_3 (green) of S_2 (blue) onto S_1 (red) along G' yields a feasible vertex cover of G.

The main argument comes from the fact that the diameter of a single edge is exactly one. More specifically, there cannot exist an edge between vertices of $V(G')$ and $(V(G) \setminus N^1_G[V(G')])$ and therefore taking the union of both S_1 and S_2 inside of $N^m_G(V(G'))$ implies that all edges remain covered by S_3. In Chapter 6 we prove the claim formally (as a corollary of the H-S-Deletion problem), but for now this showcases how m is identified for the MWVC problem.

We would like to make the observation that, intuitively, the value of m describes how far apart $(S_1 \setminus N^m_G[V(G')])$ and $(S_2 \cap V(G'))$ have to be before these do not influence the feasibility of one another. Therefore, the minimum value for m such that Π allows for m-stitching could be seen as a measure of how local problem Π is. We proceed to show that $m = 1$ is the minimum value such that the MWVC problem allows for m-stitching.

In Figure 3.3 below, we show that if we would change the 1-stitch to a 0-stitch in the example shown in Figure 3.2, the set S_3 would not be a vertex cover of G.

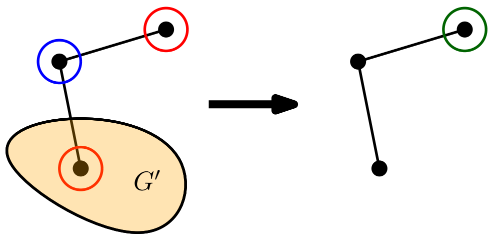


Figure 3.3: Counterexample showing that the MWVC problem does not allow for 0-stitching. It shows that the 0-stitch of S_2 (blue) onto S_1 (red) along G' is an infeasible solution S_3 (green).

Now that we have shown what it means for Π to allow for m-stitching, we proceed to elaborate upon how we will apply it. Recall that we are extending Baker's technique (see Section 2.3) by making the algorithm iterative. Starting from any solution S to Π in G, we repeatedly seek to improve S along a bounded number of induced subgraphs of G. Similar to Baker's technique, each of these induced subgraphs are guaranteed to exhibit a treewidth of $O(m/\epsilon)$. In order to find these improvements, we use the notion of m-stitching. More specifically, we need an algorithm A_Π that can update S without violating feasibility for the remaining graph. In addition, given S and an induced subgraph G' of G such that the m-stitch S_3 of some unknown solution S_2 to Π onto S along G' improves S i.e. $w(S_3) < w(S)$. Then, we want A_Π to return a solution S^* to Π in G with $w(S^*) < w(S)$. In other words, whenever an m-stitch operation onto a given S along a given G' yields a solution S_3 of lower weight than S, then we obtain a solution S^* that improves S. Moreover, in order to extend Baker's technique, we want our algorithm to run in FPT time parameterized by the treewidth of $G[N_G^m[V(G')]]$. The m-stitching property is vital because it effectively guarantees that any local modifications of this form along G' never violate the feasibility of the resulting solution in G.

Furthermore, since we start from some trivial solution to Π in G, we also need to know that for any planar graph G we can compute such a solution efficiently. We define m-LPO problems formally in Definition 3.3.

Definition 3.3 (m-locally planar optimized)**.** Given a vertex-optimization problem Π, then we say that Π is *m-locally planar optimized* (or *m-LPO* in short) if the following properties are met.

(i) For any planar instance with positive weights (G, w), we can find a feasible solution to Π in G in $|V(G)|^{O(1)}$ time.

(ii) For any planar instance (G, w) of Π, we have that problem Π allows for m-stitching.

(iii) There exists an algorithm A_Π that takes as input a planar graph with positive weights (G, w), an induced subgraph G' of G, and a known solution S to Π in G, and in time $f(t) \cdot |V(G)|^{O(1)}$ with t an upper-bound on the treewidth of $G[N_G^m[V(G')]]$ and some computable function f, does the following:

> If there exists a feasible solution S_3 of Π with $w(S_3) < w(S)$ that can be obtained as an m-stitch of an unknown solution S_2 to Π in G onto S along G', then A_Π returns a solution S^* to Π in G with $w(S^*) < w(S)$; otherwise, it returns S.

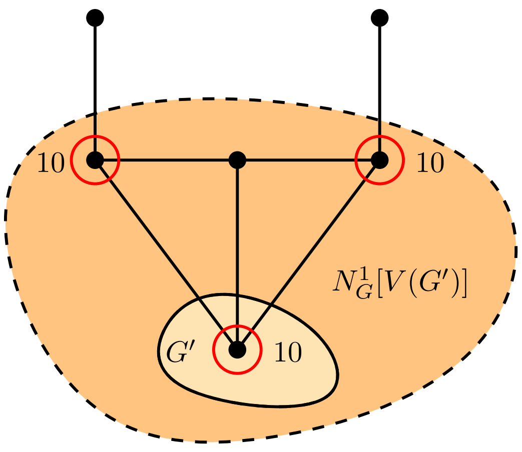
To give some more intuition, in Baker's technique we use an exact algorithm bounded by treewidth to solve the problem independently on each subproblem. Intuitively, we want to relax this independence i.e. we want to find a local solution along G' such that any conditions that are already satisfied by $S \setminus V(G')$ do not have to be satisfied again. Observe that m-stitching suggests that $(S \setminus V(G'))$ already satisfies all conditions of Π in the graph outside of $N_G^m[V(G')]$. Therefore, the task of A_Π typically reduces to satisfying all conditions of Π in $G[N_G^m[V(G')]]$ when $(S \cap N_G^m(V(G')))$ is already part of the solution.

To familiarize the reader with A_Π, we elaborate upon how A_{MWVC} can be designed for the MWVC problem (see Definition 2.5) in the next section.

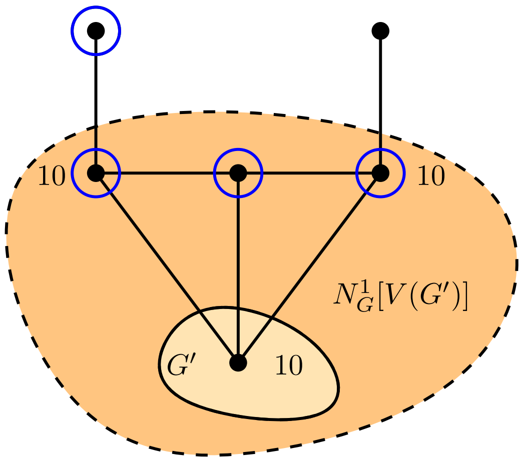
3.2 Designing A_{MWVC} for the MWVC problem

In this section we discuss a way to design algorithm A_{MWVC} for the MWVC problem. Let (G, w) be a planar graph with positive weights, G' be an induced subgraph of G, and S be a known vertex cover of G such that there exists a 1-stitch S_3 of some unknown vertex cover S_2 of G onto S along G' with $w(S_3) < w(S)$. Then by definition, A_{MWVC} invoked on (G, w), G', and S must find a vertex cover S^* of G with $w(S^*) < w(S)$.

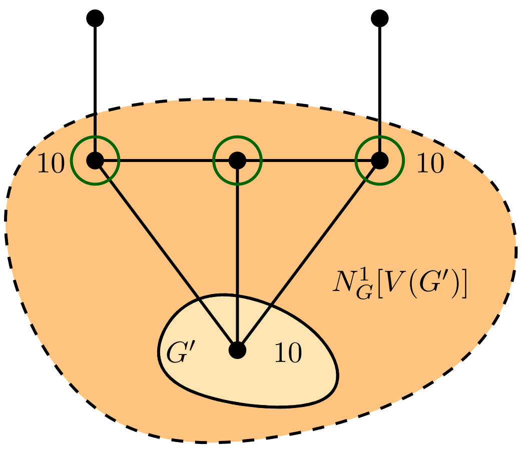
Let us make a first and incorrect attempt by using the same approach as Baker's technique. We solve the MWVC problem exactly on $(G[N^1_G[V(G')]], w)$ to obtain $S_{G'}$. This can be done in FPT time parameterized by the treewidth of $G[N^1_G[V(G')]]$ given by Theorem 2.12. Next, we want to merge S with $S_{G'}$ to obtain S^* of G. Observe that $S_{G'}$ is only a vertex cover of $G[N^1_G[V(G')]]$, but it is easy to show that the m-stitch of trivial solution $S_2 := S_{G'} \cup (V(G) \setminus N^1_G[V(G')])$ onto S along G' yields S^* as well. However, if $S_{G'}$ is computed by an algorithm that is unaware of which vertices are in $S \cap N^1_G(V(G'))$, it may cause $S_{G'}$ to pick vertices to cover edges that are already covered by $S \cap N^1_G(V(G'))$. Observe that $w(S_{G'}) \leq w(S_3 \cap N^1_G[V(G')])$, however, once we add $(S \setminus V(G'))$ back to the solution to obtain S^*, we could have that $w(S) < w(S^*)$. We illustrate this in Figure 3.4.



(a) Known solution S of G.



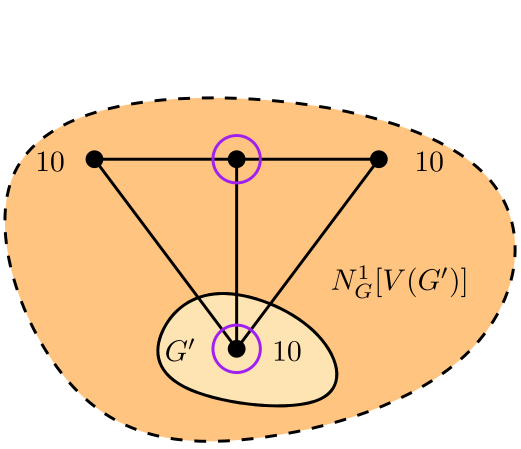
(b) Unknown solution S_2 of G.



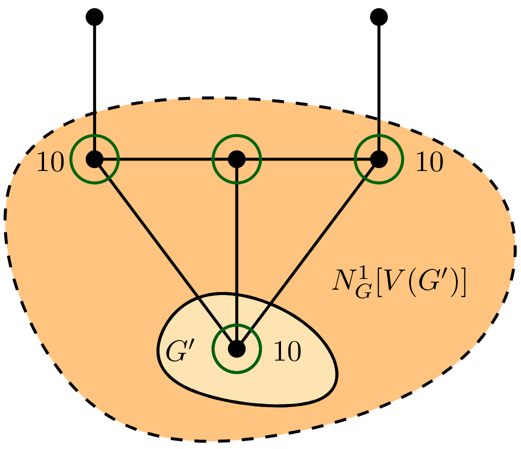
(c) m-stitch S_3 of S_2 onto S along G'.

Figure 3.4: The m-stitch S_3 of unknown solution S_2 of G onto known solution S of G along G' such that $w(S_3) = 21 < w(S) = 30$ for the MWVC problem. All vertex weights are 1 unless indicated otherwise.

Now if we were to solve the instance $(G[N^1_G[V(G')]], w)$ exactly, then we would obtain the solution $S_{G'}$ depicted in Figure 3.5(a). It is clear to see that indeed $w(S_{G'}) = 11 < w(S_3 \cap N^1_G(V(G'))) = 21$. Furthermore, as depicted in Figure 3.5(b) the solution S^* of G adds $S \cap V(G')$ to $S_{G'}$ to ensure feasibility of G; however, this solution has that $w(S^*) = 31 > w(S) = 30$.



(a) Local solution $S_{G'}$.



(b) Solution S^* of G obtained from $S_{G'}$

Figure 3.5: Solving the problem exactly on $(G[N^1_G[V(G')]], w_{G'})$ to obtain $S_{G'}$ such that $w(S) \leq w(S^*)$ while $w(S_3) < w(S)$. All vertex weights are 1 unless indicated otherwise.

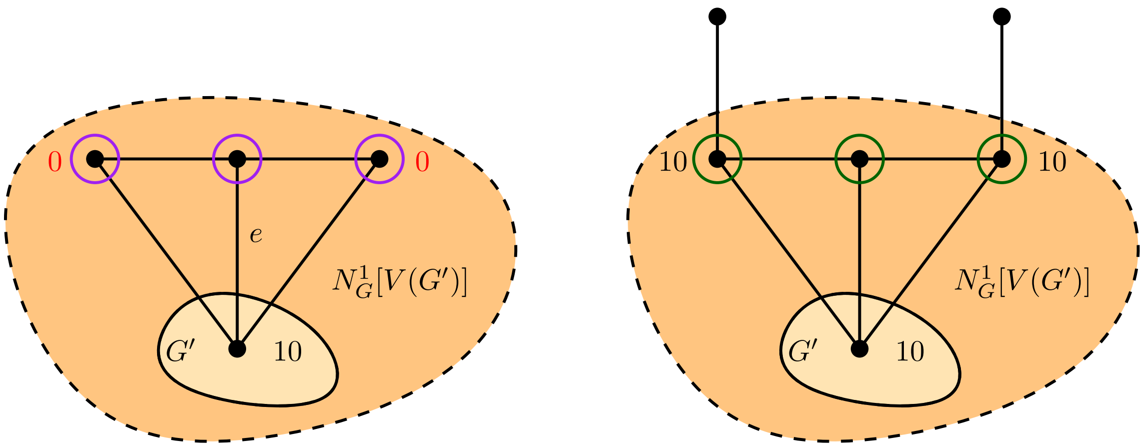
To summarize: the problem with applying Baker's approach of simply solving the MWVC problem on $(G[N^1_G[V(G')]], w)$ is that the algorithm is unaware of which edges in $G[N^1_G[V(G')]]$ are already covered by $S \cap N^1_G(V(G'))$. Even though keeping vertices from $S \cap N^1_G(V(G'))$ in our solution may lead to a suboptimal vertex cover w.r.t. $(G[N^1_G[V(G')]), w)$, the edges connecting $N^1_G[V(G')]$ to the remainder of the graph may make it advantageous to still use these vertices to obtain a globally optimal solution. Therefore, we should incorporate the vertices of $S \cap N^1_G(V(G'))$ into our local subproblem that we solve exactly.

We wish to resolve this issue, meaning that we only want $S_{G'}$ to include vertices in $(S \setminus N^1_G(V(G')))$ if it exclusively covers an edge with one endpoint in $V(G')$ and vice versa. There may be many approaches to do this by, for instance, creating some gadget graph based on $G[N^1_G[V(G')]]$ to force the solution to select vertices of S. However, we approach this by adapting the weight function. More specifically, we construct weights w'_0 that allow the vertices of $S \cap N^1_G(V(G'))$ to be selected *for free*, defined as

$$w'_0(u) := \begin{cases} 0, & \text{if } u \in (S \cap N^1_G(V(G'))) \\ w(u), & \text{otherwise.} \end{cases}$$

We proceed to solve the instance $(G[N^1_G[V(G')]], w'_0)$ exactly and obtain solution $S_{G'}$. Next, we show that this change fixes the challenge for the MWVC problem depicted in Figure 3.5 and provide the pseudocode of A_{MWVC} in Algorithm 2.

Figure 3.6(a) shows the instance $(G[N^1_G[V(G')]], w'_0)$, together with its optimal solution to the MWVC problem. Observe that the only vertex that we selected in $S \setminus N^1_G(V(G'))$ was exclusively used to satisfy edge e depicted in the figure. This edge was not satisfied by any vertices in $S \cap N^1_G(V(G'))$. In the counterexample we selected both endpoints of e, because we did not take into account that all edges on the outer face of $G[N^1_G[V(G')]]$ were already satisfied by $S \cap N^1_G(V(G'))$.



