UNIVERSITÉ LIBRE DE BRUXELLES

Faculté des Sciences
Département de Mathématique
Année Académique 2020/2021

Thèse présentée en vue de l'obtention du grade de Docteur en Sciences

Two Approaches to Approximation Algorithms for Vertex Deletion Problems

by

Matthew Drescher

Jury de thèse:
Prof. Jean Cardinal (Université libre de Bruxelles, Président)
Prof. Gwenaël Joret (Université libre de Bruxelles, Secrétaire)
Prof. Matthias Mnich (Hamburg University of Technology)
Prof. Karthekeyan Chandrasekaran (University of Illinois, Urbana-Champaign)
Prof. Samuel Fiorini (Université libre de Bruxelles, Promoteur)

Contents

Summary

Weighted graphs are an important mathematical structure which can model many types of algorithmic problems. Problems that are intractable in the general case, often become efficiently solvable when restricted to smaller *graph classes* $\mathcal{G}$. This motivates the question of how to best "modify" a graph so that it belongs to a simpler graph class.

In this thesis, the graph classes we consider are defined by forbidding a finite set of graphs $\mathcal{F}$ (called the *obstruction set*) as induced subgraphs. We study the following problem through the lens of approximation algorithms: Given a weighted graph (G, w), and a graph class $\mathcal{G}$ characterized by obstruction set $\mathcal{F}$, find the smallest weight set of vertices that can be removed from G so that the remaining graph belongs to $\mathcal{G}$ ($\mathcal{G}$-Vertex Deletion).

This problem can be modelled as a vertex cover problem on k-uniform hypergraphs (Ek-Vertex Cover) for which there is an easy k-approximation algorithm. It is known that Ek-Vertex Cover is hard to approximate within $k - \epsilon$ for any constant $\epsilon > 0$ unless the Unique Games Conjecture is false [46] or P=NP. Therefore the non-trivial approximation factors for a specific $\mathcal{G}$-Vertex Deletion problem is within the interval $[2, k)$.

We develop two different methods which give improved approximation algorithms for several instances of $\mathcal{G}$-Vertex Deletion. The first is a refinement of the *local ratio* approach given in Fiorini, Joret, Schaudt [28]. The second is rooted in linear and integer programming. We use the Sherali-Adams hierarchy [60] to "automatically" strengthen the natural LP relaxation of Ek-Vertex Cover. We then employ the standard technique of *iterative rounding*.

We use these tools to develop approximation algorithms for some well-known $\mathcal{G}$-Vertex Deletion problems: Deletion to Claw-free, Feedback Vertex Set problem in Tournaments, Cluster Vertex Deletion, Split Vertex Deletion. In these instances our methods do better than the trivial factor and we obtain the following results which are our main contributions:

- We give a 2-approximation algorithm for Cluster Vertex Deletion [3]. This tight result matches the hardness lower bound.
- We obtain a new deterministic 7/3-approximation algorithm for Feedback Vertex Set in Tournaments [2]. This result is based on the LP given by just one round of Sherali-Adams.
- We find a new, simpler deterministic $(2 + \epsilon)$-approximation algorithm for Split Vertex Deletion [26].
- We give a 3-approximation algorithm for Claw-Free Vertex Deletion in triangle-free graphs. In the case of general graphs we prove that it is UGC-hard to obtain an approximation ratio of $3 - \epsilon$.

Finally we list some open questions. Some of these naturally arose, others are motivated by computational experiments. The thesis concludes with some remarks contrasting these two distinct approaches to designing approximation algorithms.

Résumé

Les graphes pondérés sont des structures mathématiques importantes pouvant modeler de nombreux types de problèmes algorithmiques. Un grand nombre de ces problèmes, malgré leur difficulté dans le cas général, deviennent efficacement solubles une fois leurs entrées restreintes à une *classe de graphes* particulière $\mathcal{G}$.

Dans cette thèse, nous considérons des classes de graphes définies en interdisant un ensemble fini de sous-graphes induits $\mathcal{F}$ (que nous nommons *ensemble d'obstruction*). Nous étudions le problème suivant via des algorithmes d'approximation: étant donné un graphe pondéré (G, w), une classe de graphes $\mathcal{G}$ caractérisée par un ensemble d'obstruction $\mathcal{F}$, trouver l'ensemble de sommets de poids minimum tels qu'une fois retirés de G le graphe restant appartiennent à $\mathcal{G}$ ($\mathcal{G}$-Vertex Deletion).

Nous pouvons modéliser ce problème comme un problème de couverture de sommets sur des hypergraphes k-uniformes (Ek-Vertex Cover) pour lesquels il existe un algorithme de k-approximation simple. Cependant, Ek-Vertex Cover est un problème difficile à mieux approximer: il est difficile d'obtenir un ratio d'approximation meilleur que $k - \epsilon$ pour toute constante $\epsilon > 0$ à moins que la *Unique Games Conjecture (UGC)* soit fausse [46] ou que P=NP. Dès lors, les ratios d'approximation non triviaux pour un problème de $\mathcal{G}$-Vertex Deletion en particulier se situent dans l'intervalle $[2, k)$.

Nous développons deux différentes méthodes qui donnent de meilleurs algorithmes d'approximation pour plusieurs instances de $\mathcal{G}$-Vertex Deletion. Le premier est une amélioration de l'approche par *ratio local* donnée par Fiorini, Joret, et Schaudt [28]. La deuxième exploite l'optimisation linéaire et l'optimisation entière. Nous appliquons la hiérarchie de Sherali-Adams [60] pour renforcer "automatiquement" la relaxation linéaire naturelle de Ek-Vertex Cover. Nous employons ensuite la technique standard d'arrondi itératif (*iterative rounding*).

Nous utilisons ces outils pour développer des algorithmes d'approximation pour certains problèmes $\mathcal{G}$-Vertex Deletion bien connus: Deletion to Claw-free, Feedback Vertex Set problem in Tournaments, Cluster Vertex Deletion, et Split Vertex Deletion. Notre contribution principale consiste en de nouveaux algorithmes avec ratios d'approximation non-triviaux:

- Nous donnons un algorithme avec ratio d'approximation de 2 pour Cluster Vertex Deletion [3]. Ce ratio est le meilleur possible étant donné l'existence d'une borne inférieure correspondante.

- Nous obtenons un nouvel algorithme déterministe avec ratio d'approximation de $7/3$ pour Feedback Vertex Set in Tournaments [2]. Ce résultat est basé sur le programme d'optimisation linéaire obtenu après une itération de Sherali-Adams.
- Nous trouvons un nouvel algorithme avec ratio d'approximation de $2+\epsilon$ pour Split Vertex Deletion, plus simple que les algorithmes existants [26].
- Nous donnons un algorithme avec ratio d'approximation de 3 pour Claw-Free Vertex Deletion dans des graphes sans triangles. Dans le cas de graphes généraux, nous prouvons qu'obtenir un meilleur ratio est UGC-difficile.

Enfin, nous mentionnons quelques questions ouvertes, certaines d'entre-elles survenant naturellement lors de notre recherche, d'autres motivées par des expérimentations calculatoires. Nous concluons cette thèse par quelques remarques contrastant les deux méthodes algorithmiques abordées dans notre thèse.

Acknowledgements

I would like to thank the European Research Council[1] for supporting this research.

I am incredibly thankful for the patient support of my supervisor Samuel Fiorini. His ability, creativity and authentic passion for mathematics has been truly inspiring. I am truly grateful to have even had the chance to work with him.

Huge thanks to my friend and collaborator Tony Huynh for encouraging me to apply to ULB, and supporting my journey back into mathematics. Thanks for the endless patient explanations, and making life so absurdly hilarious, and fun. Thank you to Aurélien Ooms, Marco Macchia, Manuel F. Aprile, Carole Muller for making life in Brussels so pleasant socially and engaging mathematically.

I would like to thank Tim Roughgarden for being a sounding board and encouraging me to actually make the leap back into academics.

Thanks to my friends Ion Oancea, Andy Dreyfus, Charlie Wyman for keeping me focused and at least somewhat sane, and to my family for their constant love and support.

I would like to dedicate this thesis to my wonderful Tenaya, whose loving support and patient enthusiasm for this endeavor has been galaxies beyond anything I could have expected.

[1]ERC Consolidator Grant 615640-ForEFront.

Part 1

Background and Context

CHAPTER 1

Introduction

Graphs classes are of fundamental importance in Graph Theory, in particular for algorithms. One prominent example is the class of *bipartite* graphs. Knowing that a graph is bipartite gives access to significantly more tools than for general graphs. For instance, the stable set problem on a bipartite graph can be solved efficiently using network flow techniques [40]. In contrast, for general graphs the problem can be hopelessly difficult. Another major example is the class of graphs with *treewidth* at most t, where t is any constant. In this class, a host of problems can be solved in polynomial time, as formalized by Courcelle's theorem [22]. A last example is the class of *perfect graphs*, that generalize bipartite graphs. A premier result in algorithmic Graph Theory is that computing the chromatic number, and finding a minimum coloring can be done in polynomial time whenever the input graph is perfect [35].

Typically, graph classes are characterized by an *obstruction set* which is embodied by a family $\mathcal{F}$ of non-isomorphic graphs. For a given $\mathcal{F}$, we distinguish three ways to define a class of graphs: either by taking *induced subgraphs*, or *subgraphs*, or *minors*. An induced subgraph of a graph G is any graph that can be obtained by deleting vertices. A subgraph of G is any graph that can be obtained from G by deleting vertices and edges. A minor of G is any graph that can be obtained from a subgraph of G by contracting edges. *$\mathcal{F}$-free* graphs are the graphs that do not have any induced subgraph isomorphic to a graph of $\mathcal{F}$. The classes of *$\mathcal{F}$-subgraph-free* graphs and *$\mathcal{F}$-minor-free* graphs are defined similarly.

For instance, bipartite graphs are those which do not contain any odd cycle as a subgraph, as proved by Kőnig in 1916 [47]. By the Graph Minor Theorem of Robertson and Seymour [58], graphs with treewidth at most t can be characterized by a finite list of forbidden minors (although the complete list is known only up to $t = 3$). The Strong Perfect Graph Theorem of Chudnovsky and Seymour characterizes perfect graphs as those which forbid odd cycles of length 5 or more and their complements as induced subgraphs [18]. For each of these three graph classes, there is a well-defined obstruction set $\mathcal{F}$.

Beyond those three examples, there are many more graph classes. The website `www.graphclasses.org` [24] is the outcome of an effort toward a taxonomy of graphs classes, the relations between them, and their algorithmic properties.

The four graph classes that are of particular interest in the context of this thesis are the following ones, see Fig. 1.1:

(1) *Cluster graphs* are graphs whose connected components are all complete. They are also the $\mathcal{F}$-free graphs for $\mathcal{F} = \{P_3\}$, where P_3 is the path on three vertices.

(2) *Transitive tournaments* are oriented complete graphs with no cycles. They are also the $\mathcal{F}$-free tournaments for $\mathcal{F} = \{\triangle\}$, where $\triangle$ is the directed cycle on three vertices.
(3) *Claw-free graphs* are the $\mathcal{F}$-free graphs for $\mathcal{F} = \{K_{1,3}\}$, where $K_{1,3}$ is the complete bipartite graph with one part of size one, and the other of size three.
(4) *Split graphs* are the graphs whose vertices can be partitioned into a clique and a stable set. They are also the $\mathcal{F}$-free graphs for $\mathcal{F} = \{C_4, C_5, \overline{C}_4\}$, where C_k denotes the cycle graph on $k \geq 3$ vertices and $\overline{C}_4 = 2K_2$ is the complement of C_4.

Tournaments are directed graphs, not graphs, and so technically the set of transitive tournaments is not a graph class, but we treat it as such anyway. All the classes we consider here are *hereditary*: for every graph $G \in \mathcal{G}$, and every vertex subset X, the graph $G - X$ obtained by deleting the vertices of X is also in $\mathcal{G}$.

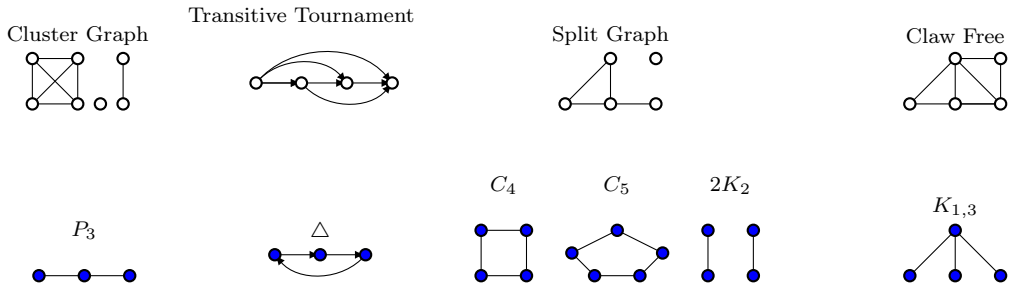


FIGURE 1.1. Top: The graph classes $\mathcal{G}$ considered in this thesis. Bottom: Their corresponding obstruction sets $\mathcal{F}$.

Let $\mathcal{G}$ be a graph class. Given a graph G that is not in $\mathcal{G}$, a natural question is: *how close is G to being in $\mathcal{G}$?* For instance, how cheaply can we delete vertices to make a given graph G bipartite? This motivates the following general problem.

PROBLEM 1 ($\mathcal{G}$-VERTEX DELETION). *Given graph G with weights on the vertices, find a minimum weight set of vertices X such that $G - X$ belongs to $\mathcal{G}$.*

We call a solution X to $\mathcal{G}$-VERTEX DELETION a *hitting set*.

Other types of graph modification problems exist as for instance $\mathcal{G}$-EDGE DELETION where edges are deleted rather than vertices, and $\mathcal{G}$-EDIT where edges can be added and deleted. We do not study these problems.

This thesis studies $\mathcal{G}$-VERTEX DELETION when $\mathcal{G}$ is one of the four classes listed above, through the lens of *Approximation Algorithms* [51, 65, 67]. Another popular viewpoint is that of *Fixed Parameter Tractable Algorithms* [23].

Perhaps the simplest example of a $\mathcal{G}$-VERTEX DELETION problem is VERTEX COVER. We use this historically significant problem as a vehicle to motivate and illustrate ideas clearly. Given a weighted graph (G, w), the goal is to find a minimum weight set of vertices to remove so that the remaining graph is "edgeless". VERTEX COVER coincides with $\mathcal{G}$-VERTEX DELETION, where $\mathcal{G}$ is the class of graphs with no edges, or equivalently $\mathcal{G}$ is the class of $\mathcal{F}$-free graphs with $\mathcal{F} := \{K_2\}$. VERTEX COVER is one of Karp's 21 NP-hard problems [45]. So

unless $P = NP$, finding a solution in polynomial time, in the hardest instances, means settling for something less than optimal. This situation naturally poses the question: *how close to optimal can we get*? Approximation algorithms provide a metric for answering this question and comparing different NP-hard problems [67].

Of course, all of the $\mathcal{G}$-Vertex Deletion problems we consider here are minimization problems. We use the standard definition for approximation algorithm, see for example [65]. For positive *approximation factor* $\alpha > 1$, an α-*approximation algorithm* for a minimization problem returns a feasible solution that must:

(1) Be computed in polynomial time in the size of the input;
(2) Weigh no more than a factor of α of the weight of an optimal solution.

Unweighted Vertex Cover has a simple greedy 2-approximation algorithm: take any inclusion-wise maximal matching and include all the matched vertices in the solution. By maximality of the matching, the vertices selected form a feasible solution to Vertex Cover. Moreover, any optimal solution must contain at least half of these vertices. Therefore this algorithm provides a 2-approximation. In the weighted case, as we will see in Chapter 3, it is not much harder to achieve a ratio of 2.

Thus, it is perhaps surprising that after considerable efforts, the current best approximation factor for Vertex Cover is $2 - o(1)$, given independently by Halperin [39], and Karakostas [44]. In [25], Dinur and, Safra showed that it is NP-hard to approximate to within a factor of 1.3606. In [46], Khot and Regev showed that Vertex Cover is NP-hard to approximate within any constant factor better than 2 assuming the *Unique Games Conjecture* (UGC).

This is evidence to suggest that $\mathcal{G}$-Vertex Deletion problems might be quite difficult. In the case of $\mathcal{F}$-subgraph-free classes $\mathcal{G}$, there is indeed a strong hardness result by Guruswami and Lee [38]. They show that when $\mathcal{F}$ contains a 2-connected graph of size k, $\mathcal{G}$-Vertex Deletion is NP-hard to approximate within a factor of $k - \epsilon$ for any constant $\epsilon > 0$ assuming the UGC.

In the case of $\mathcal{F}$-minor-free classes $\mathcal{G}$, if $\mathcal{F}$ does not contain a planar graph, we know of no result that gives a constant factor approximation. On the other hand if $\mathcal{F}$ does contain a planar graph, then all graphs of $\mathcal{G}$ have treewidth at most t, for some constant t, and the recent breakthrough of Gupta, Lee, Li, Manurangsi, and Wlodarczyk [36] gives an $O(\log t)$-factor approximation algorithm for $\mathcal{G}$-Vertex Deletion, in the unweighted case. Actually, they show this for all hereditary graph classes $\mathcal{G}$ with bounded treewidth. In the presence of weights, obtaining a constant factor approximation algorithm is a challenging open problem.

Our goal for this work is to design new deterministic approximation algorithms for $\mathcal{G}$-Vertex Deletion with non-trivial approximation ratios, where $\mathcal{G}$ is one of the hereditary graph classes listed above. The state of the art partly motivates this: most of the previous work is concerned with monotone classes (defined by forbidden subgraphs) or minor-closed classes (defined by forbidden minors). To the best of our knowledge, hereditary classes (defined by forbidden induced subgraphs) were much less studied in the literature. Also, for these classes it seems much more difficult to obtain non-trivial results. Here, we chose

to study concrete hereditary classes that are defined by a small number of small graphs.

The following four $\mathcal{G}$-Vertex Deletion problems fit these criteria:

Problem 2 (Claw-Free Vertex Deletion, Claw-VD). *Given weighted graph (G, w) find a minimum weight subset of vertices X such that $G - X$ is Claw-free.*

Problem 3 (Feedback Vertex Set problem in Tournaments, FVST). *Given tournament (T, w) find a minimum weight subset of vertices X such that $T - X$ is a transitive tournament.*

Problem 4 (Cluster Vertex Deletion, CVD). *Given weighted graph (G, w) find a minimum weight subset of vertices X such that $G - X$ is a cluster graph.*

Problem 5 (Split Vertex Deletion, SVD). *Given weighted graph (G, w) find a minimum weight subset of vertices X such that $G - X$ is a split graph.*

There has been previous work on these problems. We list here the latest results for each of the four problems. A more complete review of the literature is given in the corresponding chapter in Part 2 of this thesis.

The Claw-VD restricted to *bipartite graphs* has been studied by Kumar, Mishra, Safina Devi and Saurabh in [50], where a 3-approximation algorithm is given. The case of FVST has been settled in the randomized setting with the 2-approximation algorithm of Lokshtanov, Misra, Mukherjee, Panolan, Philip and Saurabh [54]. The best known deterministic algorithm has an approximation ratio of $7/3$ and is given by Mnich, Williams and Végh [56]. Fiorini, Joret, and Schaudt give a $9/4$-approximation algorithm for CVD [28]. A randomized $(2+\epsilon)$-approximation algorithm was given for SVD by Lokshtanov, Misra, Panolan, Philip and Saurabh in [55].

These four problems all have approximation preserving reductions to Vertex Cover, which implies that they are NP-hard to approximate with a factor better than 2 assuming the UGC and $P \neq NP$. As it turns out they can all be posed as instances of Ek-Vertex Cover. For some positive integer constant k, see Fig. 1.2 for an illustration.

Problem 6 (Ek-Vertex Cover). *Given a k-uniform* hypergraph $\mathcal{H}$ *with weights on its vertices, find a minimum weight set of vertices X that meets all of the edges of $\mathcal{H}$.*

Given a $\mathcal{G}$-Vertex Deletion problem where $\mathcal{G}$ is the class of $\mathcal{F}$-free graphs, such that $\mathcal{F}$ is finite and each graph it contains has at most k vertices, we obtain an equivalent Ek-Vertex Cover problem in the obvious way: for each induced subgraph H of G that is isomorphic to some order-k graph in $\mathcal{F}$, we create an edge in $\mathcal{H}$ whose vertices are those of H. In case H has less than k vertices, we use dummy vertices of large weight to ensure that the corresponding edge of $\mathcal{H}$ has k vertices. There is an easy k-approximation algorithm for Ek-Vertex Cover. Thus Ek-Vertex Cover is a baseline for the problems we study here. Unfortunately, assuming UGC, it is hard to approximate Ek-Vertex Cover within $k - \epsilon$ for any constant $\epsilon > 0$. This result is due to Khot and Regev

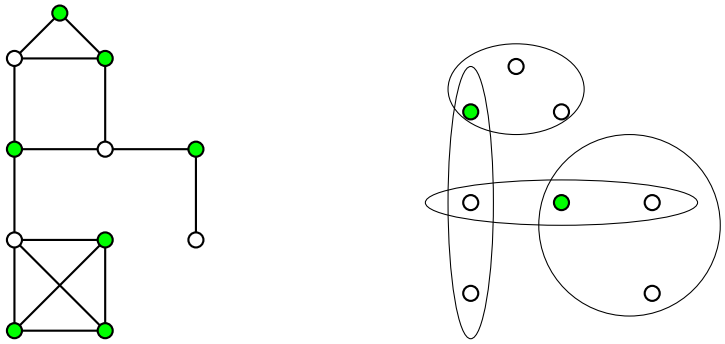


FIGURE 1.2. Comparing VERTEX COVER to E3-VERTEX COVER. On the left we have a simple graph, and show a solution to VERTEX COVER in green. On the right we have a 3-uniform hypergraph (ellipses represent edges) and show a solution to E3-VERTEX COVER in green.

[46]. Given a $\mathcal{G}$-VERTEX DELETION problem, with $\mathcal{G}$ defined by forbidden set $\mathcal{F}$ containing no graphs with more than k vertices, we obtain a k-approximation algorithm by converting it to an instance of EK-VERTEX COVER. Therefore the non-trivial approximation factors for our $\mathcal{G}$-VERTEX DELETION problems are to be within the interval $[2, k)$. In order to achieve such a ratio we need to descend from the general case of EK-VERTEX COVER and design algorithms which can leverage problem-specific properties. In this work we employ two different algorithm design techniques.

The first tool we use is Linear Programming (LP), a classic foundation for designing approximation algorithms [51, 65, 67]. Typically the most natural or naive formulation is not the one that yields the best results. Coming up with clever formulations and analysis often requires deep insight into the problem. We turn to *Sherali-Adams* for a black box method that strengthens LP relaxations for us. On the other hand, it can be difficult to analyze the quality of these strengthened linear programs.

The second tool we use is the *local ratio* method. It is roughly equivalent to the well known primal-dual method. The local ratio method has given the best known results in terms of the problems we study here [26, 28, 54]. The power of this "lighter weight" method is its flexibility. Notice that if there are vertices of zero weight, we can add them to a partial solution, and remove them from the graph. The idea is to find an induced subgraph (location), and a *local* weighting for this location, such that any optimal solution for this subgraph, and weighting must include a "good" fraction of its total weight. We then subtract the local weight function from the input weight function which produces at least one new zero-weight vertex. We then proceed recursively. Thus if we can identify obstruction sets $\mathcal{F}'$ which can be used as locations without exceeding the approximation ratio we want to achieve, we can assume the problem is restricted to an intermediary graph class $\mathcal{G}'$ defined by forbidding $\mathcal{F}'$. In all of the problems we apply this approach to, $\mathcal{G}'$ has useful structural and algorithmic properties which make the problem easier to solve.

We obtain the following results which are our main contributions.

- We give a 2-approximation algorithm for Cluster Vertex Deletion [3]. This tight result matches the hardness lower bound.
- We obtain a new deterministic 7/3-approximation algorithm for Feedback Vertex Set in Tournaments [2]. This result is based on the LP given by just one round of Sherali-Adams.
- We find a new, simpler deterministic $(2+\epsilon)$-approximation algorithm for Split Vertex Deletion [26].
- We give a 3-approximation algorithm for Claw-Free Vertex Deletion in triangle-free graphs. In the case of general graphs we prove that it is UGC-hard to obtain an approximation ratio of $3-\epsilon$.

We conclude this introduction with an outline of the thesis.

In Part 1 we set the stage. We first give some preliminary definitions and notation in Chapter 2. In Chapter 3 we introduce two generalized frameworks for designing approximation algorithms for the problems we introduced here. We give some lemmas and observations which apply to many problems and help us later on.

Part 2 features the main new results contained in this work. We utilize the general design techniques of Part 1 to attack our four $\mathcal{G}$-Vertex Deletion problems. The work in this part is based on several joint papers: [26] which has been accepted to the **European Conference on Combinatorics, Graph Theory and Applications 2021**, [3] which has appeared in the **21st International Conference on Integer Programming and Combinatorial Optimization**, and [2] which is submitted.

In Part 3, we share some experimental results, which we found surprising and illuminating but were unable to exploit theoretically. We then offer our concluding remarks.

Problem	Determinism	Factor	Run Time	LP size	
FVST	Randomized	2	$O(n^{34})$		[54]
FVST	Deterministic	2	$n^{O(\lg n)}$		[54]
FVST	Deterministic	7/3		$O(n^7)$	[56]
FVST	Deterministic	7/3		$O(n^4)$	Chapter 5
CVD	Deterministic	9/4	$O(n^5)$		[28]
CVD	Deterministic	2	$O(n^4)$		Chapter 6
SVD	Randomized	$2+\epsilon$	$O(n^{f(\epsilon)})$		[55]
SVD	Deterministic	$2+\epsilon$	$O(n^{g(\epsilon)})$		Chapter 7

Table 1. The current landscape of approximation factors, determinism, and run time or LP size.

CHAPTER 2

Preliminaries

Here we introduce and define the basic concepts and notations used throughout this work.

2.1. Sets and Weight functions

We adopt the usual notation from set theory. If S is a set, then 2^S denotes the set of all subsets of S. We use $|S|$ for the cardinality of S. The symbol $\varnothing$ represents the empty set. We use the notation $[n] := \{1, 2, \ldots, n\}$ for finite sets. We sometimes use the arithmetic operators to add or remove elements to a set, and let $A + a := A \cup \{a\}$, $A - a := A \setminus \{a\}$ for a set A and an element a. We use $\mathbb{R}$ for the set of real numbers, $\mathbb{Q}$ for the set of rational numbers, $\mathbb{Z}$ for the set of integers. If $S \subseteq \mathbb{R}$ we use the notation $S_{\geq 0} := \{s \in S \mid s \geq 0\}$ to denote the set of its non-negative elements. In particular $\mathbb{Q}_{\geq 0}$ denotes the set of non-negative rational numbers. As usual $\mathbb{R}^d$ denotes the d-dimensional Euclidean space.

