CLUSTERING AND INCONSISTENT INFORMATION: A KERNELIZATION APPROACH

A Dissertation

by

YIXIN CAO

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2012

Major Subject: Computer Science

CLUSTERING AND INCONSISTENT INFORMATION: A KERNELIZATION APPROACH

A Dissertation

by

YIXIN CAO

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2012

Major Subject: Computer Science

ABSTRACT

Clustering and Inconsistent Information: A Kernelization Approach. (May 2012)

Yixin Cao, B.E., Harbin Engineering University; M.S., Beijing University of Aeronautics and Astronautics

Chair of Advisory Committee: Dr. Jianer Chen

Clustering is the unsupervised classification of patterns into groups, which is easy provided the data of patterns are consistent. However, real data are almost always tempered with inconsistencies, which make it a hard problem, and actually, the most widely studied formulations, correlation clustering and hierarchical clustering, are both NP-hard. In the graph representation of data, inconsistencies also frequently present themselves as cycles, also called deadlocks, and to break cycles by removing vertices is the objective of the classical feedback vertex set (FVS) problem.

This dissertation studies the three problems, correlation clustering, hierarchical clustering, and disjoint-FVS (a variation of FVS), from a kernelization approach. A kernelization algorithm in polynomial time reduces a problem instance provably to speed up the further processing with other approaches. For each of the problems studied, an efficient kernelization algorithm of linear or sub-quadratic running time is presented. All the kernels obtained in this dissertation have linear size with very small constants. Better parameterized algorithms are also designed based on the kernels for the last two problems.

Finally, some concluding remarks on possible directions for future research are briefly mentioned.

To my grandmother (January 13, 1923 - August 17, 2005)

ACKNOWLEDGMENTS

Now stepping on the last stair after such a long journey, it is a perfect time to look back. Before myself is a tortuous road, and only with many helps from many people I can finally finish it. I bad been almost lost, squandering several years without knowing I was born to do, until I came back to campus as a graduate student. In the past four years I really enjoyed the tranquil life in the small town in beautiful middle Texas, and the happiness it bestows on me is only second to my academical life. Undoubtedly I owe many thanks to a lot of people, especially for those who have helped to bring me back to track.

It is hard to express my sincere gratitude to my adviser, Dr. Jianer Chen. He introduced me into this area, and more importantly, reignited my passion in algorithmic design and mathematics. His individualized advising style makes him an ideal adviser. For me this implies great freedom. The freedom to choose what to study and where to go is so essential for me that I cannot imagine the otherwise. The free and equal communication atmosphere between us is another benefit.

It is the same hard to express the profound influence of my brother, also a Ph.D. in computer science, on me during this journey. His encouragement was critical at the beginning, when I had not made the decision, and his valuable suggestions and advice were important in each stage.

I would also like to thank my committee members, Dr. Sergiy Butenko, Dr. Donald K. Friesen, and Dr. Tiffani Williams, for their precious time and help.

Nearly half of my time was spent in office, with the guys (literally, guys) in the research group led by my adviser. We together created a great environment for research. It was a great pleasure working with them: Dr. Yang liu, Jia-Hao Fan, Qilong Feng, and Cheng Cao. Although I joined this group only half-year before his graduation, Yang and I had a wonderful collaboration. Qilong and I shared office in more than one year, during that time we had many valuable discussions. I also want to thank my labmats in Parasol in my first year: Dr. Jacob Smith and Xiaolong Tang, who helped a lot at the early stage of my Ph.D. study.

The Department of Computer Science at Texas A&M University provided financial support for my whole Ph.D. study. During these years, I have been working as Teaching Assitant, Research Assitant, and Unix Administrator, and thus worked together with a lot of faculty and staff. Besides relieving my finiancial pressure, the work experience itself helped me a lot.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

In a time dubbed as "information age", to handle information (data) is one of the most important tasks of human life, and can be roughly divided into two stages: "data collection" and "data analysis". Given that in real life there are no perfect methods to collect data, the data we are going to analyze are always supposed to be tempered with errors, and then one immediate problem is, how to retrieve the valuable information from the raw data by removing errors and inconsistencies.

One rich field where data keep spraying in a speed of gigabytes per day is the computational biology and life science. Every species in nature is complex, and even the DNA of a single-celled microorganism is complicated enough to carry large quantities of data, not to mention complex lifes or even human being. Thanks to remarkable efforts of generations of biologists, more and more raw genome sequences have been accumulated via experimental methods. From these collected data, with possible errors present, computational biologists try to, among others, classify the species , e.g. to identify similar genome sequences. The classification task has many different formulations that are, from the computational aspect, collectively called *clustering* problems. They turn out to be universal and can be found in many disciplines.

