Enumeration of Minimal Separators with Special Properties

Master's Thesis

Andrea Nardi

Enumeration of Minimal Separators with Special Properties

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Andrea Nardi
born in Boston, USA

Software Engineering Research Group
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
www.ewi.tudelft.nl

Enumeration of Minimal Separators with Special Properties

Author: Andrea Nardi
Student id: 5180120

Abstract

In this paper we give a historical and theoretical background to minimal triangulation and its relation to minimal separators.

We introduce a new type of minimal separators, the minimal meta separator, which its size is a lower bound for the treewidth problem, its fill-in is a lower bound for the minimum fill-in problem and can be used for the aid of enumeration of minimal separators.

Furthermore, we introduce the idea of transformation which are controlled graph modifications that predictably modify the minimal separators of such graph. Namely, we introduce the transformations of inclusion minimal clique decomposition, saturate minimal separator, edge addition and minimal meta separator decomposition. These transformations can aid the architecting of algorithms for the enumeration of minimal separators.

Finally, we then apply such methods on the problem of enumeration of minimal almost-clique separators and demonstrate faster runtimes compared to the current state of the art.

Preface

The main topic of this thesis is the enumeration of minimal separators. The enumeration of minimal separators is a step in the solving of many graph theoretical problems, ranging from the treewidth problem, the minimum fill-in problem, the treedepth problem and many more.

In this thesis we introduced the idea of transformation which is a graph transformation that predictably changes the minimal separators in said graph. We later use the idea of transformation to design two algorithm for the enumeration of minimal almost-clique separators, which are an important tool for the preprocessing for the treewidth problem. We show that our algorithms run faster than the state of the art algorithms, and we prove that one of our algorithm has runtime complexity of $O(n^3m)$ which is a new result to the best of out knowledge. For a more in depth summary of our contribution, refer to Chapter 6.

This thesis would not have been possible without the support of many people. Many thanks to my supervisor, Emir Demirović, whom I exchanged many theoretical ideas, read numerous time my thesis and met me on a weekly basis. Many thanks to Steven Kelk, my bachelor's professor, for his engaging teaching style, which grew my curiosity for graph theory. Also thanks to Mathijs de Weerdt and Leo van Iersel for offering support. Finally I would like to thank my friends and family. They supported me with love throughout my thesis had to endure my passionate talks about separators.

Andrea Nardi
Delft, the Netherlands
August 26, 2022

Contents

Chapter 1

Introduction

Many algorithms require a set of minimal separators to generate a solution and often the runtime is mostly determined by the enumeration of said minimal separators.

The Bouchitté and Todinca dynamic programming algorithm[11] for solving treewidth and minimum fill-in problems requires the enumeration of all minimal separators of a graph, the Bodlaender algorithm[9] for treewidth preprocessing requires the enumeration of minimal almost-clique separators, Tamaki's algorithm[35] for computation of treewidth requires the computation of minimal separators of size at most k, Korhonen's algorithm[23] requires the enumeration of small minimal separators to solve the treedepth problem, graph separation problems algorithms[27] require enumeration of what the literature has refereed to as important separators.

We will discuss the relation between minimal separators and the problem of minimal triangulation and the problem of optimal triangulation and discuss how the problem of enumeration of minimal separators aids the minimal triangulation problem and vice versa.

We will introduce the idea of transformation, which are controlled graph modifications that predictably transforms the minimal separators of said graph and prove such transformations. We will use the transformations to architect an algorithm for the enumeration of minimal separators that are interesting in the field of optimal triangulation, namely the minimal almost-clique separators and what we will introduce as minimal meta separators.

Finally we will empirically compare the newly architected algorithms with functionally similar algorithms in the literature.

The chapters will be structured in the following way:

- In Chapter 2 we will introduce the graph theory notation, give the theoretical background of minimal separator, minimal triangulation, minimal separators interesting in the field of minimal triangulation and methods of enumerating said minimal separators.

- In Chapter 3 we will present the idea of transformation, which is a controlled graph modification which predictably modifies the minimal separators of said graph. Furthermore we will prove each transformation for their correctness.

- In Chapter 4 we will architect algorithms for the enumeration of minimal separator which are interesting in the field of minimal triangulation with the transformations presented in Chapter 3.
- In Chapter 5 we will empirically compare the newly architected algorithm with a functionally similar algorithm in the literature.
- In Chapter 6 we will give our final remarks.

Chapter 2

Preliminary

2.1 Notation

We use standard graph theory notation. The graphs referenced in this work are simple graphs, meaning that the graphs are finite, with no self loops and no parallel edges. We often represent with G an arbitrary graph. $V(G)$ represents the vertex set of G. $E(G)$ represents the set of edges of G. $G[X]$ where $X \subseteq V(G)$ is the induced subgraph of G with $V(G[X]) = X$ and $E(G[X]) = \{\{v,u\} | v \in X \wedge u \in X | \{v,u\} \in E(G)\}\}$. Given a vertex set S let $N(S)$ be the vertex set neighboring S or more formally $N(S) = \{v | v \notin S \wedge u \in S | \{v,u\} \in E(G)\}$. Given a graph G and a vertex set S we will represent $G + clique(S)$ the graph where G is made into a clique, or more formally $V(G + clique(S)) = V(G)$ and $E(G + clique(S)) = E(G) \cup \{\{v,u\} | v \neq u | vu \in S\}$. Given a graph G and two non adjacent vertices we will represent as $G + edge(v,u)$ the graph where $V(G + edge(v,u)) = V(G)$ and $E(G + edge(v,u)) = E(G) \cup \{v,u\}$.

2.2 Minimal Separators

Intuitively speaking, a separator is a vertex set which removal will divide the graph into separate connected components which are called associated components (Definition 2.2.3). For example in Figure 2.1, we can observe that the removal of the vertices S from G divides the graph into the connected components C_1, C_2 and C_3, therefore, we say that C_1, C_2 and C_3 are associated components of the separator S.

An associated component C of a separator S is said to be full if all its neighboring vertices are in S, or more formally, $S = N(C)$ (Definition 2.2.5). In Figure 2.1 C_1 and C_2 are full components of S as $S = N(C_1) = N(C_2)$.

A separator S is defined as a vu-separator if S "separates" the vertices v and u into two separate associated components (Definition 2.2.1). In Figure 2.1, S is a vu-separator, vw-separator and a wu-separator.

We say that a vu-separator S is minimal if there is no subset of S which is also a vu-separator (Definition 2.2.2). In Figure 2.1, S is a minimal vu-separator but not a minimal wv-separator as $\{t\} \subset S$ is a (minimal) wv-separator. It is intuitivelly discernable that a separator S is a minimal separator if and only if it has at least two full components C_1 and C_2. This is because S is a minimal vu-separator for an arbitrary $v \in C_1$ and $u \in C_2$ and there cannot be a minimal vu-separator subset of S as any vertex in $S = N(C_1) = N(C_2)$ would "bridge" C_1 and C_2 (Theorem 2.1).

Generally we say that a separator S is a minimal separator if there exists two vertices vu such that S is a minimal vu-separator. In Figure 2.1, both S and $\{t\}$ are minimal separators even though one is a subset of the other.

An inclusion minimal separator (Definition 2.2.4) is a minimal separator for which there is no subset which is a minimal separator. In Figure 2.1, $\{t\}$ is an inclusion minimal separator (but not S). It is intuitively discernible that a separator is an inclusion minimal separator if and only if all its associated component are full components (Theorem 2.2). This is because if a minimal separator S has an associated component C that is not full then the separator $Z = N(C)$ is a subset minimal separator and if S has only full components then S cannot have a subset separator $Z \subsetneq S$ as any vertex $v \in S \setminus Z$ would "bridge" all the associated components of S into one connected component.

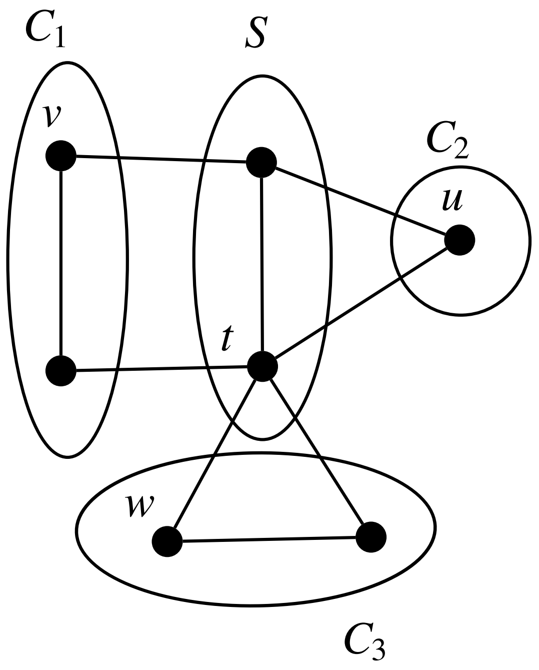

Figure 2.1: S is a minimal separator (because it minimaly separates v and u). C_1, C_2 and C_3 are associated components of S. C_1 and C_2 are full components of S (because $N(C_1) = N(C_2) = S$). S is a vw-separator but not a minimal vw-separator because $\{t\} \subset S$ is also a vw-separator. $\{t\}$ is an inclusion minimal separator but not S.

Definition 2.2.1. A vertex set S is a vu-separator if and only if the following conditions are met:

1. $S \subseteq V(G) \setminus \{v,u\}$.
2. Every path from v to u has a vertex $w \in S$.

Definition 2.2.2. A vu-separator S is a minimal vu-separator if and only if there is no subset $Z \subset S$ where Z is a vu-separator.

Generally we say that a vertex set S is a minimal separator if there exists two vertices vu such that S is a minimal vu-separator. We can observe that given a vu-separator S, if removed from G, or more formally, given $G' = G[V(G) \setminus S]$, G' has at least two connected components, one containing v and one containing u. We define these connected components as associated components of S.

Definition 2.2.3. The vertex set C is an associated component of S or $C \in \mathcal{C}(S)$ if C is a connected component of $G[V(G) \setminus S]$.

Definition 2.2.4. A minimal separator S is an inclusion minimal separator if and only if there is no minimal separator Z such that $Z \subset S$.

Full components are useful tools to determine if a separator is a minimal separator and/or an inclusion minimal separator. They are defined in Definition 2.2.5.

Definition 2.2.5. An associated component C of S is said to be a full component if $S = N(C)$.

Theorem 2.1 ([1]). A separator S has at least two full components if and only if S is a minimal separator.

Theorem 2.2 ([9]). A separator S is an inclusion minimal separator if and only if all it's associated components are full components.

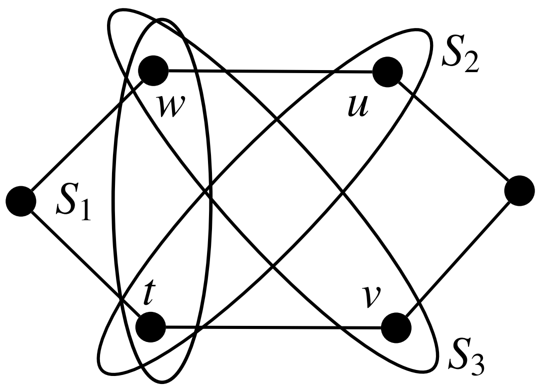

Figure 2.2: S_1 is parallel to S_2 and S_3. S_2 crosses S_3 because S_2 is a vw-separator (and S_3 is a ut-separator).

An important relation among minimal separators is whether two separators are parallel or crossing(Definition 2.2.6).

Definition 2.2.6. Two separators S and Z are said to be crossing or $S \sharp Z$ if S is a vu-separator where $vu \in Z$. Otherwise S and Z are said to be parallel or $S \parallel Z$.

It is not hard to prove that this relation is symmetric for minimal separators [1, 31].
From the Definition 2.2.6 we will prove that clique separators have no crossing separators.

Lemma 2.2.1 ([31]). A clique separator has no crossing separators.

Proof. Let S be a clique separator. Let Z for the sake of contradiction be a separator such that $Z \sharp S$. That means that there exists $vu \in S$ such that Z is a vu-separator. But this is a contradiction because S is a clique and therefore $\{v,u\} \in E(G)$. □

Lemma 2.2.2. Given two minimal separators S and Z where $S \parallel Z$, there exists an associated component $C \in \mathcal{C}(S)$ of S such that $Z \subseteq C \cup S$.

Proof. For the sake of contradiction, let us assume there is no $C \in \mathcal{C}(S)$ such that $Z \subseteq C \cup S$. That means that there are two associated components $C_v, C_u \in \mathcal{C}(S)$ where $vu \in Z$ and $v \in C_v$ and $u \in C_u$. That means that S is a uv-separator and $S \sharp Z$, which is a contradiction. □

2.3 Minimal Separators With Special Properties

Here we will describe some of the minimal separators that are important in the field of optimal triangulation.

2.3.1 Minimal Clique Separator

Minimal clique separators, as their name implies, are minimal separators which are cliques. Many NP-Hard problems can be solved with a divide and conquer approach, where the graph is decomposed around the minimal clique separators, a solution is found per atom of the decomposition and a solution is merged between the atoms. Some of the problems that can be solved with this scheme are graph coloring, maximum clique, maximum independent set, treewidth and minimum fill-in. Refer to [5] for an historical and theoretical background about minimal clique separators.

2.3.2 Minimal Separator of Size at Most k

Tamaki in [36, 35] proposes an algorithm for determining whether $tw(G) \leq k$ which requires only the minimal separators of G of size at most k.

Furthermore, certain algorithms[23, 41] for solving the treedepth problem require the set of small minimal separators. It has been later proven in [40] that only the minimal separators of size $|S| \leq 2tw(G)$ are required for solving the treedepth problem.

Minimal separators of size at most k are commonly enumerated [35, 22] with an algorithm that in the literature is referred to as Takata's recurrence[34]. Takata's recurrance[34] is a recursive formulation of the set of minimal vu-separators of G. Let $\Delta_{vu}(G,C,X)$ be the set of minimal vu-separators S of G such that $S \cap C = \emptyset$ and $S \supseteq X$. Takada's recurrence search tree has as root node $\Delta_{vu}(G,\{v\},\emptyset)$ and as leaf nodes $\Delta_{vu}(G,C,N(C)) = \{N(C)\}$. The recursive formulation of Takada's recurrence is $\Delta_{vu}(G,C,X) = \Delta_{vu}(G,C \cup \{w\},X) \cup \Delta_{vu}(G,C,X \cup \{w\})$ for $w \in N(C)$ where intuitively, $\Delta_{vu}(G,C \cup \{w\},S)$ is the set of all minimal vu-separators which do not contain w and the set $\Delta_{vu}(G,C,X \cup \{w\})$ is the set of all minimal vu-separators which do contain w.

