Report project M1:
Graphs Combinatorial Optimization:
Application to Structural Analysis

C. Bégué

May 20, 2020

Contents

1 Introduction

The importance of efficient resolutions of satisfiability issues (SAT problem) and manipulations of Boolean functions are generalized in different computer science fields, in design and verification. Thibault et al. introduced a new tool to explicit the hidden dependency structure of Boolean functions related to structural analysis [4]. They introduced a new canonical representation of Boolean function named Reduced Block Triangular Form. This representation allows to rewrite a conjunction into another one which incidence matrix is block triangular and ensures some additional properties allowing its efficient manipulation.

This representation allows to check the satisfiability of a Boolean function in conjunctive form easily. But, the cost of this reduction is still exponential. To model the reduction into Reduced Block Triangular Form, they introduced the primal graph of a Boolean function. Each variables of the function is represented by a vertex in the primal graph. Two vertexes are connected if and only if they appear in a common clause of the function.

They define the Weighted Adjacency Propagation (WAP) problem as an optimization problem on graph. We find a good elimination order for Reduced Block Triangular Form by solving WAP on the primal graph of the considered conjunction.

During this year, we studied the WAP problem and how to solve it efficiently. I worked on WAP separators. A WAP separator allows to decompose a graph into sub-graphs such that an optimal solution is composed with optimal solutions over the sub-graphs. I also worked on the implementation of WAP resolution.

The remainder of this report is organized as follows: Section 2 formalizes the WAP problem. Section 3 presents the theoretical aspect of my work on WAP. Finally, Section 4 presents an extension of Thibault's WAP-solver written in OCaml. Section 5 concludes this report.

2 Weighted Adacency Propagation problem

2.1 An optimization problem on graphs

In this section, I formally introduce the WAP problem and a similar graph problem, H-WAP.

2.1.1 Introduction of WAP

In this section, we present formally the WAP problem and relation with another graph problem : Treewidth [2].

We only focus on undirected graphs.

Definition 2.1.1. *Neighborhood*

Let $G := (V, E)$ be a undirected graph with V a set of vertexes and $E \subseteq V^2$ a set of edges. Let $X \subseteq V$.

The neighborhood of X in the graph G, denoted $N_G(X)$ or $N(G, X)$, is the subset of V at distance at most one to X :

$$\begin{aligned} N(G,X) = N_G(X) &:= \{y \in V \mid d_G(y, X) \leqslant 1\} \\ &:= \{y \in V \mid \exists x \in X,\ (x, y) \in E\} \cup X \end{aligned}$$

The strict neighborhood of X is $N_G(X) \backslash X$.

Remark To simplify the notation, if the context is not ambiguous, we assimilate a set and its cardinal. For example, $2^{|N_G(X)|}$ is denoted $2^{N_G(X)}$.

Definition 2.1.2. *Suppression*

Let $G := (V, E)$ be an undirected graph, $X \subseteq V$. We denote $G' := (V', E')$ with $V' := V \backslash X$ and $E' := \left(E \bigcup N_G(X)^2\right) \bigcap V'^2$ the graph we get after the suppression of X. In other words, the suppression of X occurs in two steps:

- *the set X is removed from V;*
- *the strict neighborhood of X in G becomes a clique in G'.*

G' is also denoted $S(G, X)$. This notation is extended by induction : $S(G, X_0, \ldots, X_k) = S(S(G, X_0), X_1, \ldots, X_k)$.

Let $\{\mathcal{X}_0, \ldots, \mathcal{X}_n\}$ be a partition of V. We denote $\mathcal{X} := (\mathcal{X}_0, \ldots, \mathcal{X}_n)$ a suppressing sequence of G: we suppress $\mathcal{X}_0$, then $\mathcal{X}_1$ etc. We denote $G^{(i)}$ the graph after the suppression of $\mathcal{X}_{i-1}$. By definition, $G^{(0)} = G$ and for $i \in [\![1, n+1]\!]$, $G^{(i)} = S(G, \mathcal{X}_0, \ldots, \mathcal{X}_{i-1})$. Also, $G^{(n+1)} = S(G, \mathcal{X})$.

We denote the set of suppressing sequences $\mathcal{SS}(G)$. In particular, if $\mathcal{X} \in \mathcal{SS}(G)$, $S(G, \mathcal{X}) = \varnothing$.

Definition 2.1.3. *[Suppressing cost]*

Let $X \in V$ and $\mathcal{X} := (\mathcal{X}_0, \ldots, \mathcal{X}_n) \in \mathcal{SS}(G)$ a suppressing sequence.