(a) Local solution $S_{G'}$ with w'_0.

(b) Solution S^* of G.

Figure 3.6: Solving the problem exactly on $(G[N_G^1[V(G')]], w'_0)$ to obtain $S_{G'}$ such that $w(S_3) < w(S)$ and $w(S^*) < w(S)$. All vertex weights are 1 unless indicated otherwise.

Algorithm 2 A_{MWVC} for the planar MWVC problem

Input planar graph G, weight function $w : V(G) \to \mathbb{R}_{>0}$, induced subgraph G' of G, and a vertex cover S of G.

Output vertex cover S^* of G with $w(S^*) < w(S)$ if there exists such a vertex cover that can be obtained by a 1-stitch onto S along G'; otherwise, S.

```
1: for $u \in N_G^1[V(G')]$ do
2:     if $u \in (S \cap N_G^1(V(G')))$ then
3:         $w'_0[u] \leftarrow 0$
4:     else
5:         $w'_0[u] \leftarrow w[u]$
6:     end if
7: end for
8: Let $S_{G'}$ be an optimal solution to the MWVC problem for $(G[N_G^1[V(G')]], w'_0)$ by the algo-
   rithm obtained from Theorem 2.12.
9: $S^* \leftarrow (S \setminus V(G')) \cup S_{G'}$
10: if $w(S^*) < w(S)$ then
11:     return $S^*$
12: else
13:     return $S$
14: end if
```

The key correctness property comes from the fact that the algorithm can use all vertices from $S \cap N_G^1(V(G'))$ for free, while all other vertices have a strictly positive weight. This preserves feasibility since all such vertices are already in solution S. This implies that vertices in $N_G^1(V(G')) \setminus S$ would exclusively be chosen to cover edges such that the other endpoint is in $V(G')$. Similarly, we would never select a vertex inside of $V(G')$ that has its other endpoint in $S \cap N_G^1(V(G'))$ for the sole reason of covering this edge, because the other endpoint can be selected with a weight of zero. Therefore, should $w(S_3) < w(S)$, then the solution S^* that is implicitly obtained from an m-stitch of $S_{G'} \cup (S \setminus V(G'))$ onto S along G' must yield a strictly lower weight than S and hence satisfies the requirements of algorithm A_{MWVC}.

Finally, a short note on the running time. It is clear to see that A_{MWVC} is dominated by the routine that solves the MWVC problem exactly on $(G[N_G^1[V(G')]], w'_0)$ (i.e. line 8).

Furthermore, by Theorem 2.12, we know that there exists an FPT time algorithm that is parameterized by the treewidth of the input graph, and hence our algorithm solves the instance in FPT time parameterized by the treewidth of $G[N_G^1[V(G')]]$, as required by property (iii) of a 1-LPO problem.

We have presented A_{MWVC} to elaborate on how such an algorithm could be constructed. Therefore, we have not proven its correctness formally, but only argued why it intuitively works. However, in Chapter 6 we prove that for each connected graph H, the H-S-Deletion problem is m-LPO with $m = diam(H)$. The H-S-Deletion problem contains the MWVC problem when H equals a single edge, meaning that the proof for the MWVC problem follows as a corollary.

We have introduced the notions of m-stitching and m-LPO problems and provided an extensive example for the MWVC problem. In the next chapter we provide our $(1 + \epsilon)$-certified algorithm for a general m-LPO problem.

Chapter 4

Certified algorithm: overview

The purpose of this chapter is to elaborate upon how the framework to design $(1+\epsilon)$-certified algorithms for m-LPO problems works. Our framework is heavily inspired from Baker's technique (see Section 2.3). However, it gives some stronger guarantees, namely in addition to behaving like a PTAS, it also returns a $(1+\epsilon)$-perturbation w' of our weight function that certifies the solution. Moreover, should the weighted graph be $(1+\epsilon)$-stable (see Definition 2.13), then our algorithm returns the optimal solution. Finally, opposed to Baker's technique, our solution is guaranteed to be minimal, otherwise it could not be certified by w'. We provide the pseudocode in Algorithm 3 below.

Algorithm 3 $(1+\epsilon)$-certified algorithm for m-LPO problem Π

Input: connected planar graph G, weight function $w : V(G) \to \mathbb{Z}_{>0}$, parameter $\epsilon > 0$
Output: solution S to Π in G, $(1+\epsilon)$-perturbed weights w' of w that certifies S

```
 1: Let $r$ be an arbitrary vertex of $G$.
 2: $d_{max} \leftarrow \max\{d(v,r) \mid v \in V(G)\}$
 3: for $i \leftarrow 0, 1, \dots d_{max}$ do
 4:     $L_i \leftarrow \{v \in V(G) \mid d(v,r) = i\}$
 5: end for
 6: Compute a feasible solution $S$ to $\Pi$ in $G$ using condition (i) $m$-LPO (see Definition 3.3).
 7: $k \leftarrow \lceil \frac{2m}{\epsilon} \rceil + 2m$
 8: $c \leftarrow 0$
 9: while $c \leq d_{max} - k + 2(m+1)$ do
10:     for $\ell \in k-m, k-m+1, \dots, d_{max}+m+1$ do
11:         $B' \leftarrow \bigcup_{i:=\ell-k+m}^{\ell-m-1} L_i$                    ▷ Block of width $k-2m$.
12:         Let $S'$ be the result of invoking algorithm $A_\Pi$ implied by condition (iii) of $m$-LPO
             on instance $(G,w)$ with solution $S$, and $G' := G[B']$.
13:         if $w(S') < w(S)$ then
14:             $S \leftarrow S'$            ▷ $m$-stitch onto $S$ along $G'$ that improves $S$ found.
15:             $c \leftarrow 0$
16:             break
17:         else
18:             $c \leftarrow c+1$
19:         end if
20:     end for
21: end while
22: $w' \leftarrow w$
23: for $u \in (V(G) \setminus S)$ do
24:     $w'[u] \leftarrow (1+\epsilon)w[u]$
25: end for
26: return $(S, w')$
```

We briefly summarize what our $(1+\epsilon)$-certified algorithm does. The algorithm takes as input any connected planar graph G with positive integer weights w. First, we divide the graph into layers such that adjacent layers may share edges, but non-adjacent layers could not. Next, we compute any trivial solution S to Π in G implied by condition (i) of Π being m-LPO. We proceed to iterate over each possible set of $k-2m$ consecutive layers, where the value of k is dependent on Π and the user-defined parameter $\epsilon > 0$. During each iteration, we invoke algorithm A_Π implied by condition (iii) of Π being m-LPO on G, the subgraph induced by these $k-2m$ layers, and the known solution S to check if an improvement to S can be found as a result of an m-stitch onto S along this induced subgraph. Observe that A_Π takes the closed m-neighborhood over these layers, implying that it uses at most consecutive k layers. Should A_Π find a solution S^* to Π in G with a lower weight than S, then we replace S with S^*; otherwise, we continue to the next $k-2m$ consecutive layers. Once no more improvements can be found, we return the final solution and $(1+\epsilon)$-perturbed weights w' of w obtained from maximally increasing the weights of all vertices in $V(G) \setminus S$, which we claim certifies S.

We organize this chapter into three sections that correspond to the three different parts of this algorithm, respectively: constructing the layer decomposition (see Section 4.1), sliding over the layer decomposition to find improvements (see Section 4.2), and finally how the solution is certified by manipulating the weights (see Section 4.3). Note that we do not prove correctness of Algorithm 3 until the next chapter.

4.1 Layer decomposition

(Lines 1-5 of Algorithm 3)

Our *layer decomposition* is easy to construct and can be computed in polynomial time. First, we take any vertex $r \in V(G)$ that we consider as our starting vertex. Next, we compute the shortest path distance from r to any vertex $u \in V(G)$ (including r) using the function $d_G(r,u)$. Now, every vertex that has the same distance i from r is put into the same *layer* L_i. Let d_{max} denote the eccentricity of r, then we obtain exactly $d_{max}+1$ layers. However, we also add *dummy layers* to the layer decomposition, which we motivate in the next section. We define the layer-composition formally in Definition 4.1.

Definition 4.1 (Layer-decomposition $\mathcal{L}_G^r$)**.** Given a connected graph G, any vertex $r \in V(G)$, then we define a *layer* L_i with $i \in \{0,\dots,d_{max}\}$ as

$$L_i := \{v \in V(G) \mid d_G(v,r) = i\}.$$

Furthermore, we define a *dummy layer* L_i with $i \in \{-m,\dots,-1\}$ and $i \in \{d_{max},\dots,d_{max}+m-1\}$ as

$$L_i := \emptyset.$$

Finally, we define the *layer decomposition* $\mathcal{L}_G^r$ of G as a set of layers corresponding to the union of all layers and dummy layers i.e.

$$\mathcal{L}_G^r := \{L_i \mid i \in \{-m,\dots,d_{max}+m-1\}.$$

We show an example of what $\mathcal{L}_G^r$ looks like on a small planar graph G in Figure 4.1. Observe that Baker's technique implicitly computes the layer decomposition in the process of constructing V_i (see Algorithm 1).

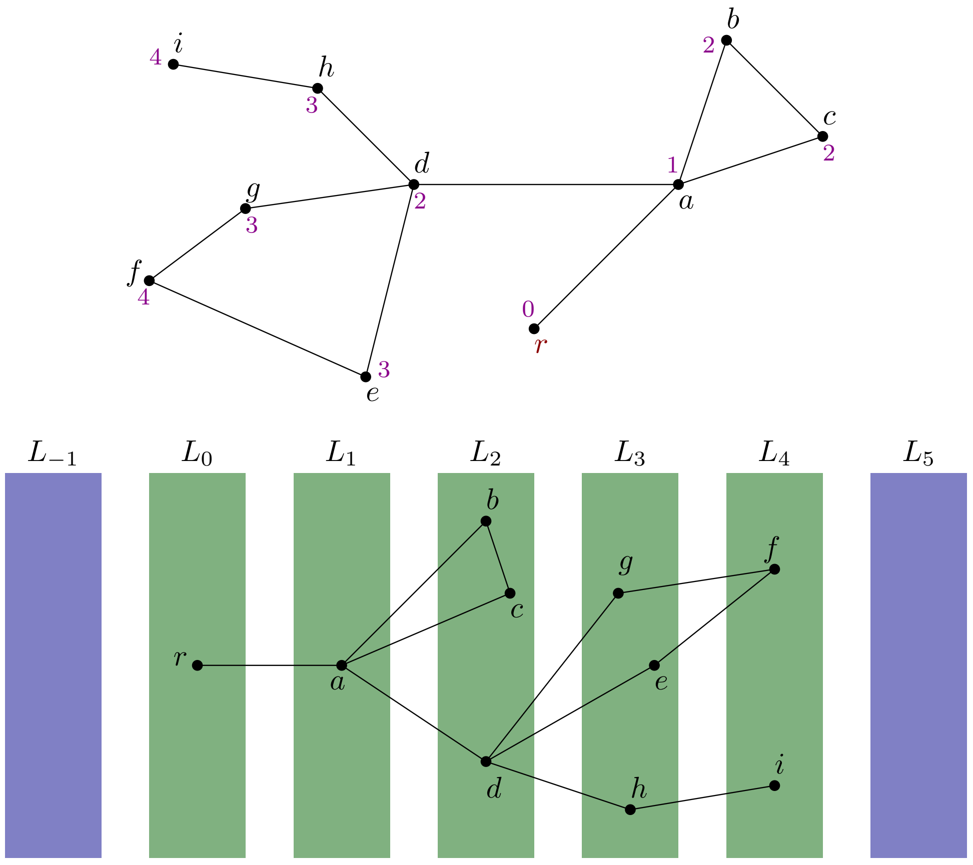


Figure 4.1: Example of a layer decomposition for a small planar graph.

4.2 Sliding over the layer decomposition

(Lines 6-21 of Algorithm 3)

After Algorithm 3 has created the layer decomposition $\mathcal{L}_G^r$ of connected graph G and any $r \in V(G)$, it proceeds to a while-loop that repeats the following process. We iterate over each possible set of $k - 2m$ consecutive layers of $\mathcal{L}_G^r$ in a for-loop until we find an improvement for S using A_Π implied by condition (iii) of Π being m-LPO. Should no improvement exist, then the while-loop terminates. Here, we refer to each set of $k - 2m$ consecutive layers as *blocks* of *width* $k - 2m$ and we formalize these blocks in Definition 4.2.

Definition 4.2 (Block)**.** Given a layer decomposition $\mathcal{L}_G^r$ of a connected graph G and any $r \in V(G)$, then we define a *block* B_a^b as the sequence of layers in $\mathcal{L}_G^r$ starting with a and ending with b, where $b \geq a$. Should some layer not be defined in the $\mathcal{L}_G^r$, then it is handled as an empty set. Moreover, we refer to the *width* of B_a^b as the number of layers it consists of i.e. $b - a + 1$.

Furthermore, we also introduce open and closed *m-layer neighborhood* operators in Definition 4.3. These are similar to the previously introduced open and closed neighborhood operators (see Definition 2.1), however they are less strict i.e. they may include vertices that are in different connected components than the vertices in the block. Therefore, for any block B we have that $N_G^a[B] \subseteq \delta_a[B]$.

Definition 4.3 (m-layer neighborhood)**.** Given a graph G with its corresponding layer decomposition $\mathcal{L}_G^r$ with any $r \in V(G)$, and a block B_a^b in $\mathcal{L}_G^r$ with $b \geq a$. Then we define its *closed m-layer neighborhood* $\delta_m[B_a^b]$ as B_a^b together with its m adjacent layers on both sides i.e.

$$\delta_m[B_a^b] := B_{a-m}^{b+m}.$$

Moreover, we also define the *open m-layer neighborhood* $\delta_m(B_a^b)$ as exclusively the m layers incident to the block on both sides that are adjacent to B_a^b i.e.

$$\delta_m(B_a^b) := \bigcup_{j:=a-m}^{a-1} L_j \cup \bigcup_{j:=b+1}^{b+m} L_j.$$

Finally, whenever the operator encounters a layer that is not defined, it considers these layers as empty sets.

We give an example of a block together with the open and closed m-layer neighborhood in Figure 4.2.