Given a graph G, an algorithm for finding whether the treewidth of G is at most k is to find a minimal triangulation H of G such that the maximum clique of H is at most $k+1$ and the minimal separators of H are at most k [36, 35]. To find such minimal triangulation only the enumeration minimal separators of G which sizes are at most k are required. Tamaki in [35] proposes the the use of Takata's recurrence to enumerate all minimal separators of size at most k by pruning branches $\Delta_{vu}(G,C,X)$ where $|X| > k$.

2.3.3 Minimal Almost-Clique Separators

Bodlaender and Koster [9] define a safe separator S as a separator such that $tw(G) = tw(G + clique(S))$. The authors prove categories of separators as being safe separators, one of them being inclusion minimal almost-clique separator. An almost-clique S is a vertex set for which there exists a vertex $v \in S$ for which $S \setminus \{v\}$ is a clique. Figure 2.3 gives an example of a inclusion minimal almost-clique separator saturated into a clique without modifying its treewidth.

Tamaki in [37] proves that also minimal almost-clique separators are safe separators. Furthermore Tamaki proposes a method for enumeration of all minimal almost-clique separators in time $O(n^2m)$. Such method of enumeration of all almost-clique minimal separators is reproduced in Algorithm 1. Furthermore Tamaki uses this method to enumerating a pairwise parallel minimal separators that are safe in G. He achieves that using the set of all almost-clique minimal separators to generate a set of pairwise parallel minimal almost-clique separators reproduced in Algorithm 2 and saturating those separators in G and repeating until G is unchanged similarly in Algorithm 3. Multiple query of pairwise parallel minimal almost-clique separators in Algorithm 3 are necessary as the saturation of a minimal separator might generate other minimal almost-clique separators.

2.3.4 Minimal Triangulation

Minimal triangulation has a tight theoretical relation with minimal separators. Many minimal triangulation algorithms require enumeration of minimal separators and many minimal separator enumeration algorithms require a minimal triangulation of a graph.

Algorithm 1 Enumerate All Minimal Almost-Clique Separators in G

```
Input: Graph $G$
Output: Set of All Minimal Almost-Clique Separators in $G$
 1: procedure EnumerateAllAlmostCliqueSeparators($G$)
 2:     $\Delta \leftarrow \emptyset$
 3:     for all $v \in V(G)$ do
 4:         $\Delta' \leftarrow \mathit{MinimalCliqueSeparators}(G[V(G) \setminus \{v\}])$
 5:         for all $S \in \Delta'$ do
 6:             if $S \cup \{v\}$ is a minimal separator of $G$ then
 7:                 $\Delta \leftarrow \Delta \cup \{S \cup \{v\}\}$
 8:             end if
 9:         end for
10:     end for
11:     return $\Delta$
12: end procedure
```

Algorithm 2 Enumerate Pairwise Parallel Minimal Almost-Clique Separators in G

```
Input: Graph $G$
Output: Set of Pairwise Parallel Minimal Almost-Clique Separators in $G$
 1: procedure PairwiseParallelAlmostCliqueSeparators($G$)
 2:     $\Delta' \leftarrow \mathit{EnumerateAllAlmostCliqueSeparators}(G)$
 3:     $\Delta \leftarrow \emptyset$
 4:     for all $S \in \Delta'$ do
 5:         if $S$ does not cross any separator in $\Delta$ then
 6:             $\Delta \leftarrow \Delta \cup \{S\}$
 7:         end if
 8:     end for
 9:     return $\Delta$
10: end procedure
```

Algorithm 3 Minimal Almost-Clique Separator Based Preprocessing

```
Input: Graph $G$
Output: Preprocessed $G$
 1: procedure Preprocess($G$)
 2:     repeat
 3:         $\Delta \leftarrow \mathit{PairwiseParallelAlmostCliqueSeparators}(G)$
 4:         for $S \in \Delta$ do
 5:             $G \leftarrow G + \mathit{clique}(S)$
 6:         end for
 7:     until $G$ is unchanged
 8:     return $G$
 9: end procedure
```

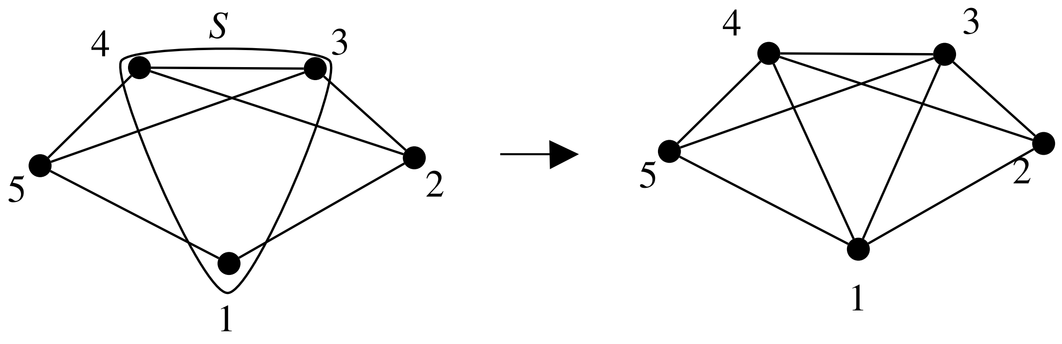

Figure 2.3: $S = \{1,3,4\}$ is an inclusion minimal almost-clique separator of G. S is an almost clique separator because S because $S \setminus \{1\}$ is a clique. S can be saturated without changing the treewidth of G

In this section we will give the definition and theoretical background of triangulation and minimal triangulation and highlight how minimal separators are used to find a minimal triangulation and how minimal triangulations are used to enumerate minimal separators.

2.4 Triangulation

Before we can explain what a triangulation is, we have to define what a chordal graph is. While there are many ways to define a chordal graph, we will use the definition in Theorem 2.3.

Theorem 2.3 ([15]). A graph G is chordal if and only if every minimal separator of G is a clique.

A triangulation of a graph G is a super graph H with $V(H) = V(G)$ and $E(H) \supseteq E(G)$ such that H is chordal. Minimal triangulation H of a graph G is a triangulation of G such that there is no triangulation H' of G such that $E(H') \subsetneq E(H)$.

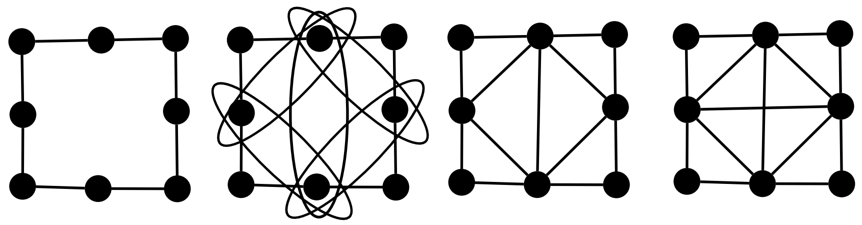

Figure 2.4: (far left) An arbitrary graph G. (middle left) Set of pairwise parallel minimal separators $\mathcal{S}$ of G. (middle right) A minimal triangulation G by saturating $S \in \mathcal{S}$ into a clique. (far right) A non-minimal triangulation of G.

2.4.1 Obtaining a Triangulation by Elimination Ordering

Elimination ordering is a problem which historically originates in the field of sparse matrix factorization. A method of factorization of sparse symmetric matrices is Gaussian elimination, where the pivoting of every rows and columns is iterativelly done which during the process sometimes the replacement of a previously zero entry with a non-zero entry is required. The entries that previously were zero and are replaced by a non-zero value are called fill-in. The appearance of fill-in values during Gaussian elimination comes with the negative effects of greater space requirements for the representation of the sparse matrix and greater computational burden for the following iterations of the Gaussian elimination. It was soon discovered that a good order of the pivoting of the rows and columns reduces the numbers of fill-ins. Parter in [32] introduced the idea of Elimination Game (EG) which, with the representation of a $n \times n$ symmetric sparse matrix A as a graph G with n vertices, where the vertices $ij \in \{1,2,...,n\}$ are adjacent if A_{ij} is non-zero, simulates the appearance of fill-in values after each pivot in A, by simulating the pivoting of the ith row and column in A by saturating the neighbors of the vertex i into a clique with what we will call fill-in edges before eliminating the vertex i from the graph G. The pseudo code of the EG can be found in Algorithm 4.

```
Algorithm 4 Elimination Game
Input: Graph $G$ and vertex ordering $\sigma$
Output: Fill-in edges from the vertex ordering $\sigma$
 1: procedure EliminationGame($G$, $\sigma$)
 2:     $F \leftarrow \emptyset$                                    ▷ Set of fill-in edges
 3:     $G_0 \leftarrow G$
 4:     $i \leftarrow 1$
 5:     for all $v = \sigma_i$ do
 6:         $F_i \leftarrow \emptyset$
 7:         for all $uw \in N_{G_{i-1}}(v)$ where $\{u,w\} \notin E(G_{i-1})$ do
 8:             $F_i \leftarrow F_i \cup \{\{u,w\}\}$
 9:         end for
10:         $F \leftarrow F \cup F_i$
11:         $V(G_i) \leftarrow V(G_{i-1}) \setminus v$
12:         $E(G_i) \leftarrow E(G_{i-1}) \cup F_i \setminus \{\{v,u\} \mid u \in N_{G_{i-1}}(v)\}$  ▷ Remove edges adjacent to
    $v$ and add fill-in edges
13:         $i \leftarrow i+1$
14:     end for
15:     return $F$
16: end procedure
```

Given an arbitrary graph G and an arbitrary elimination ordering of $V(G)$, we will represent by G^+_α the graph G with the additional fill-in edges resulting from applying the EG with the α elimination ordering. In light of [17] a graph G and an elimination ordering α, G^+_α is a chordal graph. Therefore, an arbitrary elimination ordering α can be used to obtain a triangulation G^+_α of G while in practice the use of heuristic elimination ordering is used to obtain triangulations with low minimum fill-in or treewidth. For example, Minimum Degree (MD) ordering is a common heuristic elimination ordering for the minimum fill-in problem [19] and Minimum Average Fill (MAF) [29][37] is an effective elimination ordering for treewidth.

It is important to note that not every triangulation obtained by an arbitrary elimination ordering yields a minimal triangulation. An example of elimination ordering which yields a non-minimal triangulation can be observed in Figure 2.5. For the problem of finding an elimination ordering which yields a minimal triangulation, the Maximum Cardinality Search for Minimal Triangulation (MCS-M) [3] is an easy and effective algorithm for finding minimal triangulation in time $O(nm)$ while only a restricted family of minimal triangulations can be obtained.

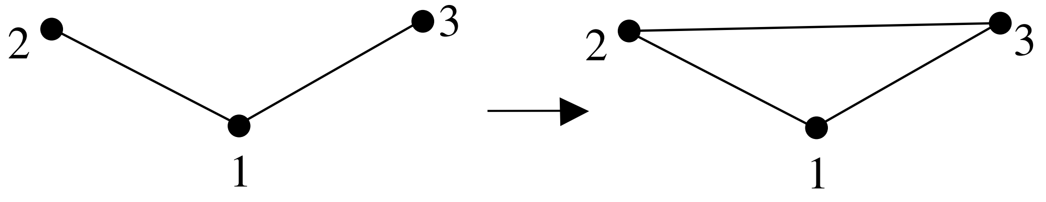

Figure 2.5: An example of elimination ordering which yields a non-minimal triangulation. The left graph G is already a chordal graph, therefore, the only minimal triangulation of G is G itself, but an edge has been added with this elimination ordering.

2.4.2 Perfect Elimination Ordering

Given a graph G, a perfect elimination ordering is an elimination ordering of G which does not yield fill-in edges. A definition of chordal graph is a graph which has a perfect elimination ordering.

Theorem 2.4 ([17]). A graph G is a chordal graph if and only if G has a perfect elimination ordering.

It is not hard to observe that given G^+_α, α is a perfect elimination ordering of G^+_α.

Given a chordal graph G a method of finding its perfect elimination ordering is an algorithm commonly referred to as Maximum Cardinality Search (MCS)[39] which runs at $O(n+m)$ time.

It is easy to observe that given a minimal triangulation H of G and the perfect elimination ordering α of H, $H = G^+_\alpha$. This is because if there is a fill-in edge e such that $e \in E(G^+_\alpha)$ and $e \notin E(H)$ then α is not a perfect elimination ordering of H as e

would be one of its fill-in and if $E(G^+_\alpha) \subsetneq E(H)$ then H is not a minimal triangulation of G.

2.4.3 Problems Defined as an Optimal Triangulation Problem

The problem of optimal triangulation is, given a graph G, finding a minimal triangulation H of G such that a certain objective is minimized.

The treewidth of a graph G can be expressed as a function of all its minimal triangulations. Let $Tr(G)$ be the set of all the minimal triangulations of G and let $MC(H)$ be the set of maximal cliques of H. To note that $MC(H)$ can be computed $O(nm)$ time for chordal graphs[2]. Then the treewidth of G is defined as:

Theorem 2.5 ([7]). $tw(G) = \min_{H \in Tr(G)} \max_{W \in MC(H)} |W| - 1$

Therefore, the problem of treewidth can be formulated as an optimal triangulation problem of finding the the minimal triangulation $H \in Tr(G)$ of minimum maximum clique.

While the problem of minimum fill-in was originally the problem of finding an elimination ordering with the minimum number of fill-ins, given the fact that the minimum fill-in elimination ordering must yield a minimal triangulation H of G with number of fill-ins equal to $|E(H)| - |E(G)|$ and given that the perfect elimination ordering of H is the elimination ordering of G that would yield H, the problem of finding an elimination ordering with minimum number of fill-ins is equivalent as finding a minimal triangulation H of G such that $|E(H)| - |E(G)|$ is minimum.

Theorem 2.6 ([42]). $mfi(G) = \min_{H \in Tr(G)} |E(H)| - |E(G)|$

Therefore, the problem of minimum fill-in can be formulated as finding the minimal triangulation H of G such that $|E(H)|$ is minimized. Once such H is found, the perfect elimination ordering of H is the elimination ordering which yields the minimum number of fill-ins in G.

For additional information regarding the treewidth and minimum fill-in problem formulated as minimal triangulation problems, refer to [11].