Not equipped with answer checkers, a challenge presents itself at the beginning and has to be answered before anything else is, what is a *solution*? Various models have been formulated on coping with inconsistencies, under different assumptions. Among them the most widely taken one should be that the erroneous data are very limited compared to the whole data set, so solutions with the minimum amount of inconsistencies are looked after. In other words, it is assumed that the solutions

This dissertation follows the style of SIAM Journal on Computing.

with the fewest errors are the most plausible. Computational problems formulated under this stipulation are all, unfortunately, NP-complete, and therefore unlikely have efficient algorithms runnable in polynomial time. In this case, people turn to other options. One is to sacrifice the optimality, trying to find a “reasonably good” solution in polynomial time. According to whether quality of the solution is guaranteed or not, algorithms in this category can be further labeled with *approximation* and *heuristic*. The other option is to allow “moderately exponential runtime”, while the solution must be optimal. As a complement to both options, before losing the optimality and starting the exponential time process, we can pre-process the problem first, by significantly reducing it, and then relay it to other approaches. This step is known as *kernelization*, and should be conducted efficiently, usually in polynomial time. The outcome of kernelization, the reduced problem, is called a *kernel*.

This dissertation, as indicated in its title, takes the kernelization approach. It starts from searching the limit that the kernelization step can reach on the problems under investigation, and then turns to the application of the obtained kernels in later approaches. The remainder of this section is a short introduction to the theoretical framework as well as the problems studied in this dissertation. All technical details will be deferred to later chapters, in the hope that this chapter can be kept as simple as possible.

1.1 Parameterized Computation

By definition, all NP-complete problems are equivalent in the sense of polynomial time solubility, while under some complexity hypotheses, their exponential solubility is disparate. Some problems, e.g. SAT, never witness an algorithm faster than $\mathcal{O}(2^n)$[1], — such one, if exists, will be a breakthrough [167] — while some have far better algorithms, e.g. subexponential or pseudo-polynomial. Under the widely accepted assumption P $\neq$ NP, some exponential explosion is inevitable in the time complexity of any algorithm for an NP-hard problem. On this ground, people turned to algorithms with *moderately* exponential runtime that, though still exponential, increases far slower than $\mathcal{O}(2^n)$.

This is a road on which many people have set foot. One notable candidate for moderately exponential functions is c^n ($c < 1$). Normally, for an exponential function $f(n) = c^n$, any small decrement of the base c will significantly decrease the value when n is large. Nevertheless, even c is as small as 1.1, and n is not too large (say $n = 1000$), c^n is still prohibitive. See Table 1.1 for a comparison (note that $1.414 = 2^{1/2}$ and $1.260 = 2^{1/3}$).

Table 1.1
The comparison of c^n for $c = 2, 1.414, 1.260, 1.1$

n	**10**	**20**	**30**	**40**	**50**	**100**	**1000**
2^n	1024	1048576	$1.1 \cdot 10^9$	$1.1 \cdot 10^{12}$	$1.1 \cdot 10^{15}$	$1.3 \cdot 10^{30}$	$1.1 \cdot 10^{301}$
1.414^n	32	1024	32768	1048576	$3.4 \cdot 10^7$	$1.1 \cdot 10^{15}$	$3.3 \cdot 10^{150}$
1.260^n	10.08	101.59	1024	10321	104032	$1.1 \cdot 10^{10}$	$2.2 \cdot 10^{100}$
1.1^n	2.59	6.73	17.45	45.26	117.40	13780	$2.5 \cdot 10^{41}$

In spite of their hardness, problems have to be solved, and heuristic is the tag most frequently seen on the weapons used to fight such problems in practice. Those heuristic methods, used independently or in combination with others, take advantage of the special characteristics of the inputs from the real applications, and disregard

[1]Although there are no mathematical proofs, it is believed that NP-complete problems are solvable in $\mathcal{O}(2^n)$, and for all natural NP-complete problems we do have such algorithms.

most artificial worst cases. Among those benign characteristics, the following two are the most frequently observed:

- only solutions of a small size are meaningful, while a solution of a size exceeding a problem-specific threshold is useless, and always disregarded;
- special structures always exist in real applications, such as (when formulated graph-theoretically): the maximum degree of vertices in the graph is upper bounded by a constant independent of the number of vertices; the graph is very sparse (or dense); it is easily decomposable into connected components (there are small separators).