Given a finite set V, we say function w is a *weight function* or equivalently $w \in \mathbb{Q}_{\geq 0}^V$ if $w : V \to \mathbb{Q}_{\geq 0}$. We limit our weight functions to $\mathbb{Q}$ because we use Turing machines as a computational model. We can assume that the vertex weights are non-negative for all our problems since otherwise we can simply include all negative weight vertices to a solution without penalty. If $X \subseteq V$, we define the *indicator weight function of X* as $\mathbf{1}_X(v) := 1$ if $v \in X$, otherwise $\mathbf{1}_X(v) := 0$.

2.2. Graphs

A (simple) *graph* G is pair of (finite) sets (V, E), where the elements of V are called *vertices* and E is a set of unordered pairs of vertices called *edges*. A *directed graph* $D := (V, A)$ is defined analogously. Instead of edges, we have *arcs*, which are ordered pairs of vertices. Given a graph G we will sometimes use $V(G)$, and $E(G)$ to denote the sets of vertices and edges respectively, and analogously for directed graphs $V(D), A(D)$. We say that vertices $u, v \in V(G)$ are *adjacent* in a graph G if $uv \in E(G)$. We use $\overline{G}$ to denote the *complement* of graph G. That is $\overline{G} := (V(G), \overline{E}(G))$ where $\overline{E}(G) := \{\{u, v\} \subseteq V(G) \mid \{u, v\} \notin E(G)\}$.

We will frequently discuss specific classes of subgraphs. It is helpful, and common to have a concise notation for each. For a positive integer k, P_k denotes the *path* on k vertices. We denote the *cycle* of length k by C_k. Two disjoint edges with no adjacency between them will be referred to as $2K_2$, and similarly two disjoint P_k's with no edges between them will be called a $2P_k$.

The set of vertices that some particular vertex is adjacent to is called its *neighborhood*. We will need notation to precisely specify this set of neighbors. If G is a graph and $v \in V(G)$, we let $N(v) := \{u \mid vu \in E(G)\}$ and $N[v] :=$

$N(v) \cup \{v\}$. Hence, $N(v)$ is the *open neighborhood* of v. The set $N[v]$ is called the *closed neighborhood* of vertex v.

Sometimes it is easier to show the existence of graphs satisfying certain properties by sampling from a set of randomly generated graphs rather than deterministically finding an example. We will make use of the Erdős-Rényi *random graph model* $G(n,p)$. Here n is the number vertices of the graph and p is the edge probability. Each pair of vertices uv appears in $E(G)$ with independent probability p. The graph-valued random variable with these parameters is denoted $G(n,p)$.

Given a graph G, we say that a subset S of its vertices is a *stable set* if no two vertices of S are adjacent. The *stability number* $\alpha(G)$, is the size of the largest stable set in G. Similarly we say that a subset of vertices is a *clique* if every pair of distinct vertices of K are adjacent in G. The *clique number* $\omega(G)$ is the size of the largest clique in G.

A *complete graph* on r vertices is a graph such that every pair of vertices is adjacent; we use K_r to denote such a graph. A *tournament* T is a directed graph obtained by orienting a complete (undirected) graph.

Given a graph G and a vertex v, we use the notation $\delta(v) := \{e \in E \mid v \in e\}$ for the set of edges incident to v.

Throughout this work we are concerned with *weighted graphs*. This is a graph G together with a weight function $w : V(G) \to \mathbb{Q}_{\geq 0}$. Often, a weighted graph is denoted as the pair (G,w). The *weight* of a set of vertices X of a weighted graph (G,w) is defined as $w(X) := \sum_{v \in X} w(v)$.

A subset of vertices X of a graph G naturally defines a subgraph of G, namely, the subgraph of G *induced* by X. This is the graph $G[X]$ with vertex set X and edge set $\{e \in E(G) \mid e \subseteq X\}$.

2.3. Hypergraphs

A *hypergraph* $\mathcal{H}$ is a pair $\mathcal{H} := (V,E)$ where the elements of V are again called vertices and $E \subseteq 2^V$ is a collection of non-empty subsets of V which we call *edges*. A *k-uniform hypergraph* is a hypergraph $\mathcal{H} = (V,E)$ such that every edge $e \in E$ has size $|e| = k$.

If $\mathcal{H} = (V,E)$ is a hypergraph, and $v \in V$, then we define $\mathcal{H} - v := (V - v, E \setminus \{e \in E \mid v \in e\})$. If $S \subseteq V$ then we extend this definition to $\mathcal{H} - S := (V \setminus S, E \setminus \{e \in E \mid e \cap S \neq \varnothing\})$.

We adapt the notation for graphs whenever its meaning is unambiguous on hypergraphs.

2.4. Combinatorial Optimization

A *polytope* is the convex hull of finitely many points in $\mathbb{R}^n$. If $P = \{x \in [0,1]^n \mid Ax \geq b\}$ is a polytope contained in the unit cube, $\mathsf{SA}_k(P)$ denotes the polytope obtained by running k rounds of the Sherali-Adams hierarchy on P. This hierarchy will be defined in detail later in Chapter 3, Section 3.3.

A *linear program* (LP) models the problem of finding a vector x that minimizes a linear weight function wx, *where w is a row vector*, and x ranges over all

vectors which satisfy a given system $Ax \geq b$. In other words if $P := \{x \mid Ax \geq b\}$, then we want to find $\operatorname{argmin}_{x \in P} wx$, which we often write as

$$\begin{aligned} \min \quad & wx \\ \text{s.t.} \quad & Ax \geq b. \end{aligned}$$

The *ellipsoid method* is an iterative algorithm which can solve LPs in a number of steps which is polynomial in the input size [34]. This method is slow in practice. On the other hand the *simplex method* designed by G.B. Dantzig in the late 1940s works very well in practice. If an LP's feasible region is a non-empty polytope P then at least one extreme point of P is an optimal solution of the LP [57]. The simplex method traverses the extreme points of P via geometric "pivot operations", which successively improve the quality of the solution, until an optimal extreme point is found. Unfortunately it is unproven that the number of extreme points traversed can be bounded by a polynomial in the dimension of matrix A, for some pivot rule [20].

All the problems we study here can be seen as cases of Ek-Vertex Cover, and as such can be modelled as *integer programs*. Given weighted k-uniform hypergraph $(\mathcal{H}, w)$, we construct an integer program as follows:

(1) Introduce variable x_v for each vertex $v \in V(\mathcal{H})$.
(2) For each variable x_v add the integrality constraint: $x_v \in \{0, 1\}$.
(3) For each edge $e \in E(\mathcal{H})$ we add a constraint forcing any feasible solution to meet each edge: $\sum_{v \in e} x_v \geq 1$.
(4) The objective then is to minimize $\sum_{v \in V(\mathcal{H})} w(v) x_v$.

We present this integer program, which we denote as :

$$\begin{aligned} \text{IP}_{\text{HVC}} \quad \min & \sum_{v \in V} w(v) x_v \\ \text{s.t.} & \sum_{v \in e} x_v \geq 1 \qquad e \in E(\mathcal{H}) \\ & x_v \in \{0, 1\} \quad v \in V(\mathcal{H}). \end{aligned}$$

We can *relax* the integrality constraints of IP_{HVC} by replacing $x_v \in \{0, 1\}$ constraints with $x_v \in [0, 1]$ constraints. Relaxing integrality constraints is a natural idea to obtain a linear program contained in the unit cube which we denote LP_{HVC}. We call this relaxation the *basic linear programming relaxation*.

Given an instance of Ek-Vertex Cover $(\mathcal{H}, w)$, let $\text{OPT}(\mathcal{H}, w)$ be the weight of an optimal solution with respect to weight function w. For the rest of this section we assume $\mathcal{H}$ is fixed. Let $d \in \mathbb{Z}_{\geqslant 0}$ be an arbitrary dimension. A system of linear inequalities $Ax \geqslant b$ in $\mathbb{R}^d$ defines an *LP relaxation* of Ek-Vertex Cover on $\mathcal{H}$ if the following hold: (i) For every vertex cover $X \subseteq V(\mathcal{H})$, we have a point $\pi^X \in \mathbb{R}^d$ satisfying $A\pi^X \geqslant b$; (ii) For every weight function $w : V(\mathcal{H}) \to \mathbb{Q}_{\geqslant 0}$, we have an affine function $f_w : \mathbb{R}^d \to \mathbb{R}$; (iii) For all vertex covers $X \subseteq V(\mathcal{H})$ and weight functions $w : V(\mathcal{H}) \to \mathbb{Q}_{\geqslant 0}$, the condition $f_w(\pi^X) = w(X)$ holds. The *size* of the LP relaxation $Ax \geqslant b$ is defined as the number of rows of A. This formalism allows for extended formulations. For

example we can use the Sherali-Adams hierarchy to obtain an LP strengthening of the basic relaxation.

For every weight function w, the quantity $\mathrm{LP}(\mathcal{H}, w) := \min\{f_w(x) \mid Ax \geqslant b\}$ gives a lower bound on $\mathrm{OPT}(H, w)$. The *integrality gap* of the LP relaxation $Ax \geqslant b$ is defined as $\sup\{\mathrm{OPT}(\mathcal{H}, w)/\,\mathrm{LP}(\mathcal{H}, w) \mid w \in \mathbb{Q}_{\geqslant 0}^{V(\mathcal{H})}\}$. Note that the integrality gap is at least 1.

2.5. Algorithms

Later on, we will need to analyze the performance of algorithms. As far as run-time is concerned we use the standard asymptotic notation. See for example [21, Chapter 3]. We say a function f is $O(g(n))$ and write $f(n) = O(g(n))$ if there exists some positive real constant c, integer constant n_0 and function g such that $|f(n)| \leq cg(n)$ for all $n \geq n_0$.

CHAPTER 3

Tools and Techniques

In this work we employ two different algorithmic frameworks for attacking various Ek-Vertex Cover problems. In this chapter we briefly introduce them and explain the ways in which we have adapted them for our purposes.

3.1. Local Ratio Lemma

The *local ratio method* is a simple and elegant paradigm for weighted NP-hard problems [4]. It is based on the simple yet surprisingly deep Local Ratio Lemma which we now state.

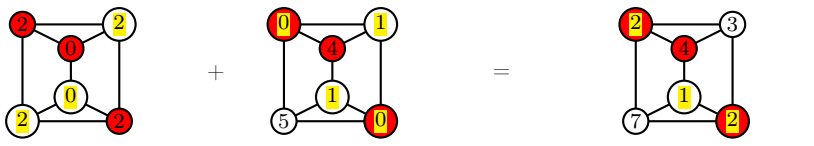
Lemma 3.1 (adapted from [4])**.** *Let $(\mathcal{H}, w)$ be a weighted hypergraph. For $i \in [2]$, let $w_i : V(\mathcal{H}) \to \mathbb{Q}_+$ be a weight function. Suppose that $w = w_1 + w_2$ and that $X \subseteq V(\mathcal{H})$ is a vertex cover of $\mathcal{H}$ such that $w_i(X) \leq \alpha \operatorname{OPT}(\mathcal{H}, w_i)$ for each $i \in [2]$, where $\alpha \geq 1$. Then $w(X) \leq \alpha \operatorname{OPT}(\mathcal{H}, w)$.*

Proof. For $i \in [2]$, let $X_i \subseteq V(\mathcal{H})$ be a vertex cover of $\mathcal{H}$ that has minimum weight with respect to w_i, and let X^* be an optimal solution with respect to w. We have

$$\begin{aligned} w(X) &= w_1(X) + w_2(X) \\ &\leq \alpha \operatorname{OPT}(\mathcal{H}, w_1) + \alpha \operatorname{OPT}(\mathcal{H}, w_2) \\ &= \alpha\, w_1(X_1) + \alpha\, w_2(X_2) \\ &\leq \alpha\, w_1(X^*) + \alpha\, w_2(X^*) \\ &= \alpha\, w(X^*)\,. \end{aligned}$$

□

We demonstrate how to use the Local Ratio Lemma to decompose a weight function for CVD on a small graph in Figure 3.1. Next, we show how the Local



$w_1(X) \leq 2\operatorname{OPT}(G, w_1)$ $w_2(X) \leq 2\operatorname{OPT}(G, w_2)$ $w(X) \leq 2\operatorname{OPT}(G, w)$

Figure 3.1. For CVD on the graph to the right we need to hit all induced P_3's. A feasible solution X is shown in red. We find weight functions w_1, w_2 such that $w_1 + w_2 = w$, and $i \in [2]$, $w_i(X) \leq 2\operatorname{OPT}(G, w_i)$. For each weight function w_i an optimal solution is shown in yellow. By the Local Ratio Lemma $w(X) \leq 2\operatorname{OPT}(G, w)$.

Ratio Lemma (Lemma 3.1) can be used to obtain an k-approximation algorithm for Ek-Vertex Cover.

Lemma 3.2. *Let $\mathcal{H}$ be any k-uniform hypergraph, and let $e \in E(\mathcal{H})$ be any edge. For $v \in V(\mathcal{H})$, let $w_1 := \mathbf{1}_e$. Then $w_1(X) \leq k \operatorname{OPT}(\mathcal{H}, w_1)$ for every vertex subset X.*

Proof. Notice that $\operatorname{OPT}(\mathcal{H}, w_1) \geq 1$ since every vertex cover of $\mathcal{H}$ must contain some vertex of e. Trivially, $w_1(X) \leq \sum_{v \in e} w_1(v) = k$. It follows that $w_1(X) \leq k \operatorname{OPT}(\mathcal{H}, w_1)$. □

This gives a k-approximation algorithm for Ek-Vertex Cover, see Algorithm 1[1].

Algorithm 1 LocalRatio-HVC$(\mathcal{H}, w)$

Input: a weighted hypergraph $(\mathcal{H}, w)$
Output: a vertex cover X of $\mathcal{H}$

```
1: if H has no edge then
2:    X ← ∅
3: else if there is some v ∈ V(H) with w(v) = 0 then
4:    H' ← H − v
5:    w' ← w restricted to V(H')
6:    X' ← LocalRatio-HVC(H', w')
7:    X ← X' if H − X' has no edges; X ← X' ∪ {v} otherwise
8: else
9:    pick any e ∈ E(H)
10:   w_1 ← 1_e
11:   λ* ← argmax{λ ∈ R_{≥0} | ∀u ∈ V(H) : w(u) − λw_1(u) ≥ 0}
12:   w_2 ← w − λ*w_1
13:   X ← LocalRatio-HVC(H, w_2)
14: end if
15: return X
```

Lemma 3.3. LocalRatio-HVC$(\mathcal{H}, w)$ *returns an inclusion-wise minimal vertex cover X such that $w(X) \leq k \operatorname{OPT}(\mathcal{H}, w)$.*

Proof. We proceed by induction on $m := |V(\mathcal{H})| + |\operatorname{supp}(w)|$, where $\operatorname{supp}(w) := \{v \in V(\mathcal{H}) \mid w(v) > 0\}$. If $m = 0$ there is nothing to prove.

Now suppose that $m > 0$, and assume that LocalRatio-HVC$(\mathcal{H}', w')$ correctly returns an inclusion-wise minimal vertex cover X' with $w'(X') \leq k \operatorname{OPT}(\mathcal{H}', w')$ whenever $|V(\mathcal{H}')| + |\operatorname{supp}(w')| < m$.

If line 2 is executed then $\mathcal{H}$ has no edges and $\varnothing$ is correctly returned.

Next, assume that lines 4 through 7 are executed. That is, there is some vertex $v \in V(\mathcal{H})$ such that $w(v) = 0$. Then the vertex cover X returned by Algorithm 1 is inclusion-wise minimal. Indeed, by induction X' is an inclusion-wise minimal vertex cover for $\mathcal{H}'$. If X' is a vertex cover for $\mathcal{H}$ then it is also clearly inclusion-wise minimal. If it is not a vertex cover for $\mathcal{H}$ then $X' \cup \{v\}$ is, and is

[1]recall $\mathcal{H} - v := (V - v, E \setminus \{e \in E \mid v \in e\})$

inclusion-wise minimal. Moreover, we have $w(X) = w(X') \leq k\,\mathrm{OPT}(\mathcal{H}', w') = k\,\mathrm{OPT}(\mathcal{H}, w)$ since $w(v) = 0$.

Finally, assume that lines 9-13 is executed. Then by Lemma 3.2, $w_1(X) \leq k\,\mathrm{OPT}(\mathcal{H}, w_1)$. By construction $|\operatorname{supp}(w_2)| < |\operatorname{supp}(w)|$, therefore by the induction hypothesis $X := \textsc{LocalRatio-HVC}(\mathcal{H}, w_2)$ is an inclusion-wise minimal vertex cover of $\mathcal{H}$ such that $w_2(X) \leq k\,\mathrm{OPT}(\mathcal{H}, w_2)$. Since $w_1 + w_2 = w$, the Local Ratio Lemma applies, and $w(X) \leq k\,\mathrm{OPT}(\mathcal{H}, w)$. □

Algorithm 1 is of similar form as all of the local ratio based algorithms contributed in this work. Generally speaking, the main step that changes is lines 9–10.

Suppose, for some factor $1 < \alpha < k$ that we will always have some, as of yet unspecified, way of finding a weight function w_1 such that for any inclusion-wise minimal vertex cover X, $w_1(X) \leq \alpha\,\mathrm{OPT}(\mathcal{H}, w_1)$. We could then adapt Algorithm 1 at lines 9–10, and the proof of Lemma 3.3 would give an α-approximation algorithm. In the next section we give some definitions which will help us to find such weight functions.

3.2. Good Subgraphs

Consider some $\mathcal{G}$-Vertex Deletion problem where $\mathcal{G}$ is an $\mathcal{F}$-free hereditary graph class. Given a weighted graph (G, w), we can look for induced subgraphs on which there exists a weighting such that any solution must include a large fraction of the total weight. These help define local weight functions for use in the Local Ratio Method, following the form of Algorithm 1. We now state things more precisely and give formal definitions.

Let H be an induced subgraph of G, and let $w_H : V(H) \to \mathbb{Q}_{\geqslant 0}$ be a weight function on H. The weighted graph (H, w_H) is said to be *α-good in G* (for some factor $\alpha \geqslant 1$) if w_H is not identically 0 and

$$\sum_{v \in X \cap V(H)} w_H(v) \leqslant \alpha \cdot \mathrm{OPT}(H, w_H) \tag{1}$$

holds for every inclusion-wise minimal solution X of G.

We will use two methods to establish α-goodness of weighted induced subgraphs. We say that (H, w_H) is *strongly* α-good if w_H is not identically 0 and

$$\sum_{v \in V(H)} w_H(v) \leqslant \alpha \cdot \mathrm{OPT}(H, w_H)\,.$$

Clearly if (H, w_H) is strongly α-good then it is also α-good. We slightly abuse terminology and allow ourselves to say that an induced subgraph H is *α-good in G*(resp. strongly α-good) if there exists a weight function w_H such that (H, w_H) is α-good (resp. strongly α-good) in G. Similarly we say that the weight function w_H is *α-good* (resp. strongly α-good). It is important to note that the local weight function w_H is not necessarily the restriction of the global weight function $w : V(G) \to \mathbb{Q}_{\geqslant 0}$ to $V(H)$.

The identification of good subgraphs forms a key component in the design of many of the approximation algorithms we study here. The idea is to use an α-good subgraph in place of an edge in Algorithm 1 towards achieving an

α-approximation factor, where $\alpha \in [2, k)$. Given an instance of $\mathcal{G}$-Vertex Deletion, (G, w), let $\mathcal{F}'$ be an obstruction set of α-good subgraphs, and $\mathcal{G}'$ the graph class defined by forbidding $\mathcal{F}'$. While G contains an induced subgraph that is contained in $\mathcal{F}'$, we can perform the steps from Algorithm 1 to recursively reduce the problem. Therefore, to obtain an α-approximation algorithm for $\mathcal{G}$-Vertex Deletion, we may assume that our input graph is in $\mathcal{G}'$. Often $\mathcal{G}'$ has algorithmically useful properties which make this sub-problem easier. We now give examples of good subgraphs from Vertex Cover, CVD, SVD, and Claw-VD.

We start with the most simple case of classical Vertex Cover. Let (G, w) be a weighted graph. Take any edge $e \in E(G)$, then $(G[e], \mathbf{1}_e)$ is a strongly 2-good subgraph since $\mathrm{OPT}(G[e], \mathbf{1}_e) = 1$, and $\mathbf{1}_e(e) = 2$. Next suppose G contains some edge $e' = uv$ where $\deg(u) = 1$. $\mathrm{OPT}(G[e'], \mathbf{1}_{e'}) = 1 = \mathrm{OPT}(G, \mathbf{1}_{e'})$. So e' is 1-good, since any minimal inclusion-wise vertex cover contains either u or v but not both. Note that e' is not strongly 1-good since $\mathbf{1}_{e'}(e') = 2$. See Figure 3.2.

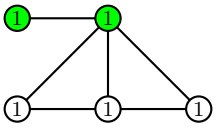


Figure 3.2. A 1-good but not strongly 1-good induced subgraph forVertex Cover.

In the case of CVD, recall that a solution needs to cover every induced P_3. Let $w_{C_4} := \mathbf{1}_{C_4}$, and observe that (C_4, w_{C_4}) is strongly 2-good since $w_{C_4}(C_4) = 4 = 2\,\mathrm{OPT}(C_4, w_{C_4})$. On the other hand, let v be a degree 1 vertex and vab an induced P_3. This gives an example of a 2-good subgraph which is not strongly 2-good. Let $H := vab, w_H := \mathbf{1}_{vab}$. Then (H, w_H) is 2-good since any inclusion-wise minimal solution X includes at most 2 vertices from induced path vab. See Figure 3.3.

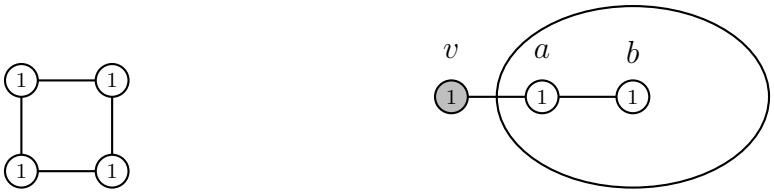


Figure 3.3. A solution to CVD must meet each induced P_3. The unit weighted C_4 on the left is strongly 2-good. On the right vab is P_3 in a larger graph such that grey vertex v has degree-1. Therefore vab is 2-good but not strongly 2-good.

In Claw-VD a solution needs to cover all induced *claws* ($K_{1,3}$). Consider a $K_{1,4}$ with vertex set $\{v, a, b, c, d\}$, and edge set $\{va, vb, vc, vd\}$. We claim that $K_{1,4}$ is strongly 3-good. Set $w(v) := 2$, and $w(a) := w(b) := w(w) := w(d) := 1$. Then the total weight is $w(\{v, a, b, c, d\}) = 6$, and any solution X must have $w(X) \geq 2$, either by choosing v or two other vertices for a ratio of $6/2 = 3$. See Figure 3.4.

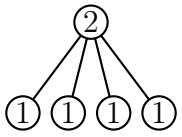


FIGURE 3.4. $K_{1,4}$ is strongly 3-good for CLAW-VD.

For our final example we consider SVD. Recall that a solution X to SVD on weighted graph (G, w) is a subset of vertices $X \subseteq V(G)$ such that $V(G - X)$ can be partitioned into disjoint sets S, K such that $S \cup K = V(G - X)$, with S a stable set, and K a clique.

Let $k > 5$, set $w_{P_k} := \mathbf{1}_{P_k}$. We will show that (P_k, w_{P_k}) is $\frac{2k}{k-4}$-good. The largest clique in P_k is of size 2, an edge uv, which, assuming k is even, can be chosen to separate P_k into two odd paths. In each odd path we can ignore one vertex, but then must pick half of the remaining vertices to include in the solution. Let these ignored vertices be a, b. Then any solution must include half of the remaining vertices from $V(P_k) - \{u, v, a, b\}$ of which there are $\frac{k-4}{2}$. If k is odd then after selecting the clique, only one of the remaining paths will be of odd length, and any solution must include $\frac{k-3}{2}$ vertices. See Figure 3.5. Thus $|V(P_k)|/\operatorname{OPT}(P_k, w_{P_k}) \leq \frac{2k}{k-4}$.

We end this chapter by giving a general recipe for an α-approximation algorithm on $\mathcal{G}$-VERTEX DELETION problems on input (G, w).

Algorithm 2 LocalRatio-HVD(G, w)

Input: a weighted graph (G, w).
Output: a hitting set X for G.

if $G \in \mathcal{G}$ **then**
  $X \leftarrow \varnothing$
else if there is some $v \in V(G)$ with $w(v) = 0$ **then**
  $G' \leftarrow G - v$
  $w' \leftarrow w$ restricted to $V(G')$
  $X' \leftarrow$ LOCALRATIO-HVD(G', w')
  $X \leftarrow X'$ if $G - X' \in \mathcal{G}$; $X \leftarrow X' \cup \{v\}$ otherwise
else if G contains an α-good subgraph (H, w_H) **then**
  $\lambda^* \leftarrow \operatorname{argmax}\{\lambda \in \mathbb{R}_{\geq 0} \mid \forall u \in V(G) : w(u) - \lambda w_H(u) \geq 0\}$
  $w_2 \leftarrow w - \lambda^* w_H$
  $X \leftarrow$ LOCALRATIO-HVD(G, w_2)
else
  $X \leftarrow$ RESTRICTEDINPUTSUBROUTINE(G, w)
end if
return X

The subroutine RESTRICTEDINPUTSUBROUTINE(G, w) is a placeholder for some α-approximation algorithm which only takes, as input, weighted graphs belonging to intermediary graph class $\mathcal{G}'$ defined by forbidding $\mathcal{F}'$, which contains α-good subgraphs.

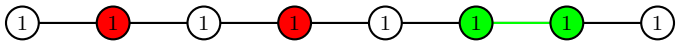


FIGURE 3.5. A solution for SVD splits the vertices not in the hitting set into a clique and a stable set. Set $H := P_8$, $w_H := \mathbf{1}_{P_8}$, then (H, w_H) is strongly 4-good. $w_H(V(H)) = 8$. $\mathrm{OPT}(H, w_H) = 2$. The clique is shown in green, and an optimal solution is shown in red.