The suppressing cost of $X \in V$ from the graph G is $2^{N_G(X)} = 2^{N(G,X)}$.

The suppressing cost of $\mathcal{X}$, denoted $\mathtt{S}^{\mathtt{c}}(G, \mathcal{X})$, is :

$$\mathtt{S}^{\mathtt{c}}(G, \mathcal{X}) = \sum_{i=0}^{n} 2^{N(G^{(i)}, \mathcal{X}_i)}$$

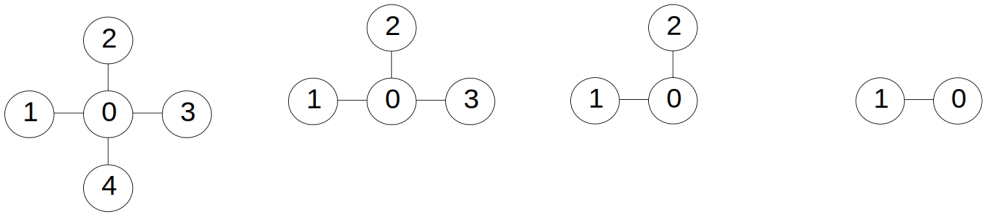
Definition 2.1.4. *[WAP problem]*

Let $G = (V, E)$ be a graph. We want to delete G by minimizing the suppressing cost. So, we define the WAP value of G as the minimal cost across all its suppressing sequences :

$$\mathtt{S}^{\mathtt{c}}(G) = \min_{\mathcal{X} \in \mathcal{SS}(G)} \mathtt{S}^{\mathtt{c}}(G, \mathcal{X})$$

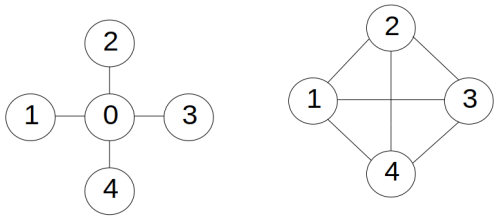
2.1.2 Examples

Let us see some examples of WAP.



(a) Graph G_1 (b) Suppression of vertex 4: cost 2^2 (c) Suppression of vertex 3: cost 2^2 (d) Suppression of vertex 2: cost 2^2

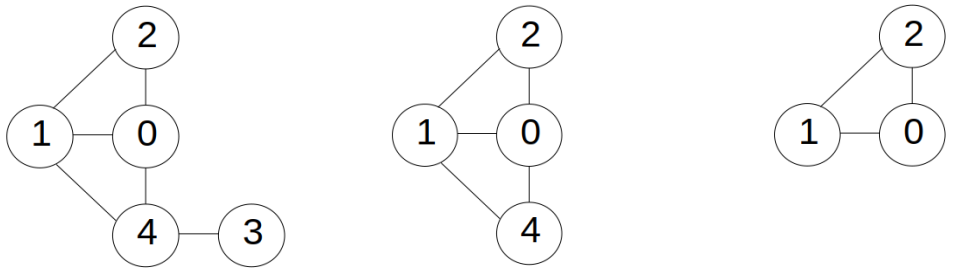
Figure 1: Example 1 for G_1: cost $2^2 + 2^2 + 2^2 + 2^2 = 2^4$



(a) Graph G_1 (b) Suppression of vertex 0: cost 2^5

Figure 2: Example 2 for G_1: cost $2^5 + 2^4$

The graph G_1 is suppressed by 2 ways in figures 1 and 2. The first one cost $2^2 + 2^3 + 2^4$ and the other $2^5 + 2^4$. Indeed, Figure 1d and Figure 2b, the graph is a clique and its suppression costs 2^k with k the size of the clique. So, the first option is better than the second. Actually, Figure 1 is the optimal solution for G_1, $S^c(G_1) = 2^2 + 2^3 + 2^4$.



(a) Graph G_2 (b) Suppression of vertex 3: cost 2^2 (c) Suppression of vertex 4: cost 2^3

Figure 3: WAP on G_2: cost $2^2 + 2^3 + 2^3 = 2^2 + 2^4$

In the figure 3, the suppression of the graph G_2 costs $2^2 + 2^4$. It is the optimal cost.

2.1.3 Relation with Treewidth

WAP seems strongly related to the minimal width tree-decomposition problem[2], usually known as treewidth. Treewidth is a well-studied NP-complete[1] problem, arising in various areas of computer science, and commonly used in parameterized complexity.