Employing these inspiring facts, numerous efficient algorithms have been developed. In other words, although the problems are hard in general, the hardness can be (partially) relieved when some structural conditions are satisfied. This property turned out to be very general and observed in many problems, on which systematized studies have been conducted, resulting in a new area on algorithmic study, *parameterized computation.*

The main idea in the core of such algorithms is the same: to identify some parameter of small size (k) independent of the problem size (n) which catches the hardness of the problem at hand (solution sizes, structural measures, etc.), and then restrict the exponential explosion of time complexity only to this parameter. The outcome then is another type of super-polynomial functions (note we cannot do away with it totally assuming P $\neq$ NP) of the form $f(k) \cdot n^c$, where f is a computable function[2] dependent only on k and c is a constant independent of k. Since the parameter k is far smaller compared to the input size n, this time complexity is arguably better than any exponential function on n. More importantly, when k is

[2]If the phrase "computable function" means nothing to you, just ignore it.

not large, such algorithms can be implemented and executed in modern computers, which makes this direction very promising.

Albeit rooted from the similar observations of de facto favor, parameterized computation, built on a rigorous theoretic framework, deviates from the heuristic approach significantly. This well-defined theory classifies problems in a two-dimensional way (compared to the one-dimensional classification of traditional complexity theory), and suggests that only *fixed-parameter tractable* problems admit such algorithms. Two *informal* comparisons might reveal some intuitions of the demarcations: 1. most NP-complete problems admit $\mathcal{O}(2^n)$ time algorithms, trivial or not; 2. if the solution size is k, many problems can be solved in $\mathcal{O}(n^k)$ time by enumerating all subsets with size no more than k. The latter one, $\mathcal{O}(n^k)$, is polynomial if k is a fixed constant, however, it is still not practical even if k is as small as 10. Informally and roughly, we can say $\mathcal{O}(f(k) \cdot n^c) < \mathcal{O}(n^k) < \mathcal{O}(2^n)$, where the symbol "<" should be interpreted as "better than".

1.2 Kernelization

Facing a problem hard to be solved directly, people would usually try splitting and/or reducing it, and see what is going on. This *technique*, in its intuitive sense, is so natural that people can master it without any learning. As examples for its occurrences in algorithms:

- when solving the SAT problem, one comes to single-literal clauses first;
- when solving the independent set problem, one only needs to work on the connected components separately.

This list can be very long, and indeed, such steps exist in almost all non-trivial computer programs. Nevertheless, they seldom, if ever, appear in literature on theoretical algorithms (heuristic algorithms are exceptions). The awkward discrepancy can be explained by the rigorous nature of worst-case time complexity analyses in theoretical algorithms, and the fact that above steps were believed to be not applicable for worst cases. This situation changed within the framework of parameterized computation, where the irreducible worst cases normally come with big solution sizes, and thus not interest us. As a matter of fact, the study of kernelization algorithms, previously called "preprocessing and data reduction", can be somehow viewed as a systematic study of such preprocessing steps, complemented by bounding techniques.

Kernelization algorithms, given an instance x and an integer k, in time polynomial to $(x + k)$, produce an *equivalent* and *reduced* instance x' and a smaller integer k' such that the size of the x' ($|x'|$) is bounded by a function of k'. Here by equivalent we mean the original instance x has a solution with size no more than k if and only if the reduced instance x' has a solution with size no more than k'. The reduced instance is called the *kernel* because we assume it is the really hard part of this problem, and for any algorithmic attack running in polynomial time there must be some kernels not surrendering, unless the polynomial hierarchy collapses. Note that the kernel size $|x'|$ is not necessarily bounded by a polynomial function of k', and when it does, we call it a *polynomial kernel*.

It is trivial that a problem is in FPT if it has a kernel, because after the kernel is obtained, whatever algorithms you apply to it, the time is only related to k, and the total time is $f(k) \cdot n^c$. The other direction, albeit not so obvious, also holds true. In other words, a problem is fixed parameter tractable if and only if it admits a kernel. This theorem connects these two concepts in principle, and then only the existence of polynomial kernels is of its own interest. This dissertation will *only* be concerned with polynomial kernels, and unless explicitly specified otherwise, all kernels mentioned in this dissertation are polynomial ones.

In literature, a traditional algorithm is described in three parts: first the procedure of the algorithm, second a proof of its correctness, and finally an analysis of its time complexity. A kernelization algorithm is a little special, in this sense that it involves one more part, the analysis of the kernel size. This is always the focus, and usually the only non-trivial part, of a kernelization algorithm. This feature is mainly due to its heuristic nature: the procedures of most kernelization algorithms can be explained with one or two sentences; their time complexity analyses are trivial; and the correctness of many works is straightforward (exceptions exist, some latest results do involve complicated arguments).