Other problems that can be formulated as optimal triangulation problems are generalized hypertreewidth[20], fractional hypertreewidth[28], phylogenetic trees identification[10], treelength[16]. For additional information about problems that can be formulated as optimal triangulation problem refer to [8, 18, 25].

2.4.4 Minimal Triangulation Sandwich Problem

The minimal triangulation sandwich problem is, given a graph G and an elimination ordering α, finding a minimal triangulation H of G such that $E(G) \subseteq E(H) \subseteq E(G^+_\alpha)$. One of the reason why finding H is desirable, is that minimal triangulations have lower or equal objectives in many optimal triangulation problems compared to their non-minimal triangulation counterpart. This would allow use of a heuristic elimination ordering α to get a triangulation G_α^+ of low objective and finding a subgraph H which is a minimal triangulation of G which will have a equal or lower objective compared to G_α^+.

This approach was introduced by Blair in [6] to further improve heuristic elimination orderings for the minimum fill-in problem.

Berry et al. in [4] developed the algorithm Minimal Minimum Degree (MMD) for obtaining a minimal triangulation which is a subgraph of the triangulation obtained from MD ordering. MMD has the advantage of giving empirically low minimum fill-in and guaranteeing a minimal triangulation with fill-in lower or equal to its MD counterpart.

The same approach was applied by Tamaki in [37] to obtain the algorithm Minimal Minimum Average Fill (MMAF) from the MAF algorithm for the heuristic computation of minimal triangulation with low treewidth. Similarly to MMD, MMAF guarantees a minimal triangulation with treewidth equal or lower to its MAF counterpart.

2.4.5 Minimal Triangulation and Minimal Separator

Parra and Scheffler in [31] proves a 1-to-1 correspondence between a maximal set of pairwise parallel minimal separators $\mathcal{S}$ of G and its minimal triangulation H. The author proves that given a minimal triangulation H of G, the minimal separators H are minimal separators of G, or in other words, $\Delta(H) \subseteq \Delta(G)$ and $\Delta(H)$ is a maximal set of pairwise parallel minimal separators of G. In the same manner, given a maximal set of pairwise parallel minimal separators $\mathcal{S}$, saturation of each minimal separator $S \in \mathcal{S}$ into a clique yields a minimal triangulation H of G where $\Delta(H) = \mathcal{S}$.

Many minimal separator enumeration algorithms use this property to find a set of pairwise parallel set of minimal separators from a graph G. In the same manner many minimal triangulation algorithm use a set of minimal separators to find a desirable minimal triangulation.

In the following sections we will explain what are some of the algorithms for the enumeration of minimal separators from a minimal triangulation and some minimal triangulation algorithms based on minimal separators.

2.4.6 Enumeration of Minimal Separators from a Minimal Triangulation

In the previous chapter, we discussed how to obtain a minimal triangulation of a graph, such as MCS-M, MMD and MMAF. Here, we will discuss, given a minimal triangulation H how to obtain its minimal separators and some of its main applications.

As we stated at the start of Section 2.4.5, Parra and Scheffler in [31] demonstrated the 1-to-1 correspondence between a maximal set of parallel minimal separators $\mathcal{S} \subseteq \Delta(G)$ and a minimal triangulation H of G. More specifically, we are interested in the following property:

Theorem 2.7 ([31]). Given a graph G and a minimal triangulation H of G, $\Delta(H)$ is a maximal set of pairwise parallel minimal separators of G.

This has some interesting uses, for example for the enumeration of minimal clique separators. Because of Lemma 2.2.1 we know that any maximal set of pairwise parallel minimal separators of a graph G must contain all of the minimal clique separator of G. This is the approach used by Leimer in [26] which is a further development of the method by Tarjan in [38] to enumerate all minimal clique separators of a graph. The algorithm uses an elimination ordering α which yields a minimal triangulation, oftentimes obtained with MCS-M to consequently enumerate all the minimal separators of G_α^+ and filter out all the minimal separators that are not cliques.

Another use of minimal triangulation for the enumeration of minimal separators is by Tamaki in [37] where he uses the minimal triangulation H of G obtained by MMAF to heuristically enumerate minimal almost-clique separators.

A common algorithm to enumeration all the minimal separator of a chordal graph is [13] and has $O(n+m)$ runtime.

2.4.7 Minimal Triangulation from a Set of Minimal Separators

As Parra and Scheffler proved in [31], given a maximal set of pairwise parallel minimal separators $\mathcal{S}$ of G, the graph H, obtained by saturating every $S \in \mathcal{S}$ from G into a clique, H is a minimal triangulation of G or more formally:

Theorem 2.8 ([31]). Let $\mathcal{S} \subseteq \Delta(G)$ a maximal set of pairwise parallel minimal separators of the graph G. Let the graph H be the graph obtained by saturating every $S \in \mathcal{S}$ in G into a clique. H is a minimal triangulation of G and $\Delta(H) = \mathcal{S}$.

This also has implications on the upper bound of the cardinality of any maximal set of pairwise parallel minimal separators. Let us introduce the following property on the number of minimal separators in a chordal graph.

Theorem 2.9 ([13]). Given a chordal graph H, $|\Delta(H)| \leq |V(H)| - 1$.

Or in other words, a chordal graph has at most $n-1$ minimal separators. This naturally provides an upper bound on any maximal set of pairwise parallel minimal separators $\mathcal{S} \subseteq \Delta(G)$. To the best of our knowledge this is a novel result.

Lemma 2.9.1. Given a set of pairwise parallel minimal separators $\mathcal{S} \subseteq \Delta(G)$ of G, $|\mathcal{S}| \leq |V(G)| - 1$.

Proof. For the sake of contradiction let $|\mathcal{S}| > |V(G)| - 1$. Let $\mathcal{S}^*$ be a maximal set of pairwise parallel minimal separator such that $\mathcal{S}^* \supseteq \mathcal{S}$. By Theorem 2.8 the minimal triangulation H obtained by saturating $S \in \mathcal{S}^*$ would have minimal separators $\mathcal{S}^* = \Delta(H) > |V(H)| - 1$ which would contradict Theorem 2.9. □

Furthermore, the Bouchitté-Todinca algorithm for finding optimal triangulation [11], Tamaki's algorithm for solving treewidth [35] and Carmeli et al. algorithm for enumerating minimal triangulations [12] all rely on Theorem 2.8 to obtain a minimal triangulation from a given list of minimal separators. For the sake of conciseness, we will only give a rough explanation of the minimal triangulation enumeration algorithm by [12].

The authors in [12] define what is referred as a separator graph $\mathcal{G}$ of a graph G where $V(\mathcal{G}) = \Delta(G)$ and for $S,Z \in \Delta(G)$, $\{S,Z\} \in E(\mathcal{G})$ if and only if $S \sharp Z$. Or in other words, each vertex of $\mathcal{G}$ is represented by a minimal separator of G, and two minimal separators $S,Z \in V(\mathcal{G})$ are adjacent in $\mathcal{G}$ if and only if S crosses Z. Once $\mathcal{G}$ is represented, any maximal independent set of $\mathcal{G}$ corresponds to a maximal set of pairwise parallel minimal separators of G which also corresponds to a unique minimal triangulation of G. Therefore, the authors employ a maximal independent set enumeration algorithm on $\mathcal{G}$ to enumerate the minimal triangulations of G. In Figure 2.6 there is an example of a graph G, its separator graph $\mathcal{G}$, two maximal independent sets of $\mathcal{G}$ and its corresponding minimal triangulations of G.

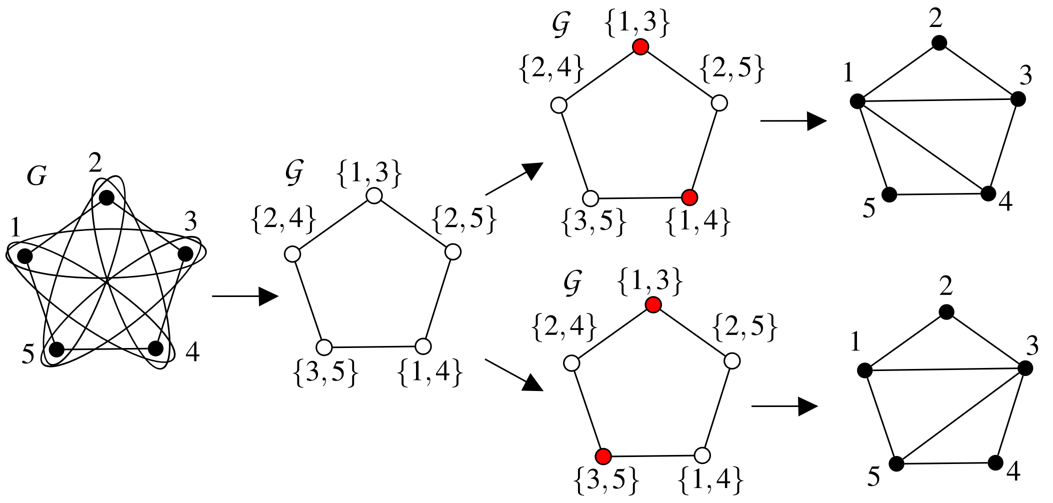

Figure 2.6: G is a graph with its minimal separator circled. $\mathcal{G}$, the separator graph of G, has as vertices the minimal separators of G, therefore, $V(\mathcal{G}) = \{\{1,3\},\{1,4\},\{2,4\},\{2,5\},\{3,5\}\}$ and edges $\{1,3\}-\{2,5\}$, $\{2,5\}-\{1,4\}$, $\{1,4\}-\{3,5\}$, $\{3,5\}-\{2,4\}$, $\{2,4\}-\{1,3\}$, because $\{1,3\}\sharp\{2,5\}$, $\{1,3\}\sharp\{2,5\}$ and so on. Each maximal independent set of $\mathcal{G}$ represents a minimal triangulation of G.

2.4.8 State of the Art Algorithms and Minimal Separators Enumeration Bottleneck

The current state of the art algorithms for the treewidth and minimum fill-in problems is currently the Bouchitté and Todinca's algorithm[11] often referred to as the

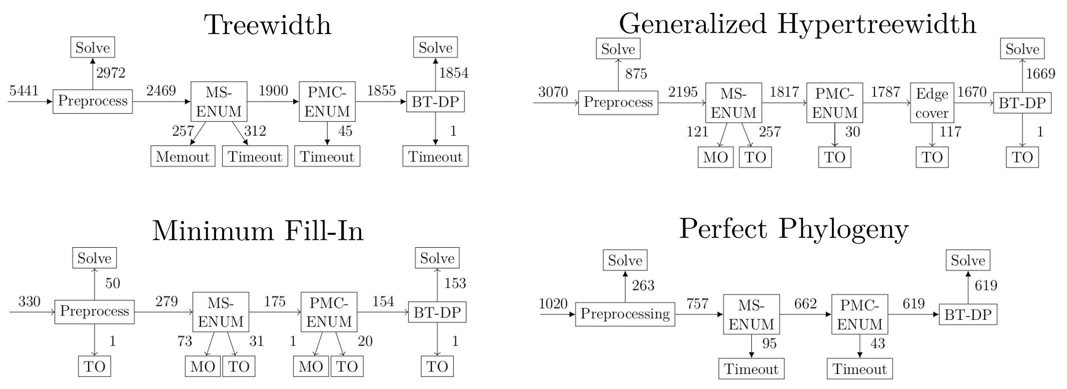

Figure 2.7: The execution of the BT algorithm presented by Korhonen in [25]. The majority of the memout and timeout happens during the enumeration of minimal separators, represented in the graphic as ”ms-enum”

BT algorithm in the literature. Both the winners of the treewidth and the minimum fill-in challenge in the PACE 2017 challenge [14] use a modified version of the BT algorithm.

The BT algorithm requires the enumeration of all minimal separators of a graph and this has been shown to be a bottleneck.

In his master thesis [24], Korhonen reports runtime of the BT algorithm in many optimal triangulation problems. From the data presented by Korhonen and shown in Figure 2.7, the majority of execution timeout and memout happens in the minimal separator enumeration phase of BT.

It can be also inferred from the research direction for many optimal triangulation papers that the enumeration of minimal separators step is the most likely to allow speedups. Tamaki’s approach in speeding up the treewidth problem is by enumerating only separators of size k or using a heuristic non-exhaustive set of minimal separators [35]. Kanig in [21] proposes a method of enumerating minimal separators in order of cardinality to speed up the treewidth problem even though her paper was conclusively wrong. Korhonen in [22] proposes a method that is fixed parameter tractable for the enumeration of minimal separators of size at most k with the possible application of speeding up the treewidth problem.

Chapter 3

Transformation

Here we will introduce the concept of transformation. A transformation in the context of this paper is a function which, given a graph G and its set of minimal separators $\Delta(G)$, applies a controlled transformation to G into G' and outputs its set of minimal separators $\Delta(G')$.

We then show that the concept of transformation can be used to architect algorithms for the enumeration of minimal separators. This concept of controlled graph modification for the enumeration of minimal separators has been explored by Kenig in [21] even though conclusively the paper was not correct. The transformations that are introduced are inclusion minimal clique separator decomposition, saturate minimal separator, edge addition and minimal meta separator decomposition.

Before we introduce the concept of transformation we must prove few simple lemmas.

Lemma 3.0.1. Given G, $S \in \Delta(G)$, $C \in \mathcal{C}(S)$ and $vu \in C \cup S$ then $S \in \Delta(G + egde(v,u))$.

Proof. Because $vu \in C \cup S$, S is not a vu-separator. Therefore, there must exist $wk \in V(G)$ such that S is a minimal wk-separator and therefore, two full components $C_w, C_k \in \mathcal{C}(S)$ such that $w \in C_w$ and $k \in C_k$. Let $G' = G + edge(vu)$. $C_w, C_k \in \mathcal{C}_{G'}(S)$ therefore $S \in \Delta(G')$. □

3.1 Inclusion Minimal Clique Separator Decomposition (IMCSD)

Minimal clique separator decomposition is a divide and conquer approach applied in many NP-Complete problems as we already explained in section 2.3.1. We will prove that given a graph G and an inclusion minimal clique separator S, the separators of the induced subgraphs resulting from the decomposition around S exhaustively contain the separators of G apart from S itself.

As a practical example let's look at Figure 3.1. Let G, the graph on the left and let $S = \{s,t\}$ a minimal clique separator of G. G_1 and G_2 are the result of the decomposition of G around the separator S as observed in the example. As we can observe from this example $\Delta(G) = \Delta(G_1) \cup \Delta(G_2) \cup \{S\}$ and $\Delta(G_1) \cap \Delta(G_2) \cap \{S\} = \emptyset$. While Leimer in [26] provides a similar property, our Theorem 3.1 is stricter as we also prove that the sets of separators is strictly equal and disjoint.