Definition 2.1.5. *[Tree Decomposition, Treewidth]*
Let $G := (V, E)$ be a graph. Let introduce $\Omega := (\mathcal{Y}, \mathcal{T})$ with:

- *$\mathcal{Y} := (\mathcal{Y}_0, \dots, \mathcal{Y}_k)$ sequence of k subsets of V;*
- *$\mathcal{T}$ a tree over the elements of $\mathcal{Y}$.*

Ω is called a tree-decomposition of G.
The width of a tree-decomposition is the size of the largest $\mathcal{Y}_i$ minus 1. The treewidth of G is defined as the minimal width among all its tree-decompositions.

To show a relation between WAP and Treewidth, we reduced WAP to Treewidth.
So, let $G := (V, E)$ be a graph and $\mathcal{X} := (\mathcal{X}_0, \dots, \mathcal{X}_k) \in \mathtt{SS}(G)$.
We define $\mathcal{Y} := (\mathcal{Y}_0, \dots, \mathcal{Y}_k)$ by $\forall i \in [\![1, k]\!]$, $\mathcal{Y}_i := N(G^{(i)}, X_i)$. The component number i of $\mathcal{Y}$ is the neighborhood of X_i when it will be suppress. With the good relation over the $\mathcal{Y}_i$, we can define a tree $\mathcal{T}$ such that $(\mathcal{Y}, \mathcal{T})$ is a tree-decomposition.
It proves a relation between WAP and Treewidth. Knowing that, we can assume WAP is NP-complete but it is not proved yet.

2.2 Quotienting True-Twins

While studying the WAP problem, J. Thibault demonstrated that, if two vertexes have the same neighborhood, we can suppress them together without increasing the suppressing cost. These vertexes are called true twins. So, we can only focus on sequences where each component is a clique of true twins while suppressing.

Definition 2.2.1. *[True-twins]*
Let $G := (V, E)$ be a graph, $u, v \in V$.
u and v are true-twins, denoted $u \; R_{TT} \; v$, if and only if $N_G(u) = N_G(v)$.

R_{TT} is clearly an equivalence relation. Clearly, if we suppress a set of vertexes $X \subseteq V$ in graph G, for each vertex $(u, v) \in V \backslash X$, if $u \; R_{TT} \; v$ in G, we get $u \; R_{TT} \; v$ in G', the graph after the suppression of X.

So, from a graph $G = (V, E)$, we can define a quotient graph G/R_{TT}. G/R_{TT} respects the WAP suppression.

From this quotient graph $G/R_{TT} = (V_{TT}, E_{TT})$, we define an undirected weighted graph $G_H = (V_H, E_H, \omega)$ with :

- $V_H = V_{TT} = \{X \subseteq V \mid \forall u, v \in X, u \; R_{TT} \; v \text{ in } G\}$;
- $E_H = E_{TT} = \{(X, Y) \in V_{TT}^2 \mid Y \subseteq N_G(X) \text{ and } X \subseteq N_G(Y)\}$;
- $\forall X \in V_H, \; \omega(X) = |X|$.

The cost in time and space of this transformation is polynomial in $|V|$ (we just need to identify true twins, it costs at most $O(|V|^3)$).

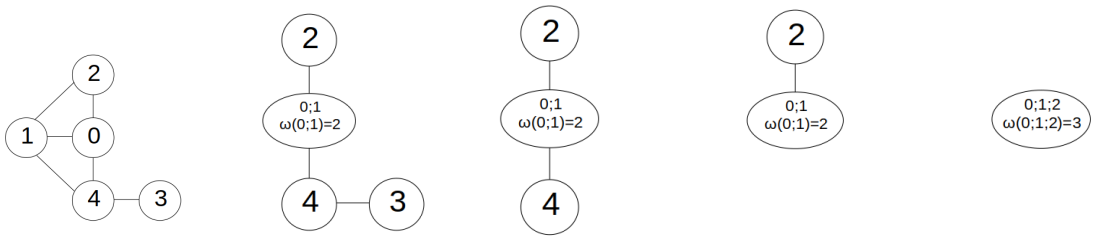
This graph G_H is called an H-graph. Deleting a vertex in G_H is the same than deleting the original vertexes in G. So, because we only focus on sequences where each component is a clique of true twins, we suppress the vertexes one by one in H-WAP.

The suppression of a vertex is similar than in WAP, but after a suppression, some vertexes can became true twins and then merge. The suppressing cost of $u \in V_H$ is $2^{N(G_H, u)}$, where $N(G_H, u) = \sum_{v \in N(G_H, u)} \omega(v)$.

Like WAP, the H-WAP problem consists on deleting a H-graph by minimizing the suppressing cost. Moreover, the optimal cost for a graph G in WAP is the same than the optimal cost of its transformation in H-WAP. We have reduced the WAP problem to H-WAP.