Historically, kernelization algorithms originated from the study of parameterized computation, and were seen in almost all such algorithms ever published. Later, their applicability was found outside of parameterized computation, and they began receiving interests out of parameterized computation community. This transformation escalated after more parameter-independent results, and nowadays, it ripens to call the study of kernelization algorithms as an independent research area. Other than designing kernelization algorithms for concrete problems, this dissertation will also study the several aspects of the nascent theorization of kernelization.

1.3 Clustering

One of the most common tasks in data analyses is to classify a (usually large) set of elements based on their relevance (the data collected). This is called *clustering*, and informally defined By Jain, Murty and Flynn [131] as “the unsupervised classification of patterns into groups (clusters)”. Clustering has incarnations in so many disciplines, including biology, archaeology, geology, geography, business management, and social sciences, and has been approached by statisticians, mathematicians, computer scientists as well as industrial engineers.

To construct such a classification is not hard, provided the given data are perfect, or consistent. The requirement of a data set to be consistent is simple: if element $\mathtt{a}$ is determined to be similar to element $\mathtt{b}$, and $\mathtt{b}$ is similar to element $\mathtt{c}$, then $\mathtt{a}$ must be also similar to $\mathtt{c}$. Unfortunately, there are seldom, if any, data collection methods which can exclude possibilities of errors and inconsistencies. As a consequence, real data always come with errors and in the computational sense, to remove those noise (incorrect information) is equivalent to do the classification.

To exacerbate the situation, we do not have a answer checker, and thus can never know how far our answers are from the reality. Various models have been proposed to measure the solutions, of which the most popular one is "minimum number of modifications", whose basic idea is that the ratio of errors is low in most cases. On one hand, we assume the data to be *almost* consistent, and this is really the case for most modern experiments where instruments and methods have been improved so much. On the other hand, if some data contain too many errors, it does not make sense to use them at all, and we have to repeat the data collection.

Corresponding to make the data consistent with the least amount of modifications, a graph-theoretical formulation of the problem is called correlation clustering that seeks a collection of edge insertions/deletions with the minimum number (or cost when it is weighted) that transforms a given graph into a disjoint union of clusters (cliques).

The correlation clustering model has a flat structure, which is simply a partition where each object belongs to exactly one cluster. Thanks to its simplicity and theoretical beauty, this model has been widely used and intensively studied. How-

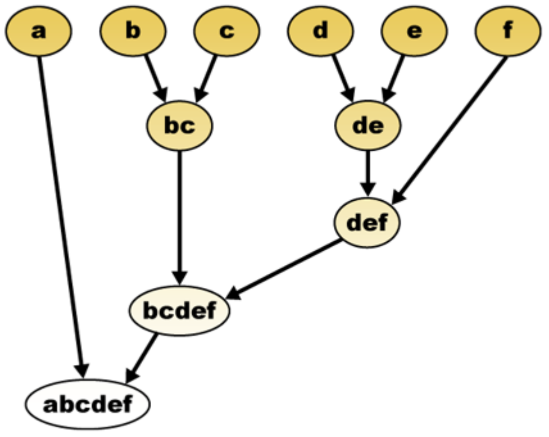


Fig. 1.1. An example of hierarchical clustering

ever, this simplicity comes with cost, and there are many applications whose rich structural information does not fit into a one-level classification. For an instance, to classify six animals: cat, tiger, dog, wolf, frog and toad, we can use three clusters, i.e. (cat, tiger), (dog, wolf) and (frog, toad). This classification is meaningful, but not sufficient, and it is easy to see that the first two classes are closer compared to the last one, which is impossible to be represented in a flat structure. In this case, a two-level structure should be more appropriate, i.e. (((cat,tiger),(dog, wolf)), (frog, toad)). When more species have to be considered, more levels might be needed, and the obtained result should be a structure similar to Figure 1.1[3].

Usually, the relevance between each pair of elements is measured by their distance, and the smaller the more similar. Inspired by above discussion, we also consider the hierarchical structure, such that the data are arranged into a tree structure. All objects are the leaves at level 0, and each non-leaf vertices are at levels between 1 and $\mathsf{M}+1$, such that the distance between two objects is the level of their first

[3]excerpted from http://en.wikipedia.org/wiki/Hierarchical_clustering

common ancestor. By definition the root node is at level $\mathsf{M}+1$. Such a tree is called the M-hierachical clustering tree.

The M-hierachical clustering, very similar to correlation clustering, asks for the minimum number of total modifications to make the given data set into an M-hierachical clustering tree.

