Fixed-Parameter Tractable Algorithms for Refined Parameterizations of Graph Problems

van Roozendaal, Tom

Award date:
2023

Department of Mathematics and Computer Science
Algorithms, Geometry & Applications

Fixed-Parameter Tractable Algorithms for Refined Parameterizations of Graph Problems

Master's thesis

Tom van Roozendaal
Eindhoven, February 2023

Supervision:
dr. Bart M. P. Jansen

Assessment committee:
dr. George H. L. Fletcher
dr. Bart M. P. Jansen
MSc. Jari J. H. de Kroon

Credits: 30

This is a public Master's thesis.

This Master's thesis has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct.

Abstract

This work contributes as a study on the parameterized complexity of established graph problems using novel parameter candidates. We will give FPT algorithms for solving Triangle Graph Packing, 4-Cycle Graph Packing and Steiner Tree, using refined parameters. These parameters are considered to be refined as they can become much smaller than the solution size or terminal set, which are considered to be standard parameter choices. We find $2^{\mathcal{O}(d)} \cdot n^{\mathcal{O}(1)}$ and $2^{\mathcal{O}(d \log d)} \cdot n^{\mathcal{O}(1)}$ time algorithms for Triangle Graph Packing and 4-Cycle Graph Packing respectively, provided with some $\mathcal{H}$-elimination forest of the input graph G, with depth d, where $\mathcal{H}$ denotes the class of triangle or 4-cycle free graphs. We introduce a $2^{\mathcal{O}(|S| \log |S|)} \cdot n^{\mathcal{O}(1)}$ time algorithm for Steiner Tree when given a multi-way cut S of the input graph G. These algorithms solve their perspective problem efficiently on a large range of inputs. To achieve our results, we use known FPT approaches based on solution size to solve subproblem instances in which their solution size is limited by the refined parameter. The results found in this paper can be used generalize an algorithm for Graph Packing based on elimination distance and to further investigate the impact of refined parameters on such graph problems.

Contents

Chapter 1

Introduction

Computational complexity theory provides a framework for analyzing the efficiency of algorithms for solving computational problems. While the traditional measure of complexity is the size of the input, this may not capture all aspects of a problem that impact its computational hardness. In such cases, additional parameters can be used to provide a more comprehensive picture of a problem's computational complexity. This is the core idea behind parameterized complexity theory [6, 11]. This theory asks to solve NP-hard problems using exact algorithms in time $f(k) \cdot n^{\mathcal{O}(1)}$, for some parameter k and unrestricted function $f(k)$. Then, these fixed-parameter tractable (FPT) algorithms solve the problem efficiently for instances in which the parameter is small, while the input size may be large.

A common approach to addressing NP-hard problems is by considering their solution size as a feasible parameter for such FPT algorithms [19, 4, 8]. In this work, we aim to solve several conventional graph problems by utilizing novel parameter candidates. This will expand our knowledge in the field of parameterized complexity, enhances our understanding of the relationships within the complexity class hierarchy and aids in the efficiency of solving numerous mathematical optimization problems on a large scale.

While formal definitions are postponed to Section 2, we will provide a high level description of the techniques employed in order to substantiate our results.

We will give refined FPT algorithms for a collection of graph problems, namely Triangle Graph Packing, 4-Cycle Graph Packing and the Steiner Tree problem. Notably, these are all NP-hard [20, 18] and are among the most fundamental problems in graph theory.

Triangle Graph Packing and 4-Cycle Graph Packing task to find the maximum number of vertex-disjoint induced copies of C_3 or C_4 in some graph G respectively. Graph packing problems have a multitude of applications in computer science, ranging from combinatorics, information theory and the design of efficient statistical experiments [25, 3].

Fellows et al. [14] have introduced algorithms for Graph Packing that can solve these problems within $2^{\mathcal{O}(|V(H)|k)} + n^{|V(H)|}$ time and $2^{\mathcal{O}(|V(H)|k)}$ space, using solution size as parameter k for a to-be-packed graph H. We will show in this paper that using elimination distance to some graph class $\mathcal{H}$ as parameter will benefit sparser input graphs,

as this parameter can be much smaller than the solution size. We will use the algorithm by Fellows et al. mentioned above for solving subproblem instances which are limited in solution size by the $\mathcal{H}$-elimination distance.

Elimination distance is a graph parameter that measures the complexity of a graph in terms of how quickly its vertices can be removed to obtain a graph conforming to some property [21, 22, 10]. The elimination distance to a graph class $\mathcal{H}$ of a graph G is the smallest number of rounds of vertex deletion required to obtain a graph within $\mathcal{H}$, when removing one vertex from each connected component in each round.

The $\mathcal{H}$-elimination distance of a graph can be captured in a graph decomposition called an $\mathcal{H}$-elimination tree [16, 17]. Then, the remaining components after the elimination process are positioned at the bottom of the tree. They are referred to as base components and belong to $\mathcal{H}$. Figure 1.1 depicts such an $\mathcal{H}$-elimination tree.

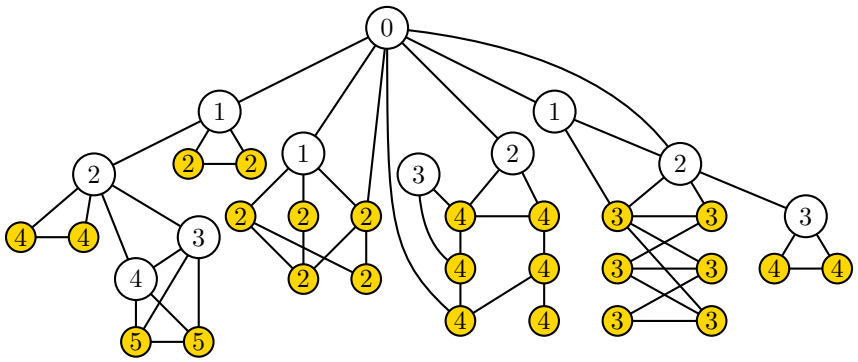


Figure 1.1: An $\mathcal{H}$-elimination forest where $\mathcal{H}$ denotes the class of triangle free graphs (containing no induced copy of C_3). The labels on the vertices denote the depth values, the white nodes correspond to the vertices removed within the rounds and the yellow vertices indicate the base components.

For triangle packing and 4-cycle packing we require $\mathcal{H}$ to be the class consisting of C_3 and C_4 free graphs respectively. The algorithms presented in this thesis uses these two $\mathcal{H}$-elimination forests as inputs.

In 2021, Jansen et al. introduced the elimination distance as a parameter to solve among others, Odd Cycle Traversal, Vertex Planarization and Cordal Vertex Deletion, using common FPT approaches such as iterative compression, branching and dynamic programming [16, 17]. Their paper shows us how to construct $\mathcal{H}$-elimination forests while proving bounds on the obtained forests and construction time [16, Table 1]. Specifically, for G satisfying $\mathbf{ed}_{\mathcal{H}}(G) \leq k$, an $\mathcal{H}$-elimination forests of depth $\mathcal{O}(k^3 \log k)$ can be found by an algorithm running in time $2^{\mathcal{O}(k)}$ when $\mathcal{H}$ belongs to the graph class forbidden subgraphs, provided that the number of subgraphs is finite, and $\mathcal{H}$ is hereditary. While using these elimination forests as input for our packing algorithms, we will use their construction as a black box.

The Steiner Tree problem asks to find a tree with a minumum number of edges induced in some graph G, connecting a set of appointed vertices (or terminals) $K \subseteq V(G)$. The applications of the steiner tree problem are among others, telecommunication, computer networks and VLSI design [1, 23, 9]. One approach for solving the steiner tree problem is presented by Cygan et al. in their book [7]. It runs in $2^{|K|}n^{\mathcal{O}(1)}$ time and uses the size of its terminal set as parameter. In this paper, we will be looking at using a multi-way cut as feasible parameter for solving Steiner Tree. We will use the algorithm presented by Cygan et al. for solving subproblem instances in which its terminal set size is limited by the multi-way cut size.

A multi-way cut for an instance of Steiner Tree with graph G and terminals $K \subseteq V(G)$ denotes a set of vertices $S \subseteq V(G)$ which when removed from G leaves connected components containing at most one terminal each. As such, the size of a multi-way cut can be seen as a measure of how well connected the set of terminal vertices is in such a problem instance. Since we may choose the terminal set as a multi-way cut, we have that a minimum size multi-way cut is always at most the number of terminals, making it a desirable parameter instead.

The book by Cygan et al. also establishes the NP-hardness of finding a multi-way cut, and a method for constructing a multi-way cut in time $\mathcal{O}(4^{|S|} \cdot |S|^3 \cdot (|V(G)| + |E(G)|))$ [7, Thm 8.54]. While using a multi-way cut as input for our steiner tree algorithm, we will use its construction as a black box.

The approaches provided in this paper utilize a technique called dynamic programming over trees and subsets [7]. This algorithmic strategy fills a table bottom-up by recursively defining how to solve larger problems based on the results of the smaller subproblems. The subproblem definition relies on a set of parameters which often directly contributes to the running time and space complexity of the algorithm. Unfortunately, this technique requires $\Omega(2^k)$ space as a result of its subproblems recursively relying on subsets.

1.1 Contributions and organization

The goal of this thesis is to make a contribution to the field of parameterized complexity by developing algorithms for the stated problems that are efficient in terms of their parameters, and to study the impact of these parameters on the computational complexity of the problems. We will present three main results that solve Triangle Graph Packing, 4-Cycle Graph Packing and Steiner Tree using FPT algorithms parameterized by their perspective refined parameters.

Lemma 2.2 proves that the $\mathcal{H}$-elimination distance where $\mathcal{H}$ forbids any graphs with a triangle exceeds at most three times the number of triangles in a triangle graph packing solution. For the triangle packing problem we will see an example in which some sparser graphs benefit from the $\mathcal{H}$-elimination distance as parameter, since solution size could greatly outgrow the $\mathcal{H}$-elimination distance in such scenarios.

Section 3.2 will introduce an algorithm to Triangle Graph Packing. It is build on the property that any edge in G must lie on a root-to-leaf path in an $\mathcal{H}$-elimination forest.

This ensures that every triangle can be found on a single search path in $\mathcal{H}$-elimination forest F. The algorithm produces the following result:

Theorem 1.1. *Let $\mathcal{H}$ be the graph class in which no graph contains an induced triangle. Then given an $\mathcal{H}$-elimination forest (F, χ) with depth d,* Triangle Graph Packing *is solvable in $2^{\mathcal{O}(d)} \cdot n^{\mathcal{O}(1)}$ time and $2^d \cdot n + 2^{\mathcal{O}(d)}$ space.*

The algorithm divides the internal nodes over the base components of F, such that they may form triangles. With a slight alteration, the approach in Section 3.2 can be used to solve the graph packing problem for a clique of any size, when provided with an $\mathcal{H}$-elimination forest which forbids any induced copies of the clique in its base components. This is not the case for any packing problem for which the required graph instance to be packed has vertices which do not share an edge. In Section 3.3 we will solve such a problem in the form of 4-Cycle Graph Packing, as a step towards generalising solving the graph packing problem using elimination distance altogether. This section produces the following result:

Theorem 1.2. *Let $\mathcal{H}$ be the graph class in which no graph contains an induced instance of C_4. Then given an $\mathcal{H}$-elimination forest (F, χ) with depth d,* 4-Cycle Graph Packing *is solvable in $2^{\mathcal{O}(d^2 \log d)} \cdot n^{\mathcal{O}(1)}$ time and $2^{\mathcal{O}(d \log d)} \cdot n$ space.*

For 4-Cycle Graph Packing it may occur that two vertices of a 4-cycle lie in different base components an $\mathcal{H}$-elimination forest. In addition to finding 4-cycles, we may also assign internal nodes in the forest to form P_3's, and combine them.

Finally we present an algorithm solving Steiner Tree using a multi-way cut S. Let $C = \{C_1, \ldots, C_m\}$ be the connected components of $G - S$. The method we introduce helps us put a bound on the time needed to solve subproblems in the components in C. We solve these problems for the components in C individually and combine them in a larger subproblem. This approach produces the following result:

Theorem 1.3. *Given graph G, and set of terminals $K \subseteq V(G)$ and multi-way cut $S \subseteq V(G)$,* Steiner Tree *can be solved in $2^{\mathcal{O}(|S| \log |S|)} \cdot n^{\mathcal{O}(1)}$ time and space.*

1.2 Related work

Graph Packing In 2004, Fellows et al. tackle the triangle packing problem by the use of cubic kernelization and a search tree datastructure, which establishes an FPT algorithm within time $\mathcal{O}(2^{2k \log k + 1.869k} n^2)$ [15, Thm 10.6.1], using solution size as parameter k. They also present a generalized algorithm able to pack any subgraph H with a running time of $\mathcal{O}(2^{2k|H| \log(k+|H|)} n^{|H|})$ [15, Thm 10.7.1]. In their introduction they mention an alternative strategy to solving the packing problem as a result of a color-coding approach by Alon, Yuster and Zwick [2] published in a paper in 1995. This color-coding approach involves considering numerous randomized colorings of a graph and a set of perfect hash functions from the set of vertices to the number of colors to find subgraphs isomorphic to some H. Theorem 6.3 in the alluded to paper can be used to obtain a $2^{\mathcal{O}(k)}$ algorithm for triangle packing, which however, according to the paper by Fellows et al., hides unfortunate

constants in its running time. In a later paper, Fellows et al. introduces an algorithm which employs this color-coding technique as a subroutine in combination with a kernelization and dynamic programming to obtain a faster algorithm of time $\mathcal{O}(n^{\mathcal{O}(|V(H)|)} + 2^{\mathcal{O}(|V(H)|k)})$ [14, Thm 2.], given the to-be-packed subgraph H. It is the latter result that will be used to solve base case subproblems in the algorithms presented in Sections 3.2 and 3.3.