This reduction from WAP to H-WAP reduces the number of sequences to $n!$. A suppressing sequence is an order on the vertexes instead of a partition.

As an example, let us apply the reduction to H-WAP in the graph G_2 in figure 4. In H-WAP, the suppression of G_2 costs $2^2 + 2^4$. We notice that the optimal cost and suppressing sequences are similar in WAP and H-WAP. It is always the case.



(a) Graph G_2 (b) Merging vertex 0 and 1 (c) Suppression of vertex 3: cost 2^2 (d) Suppression of vertex 4: cost 2^3 (e) Merging vertexes $\{0; 1\}$ and 2

Figure 4: H-WAP on G_2: cost $2^2 + 2^3 + 2^3 = 2^2 + 2^4$

In the rest of this report, we will work on H-WAP.

3 Theoretical Contribution

I mostly worked on theoretical aspects of H-WAP problem.

I first wanted to demonstrate vertex separator are WAP separators. My approaches lead me to demonstrate some interesting lemma.

3.1 Some properties on H-WAP

During the project, I demonstrated some properties on H-WAP, especially to estimate the cost of suppression of some graphs.

First of all, let us introduce the notion of non-suppressible vertex.

Definition 3.1.1. *[Non-suppressible vertex]*
Let $G := (V, E, \omega)$ be a graph, $v \notin V$ a vertex of weight $|v|$. Let us define $G\underline{v} := (V \cup \{v\}, E \cup E', \omega')$ with:

- $E' \subseteq V \times \{v\}$;
- $\omega' : u \in V \cup \{v\} \mapsto \begin{cases} \omega(u) & \text{if } u \in V \\ |v| & \text{if } u = v \end{cases}$

The vertex v is non-suppressible if and only if we cannot supress or merge it.
The graph $G\underline{v}$ is a graph with a non-suppressible vertex. The weight of v is taken into account when we suppress a vertex in its neighborhood.

Imposing non-suppressible vertexes into a graph consists on imposing the final component of any suppressing sequence.

We have some properties about the suppressing cost of a graph with a non-suppressible vertex.

Lemma 3.1.1. *Let $G := (V, E, \omega)$ be a H-graph, $v \notin V$ a vertex of weight $|v|$. We consider the graph $G\underline{v} = (V \cup \{v\}, E \cup E', \omega')$ with v a non-suppressible vertex connected with some vertexes of G.*
The suppressing cost of $G\underline{v}$ (the cost of the suppression of all vertexes except v) is at most $2^{\mathbf{v}} \times \mathtt{s}^{\mathtt{c}}(G)$ More formally:

$$\mathtt{s}^{\mathtt{c}}(G\underline{v}) \leqslant 2^{\mathbf{v}} \times \mathtt{s}^{\mathtt{c}}(G)$$

If all vertexes of G are in the neighborhood of v, this is an equality. More formally:

$$\forall x \in G,\ v \in N_{G\underline{v}}(x) \ \Rightarrow\ \mathtt{s}^{\mathtt{c}}(G\underline{v}) = 2^{\mathbf{v}} \times \mathtt{s}^{\mathtt{c}}(G)$$

We can compute optimal sequences on G and apply it in $G\underline{v}$. The weight of v is counted at most once by suppression in G, and exactly once if all vertexes of G are connected with v.
This lemma is proved in appendix (Section 6.1).

One may show that every clique of a graph is in the neighborhood of a least one vertex when it is suppress, my proof is available in Section 6.3.

Lemma 3.1.2. *Let $G = (V, E)$ be a graph, $\mathcal{X} \in \mathcal{SS}(G)$, for each clique K of G, K is in the neighborhood of a component of $\mathcal{X}$ when it is deleted.*

$$\forall \mathcal{X} \in \mathcal{SS}(G), K \in \mathtt{clique}(G), \exists i, K \subseteq N_{G_i}(\mathcal{X}_i)\ .$$

This theorem supports our conjecture that cliques play a central role in the study of optimal sequences.

3.2 Vertex separators

Definition 3.2.1. *[Separator, clique separator]*
Let $G := (V, E, \omega)$ be an undirected graph, $X \in V$ is a separator if and only if $G_{|X} := (V \backslash X, E \backslash X \times V, \omega)$ is not connected.
In particular, X is a clique separator if and only if X is a separator and X is a clique.

After solving various examples of H-WAP problems, we conjectured that clique separators are H-WAP separators. This means a clique separator allows to decompose a graph in sub-graphs such that an optimal solution is composed with optimal solutions of sub-graphs.

If this hypothesis is true, we could split the graph into sub-graphs connected by cliques allowing a divide and conquer strategy.