Lemma 3.4. *Given an obstruction set of α-good subgraphs $\mathcal{F}'$, and α-approximation algorithm* RESTRICTEDINPUTSUBROUTINE(G', w') *defined on $\mathcal{F}'$-free graphs,* LOCALRATIO-HVD(G, w) *returns an inclusion-wise minimal hitting set X such that $w(X) \leq \alpha\,\mathrm{OPT}(\mathcal{H}, w)$.*

PROOF. The proof is the essentially same as that of Lemma 3.3, with the recursion terminating either when G is $\mathcal{F}$-free or at line 13 when G is $\mathcal{F}'$-free, and an α-approximate solution is returned by the restricted input subroutine. □

3.3. Polyhedral Tools

We showed in Chapter 2, Section 2.4 how to model Ek-Vertex Cover as an integer program. Here we explain how to apply the technique of *iterative rounding* to this formulation. We will augment the standard iterative rounding technique by using stronger relaxations obtained from Sherali-Adams hierarchy [60] which we formally define.

Rounding. Let $(\mathcal{H}, w)$ be a weighted k-uniform hypergraph. Recall the integer programming formulation for its minimum vertex cover is IP_{HVC} (see page 21).

An optimal solution to this IP clearly solves Ek-Vertex Cover but unfortunately there is no known way for solving it for all instances in polynomial time. On the other hand we can solve linear programs in time polynomial of the input size [34]. We can relax the integrality requirement for our variables and obtain the basic linear relaxation:

$$\begin{aligned} \text{LP}_{\text{HVC}} \quad \min & \sum_{v \in V} w(v) x_v \\ \text{s.t.} & \sum_{v \in e} x_v \geq 1 \qquad e \in E(\mathcal{H}) \\ & x_v \in [0,1] \qquad v \in V(\mathcal{H}). \end{aligned}$$

We can solve LP_{HVC} and obtain an optimal yet potentially fractional solution x^* to LP_{HVC}. Since the feasible region of IP_{HVC} is contained within LP_{HVC}, the objective optimal value of LP_{HVC} is a useful lower bound on the optimal integral solution, $\text{LP}_{\text{HVC}}(\mathcal{H}, w) \leq \text{IP}_{\text{HVC}}(\mathcal{H}, w)$.

For each edge $e \in E(\mathcal{H})$ we have $\sum_{v \in e} x^*_v \geq 1$. Consider $X := \{v \mid x^*_v \geq \frac{1}{k}\}$. X is a feasible solution to Ek-Vertex Cover since X meets each edge at least once. Moreover, X is a k-approximation since $w(X) \leq kwx^* = k\,\text{LP}_{\text{HVC}}(\mathcal{H}, w) \leq k\text{IP}_{\text{HVC}}(\mathcal{H}, w) = k\,\text{OPT}(\mathcal{H}, w)$.

This basic technique of LP rounding works here since each edge must have a vertex with $x^*_v \geq \frac{1}{k}$. However, for $\alpha < k$ we can not assume in general that each edge contains some $x^*_v \geq \frac{1}{\alpha}$. Fortunately, if we can show the weaker condition that there must exist some $v \in V(\mathcal{H})$ such that $x^*_v \geq \frac{1}{\alpha}$, then, although this is not enough to build a complete solution outright, we can at least take action to reduce the size of the problem. We add v to our partial solution, remove v from $\mathcal{H}$ and iterate this process by finding a new solution to $\text{LP}_{\text{HVC}}(\mathcal{H} - v, w_{\mathcal{H}-v})$ and continuing, where $w_{\mathcal{H}-v}$ is the restriction of w to $V(\mathcal{H} - v)$. This is the process of iterative rounding [43, 51]. At each iteration i with input $(\mathcal{H}_i, w_i)$, where w_i is w restricted to $V(\mathcal{H}_i)$, we take an optimal solution x^i of $\text{LP}_{\text{HVC}}(\mathcal{H}_i, w_i)$, round all components with $x^i_v \geq \frac{1}{\alpha}$, add the corresponding vertices to our partial solution $X_{i+1} := X_i \cup R_{i+1}$ where $R_{i+1} := \{v \in V(\mathcal{H}_i) \mid x^i_v \geq \frac{1}{\alpha}\}$, remove them from the hypergraph $\mathcal{H}_{i+1} := \mathcal{H}_i - R_{i+1}$ obtaining $(\mathcal{H}_{i+1}, w_{i+1})$. We iterate this process until there are no more components to round at iteration l. If X_l is a feasible solution to the instance $(\mathcal{H}_0, w_0) := (\mathcal{H}, w)$, then we can show that $X := X_l$ is an α approximate solution.

Suppose that the process terminates after l iterations beginning with $X_0 := \varnothing$. We proceed by induction on l. If $l = 0$, then clearly $E(\mathcal{H}) = \varnothing$ and there is nothing to show. We assume for induction that when $0 \leq i < l$, if X_i is returned as a feasible solution for some initial input $(\mathcal{H}', w')$ then $w'(X_i) \leq \alpha \operatorname{OPT}(\mathcal{H}', w')$. Let $L := X_1$ be the vertices added when called on input $(\mathcal{H}, w)$ in the first iteration. If X_l is a feasible solution for $\mathcal{H}$ then $X_l - L$ must be feasible for $\mathcal{H} - L$ and is found in $l - 1$ iterations. Let $x := x^0$ be an optimal solution to $\operatorname{LP}_{\mathrm{HVC}}(\mathcal{H}, w)$. By induction, and since the restriction of x to $V(\mathcal{H} - L)$ is feasible for $\operatorname{LP}_{\mathrm{HVC}}(\mathcal{H} - L, w_1)$ we have

$$w(X_l - L) \leq \alpha \operatorname{LP}_{\mathrm{HVC}}(\mathcal{H} - L, w_1) \tag{1}$$

$$\leq \alpha \operatorname{LP}_{\mathrm{HVC}}(\mathcal{H}, w) - \alpha \sum_{v \in L} w(v) x_v. \tag{2}$$

By the choice of L, $w(L) \leq \alpha \sum_{v \in L} w(v) x_v$, and $w(X_l) \leq \alpha \operatorname{OPT}(\mathcal{H}, w)$ as required.

We need not use the *basic relaxation* for the LP used in this process. We can choose a stronger relaxation, and then *weaken it* in successive iterations. A stronger relaxation makes it easier to find components to round, whereas a weaker relaxation is simpler, and therefore easier to analyse. Again let $(\mathcal{H}, w)$ be an instance of Ek-Vertex Cover, and suppose $\operatorname{LP}_1(\mathcal{H}) \subseteq \operatorname{LP}_{\mathrm{HVC}}(\mathcal{H})$. Note that inequalities (1), (2) remain valid if x is an optimal solution of $\operatorname{LP}_1(\mathcal{H}, w)$ rather than $\operatorname{LP}_{\mathrm{HVC}}(\mathcal{H}, w)$.

Towards this goal we introduce the Sherali-Adams hierarchy of extended formulations. This will allow us to iteratively strengthen the basic LP relaxation.

Sherali-Adams Hierarchy. Let $P = \{x \in \mathbb{R}^n \mid Ax \geq b\}$ be a polytope contained in $[0, 1]^n$ and $P_I := \operatorname{conv}(P \cap \mathbb{Z}^n)$. Numerous optimization problems can be formulated as minimizing a linear function over P_I, where P has only a polynomial number of constraints. For example, let $\mathcal{H}$ be a 3-uniform hypergraph and $w : V(\mathcal{H}) \to \mathbb{Q}_{\geq 0}$. Then $\operatorname{OPT}(\mathcal{H}, w)$ is simply the minimum of wx over P_I, where $P = P(\mathcal{H})$ is the basic relaxation defined above.

The Sherali-Adams hierarchy [60] is a simple but powerful method to obtain improved *approximations* for P_I. Since it does not require any knowledge of the structure of P_I, it is widely applicable. The procedure comes with a parameter r, which specifies the accuracy of the approximation. That is, for each $r \in \mathbb{N}$, we define a polytope $\mathsf{SA}_r(P)$. These polytopes satisfy $P = \mathsf{SA}_0(P) \supseteq \mathsf{SA}_1(P) \supseteq \cdots \supseteq \mathsf{SA}_r(P) \supseteq \cdots \supseteq P_I$.

An important property of the procedure is that if P is described by a polynomial number of constraints and r is a constant, then $\mathsf{SA}_r(P)$ is also described by a polynomial number of constraints (in a higher dimensional space). Therefore, for NP-hard optimization problems (such as those we study here), one should not expect that $\mathsf{SA}_r(P) = P_I$ for some constant r. However, as we will see, good approximations of P_I can be extremely useful if we want to *approximately* optimize over P_I.

Here is a formal description of the Sherali-Adams hierarchy. Let $P = \{x \in \mathbb{R}^n \mid Ax \geq b\} \subseteq [0, 1]^n$ and $r \in \mathbb{N}$. Let N_r be the *nonlinear* system obtained from P by multiplying each constraint by $\prod_{i \in I} x_i \prod_{j \in J} (1 - x_j)$ for all disjoint subsets

I, J of $[n]$ such that $1 \leq |I| + |J| \leq r$. Note that if $x_i \in \{0, 1\}$, then $x_i^2 = x_i$. Therefore, we can obtain a *linear* system L_r from N_r by setting $x_i^2 := x_i$ for all $i \in [n]$ and then $x_I := \prod_{i \in I} x_i$ for all $I \subseteq [n]$ with $|I| \geq 2$. We then let $\mathsf{SA}_r(P)$ be the projection of L_r onto the variables x_i, $i \in [n]$. We let $\mathsf{SA}_r(\mathcal{H}) := \mathsf{SA}_r(P(\mathcal{H}))$, where $P(\mathcal{H})$ is the basic relaxation.

Let (G, w) be an instance of $\mathcal{G}$-VERTEX DELETION, $(\mathcal{H}, w)$ be the corresponding instance of Ek-VERTEX COVER, and $x \in \mathsf{SA}_r(\mathcal{H}, w)$ for some $r \geq 0$. Suppose for a given $\alpha > 0$ that $\mathcal{F}'$ is a a family of induced subgraphs, such that if $F \in \mathcal{F}'$ and $F \subseteq G$, then there is some $v \in V(G)$ such that $x_v \geq \frac{1}{\alpha}$. Then $\mathcal{F}'$ defines an intermediary graph class $\mathcal{G}'$ since we can apply an iterative rounding step as long as $G \notin \mathcal{G}'$. In the next section we give a useful example of $\mathcal{F}'$ in E3-VERTEX COVER problems.

3.4. Diagonals in E3-Vertex Cover

In FVST, and CVD we make use of the inequalities defining $\mathsf{SA}_1(\mathcal{H})$, which we now describe.

THEOREM 3.5. *Let $(\mathcal{H}, w)$ be an instance of* E3-VERTEX COVER. *For $\{a, b, c\} \in E(\mathcal{H})$ and $d \in V(\mathcal{H}) - \{a, b, c\}$, $\mathsf{SA}_1(\mathcal{H})$ contains the inequalities:*

$$x_a + x_b + x_c \geq 1 + x_{ab} + x_{bc}\,, \tag{1}$$

$$x_{da} + x_{db} + x_{dc} \geq x_d \quad \textit{and} \tag{2}$$

$$x_a + x_b + x_c + x_d \geq 1 + x_{ad} + x_{bd} + x_{cd}\,. \tag{3}$$

In addition, there are the inequalities for all distinct $a, b \in V(\mathcal{H})$.

$$1 \geq x_a \geq x_{ab} \geq 0 \tag{4}$$

PROOF. We first show (2). Recall that $\mathsf{SA}_0(\mathcal{H}) = \mathrm{LP}_{\mathrm{HVC}}(\mathcal{H})$, which contains $x_a + x_b + x_c \geq 1$. Therefore $x_d(x_a + x_b + x_c \geq 1)$ is equivalent to $x_d x_a + x_d x_b + x_d x_c \geq x_d$ which linearizes to $x_{da} + x_{db} + x_{dc} \geq x_d$ is an inequality of $\mathsf{SA}_1(\mathcal{H})$.

To show inequality (1), note that $(1 - x_a)(x_a + x_b + x_c \geq 1)$ is equivalent to $x_a + x_b + x_c \geq 1 + x_{ab} + x_{ac}$.

Similarly we obtain inequality (3) since $(1 - x_d)(x_a + x_b + x_c \geq 1)$ is equivalent to $x_a + x_b + x_c \geq 1 + x_{da} + x_{db} + x_{dc}$.

Finally $x_a(1 \geq x_b \geq 0)$ gives $x_a \geq x_{ab} \geq 0$, and since $1 \geq x_a$ is a constraint of $\mathsf{SA}_0(\mathcal{H})$ inequality (4) follows. □

Let $\binom{V(\mathcal{H})}{2}$ be the set of all unordered pairs of vertices of $\mathcal{H}$. The polytope $\mathsf{SA}_1(\mathcal{H})$ is the set of all $(x_a)_{a \in V(\mathcal{H})} \in \mathbb{R}^{V(\mathcal{H})}$ such that there exists $(x_{ab})_{ab \in E_2(\mathcal{H})} \in \mathbb{R}^{\binom{V(\mathcal{H})}{2}}$ so that inequalities (1)–(4) are satisfied.

Let $(\mathcal{H}, w)$ be an instance of E3-VERTEX COVER, with basic linear relaxation $\mathrm{LP}(\mathcal{H})$. We are able to give a combinatorial interpretation of some common combinations of constraints of $\mathsf{SA}_1(\mathrm{LP}(\mathcal{H}))$ which guarantee a lower bound on some component. This can allow us to design rounding algorithms keeping a general hypergraph formulation of the problems.

An (unordered) pair of vertices ab is a *diagonal* if there are vertices u, v such that $\{u, v, a\} \in E(\mathcal{H})$ and $\{u, v, b\} \in E(\mathcal{H})$. We often will denote a hyperedge $\{a, b, c\}$ as abc. We say that an edge *contains a diagonal* if at least one of its

pairs of vertices is a diagonal, and a hyperedge is *heavy* if it contains at least two diagonals. A 3-uniform hypergraph $\mathcal{H}$ is *heavy* if at least one of its hyperedges is heavy. If a 3-uniform hypergraph is not heavy, we say that it is *light*.

We now give two lemmas which will be useful in both CVD and FVST.

Lemma 3.6. *Let $\mathcal{H}$ be a 3-uniform hypergraph and $x \in \mathsf{SA}_1(\mathcal{H})$. If $x_v < 3/7$ for all $v \in V(\mathcal{H})$, then $\mathcal{H}$ is light.*

PROOF. First, let ab be a diagonal of $\mathcal{H}$. We claim that $x_{ab} \geq 1/7$. Indeed, since ab is a diagonal there must be $u, v \in V(\mathcal{H})$ with $uva, uvb \in E(T)$. We apply the inequalities of Theorem 3.5. From (1), $x_a + x_u + x_v \geq 1 + x_{au} + x_{av}$ and from (2), $x_{ab} + x_{au} + x_{av} \geq x_a$. Adding these two inequalities, we obtain $x_u + x_v + x_{ab} \geq 1$, implying our claim.

Now, suppose by contradiction that $\mathcal{H}$ is heavy. Hence there exists $abc \in E(\mathcal{H})$ such that ab and bc are diagonals. By (1), we have $x_a + x_b + x_c \geq 1 + x_{ab} + x_{bc}$. By the above claim, $x_{ab} \geq 1/7$ and $x_{bc} \geq 1/7$, making the right hand side at least $9/7$. So $\max(x_a, x_b, x_w) \geq 3/7$, a contradiction. □

Similarly we have the weaker result:

Lemma 3.7. *Let $\mathcal{H}$ be a 3-uniform hypergraph and $x \in \mathsf{SA}_1(\mathcal{H})$. If $x_v < 2/5$ for all $v \in V(\mathcal{H})$, then no edge of $\mathcal{H}$ contains a diagonal.*

PROOF. As in the previous proof, let ab be a diagonal of $\mathcal{H}$. We claim that $x_{ab} \geq 1/5$. Indeed, since ab is a diagonal there must be $u, v \in V(T)$ with $uva, uvb \in E(\mathcal{H})$. From (1), $x_a + x_u + x_v \geq 1 + x_{au} + x_{av}$ and from (2), $x_{ab} + x_{au} + x_{av} \geq x_a$. Adding these two inequalities, we obtain $x_u + x_v + x_{ab} \geq 1$, implying our claim.

Now, suppose by contradiction that $\mathcal{H}$ has a hyperedge with a diagonal. Hence there exists $abc \in E(\mathcal{H})$ such that ab is a diagonal. By (1), we have $x_a + x_b + x_c \geq 1 + x_{ab} + x_{bc}$. By the above claim, $x_{ab} \geq 1/5$ and $x_{bc} \geq 0$ making the right hand side at least $6/5$. So $\max(x_a, x_b, x_w) \geq 2/5$, a contradiction. □

FIGURE 3.6. In both CVD (left) and FVST (right), $abc, abd \in E(\mathcal{H})$, so c, d are diagonals. Also $cdu \in E(\mathcal{H})$. By Lemma 3.7, $\max(x_a, x_b, x_c, x_d, x_u) \geq \frac{2}{5}$.

If Lemma 3.7 is not applicable for an E3-VERTEX COVER instance, it means that the underlying excludes certain small induced subgraphs. See Figure 3.6.

Part 2

The Main Results

We now use the tools we have described in Part 1 to develop non-trivial approximation algorithms for Claw-VD, FVST, CVD, and SVD.

CHAPTER 4

Claw Vertex Deletion

This chapter is not based on a published work. It is the fruit of collaboration with Samuel Fiorini and Tony Huynh.

4.1. Background

Given a weighted graph (G, w), recall that a claw is an induced $K_{1,3}$ subgraph. Given a claw K we call the vertex of degree 3 in K the *center* of K. We denote the set of all claws of G as *Claw(G)*. Claw-VD asks to find the smallest weight subset $X \subseteq V(G)$ such that $G - X$ is claw-free.

Lewis and Yannakakis showed that this problem is NP-hard even when G is bipartite [52]. Kumar, Mishra, Safina-Devi, and Saurabh give a 3-approximation algorithm for this bipartite case [50]. We will show that the Local Ratio based approach of Algorithm 2 matches this approximation ratio on the broader class of triangle-free graphs. In terms of the notation from Chapter 1, we give a 3-approximation algorithm for $\mathcal{G}$-Vertex Deletion on graph class $\mathcal{G}$ defined by forbidding $\mathcal{F} = \{K_{1,3}, K_3\}$. We then show that, under UGC, it is hard to approximate the problem on general graphs better than a factor of 3.

4.2. Good Subgraphs

To employ the local ratio technique of Section 3.2 we must identify a *good subgraph*. We see from Figure 3.4 and the discussion in Section 3.2 that $K_{1,4}$ is strongly 3-good. We have seen in Section 3.1 that Algorithm 2 then gives a local ratio 3-approximation algorithm for vertex deletion to $K_{1,4}$-free graphs. By Lemma 3.4, what is left is to define a 3-approximation subroutine which takes as input graphs belonging to intermediary graph class $\mathcal{G}'$ defined by forbidding $\mathcal{F}' = \{K_{1,4}, K_3\}$. Fortunately this turns out to be another easy application of local ratio.

If $G \in \mathcal{G}'$ then by definition it is triangle-free and contains no induced $K_{1,4}$. Since G contains no K_3, for all $v \in V(G)$, $N(v)$ must be an independent set. Since G contains no $K_{1,4}$ it follows that $\deg(v) \leq 3$ for all $v \in V(G)$. We now show that any induced $K_{1,3}$ subgraph of $G \in \mathcal{G}'$ is 3-good.

Lemma 4.1. *Let $G \in \mathcal{G}'$ be a weighted $\{K_{1,4}, K_3\}$-free graph. If $K \subseteq G$ is an induced $K_{1,3}$(claw) and X is any inclusion-wise minimal solution, $|X \cap K| \leq 3$.*

Proof. Suppose by way of contradiction that $|X \cap K| = 4$. This must mean, by minimality of X, that each vertex of K appears in some other claw as well. In particular this means that the center of K, which we label v, must be in some induced claw K' with $V(K) \cap V(K') = \{v\}$. Hence, v must be adjacent to some

vertex $v' \in V(K') \setminus V(K)$. But this means $\deg(v) > 3$, implying that either G contains an induced $K_{1,4}$ or a triangle, a contradiction. □

These observations immediately plug into the generalized local ratio method we developed for the $\mathcal{G}$-VERTEX DELETION in Chapter 3, see Algorithm 2. The algorithm first iteratively removes all $K_{1,4}$'s until it either returns a solution or the remaining graph $G' \in \mathcal{G}'$. In the later case, we have by Lemma 4.1 that each claw is 3-good. Therefore we can in fact use Algorithm 1 to define the subroutine on line 13.

THEOREM 4.2. *There is a 3-approximation algorithm for* CLAW-VD *on weighted triangle-free graphs.*

Since the local ratio based approach gave this result so easily, we considered if an LP rounding approach could also work.

4.3. A Rounding Attempt

Observe that the degree-1 vertices in a $K_{1,4}$ are all diagonals by virtue of the $K_{1,3}$'s they intersect. Thus if we take $x \in \mathsf{SA}_1(K_{1,4})$ we can follow the technique used in Lemma 3.7 to derive the inequalities to show that x contains a component $x_v \geq 7/24$. This suggests that there might be an iterative rounding 24/7-approximation algorithm over $\mathsf{SA}_1(G)$. However it is unclear what to do when G no longer contains a $K_{1,4}$. Thus we ask the open problem:

QUESTION 1. *If (G, w) is triangle-free, and contains no induced $K_{1,4}$, what is the integrality gap of the basic linear programming relaxation* $\mathrm{LP}_{\mathrm{CFVD}}(G, w)$*?*

4.4. Hardness Result

In this section we will show that it is UGC-hard to approximate CLAW-VD in general graphs better than within a factor of 3. We will do so with an approximation preserving reduction to the following problem.

PROBLEM 7 (K_3 HITTING SET). *Given weighted graph (G, w) find a minimum weight set X such that $G - X$ contains no K_3.*

Guruswami and Lee [37] showed that it is UGC-hard to approximate K_3 HITTING SET better than a factor of 3.

THEOREM 4.3. *There is an approximation preserving reduction from K_3* HITTING SET *to* CLAW-VD.

PROOF. Let (G, w) be a weighted graph, and we let $\mathrm{OPT}_{K_3}(G, w)$ be the weight of an optimal solution for K_3 HITTING SET. Similarly let $\mathrm{OPT}_{CFVD}(G, w)$ be the weight of an optimal solution for CLAW-VD.

Given instance (G, w) of K_3 HITTING SET, we obtain an instance for CLAW-VD as follows. Construct a new graph G' from the complement $\overline{G}$ by adding a new vertex v as an apex vertex. That is, define G' by letting $V(G') := V(\overline{G}) \cup \{v\} = V(G) \cup \{v\}$ and $E(G') := E(\overline{G}) \cup \{vu : u \in V(G)\}$. Next define $w'(u) := w(u), u \in V(G)$, and for the apex $w'(v) := w(V(G)) + 1$, giving v more than the total weight of all of the rest of the vertices.

FIGURE 4.1. The approximation preserving reduction from K_3 hitting set to CLAW-VD. On left is the K_3 hitting set instance G, w. On the right we take the complement and add green apex with more than the total weight G', w'.

Now let X be an optimal solution for K_3 HITTING SET on input (G, w). That means $w(X) = \text{OPT}_{K_3}(G, w)$ and $G - X$ is triangle-free. Note that any independent set $\{a, b, c\}$ of size 3 in $\overline{G}$ forms a claw with v in G', and a K_3 in G. Thus $G' - X$ contains no independent set of size 3 and is therefore a feasible solution for CLAW-VD. This shows that $\text{OPT}_{CFVD}(G', w') \leq \text{OPT}_{K_3}(G, w)$.

Next let X' be an optimal solution to CLAW-VD on input (G', w'). Notice that by the choice of $w'(v)$, that $v \notin X'$. But v forms a claw with any independent set of size 3 and so $G' - X'$ has no independent set of size 3 which means that $G - X'$ has no triangles. So we have shown that $\text{OPT}_{CFVD}(G', w') = \text{OPT}_{K_3}(G, w)$, and that for every feasible solution X' to CLAW-VD on (G', w') we can obtain a solution X in polynomial time for K_3 HITTING SET on (G, w) such that $w(X) \leq w'(X')$.

Since the construction of G' also takes polynomial time, we have an approximation preserving reduction from K_3 HITTING SET to CLAW-VD. □

Corollary 4.4. *It is UGC-hard to approximate* CLAW-VD *better than a factor of 3.*

4.5. Concluding Remarks

We have shown that the CLAW-VD in general graphs is UGC-hard to approximate better than 3. The local ratio method gave an easy way to achieve this when we restrict the input to triangle free graphs. We ask:

QUESTION 2. *Is there a 3-approximation algorithm for* CLAW-VD*?*

This short chapter illustrates the immediate effectiveness of the good-subgraph local ratio framework developed in Chapter 3. Interestingly $\mathcal{F}' = \{K_{1,4}\}$ is the basis of our local ratio approach, and also seems to be useful from the perspective of rounding SA_1. See Section 4.3.

CHAPTER 5

Feedback Vertex Problem in Tournaments

This chapter is based on the paper [2], in which Aprile, Drescher, Fiorini, and Huynh develop a new 7/3-approximation algorithm based on SA_1 (see Section 3.3 in Chapter 3, Part 1).

The result matches the best deterministic approximation algorithm for FVST due to Mnich, Williams, and Végh [56], and is a significant simplification of their approach.

We gave a brief history of the FVST problem in Chapter 1. In Section 5.2 we use *diagonals* from Section 3.4 in Chapter 3, Part 1 to develop an iterative rounding procedure. We classify every tournament as either *light* or *heavy*, and derive some structural properties of light tournaments. These light tournaments serve as an intermediary class from which we stop iterating and solve the remaining problem directly via a restricted input 7/3-approximation algorithm. In Section 5.3, we describe this restricted input subroutine which utilizes breadth first search layering. Finally, in Section 5.4, we state our 7/3-approximation algorithm in full and prove its correctness, see Algorithm 3 .

5.1. Overview

We review the history of the problem, then we discuss our contribution and its relevance to previous work.