Steiner Tree In 1971, Dreyfus an Wagner [12] proposed a dynamic programming algorithm solving Steiner Tree exactly in time $\mathcal{O}(3^k n + 2kn2 + n2\log n + n|E|)$, parameterized by the terminal set. It involves recursively looking for three connected subtrees each connected by some common vertex, of which two subtrees connect a disjoint subset of terminals. Erikson et al. build upon the dynamic programming approach by Dreyfus and Wagner [13] and obtain a faster running time of $\mathcal{O}(3^k n + 2^k(m + n\log n))$. The book by Cygan et al. presents an approach solving the unweighted version by employing a branching technique on the number of Hamiltonian cycles in the graph. This procedure runs in $2^{|K|}n^{\mathcal{O}(1)}$ time and polynomial space [7, Thm 10.6]. Chemani et al. present an approach solving the Steiner tree problem in time $\mathcal{O}(B^2_{tw+2} \cdot \mathbf{tw}(G) \cdot |V|)$ [5], where B_k denotes the number of partitions of a k-element set, also known as the k-th Bell number and $\mathbf{tw}(G)$ denotes the tree-width of graph G.

Chapter 2

Preliminaries

We consider simple undirected graphs without selfloops. A graph G has vertex set $V(G)$ and edge set $E(G)$. We use $n = |V(G)|$. For $A \subseteq V(G)$, the graph induced by A is denoted $G[A]$. We use $G - A$ for the graph $G[V(G) \setminus A]$. For a vertex set S, $N_G(S)$ denotes the open neighborhood of S. $N_G(S)$ includes all vertices directly adjacent to any vertex in S, excluding S; $N_G[S]$ denotes the closed neighborhood of S. $N_G[S] = N_G(S) \cup S$.

A tree is a connected acyclic graph, a forest is a disjoint union of trees. In a tree T with root r, we say that $t \in V(T)$ is an ancestor of $t' \in V(T)$ (and t' a descendant of t) if t lies on the path from r to t'. For some node $t \in V(T)$, we denote $U_t = \{u_1, \ldots, u_{|U_t|}\}$ as the set containing the children of t, and we assume it has a provided ordering. We use $\mathcal{A}(t)$ to denote the set of ancestors of t. we use P_k to denote a simple path consisting of k vertices, C_k to denote a cycle of k vertices and K_k to denote a clique of k vertices. The intersection between two graphs G and H result in a graph $G \cap H$ with vertex set $V(G) \cap V(H)$ and edge set $E(G) \cap E(H)$. The graph $G \cup H$ has vertex set $V(G) \cup V(H)$ and edge set $E(G) \cup E(H)$.

2.1 Graph Packing

Graph Packing
Input: A graph G and graph H.
Task: Find the maximum number of vertex disjoint copies of H induced in G.

Triangle Graph Packing is a problem instance of Graph Packing where $H = C_3$; a cycle with three vertices. 4-Cycle Graph Packing is a problem instance of Graph Packing where $H = C_4$; a cycle with four vertices.

We use the following result in Chapter 3.

Theorem 2.1. Graph Packing *can be solved in* $2^{\mathcal{O}(|V(H)|k)} + n^{|V(H)|}$ *time and* $2^{\mathcal{O}(|V(H)|k)}$ *space, parameterized by solution size* k.

Proof. The paper by Fellows et al. presents an approach to solve the packing problems in the desired time [14, Thm 2.]. The approach involves color-coding the vertices of the

graph based on hash functions and using dynamic programming for finding copies of H consisting of different colors within a specific coloring. ■

We may use triangle or $\triangle$ as shorthand for an instance of C_3 and we may use 4-cycle or $\square$ as shorthand for an instance of C_4. We will use $\textsc{Pack}_H(G)$ to denote the maximum cardinality of an H packing in graph G. Let G and H be graphs, and let P denote a packing witnessing the result of Graph Packing. Packing P contains graphs isomorphic to H, and $|P|$ denotes the number of vertex disjoint copies of H induced in G; $|P| = \textsc{Pack}_H(G)$.

Definition 2.1. Let G be a graph and A be a matching on vertices $V(G)$.
$\textsc{Pack}_{C_4,P_3}(G, A)$ denotes integer k such that G contains a packing X_1 of k vertex-disjoint C_4's and a packing X_2 of vertex-disjoint P_3's such that:

(i) $V(X_1) \cap V(X_2) = \emptyset$,

(ii) for each $(t, t') \in A$, X_2 contains a vertex disjoint P_3 of the form (t, u, t'),

(iii) $|X_1|$ is largest,

or $-\infty$ if no such X_1, X_2 exist.

The algorithm referred to in Theorem 2.1 can be used to solve an instance of $\textsc{Pack}_{C_4,P_3}(G, A)$ within time $2^{\mathcal{O}(k)} + n^{\mathcal{O}(1)}$ and $2^{\mathcal{O}(k+|A|)}$ space. It involves finding colorful induced copies of P_3 consisting of a subset of colors, after which a maximum number colorful induced copies of C_4 are found on the remaining subset of colors. The detailed inner workings of this approach is omitted as it is outside the scope of this work.

Let A be a matching of some set W. We say a set $W' \subseteq W$ is *consistent* with A if and only if for every $(t, t') \in A$, if $t \in W'$ then $t' \in W'$. Let A and A' be matchings over some set W. We say that A' is *in agreement* with A if and only if for every $(t, t') \in A$, if t is matched in A', then it is matched with t'.

A graph class defines a set containing all graphs conforming to some property. We will use $\mathcal{H}$ to denote a graph class, the graph classes used in this paper are the class in which no graph contains an induced triangle and the class in which no graph contains an induced instance of C_4. An $\mathcal{H}$-elimination tree is a data structure capturing how close an original graph conforms to the properties of some graph class $\mathcal{H}$. An $\mathcal{H}$-elimination forest is a collection of $\mathcal{H}$-elimination trees. We will use F to denote $\mathcal{H}$-elimination forests and T for $\mathcal{H}$-elimination trees. Each tree in an $\mathcal{H}$-elimination forest will have its own root.

Definition 2.2 ([16])**.** For a graph class $\mathcal{H}$, an $\mathcal{H}$-elimination forest of graph G is pair (F, χ) where F is a rooted forest and $\chi : V(F) \to 2^{V(G)}$, such that:

(i) For each internal node t of F we have $|\chi(t)| = 1$.

(ii) The sets $(\chi(t))_{t \in V(F)}$ form a partition of $V(G)$.

(iii) For each edge $uv \in E(G)$, if $u \in \chi(t_1)$ and $v \in \chi(t_2)$ then $t1, t2$ are in ancestor-descendant relation in F.

(iv) For each leaf t of F, the graph $G[\chi(t)]$, called a base component, belongs to $\mathcal{H}$.

We say that the *depth* of F is the maximum number of edges on a root-to-leaf path. The $\mathcal{H}$-elimination distance $\mathbf{ed}_{\mathcal{H}}(G)$ is the minimum depth of an $\mathcal{H}$-elimination forest for G, we also use d to denote depth of such a forest. We may refer to $\chi(S)$ by stating *the vertices corresponding to/the nodes of* S.

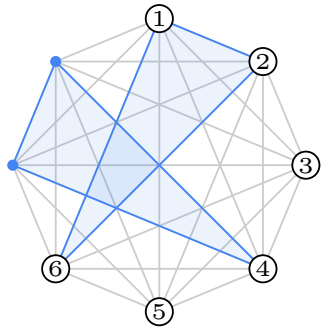
Lemma 2.2. *Let $\mathcal{H}$ be the class of graphs in which no graph contains an induced triangle. Let $G = (V, E)$ be a graph and let F denote an $\mathcal{H}$-elimination forest witnessing the smallest $\mathcal{H}$-elimination distance of G; $\mathbf{ed}_{\mathcal{H}}(G)$. Suppose that for some G as input,* Triangle Graph Packing *has solution size k. We have $\mathbf{ed}_{\mathcal{H}}(G) \leq 3k$.*

Proof. Let P denote a triangle packing in G with $|P| = k$. Consider the $\mathcal{H}$-elimination process of removing all vertices from all triangles in packing P. Let F be the $\mathcal{H}$-elimination forest witnessing this process. Observe that at most $3k$ vertices are removed in this process, and since P is optimal we cannot have that the base components of F contain any induced triangles. Since F is a valid $\mathcal{H}$-elimination forest, we have that $\mathbf{ed}_{\mathcal{H}}(G) \leq \mathit{depth}(F)$.

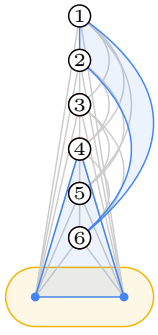
In the worst case, we have that each vertex removed in this process is removed from the same connected component, resulting in F being an $\mathcal{H}$ elimination forest with one base component per tree. We have $\mathit{depth}(F) \leq 3k$, implying $\mathbf{ed}_{\mathcal{H}}(G) \leq 3k$. ■

Example 2.1. Suppose we are solving the Triangle Graph Packing problem on $G = K_n$, a graph consisting of a clique with n vertices. Consider an $\mathcal{H}$-elimination process for G and its resulting $\mathcal{H}$-elimination forest F where $\mathcal{H}$ denotes the class of graphs in which no graph contains an induced triangle.

Since G is a clique, we must remove all but two vertices from G in order to have it not contain any induced triangle. As we may only remove a single vertex from each connected component per round, the number of rounds this will take is equal to $n - 2$; we have $\mathit{depth}(F) = n - 2$. An example is depicted in Figure 2.1.



(a) A clique with a possible Triangle Graph Packing in blue. The numbered vertices propose an $\mathcal{H}$-elimination process.

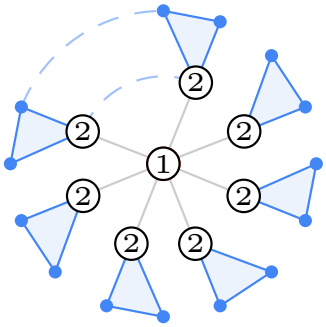


(b) $\mathcal{H}$-elimination forest of the graph depicted in (a). Yellow area shows the base component.

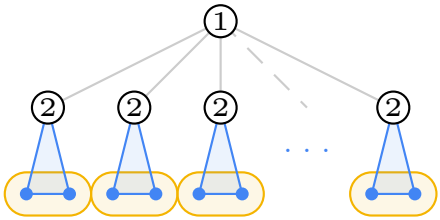
Figure 2.1: A clique and a possible $\mathcal{H}$-elimination forest. Where $\mathcal{H}$ denotes the class of graphs in which no graph contains an induced triangle.

The maximum number of vertex disjoint triangles in a clique is equal to $\lfloor \frac{n}{3} \rfloor$, as any set of three vertices in $V(G)$ will form a triangle. If we consider larger and larger cliques, the ratio between the $\mathcal{H}$-elimination distance and the solution size will approach 3. This implies that the bound proposed in Lemma 2.2 is a tight bound in the context of cliques.

Now consider a star graph where instead of the center vertex being connected to multiple single vertices, it is connected to a vertex of multiple vertex disjoint triangles (see Figure 2.2).



(a) A star-like graph with a possible Triangle Graph Packing in blue. The numbered vertices propose an $\mathcal{H}$-elimination process.



(b) $\mathcal{H}$-elimination forest of the graph depicted in (a). Yellow areas shows the base components.

Figure 2.2: A star-like graph and a possible $\mathcal{H}$-elimination forest. Where $\mathcal{H}$ denotes the class of graphs in which no graph contains an induced triangle.

For this example, we see that while the depth of the $\mathcal{H}$-elimination forest is small, the number of vertex disjoint triangles can be infinitely large. These examples show us that for dense graphs, the $\mathcal{H}$-elimination distance approaches the bound shown in Lemma 2.2, whereas sparser graphs could have an $\mathcal{H}$-elimination distance far smaller than the solution size. This example can be applied to any graph packing problem, while perhaps the factor in Lemma 2.2 might change.

2.2 Steiner Tree

STEINER TREE
Input: A graph $G = (V, E)$ and terminal set $K \subseteq V$.
Task: Find a subgraph $F \subseteq E(G)$ with a minimum number of edges induced in G which connects the vertices of K.

We use the following result in Chapter 4.

Theorem 2.3. *Given graph G, and set of terminals $K \subseteq V(G)$,* STEINER TREE *can be solved in $2^{|K|}n^{\mathcal{O}(1)}$ time in polynomial space.*

Proof. In the book by Cygan et al., an approach is presented to solve the Steiner Tree problem in the desired time and space [7, Thm 10.6]. The algorithm builds on an earlier presented algorithm that counts the number of hamiltonian cycles, and branches on these walks. This method only works for the unweighted steiner tree problem. ■

Definition 2.3. For graph G and $K \subseteq V(G)$, let $\textsc{Steiner}(G, K) :=$ minimum Total weight of a tree in G connecting the vertices of K. If no such tree exists (i.e. if the vertices of K are not connected), then this function evaluates to ∞.

For graphs G and H, Let $G \hat{\cap} H$ denote the graph with vertex set $\bigcup_{uv \in E(G) \cap E(H)} \{u, v\}$ and edge set $E(G) \cap E(H)$.

Observation. If G is a graph with terminal set $K \subseteq V(G)$ and $S \subseteq V(G)$ is a multiway cut for K, then for any Steiner tree $F \subseteq E(G)$ of G, the following holds:

1. F is connected; it consists of a single connected component.
2. F is acyclic. From any solution F containing a cycle we may remove an edge from that cycle to obtain a valid steiner tree with fewer edges.
3. Let $C = \{C_1, \ldots, C_m\}$ denote the set of connected components of $G - S$. Then each subgraph $C_i \in C$ contains at most one terminal and $N_G(C_i) \subseteq S$. Therefore if $|K| \geq 2$, F contains at least one vertex from S.

For a partition A of set U and $D \subseteq U$ define the merger of (A, D) as the partition obtained from A as follows:

(i) Let $A' \subseteq A$ contain the sets in A which contain at least one element of D.

(ii) Then, $\textsc{Merger}(A, D) := (A \setminus A') \cup \{\bigcup_{B \in A'} B\}$.

Example 2.2. Suppose we have $A = \{\{a\}, \{b, c\}, \{d, e\}, \{f\}, \{g, h, i\}\}$ and $D = \{a, c, h, j\}$. Then, $A' = \{\{a\}, \{b, c\}, \{g, h, i\}\}$ and $\textsc{Merger}(A, D) = \{\{d, e\}, \{f\}\} \cup \{\{a\} \cup \{b, c\} \cup \{g, h, i\}\} = \{\{d, e\}, \{f\}, \{a, b, c, g, h, i\}\}$.