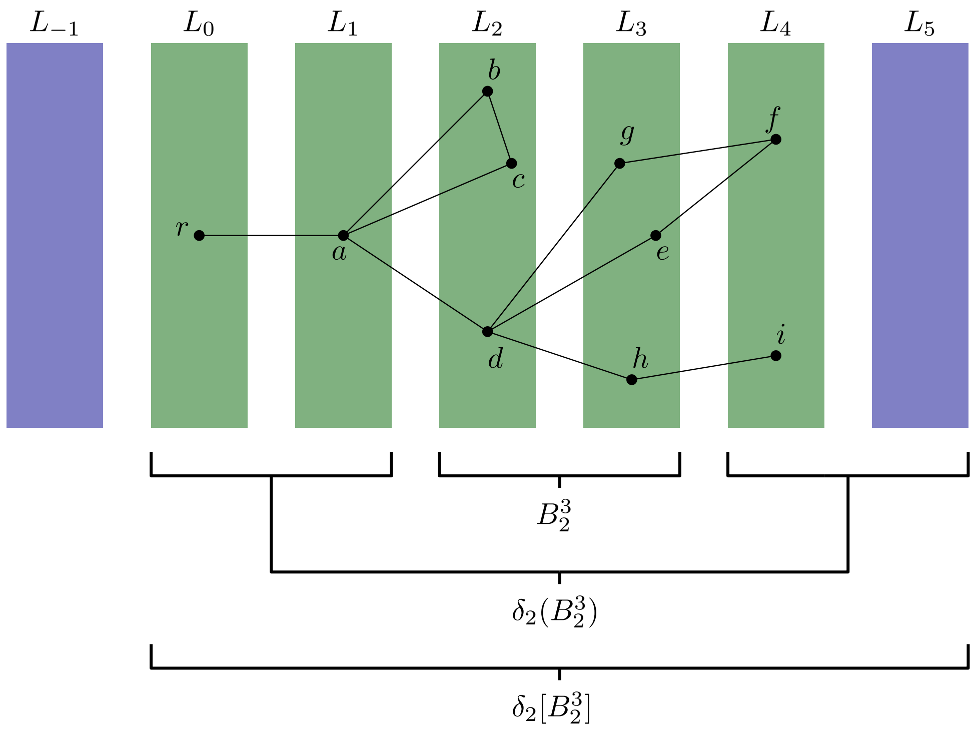
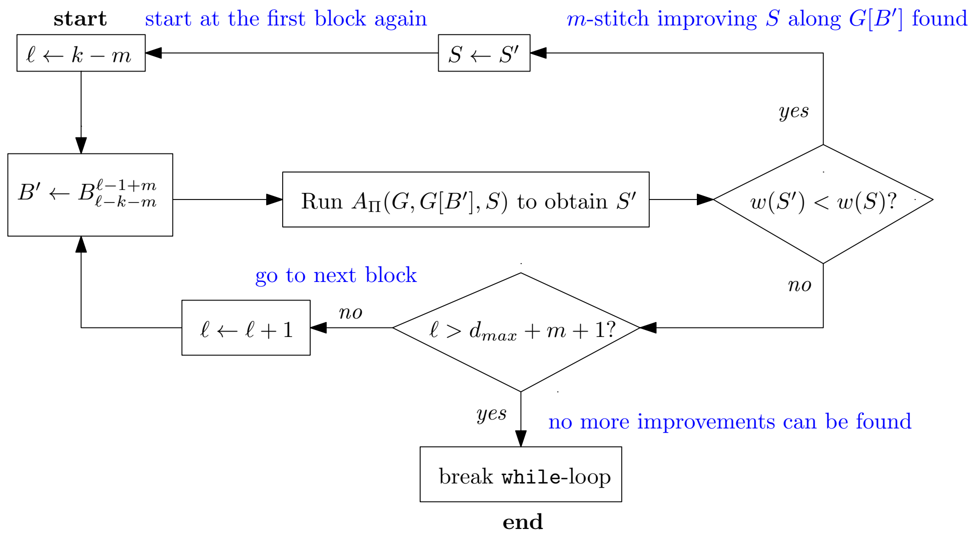


Figure 4.2: Example of a block of width 2, together with its open and closed 2-layer neighborhoods.

Algorithm 3 proceeds to compute a candidate solution S' to Π in G. This is done by invoking A_Π on G, $G' := G[B']$, and the known solution S to obtain S'. Observe that A_Π takes $N_G^m[V(G')]$, which implies that A_Π uses layers from $B \setminus B'$ as well. By definition of m-LPO, should an m-stitch S_3 exist of some unknown solution to Π in G onto S along G' such that $w(S_3) < w(S)$, then we find a solution S^* to Π in G with $w(S^*) < w(S)$. Should this be the case, then we update S by replacing it with S' and reset the while-loop; otherwise, we move on to the neighboring block by moving all indices by 1. Should this not be possible (i.e. B' corresponds to the final block of $\mathcal{L}_G^r$), then the while-loop terminates. We summarize the while-loop below.


start
start at the first block again
m-stitch improving S along $G[B']$ found
$\ell \leftarrow k - m$
$S \leftarrow S'$
yes
$B' \leftarrow B_{\ell-k-m}^{\ell-1+m}$
Run $A_\Pi(G, G[B'], S)$ to obtain S'
$w(S') < w(S)$?
go to next block
no
$\ell \leftarrow \ell + 1$
no
$\ell > d_{max} + m + 1$?
yes
no more improvements can be found
break while-loop
end

Now we can motivate why we add dummy layers to our layer decomposition. Namely, we want for each non-dummy layer L_i that every possible block B' of width $k - 2m$ that contains L_i is constructed. Therefore, we need at least m dummy layers to the left, and symmetrically, m dummy layers to the right of the layers as well. We need this because otherwise there would exist layers (and hence vertices) that are never part of a subgraph G' that is given as input to algorithm A_Π. This might cause us to miss possible improvements as a result of m-stitching onto S along the induced subgraphs of blocks of width $k - 2m$. Furthermore, we would like to make an observation here that connects the diameter of subgraphs G' given to A_Π with the choice of k (and implicitly ϵ).

Observation 4.4. Should an m-stitch S_3 exist of some unknown solution to Π in G onto known solution S along G' with $w(S_3) < w(S)$, then for any induced subgraph $G^* \supseteq G'$ of G, we have that the m-stitch S_4 of S_3 onto S along G^* implies that $S_3 = S_4$. Since $w(S_4) = w(S_3) < w(S)$, an improvement for S would be found if (G, w), G^*, and S are given as input to A_Π. Hence, once Algorithm 3 returns solution S, then there does not exist any induced subgraph G' of G of diameter at most $k - 2m$ such that A_Π would find an improvement for S in (G, w) when (G, w), G' and S are given as input.

4.3 Certifying the solution

(Lines 22-26 of Algorithm 3)

Once the while-loop terminates, we obtain solution S to Π in G that can no longer be improved by algorithm A_Π using blocks of at most width $k - 2m$. To meet the requirements of a $(1+\epsilon)$-certified algorithm, we must return weights w' that is a $(1+\epsilon)$-perturbation of w such that w' certifies S (i.e. S is optimal for the instance (G, w')). We obtain w' by maximally increasing the weights of all vertices in $V(G) \setminus S$ i.e.

$$w'(u) := \begin{cases} (1+\epsilon)w(u), & \text{if } u \in (V(G) \setminus S) \\ w(u), & \text{otherwise} \end{cases}.$$

Since our problem has a minimization objective, it is intuitive that increasing the weights of all vertices that are not in the solution could only improve the quality of the solution for the instance (G, w'). Our algorithm returns solution S together with w' and claims that w' certifies S.

We have presented our $(1+\epsilon)$-certified algorithm for a general m-LPO problem. In the next chapter we formally prove the correctness and running time of Algorithm 3.

Chapter 5

Certified algorithm: analysis

In this chapter we prove that Algorithm 3 is a $(1+\epsilon)$-certified algorithm for an m-LPO problem Π when the input is a parameter $\epsilon > 0$ and the instance (G, w) of Π consists of a connected planar graph and a positive integer weight function. This is the main result of this thesis and stated in Theorem 5.9.

Theorem 5.9 (Meta-Theorem)**.** *Given any vertex-optimization problem* Π *that is* m*-LPO and an instance* (G, w) *to* Π *such that* G *is a connected planar graph and* $w : V(G) \to \mathbb{Z}_{>0}$*, then Algorithm 3 invoked on* (G, w) *and* $\epsilon > 0$ *is a* $(1+\epsilon)$*-certified algorithm that runs in time* $W \cdot f(m/\epsilon) \cdot |V(G)|^{O(1)}$ *for some computable function* f *where* $W := \Sigma_{u \in V(G)} w(u)$*.*

The key challenge to proving Theorem 5.9 is as follows. We assume for the sake of contradiction that S is not certified by w'. Instead, there exists an optimal solution I' for (G, w') with $w'(I') < w'(S)$. We will show that this assumption implies that even if we remove a specific part of the layer decomposition $\mathcal{L}_G^r$ (see Definition 4.1) with any $r \in V(G)$, the weight of $(I' \setminus S)$ is strictly smaller than the weight of $(S \setminus I')$ in the remaining graph. Furthermore, this remaining graph is comprised of blocks (see Definition 4.2) that each have a width of at most $k - 2m$. By a stronger notion of the pigeonhole principle, we claim that at least one of these blocks B^* of width $k - 2m$ allows for an m-stitch onto S along $G[B^*]$ that improves S. By assumption of Π being m-LPO, we know that A_Π must have found an improvement to S if invoked on (G, w), S, and $G[B^*]$. Hence, we reach a contradiction and S is indeed certified by w'.

In Section 5.1 we introduce more logic over layer decompositions. We proceed to use this logic to prove the correctness of Algorithm 3 in Section 5.2.

5.1 Operators over layer decompositions

We aim to remove part of the layer decomposition such that the remaining graph only consists of blocks of width at most $k - 2m$. We do so using the concepts of *k-separators* and its corresponding *k-remainders*. The k-separator C_i^a is similar to how in Baker's technique, we remove every periodic k layers in each possible way. However, instead of a single layer being removed, we remove a layers from the layer decomposition for every periodic k layers. We formalize k-separators in Definition 5.1.

Definition 5.1 (k-separator C_i^a)**.** Given a connected graph G, an integer $a \geq 0$, an integer $k \geq a$, and a layer decomposition $\mathcal{L}_G^r$ with any $r \in V(G)$, then we define the *k-separator* C_i^a as a set of layers

$$C_i^a := \{L_j \in \mathcal{L}_G^r \mid j \pmod{k} \in \{i, \ldots, i + a - 1\}\}.$$

Moreover, we wish to construct every distinct k-separator for a fixed layer decomposition $\mathcal{L}_G^r$. Observe that the modular function implies that a k-separator behaves periodically with a period of k. Therefore, there are k distinct offsets at which a k-separator can start the cycle. More specifically, we create a k-separator C_i^a for fixed a and for all $i \in \{0, \ldots, k-1\}$, which generates all distinct k-separators. To illustrate this, we give an example of all possible k-separators that exist for some layer decomposition of a connected graph in Figure 5.2.

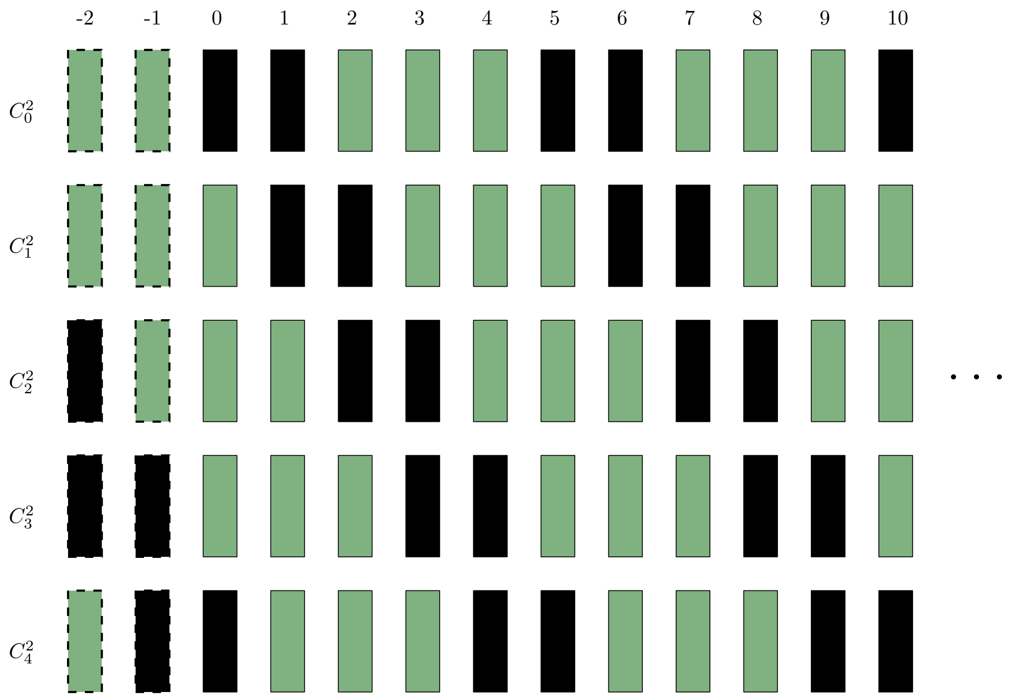


Figure 5.1: Example of all k-separators for a layer decomposition (dummy layers are dashed), where $a = 2$ and $k = 5$. Each row i of layers corresponds to a distinct k-separator C_i^2 consisting of the black-colored layers. The green layers are those in its corresponding remainder $R(C_i^2)$.

Next, every k-separator C_i^a has a corresponding k-remainder $R(C_i^a)$. This remainder consists of all layers that are **not** in C_i^a. Moreover, these remaining layers are stored as blocks and are constructed such that each block is maximal (i.e. no remaining layers exist that are incident to any block in $R(C_i^a)$). We formalize k-remainders in Definition 5.2.

Definition 5.2 (k-remainder $R(C_i^a)$)**.** Given a graph G with a layer decomposition $\mathcal{L}_G^r$ with any $r \in V(G)$ and any k-separator C_i^a of $\mathcal{L}_G^r$. Then we define the *k-remainder* $R(C_i^a)$ as a set of blocks, each corresponding to a maximal subsequence of consecutive layers of $\mathcal{L}_G^r \setminus C_i^a$. The blocks in the remainder are ordered such that the indices of the corresponding layers strictly increase.

Observe that if a is even (suffices for all our purposes), then each block (except for the possible start and end blocks) in $B_j \in R(C_i^a)$ has a width of $k - a$. Moreover, the possible start and end blocks have a width of at most $k - a$. We show which sequence of blocks exists for some remainder in Figure 5.2.

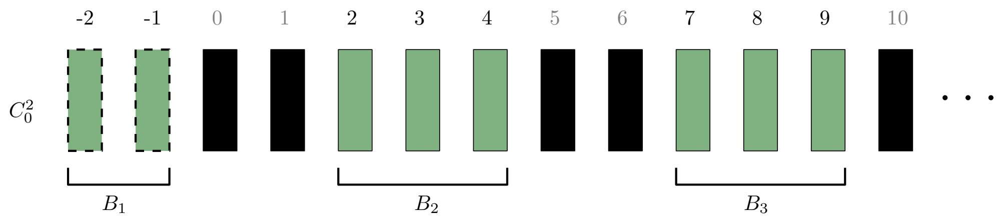


Figure 5.2: Example of the k-remainder $R(C_0^2) = \{B_1, B_2, B_3, \ldots\}$ of a k-separator C_0^2.

Next, we would like to make an observation about how a k-separator can be partitioned if a is an even integer. We use this observation in a later proof.

Observation 5.3. Given a connected graph G and the layer decomposition $\mathcal{L}_G^r$ for any $r \in V(G)$, then for any k-separator C_i^{2a} with an integer $a \geq 0$, we have that the set

$$\{\delta_a(B_j) \mid B_j \in R(C_i^{2a})\}$$

forms a partition of C_i^{2a} for any $i \in \{0, \dots, k-1\}$ such that each element corresponds to the a layers to both sides of a block $B_j \in R(C_i^{2a})$.