State of the Art. The first non-trivial approximation algorithm for FVST was a 5/2-approximation algorithm by Cai, Deng, and Zang [14]. Cai *et al.*'s approach utilizes the local ratio method but their results have polyhedral implications as well. They show that whenever a tournament avoids certain subtournaments, the basic LP relaxation of FVST is integral. We now explain the details.

Let T be a tournament and $\triangle(T)$ denote the collection of all $\{a, b, c\} \subseteq V(T)$ that induce a directed triangle in T. The *basic relaxation* for T is the polytope

$$P(T) := \{x \in [0,1]^{V(T)} \mid \forall \{a,b,c\} \in \triangle(T) : x_a + x_b + x_c \geq 1\}.$$

Let $\mathcal{T}_5$ be the set of tournaments on 5 vertices where the minimum FVS has size 2. Up to isomorphism, $|\mathcal{T}_5| = 3$ (see [14], and Figure 5.1). We say that T is *$\mathcal{T}_5$-free* if no subtournament of T is isomorphic to a member of $\mathcal{T}_5$. More generally, let $\mathcal{T}$ be a collection of tournaments. A *$\mathcal{T}$-subtournament* of T is a subtournament of T that is isomorphic to some tournament of $\mathcal{T}$. We say that T is *$\mathcal{T}$-free* if T does not contain a $\mathcal{T}$-subtournament.

Cai *et al.* prove that $P(T)$ is integral as soon as T is $\mathcal{T}_5$-free. In this case solving a polynomial-size LP gives a minimum weight FVS. We let $\mathrm{CDZ}(T, w)$,

be the polynomial-time algorithm from [14], that given a $\mathcal{T}_5$-free tournament T and $w : V(T) \to \mathbb{Q}_{\geq 0}$, finds a minimum weight feedback vertex set of T.

A 5/2-approximation algorithm follows directly from this. Using the local ratio technique, while T contains a $\mathcal{T}_5$-subtournament S, one can reduce to a smaller instance with one vertex of S removed. If one is aiming for a 5/2-approximation algorithm, one can reduce to a $\mathcal{T}_5$-free tournament T, for which one can even solve the problem exactly by applying $\mathrm{CDZ}(T, w)$.

The 5/2-approximation algorithm of [14] was improved to a 7/3-approximation algorithm by Mnich, Williams, and Végh [56]. Loosely speaking, Mnich *et al.*'s algorithm replaces $\mathcal{T}_5$ by $\mathcal{T}_7$, defined as the set of tournaments on 7 vertices where the minimum FVS has size 3. It is known that, up to isomorphism, $|\mathcal{T}_7| = 121$ (see [56]).

If one is aiming for a 7/3-approximation algorithm, one can reduce to $\mathcal{T}_7$-free tournaments. In fact, instead of using the local ratio technique, [56] use iterative rounding, see the next paragraph. However, the basic relaxation is not necessarily integral for $\mathcal{T}_7$-free tournaments, so obtaining a 7/3-approximation algorithm requires more work.

The algorithm in [56] consists of two phases. Let the *$\mathcal{T}_7$-relaxation* be the LP obtained from the basic relaxation by adding the constraint $\sum_{v \in V(S)} x_v \geq 3$ for each $\mathcal{T}_7$-subtournament S of T. The first phase is an iterative rounding procedure on the $\mathcal{T}_7$-relaxation. This reduces the problem to a residual tournament which is $\mathcal{T}_7$-free and incurs an approximation factor of 7/3. The second phase is a 7/3-approximation algorithm for FVST on the residual tournament, via an intricate layering procedure.

Recently, Lokshtanov, Misra, Mukherjee, Panolan, Philip, and Saurab [54] gave a *randomized* 2-approximation algorithm for FVST. Their algorithm does not rely on [14], but rather on the idea of guessing vertices which are not part of some optimal FVS and that of controlling the in-degree sequence of the tournament. If X is a guessed optimal solution and $v \notin X$ then any triangle $\{a, b, v\} \in \triangle(T)$ is in a sense 2-good with respect to particular solution X by taking $w_1(v) := w(v), w_1(a) := 1, w_1(b) := 1$ then $w_1(\{a, b, v\} \cap X) \leq 2$.

The derandomized version of their algorithm runs in quasi-polynomial-time [54]. A deterministic 2-approximation algorithm would be best possible, since for every $\epsilon > 0$, FVST does not have a $(2 - \epsilon)$-approximation algorithm, unless the Unique Games Conjecture is false or P=NP [46, 61].

FPT results. FVST is the restriction of the more general DFVS problem, in which the input can be any directed graph. Given a directed graph D and parameter k as input, the task is to decide if D has a hitting set X of size at most k. Chen, Liu, Lu, O'sullivan, and Razgon [17] show that DFVS (and therefore FVST) is *fixed parameter tractable* via a $4^k k! n^{O(1)}$-time algorithm. Kumar and Lokshtanov [49] give a $1.618k + n^{O(1)}$-time algorithm for FVST.

Our Contribution. We simplify Mnich *et al.*'s 7/3-approximation algorithm for FVST [56]. Our new algorithm is based on performing just one round of the *Sherali-Adams hierarchy* [60] on the basic relaxation, and is a significant simplification of [56]. The following is our main theorem. Below, $\mathsf{SA}_r(T, w)$ denotes

both the lower bound on $\mathrm{OPT}(T, w)$ provided by r rounds of the Sherali-Adams hierarchy, and the corresponding linear program (LP).

Theorem 5.1. *Algorithm 3 is a* 7/3*-approximation algorithm for* FVST. *More precisely, the algorithm outputs in polynomial time a feedback vertex set* $X := F \cup F'$ *such that* $w(X) \leq \frac{7}{3}\,\mathsf{SA}_1(T, w) \leq \frac{7}{3}\,\mathrm{OPT}(T, w)$.

Algorithm 3 FVST

Input: A Tournament T and weight function $w : V(T) \to \mathbb{Q}_{>0}$
Output: A feedback vertex set of T of weight at most $\frac{7}{3}\,\mathrm{OPT}(T, w)$

$x \leftarrow$ optimal solution to $\mathsf{SA}_1(T, w)$
$F \leftarrow \{v \in V(T) : x_v \geq 3/7\}$
if F is a FVS for T **then**
  return F
else
  $Z \leftarrow \varnothing$
  repeat
    add to Z all vertices of $T - F - Z$ that are contained in no triangle
    $x \leftarrow$optimal solution to $\mathsf{SA}_0(T - F - Z, w)$
    $F \leftarrow F \cup \{v \in V(T - F - Z) : x_v \geq 1/2\}$
  until $T - F - Z$ is empty or $x_v < 1/2$ for all $v \in V(T - F - Z)$
  $F' \leftarrow$ Layers$(T - F - Z, w, \varnothing, V(T - F - Z))$
  return $F \cup F'$
end if

Theorem 5.1 proves that the integrality gap of the relaxation obtained from the basic one after one round of Sherali-Adams is always at most 7/3. We observe that for random unweighted tournaments $(T, \mathbf{1}_T)$, letting $x_v := 3/7$ for all vertices always gives a feasible solution while the optimum value is with high probability very close to $|V(T)|$, see Corollary 5.13. Thus the worst case integrality gap of SA_1 is precisely 7/3.

Precise definitions will be given later. For now, we give a sketch of Algorithm 3, and explain how it compares with [56].

Comparison to Previous Work. Our approach simplifies both phases of Mnich *et al.*'s algorithm [56]. In our first phase (the rounding phase), instead of considering the $\mathcal{T}_7$-relaxation, we consider $\mathsf{SA}_1(T, w)$. Since the rounding phase is the bottleneck of both algorithms, we obtain a significant speedup in run-time by using a smaller LP. Note that $\mathsf{SA}_1(T, w)$ only has $O(n^4)$ constraints, while the $\mathcal{T}_7$-relaxation can have $\Omega(n^7)$ constraints. Let x be an optimal solution to $\mathsf{SA}_1(T, w)$. If x has a coordinate x_v such that $x_v \geq 3/7$, then we may round up x_v to 1. We continue the rounding using $\mathsf{SA}_0(T, w)$ (the basic relaxation) instead, in order to make sure that when we start the second phase (the layering phase), in the residual tournament, the optimum value is at least one third of the total weight. The whole rounding is done exactly as in [56], except that we replace the $\mathcal{T}_7$-relaxation with $\mathsf{SA}_1(T, w)$.

Then, we proceed to the second phase. The idea follows [56], but with a few important simplifications. We start from a minimum in-degree vertex z and build

a breadth-first search (BFS) in-arborescence that partitions $V(T)$ in layers such that every triangle of T lies within three consecutive layers. Hence, a feedback vertex set for T can be obtained by including every other layer, and, for every layer i that is not picked, a set F_i that is a feedback vertex set for that layer (we call the set F_i a *local solution*).

The main difference with the layering algorithm of [56] is how local solutions are selected. The layers obtained by the algorithm in [56] are $\mathcal{T}_5$-free. This allows them to use $\mathrm{CDZ}(T, w)$ as a subroutine to optimally select local solutions. Our algorithm implements a simpler procedure to partition $V(T)$ in layers. For the first layer produced by the BFS procedure, consisting of all vertices that point to z, we also use $\mathrm{CDZ}(T, w)$. However, for the subsequent layers, a different property is established.

Such layers can be partitioned into two subtournaments, U_i and S_i, that are both acyclic. Hence, we can choose the cheaper of the two subtournaments as our local solution F_i. Whenever the BFS procedure is stuck, that is, when none of the remaining vertices can reach the root node z, the algorithm chooses another root node and starts again (we refer to this as a *fresh start*). Our method gives an improved 9/4-approximation algorithm for FVST on our residual tournament, compared to the 7/3 factor obtained in [56].

5.2. Diagonals and Light Tournaments

Given tournament T, the polytope $P(T)$ is precisely the E3-Vertex Cover formulation $\mathrm{LP}_{\mathrm{HVC}}(\mathcal{H})$ of Chapter 1, where hypergraph $\mathcal{H} := (V(T), \triangle(T))$. We apply Lemmas 3.7 and 3.6 from Section 3.4 in Chapter 3. First we tailor our terminology specifically for FVST.

Recall that an (unordered) pair of vertices ab is a *diagonal* if there are vertices u, v such that $\{u, v, a\} \in \triangle(T)$ and $\{u, v, b\} \in \triangle(T)$. We often will denote a triangle $\{a, b, c\}$ as abc. We remind the reader that a triangle *contains a diagonal* if at least one of its pairs of vertices is a diagonal, and a triangle is *heavy* if it contains at least two diagonals. A tournament T is *heavy* if at least one of its triangles is heavy. If a tournament is not heavy, we say that it is *light*.

We re-frame Lemma 3.6 for our current purposes.

Lemma 5.2. *Let T be a tournament and $x \in \mathsf{SA}_1(T)$. If $x_v < 3/7$ for all $v \in V(T)$, then T is light.*

Next we prove some results connecting light tournaments to the work of [56], which relies on tournaments being $\mathcal{T}_7$-free. Of the three tournaments in $\mathcal{T}_5$, it turns out one of them is heavy (see Figure 5.1b), while the other two are light and can be obtained from each other by reversing the orientation of one arc (see Figure 5.1a). Moreover, although we do not use this fact, we have a computer-assisted proof which shows that 120 out of 121 of the tournaments in $\mathcal{T}_7$ are heavy, and only one is light. Thus, even though a light tournament is not necessarily $\mathcal{T}_7$-free, the property of being light forbids almost all of the tournaments in $\mathcal{T}_7$ as subtournaments.

We now establish further properties of light tournaments. Let $\mathcal{S}_5 \subseteq \mathcal{T}_5$ and $\mathcal{S}_7 \subseteq \mathcal{T}_7$ be the collection of tournaments defined in Figures 5.2 and 5.3, respectively. If T is a tournament, we let $A(T)$ be the set of arcs of T.

(A) Each orientation of ab gives a light tournament in $\mathcal{T}_5$.

(B) The unique heavy tournament in $\mathcal{T}_5$. Note that triangle dec is heavy.

FIGURE 5.1. The three tournaments in $\mathcal{T}_5$.

Lemma 5.3. *Every $S \in \mathcal{S}_5$ is either heavy or has $(u_i, u_{3-i}), (v_i, v_{3-i}) \in A(S)$ for some $i \in [2]$ (where S is labelled as in Figure 5.2).*

PROOF. Suppose $(u_1, u_2), (v_2, v_1) \in A(S)$. Observe that zv_2 is a diagonal since v_1u_1z and $v_1u_1v_2$ are triangles, and v_2u_2 is a diagonal since $v_1u_1v_2$ and $v_1u_1u_2$ are triangles. Because zv_2 and v_2u_2 are both diagonals, we conclude that the triangle v_2u_2z is heavy. The result follows by symmetry. □

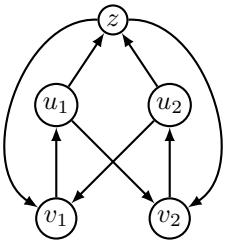


FIGURE 5.2. $\mathcal{S}_5$ is the following subset of $\mathcal{T}_5$, where the missing arcs can be oriented arbitrarily.

Lemma 5.4. *Every $S \in \mathcal{S}_7$ is heavy.*

PROOF. Suppose some $S \in \mathcal{S}_7$ is light, where S is labelled as in Figure 5.3. By symmetry, we may assume that $(u_1, u_2), (u_2, u_3) \in A(S)$. By Lemma 5.3, $(v_1, v_2), (v_2, v_3) \in A(S)$. Therefore, u_2z is a diagonal since v_1u_1z and $v_1u_1u_2$ are triangles, and zv_2 is a diagonal since v_3u_3z and $v_3u_3v_2$ are triangles. We conclude that v_2u_2z is a heavy triangle, which contradicts that S is light. □

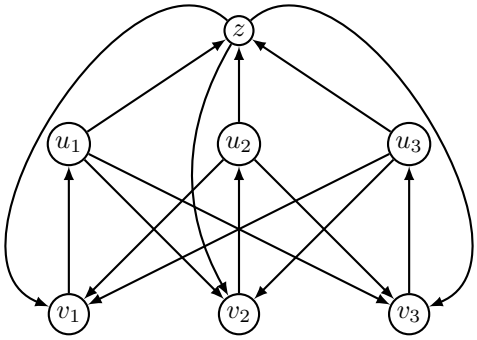


FIGURE 5.3. $\mathcal{S}_7$ is the following subset of $\mathcal{T}_7$, where the missing arcs can be oriented arbitrarily.

5.3. The Layering Procedure

This section proves the correctness of our layering algorithm, see Algorithm 4 below. A high-level, informal overview is as follows. Starting with a minimum in-degree root vertex, proceeding in a BFS manner, partition the vertices of T into layers. Each layer has the property that either it is $\mathcal{T}_5$-free, or is naturally bi-partitioned into two acyclic subsets. We use these properties to obtain a hitting set of weight at most $\frac{3}{4}w(T)$, which, is enough to achieve the desired approximation factor.

More formally, we use Lemmas 5.5 to 5.8 ensure that the algorithm actually produces a feedback vertex set. Lemmas 5.9 to 5.12 prove that Algorithm 4 is a 9/4-approximation algorithm.

Let T be a light tournament with weight function $w : V(T) \to \mathbb{Q}_{\geq 0}$. For $S \subseteq V(T)$, the *in-neighborhood* of S is $N(S) := \{v \notin S \mid (v,u) \in A(T) \text{ for some } u \in S\}$ and $N(u) := N(\{u\})$. For every $z \in V(T)$, define $V_1(z) = \{z\}$, and for $i \geq 2$ let $V_{i+1}(z) := N(\bigcup_{j\in[i]} V_j(z))$. In other words $V_i(z)$ is the set of vertices whose shortest directed path to z has length exactly $i-1$.

Given two sets $S, Z \subseteq V(T)$, we say that Z *in-dominates* S if for every $s \in S$ there is a $z \in Z$ with $(s,z) \in A(T)$. We say that Z *2-in-dominates* S if Z has a subset $Z' \subseteq Z$ with $|Z'| \leq 2$ such that Z' in-dominates S.

We start with a lemma that is key to both the correctness and the performance guarantee of Algorithm 4. See Figure 5.4.

Lemma 5.5. *Let T be a light tournament, z be any vertex of T, and $i \geq 3$. If $\{z_{i-1}, z'_{i-1}\} \subseteq V_{i-1}(z)$ (possibly $z_{i-1} = z'_{i-1}$) 2-in-dominates $V_i(z)$, then $U := N(z_{i-1}) \cap V_i(z)$ and $S := V_i(z) - U$ are triangle-free.*

PROOF. Suppose by contradiction that $u_1u_2u_3$ is a triangle in U. Since $z_{i-1} \in V_{i-1}(z)$ and $i \geq 3$, we have $(z_{i-1}, r) \in A(T)$ for some $r \in V_{i-2}(z)$. Since $U \subseteq V_i(z)$, arcs $(r,u_1),(r,u_2),(r,u_3) \in A(T)$. Thus, ru_iz_{i-1} is a triangle for all $i \in [3]$. It follows that the triangle $u_1u_2u_3$ is heavy since all of its arcs are diagonals, a contradiction. If S has a triangle, we can repeat the same argument. □

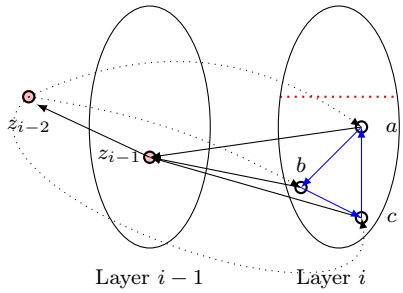


FIGURE 5.4. The blue triangle abc can not exist in a light tournament. All of its arcs are diagonals due to triangles $vz_{i-1}z_{i-2}$ for $v \in \{a, b, c\}$.

Algorithm 4 LAYERS(T, w, U_i, W)

Input: T is a light tournament, $w : V(T) \to \mathbb{Q}_{\geq 0}$, U_i is the current root layer, and W is the set of unseen vertices ($U_0 := \varnothing$ and $W := V(T)$ on the first call). We assume all objects that depend on i (including i itself) to be available throughout subsequent recursive calls.

Output: A feedback vertex set F' of T of weight at most $\frac{3}{4}w(T)$

1: **if** $W = \varnothing$ **then** {Finished}
2: $L_0 \leftarrow \cup_{j \text{ even}} U_j \cup S_j$, $L_1 \leftarrow \cup_{j \text{ odd}} U_j \cup S_j$
3: $F' \leftarrow (\cup_{j=1}^{i} F_{2j}) \cup L_1$ if $w(L_0) \geq w(L_1)$ otherwise $(\cup_{j=0}^{i-1} F_{2j+1}) \cup L_0$
4: return F'
5: **end if**
6: **if** $N(U_i) \neq \varnothing$ **then**
7: $\{z_i, z_i'\} \leftarrow$2-in-dominates$(N(U_i))$ with $w(N(z_i) \cap W) \geq w(N(z_i') \cap W)$
8: $U_{i+1} \leftarrow N(z_i) \cap W$, $S_{i+1} \leftarrow N(z_i') \cap W - U_{i+1}, W \leftarrow W - U_{i+1} - S_{i+1}$
9: $F_{i+1} = S_{i+1}$
10: $i \leftarrow i + 1$
11: return LAYERS(T, w, U_{i+1}, W)
12: **else** {Fresh Start}
13: $z_{i+1} \leftarrow$ choose $z \in W$ with $|N(z) \cap W|$ minimum
14: $U_{i+1} \leftarrow \{z_{i+1}\}$, $U_{i+2} \leftarrow N(z_{i+1}) \cap W, S_{i+1} \leftarrow \varnothing$
15: $F_{i+1} \leftarrow \varnothing$
16: $F_{i+2} \leftarrow \text{CDZ}(U_{i+2}, w)$
17: $W \leftarrow W - (U_{i+1} \cup U_{i+2})$
18: $i \leftarrow i + 2$
19: return LAYERS(T, w, U_{i+2}, W)
20: **end if**

The next lemma ensures that, in the step following a fresh start, vertices z_i, z_i' as on line 7 of Algorithm 4 exist.

Lemma 5.6. *For an arbitrary vertex z in a light tournament T, $V_2(z)$ 2-in-dominates $V_3(z)$.*

PROOF. Let $H = \{h_1, h_2, ..., h_k\} \subseteq V_2(z)$ be an inclusion-wise minimal set that in-dominates $V_3(z)$. Suppose $k \geq 3$. By minimality, for each $h_i \in H$ there must be some $v_i \in V_3(z)$ such that $(v_i, h_i) \in A(T)$ and $(h_i, v_j) \in A(T)$ for all $j \neq i$. Since $(z, v_i) \in A(T)$ for all i, it follows that $T[\{z, h_1, h_2, h_3, v_1, v_2, v_3\}]$ is isomorphic to a tournament in $\mathcal{S}_7$ (see Figure 5.3). Therefore, by Lemma 5.4, $T[\{z, h_1, h_2, h_3, v_1, v_2, v_3\}]$ is heavy, which contradicts that T is light. □

The next lemma ensures that the layer produced after a fresh start is $\mathcal{T}_5$-free, allowing us to use the exact algorithm from [14]. Its proof follows the proof of Lemma 9 of [56], except that we assume that T is light.

Lemma 5.7. *Let z be a minimum in-degree vertex in a light tournament T. Then $V_2(z)$ is $\mathcal{T}_5$-free.*

PROOF. We assume $V_2(z) \neq \varnothing$, otherwise there is nothing to show, and we suppose by contradiction that $X \subseteq V_2(z)$ is a light $\mathcal{T}_5$ (X cannot be heavy as T is light, so X is oriented as in Figure 5.1a). For every $u \in V_2(z)$ there must be a $v \in V_3(z)$ with $(v, u) \in A(T)$. If not then $N(u) \subsetneq V_2(z) = N(z)$, contradicting the minimality of $|N(z)|$. Thus $V_3(z) \neq \varnothing$. Let $H \subseteq V_3(z)$ be an inclusion-wise minimal subset of $V_3(z)$ such that for every $u \in V_2(z)$ there exists $v \in H$ with $(v, u) \in A(T)$. We distinguish cases according to the size of H.

Case 1: $H = \{h\}$. Then hu_iz are triangles for all $u_i \in X$, therefore all arcs in X are diagonals. Since X must contain at least some triangle, this triangle must be heavy since all of its arcs are diagonals, contradicting the fact that T is light.

Case 2: $H = \{f, h\}$. Let $X = \{a, b, c, d, e\}$. We can assume without loss of generality that f points to exactly three vertices of X, for the following reason. If there are less than three, we can swap h with f. If there are more than three, then f must point to a triangle of X (since $T[X]$ is a $\mathcal{T}_5$-subtournament), which would be heavy, arguing as in Case 1.

Notice that ed and ec are diagonals within X (due to triangles ade and adc, bdc and bec, respectively), hence none of ad, ae, bc, be can be diagonals, otherwise one of ade or cbe will be a heavy triangle. This implies that f cannot point to both vertices of any of the latter pairs. From this, one easily derives that f cannot point to a nor b. Hence, (a, f), (b, f), (f, d), (f, e), $(f, c) \in A(T)$, which implies $(h, a), (h, b) \in A(T)$. This forces $(e, h), (c, h), (d, h) \in A(T)$; otherwise, again, one of ad, ae, bc, be is a diagonal. See Figure 5.5 for the orientations we have determined thus far. Notice that adc and fca are triangles, so df is a diagonal. Moreover, since had and zha are triangles, dz is a diagonal. Therefore zfd is a heavy triangle, a contradiction.

Case 3: $|H| \geq 3$. In this case, one can easily find a tournament in $\mathcal{S}_7$ made of z, three vertices of $V_2(z)$ and three vertices of $V_3(z)$, in contradiction with Lemma 5.4 (see the proof of Lemma 5.6).

□

The next lemma ensures that, at each recursive call of Algorithm 4, we can find local solutions by either applying CDZ(T, w) or Lemma 5.5. Together with the previous lemmas, it is enough to conclude that Algorithm 4 outputs a feedback vertex set of our (light) tournament T. This will be formalized in Lemma 5.12.

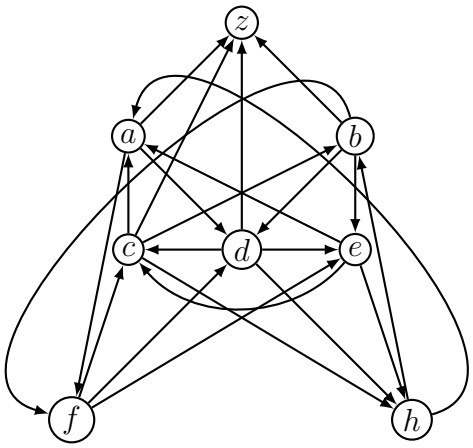


FIGURE 5.5. The orientations determined by the proof of Lemma 5.7.

Lemma 5.8. *Let $U_0, \dots, U_\ell$ and $S_1, \dots, S_\ell$ be the sets produced by Algorithm 4, run on input $(T, w, U_0 := \varnothing, W := V(T))$. For all $i \in [\ell - 1]$, if U_{i+1} is defined as on line 14 of Algorithm 4, then $T[U_{i+1}]$ is $\mathcal{T}_5$-free; and if S_{i+1} is defined as on line 8, then S_{i+1} is a feedback vertex set of $T[U_{i+1} \cup S_{i+1}]$.*

PROOF. If U_{i+1} is defined as on line 14 of Algorithm 4, then U_{i+1} is equal to $V_2(z)$ for some $z \in V(T)$. Therefore, by Lemma 5.7, $T[U_{i+1}] = T[V_2(z)]$ is $\mathcal{T}_5$-free. If S_{i+1} is defined as on line 8, then there is some vertex $z_{i-1} \in V(T)$ such that $U_{i+1} \cup S_{i+1} \subseteq V_3(z_{i-1})$. By Lemma 5.6, $N(U_i)$ 2-in-dominates $U_{i+1} \cup S_{i+1}$. Therefore, by Lemma 5.5, S_{i+1} and U_{i+1} are both triangle-free. Thus, S_{i+1} is a feedback vertex set of $T[U_{i+1} \cup S_{i+1}]$. □

After having shown the correctness of Algorithm 4, we focus on bounding the approximation ratio of its output. This mostly amounts to bounding the weight of the local solutions obtained during the algorithm.