Chapter 3

Solving Graph Packing Problems by Elimination Distance

3.1 Constructing elimination schemes

The graph packing algorithms presented in this chapter will require two types of $\mathcal{H}$-elimination forests as input. For triangle packing, we require an $\mathcal{H}_1$-elimination forest where $\mathcal{H}_1$ is the class of graphs in which no graph contains an induced triangle. For C_4 packing, we require an $\mathcal{H}_2$-elimination forest where $\mathcal{H}_2$ is the class of graphs in which no graph contains an induced cycle of four. Both of these graph characterizations belong to the superclass of forbidden subgraphs and can therefore be constructed by the earlier mentioned paper by Jansen et al. [16, Table 1].

3.2 Triangle packing

In this section we will introduce a fixed-parameter tractable algorithm to compute a solution to Triangle Graph Packing in some connected graph G, parameterized by the depth of a given $\mathcal{H}$-elimination tree (T, χ), where $\mathcal{H}$ denotes the graph class in which no graph contains an induced triangle.

By the construction of T, we have that the nodes representing the vertices for any triangle in G must lie on a single root-to-leaf path in T. Moreover, as we have that the leaves of T belong to $\mathcal{H}$, it must be that any triangle must have at least one of its endpoints represented by an internal node in T. These observations combined imply that the represented nodes of the endpoints of a triangle in G must have an ancestor-descendant relation in T. This will be used in the algorithm as defined below.

Theorem 3.1. *Let $\mathcal{H}$ be the graph class in which no graph contains an induced triangle, and for which* Triangle Graph Packing *is FPT solvable in $\mathcal{O}(2^k + n^{\mathcal{O}(1)})$ time, parameterized by solution size. Then given an $\mathcal{H}$-elimination forest (F, χ) with depth d,* Triangle Graph Packing *is solvable in $2^{\mathcal{O}(d)} \cdot n^{\mathcal{O}(1)}$ time and $2^d \cdot n + 2^{\mathcal{O}(d)}$ space.*

Proof. Given an $\mathcal{H}$-elimination tree (T, χ), we define a dynamic programming table as follows:

Definition 3.1. For each node $t \in V(T)$, set $S \subseteq \{t' \mid t' \text{ is a ancestor of } t\}$ and integer $0 \leq i \leq |U_t|$, entry $M[t, S, i]$ denotes the size of a maximum triangle packing in the graph induced by the vertices in G corresponding to the union of the subtrees rooted at the first i children of t, and the nodes in S; i.e. $G[\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_i}) \cup \chi(S)]$.

The maximum cardinality of a triangle packing in G can be found by checking cell $M[r, \{r\}, |U_r|]$, where r denotes the root of T.

The idea of the approach is to partition the ancestors S of some node t over its children U_t, such that each of t's children may only use a subset of S to form triangles with, and no element of S is used more than once. To do this structurally, we consider one child of t at a time, and decide which of the ancestors will be assigned to the considered child. This process is shown in Figure 3.1.

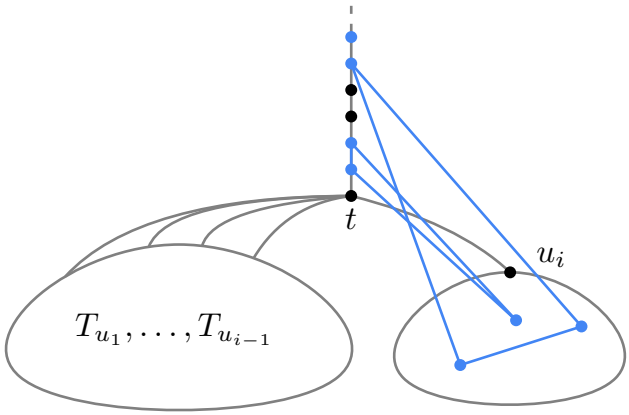


Figure 3.1: Assigning ancestors to child u_i to form C_3's.

As a base case of our algorithm, we have the case in which no children of t are considered ($i = 0$), and obtain $\textsc{Pack}_{C_3}(G[\chi(S)])$ using an FPT algorithm parameterized by solution size. As we have that the triangles must contain at least one element from $\chi(S)$, we have that for this case, the FPT algorithm by solution size will find a number of triangles which cannot exceed the number of ancestors, and is therefore bounded by the depth of T. If we consider the first i children of some node t ($1 \leq i \leq |U_t|$), we look for a subset S' of S such that the sum of the number of triangles from u_i with S' and the first $i-1$ children with $S \setminus S'$ is largest. The recurrence becomes as follows:

$$M[t, S, i] = \begin{cases} \textsc{Pack}_{C_3}(G[\chi(S)]) & \text{if } i = 0 \\ \max_{S' \subseteq S}\{M[t, \quad S \setminus S', \quad i-1] \quad + \\ \qquad\qquad M[u_i, \quad S' \cup \{u_i\}, \quad |U_{u_i}|]\} & \text{if } 1 \leq i \leq |U_t| \end{cases} \tag{3.1}$$

The table must be filled by considering nodes t with increasing height, for all possible subsets of S, while increasing i. A maximum cardinality triangle packing can be constructed using basic dynamic programming techniques.

Correctness When referring to properties in this section, we refer to the properties as provided in Definition 3.1 unless otherwise stated. The correctness proof for the base case of the recurrence directly follows from the definition of M. In order to complete the proof, we will show correctness for the second case.
($M[t, S, i] \geq \max_{S' \subseteq S}\{\ldots\}$) Choose an arbitrary $S' \subseteq S$. Let P_1 denote a maximum triangle packing of the vertices corresponding to: the nodes in the subtrees rooted at its first $i-1$ children of t and $S \setminus S'$, i.e. $G[\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_{i-1}}) \cup \chi(S \setminus S')]$. By definition, the cardinality of P_1 is stored in $M[t, S \setminus S', i-1]$. Let P_2 denote a maximum triangle packing of the vertices corresponding to the subtree rooted at u_i and S', i.e. $G[\chi(T_{u_i}) \cup \chi(S')]$. Following the definition of the recurrence, the cardinality of P_2 is stored in $M[u_i, S' \cup \{u_i\}, |U_{u_i}|]$. Observe that the combination of P_1 and P_2 is a triangle packing of $G[\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_i}) \cup \chi(S)]$. Let P be a maximum triangle packing in $G[\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_i}) \cup \chi(S)]$. We have

$$|P| \geq |P_1| + |P_2| = M[t, S \setminus S', i-1] + M[u_i, S' \cup \{u_i\}, |U_{u_i}|]$$

This holds for any arbitrary S', hence

$$M[t, S, i] \geq \max_{S' \subseteq S}\{M[t, S \setminus S', i-1] + M[u_i, S' \cup \{u_i\}, |U_{u_i}|]\}$$

($M[t, S, i] \leq \max_{S' \subseteq S}\{\ldots\}$) Let P be a maximum triangle packing in $G[\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_i}) \cup \chi(S)]$. For a triangle $\triangle \in P$, we distinguish the following cases:

1. $\triangle$ has a vertex in $\chi(T_{u_i})$ but not in $\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_{i-1}})$. Let $P_1 \subseteq P$ denote the triangles belonging to this case.
2. $\triangle$ has a vertex in $\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_{i-1}})$ but not in $\chi(T_{u_i})$. Let $P_2 \subseteq P$ denote the triangles belonging to this case.
3. $\triangle$ has no vertices in either $\chi(T_{u_i})$ or $\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_{i-1}})$. Let $P_3 \subseteq P$ denote the triangles belonging to this case.

Note that these cases cover all triangles in P; the case in which $\triangle$ has vertices in both $\chi(T_{u_i})$ and $\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_{i-1}})$ does not occur, as by property (iii) of elimination forests it must be that an edge $\{u, v\} \in E(G)$ implies an ancestor-descendant relation between nodes t and t', where $u \in \chi(t)$ and $v \in \chi(t')$. Trivially, for any pair (t, t') where $t \in T_{u_1} \cup \ldots \cup T_{u_{i-1}}$ and $t' \in T_{u_i}$, this relationship does not exist. By this fact, it must be that the triangles in P_1 may only have vertices in $\chi(T_{u_i})$ or in $\chi(S)$, and likewise, P_2 may only have vertices in $\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_{i-1}})$ or in $\chi(S)$. The triangles in P_3 must be contained in $\chi(S)$. Let $S^* \subseteq S$ be the set of nodes covering all vertices in triangles belonging to P_1, which are also in S. Due to the optimality of P, it must be that P_1 is a maximum triangle packing of the graph $G[\chi(T_{u_i}) \cup \chi(S^*)]$, and by definition, $|P_1| \leq M[u_i, S^* \cup \{u_i\}, |U_{u_i}|]$. Similarly, $P_2 \cup P_3$ is a maximum triangle packing of the graph $G[\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_{i-1}}) \cup \chi(S \setminus S^*)]$, implying $|P_2 \cup P_3| \leq M[t, S \setminus S^*, i-1]$. We have

$$\begin{aligned} M[t, S, i] = |P| &\leq M[u_i, S^* \cup \{u_i\}, |U_{u_i}|] + M[t, S \setminus S^*, i-1] \\ &\leq \max_{S' \subseteq S}\{M[u_i, S' \cup \{u_i\}, |U_{u_i}|] + M[t, S \setminus S', i-1]\} \end{aligned}$$

This concludes the proof.

To solve the packing problem on a disconnected graph, it suffices to run the algorithm individually on each connected component.

Time and space analysis To analyse the running time of the algorithm, we will consider the cases of the recurrence separately and sum the contribution of each case to the total running time.
For the base case, the number of corresponding entries in M is equal to the *number of nodes in* T $\times$ *number of subsets of* $S = |V(T)| \cdot 2^d$. For each of these entries, $\textsc{Pack}_{C_3}(G[\chi(S)])$ needs to be computed. Observe that $G[\chi(t)]$ belongs to $\mathcal{H}$, and therefore does not contain any triangles; we have seen before that this implies that all triangles in $G[\chi(S)]$ must have an endpoint in S. As such, we have that $\textsc{Pack}_{C_3}(G[\chi(S)])$ cannot exceed $|S|$, and by construction, we have that $|S| \leq d$. By this fact, using the color coding technique by M. Fellows et al. [14] as mentioned in Theorem 2.1, the $\textsc{Pack}_{C_3}(G[\chi(S)])$ can be found in $2^{\mathcal{O}(d)} \cdot n^{\mathcal{O}(1)}$ time. This totals to $2^d \cdot n \cdot (2^{\mathcal{O}(d)} + n^{\mathcal{O}(1)}) = 2^{\mathcal{O}(d)} \cdot n^{\mathcal{O}(1)}$ time overall for the base case.

Observe that the time needed to evaluate the step case of the recurrence is proportional to $2^{|S|}$, as we have to consider all possible subsets of ancestors. We can therefore bound the time needed to evaluate table cells of the second type as $O(\sum_{t \in V(T)} \sum_{S \subseteq \mathcal{A}(t)} \sum_{1 \leq i \leq |U_t|} 2^{|S|})$. We bound the latter expression by considering the contributions of sets S of different sizes separately. Note that in the calculations below, we use that the total number of child-parent relations in T is equal to the number of edges of the tree, and therefore bounded by n.

$$
\begin{aligned}
\sum_{t \in V(T)} \sum_{S \subseteq \mathcal{A}(t)} \sum_{1 \leq i \leq |U_t|} 2^{|S|} &\leq \sum_{t \in V(T)} \sum_{S \subseteq \mathcal{A}(t)} 2^{|S|} && \text{as } i \text{ contributes at most } n \text{ times} \\
&\leq n \cdot \sum_{S \subseteq \mathcal{A}(t)} 2^{|S|} && \text{since we have } n \text{ options for } t \in V(T) \\
&\leq n \cdot \sum_{j=0}^{d} \binom{d}{j} \cdot 2^j && \text{considering the sizes of } S \text{ separately} \\
&\leq 3^d \cdot n && \text{by the binomial theorem}
\end{aligned}
$$

The dominating factor of the total running time becomes that of the base case, which results in the desired running time.

The required space to run the algorithm corresponds to the number of cells in table M plus the time it takes to resolve $\textsc{Pack}_{C_3}(G[\chi(S)])$. The number of cells in table m is at most $|V(T)| \times 2^d \times |V(T)| = 2^d \cdot n^2$. However, the combination of t and i effectively loops over all nodes in T once, which implies both parameters contributing together a factor of $|V(T)|$, resulting in requiring only $|V(T)| \times 2^d \leq 2^d \cdot n$ space. The space required to resolve $\textsc{Pack}_{C_3}(G[\chi(S)])$ is $2^{\mathcal{O}(d)}$. Then, the total space required is $2^{\mathcal{O}(d)} \cdot n$.

■

3.3 4-Cycle packing

The approach introduced in this section will come in the form of an FPT algorithm parameterized by the depth of a given $\mathcal{H}$-elimination tree (T, χ), where $\mathcal{H}$ denotes the graph class in which no graph contains an induced instance of C_4.

Similar to the approach for triangle packing, we may partition the ancestors over the children of some node t to form 4-cycles with. However, as two opposing vertices in a 4-cycle do not necessarily share an edge in G, it could be that two vertices of a 4-cycle live in two distinct subtrees of T. The approach we will use to find these 4-cycles is to search for and combine two disjoint copies of induced instances of P_3's with equal endpoints.

Theorem 3.2. *Let $\mathcal{H}$ be the graph class in which no graph contains an induced instance of C_4, and for which* 4-Cycle Graph Packing *is FPT solvable in $2^{\mathcal{O}(k)} + n$ time, parameterized by solution size. Then given an $\mathcal{H}$-elimination forest (F, χ) with depth d,* 4-Cycle Graph Packing *is solvable in $2^{\mathcal{O}(d^2 \log d)} \cdot n^{\mathcal{O}(1)}$ time and $2^{\mathcal{O}(d \log d)} \cdot n$ space.*

Proof. We may solve the packing problem over all trees in F separately and sum their results. Then, given an $\mathcal{H}$-elimination tree (T, χ), the dynamic programming table is defined as follows:

Definition 3.2. For a node $t \in V(T)$, integer $0 \leq i \leq |U_t|$, sets $S \subseteq \{t' \mid t' \text{ is an ancestor of } t\}$ and a matching A of the complete graph from $\chi(W)$, where $W \subseteq S$, entry $M[t, i, S, A]$ denotes the integer k such that the graph $G[\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_i}) \cup \chi(S)]$ contains a packing X_1 of k vertex-disjoint C_4's and a packing X_2 of vertex-disjoint P_3's such that:

(i) $V(X_1) \cap V(X_2) = \emptyset$,

(ii) for each edge $\{u, v\}$ in the matching A, there exists a P_3 with endpoints $\{u, v\}$ in packing X_2,

(iii) for each P_3 in X_2, its second vertex is not contained in $\chi(S)$,

(iv) $|X_1|$ is largest,

or $-\infty$ if no such X_1, X_2 exist.