My first task was to demonstrate or refute this hypothesis for H-WAP problems.

First, we focus on vertex separators, a simple case of clique separators.

Let $G = (V, E, \omega)$ a H-graph, $v \in V$ a vertex separator of G and $\mathcal{X} = (\mathcal{X}_0, \ldots, \mathcal{X}_k)$ a suppressing sequence of G. We denote $G_0, \ldots, G_n$ the connected components of $G_{|v}$, like in figure 5. The rectangles represent a set of vertexes of G and the circle, a single vertex. Edges mean v is connected to at least one vertex of G_i.

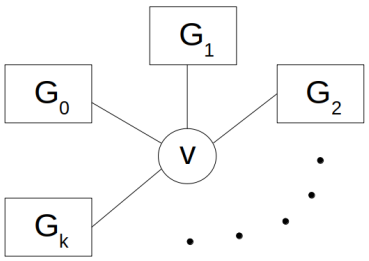


Figure 5: Vertex separator in general case

We want to demonstrate that v is a H-WAP separator. It means v is suppressed only when all the G_i except one are empty. To do so, we execute $\mathcal{X}$ until we have to suppress v. We notice that v can not merge because it is the only vertex connected to each sub-graph.

We notice that all suppressions occurring before the suppression of v are not important and so we can ignore them. So, we can suppose $\mathcal{X}_0 = \{v\}$. This kind of argument is very useful for WAP and H-WAP analysis. Actually, the graphs are dynamic and it could be very difficult to reason with too long sequences.

So, we assume $\mathcal{X}_0 = \{v\}$ and $n > 0$. We also assume $n = 1$ (in that case, there are only 2 connected components into $G \backslash \{v\}$) without a loss a generality.

Now the problem is formally described, I overestimate the cost of the suppression of G if we suppress v while it is no longer a separator and underestimate $S^c(G, \mathcal{X})$, meaning we suppress v first. We denote the cost of overestimation C_o and the cost of underestimation C_u.

In order to demonstrate the hypothesis, we want to find $C_o < C_u$. It will demonstrate that $\mathcal{X}$ is not an optimal sequence and so v is an H-WAP separator.

Overestimating the suppressing cost of G if v is the last vertex suppressed is quite easy. One way to suppress G is to consider v as a non-suppressible vertex. We compute an optimal sequence on $G'\underline{v}$ (with $G' := G \backslash \{v\}$). v is deleted with the last vertex of G', because they merged.

The suppressing cost of this solution is at most $2^v \times (S^c(G_0) + S^c(G_1))$, according to lemma 3.1.1.

So, $C_o = 2^v \times S^c(G_0) \times S^c(G_1)$.

The difficulty came with the second part of the proof. The suppression of v is described in the figure 6. In this figure, the rectangle represents a set of vertexes and the circles a single vertex. For the notation, we have :

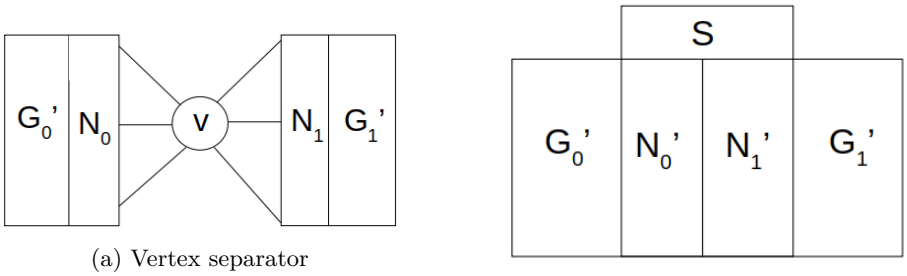
- in figure 6a, $G_0 = G'_0 \cup N_0$, $G_1 = G'_1 \cup N_1$ and $N_G(v) = N_0 \cup N_1$;
- in figure 6b, $N'_0 \cup N'_1 \cup S$ is a clique with each vertex of N'_0 connected with G'_0 and each vertex of N'_1 connected with G'_1. The vertexes of S are not connected with neither G'_0 nor G'_1.

The clique S is named a simplicial. The study of simplicials is as difficult as the clique separator problem. I will not develop it in this report. So, I assume in the rest of this section that $S = \varnothing$.

In the case when $G'_0 = G'_1 = \varnothing$, v is a vertex separator. This specific case is named *Generalized Star Pattern*. It is a corollary of Lemma 3.1.1 when all the graph is in the neighborhood of a single vertex v. We prove it formally in appendix (Section 6.2).