Lemma 5.9. *Let $F_1, \dots, F_\ell$ and $U_0, \dots, U_\ell$ be the sets produced by Algorithm 4, run on input $(T, w, U_0 := \varnothing, W := V(T))$. Then for all $i \in [\ell]$, $w(F_i) \leq w(N(U_{i-1}))/2$.*

PROOF. If $F_i = \emptyset$, then the lemma clearly holds. If F_i is defined as S_i on line 9, then, by construction $w(S_i) \leq w(N(U_{i-1}))/2$. Thus, we may suppose that F_i is defined as $\mathrm{CDZ}(U_i, w)$ on line 16, with $U_i = N(z_{i-1}) \cap W$, and $U_{i-1} = \{z_{i-1}\}$. By Lemma 5.8, $T[U_i]$ is $\mathcal{T}_5$-free, and by [14], F_i is a minimum weight feedback vertex set of $T[U_i]$. Since the all $\frac{1}{3}$-vector is feasible for the basic relaxation of $T[U_i]$, and this relaxation is integral by [14],

$$w(F_i) \leq \frac{1}{3} w(U_i) = \frac{1}{3} w(N(U_{i-1})) \leq \frac{1}{2} w(N(U_{i-1})).$$

□

In the next two lemmas, we assume that Algorithm 3 is run on input (T, w), and we establish properties of the sets defined within the algorithm during the rounding phase (lines -11).

Lemma 5.10. *After the rounding phase of Algorithm 3,*

$$w(F) \leq \frac{7}{3}\big(\mathsf{SA}_1(T,w) - \mathsf{SA}_0(T-F-Z,w)\big).$$

Proof. We proceed by induction on the number of vertices added to F on line 10. In the base case, no vertices get added to F on line 10. Letting x denote the optimal solution to $\mathsf{SA}_1(T,w)$, we get

$$\begin{aligned}
w(F) &\leq \frac{7}{3}\sum_{v\in F} w(v)x_v \\
&= \frac{7}{3}\left(\sum_{v\in V(T)} w(v)x_v - \sum_{v\in V(T-F)} w(v)x_v\right) \\
&\leq \frac{7}{3}\left(\sum_{v\in V(T)} w(v)x_v - \sum_{v\in V(T-F-Z)} w(v)x_v\right) \\
&\leq \frac{7}{3}\big(\mathsf{SA}_1(T,w) - \mathsf{SA}_1(T-F-Z,w)\big) \\
&\leq \frac{7}{3}\big(\mathsf{SA}_1(T,w) - \mathsf{SA}_0(T-F-Z,w)\big)\,.
\end{aligned}$$

Now let F_{before}, Z_{before}, F_{after}, Z_{after} denote the sets F and Z before and after a single iteration of the loop in lines 7–11. Using an argument similar as the one used above (arguing this time with an optimal solution x to $\mathsf{SA}_0(T - F_{\text{before}} - Z_{\text{before}}, w)$), we get

$$\begin{aligned}
w(F_{\text{after}}) - w(F_{\text{before}}) &\leq \frac{1}{2}\big(\mathsf{SA}_0(T - F_{\text{after}} - Z_{\text{after}}, w) - \mathsf{SA}_0(T - F_{\text{before}} - Z_{\text{before}}, w)\big) \\
&\leq \frac{7}{3}\big(\mathsf{SA}_0(T - F_{\text{after}} - Z_{\text{after}}, w) - \mathsf{SA}_0(T - F_{\text{before}} - Z_{\text{before}}, w)\big).
\end{aligned}$$

Hence, assuming that

$$w(F_{\text{before}}) \leq \frac{7}{3}\big(\mathsf{SA}_1(T,w) - \mathsf{SA}_0(T - F_{\text{before}} - Z_{\text{before}}, w)\big)\,,$$

we get

$$w(F_{\text{after}}) \leq \frac{7}{3}\big(\mathsf{SA}_1(T,w) - \mathsf{SA}_0(T - F_{\text{after}} - Z_{\text{after}}, w)\big)\,.$$

The result follows. □

Lemma 5.11. *After the rounding phase of Algorithm 3,*

$$\mathsf{SA}_0(T-F-Z,w) = w(T-F-Z)/3.$$

Proof. The proof is the same as [56, Lemma 6], but for completeness, we include it here. Let $T' = T - F - Z$. Suppose $x_v = 0$ for some $v \in V(T')$. Since every vertex of T' is contained in a triangle, v is in some triangle vab of T. Thus, $x_a + x_b \geq 1$, and so $\max(x_a, x_b) \geq 1/2$, which contradicts that neither a nor b are in F. Thus $x_v > 0$ for all $v \in V(T')$. Let y be an optimal solution to the

dual of $\mathsf{SA}_0(T', w)$. By primal-dual slackness $\sum_{\triangle : u \in \triangle} y_\triangle = w_u$ for all $u \in V(T')$. Therefore,

$$w(V(T')) = \sum_{u \in V(T')} \sum_{\triangle : u \in \triangle} y_\triangle = \sum_{\triangle \in \triangle(T')} y_\triangle \sum_{u \in \triangle} 1 = 3 \sum_{\triangle \in \triangle(T')} y_\triangle = 3\,\mathsf{SA}_0(T', w).$$

□

Lemma 5.12. *Let F' be the set output by Algorithm 4 on input $(T' := T - F - Z, w, U_0 := \varnothing, W = V(T'))$. Then $F \cup F'$ is a feedback vertex set of T and $w(F') \leq \frac{9}{4}\,\mathsf{SA}_0(T', w)$.*

Proof. Algorithm 4 partitions $V(T')$ into layers $S_i \cup U_i$. By symmetry, we may assume that the total weight of the even layers is at least the total weight of the odd layers. That is, $w(L_0) \geq w(L_1)$, using the notation of the algorithm. Then the output F' consists of all odd layers and of the sets F_i, for i even. By construction, F_i is an FVS of $T'[S_i \cup U_i]$, for each i. Since all triangles in T' are contained in three consecutive layers, F' is an FVS of T', and hence $F \cup F'$ is an FVS of T. Moreover, since $w(F_i) \leq w(S_i \cup U_i)/2$ for each i, we have $w(L_0) - w(\cup_{j \text{ even}} F_j) \geq w(V(T'))/4$. Therefore,

$$w(F') = w(V(T')) - (w(L_0) - w(\cup_{j \text{ even}} F_j)) \leq \frac{3}{4} w(V(T')) = \frac{9}{4}\,\mathsf{SA}_0(T', w),$$

where the last equality follows from Lemma 5.11. □

5.4. The Algorithm

Given the results we have already established, it is now easy prove the correctness of Algorithm 3.

Theorem 5.1. *Algorithm 3 is a 7/3-approximation algorithm for* FVST. *More precisely, the algorithm outputs in polynomial time a feedback vertex set $X := F \cup F'$ such that $w(X) \leq \frac{7}{3}\,\mathsf{SA}_1(T, w) \leq \frac{7}{3}\,\mathrm{OPT}(T, w)$.*

Proof. By Lemma 5.12, $F \cup F'$ is a feedback vertex set of T. It remains to show the approximation guarantee. Recall that $F = \{v : x_v \geq 3/7\}$ where x is an optimal solution for $\mathsf{SA}_1(T, w)$. By Lemma 5.10, $w(F) \leq \frac{7}{3}(\mathsf{SA}_1(T, w) - \mathsf{SA}_0(T - F - Z, w))$. Since x restricted to $T - F - Z$ is feasible for $\mathsf{SA}_1(T - F - Z)$, Lemma 5.12 implies that $w(F') \leq \frac{9}{4}\,\mathsf{SA}_0(T - F - Z, w) \leq \frac{7}{3}\,\mathsf{SA}_0(T - F - Z, w)$. Adding these two inequalities yields

$$w(F') + w(F) \leq \frac{7}{3}\,\mathsf{SA}_1(T, w) \leq \frac{7}{3}\,\mathrm{OPT}(T, w).$$

□

Finally, we have the following corollary on the integrality gap of SA_1 for FVST, which we now formally define. If T is a tournament and $w : V(T) \to \mathbb{R}_{\geq 0}$, we let

$$\mathsf{SA}_r(T, w) := \min \left\{ \sum_{v \in V(T)} w(v) x_v \mid x \in \mathsf{SA}_r(T) \right\}.$$

The (worst case) *integrality gap* of SA_r for FVST is

$$\sup_{(T,w)} \frac{\mathrm{OPT}(T,w)}{\mathsf{SA}_r(T,w)}$$

where the supremum is taken over all tournaments T and all weight functions $w : V(T) \to \mathbb{Q}_{\geq 0}$.

Corollary 5.13. *The integrality gap of* SA_1 *for* FVST *is exactly* $7/3$.

Proof. The fact that the integrality gap of SA_1 for FVST is at most $7/3$ follows from Theorem 5.1. For the other inequality, note that for every tournament T, the all $\frac{3}{7}$-vector is feasible for $\mathsf{SA}_1(T)$ (by setting $x_{uv} = \frac{1}{7}$ for all uv). On the other hand, we can generate a random n-node tournament by choosing the orientation of each edge of K_n with probability $1/2$. One can then show via the probabilistic method that, for a random n-node tournament T, $\mathrm{OPT}(T, \mathbf{1}_T) = n - O(\log n)$ with high probability [61]. □

5.5. Concluding Remarks

In this chapter we gave a simple $7/3$-approximation algorithm for FVST, based on performing just one round of the Sherali-Adams hierarchy on the basic relaxation. It is a bit of a miracle that $\mathsf{SA}_1(T)$ already "knows" a remarkable amount of structure about feedback vertex sets in tournaments. It is unclear how much more knowledge $\mathsf{SA}_r(T)$ acquires as r increases, but our approach naturally begs the question of whether performing a constant number of rounds of Sherali-Adams leads to a 2-approximation for FVST. This would solve the main open question from [54].

We suspect that performing more rounds does improve the approximation ratio, but the analysis becomes more complicated. Indeed, it could be that $\mathsf{SA}_2(T)$ already gives a $9/4$-approximation algorithm for FVST, since the layering procedure has a $9/4$-approximation factor. Note that $\mathsf{SA}_2(T)$ does contain new inequalities such as $x_a + x_b + x_c \geq 1 + x_{\mathrm{AB}} + x_{ac} + x_{cb} - x_{abc}$, for all $abc \in \triangle(T)$, which may be exploited.

In the next chapter we turn to *cluster vertex deletion*, and show that one round of Sherali-Adams has an integrality gap of $5/2$, and for every $\epsilon > 0$ there exists $r \in \mathbb{N}$ such that r rounds of Sherali-Adams has integrality gap at most $2 + \epsilon$. Indeed, this chapter can be seen as unifying the approaches of [56] and some of the polyhedral results of [3].

CHAPTER 6

Cluster Vertex Deletion

In Chapter 4 we found results for Claw-VD using the local ratio technique of Section 3.2 of Chapter 3. In Chapter 5 we followed the iterative rounding framework of Section 3.3 to design an algorithm for FVST.

Here we study CVD from both perspectives. This chapter is based on Aprile, Drescher, Fiorini, Huynh [3] which gives the first 2-approximation algorithm for the cluster vertex deletion problem. This is tight, since as mentioned in Chapter 1, approximating the problem within any constant factor smaller than 2 is UGC-hard.

Given a graph G, we say vertices $u, v \in V(G)$ are *true twins* if $N[u] = N[v]$, and if a graph contains no true twins it is *twin-free*. Previous approximation algorithms, see Fiorini et al. [28], reduce the problem by managing true twins and then applying the local ratio technique of Chapter 3 Section 3.2. Our algorithm also follows this methodology but adds a new recursive method for finding good subgraphs in the second neighborhood of any vertex of the twin-free input graph.

From the polyhedral perspective we prove almost matching upper and lower bounds on how well linear programming relaxations can approximate CVD.

In Section 6.1 we give a more detailed history of CVD, state our 2-approximation algorithm, and discuss how it fits into the landscape of previous work. We conclude the section with a high level overview of the proof. Section 6.2 covers the key algorithmic step of finding a 2-good subgraph. Section 6.3 shows that the run-time is $O(n^4)$. In Section 6.4 we study the problem from a polyhedral perspective. Finally in Section 6.5 we give some open questions.

6.1. Overview

Recall that P_k denotes the path on k vertices. Given graph G, we let $P_k(G)$ be the set of all induced paths in G of length k.

First recall a few definitions. A *cluster graph* is a graph that is a disjoint union of complete graphs. Let G be any graph. A set $X \subseteq V(G)$ is called a *hitting set* if $G - X$ is a cluster graph. Given a graph G and (vertex) weight function $w : V(G) \to \mathbb{Q}_{\geqslant 0}$, the *cluster vertex deletion* problem (CVD) asks to find a hitting set X whose weight $w(X) := \sum_{v \in X} w(v)$ is minimum. We denote by $\mathrm{OPT}(G, w)$ the minimum weight of a hitting set.

If G and H are two graphs, we say that G *contains* H if some *induced* subgraph of G is isomorphic to H. Otherwise, G is said to be *H-free*. Denoting by P_k the path on k vertices, we easily see that a graph is a cluster graph if and only if it is P_3-free. Hence, $X \subseteq V(G)$ is a hitting set if and only if X contains a vertex from each induced P_3.

CVD has applications in graph modeled data clustering in which an unknown set of samples may be contaminated. An optimal solution for CVD can recover a clustered data model, retaining as much of the original data as possible [42].

As we have seen in Part 1, CVD can be modelled as an instance of E3-Vertex Cover, and therefore "textbook" 3-approximation algorithms are available. Moreover, the problem has an approximation-preserving reduction from Vertex Cover, hence obtaining a $(2-\epsilon)$-approximation algorithm for some $\epsilon > 0$ would contradict either the Unique Games Conjecture or P $\neq$ NP.

The first non-trivial approximation algorithm for CVD was a 5/2-approximation due to You, Wang and Cao [68]. Shortly afterward, Fiorini, Joret and Schaudt gave a 7/3-approximation [28], and subsequently a 9/4-approximation [28].

Main Result. In this chapter, we close the gap between 2 and $9/4 = 2.25$ and prove the following *tight* result.

Theorem 6.1. CVD *has a* 2*-approximation algorithm.*

All previous approximation algorithms for CVD are based on the local ratio technique, which we have introduced in Chapter 3. However, the approach we use here significantly differs from previous algorithms in its crucial step, namely, Step 14. This step specifies how to find a 2-good subgraph, the rest of the algorithm follows the form given in Chapter 3 Section 3.2 Algorithm 2. See Theorem 6.2 below.

Recall the definition of *good* and *strongly good* subgraphs from Section 3.2 of Chapter 3.

If we cannot find a strongly α-good induced subgraph in G, we will find an induced subgraph H that has special vertex v_0 whose neighborhood $N(v_0)$ is entirely contained in H, and a weight function $w_H : V(H) \to \mathbb{Z}_{\geqslant 0}$ such that $w_H(v) \geqslant 1$ for all vertices v in the closed neighborhood $N[v_0]$ and $w_H(V(H)) \leqslant \alpha \cdot \mathrm{OPT}(H, w_H) + 1$. Since no inclusion-wise minimal hitting set X can contain all the vertices of $N[v_0]$, $w_H(X \cap V(H)) \leqslant w_H(V(H)) - 1 \leqslant \alpha \cdot \mathrm{OPT}(H, w_H)$ and so (H, w_H) is α-good in G. We say that (H, w_H) (sometimes simply H) is *centrally* α-good (in G) *with respect to* v_0. Moreover, we call v_0 the *root vertex*.

In order to illustrate these ideas, recall the two examples of Figure 3.3. First, let H be a C_4 (that is, a 4-cycle) contained in G and $\mathbf{1}_H$ denote the unit weight function on $V(H)$. Then $(H, \mathbf{1}_H)$ is strongly 2-good, since $\sum_{v \in V(H)} \mathbf{1}_H(v) = 4 = 2\,\mathrm{OPT}(H, \mathbf{1}_H)$. Second, let H be a P_3 contained in G, starting at a vertex v_0 that has degree-1 in G. Then $(H, \mathbf{1}_H)$ is centrally 2-good with respect to v_0, but it is not strongly 2-good.

Each time we find a 2-good weighted induced subgraph in G, the local ratio technique allows us to recurse on an induced subgraph G' of G in which at least one vertex of H is deleted from G. For example, the 2-good induced subgraphs mentioned above allow us to reduce to input graphs G that are C_4-free and have minimum degree at least 2.

In order to facilitate the search for α-good induced subgraphs, it greatly helps to assume that G is *twin-free*. That is, G has no two distinct vertices u, u' such that $uu' \in E(G)$ and for all $v \in V(G - u - u')$, $uv \in E(G)$ if and only

Algorithm 5 CLUSTER-VD-APX(G, w)

Input: (G, w) a weighted graph
Output: X a minimal hitting set of G

1: **if** G is a cluster graph **then**
2: $\quad X \leftarrow \varnothing$
3: **else if** there exists $u \in V(G)$ with $w(u) = 0$ **then**
4: $\quad G' \leftarrow G - u$
5: $\quad w'(v) \leftarrow w(v)$ for $v \in V(G')$
6: $\quad X' \leftarrow$ CLUSTER-VD-APX(G', w')
7: $\quad X \leftarrow X'$ if X' is a hitting set of G; $X \leftarrow X' \cup \{u\}$ otherwise
8: **else if** there exist true twins $u, u' \in V(G)$ **then**
9: $\quad G' \leftarrow G - u'$
10: $\quad w'(u) \leftarrow w(u) + w(u')$; $w'(v) \leftarrow w(v)$ for $v \in V(G' - u)$
11: $\quad X' \leftarrow$ CLUSTER-VD-APX(G', w')
12: $\quad X \leftarrow X'$ if X' does not contain u; $X \leftarrow X' \cup \{u'\}$ otherwise
13: **else**
14: $\quad$ find a weighted induced subgraph (H, w_H) that is 2-good in G
15: $\quad \lambda^* \leftarrow \max\{\lambda \mid \forall v \in V(H) : w(v) - \lambda w_H(v) \geqslant 0\}$
16: $\quad G' \leftarrow G$
17: $\quad w'(v) \leftarrow w(v) - \lambda^* w_H(v)$ for $v \in V(H)$; $w'(v) \leftarrow w(v)$ for $v \in V(G) - V(H)$
18: $\quad X \leftarrow$ CLUSTER-VD-APX(G', w')
19: **end if**
20: return X

if $u'v \in E(G)$. Equivalently, u and u' are such that $N[u] = N[u']$. Two such vertices u, u' are called *true twins*. As in [28], our algorithm reduces G whenever it has a pair of true twins u, u' (see Steps 8–12). The idea is simply to add the weight of u' to that of u and delete u'.

The crux of the analysis of the algorithm, especially Step 14, relies on the following structural result. Below, we denote by $N_{\leqslant i}[v]$ (resp. $N_i(v)$) the set of vertices at distance at most (resp. equal to) i from vertex v (we omit the subscript if $i = 1$).

THEOREM 6.2. *Let G be a twin-free graph, let v_0 be any vertex of G, and let H be the subgraph of G induced by $N_{\leqslant 2}[v_0]$. There exists a weight function $w_H : V(H) \to \mathbb{Z}_{\geqslant 0}$ such that (H, w_H) is either strongly 2-good, or centrally 2-good in G with respect to v_0. Moreover, w_H can be constructed in polynomial time.*

We also study CVD from the polyhedral point of view. In particular we investigate how well linear programming (LP) relaxations can approximate the optimal value of CVD. As in [5, 13, 16], we use the notation of Chapter 3 which, by design, allows for extended formulations.

Letting $\mathcal{P}_3(G)$ denote the collection of all vertex sets $\{u, v, c\}$ that induce a P_3 in G, we define $P(G) := \{x \in [0,1]^{V(G)} \mid \forall \{u, v, c\} \in \mathcal{P}_3(G) : x_u + x_v + x_c \geqslant 1\}$. We let $\mathsf{SA}_r(G)$ denote the relaxation obtained from $P(G)$ by applying r rounds of the Sherali-Adams hierarchy [60]. If a weight function $w : V(G) \to \mathbb{Q}_{\geqslant 0}$ is

provided, we let $\mathsf{SA}_r(G,w) := \min\{\sum_{v\in V(G)} w(v)x_v \mid x \in \mathsf{SA}_r(G)\}$ denote the optimum value of the corresponding linear programming relaxation.

It is standard to show that the straightforward LP relaxation $P(G)$ has worst case integrality gap equal to 3 (by *worst case*, we mean that we take the supremum over all graphs G). Indeed, for a random graph G in $G(n,1/2)$, $\mathrm{OPT}(G,\mathbf{1}_G) = n - O(\log^2 n)$ with high probability, while $\mathrm{LP}(G,\mathbf{1}_G) \leqslant n/3$. In other words, the worst case integrality gap after applying zero rounds of the Sherali-Adams hierarchy is 3. To see this note that Bollobás,and Erdös showed [8] that with high probability the independence and clique numbers of G are logarithmic. That is $\omega(G) = O(\log n), \alpha(G) = O(\log n)$. If X is a minimum sized hitting set for the CVD problem then it follows that with high probability $G - X$ is a collection of $O(\log n)$ cliques each of size $O(\log n)$. Thus with high probability $|X| = n - O(\log^2 n)$.

On the positive side, we show how applying one round of the *Sherali-Adams hierarchy* [60] gives a relaxation with integrality gap at most $5/2 = 2.5$, see Theorem 6.16. To complement this, we prove that the worst case integrality gap of the relaxation is precisely $5/2$, see Theorem 6.17. Then, by relying on Theorem 6.2, we show that the integrality gap decreases to $2+\epsilon$ after applying $\mathrm{poly}(1/\epsilon)$ rounds, see Theorem 6.18.

On the negative side, applying known results about Vertex Cover [5], we show that no polynomial-size LP relaxation of CVD can have integrality gap at most $2-\epsilon$ for some $\epsilon > 0$. This result is unconditional: it does not rely on P $\neq$ NP nor the Unique Games Conjecture.

Comparison to previous works. We now revisit all previous approximation algorithms for CVD [28, 28, 68]. The presentation given here slightly departs from [28, 68], and explains in a unified manner what is the bottleneck in each of the algorithms.

Fix $k \in \{3,4,5\}$, and let $\alpha := (2k-1)/(k-1)$. Notice that $\alpha = 5/2$ if $k = 3$, $\alpha = 7/3$ if $k = 4$ and $\alpha = 9/4$ if $k = 5$. In [28, Lemma 3], it is shown that if a twin-free graph G contains a k-clique, then one can find an induced subgraph H containing the k-clique and a weight function w_H such that (H, w_H) is strongly α-good.

Therefore, in order to derive an α-approximation for CVD, one may assume without loss of generality that the input graph G is twin-free and has no k-clique. Let v_0 be a maximum degree vertex in G, and let H denote the subgraph of G induced by $N_{\leqslant 2}[v_0]$. In [28], it is shown by a tedious case analysis that one can construct a weight function w_H such that (H, w_H) is 2-good in G, using the fact that G has no k-clique.

The simplest case occurs when $k = 3$. Then $N(v_0)$ is a stable set. Letting $w_H(v_0) := d(v_0) - 1$, $w_H(v) := 1$ for $v \in N(v_0)$ and $w_H(v) := 0$ for the other vertices of H, one easily sees that (H, w_H) is (centrally) 2-good in G . For higher values of k, one has to work harder.

In this paper, we show that one can *always*, and in polynomial time, construct a weight function w_H on the vertices at distance at most 2 from v_0 that makes (H, w_H) 2-good in G, provided that G is twin-free, see Theorem 6.2. This result

was the main missing ingredient in previous approaches, and single-handedly closes the approximability status of CVD.

Other related works. CVD has also been widely studied from the perspective of *fixed parameter tractability*. Given a graph G and parameter k as input, the task is to decide if G has a hitting set X of size at most k. A $2^k n^{O(1)}$-time algorithm for this problem was given by Hüffner, Komusiewicz, Moser, and Niedermeier [42]. This was subsequently improved to a $1.911^k n^{O(1)}$-time algorithm by Boral, Cygan, Kociumaka, and Pilipczuk [9], and a $1.811^k n^{O(1)}$-time algorithm by Tsur [64]. By the general framework of Fomin, Gaspers, Lokshtanov, and Saurabh [30], these parametrized algorithms can be transformed into exponential algorithms which compute the size of a minimum hitting set for G exactly, the fastest of which runs in time $O(1.488^n)$.

For polyhedral results, [41] gives some facet-defining inequalities of the CVD polytope, as well as complete linear descriptions for special classes of graphs.

For unit weights, note that CVD is equivalent to the problem of deleting as few elements as possible from a *symmetric* relation to obtain a transitive relation, while FVST is equivalent to the problem of deleting as few elements as possible from an *antisymmetric* and complete relation to obtain a transitive relation.

In Chapter 5 FVST we show that one round of the Sherali-Adams hierarchy actually provides a 7/3-approximation. This is in contrast with Theorem 6.17.

Among other related covering and packing problems, Fomin, Le, Lokshtanov, Saurabh, Thomassé, and Zehavi [31] studied both CVD and FVST from the *kernelization* perspective. They proved that the unweighted versions of both problems admit subquadratic kernels: $O(k^{\frac{5}{3}})$ for CVD and $O(k^{\frac{3}{2}})$ for FVST.

Overview of the proof. We give a sketch of the proof of Theorem 6.2. If the subgraph induced by $N(v_0)$ contains a hole (that is, an induced cycle of length at least 4), then H contains a wheel, which makes H strongly 2-good, see Lemma 6.3. If the subgraph induced by $N(v_0)$ contains an induced $2P_3$ (that is, two disjoint copies of P_3 with no edges between them), then H is strongly 2-good, see Lemma 6.4. This allows us to reduce to the case where the subgraph induced by $N(v_0)$ is chordal and $2P_3$-free.

Lemma 6.7 then gives a direct construction of a weight function w_H which certifies that (H, w_H) is centrally 2-good, provided that the subgraph induced by $N[v_0]$ is *twin-free*[1]. This is the crucial step of the proof. It serves as the base case of the induction. Here, we use a slick observation due to Lokshtanov [53]: since the subgraph induced by $N(v_0)$ is chordal and $2P_3$-free, it has a hitting set that is a clique. In a previous version, our proof of Theorem 6.2 was slightly more complicated.

We show inductively that we can reduce to the case where the subgraph induced by $N[v_0]$ is twin-free. The idea is to delete vertices from H to obtain a smaller graph H', while preserving certain properties, and then compute a suitable weight function w_H for H, given a suitable weight function $w_{H'}$ for H'. We delete vertices at distance 2 from v_0. When this creates true twins in H, we

[1] *twin-free* is *not* a hereditary property. In general, if G is twin-free and $H \subseteq G$, we can not conclude that H is twin-free.

delete one vertex from each pair of true twins. At the end, we obtain a twin-free induced subgraph of $H[N[v_0]]$, which corresponds to our base case.