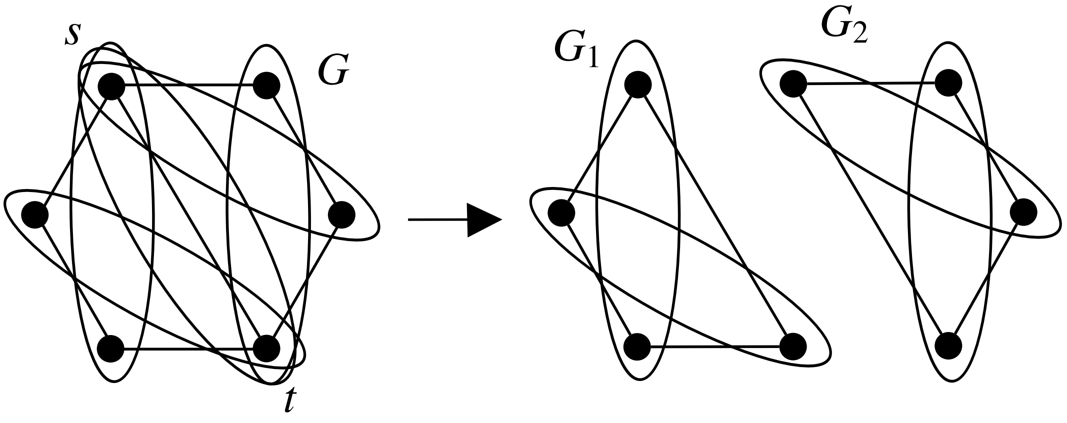

Figure 3.1: The separators of G is equal to the combination of the separators of G_1 and the separators of G_2 and the separator $\{s,t\}$.

Theorem 3.1. Given an inclusion minimal clique separator S of G

$$\Delta(G) = \{S\} \cup \bigcup_{C \in \mathcal{C}(S)} \Delta(G[S \cup C])$$

where for $\{C_1, C_2, ...C_n\} = \mathcal{C}(S)$ the sets $\{S\}, \Delta(S \cup C_1]), \Delta(G[S \cup C_2]), ..., \Delta(G[S \cup C_n])$ are disjoint.

Proof. We will divide the proof into the following. (1) For any $Z \in \Delta(G)$ where $Z \neq S$ there exists one and only one $C \in \mathcal{C}(S)$ such that $Z \in \Delta(G[S \cup C])$ (2) For any $Z \in \Delta(G[S \cup C])$, $Z \in \Delta(G)$.

1. Lemma 2.2.1 implies $Z \parallel S$ in the graph G. That means that there is an associated component $C_S \in \mathcal{C}(Z)$ such that $S \subseteq Z \cup C_S$ and an associated $C_Z \in \mathcal{C}(S)$ such that $Z \subseteq S \cup C_Z$. C_Z is unique. The only way that C_Z would not be unique would be if $Z \subset S$ but this is not possible as S is an inclusion minimal separator.

   If $Z \supset S$ then Z must have the non full associated components $\mathcal{C}_G(S) \setminus C_Z$ in G. The associated components of Z in $G[S \cup C_Z]$ are $\mathcal{C}_G(Z) \setminus (\mathcal{C}_G(S) \setminus C_Z)$ therefore Z preserves its full components in $G[S \cup C_Z]$ therefore Z is still a minimal separator in $G[S \cup C_Z]$.

   If Z has two full components C_1 and C_2 where $C_1 \neq C_S$ and $C_2 \neq C_S$ in G, Z is also a minimal separator in $G[S \cup C_Z]$ with the full components C_1 and C_2.

If Z has only two full components with C_S being one then $C_S \cap (C_Z \cup S)$ must be a full component of Z in $G[C_Z \cup S]$. This is because if $v \in Z$ neighbors a vertex $u \in C_S$ in G then either (case 1) $u \in C_Z \cup S$ or (case 2) $v \in S$ (by the fact that S is a separator and there is no path between two associated components that does not pass by S). So for any $v \in Z$ (case 1) v neighbors a vertex in $C_S \cap (C_Z \cup S)$ because C_S is a full component of Z or (case 2) v neighbors a vertex in $S \setminus Z = S \cap C_S$ because S is a clique.

2. If $G[S \cup C]$ has a minimal separator Z, Z must have two vertices $vu \in S \cup C$ such that Z is a minimal vu-separator in $G[S \cup C]$. Z is still a minimal vu-separator in G therefore $Z \in \Delta(G)$.

□

Theorem 3.1 has not only implications for inclusive minimal clique separators but also minimal clique separators. This is because if a minimal clique separator S is not an inclusive minimal clique separator then that means there exists a separator $Z \subset S$ which is an inclusion minimal clique separator. If the transformation is applied recursively to every subset minimal separator of S eventually we would get a graph where S becomes inclusion minimal clique separator. Figure 3.2 demonstrates the transformation applied to the separator $\{s,t\}$ which is not initially inclusion minimal separator.

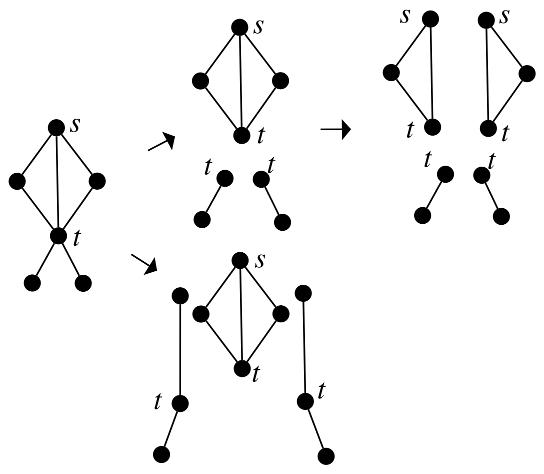

Figure 3.2: $\{s\}$ and $\{s,t\}$ are both minimal clique separators in G. While $\{s,t\}$ is not an inclusion minimal separator we can apply the IMCSD transformation by processing the transformation on $\{t\}$ first to make $\{s,t\}$ an inclusion minimal separator. Further more we can apply the transformation to $\{s,t\}$ consequently. If we process $\{s,t\}$ before $\{t\}$, then the minimal separator $\{t\}$ is present twice in two different atoms.

3.2 Saturate Minimal Separator (SMS)

Here we demonstrate that the completion of a minimal separator S into a clique removes all the separators in G which crosses S. While SMS is a common approach of minimal triangulation of a graph [30, 11], we did not observe any proofs that shows the relation of the minimal separators of the graph before and after the SMS step. While Parra and Scheffler prove in [30] the separators of a graph after saturating a maximal set of pairwise parallel separators, our proof is more general as it show the relation of the minimal separators of a graph before and after saturating a single minimal separator.

In Figure 3.3 there is an example of what happens to the separators of G once a separator is saturated into a clique.

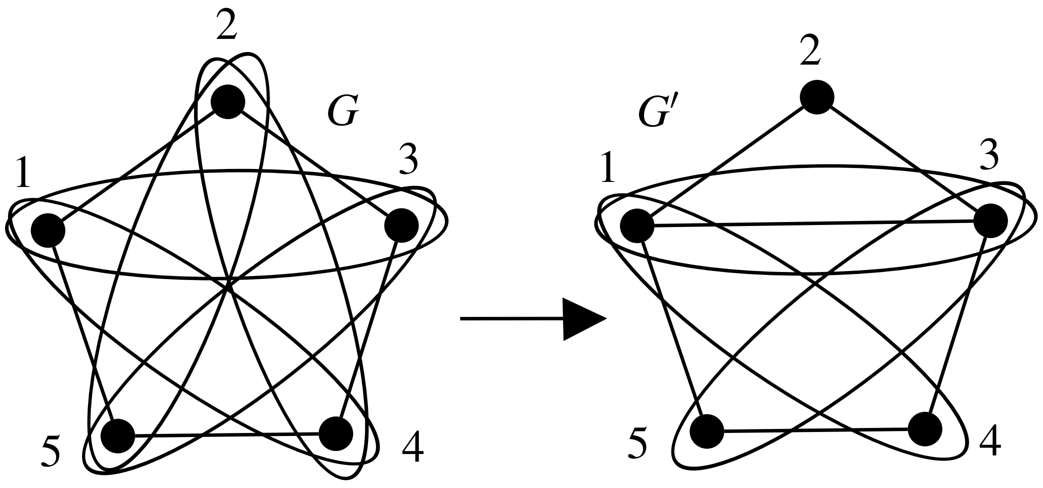

Figure 3.3: The separators of G and of $G' = G + clique(\{1,3\})$ where only separators parallel to $\{1,3\}$ remain. $\{2,5\}$, $\{2,4\} \notin \Delta(G')$ because $\{2,5\} \sharp \{1,3\}$ and $\{2,3\} \sharp \{1,3\}$.

Theorem 3.2. Given a minimal separator S of G,

$$G' = G + clique(S)$$
$$\Delta(G') = \{Z \mid Z \parallel S \mid Z \in \Delta(G)\}$$

Proof. We will split the proof into three parts: (1) Given a minimal separator $Z \in \Delta(G)$, if $Z \parallel S$ then $Z \in \Delta(G')$, (2) Given a separator $Z \in \Delta(G)$, if $Z \sharp S$ then $Z \notin \Delta(G')$, (3) For any $Z \in \Delta(G')$, $Z \in \Delta(G)$ or in other words $\Delta(G') \subseteq \Delta(G)$.

1. Because $Z \parallel S$ there must exist a component $C_S \in \mathcal{C}_G(Z)$ such that $S \subseteq C_S \cup Z$. Addition of edges in $C_S \cup Z$ does not change the fact that Z is a minimal separator due to Lemma 3.0.1.

2. Because $S \sharp Z$ there must exist two vertices $vu \in Z$ such that S is a vu-separator in G. S must be a vu-separator also in G'. For the sake of contradiction let us assume that $Z \in \Delta(G')$. That means that $S \sharp Z$ which contradicts Lemma 2.2.1.

3. $S \parallel Z$ in G' due to Lemma 2.2.1. That means that there must be an associated component $C_S \in \mathcal{C}_G(Z)$ such that $S \subseteq (C_S \cup Z)$. Let C be a full component of S for which $C \cap Z = \emptyset$. Note that $C \subseteq C_S$ as there is no path to a vertex $u \notin C_S$ which does not have a vertex in Z.

   If C_S is not a full component of Z then Z has at least two full components in both G' and G.

   If C_S is one of the two full components of Z in G' then $Z \in \Delta(G)$ if and only if C_S is a full component of Z in G. This is only possible if C_S is a connected component and $N(C_S) = Z$ in G.

   The only way that $N(C_S) = Z$ holds in G' but does not hold in G is if there are two vertices $v \in Z$ and $u \in C_S$ for which $\{v,u\} \in E(G')$ and $\{v,u\} \notin E(G')$ which only holds if $vu \in S$. But S has a full component $C \subseteq C_S$ which means that v must still neighbors a vertex in C_S if $v \in S$ as $v \in C \subseteq C_S$. This means that $Z = N(C_S)$ also in G.

   The only way that C_S is not a connected component in G is if vertices in S are in separate connected components of C_S as edges were removed only among vertices in S going from G' to G. But because S has a full component C where $C \subseteq C_S$ for which $N(C) = S$ and C is a connected component, S must be within the same connected component in G which is a contradiction.

□

3.3 Edge Addition

To describe this transformation we will need to define certain constructs first.

Let $\Delta_{vu}(G)$ be the set of minimal vu-separators in G. Let $\Gamma_{vu}(G)$ be the set of minimal vu-separators which has only two full components. Obviously, because $S \in \Gamma_{vu}(G)$ is a minimal vu-separator and has only two full components there exists only two full components $C_v, C_u \in \mathcal{C}_G(S)$ where $v \in C_v$ and $u \in C_u$.

Given two parallel vu-separators S and Z let $P_{vuG}(S,Z) = C_Z \cap C_S$ where $C_Z \in \mathcal{C}_G(S)$ with $Z \subseteq C_Z \cup S$ and $C_S \in \mathcal{C}_G(Z)$ with $S \subseteq C_S \cup Z$. Given a vu-separator and the vertex w where either $w = v$ or $w = u$ let $P_{vuG}(S,\{w\}) = C_w \setminus \{w\}$ where $C_w \in \mathcal{C}_G(S)$ with $w \in C_w$.

Given a vertex set C such that $C \subseteq V(G)$ let $Comp_G(C)$ be the sets of connected components of the induced subgraph $G[C]$.

Theorem 3.3. Given $v,u \in V(G)$ where $\{v,u\} \notin E(G)$ and $v \neq u$.

$$\begin{aligned}
&G' = G + edge(v,u)\\
&\Pi\\
&\Delta(G') = (\Delta(G) \setminus \Gamma_{vu}(G))\\
&\qquad \cup \{S \cup Z \mid S \parallel Z \wedge \exists_{C \in Comp_G(P_{vuG}(S,Z))} N(C) = S \cup Z \mid S,Z \in \Delta_{vu}(G)\}\\
&\qquad \cup \{S \cup \{v\} | \exists_{C \in Comp_G(P_{vuG}(S,\{v\}))} N(C) = S \cup \{v\} \mid S \in \Delta_{vu}(G)\}\\
&\qquad \cup \{S \cup \{u\} | \exists_{C \in Comp_G(P_{vuG}(S,\{u\}))} N(C) = S \cup \{u\} \mid S \in \Delta_{vu}(G)\}
\end{aligned}$$

Proof. We will divide the proof into the following parts: (1) $S \notin \Delta(G')$ where $S \in \Gamma_{vu}(G)$, (2) $S \in \Delta(G')$ where $S \in \Delta(G)$ and $S \notin \Gamma_{vu}(G)$, (3) $S \cup Z \in \Delta(G')$ where $S \parallel Z$ and there exists a $C \in Comp_G(P_{vuG}(S,Z))$ where $N(C) = (S \cup Z)$. (4) $S \cup \{v\} \in \Delta(G')$ where there exists a $C \in Comp_G(P_{vuG}(S\{v\}))$ where $N(C) = (S \cup \{v\})$ (5) Given $S \in \Delta(G')$, S is derived by one of the above cases.