Let us analysis the case where $G'_0 \neq \varnothing$ or $G'_1 \neq \varnothing$.
If both are not empty, $N'_0 \cup N'_1$ is a clique separator, it is a specific case of clique separator problem and it is more complex than the vertex separator case. We can not continue the reasoning without extra hypotheses on the structure of G.
If $G'_0 = \varnothing$ and $G'_1 \neq \varnothing$, N_0 is a simplicial. As we said before, the simplicial problme is as difficult as clique separator problem.

So, in H-WAP, the question of a very simple case of clique separator is already too complicated and several difficult sub-cases appear.



(a) Vertex separator

(b) Suppression of v

Figure 6: Suppression of a vertex separator

During this project, I collaborated with J. Duron who worked on K-WAP, another way of reducing the search space by noticing that many sequences induce that same decomposition. Actually, although in a possibly different order the same component appear and have the exact same neighborhood upon suppression. Proving the clique separator conjecture within K-WAP is rather simple (although not trivial). This fact leads to the abandonment of the proof effort of the conjecture in bare H-WAP. Interestingly enough, the reduction to K-WAP was based on a set of Lemma called "characterization of 2-H-sequences" (the characterization of graphs which can be suppressed in H-WAP by a sequence of length 2). My work allows to greatly simplify the proof and exposes a clearer structure of this kind of graph.

4 Implementation Effort

After studying the theoretical aspects of WAP and H-WAP problem, I participated in the implementation of a WAP solver. The program is coded in OCaml.
I implemented the detection of dominated vertexes and vertex separators.

Definition 4.0.1. *[Dominated vertex]*
Let $G := (V, E, \omega)$ be an undirected graph, $u, v \in V$. u dominates v if and only if $N_G(v) \subseteq N_G(u)$.

Domination relations are used to simplify the WAP solving. It is a generalization of the *Generalized Star Pattern*. The detection of dominated vertexes is based on inclusion tests between the neighborhood of vertexes. For a graph $G := (V, E)$, its complexity is $O(|V|)$ in space and $O(|V| \times D^2)$ in time where D is the maximal degree of a vertex in G. I implemented different units tests for this algorithm, and it passes them.

For the detection of vertex separators in $G := (V, E)$, I implemented two algorithms:

- the first is very naive: to test if a vertex v is separator, I delete it for G and test if $G\backslash\{v\}$ is connected with a Deep-First search (DFS) algorithm. This first approach has a complexity of $O(|V| + |E|)$ in space and $O(|V| \times (|V| + |E|))$ in time.
- the second algorithm is based on the propriety "if v is the root of a tree generated by a DFS, v is separator if and only if v has a least two children". In the case v is not the root of a such tree, the condition is "exists u a descendant of v such that it does not exist any backward edges from u or one of its descendants to an ancestor of v". This algorithm is described in a paper of R. Tarjan [3]. It costs $O(|V| + |E|)$ in space and time.

Currently, these two programs have not been tested yet but I am planning implement unit tests soon and compare them.

5 Conclusion

During this year, I mostly worked on the proof of vertex separators in H-WAP. My work allowed to gain a good understanding of various basic properties. J. Duron used it in his reduction of H-WAP to K-WAP. Finally, I extended Thibault's WAP-solver with several algorithms meant to detect dominant vertexes and vertex separators. Future works include assessing their respective performance, implementing an algorithm to detected clique separators, and demonstrating that the current upper bound on Generalized Star Pattern is actually the optimal suppressing cost.

References

[1] S. Arnborg and A. Proskurowski. Linear time algorithms for np-hard problems restricted to partial k-trees. Discrete Applied Mathematics, 23(1):11 – 24, 1989.

[2] Neil Robertson and P. D Seymour. Graph minors. II. Algorithmic aspects of tree-width. Journal of Algorithms, 7(3):309–322, September 1986.

[3] Robert E. Tarjan. An efficient parallel biconnectivity algorithm. SIAM J. Computing, 1985.

[4] J. Thibault, A. Blanché, and K. Ghorbal. Leveraging structural analysis for conciserepresentation of boolean formulas. SAT2018.

6 Appendix

6.1 Proof on non-suppressible vertex

Lemma 6.1.1. *Let $G := (V, E, \omega)$ be a H-graph, $v \notin V$ a vertex of weight $|v|$. We consider the graph $G\underline{v} = (V \cup \{v\}, E \cup E', \omega')$ with v a non-suppressible vertex connected with some vertexes of G.*

If all vertexes of G are in the neighborhood of v, the suppressing cost of $G\underline{v}$ (the cost of the suppression of all vertexes except v) is $2^{\mathbf{v}} \times \mathtt{s}^{\mathtt{c}}(G)$

More formally:

$$\forall x \in G,\ v \in N_{G\underline{v}}(x) \ \Rightarrow\ \mathtt{s}^{\mathtt{c}}(G\underline{v}) = 2^{\mathbf{v}} \times \mathtt{s}^{\mathtt{c}}(G)$$

Proof.