We conclude the introduction with a brief description of the different sections of the chapter. Section 6.2 is entirely devoted to the proof of Theorem 6.2. The proof of Theorem 6.1 is given in Section 6.3, together with a complexity analysis of Algorithm 5. Section 6.4 presents our polyhedral results. A conclusion is given in Section 6.5. There, we state a few open problems for future research.

6.2. Finding 2-good induced subgraphs

The goal of this section is to prove Theorem 6.2. Our proof is by induction on the number of vertices in $H := G[N_{\leq 2}[v_0]]$. First, we quickly show that we can assume that the subgraph induced by $N(v_0)$ is chordal and $2P_3$-free. Using this, we prove the theorem in the particular case where the subgraph induced by $N[v_0]$ is twin-free. Finally, we prove the theorem in the general case by showing how to deal with true twins.

Restricting to chordal, $2P_3$-free neighborhoods. As pointed out earlier in the introduction, 4-cycles are strongly 2-good. This implies that wheels of order 5 are strongly 2-good (putting a zero weight on the apex). Recall that a *wheel* is a graph obtained from a cycle by adding an apex vertex (called the *center*). We now show that *all* wheels of order at least 5 are strongly 2-good. This allows our algorithm to restrict to input graphs such that the subgraph induced on each neighborhood is chordal. In a similar way, we show that we can further restrict such neighborhoods to be $2P_3$-free.

Lemma 6.3. *Let $H := W_k$ be a wheel on $k \geqslant 5$ vertices and center v_0, let $w_H(v_0) := k-5$ and $w_H(v) := 1$ for $v \in V(H - v_0)$. Then (H, w_H) is strongly 2-good.*

Proof. Notice that $\mathrm{OPT}(H, w_H) \geqslant k-3$ since a hitting set either contains v_0 and at least 2 more vertices, or does not contain v_0 but contains $k-3$ other vertices. Hence, $\sum_{v \in V(H)} w_H(v) = k-5+k-1 = 2(k-3) \leqslant 2\,\mathrm{OPT}(H, w_H)$. □

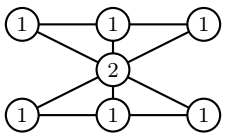


Figure 6.1. v_0 along with $2P_3$ is strongly 2-good.

Lemma 6.4. *Let H be the graph obtained from $2P_3$ by adding a universal vertex v_0. Let $w_H(v_0) := 2$ and $w_H(v) := 1$ for $v \in V(H - v_0)$. Then (H, w_H) is strongly 2-good.*

Proof. $\mathrm{OPT}(H, w_H) \geqslant 4$, see Figure 6.1. Thus, $\sum_{v \in V(H)} w_H(v) = 8 \leqslant 2\,\mathrm{OPT}(H, w_H)$. □

The twin-free case. Throughout this section, we assume that H is a *twin-free* graph with a universal vertex v_0 such that $H - v_0$ is chordal and $2P_3$-free. Our goal is to construct a weight function w_H that certifies that H is centrally 2-good.

It turns out to be easier to define the weight function on $V(H - v_0) = N(v_0)$ first, and then adjust the weight of v_0. This is the purpose of the next lemma. Below, $\omega(G, w)$ denotes the maximum weight of a clique in weighted graph (G, w).

Lemma 6.5. *Let H be a graph with a universal vertex v_0 and $H' := H - v_0$. Let $w_{H'} : V(H') \to \mathbb{Z}_{\geqslant 1}$ be a weight function such that*

(i) $w_{H'}(V(H')) \geqslant 2\omega(H', w_{H'})$ *and*
(ii) $\mathrm{OPT}(H', w_{H'}) \geqslant \omega(H', w_{H'}) - 1$.

Then we can extend $w_{H'}$ to a function $w_H : V(H) \to \mathbb{Z}_{\geqslant 1}$ by finding a weight $w_H(v_0)$, such that $w_H(V(H)) \leqslant 2\,\mathrm{OPT}(H, w_H) + 1$. In other words, (H, w_H) is centrally 2-good with respect to v_0.

Proof. Notice that

$$\mathrm{OPT}(H, w_H) = \min(w_H(v_0) + \mathrm{OPT}(H', w_{H'}), w_{H'}(V(H')) - \omega(H', w_{H'}))$$

since a hitting set of H that does not contain v_0 must be the complement of a clique in $H - v_0$.

Now, choose $w_H(v_0) \in \mathbb{Z}_{\geqslant 1}$ such that

$$\begin{aligned} \max(1, w_{H'}(V(H')) - 2\,\mathrm{OPT}(H', w_{H'}) - 1) &\leqslant w_H(v_0) \\ (\star) \qquad\qquad &\leqslant w_{H'}(V(H')) - 2\omega(H', w_{H'}) + 1. \end{aligned}$$

It is easy to check that such $w_H(v_0)$ exists because the lower bound in $(\star)$ is at most the upper bound, thanks to conditions (i) and (ii).

This choice satisfies $w_H(V(H)) \leqslant 2\,\mathrm{OPT}(H, w_H) + 1$ since it holds both in case $\mathrm{OPT}(H, w_H) = w_H(v_0) + \mathrm{OPT}(H', w_{H'})$ by the upper bound on $w_{H'}(V(H'))$ given by $(\star)$, and in case $\mathrm{OPT}(H, w_H) = w_{H'}(V(H')) - \omega(H', w_{H'})$ by the upper bound on $w_H(v_0)$ given by $(\star)$. □

We call a hitting set X of a graph G a *hitting clique* if X is also a clique. We denote the set of maximal cliques of chordal graph G by $\mathcal{C}(G)$. The *clique intersection graph* of G has vertices $\mathcal{C}(G)$, and edges KK' for every $K, K' \in \mathcal{C}(G)$ such that $K \cap K'$ is not empty and we say the weight of KK' is $|K \cap K'|$. A *clique tree* of G is a maximum weight spanning tree of the clique intersection graph (Bernstein and Goodman [6]). Every clique tree of G satisfies the *intersection property*: for every two maximal cliques K, K', the intersection $K \cap K'$ is contained in every clique of the K–K' path in T (see Blair and Peyton [7]).

Lemma 6.6. *Every chordal, $2P_3$-free graph contains a hitting clique.*

Proof. Let G be a chordal, $2P_3$-free graph. Since G is chordal, G admits a *clique tree* T. In T, the vertices are the maximal cliques of G and, for every two maximal cliques K, K', the intersection $K \cap K'$ is contained in every clique of the K–K' path in T. For an edge $e := KK'$ of T, Let T_1 and T_2 be the components of $T - e$ and G_1 and G_2 be the subgraphs of G induced by the union of all the cliques in T_1 and T_2, respectively. It is easy to see that deleting $K \cap K'$ separates

G_1 from G_2 in G. Now, since G is $2P_3$-free, at least one of $G'_1 := G_1 - (K \cap K')$ or $G'_2 := G_2 - (K \cap K')$ is a cluster graph. If both G'_1 and G'_2 are cluster graphs, we are done since $K \cap K'$ is the desired hitting clique. Otherwise, if G'_i is not a cluster graph, then we can orient e towards T_i. Proceeding this way, we define an orientation of T, which must have a sink K_0. But then removing K_0 from G leaves a cluster graph, and we are done. □

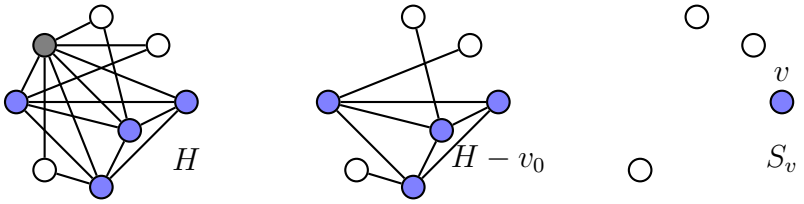


FIGURE 6.2. Here H is twin-free, v_0 is the gray vertex and the blue vertices form a hitting clique K_0 for $H - v_0$, which is chordal and $2P_3$-free. For $v \in K_0$, the set S_v defined as in the proof of Lemma 6.7 consists of the unique maximal independent set containing v. We obtain $w_H = (\mathbf{6}, 1, 1, 1, 1, 3, 3, 3)$, which is easily seen to be centrally 2-good with respect to v_0.

We are ready to prove the base case for Theorem 6.2.

Lemma 6.7. *Let H be a twin-free graph with universal vertex v_0 such that $H - v_0$ is chordal and $2P_3$-free. There exists a weight function w_H such that (H, w_H) is centrally 2-good with respect to v_0. Moreover, w_H can be found in time polynomial in the size of H.*

PROOF. By Lemma 6.6, some maximal clique of $H - v_0$, say K_0, is a hitting set.

We claim that there is a family of stable sets $\mathcal{S} = \{S_v \mid v \in K_0\}$ of $H - v_0$ satisfying the following properties:

(P1) every vertex of $H - v_0$ is contained in some S_v;
(P2) for each $v \in K_0$, S_v contains v and at least one other vertex;
(P3) for every two distinct vertices $v, v' \in K_0$, $H[S_v \cup S_{v'}]$ contains a P_3.

Before proving the claim, we prove that it implies the lemma, making use of Lemma 6.5. Consider the weight function $w_{H'} := \sum_{v \in K_0} \chi^{S_v}$ on the vertices of $H' := H - v_0$ defined by giving to each vertex u a weight equal to the number of stable sets S_v that contain u (see Figure 6.2). Let us show that $w_{H'}$ satisfies the conditions of Lemma 6.5 and can therefore be extended to a weight function w_H on $V(H)$ such that (H, w_H) is centrally 2-good with respect to v_0.

First, by (P1), we have $w_{H'}(u) \in \mathbb{Z}_{\geqslant 1}$ for all $u \in V(H')$. Second, condition (i) of Lemma 6.5 follows from (P2) since each stable set S_v contributes at least two units to $w_{H'}(V(H'))$ and at most one unit to $\omega(H', w_{H'})$. Third, (P3) implies that that every hitting set of H' meets every stable set S_v, except possibly one. Hence, $\mathrm{OPT}(H', w_{H'}) \geqslant |K_0| - 1$. Also, every clique of H' meets every stable set S_v in at most one vertex, implying that $\omega(H', w_{H'}) \leqslant |K_0|$, and equality holds since $w_{H'}(K_0) = |K_0|$. Putting the last two observations together, we see that

$\mathrm{OPT}(H', w_{H'}) \geqslant |K_0| - 1 = \omega(H', w_{H'}) - 1$ and hence condition (ii) of Lemma 6.5 holds.

Now, we prove that our claim holds. Let $K_1, \ldots, K_t$ denote the clusters (maximal cliques) of cluster graph $H - v_0 - K_0$. For $i \in [t]$, consider the submatrix A_i of the adjacency matrix $A(H)$ with rows indexed by the vertices of K_0 and columns indexed by the vertices of K_i.

Notice that A_i contains neither $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ nor $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ as a submatrix, as this would give a C_4 contained in $H - v_0$, contradicting the chordality of $H - v_0$. Hence, after permuting its rows and columns if necessary, A_i can be assumed to be staircase-shaped. That is, every row of A_i is nonincreasing and every column nondecreasing. Notice also that A_i does not have two equal columns, since these would correspond to two vertices of K_i that are true twins.

For each K_i that is not complete to $v \in K_0$, define $\varphi_i(v)$ as the vertex $u \in K_i$ whose corresponding column in A_i is the *first* containing a 0 in row v. Now, for each v, let S_v be the set including v, and $\varphi_i(v)$, for each K_i that is not complete to v.

Because K is maximal, no vertex $u \in K_i$ is complete to K_0. Since no two columns of A_i are identical, we must have $u = \varphi_i(v)$ for some $v \in K_0$. This proves (P1).

Notice that $v \in S_v$ by construction and that $|S_v| \geqslant 2$ since otherwise, v would be universal in H and thus a true twin of v_0. Hence, (P2) holds.

Finally, consider any two distinct vertices $v, v' \in K_0$. Since v, v' are not true twins, the edge vv' must be in a P_3 contained in $H - v_0$. Assume, without loss of generality, that there is a vertex $u \in K_i$ adjacent to v and not to v' for some $i \in [t]$. Then $\{v, v', \varphi_i(v')\}$ induces a P_3 contained in $H[S_v \cup S_{v'}]$, proving (P3). This concludes the proof of the claim.

We remark that the weight function w_H can be computed in polynomial time. We first obtain efficiently the collection $\mathcal{S}$, hence the restriction of w_H to H', and then just let $w_H(v_0) := w_{H'}(V(H')) - 2\omega(H', w_{H'}) + 1 = w_{H'}(V(H')) - 2|K_0| + 1$. This sets $w_H(v_0)$ to its upper bound in $(\star)$, see the proof of Lemma 6.5. □

Handling true twins. We start with an extra bit of terminology relative to true twins. Let G be a twin-free graph, and $v_0 \in V(G)$. Suppose that u, u' are true twins in $G[N[v_0]]$. Since G is twin-free, there exists a vertex v that is adjacent to exactly one of u, u'. We say that v is a *distinguisher* for the edge uu' (or for the pair $\{u, u'\}$). Notice that either $uu'v$ or $u'uv$ is an induced P_3. Notice also that v is at distance either 1 or 2 from v_0.

Now, consider a graph H with a special vertex $v_0 \in V(H)$ (the root vertex) such that

(H1) every vertex is at distance at most 2 from v_0, and
(H2) every pair of vertices that are true twins in $H[N[v_0]]$ has a distinguisher.

Let v be any vertex that is at distance 2 from v_0. Consider the equivalence relation $\equiv$ on $N[v_0]$ with $u \equiv u'$ whenever $u = u'$ or u, u' are true twins in $H - v$. Observe that the equivalence classes of $\equiv$ are of size at most 2 since, if u, u', u'' are distinct vertices with $u \equiv u' \equiv u''$, then v cannot distinguish every edge of

the triangle on u, u' and u''. Hence, two of these vertices are true twins in H, which contradicts (H2).

From what precedes, the edges contained in $N[v_0]$ that do not have a distinguisher in $H - v$ form a matching $M := \{u_1u'_1, \dots, u_ku'_k\}$ (possibly, $k = 0$). Let H' denote the graph obtained from H by deleting v and exactly one endpoint from each edge of M. Notice that the resulting subgraph is the same, up to isomorphism, no matter which endpoint is chosen.

The lemma below states how we can obtain a weight function w_H that certifies that H is centrally 2-good from a weight function $w_{H'}$ that certifies that H' is centrally 2-good. It is inspired by [28, Lemma 3]. See Figure 6.3 for an example.

Lemma 6.8. *Let H be any graph satisfying (H1) and (H2) for some $v_0 \in V(H)$. Let $v \in N_2(v_0)$. Let $M := \{u_1u'_1, \dots, u_ku'_k\}$ be the matching formed by the edges in $N[v_0]$ whose unique distinguisher is v, where $u'_i \neq v_0$ for all i (we allow the case $k = 0$). Let $H' := H - u'_1 - \cdots - u'_k - v$. Given a weight function $w_{H'}$ on $V(H')$, define a weight function w_H on $V(H)$ by letting $w_H(u'_i) := w_{H'}(u_i)$ for $i \in [k]$, $w_H(v) := \sum_{i=1}^k w_{H'}(u_i) = \sum_{i=1}^k w_H(u'_i)$, and $w_H(u) := w_{H'}(u)$ otherwise. First, H' satisfies (H1) and (H2). Second, if $(H', w_{H'})$ is centrally 2-good, then (H, w_H) is centrally 2-good.*

Proof. For the first part, notice that H' satisfies (H1) by our choice of v. Indeed, deleting v does not change the distance of the remaining vertices from v_0. By definition, H' satisfies (H2).

For the second part, notice that $w_H(u) \geqslant 1$ for all $u \in N[v_0]$ since $w_{H'}(u) \geqslant 1$ for all $u \in N[v_0] - \{v, u'_1, \dots, u'_k\}$. To argue that $w_H(V(H)) \leqslant 2\,\mathrm{OPT}(H, w_H) + 1$, one can check that any hitting set of H must either contain v or at least one endpoint of each edge $u_iu'_i \in M$. Hence $\mathrm{OPT}(H, w_H) \geqslant \sum_{i=1}^k w_{H'}(u_i) + \mathrm{OPT}(H', w_{H'})$.

Since $(H, w_{H'})$ is centrally 2-good, $w_{H'}(V(H')) \leqslant 2\,\mathrm{OPT}(H', w_{H'}) + 1$. It follows that

$$\begin{aligned} w_H(V(H)) &= \overbrace{w_H(v) + \sum_{i=1}^{k} w_H(u'_i)}^{=2\sum_{i=1}^k w_{H'}(u_i)} + \overbrace{w_{H'}(V(H'))}^{\leqslant 2\,\mathrm{OPT}(H', w_{H'}) + 1} \\ &\leqslant 2\,\mathrm{OPT}(H, w_H) + 1\,. \end{aligned}$$

□

Putting things together. We are ready to prove Theorem 6.2.

Proof of Theorem 6.2. We can decide in polynomial time (see for instance [63]) if $H[N(v_0)]$ is chordal, and if not, output a hole of $H[N(v_0)]$. If the latter holds, we are done by Lemma 6.3. If the former holds, we can decide in polynomial time (see [62], and the proof of Lemma 6.6) whether H contains a $2P_3$. If it does, we are done by Lemma 6.4.

From now on, assume that the subgraph induced by $N(v_0)$ is chordal and $2P_3$-free. This is done without loss of generality. Notice that hypotheses (H1) and (H2) from Section 6.2 hold for H. This is obvious for (H1). To see why (H2) holds, remember that G is twin-free. Hence, every edge uu' contained in $N[v_0]$

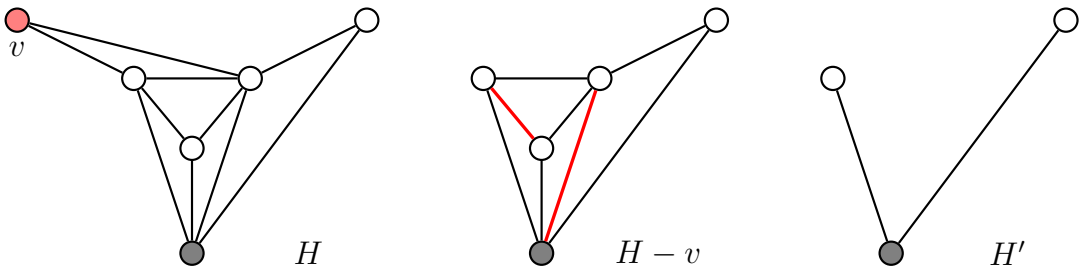


FIGURE 6.3. Here v_0 is the gray vertex and v is the red vertex. $H-v$ violates (H2), and contains two pairs of true twins, indicated by the red edges. Lemma 6.8 applies. We see that H' is a P_3, for which Lemma 6.7 gives $w_{H'} = \mathbf{1}_{H'}$. In (H, w_H), all vertices get a unit weight except v, which gets a weight of 2, since there are 2 pairs of true twins in $H-v$. Thus, $w_H = (\mathbf{1}, 2, 1, 1, 1, 1)$, where the entries corresponding to v_0 and v are bold and red, respectively.

must have a distinguisher in G, which is in $N_{\leqslant 2}[v_0]$. (In fact, notice that if u and u' are true twins in $H[N[v_0]]$ then the distinguisher is necessarily in $N_2(v_0)$.)

We repeatedly apply Lemma 6.8 in order to delete each vertex of $N_2(v_0)$ one after the other and reduce to the case where H is a twin-free graph for which v_0 is universal. We can then apply Lemma 6.7. The whole process takes polynomial time. □

6.3. Running-time Analysis

We now analyze the running-time of Algorithm 5. We assume that input graphs are given by their adjacency matrix. We need the following easy lemma, whose proof we include for completeness.

Lemma 6.9. *Given a matrix $N \in \{0,1\}^{r\times c}$, the set of all equivalence classes of equal rows of N can be found in time $O(rw)$.*

PROOF. Let R_0 and R_1 be the set of rows of N whose first entry is 0 and 1, respectively. We can determine R_0 and R_1 by reading the first column of N, which takes time $O(r)$. We then recurse on N'_0 and N'_1, where N'_i is the submatrix of N induced by R_i and the last $c-1$ columns of N. □

Before proving the next Lemma we remark that, given a graph H with n vertices and m edges, one can check whether H is a cluster by checking that each of its components is a clique, which takes $O(n^2)$ time.

Lemma 6.10. *Let G be an n-vertex, twin-free graph. Step 14, i.e. the construction of the 2-good weighted induced subgraph (H, w_H), can be performed in time $O(n^3)$.*

PROOF. We fix any vertex $v_0 \in V(G)$, and let $H = G[N_{\leqslant 2}[v_0]]$. We can check in $O(n^2)$ time whether $H[N(v_0)]$ is chordal by using the algorithm from [63]. If $H[N(v_0)]$ is not chordal this algorithm returns, as a certificate, an induced hole C. By Lemma 6.3, $H[V(w) + v_0]$ is strongly 2-good and the corresponding function w_H can be computed straightforwardly, hence we are done. Suppose

now that $H[N(v_0)]$ is chordal. We can construct the clique tree of $H[N(v_0)]$ (see for instance [62]) in $O(n^2)$ time. Each edge of the tree induces a separation of $H[N(v_0)]$, and we can check if each side is a cluster graph in $O(n^2)$ time. If neither side is a cluster graph, then we have found a $2P_3$ in $H[N(v_0)]$. Hence, $H[V(2P_3) + v_0]$ is strongly 2-good and the corresponding function w_H can be computed straightforwardly. Since the clique tree has at most $|H| \leq n$ vertices, by orienting its edges as in the proof of Lemma 6.6 we find, in $O(n^3)$ time, a hitting clique. By applying Lemma 6.8 to get rid of true twins, which can be done in $O(n^2)$ time, we obtain a subgraph satisfying the hypotheses of Lemma 6.7. The weight function constructed in the proof of Lemma 6.7 can be obtained in $O(n^3)$ time. □

Lemma 6.11. *Algorithm 1 runs in $O(n^4)$-time.*

Proof. By Lemma 6.9, finding all true twins in G can be done in time $O(n^2)$. Therefore, the most expensive recursive call of the algorithm is the construction of the 2-good weighted induced subgraph (H, w_H) from Lemma 6.10, which can be done in time $O(n^3)$. Therefore, the running-time $T(n)$ of the algorithm satisfies $T(n) \leqslant T(n-1) + O(n^3)$, which gives $T(n) = O(n^4)$. □

Before we formally prove the main theorem, we give a lemma from [28, 28] which formally justifies Steps 8-12.

Lemma 6.12 ([28])**.** *Let (G, w) be a weighted graph and $u, u' \in V(G)$ be true twins. Let (G', w') denote the weighted graph obtained from G by transferring the whole weight of u' to u and then deleting u', that is, let $G' := G - u'$ and $w'(v) := w(v)$ if $v \in V(G'), v \neq u$ and $w'(v) := w(u) + w(u')$ if $v = u$. Then $\mathrm{OPT}(G, w) = \mathrm{OPT}(G', w')$.*

Proof. Note that if X' is a hitting set for G' then it naturally gives a hitting set X for G of the same weight. Because no induced P_3 can contain both u and u' we let $X := X' \cup \{u'\}$ if $u \in X$ and $X := X'$ otherwise. Thus $\mathrm{OPT}(G, w) \leq \mathrm{OPT}(G', w')$.

On the other hand we have $\mathrm{OPT}(G', w') \leq \mathrm{OPT}(G, w)$ since an optimal solution X must be inclusionwise minimal, and therefore either contains both u, u' of neither of them. □

Proof of Theorem 6.1. We proceed by induction on the number of recursive calls of algorithm 5 to show that the set X returned is an inclusionwise minimimal hitting set of G with $c(X) \leq 2\,\mathrm{OPT}(G, w)$. The base case trivially holds. Here there are no recursive calls, which occurs when input graph G is already a cluster graph and $X = \varnothing$ is returned on line 2. Now assume that there is at least 1 recursive call returning X' which is an inclusionwise minimal hitting set of its input (G', w') satisfying $w'(X') \leq 2\,\mathrm{OPT}(G', w')$. We have the following cases:

Case 1: The recursive call occurs at Step 6. We have $w'(X) = w(X)$ and $\mathrm{OPT}(G, w) = \mathrm{OPT}(G', w')$ since $G' = G - u$, $w(u) = 0$, for all other $v \in V(G'), w'(v) = w(v)$. Therefore $w(X) = w'(X) \leq 2\,\mathrm{OPT}(G', w) = 2\,\mathrm{OPT}(G, w)$.

Case 2: The recursive call occurs at Step 11. By Lemma 6.12 $w(X) = w'(X') \leq 2\,\mathrm{OPT}(G', w') = 2\,\mathrm{OPT}(G, w)$.

Case 3: The recursive call occurs at Step 18. Here $G = G'$ and $X = X'$, therefore by induction X is an inclusionwise minimal hitting set. Let w'' be the extension of w_H to $V(G)$ defined as $w''(v) := 0$ for $v \in V(G) - V(H)$. By induction $w'(X) \leq 2\,\mathrm{OPT}(G, w')$, $\lambda^* w''(X) \leq 2\,\mathrm{OPT}(G, \lambda^* w'')$ since (H, w_H) is 2-good in G. So $w(X) = w'(X) + \lambda^* w''(X) \leq 2\,\mathrm{OPT}(G, w)$ by the Local Ratio Lemma 3.1. □

6.4. Polyhedral results

Recall the Sherali-Adams hierarchy [60] which we introduced in Section 3.3 of Chapter 3.

Let $\mathcal{P}_3(G)$ denote the collection of all vertex sets $\{u, v, w\}$ that induce a P_3 in G and let $\mathsf{SA}_r(G) := \mathsf{SA}_r(P(G))$, where

$$P(G) := \{x \in [0,1]^{V(G)} \mid \forall \{u, v, w\} \in \mathcal{P}_3(G) : x_u + x_v + x_w \geqslant 1\}.$$

If a weight function $w : V(G) \to \mathbb{R}_{\geqslant 0}$ is provided, we let

$$\mathsf{SA}_r(G, w) := \min \left\{ \sum_{v \in V(G)} w(v) x_v \mid x \in \mathsf{SA}_r(G) \right\}$$

denote the optimum value of the corresponding linear programming relaxation. For the sake of simplicity, we sometimes denote by $\mathsf{SA}_r(G, w)$ the above linear program itself.

The next lemma is a re-framing of Lemma 3.7 which applies to all E3-Vertex Cover problems.