1. S is not a minimal separator in G' because it has the only full component $C_v \cup C_u$ in G' where $C_v, C_u \in \mathcal{C}_G(S)$ and $v \in C_v$ and $u \in C_u$.

2. If $S \notin \Gamma_v u(G)$ then either S has more than two full components in G in which case S will have at least two full components in G', or S has at least one full component not containing either v or u in G, which means that S will still be a minimal separator in G'

3. Given $C_v, C_u \in \mathcal{C}_G(S)$ and $C'_v, C'_u \in \mathcal{C}_G(Z)$ where $v \in C_v$, $v \in C'_v$, $u \in C_u$, $u \in C'_u$, $S \cup Z$ has the full component C and either $C_v \cup C'_u$ or $C_u \cup C'_v$ in the graph G'.

4. $S \cup \{v\}$ must have the full components C and $C_u \in \mathcal{C}_G(S)$ where $u \in C_u$ in G'.

5. If there exists $C \in \mathcal{C}(S)$ where $vu \in C$ then:

   - If C is not a full component of S in G' then S must have the same full components in G which means $S \in \Delta(G)$, $S \notin \Delta_{vu}(G)$, $S \notin \Gamma_{vu}(G)$.
   - If C is a full component of S in G' and C is a connected component then C must be still a full component in G which means $S \in \Delta(G)$, $S \notin \Delta_{vu}(G)$, $S \notin \Gamma_{vu}(G)$.
   - If C is a full component of S in G' and C is not a connected component in G then there must exist two connected components C_1 and C_2 in the induced subgraph $G[C]$ where $v \in C_1$, $u \in C_2$, $S = N(C_1) \cup N(C_2)$ and $N(C_1), N(C_2) \in \Delta_{vu}(G)$

   Otherwise, $C \in \mathcal{C}_{G'}(S)$ where $v \in C$ and $u \in S$.

   - If u neighbors a vertex other than v in C then S must be a minimal separator in G as S retains its minimal separators therefore $S \notin \Gamma_{vu}(G)$.

- If u neighbors only v in C then C is not a full component in G'. This means that $N(C \setminus \{u\})$ is a minimal separator in G.

□

An example of this operation can be seen in Figure 3.4.

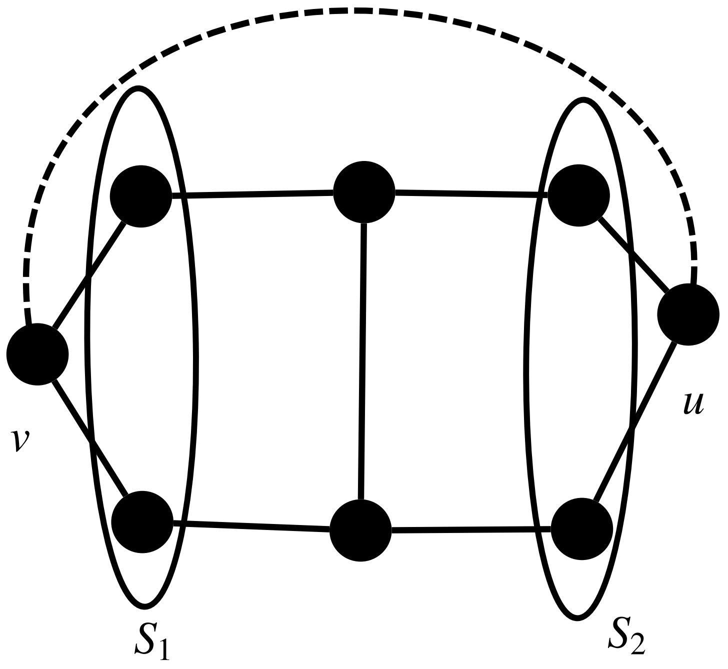

Figure 3.4: Let G be the graph without the edge $\{v,u\}$ and the G' be the graph with the edge $\{v,u\}$. S_1 and S_2 are two parallel minimal separators of G. G' has the minimal separators $\{v\} \cup S_2$, $\{u\} \cup S_1$ and $S_1 \cup S_2$ resulting from the separators S_1 and S_2.

3.4 Minimal Meta Separator Decomposition (MMSD)

We introduce the idea of minimal meta separator. A minimal separator S is meta if and only if for all $v \in S$, $|N(v) \cap S| \geq |S| - 2$.

Definition 3.4.1. A minimal separator S is meta if and only if for all vertices $v \in S$, $|N(v) \cap S| \geq |S| - 2$.

We consider minimal meta separators interesting in the field of minimal triangulation, as every crossing separator Z must contain for $uv \in S$, $Z \subseteq S \setminus \{v,u\}$. This is because for any two non-adjacent vertices in $vu \in S$ a minimal vu-separator Z (therefore crossing S) must contain $N(v) \cap N(u)$.

Lemma 3.3.1. Given a minimal meta separator S every crossing minimal separator Z where Z is a minimal vu-separator for $vu \in S$, Z must contain $S \setminus \{v,u\}$.

Proof. Let Z be a crossing separator of S. This means that there exists $vu \in S$ such that Z is a minimal vu-separator. Any vu-separator must contain $N(v) \cap N(u)$, therefore $Z \supseteq N(v) \cap N(u)$. Because S is meta either $|N(v) \cap S| = |S| - 1$ or $|N(v) \cap S| = |S| - 2$. If $|N(v) \cap S| = |S| - 1$ this would be a contradiction as v would have to neighbor u and there cannot be any vu-separators. Therefore, $|N(v) \cap S| = |S| - 2$ and because v does not neighbor u, $N(v) \cap S = N(u) \cap S = N(u) \cap N(v) \cap S = S \setminus \{v,u\}$. Therefore $Z \supseteq N(u) \cap N(v) \supseteq S \setminus \{v,u\}$ □

Corollary 3.3.1. If a graph G has a minimal meta separator S then $tw(G) \geq |S|$.

Proof. Any crossing minimal separator Z of S must contain at least $|S| - 2$ vertices in S because of Lemma 3.3.1 and 2 outside S because S is a minimal vu-separator for $vu \in Z, vu \notin S$. This means for any crossing separator Z of S, $|Z| \geq |S|$. Any minimal triangulation H of G, H must either have S or one of its crossing separator in G as a clique. This means that a maximal clique of H has at least size $|S| + 1$. Therefore $tw(G) \geq |S|$. □

Corollary 3.3.2. If a graph G has a minimal meta separator S then the minimum fill-in of G is at least $F(S)$.

Proof. If S is filled then the fill of G is at least $F(S)$. If a crossing minimal vu-separator Z of S is filled where $vu \in S$ then the fill of G is at least $F(S \setminus \{v,u\}) + F(\{v,u\})$, which because $N(v) \cap S = N(u) \cap S = S \setminus \{v,u\}$ equals $F(S)$. □

Corollary 3.3.3. Given a graph G, a minimal meta separator S and two non adjacent vertices $vu \in S$ of G, $\Delta_{vu}(G) = \{\bigcup_i Z_i \ |Z_1 \in \Delta_{vu}(G[C_1 \cup S])$ and $Z_2 \in \Delta_{vu}(G[C_2 \cup S])...Z_k \in \Delta_{vu}(G[C_k \cup S])\}$ where $C_i \in \mathcal{C}(S)$.

Proof. For any separator in $Z \in \Delta(G_{vu}[S \cup C])$ where $C \in \mathcal{C}_G(S)$, $Z \supseteq N(v) \cap N(u) \supseteq S \setminus \{v,u\}$ must hold because $Z \sharp S$. Let $\{C_1, C_2, C_3...C_n\} = \mathcal{C}_G(S)$. Let K_v^i and K_u^i be the associated component of $Z_i \in \Delta_{vu}(G[C_i \cup S])$ such that $v \in K_u^i$ and $u \in K_u^i$. K_v^i and K_u^i must be a full components of Z_i in $G[C_i \cup S]$ because Z_i is a minimal vu-separator. This means $Z_i = N(K_v^i)$. Given two arbitrary separator Z_i and Z_j, $Z_i \cap Z_j = S \setminus \{v,u\}$. $Z \in \Delta_{vu}(G)$ where $Z = \bigcup_{i \in \{1...n\}} Z_i$ with full components $K_u = \bigcup_{i \in \{1...n\}} K_v^i$ and $K_u = \bigcup_{i \in \{1...n\}} K_u^i$.

For any minimal separator $Z \in \Delta_{vu}(G)$ with the associated components $C_v, C_u \in \mathcal{C}(Z)$ where $v \in C_v$ and $u \in C_u$. $Z \cap (C \cap S)$ must be a minimal vu-separator in $G[C \cup S]$ with associated full components $C_v \cap (C \cap S)$ and $C_u \cap (C \cap S)$. □

An example of Corollary 3.3.3 is showcased in Figure 3.5.

All the minimal separators generated by Corollary 3.3.3 applied to S are crossing separators of S. This is because we are enumerating the minimal vu-separators where $vu \in S$. If we want to enumerate all the separators of G, we can use Corollary 3.3.3 to enumerate all the crossing minimal separators of S and use the SMS transformation

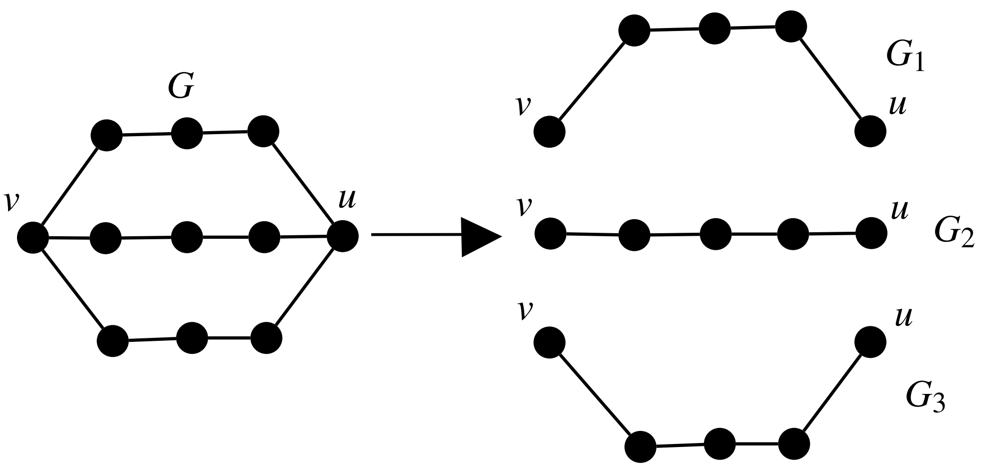

Figure 3.5: $S = \{v, u\}$ is a minimal meta separator of G. This means the $\Delta_{vu}(G) = \{\ S_1 \cup S_2 \cup S_3 \mid S_1 \in \Delta_{vu}(G_1)$ and $S_2 \in \Delta_{vu}(G_2)$ and $S_3 \in \Delta_{vu}(G_3)\}$.

to enumerate all the separators parallel to S. We can further decompose the graph by applying IMCSD to S, while ensuring beforehand that S is made into an inclusion minimal separator. An example of this can be seen in Figure 3.6

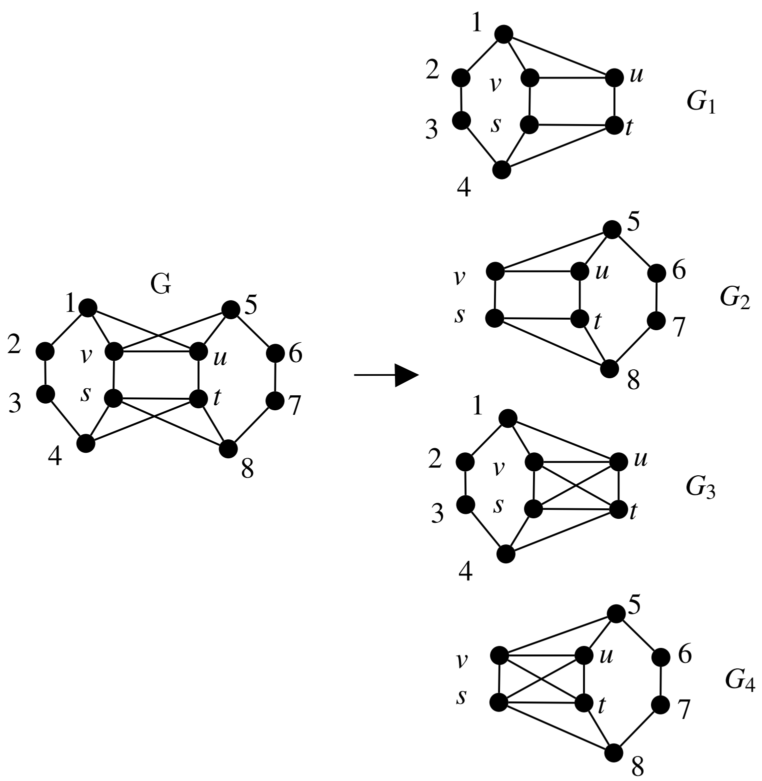

Figure 3.6: $S = \{v,u,s,t\}$ is a minimal meta separator of G. The crossing separators of S in G are $\Delta_{\sharp S}(G) = \{S_1 \cup S_2 \mid S_1 \in \Delta_{vt}(G_1) \text{ and } S_2 \in \Delta_{vt}(G_2)\} \cup \{S_1 \cup S_2 \mid S_1 \in \Delta_{su}(G_1) \text{ and } S_2 \in \Delta_{su}(G_2)\}$. The parallel separator of S in G are $\Delta_{\|S}(G) = \Delta(G_3) \cup \Delta(G_4)$. Therefore, all the minimal separators of G are $\Delta(G) = \{S\} \cup \Delta_{\sharp S}(G) \cup \Delta_{\|S}(G)$

Chapter 4

Enumeration of Minimal Separators With Special Properties

4.1 Number of Atoms After Decomposition By A Set of Pairwise Parallel Separators

Many of the algorithms for the enumeration of minimal separators will heavily rely on a step of SMS and a consequent IMCSD step. We will prove that at any step of repeated application of SMS and a consecutive IMCSD step, the number of atoms is at most $V(G)$. One of the properties that we will be using to prove such is that the number of maximal cliques in a chordal graph H is at most $V(H)$. While we could not find a source that explicitly proves such, Berry in [2] presents an algorithm which exhaustively enumerates the maximal cliques of H which enumerates an upper bound of n maximal cliques in any given chordal graph.

Lemma 4.0.1 ([2]). The number of maximal cliques in a chordal graph H is at most $V(H)$.