- $\mathtt{s}^{\mathtt{c}}(G\underline{v}) \leqslant 2^{\mathbf{v}} \times \mathtt{s}^{\mathtt{c}}(G)$: To get a suppressing sequence for $G\underline{v}$, we get an optimal sequence for G, denoted $S = \{X_1, \ldots, X_n\}$.
  For all $i \in [\![1, n]\!]$, $C_{(G\cup v)_{i-1}}(X_i) = 2^{\mathbf{v}} \times C_{G_{i-1}}(X_i)$ because $v \in N_{G\cup v}(X_i)$ and $v \notin N_G(X_i)$.
  Thus, $S(G\underline{v}, X_1, \ldots, X_n) = 2^{\mathbf{v}} \times \mathtt{s}^{\mathtt{c}}(G)$.
  So, $\mathtt{s}^{\mathtt{c}}(G\underline{v}) \leqslant 2^{\mathbf{v}} \times \mathtt{s}^{\mathtt{c}}(G)$.

- $\mathtt{s}^{\mathtt{c}}(G\underline{v}) \geqslant 2^{\mathbf{v}} \times \mathtt{s}^{\mathtt{c}}(G)$: To get a suppressing sequence for G, we get an optimal sequence for $G\underline{v}$, denoted $S = \{X_1, \ldots, X_n\}$.
  For all $i \in [\![1, n]\!]$, $C_{G_{i-1}(X_i)} = \frac{1}{2^{\mathbf{v}}} \times C_{(G\cup v)_{i-1}}(X_i)$ because $v \in N_{G\cup v}(X_i)$ and $v \notin N_G(X_i)$.
  Thus, $S(G, X_1, \ldots, X_n) = \frac{1}{2^{\mathbf{v}}} \times \mathtt{s}^{\mathtt{c}}(G\underline{v})$.
  So, $\mathtt{s}^{\mathtt{c}}(G\underline{v}) \geqslant 2^{\mathbf{v}} \times \mathtt{s}^{\mathtt{c}}(G)$.

So, we get $\mathtt{s}^{\mathtt{c}}(G\underline{v}) = 2^{\mathbf{v}} \times \mathtt{s}^{\mathtt{c}}(G)$. □

Lemma 6.1.2. *Let $G := (V, E, \omega)$ be a H-graph, $v \notin V$ a vertex of weight $|v|$. We consider the graph $G\underline{v} = (V \cup \{v\}, E \cup E', \omega')$ with v a non-suppressible vertex connected with some vertexes of G.*

The suppressing cos of $G\underline{v}$ is at most $2^{\mathbf{v}} \times \mathtt{s}^{\mathtt{c}}(G)$. More formally:

$$\mathtt{s}^{\mathtt{c}}(G\underline{v}) \leqslant 2^{\mathbf{v}} \times \mathtt{s}^{\mathtt{c}}(G)$$

Proof. To get a suppressing sequence for $G\underline{v}$, we get an optimal sequence for G, denoted $S = \{X_1, \ldots, X_n\}$.
For all $i \in [\![1, n]\!]$, $C_{(G\cup v)_{i-1}}(X_i) \leqslant 2^{\mathbf{v}} \times C_{G_{i-1}}(X_i)$ because $v \notin N_G(X_i)$.
Thus, $S(G\underline{v}, X_1, \ldots, X_n) \leqslant 2^{\mathbf{v}} \times \mathtt{s}^{\mathtt{c}}(G)$ and $\mathtt{s}^{\mathtt{c}}(G\underline{v}) \leqslant 2^{\mathbf{v}} \times \mathtt{s}^{\mathtt{c}}(G)$. □

6.2 Generalized Star Pattern

Lemma 6.2.1 (Generalized Star Pattern)**.** *Let $G = (V, E, \omega)$ be a H-graph and $v \in V$ a separator of G. Such as $G = N_G(v)$ (figure **??**). v_1 and v_2 sub graphs of G separated by v.*

So v is a H-WAP separator.

Proof. We compute the suppressing cost of G depending on when v is removed.

First case: if v is removed while v_1 and v_2 are not empty

The suppressing cost of v is $2^{\mathbf{v_1}+\mathbf{v_2}+\mathbf{v}}$. After suppressing v, $v_1 \cup v_2$ is a clique, remove in one step for a cost of $2^{\mathbf{v_1}+\mathbf{v_2}}$.