Lemma 6.13. *Let $x \in \mathsf{SA}_1(G)$. If G contains a P_3 which has a diagonal, then $x_v \geqslant 2/5$ for some vertex of the P_3.*

In the next result, the *order* of weighted graph (G, w) is simply defined as the order $|G|$ of G.

Lemma 6.14. *Fix $\alpha \geqslant 1$ and $r \in \mathbb{Z}_{\geqslant 0}$. Let (G, w) be a minimum order weighted graph such that $\mathrm{OPT}(G, w) > \alpha \cdot \mathsf{SA}_r(G, w)$. The following two assertions hold:*

(i) if x is an optimal solution to $\mathsf{SA}_r(G, w)$, then $x_v < 1/\alpha$ for all $v \in V(G)$;
(ii) G is connected and twin-free.

Proof. (i) Suppose for contradiction that there is some component $x_v \geqslant 1/\alpha$. Note that x restricted to components other than v is a feasible solution to $\mathsf{SA}_r(G - v, w)$. Thus $\mathsf{SA}_r(G - v, w) \leqslant \mathsf{SA}_r(G, w) - w(v)x_v$. By minimality of G, there is a hitting set X' of $G - v$ such that $c(X') \leqslant \alpha \cdot \mathsf{SA}_r(G - v, w)$. Therefore $X := X' + v$ is a hitting set of G with $c(X) = w(v) + c(X') \leqslant w(v) + \alpha \cdot \mathsf{SA}_r(G - v, w) \leqslant \alpha \cdot w(v)x_v + \alpha \cdot \mathsf{SA}_r(G - v, w) \leqslant \alpha \cdot \mathsf{SA}_r(G, w)$, a contradiction.

(ii) First, note that G is connected, otherwise there exists a connected component H of G such that $\mathrm{OPT}(H, w_H) > \alpha \cdot \mathsf{SA}_r(H, w_H)$, where w_H is the restriction of w to $V(H)$, contradicting the minimality of G.

Second, we show that G is twin-free. Note that if u, v are true twins we can delete either of them, say v, and set $w'(u) := w(u) + w(v)$, $w'(c) := w(c)$ for c distinct from u and obtain a smaller weighted graph (G', w'). We claim that

$\mathsf{SA}_r(G', w') \leqslant \mathsf{SA}_r(G, w)$. To see this, let x be an optimal solution to $\mathsf{SA}_r(G, w)$, let $x'_u := \min(x_u, x_v)$ and $x'_w := x_w$ for $w \neq u, v$. By symmetry, this defines a feasible solution x' to $\mathsf{SA}_r(G', w')$ of weight

$$\sum_{w \neq v} w'(w)x'_w = (w(u) + w(v)) \min(x_u, x_v) + \sum_{c \neq u,v} w(c)x_c \leqslant \sum_c w(c)x_c \,.$$

This proves the claim.

Since $|G'| < |G|$ there is some hitting set X' of G' such that $w'(X') \leqslant \alpha \cdot \mathsf{SA}_r(G', w')$. Let $X := X'$ if $u \notin X'$ and $X := X' + v$ otherwise. In either case, $w(X) = w'(X') \leqslant \alpha \cdot \mathsf{SA}_r(G', w') \leqslant \alpha \cdot \mathsf{SA}_r(G, w)$, a contradiction. Hence, G is twin-free. □

Lemma 6.15. *If (G, w) is a minimum order weighted graph such that $\mathrm{OPT}(G, w) > 5/2 \cdot \mathsf{SA}_1(G, w)$, then G is triangle-free and claw-free.*

Proof. First, we show that G is triangle-free. Suppose G contains a triangle with vertices u, v and w. Since G is twin-free, every edge of the triangle has a distinguisher. Without loss of generality, $\mathcal{P}_3(G)$ contains $\{u, v, y\}$, $\{u, w, y\}$, $\{v, w, z\}$ and $\{u, v, z\}$ where y, z are distinct vertices outside the triangle. It is easy to see that, for instance, edge uw is a diagonal contained in a P_3. By Lemmas 6.13 and 6.14.(i), we obtain a contradiction, and conclude that G is triangle-free.

A similar argument shows that G cannot contain a claw because again there will be at least one induced P_3 containing a diagonal, which yields a contradiction by Lemmas 6.13 and 6.14.(i). We leave the details to the reader. □

Theorem 6.16. *For every graph G, the integrality gap of $\mathsf{SA}_1(G)$ is at most $5/2$.*

Proof. We show that for every cost function c on the vertices of G, there exists some hitting set X such that $c(X) \leqslant 5/2 \cdot \mathsf{SA}_1(G, w)$. Suppose not, and let (G, w) be a minimum order counterexample. By Lemma 6.15, G is triangle-free and claw-free. Hence the maximum degree of G is at most 2. Since G is connected, by Lemma 6.14.(ii), G is either a path or a cycle.

We claim that in fact $\mathsf{SA}_0(G, w)$ (the basic LP) has integrality gap at most 2 in this case.

Paths are solved exactly since the coefficient matrix of the LP is totally unimodular in this case, by the consecutive ones property [59].

Now suppose that G is a cycle, and let x be an extreme optimal solution of $\mathsf{SA}_0(G, w)$. First, assume that there is some $v \in V(G)$ such that $x_v \geqslant 1/2$. Since $G - v$ is a path, there exists a hitting set X' in $G - v$ of weight $w(X') \leqslant \sum_{u \neq v} w(u)x_u$, by the previous paragraph. Hence, we see that $X := X' + v$ is a hitting set of G with $w(X) = w(v) + w(X') \leqslant w(v) + \sum_{u \neq v} w(u)x_u \leqslant 2\sum_u w(u)x_u = 2\,\mathsf{SA}_0(G, w)$. On the other hand if $x_v < 1/2$ for all vertices v, then there can be no vertex v with $x_v = 0$ since then $x_u + x_v + x_w \geqslant 1$ implies $\max(x_u, x_w) \geqslant 1/2$, where u, w are the neighbors of v in G. So $0 < x_v < 1/2$ for all $v \in V(G)$.

Therefore, extreme point x is the unique solution of $|G|$ equations of the form $x_u + x_v + x_c = 1$ for $\{u, v, c\} \in \mathcal{P}_3(G)$. Hence $x_v = 1/3$ for all vertices. Thus

$\mathsf{SA}_0(G,w) = 1/3 \cdot w(G)$. Now notice that since G is a cycle we can partition the vertices of G into two disjoint hitting sets X and Y. Without loss of generality assume that $w(X) \leqslant 1/2 \cdot w(G)$. Then $w(X) \leqslant 3/2 \cdot \mathsf{SA}_0(G,w)$. This concludes the proof that the integrality gap is at most $5/2$. □

Theorem 6.17. *For every $\epsilon > 0$ there is some instance (G,w) of* CVD *such that* $\mathrm{OPT}(G,w) \geqslant (5/2-\epsilon)\,\mathsf{SA}_1(G,w)$.

Proof. We show there is a graph G for which every hitting set X has $w(X) \geqslant (5/2-\epsilon)\,\mathsf{SA}_1(G,w)$ for $w := \mathbf{1}_G$. Let G be a graph whose girth is at least k for some constant $k \geqslant 5$ and with the independence number $\alpha(G) \leqslant n/k$ where $n := |G|$. It can be shown via the probabilistic method that such a G exists, see [1]. Set $w(v) := 1$ for all $v \in V(G)$. We have $w(X) \geqslant n(1-2/k)$ for every hitting set X. To see this observe that since G is triangle-free and $\alpha(G) \leqslant n/k$, when we remove X we will get at most n/k components each of size at most 2. Thus there are at most $2n/k$ vertices in $G - X$, so $|X| \geqslant n - 2n/k$. Therefore, $\mathrm{OPT}(G,w) \geqslant (1-2/k)n$.

In order to show $\mathsf{SA}_1(G,w) \leqslant 2n/5$, we construct the following feasible solution x to $\mathsf{SA}_1(G,w)$. Set $x_v := 2/5$ for all $v \in V(G)$ and $x_{vu} := 0$ if $vu \in E(G)$ and $x_{vu} := 1/5$ if $vu \notin E(G)$. The inequalities defining $\mathsf{SA}_1(G)$ are all satisfied by x. This is obvious for inequalities (1), (3) and (4). For inequality (2), notice that at most one of da, db, dc can be an edge of G, since otherwise G would have a cycle of length at most 4. Thus (2) is satisfied too, $x \in \mathsf{SA}_1(G)$ and $\mathsf{SA}_1(G,w) \leqslant 2n/5$.

This completes the proof since, by taking $k \geqslant 5/\epsilon$, we have $\mathrm{OPT}(G,w) \geqslant n(1-2/k) \geqslant (5/2-\epsilon)2n/5 \geqslant (5/2-\epsilon)\,\mathsf{SA}_1(G,w)$. □

By relying on Theorem 6.2, we now show that the integrality gap decreases to $2+\epsilon$ after applying $\mathrm{poly}(1/\epsilon)$ rounds of Sherali-Adams.

Theorem 6.18. *For every fixed $\epsilon > 0$, performing $r = \mathrm{poly}(1/\epsilon)$ rounds of the Sherali-Adams hierarchy produces an LP relaxation of* CVD *whose integrality gap is at most $2+\epsilon$. That is,* $\mathrm{OPT}(G,w) \leqslant (2+\epsilon)\,\mathsf{SA}_r(G,w)$ *for all weighted graphs (G,w).*

Proof. In order to simplify the notation below, let us assume that $2/\epsilon$ is integer. For instance, we could restrict to $\epsilon = 2^{-l}$ for some $l \in \mathbb{Z}_{\geqslant 1}$. This does not hurt the generality of the argument. We take $r := 1+(2/\epsilon)^4$. We may assume that $\epsilon < 1/2$ since otherwise we invoke Theorem 6.16 (taking $r = 1$ suffices in this case).

Let (G,w) be a counterexample to the theorem, with $|G|$ minimum. By Lemma 6.14.(i), for every optimal solution x to $\mathsf{SA}_r(G,w)$, every vertex $v \in V(G)$ has $x_v < 1/(2+\epsilon)$. By Lemma 6.14.(ii), G is twin-free (and connected).

We will use the following fact several times in the proof: for all $R \subseteq V(G)$ with $|R| \leqslant r$ and every $x \in \mathsf{SA}_r(G)$, the restriction of x to the variables in R is a convex combination of hitting sets of $G[R]$. This is easy to see since, denoting by x_R the restriction of x, we get $x_R \in \mathsf{SA}_r(G[R])$ and the Sherali-Adams hierarchy is known to converge in at most "dimension-many" rounds, see for instance [19].

First, we claim that G has no clique of size at least $2/\epsilon$. Suppose otherwise, let C be a clique of size $k := 2/\epsilon$ and let D be a minimal set such that each edge of

C has a distinguisher in D. Let $H := G[C \cup D]$. Then, following the construction from Section 6.2, one can obtain a weight function w_H such that $w_H(H) = 2k-1$, and $w_H(X) \geqslant k-1$ for any hitting set X of H: see [28, Lemma 3] for the full construction, whose proof also shows that $|H| \leqslant 2k-1 \leqslant r$. Since every valid inequality supported on at most r vertices is valid for $\mathsf{SA}_r(G)$, the inequality $\sum_v w_H(v)x_v \geqslant k-1$ is valid for $\mathsf{SA}_r(G)$. Since $w_H(H) = 2k-1$, this implies that for all $x \in \mathsf{SA}_r(G)$, there is some vertex $a \in V(H)$ with $x_a \geqslant (k-1)/(2k-1)$. Since $(k-1)/(2k-1) \geqslant 1/(2+\epsilon)$, we get a contradiction. This proves our first claim.

Second, we claim that for every $v_0 \in V(G)$, the subgraph of G induced by the neighborhood $N(v_0)$ has no stable set of size at least $2/\epsilon$. The proof is similar to that for cliques given above, except that this time we let H be the induced star $(K_{1,k})$ on v_0 and a stable set S of size $k = 2/\epsilon$. The weight function w_H given by Algorithm 6.7 has $w_H(v_0) = k-1$ and $w_H(v) = 1$ for all $v \in S$. Notice that once again $w_H(H) = 2k-1$. The *star inequality* $\sum_v w_H(v)x_v \geqslant k-1$ is valid for $\mathsf{SA}_r(G)$, which guarantees that for every $x \in \mathsf{SA}_r(G)$ there is some $a \in V(H)$ which has $x_a \geqslant (k-1)/(2k-1) \geqslant 1/(2+\epsilon)$. This establishes our second claim.

Third, we claim that the neighborhood of every vertex v_0 induces a chordal subgraph of G. Suppose that C is a hole in $G[N(v_0)]$. We first deal with the case $|C| \leqslant r-1 = (2/\epsilon)^4$. We can repeat the same proof as above, letting H be the induced wheel on $V(w)+v_0$ and using the weight function w_H defined in the proof of Lemma 6.3. Consider the *wheel inequality* $\sum_v w_H(v)x_v \geqslant k-3$, where $k := |H| = |C|+1$. Since the wheel has at most r vertices, the wheel inequality is valid for $\mathsf{SA}_r(G)$. Since $w_H(H) = 2k-6 = 2(k-3)$, for every $x \in \mathsf{SA}_r(G)$, there is some $a \in V(H)$ which has $x_a \geqslant 1/2 \geqslant 1/(2+\epsilon)$. This concludes the case where $|C|$ is "small".

Now, assume that $|C| \geqslant r$, and consider the wheel inequality with right-hand side scaled by $2/(2+\epsilon)$. Suppose this inequality is valid for $\mathsf{SA}_r(G)$. This still implies that some vertex a of H has $x_a \geqslant 1/(2+\epsilon)$, for all $x \in \mathsf{SA}_r(G)$, which produces the desired contradiction. It remains to prove that the scaled wheel inequality is valid for $\mathsf{SA}_r(G)$.

Let F denote any r-vertex induced subgraph of H that is a fan. Hence, F contains v_0 as a universal vertex, plus a path on $r-1$ vertices. Letting $w_F(v_0) := r-3-\lfloor (r-1)/3 \rfloor$ and $w_F(v) := 1$ for $v \in V(F-v_0)$, we get the inequality $\sum_v w_F(v)x_v \geqslant r-3$, which is valid for $\mathsf{SA}_r(G)$. By taking all possible choices for F, and averaging the corresponding inequalities, we see that the inequality

$$\left(r-3-\left\lfloor \frac{r-1}{3} \right\rfloor\right) x_{v_0} + \frac{r-1}{k-1} \sum_{v \in V(H-v_0)} x_v \geqslant r-3$$

$$\iff \frac{k-1}{r-1}\left(r-3-\left\lfloor \frac{r-1}{3} \right\rfloor\right) x_{v_0} + \sum_{v \in V(H-v_0)} x_v \geqslant \frac{k-1}{r-1}(r-3)$$

is valid for $\mathsf{SA}_r(G)$. It can be seen that this inequality dominates the scaled wheel inequality, in the sense that each left-hand side coefficient is not larger than the corresponding coefficient in the scaled wheel inequality, while the right-hand side is not smaller than the right-hand side of the scaled wheel inequality. Therefore,

the scaled wheel inequality is valid for $\mathsf{SA}_r(G)$. This concludes the proof of our third claim.

By the first, second and third claim[2], $|N(v_0)| \leqslant \omega(G[N(v_0)]) \cdot \alpha(G[N(v_0)]) \leqslant 4/\epsilon^2$ for all choices of v_0. This implies in particular that $|N_{\leqslant 2}[v_0]| \leqslant 1+16/\epsilon^4 = r$. Now let $H := G[N_{\leqslant 2}[v_0]]$. Theorem 6.2 applies since G is twin-free, by our second claim. Let w_H be the corresponding weight function. The inequality $\sum_v w_H(v)x_v \geqslant \mathrm{OPT}(H, w_H)$ is valid for $\mathsf{SA}_r(G)$.

Let λ^* be defined as in Step 15 of Algorithm 5, and let $a \in V(G)$ denote any vertex such that $(c-\lambda^* w_H)(a) = 0$. By minimality of G, there exists in $(G', w') := (G-a, c-\lambda^* w_H)$ a minimal hitting set X' of weight $w'(X') \leqslant (2+\epsilon)\,\mathsf{SA}_r(G', w')$. We let $X := X'$ in case X' is a hitting set of G, and $X := X' + a$ otherwise. Assume that $X = X' + a$, the other case is easier. We have

$$\begin{aligned} c(X) &= w'(X') + \lambda^* w_H(X) \\ &\leqslant (2+\epsilon)\,\mathsf{SA}_r(G', w') + \lambda^*(w_H(H) - 1) \\ &\leqslant (2+\epsilon)\,\mathsf{SA}_r(G', w') + 2\lambda^* \,\mathrm{OPT}(H, w_H) \\ &\leqslant (2+\epsilon)\,(\mathsf{SA}_r(G', w') + \lambda^* \,\mathrm{OPT}(H, w_H))\ . \end{aligned}$$

By LP duality, we have $\mathsf{SA}_r(G, w) \geqslant \mathsf{SA}_r(G', w') + \lambda^* \,\mathrm{OPT}(H, w_H)$. This implies that $c(X) \leqslant (2+\epsilon)\,\mathsf{SA}_r(G, w)$, contradicting the fact that (G, w) is a counterexample. This concludes the proof. □

We now complement the result above by showing that every LP relaxation of CVD with (worst case) integrality gap at most $2-\epsilon$ must have super-polynomial size. The result is a simple consequence of an analogous result of [5] on the integrality gap of Vertex Cover, and of the straightforward reduction from Vertex Cover to CVD.

Proposition 19. *For infinitely many values of n, there is a graph G on n vertices such that every size-$n^{o(\log n/\log\log n)}$ LP relaxation of* CVD *on G has integrality gap $2 - o(1)$.*

Proof. In [5] a similar result is proved for LP-relaxations of Vertex Cover: for infinitely many values of n, there is a graph G on n vertices such that every size-$n^{o(\log n/\log\log n)}$ LP relaxation of Vertex Cover on G has integrality gap at least $2-\epsilon$, where $\epsilon = \epsilon(n) = o(1)$ is a non-negative function.

Let G be such a graph, and let G^+ be the graph obtained from G by attaching a pendant edge to every vertex. It is easy to see that $U \subseteq V(G)$ is a hitting set for G^+ if and only if U is a vertex cover of G.

Toward a contradiction, suppose that $Ax \geqslant b$ is a size-$n^{o(\log n/\log\log n)}$ LP relaxation of CVD on G^+ with integrality gap at most $2-\delta$, for a fixed $\delta > \epsilon$ (where $x \in \mathbb{R}^d$ for some dimension d depending on G). For every $c^+ \in \mathbb{Q}_{\geqslant 0}^{V(G^+)}$ there exists a hitting set X of G^+ such that $c^+(X) \leqslant (2-\delta)\,\mathrm{LP}(G^+, c^+)$.

We can easily turn $Ax \geqslant b$ into an LP relaxation for Vertex Cover. For every vertex cover U of G, we let the corresponding point be the point $\pi^U \in \mathbb{R}^d$ for U seen as a hitting set in G^+. For every $c \in \mathbb{Q}_{\geqslant 0}^{V(G)}$, we define $c^+ \in \mathbb{Q}_{\geqslant 0}^{V(G^+)}$ via

[2]Recall that, for any perfect graph H, one has $|H| \leqslant \alpha(H) \cdot \omega(H)$.

$c^+(v) := w(v)$ for $v \in V(G)$, and $c^+(v) := \sum_{u \in V(G)} w(u)$ for $v \in V(G^+) - V(G)$. Then, we let the affine function f_c for c be the affine function f_{c^+} for c^+.

Since the integrality gap of $Ax \geqslant b$, seen as an LP relaxation of CVD, is at most $2-\delta$, for every $c \in \mathbb{Q}_{\geqslant 0}^{V(G)}$ there exists a hitting set X of G^+ whose weight is at most $(2-\delta)\,\mathrm{LP}(G^+, c^+)$, where c^+ is the weight function corresponding to c. If X contains any vertex of $V(G^+)-V(G)$, we can replace this vertex by its unique neighbor in $V(G)$, without any increase in weight. In this way, we can find a vertex cover U of G whose weight satisfies $w(u) \leqslant c^+(X) \leqslant (2-\delta)\,\mathrm{LP}(G^+, c^+) = (2-\delta)\,\mathrm{LP}(G, w)$. Hence, the integrality gap of $Ax \geqslant b$ as an LP relaxation of Vertex Cover is also at most $2-\delta < 2-\epsilon$. As the size of $Ax \geqslant b$ is $n^{o(\log n/\log\log n)}$, this provides the desired contradiction. □

We point out that the size bound in the previous result can be improved. Kothari, Meka and Raghavendra [48] have shown that for every $\epsilon > 0$ there is a constant $\delta = \delta(\epsilon) > 0$ such that no LP relaxation of size less than 2^{n^δ} has integrality gap less than $2-\epsilon$ for Max-CUT. Since Max-CUT acts as the source problem in [5], one gets a 2^{n^δ} size lower bound for Vertex Cover in order to achieve integrality gap $2-\epsilon$. This also follows in a black-box manner from [48] and [12]. The proof of Proposition 19 shows that the same bound applies to CVD.

6.5. Concluding Remarks

In this chapter we provide a tight approximation algorithm for the cluster vertex deletion problem (CVD). Our main contribution is the efficient construction of a local weight function on the vertices at distance at most 2 from any vertex v_0 such that every minimal hitting set of the input graph has local weight at most *twice* the local optimum. If the subgraph induced by $N(v_0)$ (the first neighborhood of v_0) contains a hole, or a $2P_3$, then this turns out to be straightforward. The most interesting case arises when the local subgraph H is twin-free, has radius 1, and moreover $H[N(v_0)] = H - v_0$ is chordal and $2P_3$-free. Such graphs are very structured, which we crucially exploit.

Lemma 6.5 allows us to define the local weight function on the vertices distinct from v_0 and then later adjust the weight of v_0. We point out that condition (ii) basically says that the local weight function should define a hyperplane that "almost" separates the hitting set polytope and the clique polytope of the chordal, $2P_3$-free graph $H - v_0$. This was a key intuition which led us to the proof of Theorem 6.2. If these polytopes were disjoint, this would be easy. But actually it is not the case since they have a common vertex (as we show, $H - v_0$ has a hitting clique).

One natural question arising from our approach of CVD in general graphs is the following: is the problem polynomial-time solvable on *chordal* graphs? This seems to be a non-trivial open question, also mentioned in [15], where similar vertex deletion problems are studied for chordal graphs. It could well be that CVD in general chordal graphs is hard. Now, what about chordal, $2P_3$-*free* graphs? We propose this as an open question.

Our second contribution is to study the CVD problem from the polyhedral point of view, in particular with respect to the tightness of the Sherali-Adams

hierarchy. Our results on Sherali-Adams fail to match the 2-approximation factor of our algorithm (by epsilon), and we suspect this is not by chance. We believe that, already for certain classes of triangle-free graphs, the LP relaxation given by a bounded number of rounds of the Sherali-Adams hierarchy has an integrality gap strictly larger than 2. Settling this is another open question. Our intuition is that Sherali-Adams seems to have a hard time recovering the *star inequality* $(k-1)x_{v_0} + \sum_{i=1}^{k} x_{v_i} \geqslant k-1$, valid when $N(v_0) = \{v_1, \ldots, v_k\}$ is a stable set.

As mentioned already in the introduction, we do not know any polynomial-size LP or SDP relaxation with integrality gap at most 2 for CVD. In order to obtain such a relaxation, it suffices to derive each valid inequality implied by Lemmas 6.3, 6.4, 6.7 and also somehow simulate Lemma 6.8. A partial result in this direction is that the star inequality has a bounded-degree sum-of-squares proof. Using earlier results [28, Algorithm 1], this implies that a bounded number of rounds of the Lasserre hierarchy provides an SDP relaxation for CVD with integrality gap at most 2, whenever the input graph is triangle-free. This should readily generalize to the wheel inequalities of Lemma 6.3 and of course to the inequality of Lemma 6.4 (since the underlying graph has bounded size). However, we do not know how for instance to derive the inequalities of Lemma 6.7. We leave this for future work as our third open question from this chapter.

Our fourth open question is as follows: what is the best running time for Algorithm 5? We think that it is possible to improve on our $O(n^4)$ upper bound.

CHAPTER 7

Split Graph Deletion

In this chapter, which is based on Drescher, Fiorini and Huynh [26], we move to a problem which can be posed as an instance of E5-Vertex Cover. We study this problem from the local ratio perspective. It is an example of *good subgraphs* unlocking advanced algorithmic techniques from Graph Theory.

Recall from Chapter 1 that a *split graph* is a graph whose vertex set can be partitioned into a clique K and a stable set S. Given a graph G and weight function $w : V(G) \to \mathbb{Q}_{\geq 0}$, the Split Vertex Deletion (SVD) problem asks to find a minimum weight set of vertices X such that $G - X$ is a split graph. It was shown by Foldes and Hammer [29] that a graph is a split graph if and only it it does not contain a 4-cycle, 5-cycle, or a two edge matching as an induced subgraph. Therefore, as an instance of E5-Vertex Cover, as seen in Chapter 3, we can easily obtain a 5-approximation algorithm for SVD via Algorithm 1. On the other hand, it is known that for every $\delta > 0$, SVD does not admit a $(2 - \delta)$-approximation algorithm, unless P=NP or the Unique Games Conjecture fails.

In Section 7.1 we start by explaining this hardness status in more detail. We then state our result and put it into the context of previous work on SVD. In Section 7.2 we define Clique-Stable Separators and state Theorem 7.2 by Bousquet, Lagoutte and Thomassé [10] that our local ratio approach relies on. We show P_k and $\overline{P}_k$ are almost 2-good subgraphs in Lemma 7.6, and that the intermediary graph class which forbids $\{P_k, \overline{P}_k\}$ has a direct polynomial time 2-approximation algorithm for SVD. Plugging these parts into Algorithm 2 of Section 3.2 in Chapter 3 gives the result.

7.1. Hardness

There is a simple approximation preserving reduction from Vertex Cover to SVD given by Lokshtanov, Misra, Panolan, Philip, and Saurabh [55].

Their reduction is as follows. Given weighted graph (G, w), add to it a single vertex v with weight $W := \sum_{u \in V(G)} w(u)$ that is adjacent to none of the vertices of G. The weights on the vertices of G remain the same. Denote the resulting weighted graph (G', w').

Clearly any feasible solution X' for (G', w') to SVD can be modified to be disjoint from v. Such a solution X' is actually a vertex cover in G, that is, a solution to Vertex Cover on (G, w), with the same weight.