One of the consequences of the paper [11] is that in a chordal graph, any maximal clique Ω is a strict superset of its containing minimal separators S, or in other words, $S \subsetneq \Omega$.

Lemma 4.0.2 ([11]). Given a maximal clique Ω of a chordal graph H, There is no minimal separator $S \in \Delta(H)$ for which $\Omega \subseteq S$.

Lemma 4.0.3 ([11]). Given a chordal graph H, a maximal clique Ω of H and a minimal separator S of H, there is an unique associated component $C \in \mathcal{C}(S)$ for which $\Omega \subseteq C \cup S$

Theorem 4.1. Given a set of minimal clique separators $\mathcal{S}$ of G such that S is either inclusion minimal clique separator or every subset minimal separator is in $\mathcal{S}$, iteratively applying IMCSD on $S \in \mathcal{S}$ in order of cardinality produces at most $V(G)$ atoms.

Proof. Let $\mathcal{A}$ be the set of atoms. Let $\mathcal{S}^*$ be an arbitrary maximal set of pairwise parallel minimal separators of G such that $\mathcal{S}^* \supseteq \mathcal{S}$. Let H be the minimal triangulation obtained from $\mathcal{S}^*$ described in the process of Theorem 2.8. For each maximal clique Ω of H there must exist uniquely an atom $A \in \mathcal{A}$ such that $\Omega \subseteq A$ because of Lemma 4.0.3. Therefore $\mathcal{A}$ must have at most $V(G)$ elements. □

By proving Theorem 4.1, we show that iteratively applying IMCSD with the set of minimal clique separators in order of cardinality can be done with a runtime complexity of at most $O(nm)$.

4.2 Minimal Separator Enumeration Preprocessing

As we saw from Theorem 3.1 and the IMCSD transformation, the decomposition around an inclusion minimal clique separator does not change the minimal separators of a graph. We can use this fact to preprocess a graph G into smaller atoms which minimal separators can be enumerated independently to eventually get the separators of G.

We process the minimal clique separators in order of cardinality to make sure that by the time we process a minimal clique separator, such separator is inclusive. The algorithm is showcased in Algorithm 5. The enumeration of minimal clique separators can be efficiently done in $O(nm)$[26], the sorting of the minimal clique separators can be done in time $O(n)$ with bucket sort because minimal clique separators are a set of pairwise parallel minimal separators because of Lemma 2.3.1, therefore, because of Lemma 2.9.1 the set of minimal clique separators is at most of size $n-1$. Once we have the set of minimal clique separators ordered by cardinality we can apply a divide and conquer approach as showcased in Algorithm 5 to achieve a runtime of $O(nm)$.

4.3 Enumeration of All Minimal Almost Clique Separators

Because Bodlaender and Koster in [9] centered in finding inclusion minimal almost clique separators, some of their proofs are centered around inclusion minimal almost clique separators, here we will prove a similar lemma but more generally for minimal almost clique separators before proceeding to show the algorithm to enumerate minimal almost clique separators which are not necessarily inclusive.

Lemma 4.1.1. Given a graph G which has no minimal clique separator, S is a minimal clique separator in $G[V(G) \setminus \{v\}]$ if and only if $S \cup \{v\}$ is a minimal almost clique separator in G.

Proof. If $S \cup \{v\}$ is a minimal separator, there must be two vertices $uw \in V(G)$ for which there is no path from u to w which does not have a vertex $y \in (S \cup \{v\})$, therefore if there is minimally no path between uw which goes trough the vertices $V(G)$ that does not pass by $S \cup \{v\}$, there must be minimally no path between uw

Algorithm 5 Decompose G into its atoms such that the minimal separators are preserved

```
Input: Graph $G$ and
Output: $\mathcal{A}$, the atoms of $G$, and $\Delta_{clique}$, the clique minimal separators of $G$
 1: procedure MinimalCliqueSeparatorDecomposition($G$)
 2:     $\Delta_{clique} \leftarrow$ GetCliqueMinimalSeparators($G$)
 3:     $\mathcal{A} \leftarrow$ IMCSDRecursive($G$, $V(G)$, $\Delta_{clique}$)
 4:     return $(\mathcal{A}, \Delta_{clique})$
 5: end procedure
 6: procedure IMCSDRecursive($G$, $A$, $\Delta$)
 7:     if $\Delta = \emptyset$ then
 8:         return $\{A\}$
 9:     end if
10:     $\mathcal{A} \leftarrow \emptyset$
11:     $S \in \Delta$ where $S$ is the smallest element in $\Delta$
12:     for all $C \in \mathcal{C}_{G[A]}(S)$ do
13:         $\mathcal{A} \leftarrow$ IMCSDRecursive($G$, $C \cup S$, $\{Z \mid Z \subseteq C \cup S \mid Z \in \Delta\}$)
14:     end for
15:     return $\mathcal{A}$
16: end procedure
```

which goes through the vertices $V(G) \setminus \{v\}$ that does not pass by S. Therefore, if $S \cup \{v\}$ is a minimal separator in G then S is a minimal separator in $G[V(G) \setminus \{v\}]$

If $G[V(G) \setminus \{v\}]$ has a minimal clique separator S, there must exist $uw \in V(G) \setminus \{v\}$ such that S is a minimal clique uw-separator in $G[V(G) \setminus \{v\}]$. S cannot be a minimal separator in G because G has no minimal clique separator. The only way that S is not a minimal uw-separator in G is if there is a path between u and w that has the vertex v. Therefore, $S \cup \{v\}$ must be a minimal almost clique separator in G. □

If we apply the preprocessing step introduced in step Chapter 4.2, we are sure that there are no atoms such that $G[A]$ has a minimal clique separators. Therefore, given a graph G, the minimal clique separators Δ_{clique} of G and its atoms $\mathcal{A}$ after each step of IMCSD, we know that $\Delta(G) = \Delta_{clique} \cup \bigcup_{A \in \mathcal{A}} \Delta(G[A])$. Algorithm 6 describes how to compute all the almost clique minimal separators in $\Delta(G[A])$.

If we combine the minimal clique separators from the preprocessing described in Algorithm 5 and the minimal almost clique separators from each atom, we get Algorithm 7 which computes all the minimal almost clique separators of a graph. Note that all minimal clique separators are also minimal almost clique separators

Algorithm 6 Enumerate All Minimal Almost-Clique Separators in G Atom

```
Input: Graph $G$ which has no minimal clique separators
Output: Set of All Minimal Almost-Clique Separators in $G$
 1: procedure EnumerateAllAlmostCliqueSeparators($G$)
 2:     $\Delta \leftarrow \emptyset$
 3:     for all $v \in V(G)$ do
 4:         $(\Delta', \mathcal{A}') \leftarrow MinimalCliqueSeparatorDecomposition(G[V(G) \setminus \{v\}])$
 5:         for all $S \in \Delta'$ do
 6:             $\Delta \leftarrow \Delta \cup \{S \cup \{v\}\}$
 7:         end for
 8:     end for
 9:     return $\Delta$
10: end procedure
```

Algorithm 7 Enumerate All Minimal Almost-Clique Separators

```
Input: Graph $G$
Output: Set of All Minimal Almost-Clique Separators in $G$
 1: procedure EnumerateAlmostCliqueSeparator($G$, $A$)
 2:     $(\Delta, \mathcal{A}) \leftarrow MinimalCliqueSeparatorDecomposition(G)$
 3:     for all $A \in \mathcal{A}$ do
 4:         $\Delta \leftarrow \Delta \cup EnumerateAllAlmostCliqueSeparators(G[A])$
 5:     end for
 6:     return $\Delta$
 7: end procedure
```

4.4 Enumeration of Maximal Set of Pairwise Parallel Minimal Almost Clique Separators

In the previous chapter we described how to enumerate all minimal almost cliques separators within a graph. Most times though, we are interested in a single minimal triangulation of minimum treewidth. Therefore, we are interested in only a set of maximal pairwise parallel minimal almost clique separators. Furthermore, the completion of a separator into a clique has the possible effect of making, what once was a minimal non-almost clique separator, into a minimal almost clique separator which means that similarly to Algorithm 2, multiple queries of minimal almost clique separators are required.

We will first prove that given a set of minimal almost clique separators $\mathcal{S}$ such that for every $S \in \mathcal{S}$, $S \setminus \{v\}$ is a clique, $\mathcal{S}$ must be a set of pairwise parallel minimal almost clique separators.

Lemma 4.1.2. Given a set of minimal almost clique separators $\mathcal{S}$ and a vertex v such that for every $S \in \mathcal{S}$, $S \setminus \{v\}$ is a clique, $\mathcal{S}$ must be a set of pairwise parallel minimal almost clique separators.

Proof. Let $S, Z \in \mathcal{S}$. There is no uw-separator where $uw \in S \setminus \{v\}$, $u \neq v$ and $w \neq v$ because $S \setminus \{v\}$ is a clique. Therefore, any crossing separator of S must be a vu-separator where $u \in S \setminus \{v\}$. Z is not a vu-separator because $v \in Z$, therefore $S \parallel Z$ □

We can use the property from Lemma 4.1.2 and SMS to ensure that only the set of pairwise parallel minimal almost clique separators are enumerated and then we use IMCSD to ensure that the graph does not have any minimal clique separator at any given time. Our algorithm differently from Algorithm 2 does not explicitly compare two separators to check whether they are parallel nor it checks whether $S \cup \{v\}$ is a minimal separator. The algorithm is shown in Algorithm 8. Each repetition at line 4 takes $O(n^2 m)$. There can be at most $n-1$ pairwise parallel minimal separators because of Lemma 2.9.1 making this algorithm run at $O(n^3 m)$ runtime complexity. Neither [9] nor [37] provided a runtime complexity for the enumeration for a maximal set of pairwise parallel minimal separators and to best of our knowledge, our paper is the first to provide such runtime complexity.

Algorithm 8 Enumerate Pairwise Parallel Minimal Almost-Clique Separators in G

```
Input: Graph $G$ which has no minimal clique separators
Output: Set of Pairwise Parallel Minimal Almost-Clique Separators in $G$
 1: procedure EnumeratePairParallelAlmostCliqueSeparators($G$)
 2:     $\Delta \leftarrow \emptyset$
 3:     $\mathcal{A} \leftarrow \{V(G)\}$
 4:     repeat
 5:         for all $v \in V(G)$ do
 6:             for all $A \in \mathcal{A}$ where $v \in A$ do
 7:                 $(\mathcal{A}', \Delta') \leftarrow MinimalCliqueSeparatorDecomposition(G[A \setminus \{v\}])$
 8:                 for all $S \in \Delta'$ do
 9:                     $\Delta \leftarrow \Delta \cup \{S \cup \{v\}\}$
10:                     $G \leftarrow G + clique(S \cup \{v\})$
11:                 end for
12:                 $\mathcal{A} \leftarrow \mathcal{A} \setminus \{A\}$
13:                 for all $A' \in \mathcal{A}'$ do
14:                     $\mathcal{A} \leftarrow \mathcal{A} \cup \{A' \cup \{v\}\}$
15:                 end for
16:             end for
17:         end for
18:     until $\Delta$ is unchanged
19:     return $\Delta$
20: end procedure
```

4.5 Enumerate of Minimal Separators with Condition

Many minimal separators that are interesting in the field of minimal triangulation which are characterized by a condition, satisfies said conditions for an arbitrary subset of the separator, some examples being:

1. Any subset of an almost clique separator is also almost clique.
2. Any subset of a minimal separator of size at most k is also of size at most k.
3. Any subset of a minimal meta separator is also meta.

In this section we will demonstrate how we can use edge addition to enumerate such minimal separators.

4.5.1 Enumeration by Edge Addition

With Theorem 3.0.1 we demonstrated that $\Delta(G+edge(v,u))$ can be obtained from $\Delta(G)$ by operation of union among two separator $S,Z \in \Delta(G)$ or by operation of union of a minimal separator $S \in \Delta(G)$ with either $\{v\}$ or $\{u\}$. The enumeration of minimal separators of $\Delta(G)$ therefore can be done inductively starting with a graph G_0 where $V(G_0) = V(G)$ and $E(G_0) = \emptyset$ and obtaining $\Delta(G_i)$ from $\Delta(G_{i-1})$ where $G_i = G_{i-1} + edge(v,u)$ where $\{v,u\} \in E(G) \setminus E(G_{i-1})$. Our implementation of edge addition take $O(m\Delta(G_{i-1})^2)$ per edge addition. For the enumeration of minimal separators which satisfy a certain condition, namely almost clique separator, separators of size at most k and minimal meta separators, we filter for separators $S \in \Delta(G_i)$ if said condition is not satisfied for $G[S]$.

The order at which the edges are added greatly affects the runtime of the algorithm. We empirically observed that adding edges adjacent to a specific vertex at a time in the order of the reverse elimination ordering of the MCS-M works best. The intuition behind the reverse MCS-M elimination ordering is that if the graph happens to be a chordal graph, all the intermediate graphs G_i are also chordal, bounding the number of minimal separators to $n-1$ because of Lemma 2.9. While this rationale only works for a narrow family of graphs, we observed that empirically it works relatively well for any graph.

Chapter 5

Experimental Results

5.1 Implementation, Hardware and Instances

We implemented all algorithms in rust, with the rust standard library data structures. The code runs in a single thread. The code has been published in https://www.github.com/Andful/adagraph. All experiments were carried out on a 2.6GHz 12 core laptop with 16GB of RAM running Ubuntu 20.04.3 LTS.

The experiments were conducted on the PACE 2017 [14] treewidth exact track instances.

5.2 Enumeration of All Minimal Almost Clique Separators

We have presented so far 3 algorithms for the enumeration of all minimal almost clique separators. We presented Tamaki's method as Algorithm 1, we presented the combination of IMCSD and SMS for the enumeration of minimal almost clique separators as Algorithm 7 and we presented the Edge Addition algorithm for the enumeration of minimal separators with special characteristics in section 4.5.1.

For Tamaki's method while we would have preferred to use Tamaki's implementation of the algorithm, his implementation present in https://github.com/twalgor/tw seems to have a bug documented (by us) in https://github.com/twalgor/tw/issues/2. Therefore, Tamaki's method is based on our implementation of his method recreated to the most fidelity to the best of our abilities.

In Figure 5.1 is displayed Tamaki's method and Algorithm 7 for the enumeration of minimal almost clique separators. We can observe that vast majority of times our method is faster for the enumeration of minimal almost clique separators.