The final cost of this solution is: $\boxed{\mathcal{C}_1 = 2^{\mathbf{v_1}+\mathbf{v_2}+\mathbf{v}} + 2^{\mathbf{v_1}+\mathbf{v_2}}}$

$2^{\mathbf{v_1}+\mathbf{v_2}} \geqslant 2^{\mathbf{v_1}} + 2^{\mathbf{v_2}}$, because v_1 and v_2 are not empty, so $\mathcal{C}_1 \geqslant (2^{\mathbf{v}} + 1) \times (2^{\mathbf{v_1}} + 2^{\mathbf{v_2}})$

Second case: if v is deleted while v_1 or v_2 is empty

A non-optimal solution is to remove v at the end.

We considered the two sub graphs $v_1\underline{v}$ and $v_2\underline{v}$.

The final cost of this solution is: $\mathcal{C}_2 = \mathtt{s}^{\mathtt{c}}(v_1\underline{v}) + \mathtt{s}^{\mathtt{c}}(v_2\underline{v}) = 2^{\mathbf{v}} \times (\mathtt{s}^{\mathtt{c}}(v_1) + \mathtt{s}^{\mathtt{c}}(v_2))$ according to propriety 6.1.1.

However, $\mathtt{S}^{\mathtt{c}}(v_i) \leqslant 2^{\mathbf{v_i}}$ so $\mathcal{C}_2 \leqslant 2^{\mathbf{v}} \times (2^{\mathbf{v_1}} + 2^{\mathbf{v_2}})$

So we get:

$$\mathbf{v} \times (2^{\mathbf{v_1}} + 2^{\mathbf{v_2}}) \leqslant (2^{\mathbf{v}} + 1) \times (2^{\mathbf{v_1}} + 2^{\mathbf{v_2}}) \Rightarrow \mathcal{C}_2 \leqslant \mathcal{C}_1$$

However:

$$\begin{array}{rrcl} & 2^{\mathbf{v}} \times (2^{\mathbf{v_1}} + 2^{\mathbf{v_2}}) & \leqslant & (2^{\mathbf{v}} + 1) \times (2^{\mathbf{v_1}} + 2^{\mathbf{v_2}}) \\ \Leftrightarrow & 2^{\mathbf{v}} & \leqslant & 2^{\mathbf{v}} + 1 \\ \Leftrightarrow & 0 & \leqslant & 1 \end{array}$$

So v is a WAP-separator.

□

remark We can easily generalize this property for more sub-graphs $v_1, ..., v_n$ by induction. So, we can suppose each sub-graphs v_i are connected.

remark we can define an elimination scheme of $G = (V, E\omega)$ from this property:

- identify a vertex v such as $N_G(v) = V$
- split $G - v$ into $v_1, ..., v_n$, connected sub-graphs of G
- apply H-WAP algorithm on each sub-graphs, each suppression costs $2^{\mathbf{v}} \times \mathtt{S}^{\mathtt{c}}(v_i)$
- suppress v with the latest vertex of v_n

Supposing H-WAP can be reduce to K-WAP, this is an optimal algorithm. Thanks to theorem **??**, each node is a part of an optimal terminal component.

6.3 Clique neighborhood

Lemma 6.3.1. *Let $G = (V, E)$ be a graph, $\mathcal{X} \in \mathcal{SS}(G)$, for each clique K of G, K is in the neighborhood of a component of $\mathcal{X}$ when it is deleted.*

$$\forall \mathcal{X} \in \mathcal{SS}(G), K \in \mathtt{clique}(G), \exists i, K \subseteq N_{G_i}(\mathcal{X}_i) \ .$$

Proof. Let G a H-graph, $\mathcal{X} = \{\mathcal{X}_0, \ldots, \mathcal{X}_n\} \in SS(G)$, $K \in \mathtt{clique}(G)$.

Define $i^* = \underset{i}{argmin}\{\mathcal{X}_i \cap K \neq \varnothing\}$. By induction on $i \in [\![1, n]\!]$, $N_G(\mathcal{X}_i) \backslash \{\mathcal{X}_1, \ldots, \mathcal{X}_{i-1}\} \subseteq N_{G^{(i)}}(\mathcal{X}_i)$.

$\forall i \in [\![1, n]\!]$

So, we get $K \subseteq N_G(\mathcal{X}_{i*}) \backslash \{\mathcal{X}_1, \ldots, \mathcal{X}_{i*-1}\}$, by definition of i^* because we do not delete any node of K and no edges have been deleted.

So, using equation induction, we conclude $\underline{K \subseteq N_{G^{(i*)}}(\mathcal{X}_i^*)}$ □