Therefore, for every $\delta > 0$, SVD does not admit a $(2 - \delta)$-approximation algorithm, unless P=NP or the Unique Games Conjecture fails [46].

7.2. A Simple, almost tight approximation algorithm

Here we give a short $(2+\epsilon)$-approximation algorithm. For every $\epsilon > 0$, Lokshtanov et al. [55] recently gave a *randomized* $(2+\epsilon)$-approximation algorithm for SVD. Their approach is based on the randomized 2-approximation algorithm, for FVST [54] due to Lokshtanov, Misra, Mukherjee, Panolan, Philip, and Saurabh. As we mentioned in Section 5.1 of Chapter 5, we viewed this approach as being partly based on finding 2-good subgraphs with respect to a particular optimal solution which is randomly found. Here, in the case of SVD guessing an optimal solution seems to be more complicated and requires several new ideas and insights.

Our main result is a much simpler deterministic $(2+\epsilon)$-approximation algorithm for SVD.

Theorem 7.1. *For every $\epsilon > 0$, there is a (deterministic) $(2+\epsilon)$-approximation algorithm for* SVD.

As far as we can tell, the easy 5-approximation described above was the previously best (deterministic) approximation algorithm for SVD. Before describing our algorithm and proving its correctness, we need a few definitions.

Definitions. A *cut* in a graph G is a pair (A, B) such that $A \cup B = V(G)$ and $A \cap B = \varnothing$. The cut (A, B) is said to *separate* a pair (K, S) where K is a clique, and S a stable set if $K \subseteq A$ and $S \subseteq B$. A family of cuts $\mathcal{F}$ is called a *clique-stable set separator* if for all pairs (K, S) where K is a clique and S is a stable set disjoint from K, there exists a cut (A, B) in $\mathcal{F}$ such that (A, B) separates (K, S). For each $k \in \mathbb{N}$, let P_k be the path on k vertices.

Clique-Stable Separator. The main technical ingredient we require is the following theorem of Bousquet, Lagoutte and Thomassé [10].

Theorem 7.2 (Theorem 12 [10]). *For every $k \in \mathbb{N}$, there exists $c(k) \in \mathbb{N}$ such that every n-vertex, $\{P_k, \overline{P_k}\}$-free graph has a clique-stable set separator of size at most $n^{c(k)}$.*

We remark that [10] does not state that the clique-stable set separator can be found in polynomial time, but it is easy to check that this is the case. The given proof of Theorem 7.2 is a constructive, induction argument which can be turned into a recursive algorithm that clearly runs in polynomial time, given that each recursive step can be accomplished in polynomial time. The recursive step relies on obtaining the disjoint sets V_1, V_2 each of size at least $t_k n$ for some $0 < t_k \leq 1/2$, as defined below in the theorem of Bousquet, Lagoutte and Thomassé [11]. Let $c(k) := (-1/\log_2(1-t_k))$. Let $V_3 := V(G - V_1 - V_2)$. The algorithm recursively builds separators on $V_1 \cup V_3$ and on $V_2 \cup V_3$ of size $(|V_i| + |V_3|)|)^{c(k)}$ for $i \in \{1, 2\}$, which it used to then obtain a separator for G of size $n^{c(k)}$. The run time is $T_{7.2}(n) = 2T_{7.2}((1-t_k)n) + O(n^{c(k)}) + T_{7.3}(n)$. Where $T_{7.3}(n)$ is the run-time required to obtain sets V_1, V_2.

Lemma 7.3 (Theorem 4 [11], Theorem 13 [10].)**.** *For every k, there is a constant $t_k \in (0, 1/2]$, such that every $\{P_k, \overline{P_k}\}$-free graph G contains two disjoint subsets*

of vertices V_1, V_2, each of size at least $t_k \cdot n$, such that V_1 and V_2 are completely adjacent or completely non-adjacent.

The proof of Lemma 7.3 is also constructive, and algorithmically efficient, but relies on obtaining an "almost stable" set S_0 guaranteed to exist by the following theorem of Fox and Sudakov. It then finds and examines the connected components of a subgraph of size at most $|S_0|$. Therefore $T_{7.3}(n) \leq T_{7.4}(n) + O(n^2)$, where $T_{7.4}(n)$ is the run-time required to build S_0 as defined below.

Theorem 7.4 (Theorem 1.1 [32]). *For every positive integer k and every $\gamma \in (0, 1/2)$, there exists $\delta > 0$ such that every n-graph G satisfies one of the following:*

- *G induces all graphs on k vertices.*
- *G contains a set S_0 of size at least δn with at most $\epsilon\binom{n}{2}$ edges in $G[S_0]$.*
- *G contains a set K_0 of size at least δn with at least $\epsilon\binom{n}{2}$ edges in $G[K_0]$.*

The proof of Theorem 7.4 is iterative and constructs S_0 in polynomial time but assumes the existence of sets A, B guaranteed by Erdős, and Hajnal [27]. The run time is $T_{7.4}(n) = \lceil \frac{\log \gamma^{-2}}{\log 3/2} \rceil T_{7.5}(n)$ where $T_{7.5}(n)$ is the run time required to find sets A, B as defined in Lemma 7.5 below.

Lemma 7.5 (Lemma 1.5 [27])**.** *For each $\gamma \in (0, 1/2)$, graph H on k vertices, and H-free graph G on $n \geq 2$ vertices, there are disjoint subsets A and B of $V(G)$ with $|A|, |B| \geq \gamma^{k-1}\frac{n}{k}$ such that either every vertex in A has at most $\gamma|B|$ neighbors in B, or every vertex in A has at least $(1-\gamma)|B|$ neighbors in B.*

Fortunately the proof of Lemma 7.5 is self-contained, algorithmic, and has run time $T_{7.5}(n) \leq T_{7.5}(\frac{\gamma(k-1)n}{k}) + O(n)$, where n is the size of the graph, which by the Master Theorem [21], is polynomial in n. Therefore, it is easy to see that by composing these run-time formulas that $T_{7.2}(n)$ is polynomial in n.

Proof of Theorem 7.1. We are now ready to state and prove the correctness of our local ratio based algorithm.

Lemma 7.6. *Let (G, w) be a weighted graph. For any ϵ there is a k such that P_k is a $2+\epsilon$ good subgraph.*

Proof. Choose k even and sufficiently large so that $\frac{2k}{k-4} \leq 2 + \epsilon$. Suppose some $P_k \subseteq G$. Define $w_1(v) := 1$ if $v \in P_k$ and $w_1(v) := 0$ otherwise. Let X be an optimal solution to SVD on P_k. A minimum cardinality vertex cover of P_k has size at least $\frac{k}{2}$. Therefore, X must include all of this vertex cover, with the exception of one clique of size 2. See Page 28 in Chapter 3. Thus $w(X) = \mathrm{OPT}(P_k, w_1) \geq \frac{k-4}{2}$ so

$$\sum_{v \in P_k} w_1(v) = k \leq \frac{2k}{k-4} \cdot \mathrm{OPT}(P_k, w_1) \leq (2+\epsilon)\,\mathrm{OPT}(P_k, w_1).$$

Therefore P_k is strongly $(2+\epsilon)$-good. □

Corollary 7.7. *Let (G, w) be a weighted graph. For any ϵ there is a k such that $\overline{P_k}$ is a $(2+\epsilon)$-good subgraph.*

PROOF. Since G is a split graph if and only if $\overline{G}$ is a split graph, this follows from Lemma 7.6. □

Lemma 7.8. *There is an algorithm that, given a weighted graph (G, w) and a clique-stable set separator $\mathcal{F}$ of G, finds a feasible solution X to* SVD *such that $w(X) \leq 2\,\mathrm{OPT}_{\mathrm{SVD}}(G, w)$ in time $|\mathcal{F}| \cdot O(n^c)$ where c is a constant, and $n := |V(G)|$.*

PROOF. Let $(A, B) \in \mathcal{F}$, and let X_A denote be a 2-approximate solution for vertex cover on $(\overline{G}[A], w)$, and X_B for vertex cover on $(G[B], w)$. Since (A, B) is a cut, $X_{\mathrm{A}} \cup X_{\mathrm{B}}$ is a hitting set of SVD. Let X_{AB} be a minimal weight hitting set such that $G - X_{\mathrm{AB}}$ splits into parts K_A and S_B where $K_A \subseteq A$ is a clique, and $S_B \subseteq B$ is a stable set. Notice that K_A is a vertex cover for $\overline{G}[A]$, and S_B is a vertex cover for $G[B]$. Since X_B, X_A are 2-approximate solutions for vertex cover on $(G[B], w)$ and $(\overline{G}[A], w)$ respectively, we have that $w(X_A \cup X_B) = w(X_A) + w(X_B) \leq 2\,\mathrm{OPT}_{\mathrm{VC}}(\overline{G}[A], w) + 2\,\mathrm{OPT}_{\mathrm{VC}}(G[B], w) \leq 2w(X_{\mathrm{AB}})$.

Now let X^* be a minimum weight hitting set for (G, w). Thus $G - X^*$ splits into parts K^* and S^* where K^* is a clique and S^* a stable set. By the definition of $\mathcal{F}$ there must be some cut $(A^*, B^*) \in \mathcal{F}$ such that $K^* \subseteq A^*, S^* \subseteq B^*$. We can find it in time $|\mathcal{F}| \cdot O(n^c)$ by examining each pair $(A, B) \in \mathcal{F}$. Computing 2-approximate solutions X_A, X_B for vertex cover on $(G[B], w)$ and $(\overline{G}[A], w)$ respectively can be done in $O(n^2)$ time. We choose the pair (A^*, B^*) which minimizes $(w(X_A) + w(X_B))$. By the remarks above, $w(X_{A^*} \cup X_{\mathrm{B}^*}) \leq 2w(X^*) = 2\,\mathrm{OPT}_{\mathrm{SVD}}(G, w)$. □

We now prove the main theorem, which clearly plugs into the local ratio $\mathcal{G}$-VERTEX DELETION framework of Algorithm 2 with $\alpha = 2 + \epsilon$, as given in Chapter 3.

PROOF OF THEOREM 7.1. Let G be an n-vertex graph, $w : V(G) \to \mathbb{Q}_{\geq 0}$, and $\epsilon > 0$. By Lemma 7.6, P_k is $(2 + \epsilon)$-good, where $k = k(\epsilon)$ is a sufficiently large integer. Therefore by the local ratio method [33], as in Algorithm 2, we may assume that G is P_k-free. By Corollary 7.7 we may also assume that G is $\overline{P_k}$-free. It remains to define the subroutine at line 13 in Algorithm 2. By Theorem 7.2, we can efficiently compute a polynomial-size clique-stable set separator $\mathcal{F}$ for G. Then, plugging $\mathcal{F}$ in the algorithm of Lemma 7.8, we find a hitting set X for G such that $w(X) \leq 2\,\mathrm{OPT}_{\mathrm{SVD}}(G, w)$.

□

Part 3

Conclusions

CHAPTER 8

Open Questions

We now list some questions which naturally presented themselves during our work in Part 2, which we were unable to answer. We also list some problems which appear to be suitable targets for the techniques presented in this thesis.

The first question is motivated by experimental results we obtained. We developed code which strengthens a basic LP formulation by applying k rounds of Sherali-Adams. It is freely available and packaged here `https://pypi.org/project/sherali-adams`. We ran exhaustive experiments for tournaments of size at most 9 and could not find a tournament T such that $\frac{\mathrm{OPT}(T,\mathbf{1}_T)}{\mathsf{SA}_2(T,\mathbf{1}_T)} > 2$.

QUESTION 3. *For* FVST*, what is the integrality gap of* SA_2 *?*

We saw that the integrality gap of SA_1 for CVD is $\frac{5}{2}$ in Chapter 6 and in Chapter 5 that it is $\frac{7}{3}$ for FVST. This evidence suggests that Sherali-Adams might converge faster for FVST than for CVD. We state the analog of Theorem 6.18 as an open question.

QUESTION 4. *For* FVST*, is there a constant* k *such that the integrality gap of* SA_k *is* $2+\epsilon$*?*

In Chapter 6 we gave a 2-approximation algorithm for the CVD. We consider a seemingly similar problem: Deletion to P_4-free (*cograph*) in a weighted graph. This problem can be posed as an instance of E4-VERTEX COVER, and therefore there is a trivial a 4-approximation algorithm. We wonder if some of the techniques of Chapter 6 could lead to a better approximation factor.

QUESTION 5 (Deletion to Cograph). *Recall* P_4 *are paths of order 4. Let* (G,w) *be a weighted graph. Is there an* $\alpha < 4$ *such that there is an* α*-approximation algorithm to the problem of finding minimum* $w(X)$ *such that* $G-X$ *contains no* P_4*?*

In Chapter 6 we gave a $O(n^4)$ worst case analysis of the run-time for Algorithm 5. We wonder if there are more sophisticated ideas that could give a faster implementation or tighter analysis. For example, $O(n^2)$ time is spent computing the new sets of true twins each time a vertex is removed. Perhaps a dynamic data structure could help improve this.

QUESTION 6. *Is there a faster implementation or tighter analysis of Algorithm 5* CLUSTER-VD-APX(G,w)*?*

In Chapter 4 we gave a quick 3-approximation algorithm for CLAW-VD restricted to triangle-free, weighted graphs.

QUESTION 7. *Is there a 3-approximation algorithm for the* CLAW-VD *problem in general weighted graphs?*

Another intriguing problem is, to what extent our methods can be adapted to other E3-Vertex Cover problems. We mention an open question due to László Végh [66]. We suspect that our results on diagonals from Chapter 3 could help to develop rounding algorithms in many classes of 3-uniform hypergraphs. In particular, we see that Lemma 3.7 is applicable in both CVD, and FVST.

Question 8. *For which classes of* 3*-uniform hypergraphs and which* $\epsilon > 0$ *does* Ek-Vertex Cover *admit a* $(3 - \epsilon)$*-approximation algorithm?*

CHAPTER 9

Conclusions

In this work we explored two general frameworks for designing approximation algorithms for instances of $\mathcal{G}$-Vertex Deletion. *Iterative rounding* and the *local ratio method*.

As we have seen, both methods give natural, and immediate algorithms for trivial k-approximation guarantee. The use of SA_1 and *diagonals*, seems to help 'juice' the LP rounding approach while still remaining at a fairly general level.

9.1. Constant rounds of Sherali-Adams

LP strengthening via a constant number of rounds of Sherali-Adams is a natural rounding approach. It plugs into the common iterative rounding method. Its inequalities, though derived through a mechanical procedure, encode a surprising amount of information about the structure of an optimal solution. Furthermore, it is something, as algorithm designers we can just try, without having to study the problem in depth. As such, these extended formulations can be a valuable source of insight about the problem.

On the other hand, it can be hard to understand what additional information is obtained during each round and how exactly it is encoded in the model. This understanding is necessary however, in order to prove properties about the solutions it returns. Further more, even when applied to simple LP's, the number of variables and inequalities quickly becomes unwieldy for a human to analyze by hand.

9.2. Local Ratio

We are amazed at the effectiveness of this deceptively simple design framework. It does not even involve solving an LP, yet completely closes the hardness gap for two of the main problems we studied. Deterministically in the case of CVD, and for FVST in the innovative randomized setting of [54]. We come within ϵ of closing the gap for the SVD by simply exploiting properties of graphs that do not contain paths of order k or their complement.

9.2.1. Flexibility. We saw in the CVD problem, how to recursively apply the method to always obtain a good weight function. On the other hand, although we can iteratively round an LP, it is unclear how to utilize a recursive combinatorial structure of the underlying graph. In contrast, our good subgraph approach only requires us to find a weight function with a subgraph, but we are free to do so by any means. We see a good example of this with our algorithm for SVD, where we find some k for which induced paths of order k are $2+\epsilon$ good, and then deduce that the remaining graph has a 2-approximate solution.

9.2.2. Good Subgraphs. A *strongly good subgraph* of size at most l has an analogous inequality in SA_k for some $k \leq l$. However the weaker notion of *good subgraph* does not seem to have an obvious analog in the LP world. We saw this in our work on Claw-VD. $K_{1,4}$, a strongly 3 good subgraph was exploitable in SA_1 via diagonal arguments, however $K_{1,3}$ which was 3 good in the case of triangle free graphs, did not seem to give us much help in SA_1.

9.2.3. Run Time. One might be tempted to conclude that the local ratio technique is typically slower. This is because a common approach is to first define subgraphs of size k which are α good, and then delete them. This generally leads to a run time of $O(n^k)$ unless there is some otherwise special method for finding them. The approach we use in CVD however, can obtain potentially large, not constantly bounded subgraphs which are 2-good, quickly through a recursive definition.

Bibliography

[1] N. Alon and J. H. Spencer. *The Probabilistic Method*. Wiley Publishing, 4th edition, 2016.

[2] M. Aprile, M. Drescher, S. Fiorini, and T. Huynh. A simple 7/3-approximation algorithm for feedback vertex set in tournaments. *arXiv:2008.08779*, 2020.

[3] M. Aprile, M. Drescher, S. Fiorini, and T. Huynh. A tight approximation algorithm for the cluster vertex deletion problem. In M. Singh and D. P. Williamson, editors, *Integer Programming and Combinatorial Optimization*, pages 340–353, Cham, 2021. Springer International Publishing.

[4] R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. In G. Ausiello and M. Lucertini, editors, *Analysis and Design of Algorithms for Combinatorial Problems*, volume 109 of *North-Holland Mathematics Studies*, pages 27–45. North-Holland, 1985.

[5] A. Bazzi, S. Fiorini, S. Pokutta, and O. Svensson. No small linear program approximates vertex cover within a factor $2 - \epsilon$. *Mathematics of Operations Research*, 44(1):147–172, 2019.

[6] P. A. Bernstein and N. Goodman. Power of natural semijoins. *SIAM Journal on Computing*, 10(4):751–771, 1981.

[7] J. R. Blair and B. Peyton. An introduction to chordal graphs and clique trees. In *Graph theory and sparse matrix computation*, pages 1–29. Springer, 1993.

[8] B. Bollobás and P. Erdös. Cliques in random graphs. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 80, pages 419–427. Cambridge University Press, 1976.

[9] A. Boral, M. Cygan, T. Kociumaka, and M. Pilipczuk. A fast branching algorithm for cluster vertex deletion. *Theory Comput. Syst.*, 58(2):357–376, 2016.

[10] N. Bousquet, A. Lagoutte, and S. Thomassé. Clique versus independent set. *European Journal of Combinatorics*, 40:73 – 92, 2014.

[11] N. Bousquet, A. Lagoutte, and S. Thomassé. The Erdős–Hajnal conjecture for paths and antipaths. *Journal of Combinatorial Theory, Series B*, 113:261–264, 2015.

[12] G. Braun, S. Pokutta, and A. Roy. Strong reductions for extended formulations. *Math. Program.*, 172(1–2):591–620, 2018.

[13] G. Braun, S. Pokutta, and D. Zink. Inapproximability of combinatorial problems via small LPs and SDPs. In *Proceedings of STOC 2015*, pages 107–116, New York, NY, USA, 2015. ACM.

[14] M.-C. Cai, X. Deng, and W. Zang. An approximation algorithm for feedback vertex sets in tournaments. *SIAM J. Comput.*, 30(6):1993–2007, 2001.

[15] Y. Cao, Y. Ke, Y. Otachi, and J. You. Vertex deletion problems on chordal graphs. *Theoretical Computer Science*, 745:75–86, 2018.

[16] S. O. Chan, J. R. Lee, P. Raghavendra, and D. Steurer. Approximate Constraint Satisfaction Requires Large LP Relaxations. *Proc. FOCS 2013*, 0:350–359, 2013.

[17] J. Chen, Y. Liu, S. Lu, B. O'sullivan, and I. Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 177–186, 2008.

[18] M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas. The strong perfect graph theorem. *Annals of Mathematics*, pages 51–229, 2006.

[19] M. Conforti, G. Cornuéjols, and G. Zambelli. *Integer programming*, volume 271 of *Graduate Texts in Mathematics*. Springer, Cham, 2014.

[20] W. J. Cook, W. Cunningham, W. Pulleyblank, and A. Schrijver. Combinatorial optimization. *Oberwolfach Reports*, 5(4):2875–2942, 2009.

[21] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms, 3rd Edition*. MIT press, 2009.

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

[23] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized algorithms*, volume 5. Springer, 2015.

[24] H. N. de Ridder et al. Information System on Graph Classes and their Inclusions (ISGCI). `https://www.graphclasses.org`.

[25] I. Dinur and S. Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, pages 439–485, 2005.

[26] M. Drescher, S. Fiorini, and T. Huynh. A simple $(2 + \epsilon)$-approximation algorithm for split vertex deletion. *arXiv preprint arXiv:2009.11056*, 2020.

[27] P. Erdős and A. Hajnal. Ramsey-type theorems. volume 25, pages 37–52. 1989. Combinatorics and complexity (Chicago, IL, 1987).

[28] S. Fiorini, G. Joret, and O. Schaudt. Improved approximation algorithms for hitting 3-vertex paths. In *International Conference on Integer Programming and Combinatorial Optimization (IPCO '16)*, pages 238–249. Springer, 2016.

[29] S. Foldes and P. L. Hammer. Split graphs having dilworth number two. *Canadian Journal of Mathematics*, 29(3):666–672, 1977.

[30] F. V. Fomin, S. Gaspers, D. Lokshtanov, and S. Saurabh. Exact algorithms via monotone local search. *J. ACM*, 66(2):Art. 8, 23, 2019.

[31] F. V. Fomin, T. Le, D. Lokshtanov, S. Saurabh, S. Thomassé, and M. Zehavi. Subquadratic kernels for implicit 3-hitting set and 3-set packing problems. *ACM Trans. Algorithms*, 15(1):13:1–13:44, 2019.

[32] J. Fox and B. Sudakov. Induced Ramsey-type theorems. *Advances in Mathematics*, 219(6):1771–1800, 2008.

[33] A. Freund, R. Bar-Yehuda, and K. Bendel. Local ratio: a unified framework for approximation algorithms. *ACM Computing Surveys*, 36:422–463, 01 2005.

[34] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197,

1981.

[35] M. Grötschel, L. Lovász, and A. Schrijver. Polynomial algorithms for perfect graphs. In *North-Holland mathematics studies*, volume 88, pages 325–356. Elsevier, 1984.

[36] A. Gupta, E. Lee, J. Li, P. Manurangsi, and M. Wlodarczyk. Losing treewidth by separating subsets. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '19)*, pages 1731–1749. SIAM, 2019.

[37] V. Guruswami and E. Lee. Inapproximability of feedback vertex set for bounded length cycles. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 21, page 2, 2014.

[38] V. Guruswami and E. Lee. Inapproximability of H-transversal/packing. *SIAM Journal on Discrete Mathematics*, 31(3):1552–1571, 2017.

[39] E. Halperin. Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. *SIAM Journal on Computing*, 31(5):1608–1623, 2002.

[40] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.

[41] S. Hosseinian and S. Butenko. Polyhedral properties of the induced cluster subgraphs. *Discrete Applied Mathematics*, 297:80–96, 2021.

[42] F. Hüffner, C. Komusiewicz, H. Moser, and R. Niedermeier. Fixed-parameter algorithms for cluster vertex deletion. *Theory Comput. Syst.*, 47(1):196–217, 2010.

[43] K. Jain. A factor-2 approximation algorithm for the generalized steiner network problem. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (FOCS '98)*, pages 448–457, 1998.

[44] G. Karakostas. A better approximation ratio for the vertex cover problem. In *International Colloquium on Automata, Languages, and Programming (ICALP '05)*, pages 1043–1050. Springer, 2005.

[45] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[46] S. Khot and O. Regev. Vertex cover might be hard to approximate to within $2-\epsilon$. *J. Comput. System Sci.*, 74(3):335–349, 2008.

[47] D. König. Über graphen und ihre anwendung auf determinantentheorie und mengenlehre. *Mathematische Annalen*, 77(4):453–465, 1916.

[48] P. K. Kothari, R. Meka, and P. Raghavendra. Approximating rectangles by juntas and weakly-exponential lower bounds for LP relaxations of CSPs. In H. Hatami, P. McKenzie, and V. King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 590–603. ACM, 2017.

[49] M. Kumar and D. Lokshtanov. Faster exact and parameterized algorithm for feedback vertex set in bipartite tournaments. In *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[50] M. Kumar, S. Mishra, N. Safina Devi, and S. Saurabh. Approximation algorithms for node deletion problems on bipartite graphs with finite forbidden subgraph characterization. *Theoretical Computer Science*, 526:90–96, 2014.

[51] L. C. Lau, R. Ravi, and M. Singh. *Iterative methods in combinatorial optimization*, volume 46. Cambridge University Press, 2011.

[52] J. M. Lewis and M. Yannakakis. The node-deletion problem for hereditary properties is np-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980.

[53] D. Lokshtanov. Personal communication. October 2020.

[54] D. Lokshtanov, P. Misra, J. Mukherjee, F. Panolan, G. Philip, and S. Saurabh. 2-approximating feedback vertex set in tournaments. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '20)*, pages 1010–1018. SIAM, 2020.

[55] D. Lokshtanov, P. Misra, F. Panolan, G. Philip, and S. Saurabh. A $(2+\varepsilon)$-factor approximation algorithm for split vertex deletion. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[56] M. Mnich, V. V. Williams, and L. A. Végh. A 7/3-approximation for feedback vertex sets in tournaments. In P. Sankowski and C. D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, volume 57 of *LIPIcs*, pages 67:1–67:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[57] K. Murty. G.(1983) linear programming.

[58] N. Robertson and P. D. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of algorithms*, 7(3):309–322, 1986.

[59] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1998.

[60] H. D. Sherali and W. P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM J. Discrete Math.*, 3(3):411–430, 1990.

[61] E. Speckenmeyer. On feedback problems in digraphs. In *Graph-theoretic Concepts in Computer Science (Kerkrade, 1989)*, volume 411 of *Lecture Notes in Comput. Sci.*, pages 218–231. Springer, Berlin, 1990.

[62] R. E. Tarjan. Decomposition by clique separators. *Discrete mathematics*, 55(2):221–232, 1985.

[63] R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13(3):566–579, 1984.

[64] D. Tsur. Faster parameterized algorithm for cluster vertex deletion. *Theory of Computing Systems*, 65(2):323–343, 2021.

[65] V. V. Vazirani. *Approximation algorithms*. Springer, 2001.

[66] L. A. Végh. Personal communication.

[67] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

[68] J. You, J. Wang, and Y. Cao. Approximate association via dissociation. *Discret. Appl. Math.*, 219:202–209, 2017.