In Figure 5.2 is displayed the runtime of Algorithm 7 against the Edge Addition method. We can observer that Edge Addition runs much faster than Algorithm 7.

In Figure 5.3 we add the prepossessing step described in section 4.2. We can observe that while a minor improvement might be present, the preprocessing does not seem to distinctly improve the runtime of Edge Addition.

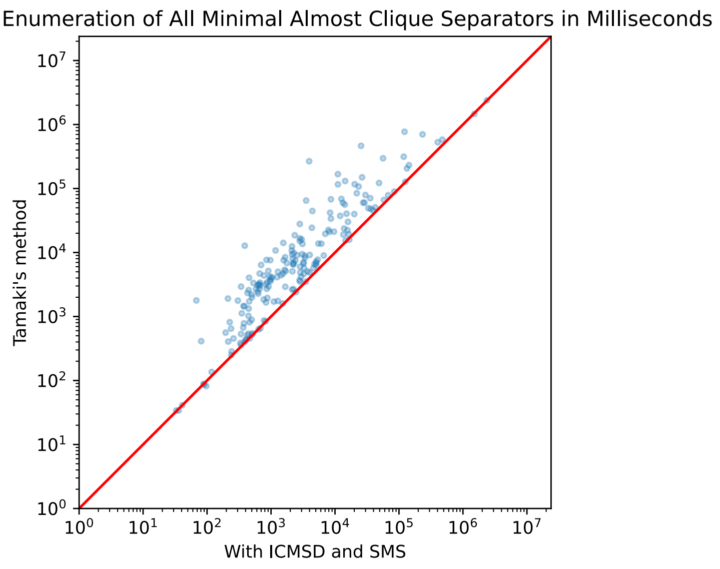

Figure 5.1: A comparison between Tamaki's method of enumeration ofs all minimal almost clique separators in a graph with ours. The red line represents the instances for which the two methods would run at equal runtime.

5.3 Enumeration Pairwise Parallel Minimal Almost Clique Separators

In this paper we studied two algorithms for the enumeration of pairwise parallel minimal almost clique separators. One of them is Tamaki's method showcased as Algorithm 3 and the one developed by us and showcased as Algorithm 8. In Figure 5.4 we can observe that the transformation introduce a great speed up on the runtime of the algorithm.

5.4 Treewidth Lowerbound Comparison

The cardinality of minimal clique separators, minimal almost-clique separators and minimal meta separators are all lowerbounds for treewidth. In this chapter we will compare the quality of these lowerbounds

We enumerated all minimal clique separators with the method developed by Leimer in [26], we enumerated all the minimal meta separators of the graph G with the Edge Addition algorithm, we enumerated all the almost clique minimal separators with the Algorithm 7 and we enumerated a pairwise parallel minimal almost clique separators with the Algorithm 8. We represented the lowerbound obtained by minimal clique separators by "clb", the lowerbound obtained by all

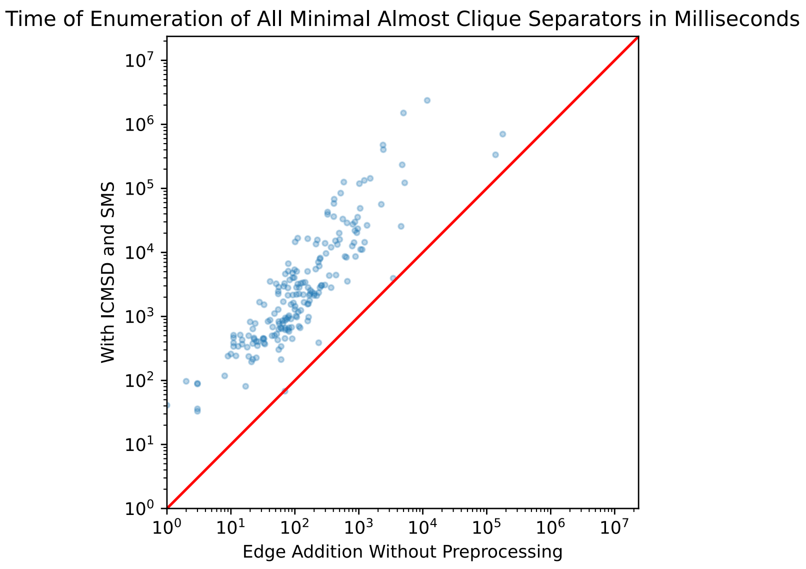

Figure 5.2: Comparison of the methods of enumeration of all minimal between the use of ICMSD and SMS or enumeration of edge addition without preprocessing.

minimal meta separators by ”mlb”, the lower bound obtained by all the minimal almost clique separators by ”alb” and the lowerbound obtained by a maximal set of pairwise parallel set of minimal almost clique separators by ”plb”. The various lowerbounds per graph are compiled toghether with their treewidth on Table 5.4.

As we can see from Table 5.4 minimal clique separator already give in 35 out of 200 cases the best lowerbound and further enumeration of minimal separators did not increase the lowerbound. Only on 8 instances out of 200, minimal meta separators gave better result than the enumeration of all minimal almost clique separators and the enumeration of pairwise parallel minimal almost clique separators. All almost clique separators give a better lower bound than a pairwise parallel minimal almost clique separators only once in the graph 74. Pairwise parallel minimal almost clique separators gave overall 26 better lower bound than minimal meta separators and all almost clique minimal separators.

From the result from Table 5.4 using a set of pairwise parallel minimal almost clique separators seems to be nearly dominating the lowerbound obtained by the set of all almost clique minimal separators.

gr	clb	mlb	alb	plb	tw	gr	clb	mlb	alb	plb	tw	gr	clb	mlb	alb	plb	tw	gr	clb	mlb	alb	plb	tw
001	3	3	4	4	10	051	2	4	3	3	10	101	60	60	60	60	540	151	3	3	3	3	12
002	14	14	14	14	49	052	3	4	9	9	9	102	21	21	26	26	54	152	5	5	6	6	12
003	0	0	6	6	44	053	2	3	3	3	9	103	2	3	3	4	10	153	25	25	25	25	47
004	57	57	57	57	486	054	2	3	3	3	9	104	61	61	61	61	540	154	20	20	20	20	48
005	2	3	3	3	7	055	7	7	8	8	18	105	61	61	61	61	540	155	25	25	25	25	47
006	2	3	3	3	7	056	2	3	3	3	34	106	5	6	5	5	70	156	3	3	4	4	11
007	5	6	6	6	12	057	9	9	9	9	117	107	3	3	4	4	12	157	2	3	3	3	9
008	3	3	4	4	10	058	3	3	3	4	10	108	4	5	5	6	11	158	3	3	4	4	11
009	2	3	3	3	7	059	4	4	6	6	10	109	2	3	3	3	7	159	6	6	8	8	18
010	2	3	3	3	9	060	2	2	3	3	10	110	2	3	3	3	8	160	3	4	8	8	12
011	3	3	4	4	9	061	2	6	5	6	22	111	2	3	3	4	9	161	5	5	6	6	12
012	3	4	4	4	12	062	6	6	8	8	18	112	2	3	3	4	10	162	2	3	3	3	9
013	0	0	0	0	29	063	2	3	3	4	34	113	0	3	4	5	14	163	2	3	3	4	10
014	5	7	8	8	18	064	2	3	3	3	7	114	2	3	3	3	10	164	4	5	5	5	14
015	6	6	6	6	15	065	0	0	0	0	25	115	903	903	903	903	908	165	4	5	5	5	14
016	2	3	3	3	8	066	4	4	5	5	15	116	2	2	3	4	9	166	2	4	3	3	10
017	2	4	3	4	9	067	2	2	3	4	10	117	3	4	4	4	13	167	2	2	3	3	10
018	2	2	3	3	9	068	3	3	4	4	8	118	0	0	0	0	54	168	4	5	5	5	14
019	2	3	4	4	11	069	2	3	3	4	9	119	0	5	5	5	23	169	5	7	6	5	22
020	0	6	6	6	20	070	0	0	0	0	8	120	2	3	3	3	9	170	2	3	3	3	9
021	2	3	3	4	9	071	2	3	3	3	9	121	2	3	4	4	34	171	4	5	5	5	14
022	3	3	10	10	16	072	2	3	3	3	9	122	26	26	26	26	76	172	0	0	0	0	32
023	4	5	5	5	8	073	2	3	3	4	7	123	2	3	3	3	35	173	2	2	3	4	10
024	3	3	10	10	16	074	24	24	26	24	47	124	2	2	3	4	10	174	24	24	24	24	24
025	0	6	6	6	20	075	3	4	4	7	8	125	0	2	2	2	70	175	4	4	8	8	17
026	2	2	3	3	9	076	5	8	8	8	17	126	2	3	3	3	9	176	2	3	3	4	10
027	2	2	3	3	11	077	2	3	4	4	10	127	4	5	5	5	10	177	4	5	5	5	14
028	2	3	3	3	9	078	2	3	3	3	9	128	17	17	28	28	28	178	2	3	3	3	10
029	2	3	3	3	9	079	20	20	20	20	42	129	5	5	6	6	14	179	2	3	3	3	10
030	2	3	3	3	7	080	2	3	3	4	9	130	6	6	9	9	19	180	2	2	2	2	70
031	2	3	3	3	8	081	4	5	5	5	6	131	6	6	8	8	18	181	0	3	6	7	18
032	3	3	5	5	11	082	3	3	6	6	16	132	7	7	8	8	18	182	2	2	3	4	10
033	2	2	3	3	7	083	2	3	3	3	10	133	3	3	4	4	11	183	2	3	3	4	11
034	2	2	3	3	11	084	0	0	0	0	70	134	2	3	3	3	8	184	2	3	3	3	10
035	4	4	5	5	14	085	2	3	3	3	8	135	82	82	82	82	87	185	4	5	5	5	14
036	42	42	42	42	119	086	2	3	3	3	31	136	2	3	3	3	34	186	2	3	3	3	10
037	2	2	3	3	10	087	25	25	25	25	47	137	6	6	8	8	19	187	2	3	3	3	10
038	15	15	26	26	26	088	25	25	25	25	47	138	16	16	26	26	27	188	16	16	26	26	27
039	0	0	0	0	32	089	2	4	3	3	9	139	2	3	3	3	9	189	5	7	5	5	70
040	2	3	3	3	9	090	2	3	4	4	11	140	2	3	3	4	10	190	5	5	6	6	15
041	2	2	3	3	9	091	2	3	3	3	9	141	2	3	3	3	34	191	4	4	5	5	15
042	2	4	3	4	9	092	24	24	24	24	53	142	2	2	4	4	10	192	0	0	0	0	29
043	2	3	3	3	9	093	2	3	3	3	7	143	2	3	4	5	35	193	3	3	4	4	10
044	3	4	4	4	6	094	2	3	3	4	11	144	2	3	3	3	10	194	3	3	4	4	11
045	2	3	3	3	7	095	3	3	4	4	11	145	0	0	0	0	12	195	2	3	3	3	10
046	2	4	3	3	9	096	2	3	9	9	9	146	4	4	5	5	12	196	2	3	3	4	11
047	5	7	5	5	21	097	20	20	20	20	48	147	0	3	5	6	16	197	5	5	6	6	15
048	6	6	6	6	15	098	2	3	3	3	9	148	5	5	6	6	12	198	83	83	83	83	87
049	7	7	7	7	13	099	2	3	3	3	7	149	5	5	6	6	12	199	2	3	3	3	9
050	17	17	28	28	28	100	3	3	4	4	12	150	10	10	10	10	117	200	4	4	5	5	16

Table 5.1: Lower bounds of the treewidth obtained by the cardinality of the maximum minimal clique separator (clb), maximum minimal meta separator (mlb), maximum minimal almost clique separator (alb) and the maximum separator from a pairwise parallel almost-clique separators (plb). This data is coupled with the treewidth of the graph (tw). In bold are the graphs where mlb is the largest.

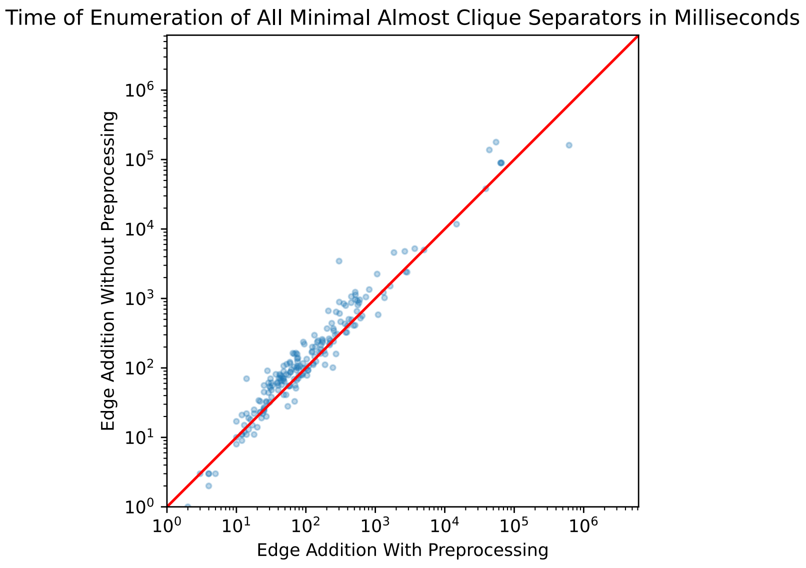

Figure 5.3: Difference in runtime of the edge addition enumeration method with and without ICMSD preprocessing.

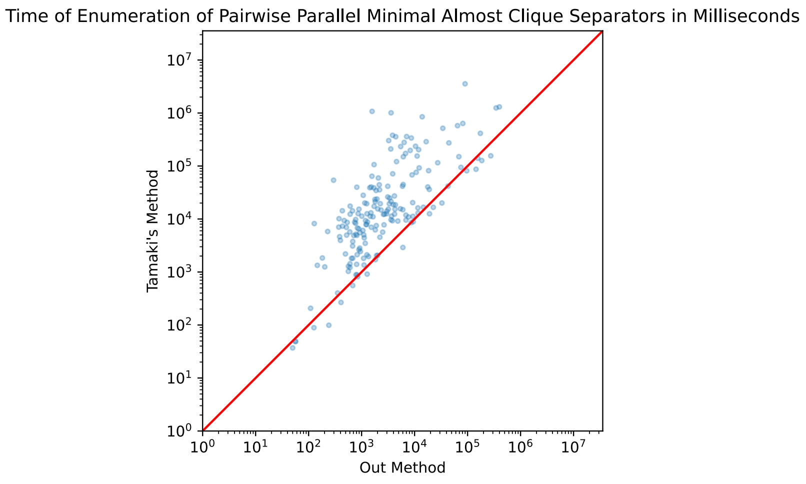

Figure 5.4: Comparison of the methods between Tamaki's method and our for the enumeration of pairwise parallel minimal almost clique separators.