We give an example of Observation 5.3 in Figure 5.3 when we set $a = 1$.

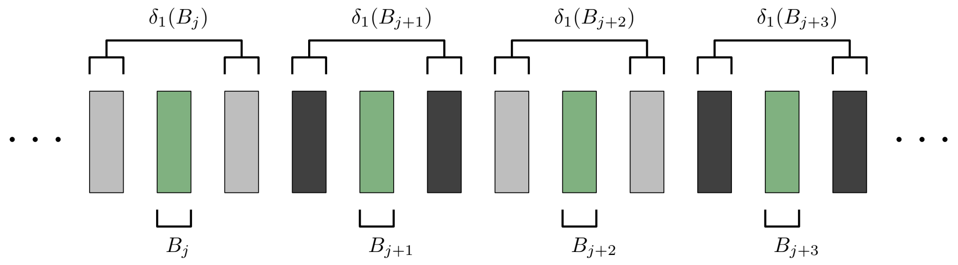


Figure 5.3: Example of how the set $\{\delta_1(B_j) \mid B_j \in R(C_i^2)\}$ forms a partition of a 3-separator C_i^2 as stated in Observation 5.3 for an arbitrary $i \in \{0, 1, 2\}$.

Finally, we would like to make an important statement about the treewidth (see Definition 2.10) of the blocks. We use this in the running time analysis of Algorithm 3.

Lemma 5.4. Given a connected planar graph G, together with its layer decomposition $\mathcal{L}_G^r$ for any $r \in V(G)$. Then for any block B of width $k \geq 0$ in $\mathcal{L}_G^r$, we have that the induced subgraph $G[B]$ has a treewidth of at most $3k + 2$.

Proof. In order to prove this claim, we must obtain a graph G_j with for some j in $\{0, \dots, O(k)\}$ stated in Lemma 2.17 such that B is fully contained in G_j. Observe that we can ignore dummy layers of $\mathcal{L}_G^r$, since these are empty sets and its induced subgraphs have a treewidth of zero. Let B begin at layer L_x and end at L_{x+k-1} of $\mathcal{L}_G^r$ for some integer $0 \leq x \leq d_{max} - k + 1$. Then, we need to construct a V_j such that layers L_{x-1} and L_{x+k} are in V_j, while $B \cap V_j = \emptyset$. This is achieved for the set $V_j = \{v \in V(G) \mid d(r, v) \pmod{k+1} \equiv i\}$ by exactly one $j \in \{0, \dots, k\}$. This is because $x - 1$ and $x + k$ are exactly $k + 1$ layers apart. Furthermore, by Lemma 2.17, we know that all connected components of $G_j := G[V(G) \setminus V_j]$ have a bounded treewidth of $3k+2$. Since $G[B]$ is fully contained in G_j, we know that all connected components of $G[B]$ have a treewidth of at most $3k+2$ as well. Finally, computing the treewidth over a disconnected graph reduces to taking the maximum treewidth over all connected components, hence this bound is preserved for $G[B]$. Our claim follows. $\square$

5.2 Strengthening the pigeonhole principle

In this section we argue how the weights of solutions are distributed over the layer decomposition $\mathcal{L}_G^r$ with any $r \in V(G)$. In Baker's technique we make the claim that at least one of the ways to remove every k periodic layer deletes at most $1/k$ times the total weight of any vertex set $X \subseteq V(G)$ (see Lemma 2.16). This is a direct application of the pigeonhole principle. Now we make a generalized argument using the k-separators. Since we remove a layers rather than one, this naturally extends to removing at most a/k times the total weight of set X. Observe that if we set $a = 1$, then the k-separator exhibits the same behavior the vertex set V_i constructed in Baker's Technique. We formalize the generalized claim in Lemma 5.5.

Lemma 5.5. Given a layer decomposition $\mathcal{L}_G^r$ of a connected weighted graph (G, w) and any $r \in V(G)$, and a set $X \subseteq V(G)$, then for each $a \geq 1$ there exists a k-separator C_i^a with $i \in \{0, \ldots, k-1\}$ such that

$$w(X \cap C_i^a) \leq \frac{a}{k} w(X).$$

Proof. Let us assume for the sake of contradiction that, for all C_j^a with $j \in \{0, \ldots, k-1\}$ it holds that $w(X \cap C_j^a) > \frac{a}{k} w(X)$. Now this implies that if we sum over each C_j^a we obtain:

$$\begin{aligned}
\sum_{j=0}^{k-1} w(X \cap C_j^a) &> \sum_{j=0}^{k-1} \frac{a}{k} w(X) && \text{by assumption} \\
&= \frac{ak}{k} w(X) && \text{by independence of terms in summation} \\
&= aw(X).
\end{aligned}$$

However, observe that any layer of $\mathcal{L}_G^r$ is in a different k-separators. This shows that the sum over all k k-separators is a times the weight of X i.e.

$$\sum_{j=0}^{k-1} w(X \cap C_j^a) = aw(X).$$

We have reached a contradiction and therefore there must exist an $i \in \{0, \ldots, k-1\}$ such that the k-separator C_i^a has that $w(X \cap C_i^a) \leq \frac{a}{k} w(X)$. $\square$

In Baker's technique we are essentially done at this point, because we directly use this result to obtain a PTAS (see Definition 2.4). However, we want to prove that Algorithm 3 is stronger than a PTAS, namely, we want to show that it is a $(1 + \epsilon)$-certified algorithm for connected planar graphs with integer weights. Our next step is to assume for the sake of contradiction that w' constructed by the algorithm does not certify the returned solution S. This implies that there exists an optimal solution I' for (G, w') such that $w'(I') < w'(S)$. We show that if we apply Lemma 5.5 on the vertex set $(S \setminus I')$ and set $a = 2m$ to obtain C_i^{2m}, then even if we only consider $(S \setminus I') \cap R(C_i^{2m})$, its weight is still strictly larger than $(I' \setminus S)$ in the original instance (G, w). Intuitively this means that I' would be a significantly better solution than S in (G, w). We prove this claim formally in Lemma 5.7. However, in order to do so we need Lemma 5.6, where we prove a simple lower-bound on ϵ with respect to how we define k in line 0 in Algorithm 3.

Lemma 5.6. Suppose we have a positive real $\epsilon > 0$, a non-negative integer $m \geq 0$, and an integer k that is defined as $k = \lceil \frac{2m}{\epsilon} \rceil + 2m$, then

$$\epsilon \geq \frac{2m}{k - 2m}.$$

Proof. We apply case distinction on whether $\frac{2m}{\epsilon} \in \mathbb{Z}$ or $\frac{2m}{\epsilon} \notin \mathbb{Z}$. We start with the former case:

$$\begin{aligned}
k &= \lceil \frac{2m}{\epsilon} \rceil + 2m \\
k &= \frac{2m}{\epsilon} + 2m && \text{by assumption that } \tfrac{2m}{\epsilon} \in \mathbb{Z}, \text{ we have that } \lceil \tfrac{2m}{\epsilon} \rceil = \tfrac{2m}{\epsilon} \\
k - 2m &= \frac{2m}{\epsilon} && \text{subtract } 2m \text{ from both sides} \\
\epsilon &= \frac{2m}{k - 2m} && \text{multiply both sides by } \tfrac{\epsilon}{k-2m}.
\end{aligned}$$

We proceed to the latter case:

$$\begin{aligned}
k &= \lceil \frac{2m}{\epsilon} \rceil + 2m & \\
k &\geq \frac{2m}{\epsilon} + 2m & \text{by definition of the ceiling operator} \\
k - 2m &\geq \frac{2m}{\epsilon} & \text{subtract } 2m \text{ from both sides} \\
\epsilon &\geq \frac{2m}{k - 2m} & \text{multiply both sides by } \tfrac{\epsilon}{k-2m}.
\end{aligned}$$

We conclude that in both cases $\epsilon \geq \frac{2m}{k-2m}$ holds. □

Lemma 5.7. Let Π be an m-LPO problem, (G, w) be an instance of Π such that G is a connected planar graph and $w : V(G) \to \mathbb{R}_{>0}$, $\epsilon > 0$ a positive real number, k an integer defined as $k = \lceil \frac{2m}{\epsilon} \rceil + 2m$, $\mathcal{L}^r_G$ the layer decomposition of G for any $r \in V(G)$, S a solution to Π in G, and I' an optimal solution of Π in (G, w') with

$$w'(u) := \begin{cases} (1+\epsilon)w(u), & \text{if } u \in (V(G) \setminus S) \\ w(u), & \text{otherwise} \end{cases}, \qquad \forall u \in V(G).$$

If we in addition assume that $w'(I') < w'(S)$, then there exists an $i \in \{0, \ldots, k-1\}$ such that the $R(C^{2m}_i)$ of the k-separator C^{2m}_i over $\mathcal{L}^r_G$ satisfies

$$w((S \setminus I') \cap R(C^{2m}_i)) > w(I' \setminus S).$$

Proof. Assuming that $w'(I') < w'(S)$ allows us to observe the following:

$$w'(S \setminus I') + w'(I' \cap S) = w'(S) > w'(I') = w'(I' \setminus S) + w'(I' \cap S).$$

Subtracting $w'(I' \cap S)$ from both sides of the strict inequality gives us that $w'(S \setminus I') > w'(I' \setminus S)$. In addition, since all vertices of S have the same weight for both w and w' and all other vertices $u \in (V(G) \setminus S)$ have an additional $\epsilon w(u)$ weight in w', we can derive that

$$w(S \setminus I') = w'(S \setminus I') > w'(I' \setminus S) = (1+\epsilon)w(I' \setminus S). \tag{5.1}$$

Now we show that there must exist a k-separator C^{2m}_i such that $w((S \setminus I') \cap R(C^{2m}_i)) > w(I' \setminus S)$. First, we invoke Lemma 5.5 on (G, w) and $\mathcal{L}^r_G$ with $X = (S \setminus I')$ to obtain k-separator C^{2m}_i satisfying $w((S \setminus I') \cap C^{2m}_i) \leq \frac{2m}{k} w(S \setminus I')$. Next, we show that this k-separator exhibits the desired property:

$$w((S \setminus I') \cap R(C^{2m}_i)) = w(S \setminus I') - w((S \setminus I') \cap C^{2m}_i)$$

by definition of a k-remainder

$$\geq w(S \setminus I') - \frac{2m}{k} w(S \setminus I')$$

since our choice of C^{2m}_i implies that $w((S \setminus I') \cap C^{2m}_i) \leq \frac{2m}{k} w(S \setminus I')$

$$= \frac{k - 2m}{k} w(S \setminus I')$$

by multiplying the first term with k/k and combining the fractions

$$> \frac{k - 2m}{k} (1+\epsilon) w(I' \setminus S)$$

by strict inequality (5.1)

$$\geq \frac{k-2m}{k}(1+\frac{2m}{k-2m})w(I' \setminus S)$$

because $\epsilon \geq \lceil \frac{2m}{k-2m} \rceil$ as stated in Lemma 5.6

$$= \frac{k-2m}{k} \cdot \frac{k}{k-2m} w(I' \setminus S)$$

since $1 + \frac{2m}{k-2m} = \frac{k-2m}{k-2m} + \frac{2m}{k-2m} = \frac{k}{k-2m}$

$$= w(I' \setminus S).$$

We conclude that our claim is true. □

In Lemma 5.7 we proved that there exists an $i \in \{0, \ldots, k-1\}$ such that the remainder of the k-separator C_i^{2m} satisfies that

$$w((S \setminus I') \cap R(C_i^{2m})) > w(I' \setminus S).$$

Now the pigeonhole principle is embedded in C_i^{2m} and, in addition, we have applied some algebra over the vertex-weights to obtain this strict inequality. We want to take it one step further by effectively applying the pigeonhole principle once more. To be more precise, we have a strict inequality relation over the symmetric difference of I' and S that is divided over a finite set of blocks. Moreover, we observe that taking the disjoint union of $\delta_m[B_j]$ (see Definition 4.3) for all $B_j \in R(C_i^{2m})$ yields $V(G)$. Therefore, there must exist at least one block $B^* \in R(C_i^{2m})$ such that the strict inequality holds locally in the layer decomposition i.e.

$$w((S \setminus I') \cap B^*) > w((I' \setminus S) \cap \delta_m[B^*]).$$

We formalize the existence of block B^* in Lemma 5.8.

Lemma 5.8. Given a layer decomposition $\mathcal{L}_G^r$ with any $r \in V(G)$ of any connected weighted graph (G, w), together with a k-separator C_i^{2m} and two vertex sets $I', S \subseteq V(G)$ such that

$$w((S \setminus I') \cap R(C_i^{2m})) > w(I' \setminus S),$$

then there exists a block $B^* \in R(C_i^{2m})$ that satisfies that

$$w((S \setminus I') \cap B^*) > w((I' \setminus S) \cap \delta_m[B^*]).$$

Proof. Let us assume for the sake of contradiction that no such block exists. This implies that the following must hold:

$$w((I' \setminus S) \cap \delta_m[B_j]) \geq w((S \setminus I') \cap B_j) \qquad \forall B_j \in R(C_i^{2m}). \tag{5.2}$$

We aim to derive a contradiction by summing over the left-hand side and the right-hand side of inequality (5.2). We begin with the right-hand side:

$$\sum_{B_j \in R(C_i^{2m})} w((S \setminus I') \cap B_j) = w((S \setminus I') \cap R(C_i^{2m})) \tag{5.3}$$

This holds because we sum over the remainder of C_i^{2m} exactly. Next, we sum over the left-hand side term of inequality (5.2).

$$\sum_{B_j \in R(C_i^{2m})} w((I' \setminus S) \cap \delta_m[B_j]) = \sum_{B_j \in R(C_i^{2m})} w((I' \setminus S) \cap B_j) + w((I' \setminus S) \cap \delta_m(B_j))$$

we can split the two terms in the summation since $\delta_m[B_j] = B_j + \delta_m(B_j)$

$$= w((I' \setminus S) \cap R(C_i^{2m})) + \sum_{B_j \in R(C_i^{2m})} w((I' \setminus S) \cap \delta_m(B_j))$$

summing over all blocks in the remainder yields $R(C_i^{2m})$

$$= w((I' \setminus S) \cap R(C_i^{2m})) + w((I' \setminus S) \cap C_i^{2m})$$

taking the m-open layer neighborhood of all blocks $B_j \in R(C_i^{2m})$ forms a partition of C_i^{2m} as shown in Observation 5.3, implying that the sum over the weight of $(I' \setminus S)$ in the open m-layer neighborhood of all blocks $B_j \in R(C_i^{2m})$ is equal to the weight of $(I' \setminus S)$ in C_i^{2m}

$$= w(I' \setminus S). \tag{5.4}$$

We proceed to derive an upper-bound on $w((S \setminus I') \cap R(C_i^{2m})$ to derive a contradiction with the preconditions of C_i^{2m} i.e.

$$w((S \setminus I') \cap R(C_i^{2m})) = \sum_{B_j \in R(C_i^{2m})} w((S \setminus I') \cap B_j)$$

by equation (5.3)

$$\leq \sum_{B_j \in R(C_i^{2m})} w((I' \setminus S) \cap \delta_m[B_j])$$

since inequality (5.2) holds for all B_j

$$= w(I' \setminus S)$$

by equation (5.4).