Note that the properties stated here will be referred to in the coming correctness proof. The maximum cardinality of a 4-cycle packing in G can be found by checking entry $M[r, |U_r|, \{r\}, \emptyset]$, where r denotes the root of T.

In addition to assigning ancestors to children of t for the purpose of forming 4-cycles, we will also assign ancestors to children to form instances of P_3, which are intended to form 4-cycles with another P_3. The perfect matching A is used to determine which elements of S should form P_3's together, the remaining elements are used to form C_4's. In the entries of M for which t is an internal node, it is checked for which options of S and A the subproblems are feasible such that the resulting P_3's can be combined and which of the options of S and A result in the largest 4-cycle packing. A visual explanation is depicted in Figure 3.2.

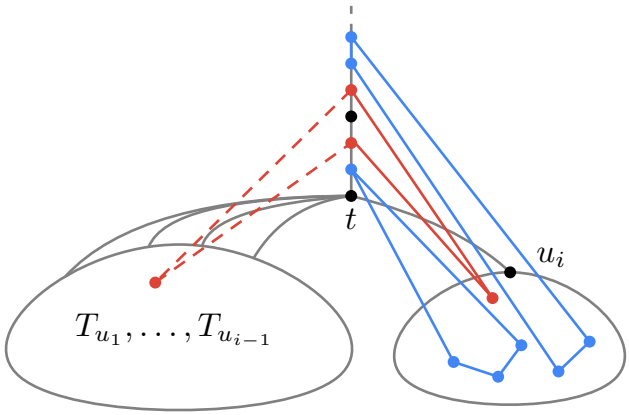


Figure 3.2: An assignment of ancestors for child u_i to form C_4's (shown in blue) and P_3's (shown in red).

The recurrence for entry $M[t, i, S, A]$ will become as follows:

$$M[t,i,S,A] = \begin{cases} \textsc{Pack}_{C_4,P_3}(G[\chi(S)], A) & \text{if } i = 0 \\ \max_{S',A_R,A_S}\{M[t, \quad i-1, \quad (S \setminus V(A_R)) \setminus S', \quad A_S \cup (A \setminus A_R)] \; + \\ \qquad\qquad M[u_i, \quad |U_{u_i}|, \quad V(A_S) \cup V(A_R) \cup S' \cup \{u_i\}, \quad A_S \cup A_R] \; + & \text{if } 1 \leq i \leq |U_t| \\ \qquad\qquad |A_S|\} \end{cases} \tag{3.2}$$

Where the maximizer of the step case considers all combinations of:

- $S' \subseteq S$,
- $A_R \subseteq A$,
- $A_S \subseteq E(K^{S^*})$, where K^{S^*} denotes the complete graph of the set $S^* = (S \setminus V(A_R)) \setminus S'$.

In the step case, it may be that unmatched ancestors s and s' should be matched such that the pair $(\chi(s), \chi(s'))$ will have to form two instances of P_3 with $G[\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_{i-1}})]$ and $G[\chi(T_{u_i})]$. In the recurrence, s and s' are elements of set $V(A_S)$ and matched in A_S.

Correctness When referring to properties in this section, we refer to the properties as provided in Definition 3.2 unless otherwise stated. The correctness proof for the base case of the recurrence directly follows from the definition of M. In order to complete the proof, we will show correctness for the step case.

($M[t, i, S, A] \geq \max_{S',A_R,A_S}\{\ldots\}$) Choose an arbitrary assignment for $S' \subseteq S$, $A_R \subseteq A$ and $A_S \subseteq E(K^{S^*})$. Let X_1, X_2 denote a C_4 packing and P_3 packing respectively of the graph $G[\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_{i-1}}) \cup \chi((S \setminus V(A_R)) \setminus S')]$, upholding the properties (i), (ii), (iii) and (iv) using matching $A_S \cup (A \setminus A_R)$. By the definition of M, we have $|X_1| = M[t, i-1, (S \setminus V(A_R)) \setminus S', A_S \cup (A \setminus A_R)]$.
Let X_1', X_2' denote a C_4 packing and P_3 packing respectively of the graph $G[\chi(T_{u_i}) \cup$

$\chi(V(A_S) \cup V(A_R) \cup S')]$, upholding the properties (i), (ii), (iii) and (iv) using matching $A_S \cup A_R$. We have $|X_1'| = M[u_i, |U_{u_i}|, V(A_S) \cup V(A_R) \cup S', A_S \cup A_R]$.
By property (ii), X_1' and X_2' must contain P_3's for which its endpoints are endpoints of edges in $A_S \cup (A \setminus A_R)$ and $A_S \cup A_R$ respectively. Hence both X_1' and X_2' contain an instance of P_3 for each edge in A_S. By property (iii), the second vertex of a P_3 can not be in $\chi(S)$, implying that for the two instances of P_3 for an edge in A_S, these vertices must be contained in $G[\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_{i-1}})]$ or $G[\chi(T_{u_i})]$ for X_1' and X_2' respectively. Then the combination of the two instances of P_3 implies an instance of C_4. This holds for each edge in A_S. Denote the 4-cycles obtained this way by X_S.

By (i), 4-cycles in X_1, X_1' and X_S are vertex-disjoint, let $Y_1 = X_1 \cup X_1' \cup X_S$. Observe that this is a valid C_4 packing for the graph $G[\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_i}) \cup \chi(S)]$. Let Z_1 and Z_2 denote a C_4 and P_3 packing respectively of the graph $G[\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_i}) \cup \chi(S)]$, upholding the properties (i), (ii), (iii) and (iv) using matching A. Note that $|Z_1| = M[t, i, S, A]$. We have

$$
\begin{aligned}
|Z_1| &\geq |Y_1| && \text{as } Y_1 \text{ is a } \textit{valid } C_4 \text{ packing} \\
&= |X_1| + |X_1'| + |X_S| \\
&= M[t, i-1, S \setminus V(A_R) \setminus S', A_S \cup (A \setminus A_R)] + \\
&\quad M[u_i, |U_{u_i}|, V(A_S) \cup V(A_R) \cup S', A_S \cup A_R] + \\
&\quad |A_S|
\end{aligned}
$$

This holds for any arbitrary assignment of S', A_R and A_S, hence

$$
\begin{aligned}
M[t, i, S, A] \geq \max_{S', A_R, A_S} \{ & M[t, i-1, (S \setminus V(A_R)) \setminus S', A_S \cup (A \setminus A_R)] + \\
& M[u_i, |U_{u_i}|, V(A_S) \cup V(A_R) \cup S', A_S \cup A_R] + \\
& |A_S| \}
\end{aligned}
$$

($M[t, i, S, A] \leq \max_{S', A_R, A_S}\{\ldots\}$) Let Z_1, Z_2 be a maximum C_4 and P_3 packing in $G[\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_i}) \cup \chi(S)]$ respectively, upholding properties (i), (ii), (iii) and (iv) using matching A. We have $|Z_1| = M[t, i, S, A]$. For a 4-cycle $\square \in Z_1$, we distinguish the following cases:

1. $\square$ has a vertex in $\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_{i-1}})$ but not in $\chi(T_{u_i})$. Let $X_1 \subseteq Z_1$ denote the 4-cycles belonging to this case.
2. $\square$ has a vertex in $\chi(T_{u_i})$ but not in $\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_{i-1}})$. Let $X_2 \subseteq Z_1$ denote the 4-cycles belonging to this case.
3. $\square$ has no vertices in either $\chi(T_{u_i})$ or $\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_{i-1}})$. Let $X_3 \subseteq Z_1$ denote the 4-cycles belonging to this case.
4. $\square$ has a vertex in both $\chi(T_{u_i})$ and $\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_{i-1}})$. Let $X_4 \subseteq Z_1$ denote the 4-cycles belonging to this case.

As we have that both the leaves of T are part of graph class $\mathcal{H}$ and no separate subtrees in T shares an edge, it must be that each $\square$ in X_4, has the form $\langle u, s, v, s' \rangle$, where $s, s' \in \chi(S)$

and $u, v \in \chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_i})$. Assume w.l.o.g. that $u \in \chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_{i-1}})$ and $v \in \chi(T_{u_i})$. Construct sets W_1, W_2 and A_4 as follows: For each $\square \in X_4$, add a P_3 of the shape $\{s, u, s'\}$ to W_1, add a P_3 of the shape $\{s, v, s'\}$ to W_2 and add an edge of the shape $\{s, s'\}$ to A_4. Observe that the total number of 4-cycles deconstructed is equal to $|A_4|$.

For $1 \leq j \leq 4$, let $S_j \subseteq S$ be the set of nodes covering all vertices in 4-cycles belonging to X_j, which are also in S.

For a P_3 $\pi \in Z_2$, we distinguish the following cases:

1. π has a vertex in $\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_{i-1}})$ but not in $\chi(T_{u_i})$. Let $Y_1 \subseteq Z_2$ denote the P_3's belonging to this case.
2. π has a vertex in $\chi(T_{u_i})$ but not in $\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_{i-1}})$. Let $Y_2 \subseteq Z_2$ denote the P_3's belonging to this case.

By property (iii), we have that the second vertex of π must be contained in either $\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_{i-1}})$ or $\chi(T_{u_i})$, and therefore the case in which π has no vertices in either of the sets cannot occur. Let $A_1 \subseteq A$ be the set of edges for which its endpoints are covering all endpoints of P_3's belonging to Y_1, and let $A_2 \subseteq A$ be the set of edges for which its endpoints are covering all endpoints of P_3's belonging to Y_2.

Pair $X_1, Y_1 \cup W_1$ is a valid 4-cycle and P_3 packing pair of the graph $G[\chi(T_{u_1}) \cup \ldots \cup \chi(T_{u_{i-1}}) \cup \chi(S \setminus S_2 \setminus S_3)]$ upholding properties (i), (ii) and (iii) using matching $A_4 \cup (A \setminus A_2)$. Hence, $|X_1| \leq M[t, i-1, S \setminus S_2 \setminus S_3, A_4 \cup (A \setminus A_2)]$.
Pair $X_2 \cup X_3, Y_2 \cup W_2$ is a valid 4-cycle and P_3 packing pair of the graph $G[\chi(T_{u_i}) \cup \chi(S_4 \cup S_2 \cup S_3)]$ upholding properties (i), (ii) and (iii) using matching $A_2 \cup A_4$. Hence, $|X_2 \cup X_3| \leq M[u_i, |U_{u_i}|, S_4 \cup S_2 \cup S_3 \cup \{u_i\}, A_2 \cup A_4]$.
We have

$$
\begin{aligned}
M[t, i, S, A] = |Z_1| &= |X_1| + |X_2 \cup X_3| + |X_4| \\
&\leq M[t, i-1, S \setminus S_2 \setminus S_3, A_4 \cup (A \setminus A_2)] + \\
&\quad M[u_i, |U_{u_i}|, S_4 \cup S_2 \cup S_3 \cup \{u_i\}, A_4 \cup A_2] + \\
&\quad |A_4| \\
&\leq \max_{S', A_R, A_S} \{M[t, i-1, S \setminus V(A_R) \setminus S', A_S \cup (A \setminus A_R)] \; + \\
&\quad M[u_i, |U_{u_i}|, V(A_S) \cup V(A_R) \cup S' \cup \{u_i\}, A_S \cup A_R] \; + \\
&\quad |A_S|\}
\end{aligned}
$$

Where,

- S_3 is a candidate for S',
- A_4 is a candidate for A_S, $S_4 = V(A_S)$ follows,
- and A_2 is a candidate for A_R, $S_2 = V(A_R)$ follows.

This concludes the correctness proof. To solve the packing problem on a disconnected graph, it suffices to run the algorithm individually on each connected component and summing their results.

Time and space analysis To analyse the running time of the algorithm, we will consider the cases of the recurrence separately and sum the contribution of each case to the total running time.
For the base case, the number of corresponding entries in M is equal to the *number of nodes in* T $\times$ *number of subsets of* S $\times$ *number of perfect matchings of a complete graph consisting of elements of* S^* $= |V(T)| \cdot 2^{\mathcal{O}(d)} \cdot d!! = 2^{\mathcal{O}(d \log d)} \cdot n$. For each of these entries, $\textsc{Pack}_{C_4,P_3}(G[\chi(S)], A)$ needs to be computed. Similar to the time analysis of triangle packing, the number of 4-cycles cannot exceed the amount of ancestors, and is therefore bounded by d. Using the color coding technique by M. Fellows et al. [14] as mentioned in Theorem 2.1, $\textsc{Pack}_{C_4,P_3}(G[\chi(S)], A)$ can be resolved in $2^{\mathcal{O}(d)} + n^{\mathcal{O}(1)}$ time. This totals to $2^{\mathcal{O}(d \log d)} \cdot n \cdot (2^{\mathcal{O}(d)} + n^{\mathcal{O}(1)}) = 2^{\mathcal{O}(d \log d)} \cdot n^{\mathcal{O}(1)}$ time overall for the base case.

To determine the time needed to evaluate the recurrence for the step case, we have to consider all combinations for S', A_R and A_S: For S', we have $2^{|S|}$ options. Similarly, the possible options for A_R adds a factor $2^{|A|}$. Lastly we must also consider all options for A_S, which may add a factor of at most $|S|!! \leq 2^{\mathcal{O}(|S| \log |S|)}$. The overall time to evaluate a single step case becomes at most $2^{\mathcal{O}(|S| \log |S| + |A|)} = 2^{\mathcal{O}(|S| \log |S|)}$, as the terms inside the minimizer can be resolved in constant time. If we consider the counting the cells based on the parameters, we arrive at a total time for the step case of

$$\sum_{t \in V(T)} \sum_{1 \leq i \leq |U_t|} \sum_{S \subseteq \mathcal{A}(t)} \sum_{A \subseteq K^S} 2^{\mathcal{O}(|S| \log |S|)}$$

As we have seen in the analysis of Section 3.2, the choice of i an t contribute at most a factor n to the total running time. Moreover, the choices of S and A add at most a factor 2^d or $2^{d \log d}$ respectively. Both of these factors fall under the big-Oh notation of $2^{\mathcal{O}(d^2 \log d)} \cdot n$. Since we have that $|S| \leq d$, the total time for the step case becomes $2^{\mathcal{O}(d^2 \log d)} \cdot n^{\mathcal{O}(1)} \cdot 2^{\mathcal{O}(|S| \log |S|)} \cdot n = 2^{\mathcal{O}(d^2 \log d)} \cdot n^{\mathcal{O}(1)}$.