Chapter 6

Conclusion

This paper gave a theoretical and historical overview of minimal triangulation and minimal separators and we further contributed to what is currently state of the art in the field.

Our contributions in the field of minimal triangulation are:

- The introduction of the idea of transformation and the demonstration of four transformations, which are, inclusion minimal clique separator decomposition, saturate minimal separator, edge addition, minimal meta separator decomposition.
- The introduction of the concept of minimal meta separator and the proof that its presence in a graph can be used to tighten its treewidth and minimum fill-in lower bounds.
- The proof that any set of pairwise parallel minimal separators of a graph has at most $n-1$ elements in Lemma 2.9.1.
- The generalization of the Lemma by Bodlaender in [9] for the identification of inclusion minimal almost-clique separators to also encompass minimal almost-clique separators in Lemma 4.1.1. We also have proven that given a set of minimal almost-clique separators found with such method, they are all pairwise parallel in Lemma 4.1.2.
- The construction of an algorithm for listing a maximal set of pairwise parallel almost-clique separators which leverages the concept of transformation and runs faster than the state of the art algorithms proposed by [37].
- The proof that such algorithm runs at time $O(n^3m)$ which is a novel result to the best of our knowledge and not presented both in [9] and [37].
- We showed that Edge Addition can be used for the enumeration of minimal almost-clique separators with a competitive runtime.

We believe that the concept of transformation is a theoretically unexplored aspect of minimal separators which can be explored in many manners.

We could not check whether if minimal meta separator decomposition could bring if any speed-ups in the enumeration of minimal separators and we think this is a interesting topic for future research. Furthermore it would be interesting to see whether multiple minimal meta separator decomposition can be applied to the same graph effectively. Also it would be interesting to see whether it is possible to improve the lowerbound of the minimum fill-in problem provided by the minimal meta separator as it is currently a very weak lowerbound.

We used Edge Addition for the enumeration of minimal almost-clique separators which yield a very fast runtime. The enumeration of minimal separators is functionally similar to Takata's recurrance[34] and both enumeration by Edge Addition and Takata's recurrance can enumerate minimal separators which any subset satisfies a specific condition and therefore allow pruning during enumeration. It would be interesting to see a comparison between the two algorithms for the enumeration of minimal almost clique separators, minimal separators of size at most k and minimal meta separators.

A transformation that we did not discuss is edge removal. While in the proof of Theorem 3.3 we studied the cases of what would happen to a minimal separator if an edge would have been removed, we could not exhaustively tackle all the cases. But we believe edge removal can be used to efficiently enumerate minimal separators for graph with low minimum fill-in, as there are only few edge removals to go from a chordal graph, which minimal separators are easy to enumerate on, to the target graph.

Another direction future research can go is in the enumeration of minimal separators in highly parallel systems for larger graphs. Some of our transformation allowed the decomposition to multiple atoms which can be processed independently allowing enumeration of minimal separators in parallel systems.

And another direction, even if very ambitious, is whether there is a connection between graph transformation a graph minor theory[33]. We demonstrated historical and theoretical connection between minimal separators and treewidth, it is well established the connection between treewidth and graph minor theory, and the idea of edge addition transformation resembles very much edge deletion in graph minor theory.

Bibliography

[1] Andreas Parra Asensio and Andreas Parra Asensio. Structural and algorithmic aspects of chordal graph embeddings. 1996. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.7640.

[2] Anne Berry and Romain Pogorelcnik. A simple algorithm to generate the minimal separators and the maximal cliques of a chordal graph. 111(11):508–511. ISSN 0020-0190. doi: 10.1016/j.ipl.2011.02.013. URL https://www.sciencedirect.com/science/article/pii/S0020019011000536.

[3] Anne Berry, Jean R. S. Blair, Pinar Heggernes, and Barry W. Peyton. Maximum Cardinality Search for Computing Minimal Triangulations of Graphs. 39(4):287–298, . ISSN 1432-0541. doi: 10.1007/s00453-004-1084-3. URL https://doi.org/10.1007/s00453-004-1084-3.

[4] Anne Berry, Pinar Heggernes, and Geneviève Simonet. The Minimum Degree Heuristic and the Minimal Triangulation Process. In Hans L. Bodlaender, editor, Graph-Theoretic Concepts in Computer Science, Lecture Notes in Computer Science, pages 58–70. Springer, . ISBN 9783540398905. doi: 10.1007/978-3-540-39890-5_6.

[5] Anne Berry, Romain Pogorelcnik, and Geneviève Simonet. An Introduction to Clique Minimal Separator Decomposition. 3(2):197–215, . ISSN 1999-4893. doi: 10.3390/a3020197. URL https://www.mdpi.com/1999-4893/3/2/197.

[6] Jean R. S. Blair, Pinar Heggernes, and Jan Arne Telle. A practical algorithm for making filled graphs minimal. 250(1):125–141. ISSN 0304-3975. doi: 10.1016/S0304-3975(99)00126-7. URL https://www.sciencedirect.com/science/article/pii/S0304397599001267.

[7] Hans L. Bodlaender. Discovering treewidth. Lecture Notes in Computer Science, 3381:1–16, 2005. ISSN 03029743. doi: 10.1007/978-3-540-30577-4_1. URL https://link-springer-com.tudelft.idm.oclc.org/chapter/10.1007/978-3-540-30577-4_1.

[8] Hans L. Bodlaender and Fedor V. Fomin. Tree decompositions with small cost. 145(2):143–154. ISSN 0166-218X. doi: 10.1016/j.dam.2004.01.008. URL https://www.sciencedirect.com/science/article/pii/S0166218X04002410.

[9] Hans L. Bodlaender and Arie M.C.A. Koster. Safe separators for treewidth. Discrete Mathematics, 306:337–350, 2 2006. ISSN 0012365X. doi: 10.1016/j.disc.2005.12.017.

[10] Magnus Bordewich, Katharina T. Huber, and Charles Semple. Identifying phylogenetic trees. 300(1):30–43. ISSN 0012-365X. doi: 10.1016/j.disc.2005.06.015. URL https://www.sciencedirect.com/science/article/pii/S0012365X05003389.

[11] Vincent Bouchitté and Ioan Todinca. Treewidth and minimum fill-in: Grouping the minimal separators. SIAM Journal on Computing, 31:212–232, 7 2001. ISSN 00975397. doi: 10.1137/S0097539799359683. URL https://epubs.siam.org/page/terms.

[12] Nofar Carmeli, Batya Kenig, and Benny Kimelfeld. Efficiently Enumerating Minimal Triangulations. In Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS '17, pages 273–287. Association for Computing Machinery. ISBN 9781450341981. doi: 10.1145/3034786.3056109. URL https://doi-org.tudelft.idm.oclc.org/10.1145/3034786.3056109.

[13] L. Sunil Chandran. A Linear Time Algorithm for Enumerating All the Minimum and Minimal Separators of a Chordal Graph. In Jie Wang, editor, Computing and Combinatorics, Lecture Notes in Computer Science, pages 308–317. Springer. ISBN 9783540446798. doi: 10.1007/3-540-44679-6_34.

[14] Holger Dell, Christian Komusiewicz, Nimrod Talmon, and Mathias Weller. The pace 2017 parameterized algorithms and computational experiments challenge: The second iteration. DROPS-IDN/8558, 89, 2 2018. ISSN 18688969. doi: 10.4230/LIPICS.IPEC.2017.30.

[15] G. A. Dirac. On rigid circuit graphs. Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg, 25:71–76, 1961. ISSN 18658784. doi: 10.1007/BF02992776. URL https://link.springer.com/article/10.1007/BF02992776.

[16] Yon Dourisboure and Cyril Gavoille. Tree-decompositions with bags of small diameter. 307(16):2008–2029. ISSN 0012-365X. doi: 10.1016/j.disc.2005.12.060. URL https://www.sciencedirect.com/science/article/pii/S0012365X06007692.

[17] D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. Pacific Journal of Mathematics, 15:835–855, 1965. ISSN 00308730. doi: 10.2140/pjm.1965.15.835.

[18] Masanobu Furuse and Koichi Yamazaki. A revisit of the scheme for computing treewidth and minimum fill-in. 531:66–76. ISSN 0304-3975. doi: 10.1016/j.tcs.2014.03.013. URL https://www.sciencedirect.com/science/article/pii/S0304397514002047.

[19] Alan George and Joseph W.H. Liu. The Evolution of the Minimum Degree Ordering Algorithm. 31(1):1–19. ISSN 0036-1445. doi: 10.1137/1031001. URL https://epubs.siam.org/doi/abs/10.1137/1031001.

[20] Georg Gottlob, Zoltán Miklós, and Thomas Schwentick. Generalized hypertree decompositions: NP-hardness and tractable variants. 56(6):30:1–30:32. ISSN 0004-5411. doi: 10.1145/1568318.1568320. URL https://doi.org/10.1145/1568318.1568320.

[21] Batya Kenig. Enumerating Minimal Separators in Ranked Order. URL http://arxiv.org/abs/2111.07647. arXiv: 2111.07647.

[22] Tuukka Korhonen. Listing Small Minimal Separators of a Graph. . URL http://arxiv.org/abs/2012.09153. arXiv: 2012.09153.

[23] Tuukka Korhonen. PACE Solver Description: SMS. In Yixin Cao and Marcin Pilipczuk, editors, 15th International Symposium on Parameterized and Exact Computation (IPEC 2020), volume 180 of Leibniz International Proceedings in Informatics (LIPIcs), pages 30:1–30:4. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, . ISBN 9783959771726. doi: 10.4230/LIPIcs.IPEC.2020.30. URL https://drops.dagstuhl.de/opus/volltexte/2020/13333.

[24] Tuukka Korhonen. Finding optimal triangulations parameterized by edge clique cover. arXiv preprint arXiv:1912.10989, 2019.

[25] Tuukka Korhonen. Finding optimal tree decompositions, 2020. URL http://www.cs.helsinki.fi/.

[26] Hanns-Georg Leimer. Optimal decomposition by clique separators. 113(1):99–123. ISSN 0012-365X. doi: 10.1016/0012-365X(93)90510-Z. URL https://www.sciencedirect.com/science/article/pii/0012365X9390510Z.

[27] Dániel Marx. Parameterized graph separation problems. 351(3):394–406. ISSN 0304-3975. doi: 10.1016/j.tcs.2005.10.007. URL https://www.sciencedirect.com/science/article/pii/S0304397505006328.

[28] Lukas Moll, Siamak Tazari, and Marc Thurley. Computing hypergraph width measures exactly. 112(6):238–242. ISSN 0020-0190. doi: 10.1016/j.ipl.2011.12.002. URL https://www.sciencedirect.com/science/article/pii/S0020019011003243.

[29] Hiromu Otsuka, Tomoki Kuida, Takumi Sato, and Hisao Tamaki. Experimental evaluation of greedy treewidth heuristics on huge graphs.

[30] Andreas Parra and Petra Scheffler. Characterizations and algorithmic applications of chordal graph embeddings. 79(1):171–188. ISSN 0166-218X. doi: 10.1016/S0166-218X(97)00041-3. URL https://www.sciencedirect.com/science/article/pii/S0166218X97000413.

[31] Andreas Parra, Petra Schefl,] Er, and T U Berlin. How to use the minimal separators of a graph for its chordal triangulation. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 944:123–134, 1995. ISSN 16113349. doi: 10.1007/3-540-60084-1_68. URL https://link.springer.com/chapter/10.1007/3-540-60084-1_68.

[32] S. Parter. The use of linear graphs in gauss elimination. 3(2):119–130. doi: 10.1137/1003021. URL https://doi.org/10.1137/1003021.

[33] Neil Robertson and P. D. Seymour. Graph minors. ii. algorithmic aspects of tree-width. Journal of Algorithms, 7:309–322, 9 1986. ISSN 01966774. doi: 10.1016/0196-6774(86)90023-4.

[34] Ken Takata. Space-optimal, backtracking algorithms to list the minimal vertex separators of a graph. 158(15):1660–1667. ISSN 0166-218X. doi: 10.1016/j.dam.2010.05.013. URL https://www.sciencedirect.com/science/article/pii/S0166218X10001824.

[35] Hisao Tamaki. Computing treewidth via exact and heuristic lists of minimal separators. volume 11544 LNCS, pages 219–236. Springer, 6 2019. ISBN 9783030340285. doi: 10.1007/978-3-030-34029-2_15. URL https://doi.org/10.1007/978-3-030-34029-2_15.

[36] Hisao Tamaki. Positive-instance driven dynamic programming for treewidth. Journal of Combinatorial Optimization, 37:1283–1311, 5 2019. ISSN 15732886. doi: 10.1007/S10878-018-0353-Z/TABLES/6. URL https://link.springer.com/article/10.1007/s10878-018-0353-z.

[37] Hisao Tamaki. A heuristic for listing almost-clique minimal separators of a graph. 2021.

[38] Robert E. Tarjan. Decomposition by clique separators. Discrete Mathematics, 55:221–232, 7 1985. ISSN 0012-365X. doi: 10.1016/0012-365X(85)90051-2.

[39] Robert E. Tarjan and Mihalis Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. 13(3):566–579. doi: 10.1137/0213035. URL https://doi.org/10.1137/0213035.

[40] Zijian Xu and Vorapong Suppakitpaisarn. On the Size of Minimal Separators for Treedepth Decomposition. Technical report, arXiv. URL http://arxiv.org/abs/2008.09822. arXiv:2008.09822 [cs] type: article.

[41] Zijian Xu, Dejun Mao, and Vorapong Suppakitpaisarn. PACE Solver Description: Computing Exact Treedepth via Minimal Separators. In Yixin Cao and Marcin Pilipczuk, editors, 15th International Symposium on Parameterized and Exact Computation (IPEC 2020), volume 180 of Leibniz International Proceedings in Informatics (LIPIcs), pages 31:1–31:4. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. ISBN 9783959771726. doi: 10.4230/LIPIcs.IPEC.2020.31. URL https://drops.dagstuhl.de/opus/volltexte/2020/13334.

[42] Mihalis Yannakakis. Computing the minimum fill-in is np-complete. 2(1):77–79. doi: 10.1137/0602010. URL https://doi.org/10.1137/0602010.