However, this is a contradiction with our choice of C_i^{2m}, and hence there must exist a block $B^* \in R(C_i^{2m})$ such that $w((I' \setminus S) \cap \delta_m[B^*]) < w((S \setminus I') \cap B^*)$. □

We have gathered all the necessary ingredients to prove Theorem 5.9. More precisely, now we can show that the m-stitch S_3 of I' onto S along $G' := G[B^*]$ satisfies

$$w(S_3) < w(S).$$

We show that since Π is m-LPO, we know that the algorithm A_Π with input G, G', and S finds a solution S^* to Π in G such that

$$w(S^*) < w(S).$$

Hence, we know that the algorithm could not have returned S yet, implying that we have reached a contradiction. We show the position of this m-stitch in Figure 5.4.

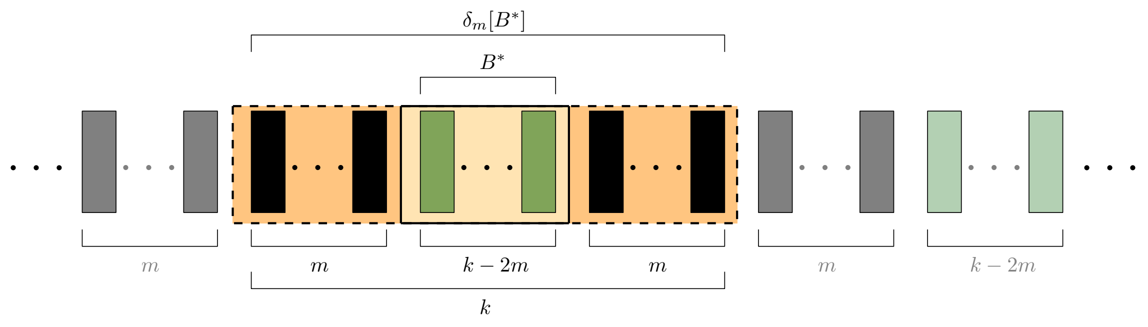


Figure 5.4: Position of the m-stitch S_3 of I' onto S along the subgraph induced by block $B^* \in R(C_i^{2m})$ with C_i^{2m} satisfying the conditions of Lemma 5.8.

Theorem 5.9 (Meta-Theorem)**.** Given any vertex-optimization problem Π that is m-LPO and an instance (G, w) to Π such that G is a connected planar graph and $w : V(G) \to \mathbb{Z}_{>0}$, then Algorithm 3 invoked on (G, w) and $\epsilon > 0$ is a $(1 + \epsilon)$-certified algorithm that runs in time $W \cdot f(m/\epsilon) \cdot |V(G)|^{O(1)}$ for some computable function f where $W := \Sigma_{u\in V(G)} w(u)$.

Proof. We begin by proving that Algorithm 3 is a $(1 + \epsilon)$-certified algorithm for Π. This implies that for any input, the algorithm converges, must return a feasible solution S to Π and weights w' that is a $(1 + \epsilon)$-perturbation of w such that S is optimal for (G, w').

First, the algorithm always converges, which becomes clear in the running time analysis. Second, we argue that the returned weights w' are a $(1 + \epsilon)$-perturbation of w. It is easy to see that by construction of w' (in lines 22 to 25), all unchanged weights are trivially satisfied, and the weights for vertices $u \in (V(G) \setminus S)$ are set to $(1 + \epsilon)w(u)$, which is allowed by definition of a $(1 + \epsilon)$-perturbation. Next, we argue that S is a feasible solution to Π in G. By definition of Π being m-LPO, we know that the initial solution computed in line 6 must be a feasible solution. Moreover, whenever S is replaced in Line 14, it is replaced the solution S' returned by A_Π invoked on (G, w), S and the graph $G[B']$ induced by the current block B'. Since Π is m-LPO, we know that any such S' must be a feasible solution to Π in G.

Finally, we argue that S is an optimal solution for (G, w'). Let us assume for the sake of contradiction that S is not optimal for (G, w'). This implies that there must exist an optimal solution I' of Π in (G, w') such that $w'(I') < w'(S)$. We can invoke Lemma 5.7 on Π, (G, w), w', I', and S. This gives us a k-separator C_i^{2m} with $i \in \{0, \dots, k - 1\}$ such that

$$w((S \setminus I') \cap R(C_i^{2m})) > w(I' \setminus S).$$

Next, we invoke Lemma 5.8 on C_i^{2m}, S, and I' to obtain a block $B^* \in R(C_i^{2m})$ of width at most $k - 2m$ such that

$$w((S \setminus I') \cap B^*) > w((I' \setminus S) \cap \delta_m[B^*]).$$

Let $G' := G[B^*]$, then we define the m-stitch S_3 of I' onto S along G'. Since Π is m-LPO, we know that S_3 must be a feasible solution to Π in G. Moreover, it is clear that

$$w(S_3) < w(S);$$

hence, running A_Π on G, G', and S would give us a solution S^* with

$$w(S^*) < w(S).$$

Since the for-loop that is nested in the while-loop iterates over all possible blocks of width $k - 2m$, one of these must correspond to B^*. Furthermore it would have called algorithm A_Π on (G, w), S, and G' and replaced S' with S^*. Therefore, Algorithm 3 would have found an improvement to S and have proceeded to reset the while-loop. We have reached a contradiction and therefore Algorithm 3 correctly returns a weight function w' that certified the returned solution S.

We continue to prove the running time of Algorithm 3. It is clear to see that every line that does not call A_Π can be computed in $|V(G)|^{O(1)}$ time. During every iteration of the while-loop, we run a for-loop consisting of at most $d_{max} + 2m - k = O(|V(G)|)$ iterations. During each iteration the running time is clearly dominated by line 12. By assumption that Π is m-LPO, we can invoke A_Π on (G, w), S, and $G[B']$ for some computable function f' in time

$$f'(tw(G[N_G^m[B']]) \cdot |V(G)|^{O(1)} \leq f'(tw(G[\delta_m[B']]) \cdot |V(G)|^{O(1)}$$

because $N_G^m[B'] \subseteq \delta_m[B']$ and deleting vertices and edges can only decrease the treewidth

$$\leq f'(3k + 2) \cdot |V(G)|^{O(1)}$$

since $\delta_m[B']$ is a block of width k, and by Lemma 5.4, we know that its treewidth is at most $3k+2$.

It follows that the running time of each iteration of the while-loop becomes

$$O(|V(G)|) \cdot f'(3k+2) \cdot |V(G)|^{O(1)} = f'(3k+2) \cdot |V(G)|^{O(1)}.$$

During every iteration of the while-loop, we either conclude that no improvements for a block of width $k - 2m$ can be found, or we have found an improvement to the solution. Should no next block exist, then our while-loop terminates. Since w is assumed to consist of positive integer weights, we must have that any improvement decreases the weight of S by at least one. Therefore, we can find at most $\Sigma_{u \in V} w(u) = W$ improvements for S. Finally, $k = O(m/\epsilon)$ and thus our running time becomes

$$W \cdot f(m/\epsilon) \cdot |V(G)|^{O(1)}$$

for some computable function f. Our claim follows. □

We have proven that Algorithm 3 is a $(1+\epsilon)$-certified algorithm that runs in FPT time parameterized by m/ϵ for an m-LPO problem Π when the input consists of a parameter $\epsilon > 0$, a connected planar graph G our weights are positive integers. In the next chapter we aim to populate m-LPO problems beyond the MWVC problem (see Section 3.2). More specifically, we prove for both the H-S-Deletion problem and the MWDOM problem that there exists an $m > 0$ such that the problem is m-LPO. We also show that the MWIS problem allows for a $(1+\epsilon)$-certified algorithm that runs in FPT time parameterized by $(1/\epsilon)$ when the same assumptions on the input apply.

Chapter 6

Populating locally planar optimized problems

In this chapter we seek to populate m-LPO problems by using (variations of) the local improvement approach that was introduced in Section 3.2 for the MWVC problem. In Section 6.1 we prove that the H-S-Deletion problem is σ_H-LPO with $\sigma_H := diam(H)$. Next, in Section 6.2, we show that the MWDOM problem is 2-LPO by slightly modifying the approach. Finally, in Section 6.3, we show that the MWIS problem admits a $(1 + \epsilon)$-certified algorithm by using the $(1 + \epsilon)$-certified algorithm for the MWVC problem.

6.1 H-S-Deletion problem is locally planar optimized

We show that the H-S-Deletion problem (see Definition 2.8) is σ_H-LPO. In order to do so we must show that all three properties of σ_H-LPO hold (see Definition 3.3). We formally prove the first property in Lemma 6.1 i.e. some solution can be found efficiently in any planar graph G.

Lemma 6.1. For any connected graph H, the H-S-Deletion problem on any planar graph G has $V(G)$ as a feasible solution.

Proof. For any subgraph G_H of G that is isomorphic to H, we have that $V(G) \cap V(G_H) = V(G_H)$, which implies that $V(G)$ includes at least one vertex of $V(G_H)$. Therefore, $V(G)$ is a feasible H-S-Deletion set of G. □

We proceed to show that property (ii) of σ_H-LPO problems holds as well i.e. the H-S-Deletion problem allows for σ_H-stitching (see Definition 3.2).

Lemma 6.2. The H-S-Deletion problem allows for σ_H-stitching for any connected graph H, where $\sigma_H := diam(H)$.

Proof. Let (G, w) be any instance to the H-S-Deletion problem, S_1 and S_2 be any two H-S-Deletion sets of G, and G' be any induced subgraph of G. Then it suffices to prove that the set

$$S_3 := (S_1 \setminus V(G')) \cup (S_2 \cap N_G^m[V(G')])$$

is an H-S-Deletion of G. By definition of the H-S-Deletion problem, we must have that for every subgraph G_H of G that is isomorphic to H, there exists at least one vertex in $S_3 \cap V(G_H)$. We apply case distinction on the position of G_H in the graph G:

- $(V(G_H) \cap V(G')) = \emptyset$: since $(S_1 \setminus V(G')) \subseteq S_3$ we must have that all subgraphs G_H that live in $(V(G) \setminus V(G'))$ are hit by S_3;
- $(V(G_H) \cap V(G')) \neq \emptyset$: since $(S_2 \cap N_G^{\sigma_H}[V(G')]) \subseteq S_3$ is an H-S-Deletion of $N_G^{\sigma_H}[V(G')]$ and, by definition, a connected subgraph of G of diameter σ_H that contains a vertex of

$V(G')$ cannot contain a vertex outside of $N_G^{\sigma_H}[V(G')]$. Therefore, it must be the case that G_H is hit by S_3.

Together these two cases are exhaustive and, therefore, S_3 is an H-S-Deletion of G. We conclude that the H-S-Deletion problem allows for σ_H-stitching. □

So far we have shown that both property (i) and property (ii) of σ_H-LPO problems hold for the H-S-Deletion problem. We continue to show that property (iii) holds as well. This requires us to provide an algorithm A_{HS} that meets the requirements of (iii) (HS is short for the H-S-Deletion problem). This algorithm is a direct generalization of A_{MWVC} of the MWVC problem provided in Section 3.2. Furthermore, the pseudocode of algorithm A_{HS} can be found in Algorithm 4.

Algorithm 4 A_{HS} for the planar H-S-Deletion problem

Input planar graph G, weight function $w : V(G) \to \mathbb{R}_{>0}$, induced subgraph G' of G, and an H-S-Deletion S of G.

Output H-S-Deletion S^* of G with $w(S^*) < w(S)$ if there exists such an H-S-Deletion that can be obtained by a σ_H-stitch onto S along G'; otherwise, S.

```
1: $\sigma_H \leftarrow diam(H)$
2: for $u \in N_G^{\sigma_H}[V(G')]$ do
3:     if $u \in (S \cap N_G^{\sigma_H}(V(G')))$ then
4:         $w'_0[u] \leftarrow 0$
5:     else
6:         $w'_0[u] \leftarrow w[u]$
7:     end if
8: end for
9: Let $S_{G'}$ be an optimal solution to the $H$-S-Deletion problem for $(G[N_G^{\sigma_H}[V(G')]], w'_0)$ by the
   algorithm obtained from Theorem 2.12.
10: $S^* \leftarrow (S \setminus V(G')) \cup S_{G'}$
11: if $w(S^*) < w(S)$ then
12:     return $S^*$
13: else
14:     return $S$
15: end if
```

The algorithm takes as input any weighted planar instance (G, w), an H-S-Deletion set S of G, and an induced subgraph $G' \subseteq G$. The algorithm starts by setting σ_H equal to the diameter of H. Next, it constructs the new weights w' by looping over all vertices in $N_G^{\sigma_H}[V(G')]$. For each vertex in $u \in (S \cap N_G^{\sigma_H}(V(G')))$, it sets its weight to zero in w' (i.e. $w'_0(u) = 0$). Otherwise, the weight is copied from the original weights w (i.e. $w'_0(u) = w(u)$). It continues to compute the H-S-Deletion $S_{G'}$ of the instance $(G[N_G^{\sigma_H}[V(G')]], w'_0)$ exactly. We use the FPT time algorithm created by Cygan, Marx, Pilipczuk, and Pilipczuk [14] stated in Theorem 2.12 that is parameterized by the treewidth of the input graph and the maximum size of a minimal separator in H. Next, it computes the candidate H-S-Deletion S^* obtained, implicitly, from the m-stitch operation of $S_{G'} \cup (V(G) \setminus N_G^{\sigma_H}[V(G')])$ onto S along G' (see Observation 6.3). Finally, should the weight of S^* be strictly lower than that of S, then it returns S^*; otherwise, it returns S.

Observation 6.3. Suppose that we invoke Algorithm 4 on any feasible input. The set $S_G := S_{G'} \cup (V(G) \setminus N_G^{\sigma_H}[V(G')])$ is a trivial H-S-Deletion set of G i.e. $S_{G'}$ intersects with all subgraphs isomorphic to H in $G[N_G^{\sigma_H}[V(G')]]$, while $(V(G) \setminus N_G^{\sigma_H}[V(G')])$ trivially does so for all such subgraphs in the remainder of the graph G. Furthermore, the set S^* (computed in line 9) can be obtained from the m-stitch of S_G onto S along G', since the m-stitch operation does not change solution S outside of $N_G^{\sigma_H}[V(G')]$.

We continue to formally prove that Algorithm 4 satisfies condition (iii) of a σ_H-LPO problem. However, first we need to prove a general result about the weight distribution when an m-stitch exists that yields a lower weight solution in Lemma 6.4.