The dominating factor of the total running time becomes that of the step case, which results in the desired running time.

Observation. Note that in order to solve Triangle Graph Packing from scratch, we would need to find an $\mathcal{H}$-elimination forest first. We may consider using the approach presented in the book by Jansen et al. [Table 1][16]. To obtain an $\mathcal{H}$-elimination forest of depth $d = \mathcal{O}(k \log k)$ of graph G satisfying $\mathbf{ed}_{\mathcal{H}}(G) \leq k$ would take $2^{\mathcal{O}(k)}$ time. We have that $d = \mathcal{O}(k \log k)$, which would imply that the term $2^{\mathcal{O}(k)}$ has an insignificant contribution to the remainder of the algorithm. Therefore the total running time remains the same.

The required space to run the algorithm corresponds to the number of cells in table M added with the space required to resolve $\textsc{Pack}_{C_4,P_3}(G[\chi(S)], A)$. The number of cells is at most $|V(T)| \times |V(T)| \times 2^d \times d!! = 2^{\mathcal{O}(d \log d)} \cdot n^2$. Again, as we know that each node t in the tree is considered once, we can reduce this to $2^{\mathcal{O}(d \log d)} \cdot n$. The space required to evaluate $\textsc{Pack}_{C_4,P_3}(G[\chi(S)], A)$ is $2^{\mathcal{O}(d+|A|)} = 2^{\mathcal{O}(d)}$. Combined we require in total $2^{\mathcal{O}(d \log d)} \cdot n$ space. ■

Chapter 4

Solving the Steiner Tree Problem by Multi-Way Cut

The Steiner Tree problem asks to construct a tree of minimum weight within some graph G, connecting a set of terminals $K \subseteq V(G)$. A multi-way cut $S \subseteq V(G)$ is a set of vertices, which, in the context of Steiner Trees, after removal of G leaves connected components containing at most one terminal. Note that it may be that $S \cap K$ is non-empty. A visual representation of such a cut can be seen in Figure 4.1. In the image, the squares denote the terminals, set S denotes a multi-way cut and $C = \{C_1, \ldots, C_m\}$ denotes the set of connected components obtained after removing S from G. The red tree shows a valid Steiner Tree.

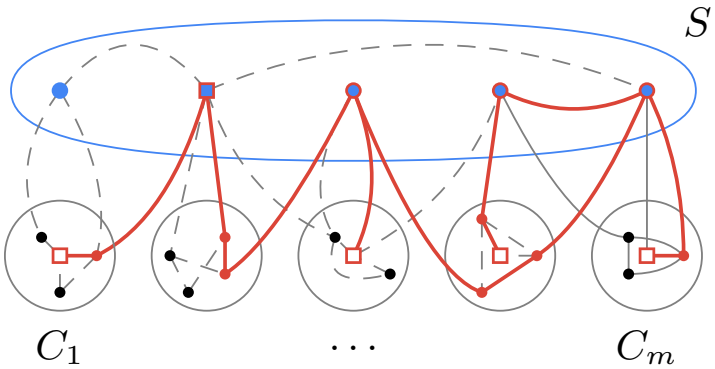


Figure 4.1: Visualisation of a multi-way cut on G, with a feasible Steiner Tree solution in red.

Trivially, we have that for any Steiner Tree problem with $|K| \geq 2$, any valid solution contains at least one vertex from the multi-way cut. In this Section we will introduce a fixed-parameter tractable approach which solves the Steiner Tree problem parameterized by the size of a given multi-way cut S. Lemma 4.1 will help acquire intuition for this approach.

Lemma 4.1. *Given a multi-way cut S of G, a set of terminals $K \subseteq V(G)$ where $|K| \geq 2$ and set of connected components $C = \{C_1, \dots C_m\}$ of the graph $G - S$, each optimal Steiner Tree can be written as the disjoint union of trees of the form:*

(*i*) *A tree with minimum number of edges in $G[D \cup C_i]$, connecting the vertices in D, where $D \subseteq N_G(C_i) \cup (C_i \cap K)$ and $1 \leq i \leq m$.*

(*ii*) *A tree with minimum number of edges in $G[S']$, connecting the vertices in S', where $S' \subseteq S$.*

Proof. Suppose that with a provided multi-way cut S, there exists an optimal Steiner Tree T that cannot be written as the disjoint union of trees of the provided form as described in Lemma 4.1.

Consider partitioning the edges of T into three categories:

- Let E_1 denote the set of edges of T which have both endpoints in S.
- Let E_2 denote the set of edges of T which have an endpoint in S and an endpoint in some C_i, where $1 \leq i \leq m$.
- Let E_3 denote the set of edges of T which have both endpoints in some C_i, where $1 \leq i \leq m$.

We will isolate the edges of T as follows: For each external component C_i, let F_i denote the set of trees induced by the edges in E_2 which have an endpoint in C_i, combined with the edges in E_3 which have both endpoints in C_i. We argue that the trees in these sets are of the first proposed form as stated in Lemma 4.1.

For tree $T_i \in F_i$ where $1 \leq i \leq m$, let D denote the set of vertices such that $D = V(T_i) \cap (N_G(C_i) \cup (C_i \cap K))$. Observe that if $|D| < 2$, we have that removing the edges of T_i from T does not disconnect T, and would reduce the number of edges of T. As we have assumed optimality of tree T, it must be that $|D| \geq 2$. In order to show that T_i is a valid candidate for the first proposed form, we need to show that it has the minimum number of edges in $G[D \cup C_i]$ which connects D. Suppose that there exists a tree T_i' which connects D in $G[D \cup C_i]$ using fewer edges. Then, we may substitute the edges of T_i in T with the edges of T_i', resulting in T having fewer edges overall. As we have assumed optimality of T, we have arrived at a contradiction, implying there cannot be such a T_i' and T_i has the minimum number of edges within its context. Since this applies to every tree T_i in every F_i for $1 \leq i \leq m$, we can say that each of these trees are of the first proposed form of the lemma stated above.

Let F denote the forest induced by the edges in E_1. We argue that the trees in F are of the second proposed form as stated in Lemma 4.1. Consider tree $T_S \in F$, let $S' = V(T_S)$. In order to show that T_S is a valid candidate for the second proposed form, we need to show that it has the minimum number of edges in $G[S']$, connecting S'. We again invoke a similar argument to the one stated above; there cannot be a tree T_S' connecting S' within $G[S']$ with fewer edges, as otherwise its substitution with T_S would result in T having fewer edges, clashing with our assumption of T being optimal. This is the case for every

tree in F, hence we can say that each tree in F are of the second proposed form of the lemma stated above.

Since E_1, E_2 and E_3 cover all edges in T, and all edges are considered in the isolation, we may conclude that T can be written as the disjoint union of the trees in $F_1, \ldots, F_m$ and the trees in F, which we have seen are proper candidates for the first and second proposed forms of Lemma 4.1 respectively. ■

The approach in this section relies on an existing algorithm solving the Steiner Tree problem in a time parameterized by the number of terminals. Theorem 2.3 introduces this algorithm and can be found in Section 2.2.

Theorem 4.2. *Given graph G, and set of terminals $K \subseteq V(G)$ and multi-way cut $S \subseteq V(G)$,* Steiner Tree *can be solved in $2^{\mathcal{O}(|S| \log |S|)} \cdot n^{\mathcal{O}(1)}$ time and space.*

Proof. Let $C = \{C_1, \ldots, C_m\}$ denote the set of connected components of $G - S$. Observe that due to the construction of C, we have that $N_G(C_i) \subseteq S$ for all $0 \leq i \leq m$. We define a dynamic programming table M as follows:

Definition 4.1. Let $0 \leq i \leq m$, $A = \{B_1, \ldots, B_\ell\}$ be a partition over some $S' \subseteq S$ and $0 \leq j \leq |N_G(C_i)|$. Then define $M[i, A, j]$ as the minimum total weight of a subgraph F in $G[S \cup C_1 \cup \ldots \cup C_i]$, where:

(i) for each set $B \in A$, there exists a connected component in F containing the elements of B,

(ii) each connected component in F contains a vertex from S',

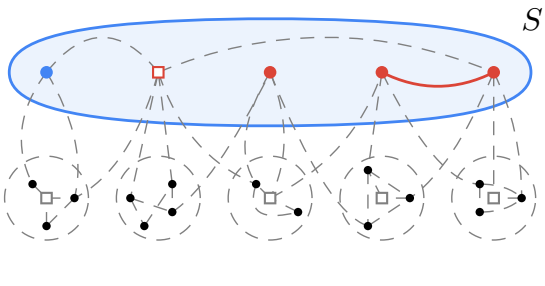
(iii) for each terminal $v \in (C_1 \cup \ldots \cup C_i) \cap K$, some component in F contains v,

(iv) $F \hat{\cap} G[N_G[C_i]]$ consists of at most j connected components,

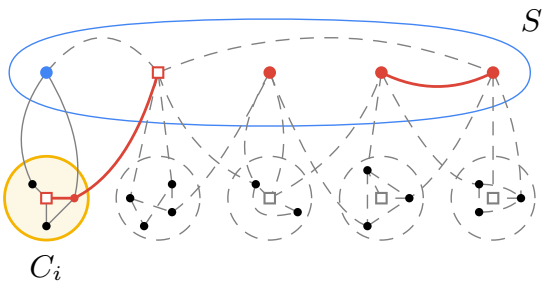
or ∞ if no such F exists.

The optimal solution can be found by finding the smallest entry of the entries $M[m, \{S'\}, |N_G(C_m)|]$ where $\emptyset \neq S' \subseteq S$ and $S' \cap K = S \cap K$. We then consider finding subgraph F where: (i, ii) F contains all elements of S', (iii) all terminals in $C_1, \ldots, C_m$ are connected by F and (iv) $F \hat{\cap} G[N_G[C_m]]$ may have at most $|N_G(C_m)|$ components. Here, $|N_G(C_m)|$ is an upperbound for the number of connected components in $F \hat{\cap} G[N_G[C_m]]$; assuming that each connected component connects a minimum of two vertices from K and $|N_G(C_m)|$, having more than this upperbound will clash with the optimality of F. A more detailed argument is given in the correctness proof. The choice for S' ensures we include all vertices in $S \cap K$, while trying all subsets of $S \setminus K$. Observe the operator $\hat{\cap}$ in property (iv); this operator is similar to the graph intersection in that its result has the same edge set, but it does not include vertices which are not connected to an edge. Its formal definition can be found in Section 2.2.

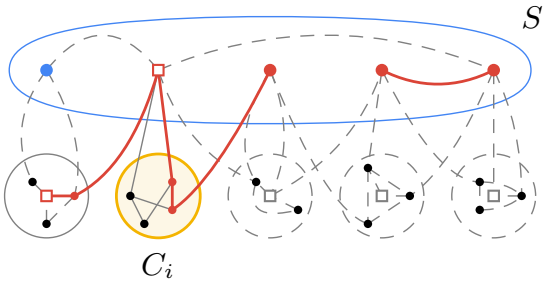
The intuition for the approach is as follows: for an entry $M[i, A, j]$ we consider C_i and finding up to j subtrees in $G[N_G[C_i]]$, connecting disjoint subsets of $D = (C_i \cap K) \cup N_G(C_i)$. We only have to connect elements of $D \cap S$ that are in the same set of A; partition A shows us which elements of S will have to be connected. Given the choice, A is updated accordingly; if we manage to connect vertex set D, we split D in A. The algorithm uses a bottom-up approach, meaning it will solve the smaller subproblems first and base the solutions of larger problems on solutions to smaller subproblems. Figure 4.3 visualises a possible use case.



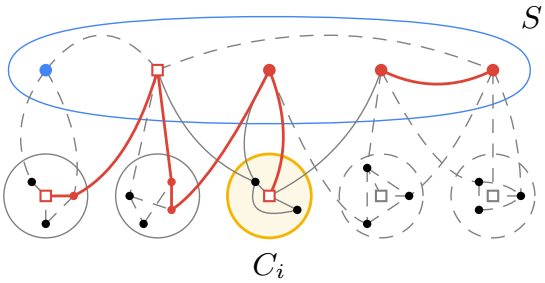
(a) Possible solution to $M[0, \{\{s_2\}, \{s_3\}, \{s_4, s_5\}\}, 0]$, we have $S' = \{s_2, s_3, s_4, s_5\}$.



(b) Possible solution to $M[1, \{\{s_2\}, \{s_3\}, \{s_4, s_5\}\}, 1]$, using solution to subprocess (a).

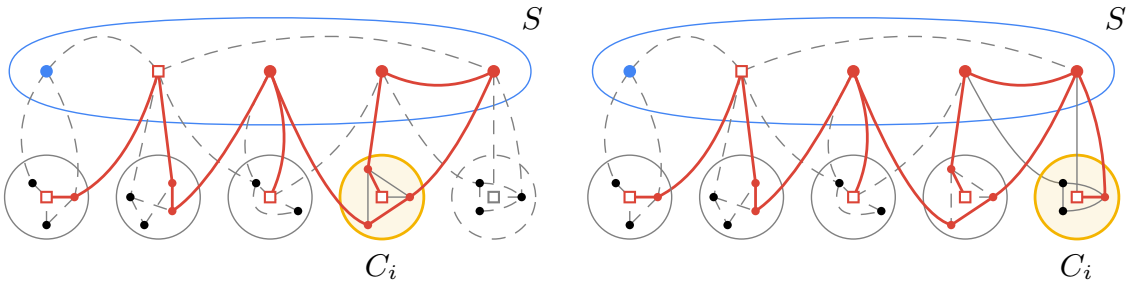


(c) Possible solution to $M[2, \{\{s_2, s_3\}, \{s_4, s_5\}\}, 1]$, using solution to subprocess (b).



(d) Possible solution to $M[3, \{\{s_2, s_3\}, \{s_4, s_5\}\}, 1]$, using solution to subprocess (c).