Lemma 6.4. Suppose we have a weighted graph (G, w) with $w : V(G) \to \mathbb{R}_{>0}$, two vertex sets $S_1, S_2 \subseteq V(G)$, and any induced subgraph G' of G. If the weight of the m-stitch S_3 of S_2 onto S_1 along G' satisfies that $w(S_3) < w(S_1)$, then the following holds:

$$w((S_2 \setminus S_1) \cap N_G^m[V(G')]) < w((S_1 \setminus S_2) \cap V(G')).$$

Proof. We define the m-stitch S_3 obtained from stitching S_2 onto S_1 along G' as (see Definition 3.1)

$$S_3 := (S_1 \setminus V(G')) \cup (S_2 \cap N_G^m[V(G')]).$$

Since we assume that $w(S_3) < w(S_1)$, we obtain the following:

$$w(S_3) < w(S_1)$$
$$w((S_1 \setminus V(G')) \cup (S_2 \cap N_G^m[V(G')])) < w(S_1)$$

by definition the m-stitch S_3 of S_2 onto S_1 along G'

$$w(S_1 \setminus N_G^m[V(G')]) + w((S_1 \cup S_2) \cap N_G^m(V(G'))) + w(S_2 \cap V(G')) < w(S_1)$$

since these three terms are mutually exclusive

$$w((S_1 \cup S_2) \cap N_G^m(V(G'))) + w(S_2 \cap V(G')) < w(S_1 \cap N_G^m[V(G')])$$

subtract $w(S_1 \setminus N_G^m[V(G')])$ from both sides

$$w((S_2 \setminus S_1) \cap N_G^m(V(G'))) + w(S_2 \cap V(G')) < w(S_1 \cap V(G'))$$

subtract $w(S_1 \cap N_G^m(V(G')))$ from both sides

$$w((S_2 \setminus S_1) \cap N_G^m(V(G'))) + w((S_2 \setminus S_1) \cap V(G')) < w((S_1 \setminus S_2) \cap V(G'))$$

subtract $w((S_1 \cap S_2) \cap V(G'))$ from both sides

$$w((S_2 \setminus S_1) \cap N_G^m[V(G')]) < w((S_1 \setminus S_2) \cap V(G')).$$

Indeed our claim holds when such an m-stitch along G' exists. □

Using Lemma 6.4, we can formally prove that the H-S-Deletion problem satisfies property (iii) of a σ_H-LPO problem formally in Lemma 6.5.

Lemma 6.5. For any connected graph H, we have that given a weighted instance (G, w) where G is a planar graph and $w : V(G) \to \mathbb{R}_{>0}$, an induced subgraph $G' \subseteq G$ such that there exists an m-stitch S_3 of G of an unknown H-S-Deletion S_2 of G onto a known H-S-Deletion S of G with

$$w(S_3) < w(S),$$

then Algorithm 4 finds an H-S-Deletion S^* of G such that

$$w(S^*) < w(S);$$

otherwise, the algorithm returns S. Furthermore, for any connected graph H, the algorithm runs in $2^{O(t^{\mu_H})}|V(G)|^{O(1)}$ time, where t denotes the treewidth of $G[N_G^{\sigma_H}[V(G')]]$ and μ_H the maximum size of a minimal separator in H.

Proof. We prove this claim by first providing an upper-bound for the weight of $S_{G'}$ computed in line 6 and then we show that this upper-bound implies that $w(S^*) < w(S)$.

$$w_0'(S_{G'}) \leq w_0'(S_3 \cap N_G^{\sigma_H}[V(G')])$$

since $S_{G'}$ is optimal and $S_3 \cap N_G^{\sigma_H}[V(G')]$ is a candidate solution of $G[N_G^{\sigma_H}[V(G')]]$

$$= w_0'(((S \cup S_2) \cap N_G^{\sigma_H}(V(G'))) \cup (S_2 \cap V(G')))$$

by definition of the m-stitch S_3 of S_2 onto S along G'

$$= w_0'((S \cup S_2) \cap N_G^{\sigma_H}(V(G'))) + w_0'(S_2 \cap V(G'))$$

because $N_G^{\sigma_H}(V(G')) \cap V(G') = \emptyset$, the terms can be split

$$= w_0'(S \cap N_G^{\sigma_H}(V(G'))) + w_0'((S_2 \setminus S) \cap N_G^{\sigma_H}(V(G'))) + w_0'(S_2 \cap V(G'))$$

since $(S \cup S_2) = (S \cup (S_2 \setminus S))$

$$= w_0'((S_2 \setminus S) \cap N_G^{\sigma_H}(V(G'))) + w_0'(S_2 \cap V(G'))$$

since $w_0'(S \cap N_G^{\sigma_H}(V(G'))) = 0$

$$= w((S_2 \setminus S) \cap N_G^{\sigma_H}(V(G'))) + w(S_2 \cap V(G'))$$

because w and w_0' are equal unless $u \in (S_1 \cap N_G^{\sigma_H}(V(G')))$

$$= w((S_2 \setminus S) \cap N_G^{\sigma_H}[V(G')]) + w((S_2 \cap S) \cap V(G')) \tag{6.1}$$

by adding $w((S_2 \setminus S) \cap V(G'))$ to the first term and subtracting it from the second.

Next, we wish to show that the computed S^* of line 10 indeed improves S i.e. $w(S^*) < w(S)$. Observe that $(S^* \setminus N_G^{\sigma_H}[V(G')]) = (S \setminus N_G^{\sigma_H}[V(G')])$. Therefore, in order to prove that $w(S^*) < w(S)$, it suffices to show that $w(S^* \cap N_G^{\sigma_H}[V(G')]) < w(S \cap N_G^{\sigma_H}[V(G')])$. We do so as follows:

$$w(S^* \cap N_G^{\sigma_H}[V(G')]) = w(\big((S \setminus V(G')) \cup S_{G'}\big) \cap N_G^{\sigma_H}[V(G')])$$

by definition of $S^* := (S \setminus V(G')) \cup S_{G'}$

$$= w(\big((S \setminus V(G')) \cap N_G^{\sigma_H}[V(G')]\big) \cup S_{G'})$$

since $S_{G'} \setminus N_G^{\sigma_H}[V(G')] = \emptyset$

$$= w(\big(S \cap N_G^{\sigma_H}(V(G'))\big) \cup S_{G'})$$

because $(S \setminus V(G')) \cap N_G^{\sigma_H}[V(G')] = S \cap N_G^{\sigma_H}(V(G'))$

$$= w((S \cup S_{G'}) \cap N_G^{\sigma_H}(V(G'))) + w(S_{G'} \cap V(G'))$$

since $V(G')$ and $N_G^{\sigma_H}(V(G'))$ are disjoint

$$= w(S \cap N_G^{\sigma_H}(V(G'))) + w((S_{G'} \setminus S) \cap N_G^{\sigma_H}(V(G'))) + w(S_{G'} \cap V(G'))$$

since the disjoint union of S and $(S_{G'} \setminus S)$ is $(S \cup S_{G'})$

$$= w(S \cap N_G^{\sigma_H}(V(G'))) + w(S_{G'} \cap \left(N_G^{\sigma_H}(V(G')) \setminus S\right)) + w(S_{G'} \cap V(G'))$$

since $(S_{G'} \setminus S) \cap N_G^{\sigma_H}(V(G')) = S_{G'} \cap (N_G^{\sigma_H}(V(G')) \setminus S)$ implied by distributive laws over $\{\setminus, \cap\}$

$$= w(S \cap N_G^{\sigma_H}(V(G'))) + w(S_{G'} \cap \left((N_G^{\sigma_H}(V(G')) \setminus S) \cup V(G')\right))$$

because $(N_G^{\sigma_H}(V(G')) \setminus S)$ and $V(G')$ are mutually exclusive

$$= w(S \cap N_G^{\sigma_H}(V(G'))) + w'_0(S_{G'})$$

as w and w'_0 are equal for all vertices outside of $(S \cap N_G^{\sigma_H}(V(G')))$

$$\leq w(S \cap N_G^{\sigma_H}(V(G'))) + w((S_2 \setminus S) \cap N_G^{\sigma_H}[V(G')]) + w((S \cap S_2) \cap V(G'))$$

by inequality (6.1)

$$< w(S \cap N_G^{\sigma_H}(V(G'))) + w((S \setminus S_2) \cap V(G')) + w((S \cap S_2) \cap V(G'))$$

by Lemma 6.4 with $(S_1 = S), (S_2 = S_2), (S_3 = S_3), (G' = G')$, and $(m = \sigma_H)$

$$= w(S \cap N_G^{\sigma_H}(V(G'))) + w(S \cap V(G'))$$

since the disjoint union of $(S \setminus S_2)$ and $(S \cap S_2)$ is S

$$= w(S \cap N_G^{\sigma_H}[V(G')]).$$

Next, as argued in Observation 6.3, the set S^* can be obtained from an m-stitch of trivial solution $(S_{G'} \cup \left(V(G) \setminus N_G^{\sigma_H}[V(G')]\right))$ onto S along G'. Furthermore, by Lemma 6.2, we know that the problem allows for σ_H-stitching. Therefore, S^* is a feasible H-S-Deletion set of G with $w(S^*) < w(S)$ that is obtained by Algorithm 4 if there exists an m-stitch S_3 of some unknown solution onto S along G' with $w(S_3) < w(S)$. Should no improvement be found, then the algorithm returns S, which is trivially a feasible solution.

We proceed to analyze the running time of Algorithm 4. First, it is trivial to see that all lines of Algorithm 4 can be computed in polynomial time with respect to the size of the input, except for line 9 of the algorithm. Here we solve the H-S-Deletion problem exactly for $(G[N_G^{\sigma_H}[V(G')]], w'_0)$. By Theorem 2.12, we know that an FPT time algorithm parameterized by treewidth and the maximum size of a minimal separator in H (denoted as μ_H) exists with the same running time as given in this lemma statement, which implies the running time of Algorithm 4.

We conclude that Algorithm 4 meets the requirements of algorithm A_{HS} described in requirement (iii) of Definition 3.3. □

Finally, as a trivial consequence of Lemma 6.2, Lemma 6.5, and the fact that fixing a graph H implies that $diam(H) = O(1)$, we present Theorem 6.6.

Theorem 6.6. For any fixed connected graph H, the H-S-Deletion problem is $O(1)$-LPO.

6.2 MWDOM problem is locally planar optimized

In this section we prove that the MWDOM problem (see Definition 2.9) is 2-LPO. Our proving strategy is the same as the previous section, namely we prove that all three properties of 2-LPO problems (see Definition 3.3) hold for the MWDOM problem. We begin with property (i) i.e. we show that a trivial dominating set exists for any planar graph G.

Lemma 6.7. Suppose we are given any planar graph G, then the set $V(G)$ is a feasible dominating set of G.

Proof. The set $V(G)$ trivially satisfies that for every $u \in V(G)$ there exists a v in its closed neighborhood that is in $V(G)$. □

We continue to property (ii). Here we need to prove that the MWDOM problem allows for 2-stitching (see Definition 3.2) on (planar) graphs.

Lemma 6.8. The MWDOM problem allows for 2-stitching.

Proof. Let (G, w) be any instance to the MWDOM problem, $G' \subseteq G$ be any induced subgraph of G and S_1 and S_2 be any two dominating sets of G. It suffices to prove that the m-stitch S_3 of S_2 onto S_1 along G' yields a dominating set of G. Therefore we prove for any $u \in V(G)$ that there exists $v \in N_G^1[u]$ such that $v \in S_3$ with

$$S_3 := (S_1 \setminus V(G')) \cup (S_2 \cap N_G^2[V(G')]).$$

We apply case distinction on the position of $u \in V(G)$.

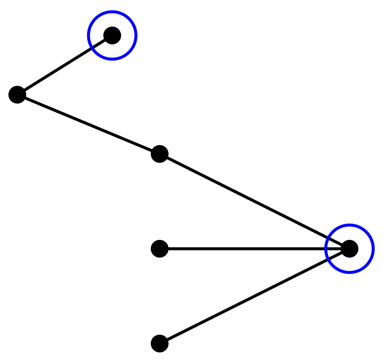
- $u \in N_G^1[V(G')]$: since $S_2 \cap N_G^2[V(G')] \subseteq S_3$ and no vertices of $N_G^1[V(G')]$ could be dominated from outside $N_G^2[V(G')]$, we know by feasibility of S_2 that there exists a $v \in N_G^1[\{u\}]$ with $v \in S_3$.
- $u \in (V(G) \setminus N_G^1[V(G')])$: since $(S_1 \setminus V(G')) \subseteq S_3$ and no vertices of $(V(G) \setminus N_G^1[V(G')])$ could be dominated from inside $V(G')$, we know by feasibility of S_1 that there exists a $v \in N_G^1[\{u\}]$ with $v \in S_3$.

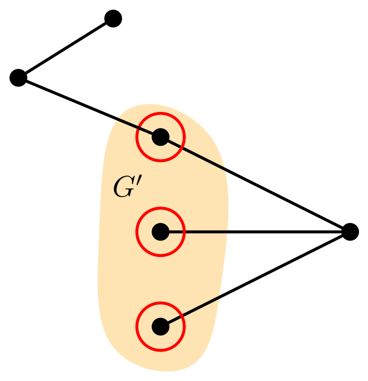
Together these two cases are exhaustive, and therefore, S_3 is a feasible dominating set of G. We conclude that the MWDOM problem allows for 2-stitching. □

Finally, we must prove property (iii) of 2-LPO problems by providing Algorithm 5 that satisfies the requirements of algorithm A_{MWDOM}. In the previous section we have shown that the H-S-Deletion problem is σ_H-LPO with $\sigma_H = diam(H)$. We approached this by setting the weight of the vertices in $S \cap N_G^{\sigma_H}(V(G'))$ to zero. However, it turns out that this approach does not generalize directly to the MWDOM problem. This is due to the following observation: let I be an optimal solution to the H-S-Deletion problem to any instance (G, w), then

$$\min\{w(S') \mid S' \text{ is an } H\text{-S-Deletion of } G'\} \leq w(I \cap V(G')), \qquad \forall G' \subseteq G.$$

Or, in other words, solving the H-S-Deletion problem exactly over any subgraph G' of G yields a lower-bound over the optimal solution I of G restricted to the vertices of G'. This however, is not always true for the MWDOM problem. Take any strict subgraph $G' \subset G$ and an optimal dominating set I of (G, w), then it could be the case that vertices of $V(G')$ are dominated by vertices in $N_G^1(V(G'))$. Therefore, should we solve the MWDOM exactly in G', we would not take into account that some vertices could be dominated from outside of G'. We show this in Figure 6.1.

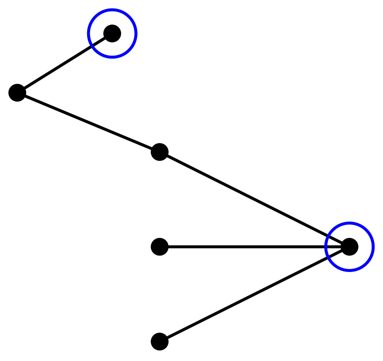

(a) Graph G and an optimal MWDOM I.

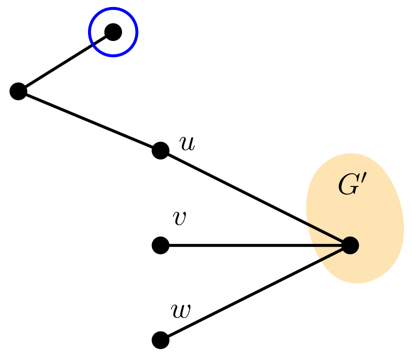


(b) Optimal MWDOM I' for instance restricted to $G' \subset G$.

Figure 6.1: Example that solving the MWDOM problem exactly for a subgraph G' of G does not yield a lower-bound i.e. $w(I') = 3 > w(I \cap V(G')) = 0$. All vertex weights are 1.

We essentially need to know which vertices of an optimal solution outside of $V(G')$ are dominating vertices of $V(G')$ i.e. the set $I \cap N_G^1(V(G'))$. This would allow us to update G' to include these vertices and set their weights to zero. Now we could dominate the vertices of $V(G')$ that are already dominated by I from outside of $V(G')$ free of cost. Clearly, we cannot assume that we explicitly know solution I. Alternatively, we could add all vertices of $N_G^1(V(G'))$ to G' and set their weights to zero and remove these added vertices from the solution afterwards would yield a lower-bound. However, this may yield infeasibility of the resulting dominating set of G. We show an example in Figure 6.2.


(a) Graph (G, w) and an optimal MWDOM I.



(b) Resulting infeasible solution if (G', w) is solved with u, v, and w added with a weight of zero.

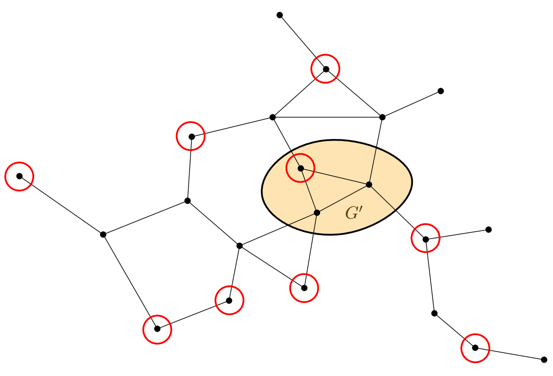
Figure 6.2: Counterexample that shows that using all of $N_G^1(V(G'))$ with weight of zero to solve (G', w) may yield an infeasible global dominating set of G. All vertex weights are 1.

We should not forget however, that our approach is based on 2-stitching. More specifically, we keep all vertices of $S \setminus V(G')$ in our solution. This shows that all vertices in $N_G^{2,2}(V(G'))$ (recall Definition 2.1) remain satisfied by S, because these could only be dominated from vertices in $N_G^3(V(G'))$. Therefore, we could add all vertices of $N_G^{3,3}(V(G'))$, all with a weight of zero to the instance, without sacrificing the feasibility of the global solution after the 2-stitching operation.

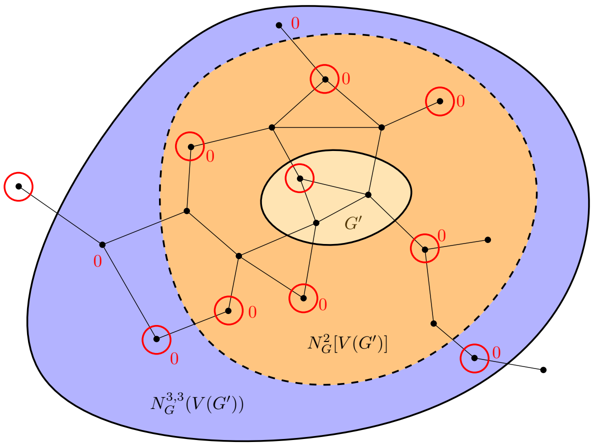
We wish to extend the approach of Algorithm 4 to also work for the MWDOM problem. Recall that in the H-S-Deletion problem, we solve the instance $(G[N_G^{\sigma_H}[V(G')], w_0')$ exactly to obtain a solution $S_{G'}$. Here w_0' is defined to set all vertices in $S \cap N_G^{\sigma_H}(V(G'))$ to zero. Now since the MWDOM problem allows for 2-stitching, we would solve the instance $(G[N_G^2[V(G')], w_0')$ instead where w_0' sets all vertices of $S \cap N_G^2(V(G'))$ to zero. We extend this by adding all vertices from $N_G^{3,3}(V(G'))$ to $G[N_G^2[V(G')]]$ and set the weights of its vertices to zero. Therefore, after the extension, we solve the instance $(G[N_G^3[V(G')], w_0')$ with

$$w_0'(u) := \begin{cases} 0, & \text{if } u \in (S \cap N_G^2(V(G'))) \cup N_G^{3,3}(V(G')) \\ w(u), & \text{otherwise} \end{cases}.$$

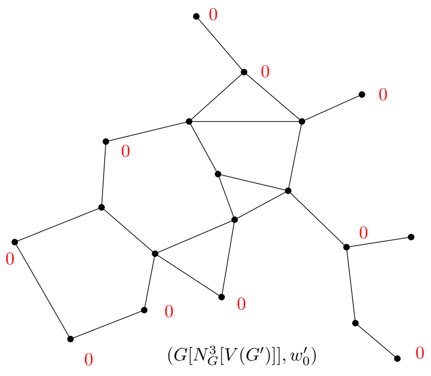
We show the construction of $(G[N_G^3[V(G')], w_0')$ in Figure 6.3.



(a) Dominating set S and G'.



(b) Construction of instance $(G[N_G^3[V(G')]], w_0')$, where the red weights are those updated by Algorithm 5.



(c) Final instance $(G[N_G^3[V(G')]], w_0')$ that Algorithm 5 solves exactly.

Figure 6.3: Construction of instance $(G[N_G^3[V(G')]], w_0')$ for an instance (G, w) together with a sub-optimal dominating set S. All vertex weights are 1, unless indicated otherwise.

Finally, we construct the dominating set S^* of G by removing all of $V(G')$ from S and by adding $(S_{G'} \cap N_G^2[V(G')])$ to S. Therefore, we remove all of $N_G^{3,3}(V(G'))$ from $S_{G'}$, which does not violate feasibility of S^*. We claim that this approach satisfies the requirements imposed by A_{MWDOM}. We provide the pseudocode in Algorithm 5.

Algorithm 5 A_{MWDOM} for the planar MWDOM problem

Input planar graph G, weight function $w : V(G) \to \mathbb{R}_{>0}$, induced subgraph G' of G, and a dominating set S of G.

Output dominating set S^* of G with $w(S^*) < w(S)$ if there exists such a dominating set that can be obtained as a 2-stitch onto S along G'; otherwise, S.

```
1: for $u \in N_G^3[V(G')]$ do
2:     if $u \in (S \cap N_G^{\sigma_H}(V(G'))) \cup (N_G^3(V(G')) \setminus N_G^2(V(G')))$ then
3:         $w'_0[u] \leftarrow 0$
4:     else
5:         $w'_0[u] \leftarrow w[u]$
6:     end if
7: end for
8: Let $S_{G'}$ be an optimal solution to the MWDOM problem for $(G[N_G^3[V(G')]], w'_0)$ by the
   algorithm obtained from Theorem 2.12.
9: $S^* \leftarrow (S \setminus V(G')) \cup (S_{G'} \cap N_G^2[V(G')])$
10: if $w(S^*) < w(S)$ then
11:     return $S^*$
12: else
13:     return $S$
14: end if
```

Observation 6.9. Suppose that we invoke Algorithm 5 on any feasible input. The set $S_G := S_{G'} \cup (V(G) \setminus N_G^2[V(G')])$ is a dominating set of G i.e. $S_{G'} \cup N_G^{3,3}(V(G')) \subseteq S_G$ is feasible dominating set of $N_G^2[V(G')]$. Furthermore, all vertices in $V(G) \setminus N_G^2[V(G')]$ are all in S_G, and hence, S_G is a feasible dominating set of G. Furthermore, the set S^* (computed in line 9) can be obtained from the m-stitch of S_G onto S along G', since the m-stitch operation does not change solution S outside of $N_G^2[V(G')]$.

We proceed to formally prove that Algorithm 5 satisfies the requirements of property (iii) of 2-LPO problems in Lemma 6.10 i.e. it satisfies the requirements of A_{MWDOM}. Note that this proof is very similar to Lemma 6.5, however there are a few subtle differences due to the extension of setting all weights in $N_G^{3,3}(V(G'))$ to zero.

Lemma 6.10. Given a weighted planar graph (G, w) with $w : V(G) \to \mathbb{R}_{>0}$, an induced subgraph $G' \subseteq G$ such that there exists an m-stitch S_3 of G of an unknown dominating set S_2 of G onto a known dominating set S of G with

$$w(S_3) < w(S),$$

then Algorithm 5 finds a dominating set S^* of G such that

$$w(S^*) < w(S);$$

otherwise, the algorithm returns S. Furthermore, the algorithm does so in $2^{O(t)}|V(G)|^{O(1)}$ time, where t denotes an upper-bound on the treewidth of $G[N_G^3[V(G')]]$.

Proof. We prove this claim by first providing an upper-bound for the weight of $S_{G'}$ computed in line 6 and then we show that this upper-bound implies that $w(S^*) < w(S)$.

$$w'_0(S_{G'}) \leq w'_0(S_3 \cap N_G^2[V(G')] \cup N_G^{3,3}(V(G')))$$

since $S_{G'}$ is optimal and $S_3 \cap N_G^3[V(G')] \cup N_G^{3,3}(V(G'))$ is a candidate solution of $G[N_G^3[V(G')]]$

$$= w_0'(S_3 \cap N_G^2[V(G')]))$$

since all vertices of $N_G^{3,3}(V(G'))$ have a weight of zero in w_0'

$$= w_0'(((S \cup S_2) \cap N_G^2(V(G'))) \cup (S_2 \cap V(G')))$$

by definition of the m-stitch S_3 of S_2 onto S along G'

$$= w_0'((S \cup S_2) \cap N_G^2(V(G'))) + w_0'(S_2 \cap V(G'))$$

because $N_G^2(V(G')) \cap V(G') = \emptyset$, the terms can be split

$$= w_0'(S \cap N_G^2(V(G'))) + w_0'((S_2 \setminus S) \cap N_G^2(V(G'))) + w_0'(S_2 \cap V(G'))$$

since $(S \cup S_2) = (S \cup (S_2 \setminus S))$

$$= w_0'((S_2 \setminus S) \cap N_G^2(V(G'))) + w_0'(S_2 \cap V(G'))$$

since $w_0'(S \cap N_G^2(V(G'))) = 0$

$$= w((S_2 \setminus S) \cap N_G^2(V(G'))) + w(S_2 \cap V(G'))$$

because w and w_0' are equal unless $u \in (S_1 \cap N_G^2(V(G')))$

$$= w((S_2 \setminus S) \cap N_G^2[V(G')]) + w((S_2 \cap S) \cap V(G')) \tag{6.2}$$

by adding $w((S_2 \setminus S) \cap V(G'))$ to the first term and subtracting it from the second.

Next, we wish to show that the computed S^* of line 9 indeed improves S i.e. $w(S^*) < w(S)$. Observe that $(S^* \setminus N_G^2[V(G')]) = (S \setminus N_G^2[V(G')])$. Therefore, in order to prove that $w(S^*) < w(S)$, it suffices to show that $w(S^* \cap N_G^2[V(G')]) < w(S \cap N_G^2[V(G')])$. We do so as follows:

$$w(S^* \cap N_G^2[V(G')]) = w\big(\big((S \setminus V(G')) \cup (S_{G'} \cap N_G^2[V(G')])\big) \cap N_G^2[V(G')]\big)$$

by definition of $S^* := (S \setminus V(G')) \cup (S_{G'} \cap N_G^2[V(G')])$

$$= w\big(\big((S \setminus V(G')) \cap N_G^2[V(G')]\big) \cup (S_{G'} \cap N_G^2[V(G')])\big)$$

by distributing intersection with $N_G^2[V(G')]$

$$= w\big(\big(S \cap N_G^2(V(G'))\big) \cup (S_{G'} \cap N_G^2[V(G')])\big)$$

because $(S \setminus V(G')) \cap N_G^2[V(G')] = S \cap N_G^2(V(G'))$

$$= w((S \cup S_{G'}) \cap N_G^2(V(G'))) + w(S_{G'} \cap V(G'))$$

since $V(G')$ and $N_G^2(V(G'))$ are disjoint

$$= w(S \cap N_G^2(V(G'))) + w((S_{G'} \setminus S) \cap N_G^2(V(G'))) + w(S_{G'} \cap V(G'))$$

since $(S \cup S_{G'})$ can be split into disjoint terms $(S_{G'} \setminus S)$ and S

$$= w(S \cap N_G^2(V(G'))) + w\big(S_{G'} \cap \big(N_G^2(V(G')) \setminus S\big)\big) + w(S_{G'} \cap V(G'))$$

since $(S_{G'} \setminus S) \cap N_G^2(V(G')) = S_{G'} \cap (N_G^2(V(G')) \setminus S)$ implied by distributive laws over $\{\setminus, \cap\}$

$$= w(S \cap N_G^2(V(G'))) + w\big(S_{G'} \cap \big((N_G^2(V(G')) \setminus S) \cup V(G')\big)\big)$$

because $(N_G^2(V(G')) \setminus S)$ and $V(G')$ are mutually exclusive

$$= w(S \cap N_G^2(V(G'))) + w_0'(S_{G'})$$

as w and w_0' are equal for all vertices outside of $(S \cap N_G^2(V(G'))) \cup N_G^{3,3}(V(G'))$

$$\leq w(S \cap N_G^2(V(G'))) + w((S_2 \setminus S) \cap N_G^2[V(G')]) + w((S \cap S_2) \cap V(G'))$$

by inequality (6.2)

$$< w(S \cap N_G^2(V(G'))) + w((S \setminus S_2) \cap V(G')) + w((S \cap S_2) \cap V(G'))$$

by Lemma 6.4 invoked on $(S_1 = S), (S_2 = S_2), (S_3 = S_3), (G' = G')$, and $(m = 2)$

$$= w(S \cap N_G^2(V(G'))) + w(S \cap V(G'))$$

since the disjoint union of $(S \setminus S_2)$ and $(S \cap S_2)$ is S

$$= w(S \cap N_G^2[V(G')]).$$

Next, as argued in Observation 6.9, the set S^* can be obtained from a 2-stitch of trivial solution $(S_{G'} \cup \big(V(G) \setminus N_G^2[V(G')]\big))$ onto S along G'. Furthermore, by Lemma 6.8, we know that the problem allows for 2-stitching. Therefore, S^* is a feasible dominating set of G with $w(S^*) < w(S)$ that is obtained by Algorithm 5 if there exists an m-stitch S_3 of some unknown solution onto S along G' with $w(S_3) < w(S)$. Should no improvement be found, then the algorithm returns S, which is a trivial feasible solution.

We proceed to analyze the running time of Algorithm 5. First, it is trivial to see that all lines of Algorithm 5 can be computed in polynomial time with respect to the size of the input, except for line 9 of the algorithm. Here we solve the MWDOM problem exactly for $(G[N_G^3[V(G')]], w_0')$. By Theorem 2.12, we know that an FPT time algorithm parameterized by treewidth exists with the same running time given in this lemma statement, which implies the running time of Algorithm 5.

We conclude that Algorithm 5 meets the requirements of algorithm A_{MWDOM} described in requirement (iii) of Definition 3.3. □

Now that we have proven that the MWDOM problem allows for a trivial solution for any (planar) graph (Lemma 6.7), allows for 2-stitching (Lemma 6.8), and we have provided an algorithm that satisfies the conditions of A_{MWDOM} (Lemma 6.10), we obtain Theorem 6.11:

Theorem 6.11. The MWDOM problem is 2-LPO.

6.3 A $(1+\epsilon)$-certified algorithm for the planar MWIS problem

Finally, we provide a $(1 + \epsilon)$-certified algorithm for the MWIS problem (see Definition 2.6) for connected planar graphs with positive integer weights. We included this $(1 + \epsilon)$-certified algorithm to show that the XP running time of the existing $(1+\epsilon)$-certified algorithm for planar MWIS problem by Angelidakis, Awasthi, Blum, Chatziafratis, and Dan [2] is not necessary to obtain a $(1 + \epsilon)$-certified algorithm for the planar MWIS problem. Recall from Proposition 2.7, that we can easily obtain an optimal solution for the MWIS problem when we know an optimal solution to the MWVC problem (see Definition 2.5). This is done by simply taking the complement of the MWVC.