Figure continues on the next page.



(a) Possible solution to $M[4, \{\{s_2, s_3, s_4, s_5\}\}, 2]$, using solution to subproblem (d).

(b) Possible solution to $M[5, \{\{s_2, s_3, s_4, s_5\}\}, 1]$, using solution to subproblem (e).

Figure 4.3: Bottom-up approach visualised. Top left shows a solution to the smallest subproblem. During each subproblem, a multitude of smaller subproblems are considered on which to build, this figure shows a sequence of scenarios in which the depicted solutions are build on top of each other.

To find the optimal trees in the connected components of C and S', we can use the algorithm stated in Theorem 2.3 and will be denoted by the function $\textsc{Steiner}(G, K)$. Here, G denotes the graph and $K \subseteq V(G)$ denotes the set of terminals.

Since a set that has minimum weight while containing a prescribed set of terminals F will always be acyclic, we can consider the process of finding a minimum Steiner tree as finding a minimum-weight connected subgraph containing the terminals T. This is the viewpoint we take in the rest of the proof.

We define the recurrence for an entry $M[i, A, j]$ as follows:

$$M[i, A, j] = \begin{cases} \sum_{B \in A} \textsc{Steiner}(G[B], B) & \text{if } i = 0 \\ M[i-1, A, |N_G(C_{i-1})|] & \text{if } j = 0 \text{ and } C_i \cap K = \emptyset \\ \infty & \text{if } j = 0 \text{ and } C_i \cap K \neq \emptyset \\ \min_{D^*, A^*} \{\textsc{Steiner}(G[N_G[C_i]], D^*) + M[i-1, A^*, |N_G(C_{i-1})|]\} & \text{if } j = 1 \text{ and } C_i \cap K \neq \emptyset \\ \min_{D', A'} \{\textsc{Steiner}(G[N_G[C_i]], D') + M[i, A', j-1], M[i, A, j-1]\} & \text{otherwise} \end{cases} \tag{4.1}$$

Where the minimizer of the fourth base case considers all combinations of:

- $D^* = (C_i \cap K) \cup D'$, where $\emptyset \neq D' \subseteq N_G(C_i)$,
- A^* denotes a possible partition of $S' = \bigcup_{B \in A} B$ such that $\textsc{Merger}(A^*, D^*) = A$.

And the minimizer of the step case considers all combinations of:

- $\emptyset \neq D' \subseteq N_G(C_i)$,
- A' denotes a possible partition of $S' = \bigcup_{B \in A} B$ such that $\textsc{Merger}(A', D') = A$.

Here, $\textsc{Merger}(A, D)$ constructs a new partition such that all sets containing some element of D in the original partition are combined. A more detailed definition is provided in the Section 2.2.

Correctness We will prove the correctness of the recurrence by considering each of the cases independently, often proving the equality by proving both inequalities. When referring to properties in this section, we refer to the properties as provided in Definition 4.1 unless otherwise stated.

First base case The recurrence claims that $M[i, A, j] = \sum_{B \in A} \textsc{Steiner}(G[B], B)$ if $i = 0$. We will prove the equality by proving inequalities in both directions.

($M[i, A, j] \leq \sum_{B \in A} \textsc{Steiner}(G[B], B)$ Let F denote a subgraph witnessing the value of $\sum_{B \in A} \textsc{Steiner}(G[B], B)$. We will show that this is a valid candidate for the term $M[i, A, j]$ by showing F conforms to properties (i), (ii), (iii) and (iv) as defined in the definition of M.

(i) We need to show that for each set $B \in A$, there exists a connected component in F containing the elements of B. The $\textsc{Steiner}(G[B], B)$ function computes the minimum total weight of a tree in $G[B]$ connecting the vertices in B. Hence a subgraph $T \in F$ witnessing the value of this function is a tree with minimum total weight. This is the case for every $B \in A$, hence the property holds.

(ii) We need to show that each connected component in F contains a vertex from S'. We have by construction that $\bigcup_{B \in A} B = S'$. Since each element of A is connected by some subgraph in F, this property holds.

(iii) We have to show that for each terminal $v \in (C_i \cup \ldots \cup C)i) \cap K$,m some component in F contains v. Since $i = 0$, $C_i \cup \ldots \cup C_i = \emptyset$, hence the property holds.

(iv) We must show that $F \hat{\cap} G[N_G[C_i]]$ consists of at most j components. Since we have that $F \hat{\cap} G[N_G[C_i]]$ is an empty graph, the property holds for any value of j.

We have found that F satisfies conditions (i), (ii), (iii) and (iv) and is therefore a valid candidate for $M[i, A, j]$. Let $\mathbf{w}$ denote the function evaluating to the total weight of a subgraph. We can conclude that $\mathbf{w}(F) = \sum_{B \in A} \textsc{Steiner}(G[B], B) \geq M[i, A, j]$. This approach of constructing a subgraph based on one side of the inequality and showing that it (or an alteration) is a valid candidate for the term on the opposite side of the inequality, is a recurring strategy in this proof.

($M[i, A, j] \geq \sum_{B \in A} \textsc{Steiner}(G[B], B)$ Let F denote a subgraph in $G[S']$ with minimum total weight conforming to properties (i), (ii), (iii) and (iv) as defined in the definition of M, given partition A over S' and j. We can say that F witnesses the value of $M[i, A, j]$. We need to argue that F is a valid candidate for $\sum_{B \in A} \textsc{Steiner}(G[B], B)$. In order to do so, we can show that F contains a tree for each $B \in A$ connecting the vertices of B. Consider some $B \in A$. We have that F conforms to property (i), implying that there

exists a component in F containing the elements of B. Let H be the connected component containing the elements of B. Since we have that the weight of F is minimized, this implies that the weight of H is minimized. We have established earlier that a minimum weighted connect subgraph is always acyclic, and therefore a tree. We have found that H is a valid candidate for $\textsc{Steiner}(G[B], B)$. This is the case for all $B \in A$, hence F is a valid candidate for the term $\sum_{B \in A} \textsc{Steiner}(G[B], B)$. We can conclude that $\mathbf{w}(F) = M[i, A, j] \geq \sum_{B \in A} \textsc{Steiner}(G[B], B)$.

Second base case The recurrence claims that $M[i, A, j] = M[i-1, A, |N_G(C_{i-1})|]$ when $j = 0$ and $C_i \cap K = \emptyset$. We will prove the equality by proving inequalities in both directions.

($M[i, A, j] \leq M[i-1, A, |N_G(C_{i-1})|]$) Let F be a subgraph in $G[S \cup C_1 \cup \ldots \cup C_{i-1}]$ with minimum total weight conforming to properties (i), (ii), (iii) and (iv) as defined in the definition of M, given partition A over S' and $j = |N_G(C_{i-1})|$. We can say that F witnesses the value of $M[i-1, A, |N_G(C_{i-1})|]$. In order to prove the inequality, we would need to argue why F is a valid candidate for $M[i, A, j]$, while perhaps not optimal. We can do so by showing that F conforms to properties (i), (ii), (iii) and (iv) as defined by the entry $M[i, A, j]$. For each property, we will use that F conforms to the property in the case of entry $M[i-1, A, |N_G(C_{i-1})|]$ and derive that it holds for entry $M[i, A, j]$.

(i) We need to show that for each set $B \in A$, there exists a connected component in F containing the elements of B. We have that (i) does not depend on parameters i and j; the candidate F remains unchanged, that is, no edges are added or removed from the subgraph depending on which side of the inequality we focus. Partition A has not changed, and since this property only depends on A and F, we know that this property still holds.

(ii) We have to show that each connected component in F contains a vertex from S'. This property has a similar reasoning to that of property (i); since the subgraph F remains the same and this property does not depend on i or j, we can state that this property still holds.

(iii) We need to show that for each terminal $v \in (C_1 \cup \ldots \cup C_i) \cap K$, some component in F contains v. We know that by construction, for each $v \in (C_1 \cup \ldots \cup C_{i-1}) \cap K$, some component in F contains v. Moreover, we know that component C_i does not contain a terminal, as this is a requirement for the case. As such, the property still holds.

(iv) We must show that $F \hat{\cap} G[N_G[C_i]]$ consists of at most $j = 0$ components. By construction, we have that F is contained in $G[S \cup C_1 \cup \ldots \cup C_i]$, which implies that $F \hat{\cap} G[N_G[C_i]]$ consists of no components. The property holds.

We have found that F satisfies conditions (i), (ii), (iii) and (iv) and is therefore a valid candidate for $M[i, A, j]$. We can conclude that $\mathbf{w}(F) = M[i-1, A, |N_G(C_{i-1})|] \geq M[i, A, j]$.

($M[i, A, j] \geq M[i-1, A, |N_G(C_{i-1})|]$) Let F be a subgraph in $G[S \cup C_1, \ldots, C_i]$ with minimum total weight conforming to properties (i), (ii), (iii) and (iv) as defined in the definition of M, given partition A over S and $j = 0$. We argue that F is a valid candidate for $M[i-1, A, |N_G(C_{i-1})|]$ by showing that F conforms to properties (i), (ii), (iii) and (iv):

(i),(ii) We have to show that (i): for each set $B \in A$, there exists a connected component in F containing the elements of B and (ii): each connected component in F contains a vertex from S'. These properties do not depend on values of i or j. Since F and A remain unchanged, the properties still hold.

(iii) We need to show that for each terminal $v \in (C_1 \cup \ldots \cup C_{i-1}) \cap K$, some component in F contains v. We know that by construction each terminal $v \in (C_1 \cup \ldots \cup C_i) \cap K$ is contained in some component of F, hence the property holds.

(iv) We need to show that $F \hat{\cap} G[N_G[C_{i-1}]]$ consists of at most $|N_G(C_{i-1})|$ components. Since we have that F upholds to property (ii), we know that each component in $F \hat{\cap} G[N_G[C_{i-1}]]$ contains a vertex from $N_G(C_i)$. It follows that there can be at most $|N_G(C_{i-1})|$ connected components in $F \hat{\cap} G[N_G[C_{i-1}]]$; the property holds.

Hence we have found that F is a valid candidate for $M[i-1, A, |N_G(C_{i-1})|]$. We have $\mathbf{w}(F) = M[i, A, j] \geq M[i-1, A, |N_G(C_{i-1})|]$.

Third base case The recurrence claims that $M[i, A, j] = \infty$ if $j = 0$ and $C_i \cap K \neq \emptyset$. Observe that by the definition of M, we ought to find the total weight of a subgraph F in $G[S \cup C_1 \cup \ldots \cup C_i]$ which by property (iii) should contain the terminal $v \in C_i \cap K$ while by property (iv) must consist of at most $j = 0$ connected components in $F \hat{\cap} G[N_G[C_i]]$. This cannot be done, hence the value evaluates to ∞ (as defined by M).

Fourth base case We have that $M[i, A, j] = \min_{D^*, A^*}\{\textsc{Steiner}(G[N_G[C_i]], D^*) + M[i-1, A^*, |N_G(C_{i-1})|]\}$ if $j = 1$ and $C_i \cap K \neq \emptyset$. We will prove the equality by proving inequalities in both directions.

($M[i, A, j] \leq \min_{D^*, A^*}\{\ldots\}$) Choose an arbitrary $D' \subseteq N_G(C_i)$, $D^* = (C_i \cap K) \cup D'$ and A^* such that $\textsc{Merger}(A^*, D^*) = A$. Let T be the connected subgraph witnessing the value of $\textsc{Steiner}(G[N_G[C_i]], D^*)$. Let F' be a subgraph in $G[S \cup C_1 \cup \ldots \cup C_{i-1}]$ with minimum total weight conforming to properties (i), (ii), (iii) and (iv) provided with partition A^* over S and integer $j = |N_G(C_{i-1})|$ as defined in the definition of M. Let $F = F' \cup T$.

We will show that F is a valid candidate subgraph for the term $M[i, A, j]$ by showing that F conforms to properties (i), (ii), (iii) and (iv) as stated by the definition of M:

(i) We have that for each set $B^* \in A^*$, there exists a connected component in F' containing the elements of B^*. Consider a pair v, w that is contained in a common set B^* of the partition A^*, implying that a solution to the subproblem $M[i-1, A^*, |N_G(C_{i-1})|]$

is required to have v and w in the same connected component of F'. For v, w, we can distinguish the following two cases:

1. Suppose that $v, w \in B^*$ where $B^* \in A^*$.
   Property (i) implies that F' contains a vw path. We have that F contains all edges contained in F'. Therefore if a vw path exists in F' it also exists in F. Hence the property holds for F.
2. Suppose that $v \in B_1^*$ and $w \in B_2^*$, where $B_1^*, B_2^* \in A^*$ and $B_1^* \neq B_2^*$.
   Recall that $\textsc{Merger}(A^*, D^*) := (A^* \setminus A') \cup \{\bigcup_{B' \in A'} B'\}$ where $A' \subseteq A$ contains all sets from A which contain at least one element from D^*. This implies that there exists some $a \in D^* \cap B_1^*$. Likewise, there exists some $b \in D^* \cap B_2^*$. It follows that there exists va and bw paths in F', denote these paths as P_1 and P_2 respectively. By the definition of the $\textsc{Steiner}$ function, the connected subgraph T connects the elements from D^*, hence there exists an ab path in T, denote this path as P_3.

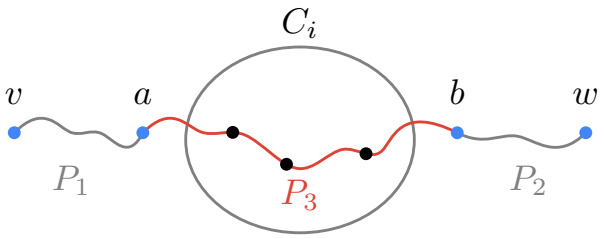


Figure 4.4: vw path decomposed into three known subpaths.

   F contains all edges from both F' and T, hence it contains paths P_1, P_2 and P_3, Figure 4.4 visualises these paths. P_1, P_2 and P_3 can be composed into one vw path implying that the property holds.

These cases cover all possibilities for any vw pair such that $v, w \in B$ where $B \in A$. The property holds in each of these cases, hence the property holds.

(ii) We have that each connected component in F' contains a vertex from S'. Moreover, T connects the elements from D^*. By construction, we have that D^* contains at least one element from $N_G(C_i)$, since $\emptyset \neq D' \subseteq N_G(C_i)$. We know that $N_G(C_i)$ is fully contained in S'. This implies that T contains a vertex from S'. As such we have that each connected component in F contains a vertex from S'.

(iii) We have that for each terminal v in $(C_1 \cup \ldots \cup C_{i-1}) \cap K$, some component in F' contains v. Since F is a supergraph of F', it must be that F also contains the terminals in $(C_1 \cup \ldots \cup C_{i-1}) \cap K$. Moreover, we have that T connects the elements of D^*, where $C_i \cap K \subseteq D^*$. Therefore F contains the terminals in $(C_1 \cup \ldots \cup C_i) \cap K$.

(iv) We need to show that $F \hat{\cap} G[N_G[C_i]]$ contains at most one component. We know by property (i) of F' that for each $B^* \in A^*$, there exists a connected component in F' containing the elements of B^*. By the $\textsc{Merger}$ function, we know that each element $B^* \in A^*$ must contain at least one element from D^*, implying that each

component in F' contains an vertex from D^*. T by definition connects the elements from D^* using a single subgraph, it follows that $F \hat{\cap} G[N_G[C_i]]$ contains at most one component.

Hence we have found that F satisfies all requirements for subproblem $M[i, A, j]$ and has cost $\mathbf{w}(F) = \textsc{Steiner}(G[N_G[C_i], D^*] + M[i-1, A^*, |N_G(C_i)|])$. As this argument applies to every arbitrary choice of D^* and A^*, we have that $M[i, A, j]$ is at most the minimum of $\textsc{Steiner}(G[N_G[C_i], D^*] + M[i-1, A^*, |N_G(C_i)|])$ for all choices of D^* and A^*.

($M[i, A, j] \geq \min_{D^*, A^*}\{\ldots\}$) Let F be a subgraph in $G[S \cup C_1 \cup \ldots \cup C_i]$ with minimum total weight conforming to properties (i), (ii), (iii) and (iv) provided with partition A over S' and integer j as defined in the definition of M. Since we have that $C_i \cap K \neq \emptyset$, it must be that $F \hat{\cap} G[N_G[C_i]]$ contains a single connected component connecting the terminal in C_i. Let H be this connected component. Let F' be the graph with vertex set $V(F) \setminus (V(H) \setminus N_G(C_i))$ and edge set $E(F) \setminus E(H)$. Figure 4.5 depicts a possible configuration for this case.

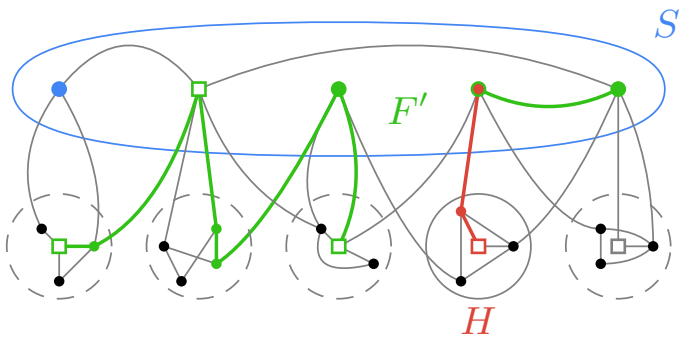


Figure 4.5: Decomposition of F into F' (shown in green) and H (shown in red).

Let $D^* = V(H) \cap (N_G(C_i) \cup (C_i \cap K))$ and pick A^* such that $\textsc{Merger}(A^*, D^*) = A$. We have that $\mathbf{w}(H) \geq \textsc{Steiner}(G[N_G[C_i]], D^*)$. We will show that F' is a valid candidate subgraph for the term $M[i-1, A^*, |N_G(C_{i-1})|]$ as stated by the definition of M, by showing F' conforms to properties (i), (ii), (iii) and (iv):

(i) We must show that for each set $B^* \in A^*$, there exists a connected component in F' containing the elements of B^*. This follows directly from the choice of A^* as defined above.

(ii) We have to show that each connected component in F' contains a vertex from S'. For a connected component I in F' we may find two options:

1. $I \cap H = \emptyset$. Then since F conformed to property (ii), it must be that I contains a vertex from S'.
2. $I \cap H \neq \emptyset$. We have that H connects vertices in D^*, and therefore I must contain a vertex from D^*. By construction we have that D^* consists of elements from

S', since the open neighborhood of any $C_i \in C$ may only consist of vertices from the multi-way cut; I contains a vertex from S'.

Thus the property holds for F'.

(iii) We have that every terminal $v \in (C_1 \cup \ldots \cup C_i) \cap K$, some component in F contains v. We need to show that for each terminal $v \in (C_1 \cup \ldots \cup C_{i-1}) \cap K$, some component in F' contains v. Only vertices from C_i were removed from F to construct F', hence since the property holds for F it must hold for F'.

(iv) We must show that $F' \hat{\cap} G[N_G[C_{i-1}]]$ consists of at most $|N_G(C_{i-1})|$ components. We have shown before (second base case) that $|N_G(C_{i-1})|$ is an upperbound for the number of connected components in $F' \cap G[N_G[C_{i-s1}]]$. The property holds.

We have that $F \hat{\cap} G[N_G[C_i]]$ consists of at most j components. One of the components in $F \hat{\cap} G[N_G[C_i]]$ (H) was removed, therefore the property holds for F'.

Hence we have found that F' satisfies all requirements for subproblem $M[i-1, A^*, |N_G(C_{i-1})|]$. We have

$$\begin{aligned} M[i, A, j] &= \mathbf{w}(F) \\ &= \mathbf{w}(H) + \mathbf{w}(F) \\ &\geq \textsc{Steiner}(G[N_G[C_i]], D^*) + M[i-1, A^*, |N_G(C_{i-1})|] \\ &\geq \min_{D^*, A^*} \{\textsc{Steiner}(G[N_G[C_i]], D^*) + M[i-1, A^*, |N_G(C_{i-1})|]\}) \end{aligned}$$

Step case The recurrence claims that $M[i, A, j] = \min_{D', A'}\{\textsc{Steiner}(G[N_G[C_i]], D') + M[i, A', j-1], M[i, A, j-1]\}$ when no other cases apply. We will prove the equality by proving inequalities in both directions.

$(M[i, A, j] \leq \min_{D', A'}\{\ldots\})$ First, we will argue that $M[i, A, j] \leq M[i, A, j-1]$. Let F be the subgraph witnessing the value of $M[i, A, j-1]$, conforming to properties (i), (ii), (iii) and (iv) as defined in the definition of M, when provided with the relevant subproblem information.

We will show that F is a valid candidate subgraph for the term $M[i, A, j]$ by showing that is upholds to properties (i), (ii), (iii) and (iv) as stated in the definition of M. Observe that between the two terms of the equation, only the parameter j has changed. Since properties (i), (ii) and (iii) do not rely on parameter j, we can say these properties directly follow from the construction of F. We are left to show property (iv): that $F \hat{\cap} G[N_G[C_i]]$ consists of at most j components. We have by construction that $F \hat{\cap} G[N_G[C_i]]$ consists of at most $j-1$ components. Property (iv) follows directly.

Hence, we have found that F' is also a valid solution for $M[i, A, j]$, and therefore $M[i, A, j-1] = \mathbf{w}(F') \leq M[i, A, j]$.

Now we will show that $M[i, A, j] \leq \min_{D', A'}\{\textsc{Steiner}(N_G[C_i], D') + M[i, A', j-1]\}$.

Choose an arbitrary D' where $\emptyset \neq D' \subseteq N_G(C_i)$ and A' such that $\textsc{Merger}(A', D') = A$. Let T be the subgraph witnessing the value of $\textsc{Steiner}(N_G[C_i], D')$. Let F' denote the

subgraph witnessing the value of $M[i, A', j-1]$, conforming to properties (i), (ii), (iii) and (iv) as defined in the definition of M, given partition A' over S and $j-1$. Let $F = F' \cup T$. We will show that F is a valid candidate subgraph for the term $M[i, A, j]$ as stated by the definition of M, by showing that F conforms to properties (i), (ii), (iii) and (iv):

(i) We have that for each $B' \in A'$, there exists a connected component in F' containing the elements of B'. Consider a pair v, w that is contained in a common set B' of the partition A', implying that a solution to the subproblem $M[i, A', j-1]$ is required to have v and w in the same connected component of F'.
We can use the same case distinction argument as seen in the fourth base case. If $v, w \in B'$ where $B' \in A'$ we have that an vw path exists in F', and therefore also in F. If $v \in B'_1$ and $w \in B'_2$ where $B'_1, B'_2 \in A'$ and $B'_1 \neq B'_2$, we can show that there exists a composed path in F as depicted in Figure 4.4.

These cases cover all possibilities for any vw pair such that $v, w \in B$ where $B \in A$. The property holds in each of these cases, hence the property holds.

(ii) We have that each connected component in F' contains a vertex from S. Moreover, T connects the elements from D'. We have $\emptyset \neq D' \subseteq N_G(C_i)$ and $N_G(C_i)$ is fully contained in S'. This implies that T contains a vertex from S'. It follows that each connected component in F contains a vertex from S.

(iii) We have that for each terminal v in $(C_1 \cup \ldots \cup C_i) \cap K$, some component in F' contains v. Since F is a supergraph of F', it must be that F also contains the terminals in $(C_1 \cup \ldots \cup C_i) \cap K$.

(iv) We have that $F' \cap G[N_G[C_i]]$ consists of at most $j-1$ components. Adding T to F' adds at most one component. Hence we have that $F \hat{\cap} G[N_G[C_i]]$ consists of at most j components.

We have found that F satisfies all requirements for subproblem $M[i, A, j]$. We have

$$\begin{aligned} M[i, A, j] &= \mathbf{w}(F) \\ &\leq \mathbf{w}(F') + \mathbf{w}(T) \\ &= \textsc{Steiner}(N_G[C_i], D') + M[i, A', j-1] \end{aligned}$$

Thus

$$M[i, j, A] \leq \min_{D', A'} \{\textsc{Steiner}(N_G[C_i], D') + M[i, A', j-1]\}$$

($M[i, A, j] \geq \min_{D', A'}\{\ldots\}$) Let F be a subgraph in $G[S \cup C_1 \cup \ldots \cup C_i]$ with minimum total weight conforming to properties (i), (ii), (iii) and (iv) provided with partition A over S and integer $1 \leq j \leq |N_G(C_i)|$ as defined in the definition of M.
For F, we may distinguish two cases:

1. $F \hat{\cap} G[N_G[C_i]]$ contains a connected component not containing a terminal.

   Let H be such a connected component. Let F' be the graph with vertex set $V(F) \setminus (V(H) \setminus N_G(C_i))$ and edge set $E(F) \setminus E(H)$. Figure 4.6 depicts a possible configuration for this case.

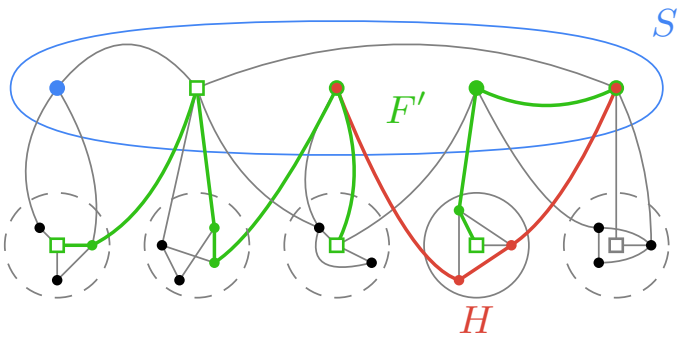


Figure 4.6: Decomposition of F into F' (shown in green) and H (shown in red).

Let $D' = V(H) \cap N_G(C_i)$ and pick A' such that $\textsc{Merger}(A', D') = A$ We have $\mathbf{w}(H) \geq \textsc{Steiner}(G[N_G[C_i]], D')$.

We will show that F' is a valid candidate subgraph for the term $M[i, A', j-1]$ as stated by the definition of M, by showing that F' conforms to properties (i), (ii), (iii) and (iv):

(i) We must show that for each set $B' \in A'$, there exists a connected component in F' containing the elements of B'. This follows directly from the choice of A' as defined above.

(ii) We have to show that each connected component in F' contains a vertex from S'. Here we can make a similar argument to the one made for the fourth base case; we have that D' consists of elements from S', as such, each altered component from F' as result of the removal of H must contain an element from D' and therefore S'. In addition, any unaltered components from F' contains an element from S' as F conforms to property (ii). Thus the property holds for F'.

(iii) We have that each terminal $v \in (C_1 \cup \ldots \cup C_i) \cap K$, some component in F contains v. Since we have that H does not contain the terminal of C_i, F still contains the terminal of C_i. Hence the property holds for F'.

(iv) We must show that $F' \cap G[N_G[C_i]]$ consists of at most $j-1$ components. We have that $F \hat{\cap} G[N_G[C_i]]$ consists of at most j components. One of the components in $F \hat{\cap} G[N_G[C_i]]$ (H) was removed, therefore the property holds for F'.

It follows that F' is valid candidate subgraph for the term $M[i, A', j-1]$. We have $\mathbf{w}(F') \geq M[i, A', j-1]$.

And moreover,

$$\begin{aligned} M[i, A, j] &= \mathbf{w}(F) = \mathbf{w}(H) + \mathbf{w}(F') \\ &\geq \textsc{Steiner}(N_G[C_i], D') + M[i, A', j-1] \\ &\geq \min_{D', A'} \{\textsc{Steiner}(N_G[C_i], D') + M[i, A', j-1]\} \end{aligned}$$

2. $F \hat{\cap} G[N_G[C_i]]$ contains no connected component without a terminal. By the multiway cut, C_i contains at most one terminal and therefore $F \hat{\cap} G[N_G[C_i]]$ contains at most one connected component. If $F \hat{\cap} G[N_G[C_i]]$ consists of one component, it must contain $K \cap C_i$. In order to show that F is a valid candidate for $M[i, A, j-1]$, we need to show that properties (i), (ii), (iii) and (iv) hold. Trivially, (i), (ii) and (iii) hold. We are left to show (iv); that $F \hat{\cap} G[N_G[C_i]]$ consists of at most $j-1$ components. To show this, we will distinguish two cases:

   2*a*. Suppose we have that $F \hat{\cap} G[N_G[C_i]]$ consists of one connected component and therefore $C_i \cap K \neq \emptyset$. Then, it must be that $j \geq 2$, as a result of the entry not satisfying the conditions of the first four base cases. It follows that for F, $j - 1 \geq 1$ aligns with property (iv).