Now this proposition allows us to design the $(1+\epsilon)$-certified algorithm for the planar MWIS problem as follows. Our restrictions on the input are the same as Algorithm 3 i.e. a connected planar graph G, strictly positive integer weights w, and a parameter $\epsilon > 0$. First, we observe that the H-S-Deletion with H being equal to a single edge yields the MWVC problem. This implies that the MWVC problem is 1-LPO, which we state formally in Corollary 6.12 from Theorem 6.6.

Corollary 6.12. The MWVC problem is 1-LPO.

Next, we run Algorithm 3 with Π set to the MWVC problem on (G, w) and ϵ and obtain a solution C to the MWVC problem together with weights w' such that w' certifies C. We proceed to create an independent set S by using Proposition 2.7 i.e. $S := (V(G) \setminus C)$. Finally, we return S and w' and claim that w' certifies S. We provide the pseudocode in Algorithm 6.

Algorithm 6 $(1 + \epsilon)$-certified algorithm for the MWIS problem

```
Input: connected planar graph $G$, weight function $w : V(G) \to \mathbb{Z}_{>0}$, parameter $\epsilon > 0$
Output: independent set $S$ of $G$, $(1 + \epsilon)$-perturbed weights $w'$ of $w$ that certifies $S$
1: $(C, w') \leftarrow$ Algorithm 3 for $H$-S-Deletion (with $H$ a single edge) invoked on $(G, w)$ and $\epsilon$
2: $S \leftarrow (V(G) \setminus C)$
3: return $(S, w')$
```

The proof that Algorithm 6 is a $(1 + \epsilon)$-certified algorithm for the MWIS problem when the instance (G, w) consists of a connected planar graph and positive integer weights is a trivial result from the fact that the MWVC is 1-LPO (Corollary 6.12), we can obtain an optimal MWIS from an optimal MWVC (Proposition 2.7), and by the correctness of Algorithm 3 (Theorem 5.9). We state this result formally in Theorem 6.13.

Theorem 6.13. Algorithm 6 is a $(1 + \epsilon)$-certified algorithm for the MWIS problem if the input consists of a connected planar graph G, positive integer weights w, and a parameter $\epsilon > 0$.

We have successfully populated locally planar optimized problems with the H-S-Deletion problem, the MWDOM problem, the MWVC problem, and implicitly, the MWIS problem. Therefore, by Theorem 5.9, Theorem 6.6, Theorem 6.11, Corollary 6.12, and Theorem 6.13, we obtain Corollary 6.14.

Corollary 6.14. Algorithm 3 is a $(1 + \epsilon)$-certified algorithm for the following optimization problems when the input (G, w) consists of a connected planar graph and $w : V(G) \to \mathbb{Z}_{>0}$:

- MWVC problem with a running time of $W \cdot 2^{O(1/\epsilon)}|V(G)|^{O(1)}$;
- For each fixed connected graph H, the H-S-Deletion problem can be solved in time $W \cdot 2^{O(1/\epsilon^c)}|V(G)|^{O(1)}$ for a constant $c \geq 1$;
- MWDOM problem with a running time of $W \cdot 2^{O(1/\epsilon)}|V(G)|^{O(1)}$;
- MWIS problem implicitly, by using Algorithm 6, with a running time of $W \cdot 2^{O(1/\epsilon)}|V(G)|^{O(1)}$,

with $W := \Sigma_{u \in V(G)} w(u)$.

We have reached the end of the explanation of our contributions. In the next chapter we give a summary of our findings, followed by a discussion of possible extensions to Algorithm 3, and future work and limitations.

Chapter 7

Conclusion

We provided $(1 + \epsilon)$-certified algorithms (with $\epsilon > 0$) for vertex-optimization problems on connected planar graphs with integer weights that satisfy the conditions of our introduced notion of m-LPO problems (see Definition 3.3). For any fixed vertex-optimization problem that admits a constant value for m, the algorithm runs in FPT time parameterized by $1/\epsilon$. We showed that the MWVC problem is 1-LPO, the MWDOM problem is 2-LPO, and for any fixed subgraph H, the H-S-Deletion problem is $O(1)$-LPO. By a linear-time reduction from the MWVC problem to the MWIS problem, we have obtained a $(1 + \epsilon)$-certified algorithm for the MWIS problem as well. Our algorithm extends Baker's technique for building PTASs on planar graphs (see Section 2.3). We made the algorithm iterative, meaning that we start from some trivial solution S and repeat the process of replacing S with lower-weight solutions found locally along subgraphs, by using our coined notion of m-stitching (see Definition 3.2). Once no more improvements of this form can be found, we use a stronger notion of the pigeonhole principle to prove that S is certified by the weight function obtained by perturbing the weights of all vertices that are not in S by $(1+\epsilon)$. Therefore, we have simultaneously improved the running time of the existing $(1 + \epsilon)$-certified algorithm obtained by Angelidakis, Awasthi, Blum, Chatziafratis, and Dan [2] for the MWIS problem on planar graphs with positive integer polynomially-bounded weights, while also providing a framework to design $(1+\epsilon)$-certified algorithms on planar graphs.

In this final chapter, we reflect on our work by discussing possible extensions, limitations, and future work towards encompassing $(1 + \epsilon)$-certified algorithms for all attainable combinatorial optimization problems on planar and, possibly, more general graphs that allow for a PTAS.

7.1 Overview of extensions

For the sake of presentation, time, and to avoid cumbersome definitions, we have limited our presentation to vertex-optimization problems with minimization objectives on connected planar graphs with positive integer weights.

We can extend our technique very easily to general planar graphs. Should the definition for m-LPO problems be extended such that the problem can be solved independently on connected components of a general planar graph, then we can apply Algorithm 3 independently for each connected component and proceed to merge its solutions. Furthermore, it is easy to convince oneself that the key correctness property of Algorithm 3 follows analogously for vertex-optimization problems with maximization objectives. Similarly, m-LPO problems (see Definition 3.3) can be defined for maximization objectives as well with minor effort. However, the m-stitching operation should take the intersection of two solutions in $N_G^m(V(G'))$, rather than the union for some induced subgraph G' of G. Moreover, the technique also generalizes to similar local problems that seek to minimize (or maximize) the weight of a set of edges.

Finally, we would like to note that the induced subgraph variant of the H-S-Deletion problem is likely $O(1)$-LPO as well for every fixed H. The problem is the same, except that it seeks to remove all induced subgraphs isomorphic to H. The work by Sau and Souza [36] showed that for any fixed connected graph H, there exists an FPT time algorithm parameterized by treewidth that solves the problem exactly.

7.2 Future work and limitations

We assume that our weight function solely of consists of integers, rather than allowing for real values as well. This allows us to bound the number of iterations of the while-loop by $W := \Sigma_{u \in V(G)} w(u)$ i.e. every improvement decreases the weight of S by at least 1, which can happen at most W times. Should our input consist of arbitrary positive real weights, then we might only decrease the weight of S by a value arbitrarily close to zero, and hence, our dependence of W would not be valid. Perhaps there exists some non-trivial way to analyze the running time of Algorithm 3 such that real weights functions are allowed. However, we leave this to future research.

Furthermore, compared to the existing $(1+\epsilon)$-certified algorithm for the MWIS problem [2], our technique does not exhibit the *robustness* property. The robustness property implies that the algorithm reports when the instance is $(1+\epsilon)$-stable. This allows the user to know that an optimal solution is obtained for the original weight function. However, it is often a difficult task to obtain the robustness property [29]. Therefore, it might not be possible to obtain a robust $(1+\epsilon)$-certified algorithm for m-LPO problems that runs in FPT time parameterized by m/ϵ. We leave this question to future research.

Finally, we would like to address the restrictions that are imposed on vertex-optimization problems characterized as m-LPO. Given that our framework extends Baker's technique to obtain PTASs, it does not come as a surprise that we roughly face the same limitations. Baker's technique works on local problems, meaning that some solution is feasible if, and only if, each vertex of the graph satisfies some condition in its (constant) closed neighborhood [18]. Now m-stitching (which is a requirement for m-LPO) implicitly measures how local a problem is (see Section 3.1). It can be shown that no constant value for m exists for global problems such that the problem is m-LPO. Take e.g. the planar weighted FVS problem (given an instance (G, w), we seek a minimum weight $S \subseteq V(G)$ such that $G[V(G) \setminus S]$ is acyclic), then for any constant m the following scenario could happen. *Suppose we have two solutions S_1 and S_2 of G and some cycle $C \subseteq V(G)$ such that S_1 exclusively contains a vertex of C in $V(G')$, while S_2 exclusively contains a vertex of C outside of $N_G^m[V(G')]$. Then, the m-stitch of S_2 onto S_1 along G' would not contain any vertices of C, and hence the resulting solution is infeasible.* As reviewed in Section 1.2, Baker's technique is not the only framework to obtain PTASs for problems on planar graphs, and therefore, it might be interesting to investigate the opportunities of extending other frameworks to $(1+\epsilon)$-certified algorithms. A natural choice would be the work by Demaine and MohammadTaghi [17], who generalized Baker and the Lipton-Tarjan separator-based approach [26] to obtain EPTASs for problems that also exhibit a global structure e.g. the planar FVS problem. However, Demaine, Hajiaghayi, and MohammadTaghi [15] posed that the framework does not generalize directly to the weighted case. Perhaps it is possible to generalize the framework to positive integer weights. Note that this is likely nontrivial, since e.g. there does not exist an EPTAS for the weighted planar FVS problem. Therefore, it is likely that the weighted case would no longer yield EPTASs, but we leave these questions to future research.

As a closing remark, given the vast number of ways researchers have designed PTASs for weighted problems, there is much opportunity to obtain $(1+\epsilon)$-certified algorithms.

Bibliography

[1] Alber, Bodlaender, Fernau, Kloks, and Niedermeier. Fixed parameter algorithms for dominating set and related problems on planar graphs. *Algorithmica*, 33(4):461–493, Aug 2002.

[2] Haris Angelidakis, Pranjal Awasthi, Avrim Blum, Vaggos Chatziafratis, and Chen 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.

[3] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, may 1998.

[4] Brenda S. Baker. Approximation algorithms for np-complete problems on planar graphs. *J. ACM*, 41(1):153–180, jan 1994.

[5] Daniel Bienstock and Nuri Ozbay. Tree-width and the sherali–adams operator. *Discrete Optimization*, 1(1):13–21, 2004.

[6] Yonatan Bilu and Nathan Linial. Are stable instances easy? *Comb. Probab. Comput.*, 21(5):643–660, sep 2012.

[7] Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1):1–45, 1998.

[8] Glencora Borradaile, Philp Klein, Dániel Marx, and Claire Mathieu. Algorithms for optimization problems in planar graphs (dagstuhl seminar 13421). In *Dagstuhl Reports*, volume 3. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.

[9] Eden Chlamtac and Madhur Tulsiani. *Convex Relaxations and Integrality Gaps*, pages 139–169. Springer US, Boston, MA, 2012.

[10] Vincent Cohen-Addad and Chris Schwiegelshohn. On the local structure of stable clustering instances. *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 49–60, 2017.

[11] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Integer programming.* Graduate Texts in Mathematics. Springer International Publishing, Basel, Switzerland, 2014 edition, December 2014.

[12] Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990.

[13] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms.* Springer Publishing Company, Incorporated, 1st edition, 2015.

[14] Marek Cygan, Dániel Marx, Marcin Pilipczuk, and Michał Pilipczuk. Hitting forbidden subgraphs in graphs of bounded treewidth. *Information and Computation*, 256:62–82, 2017.

[15] Erik Demaine and MohammadTaghi Hajiaghayi. The bidimensionality theory and its algorithmic applications. *Comput. J.*, 51:292–302, 05 2008.

[16] Erik D. Demaine, Fedor V. Fomin, MohammadTaghi Hajiaghayi, and Dimitrios M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and ¡i¿h¡/i¿-minor-free graphs. *J. ACM*, 52(6):866–893, nov 2005.

[17] Erik D. Demaine and MohammadTaghi Hajiaghayi. Bidimensionality: New connections between fpt algorithms and ptass. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2005)*, pages 590–601, Vancouver, British Columbia, Canada, January 23–25 2005.

[18] Erik D. Demaine and MohammadTaghi Hajiaghayi. *Approximation Schemes for Planar Graph Problems*, pages 59–62. Springer US, Boston, MA, 2008.

[19] Rodney G Downey and M R Fellows. *Parameterized complexity.* Monographs in Computer Science. Springer, New York, NY, 1999 edition, November 1998.

[20] Fedor V Fomin, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. Bidimensionality and eptas. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 748–759. SIAM, 2011.

[21] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Bidimensionality and kernels. *SIAM Journal on Computing*, 49(6):1397–1422, 2020.

[22] Michael R Garey and David S Johnson. *Computers and intractability.* W.H. Freeman, New York, NY, April 1979.

[23] Martin Grohe. Local tree-width, excluded minors,and approximation algorithms. *Combinatorica*, 23(4):613–632, Dec 2003.

[24] Tamir Hazan, George Papandreou, and Daniel Tarlow. *Bilu-Linial Stability*, pages 375–400. The MIT Press, 2016.

[25] Frank Kammer and Torsten Tholey. Approximate tree decompositions of planar graphs in linear time. *Theoretical Computer Science*, 645:60–90, 2016.

[26] Richard J Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

[27] Richard J Lipton and Robert Endre Tarjan. Applications of a planar separator theorem. *SIAM journal on computing*, 9(3):615–627, 1980.

[28] Konstantin Makarychev and Yury Makarychev. Certified Algorithms: Worst-Case Analysis and Beyond. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 49:1–49:14, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[29] Konstantin Makarychev and Yury Makarychev. *Perturbation Resilience*, page 95–119. Cambridge University Press, 2021.

[30] Konstantin Makarychev, Yury Makarychev, and Aravindan Vijayaraghavan. *Bilu–Linial Stable Instances of Max Cut and Minimum Multiway Cut*, pages 890–906. Society for Industrial and Applied Mathematics, 2014.

[31] Barnaby Martin, Daniël Paulusma, and Siani Smith. Hard problems that quickly become very easy. *Information Processing Letters*, 174:106213, 2022.

[32] USR Murty and Adrian Bondy. *Graph Theory (graduate texts in mathematics 244)*. Springer, 2008.

[33] Marcin Pilipczuk and Michał Pilipczuk. Lecture notes in sparsity, lecture 7: Structural measures, November 2019.

[34] Prasad Raghavendra. *Approximating Np-Hard Problems Efficient Algorithms and Their Limits*. PhD thesis, USA, 2009. AAI3377316.

[35] Benoît Rittaud and Albrecht Heeffer. The pigeonhole principle, two centuries before dirichlet. *The Mathematical Intelligencer*, 36(2):27–29, Jun 2014.

[36] Ignasi Sau and Uéverton dos Santos Souza. Hitting forbidden induced subgraphs on bounded treewidth graphs. *Information and Computation*, 281:104812, 2021.

[37] Anastasios Sidiropoulos. Math 8500 algorithmic graph theory lecture 18: Baker's technique, 2017.

[38] Jouko Väänänen. Second-order and Higher-order Logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2021 edition, 2021.