   2*b*. Suppose we have that $F \hat{\cap} G[N_G[C_i]]$ consists of no connected components and therefore $C_i \cap K = \emptyset$. Then, it must be that $j \geq 1$, as a result of the entry not satisfying the conditions of the first four base cases. It follows that for F, $j - 1 \geq 0$ aligns with property (iv).

   Thus F is a valid candidate for $M[i, A, j-1]$, we have $M[i, A, j] = \mathbf{w}(F) \geq M[i, A, j-1]$.

Combining the two cases above gives us that $M[i, A, j] \geq \min_{D', A'}\{\textsc{Steiner}(N_G[C_i], D') + M[i, A', j-1], M[i, A, j-1]\}$ This concludes the correctness proof.

Time and space analysis A recurring term in the cardinality of entries for the recurrence cases will be the number of combinations for A and $S' \subseteq S$. The number of options for S' is at most $2^{|S|}$. Observe that A can be represented as a sequence consisting of integers ranging from 1 to $|S'|$ of length $|S'|$. Suppose we have an order in $S' = \{s_1, \ldots, s_{|S|}\}$. Then, integer $1 \leq i \leq |S'|$ in this sequence denotes to which set of A element s_i belongs to. We can use this representation to bound the number of possible options for A, which evaluates to $|S'|^{|S'|} \leq |S|^{|S|} = 2^{|S| \log |S|}$ possible combinations for A and S'.

As always, to analyse the running time of the algorithm, we will consider the cases of the recurrence separately and sum the contribution of each case to the total running time.

For the first base case, the number of corresponding entries in M is equal to the number of combinations for $S' \subseteq S$ and A, and will therefore be equal to $2^{|S| \log |S|}$ as stated above. For each entry, the time to compute $\textsc{Steiner}(G[B], B)$ can be determined using the result from Theorem 2.3. This result applies to each element $B \in A$, therefore we find that the time per entry becomes as follows:

$$\begin{aligned}
\sum_{B \in A} 3^{|B|} n^{\mathcal{O}(1)} &\leq \sum_{B \in A} 2^{\mathcal{O}(|S|)} n^{\mathcal{O}(1)} && \text{since } |B| = \mathcal{O}(|S|) \\
&\leq 2^{\mathcal{O}(|S|)} n^{\mathcal{O}(1)} |S| && \text{at most } |S| \text{ sets in } A \\
&= 2^{\mathcal{O}(|S|)} n^{\mathcal{O}(1)} && \text{since } |S| = n^{\mathcal{O}(1)}
\end{aligned}$$

Multiplying the number of entries by the time per entry results in a total time of at most $2^{\mathcal{O}(|S| \log |S|)} \cdot n^{\mathcal{O}(1)}$ for the first base case.

The number of corresponding entries in M for the second and third base cases is at most the number of combinations for $S' \subseteq S$ and A, multiplied with the number of components in C. The time per entry is constant, hence the total time for these cases becomes $2^{\mathcal{O}(|S| \log |S|)} \cdot m \leq 2^{\mathcal{O}(|S| \log |S|)} \cdot n$.

The number of corresponding entries in M for the fourth base case has the same upper-bound as the second and third base cases: $2^{\mathcal{O}(|S| \log |S|)} \cdot m$. The time per entry is the result of checking all combinations for D^* and A^*, and computing the $\textsc{Steiner}(G[N_G[C_i]], D^*)$ term for each of the combinations. From the definition of the $\textsc{Merger}$ function, combined with $\textsc{Merger}(A^*, D^*) = A$, we have that D^* is implied when we know A^* and A. A is known for each of these entries, hence the number of combinations relies solely on the number of options for A^*. This once again can be bounded by $2^{|S| \log |S|}$ using the same argument as we have seen above. For each of the combinations, we require to evaluate the time it takes to compute the term $\textsc{Steiner}(G[N_G[C_i]], D^*)$. Using the result from Theorem 2.3, this evaluates to $3^{|D^*|} n^{\mathcal{O}(1)} \leq 2^{\mathcal{O}(|S|)} n^{\mathcal{O}(1)}$. Then the total time per entry becomes at most $2^{\mathcal{O}(|S| \log |S|)} \cdot n^{\mathcal{O}(1)}$. This multiplied with the bound on the number of entries results in $2^{\mathcal{O}(|S| \log |S|)} \cdot n^{\mathcal{O}(1)} \cdot m = 2^{\mathcal{O}(|S| \log |S|)} \cdot n^{\mathcal{O}(1)}$ in total for the fourth base case.

For the step case, the time per entry remains the same for the same reasoning ($2^{\mathcal{O}(|S| \log |S|)} \cdot n^{\mathcal{O}(1)}$). Though the number of entries is multiplied by the number of possible values for $j > 1$. We have that $0 \leq j \leq |N_G(C_i)|$, since $|N_G(C_i)| \leq |S|$, the total number of entries is that of base case four, multiplied with $|S|$. Then the total time for the step case becomes $2^{\mathcal{O}(|S| \log |S|)} \cdot n^{\mathcal{O}(1)} \cdot |S| \cdot m = 2^{\mathcal{O}(|S| \log |S|)} \cdot n^{\mathcal{O}(1)}$.

The combined running time for the entire algorithmic approach can be obtained by totaling the contributions of each of the cases, and therefore results in $2^{\mathcal{O}(|S| \log |S|)} \cdot n^{\mathcal{O}(1)}$ total time.

Observation. Note that in order to solve $\textsc{Steiner Tree}$ from scratch, we would need to find a multi-way cut S first. Using the approach presented in the book by Cygan et al. [7], this adds a total of $\mathcal{O}(4^{|S|} \cdot |S|^3 \cdot n^{\mathcal{O}(1)})$ to the running time. Then the total running time becomes $\mathcal{O}(2^{\mathcal{O}(|S| \log |S|)} \cdot n^{\mathcal{O}(1)})$.

The space required to run the algorithm consists of the dynamic programming table itself, added to the space required to evaluate the $\textsc{Steiner}$ subroutine. The number of entries in the table becomes $2^{|S| \log |S|} \cdot m \cdot |S| = 2^{\mathcal{O}(|S| \log |S|)} \cdot n^{\mathcal{O}(1)}$, accounting for all possible values of i, A and j.

Using the approach by Cygan et al. [7, Thm 10.6], the space required to evaluate $\textsc{Steiner}(G, D)$ becomes $2^{|D|} \leq 2^{|S|}$. Resulting in a total of $2^{|S| \log |S|} \cdot n^{\mathcal{O}(1)}$ required total space.

■

The algorithm presented here could also be applied to solve the Weighted Steiner Tree problem; an instance of the problem where the input graph G has non-negative edge weights. This however would require the subroutine to resolve the Steiner function to be changed, as the algorithm by Cygan et al. [7, Thm 10.6] does not accommodate the weighted variant. We may use the algorithm by Dreyfus and Wagner [12] resolving the weighted variant in time $3^{|K|} \cdot n^{\mathcal{O}(1)}$.

Chapter 5

Conclusion

This work presents three main results in the field of parameterized complexity. We explore refined parameterizations to solve Triangle Graph Packing, 4-Cycle Graph Packing and Steiner Tree exactly. For the former two problems, the presented algorithms build on the concept of $\mathcal{H}$-elimination distance. Their methods show how to use $\mathcal{H}$-elimination forests to find graph packings. In addition, the approach for solving 4-Cycle Graph Packing provides a step towards generalizing Graph Packing by showing how to search a finite number of subtrees in the elimination forest to combine subgraphs into a 4-cycle.

The FPT algorithms admit a running time of $2^{\mathcal{O}(d)} \cdot n^{\mathcal{O}(1)}$ and $2^{\mathcal{O}(d^2 \log d)} \cdot n^{\mathcal{O}(1)}$ for Triangle Graph Packing and 4-Cycle Graph Packing respectively, parameterized by the depth d of a given $\mathcal{H}$-elimination forest. In contrary to formerly known algorithms that are parameterized in the solution size, we have shown that the use of the novel parameter benefits sparser graphs which admit to a lower $\mathcal{H}$-elimination distance than solution size. For Steiner Tree, we have seen an algorithm solving the problem in $2^{\mathcal{O}(|S| \log |S|)} \cdot n^{\mathcal{O}(1)}$ time and space, parameterized in the size of a given multi-way cut. The size of an optimal multi-way cut is at most that of the terminal set, which is a more common parameter for Steiner Tree.

While exponential space is often the case for fast FPT algorithms [24], requiring space exponential to a parameter could be obstructive. Therefore a feasible improvement could be to obtain an FPT algorithm requiring polynomial space. The exponential space complexity required for our procedures originates from the size of our dynamic programming tables, as we have to consider the possible subsets/subgraphs that are associated with smaller subproblems. In comparison to dynamic programming, one may step towards alternative techniques in order to attain polynomial space. We have seen branching algorithms [7, Thm 10.6] and kernelization approaches [14] capable of doing so for algorithms with more conventional parameter choices.

Lastly, the packing procedures given in this paper could be expanded to solve the generalized graph packing problem. We suggest a similar framework to that of our 4-cycle packing algorithm, where we consider decomposing the to-be-packed graph $H = C_4$. One may look into separators for doing so for any H.

Bibliography

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows - theory, algorithms and applications*. Prentice Hall, 1993.

[2] Noga Alon, Raphael Yuster, and Uri Zwick. Color coding. In *Encyclopedia of Algorithms*, pages 335–338. 2016.

[3] Béla Bollobás and Stephen E. Eldridge. Packings of graphs and applications to computational complexity. *J. Comb. Theory, Ser. B*, 25(2):105–124, 1978.

[4] Yixin Cao and Dániel Marx. Interval deletion is fixed-parameter tractable. *ACM Trans. Algorithms*, 11(3):21:1–21:35, 2015.

[5] Markus Chimani, Petra Mutzel, and Bernd Zey. Improved steiner tree algorithms for bounded treewidth. *Journal of Discrete Algorithms*, 16:67–78, 2012. Selected papers from the 22nd International Workshop on Combinatorial Algorithms (IWOCA 2011).

[6] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[7] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[8] Marek Cygan, Dániel Marx, Marcin Pilipczuk, and Michal Pilipczuk. The planar directed k-vertex-disjoint paths problem is fixed-parameter tractable. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 197–206. IEEE Computer Society, 2013.

[9] Dipti Dash and Debarshi Kumar Sanyal. Steiner system-based topology-transparent priority scheduling for wireless ad hoc networks. *Internet Technol. Lett.*, 2(3), 2019.

[10] Jitender S. Deogun, Ton Kloks, Dieter Kratsch, and Haiko Müller. On vertex ranking for permutations and other graphs. In Patrice Enjalbert, Ernst W. Mayr, and Klaus W. Wagner, editors, *STACS 94, 11th Annual Symposium on Theoretical Aspects of Computer Science, Caen, France, February 24-26, 1994, Proceedings*, volume 775 of *Lecture Notes in Computer Science*, pages 747–758. Springer, 1994.

[11] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[12] Stuart E. Dreyfus and R. A. Wagner. The steiner problem in graphs. *Networks*, 1(3):195–207, 1971.

[13] Ranel E. Erickson, Clyde L. Monma, and Arthur F. Veinott Jr. Send-and-split method for minimum-concave-cost network flows. *Math. Oper. Res.*, 12(4):634–664, 1987.

[14] Michael R. Fellows, Christian Knauer, Naomi Nishimura, Prabhakar Ragde, Frances A. Rosamond, Ulrike Stege, Dimitrios M. Thilikos, and Sue Whitesides. Faster fixed-parameter tractable algorithms for matching and packing problems. *Algorithmica*, 52(2):167–176, 2008.

[15] Mike Fellows, Pinar Heggernes, Frances A. Rosamond, Christian Sloper, and Jan Arne Telle. Finding k disjoint triangles in an arbitrary graph. In Juraj Hromkovic, Manfred Nagl, and Bernhard Westfechtel, editors, *Graph-Theoretic Concepts in Computer Science, 30th International Workshop,WG 2004, Bad Honnef, Germany, June 21-23, 2004, Revised Papers*, volume 3353 of *Lecture Notes in Computer Science*, pages 235–244. Springer, 2004.

[16] Bart M. P. Jansen, Jari J. H. de Kroon, and Michal Wlodarczyk. Vertex deletion parameterized by elimination distance and even less. *CoRR*, abs/2103.09715, 2021.

[17] Bart M. P. Jansen, Jari J. H. de Kroon, and Michal Wlodarczyk. Vertex deletion parameterized by elimination distance and even less. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1757–1769. ACM, 2021.

[18] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

[19] Karthik C. S., Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating dominating set. *J. ACM*, 66(5):33:1–33:38, 2019.

[20] David G. Kirkpatrick and Pavol Hell. On the completeness of a generalized matching problem. In Richard J. Lipton, Walter A. Burkhard, Walter J. Savitch, Emily P. Friedman, and Alfred V. Aho, editors, *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 240–245. ACM, 1978.

[21] Jaroslav Nesetril and Patrice Ossona de Mendez. Grad and classes with bounded expansion I. decompositions. *Eur. J. Comb.*, 29(3):760–776, 2008.

[22] Jaroslav Nesetril and Patrice Ossona de Mendez. Grad and classes with bounded expansion II. algorithmic aspects. *Eur. J. Comb.*, 29(3):777–791, 2008.

[23] Hao Tang, Genggeng Liu, Xiaohua Chen, and Naixue Xiong. A survey on steiner tree construction and global routing for VLSI design. *IEEE Access*, 8:68593–68622, 2020.

[24] Gerhard J. Woeginger. Space and time complexity of exact algorithms: Some open problems (invited talk). In Rodney G. Downey, Michael R. Fellows, and Frank K. H. A. Dehne, editors, *Parameterized and Exact Computation, First International Workshop, IWPEC 2004, Bergen, Norway, September 14-17, 2004, Proceedings*, volume 3162 of *Lecture Notes in Computer Science*, pages 281–290. Springer, 2004.

[25] Raphael Yuster. Combinatorial and computational aspects of graph packing and graph decomposition. *Comput. Sci. Rev.*, 1(1):12–26, 2007.