1.4 Feedback Set Problems

Feedback sets problems are a collection of problems, whose objective, as the name suggests, is to break all cycles in the given (di)graphs by removing vertices or edges/arcs. There are several incarnations based on the type of the input (di)graphs, and the specified operations. The graphs can be an undirected graph, a digraph, or a tournament, which is a special digraph[4].

Table 1.2
Variations of feedback set problems.

	vertices	**edges/arcs**
Undirected	FVS	maximum spanning tree
Directed	directed FVS	FAS
Tournament	FVS in tournaments	FAS in tournaments

This gives six variations, enlisted in Table 1.2, of which five are NP-hard. The only exception is maximum spanning tree, which is equivalent to the famous minimum spanning tree problem. Thanks to their theoretical importance and wide applications, FVS, DFVS, and FAST are all very popular research topics, where the other two, FAS and FVST, receive only marginal consideration. Particularly, FVS and FAST will be studied in this dissertation.

[4]Think about the result of a tournament which does not allow draws.

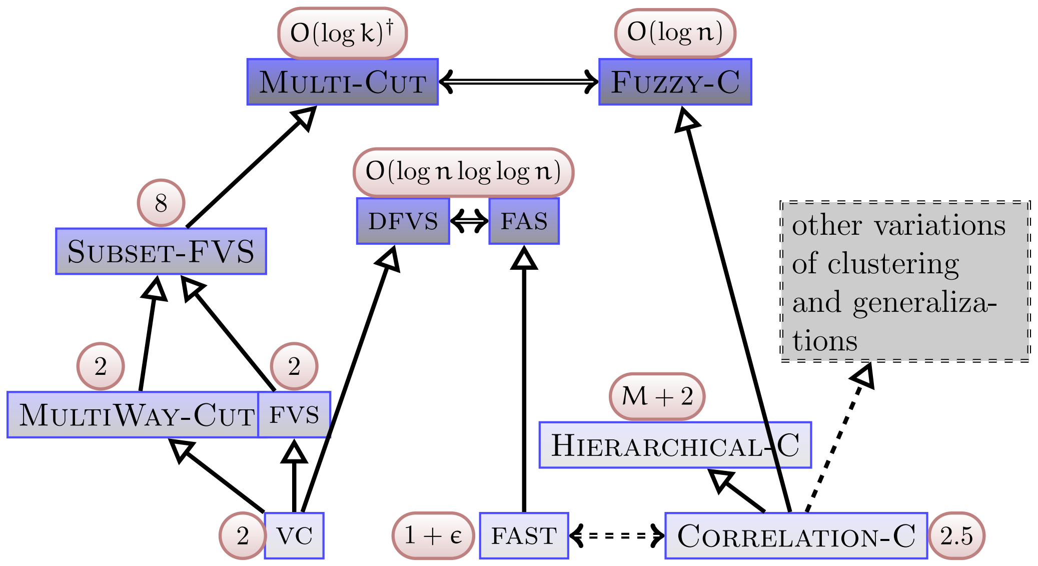
In operating systems, DFVS play a prominent role in the study of deadlock recovery. In the wait-for graph of an operating system, each directed cycle corresponds to a deadlock situation. In order to resolve all deadlocks, some blocked processes have to be aborted. A minimum DFVS in this graph corresponds to a minimum number of processes that one needs to abort. They can also be found in database system, genome assembly, and VLSI chip design.

The feedback arc set in tournaments (FAST) is the FAS problem restricted to tournament. Thanks to increasing interest on data mining, search engine, as well as artificial intelligence, this problem has becomes a hot topic in theoretical computer science, and its identity of NP-completeness was finally settled recently.

Interestingly, the FAST problem is also closely related to correlation clustering problem. They are both. Some techniques are applicable to both, among which the most important ones are linear program and modular decomposition, and in particular, they can be formulated into exactly same linear program.

1.5 A Big Map

As usual, the best way to understand some topics is to put them into a big map, where the related topics and especially their relations present, even most of them are not of direct interest. To understand the problems studied here from algorithmic aspect, the big map comprising the major problems currently under parameterized study, is depicted in Figure 1.2. These problems are listed by increasing hardness (informally and intuitively) from bottom to up, and beside each problem its best approximation ratio is algo given. All of these problem have been shown to be in FPT, however, for most of them no polynomial kernel is known. The five problems listed here which have not been mentioned above are: 1. Vertex Cover (VC) asks for



$A \longleftrightarrow B$ means that problems A and B are computationally equivalent.
$A \longrightarrow B$ means that problem A is a special case of problem B.
$A \dashleftarrow\dashrightarrow B$ means that problems A and B bear striking resemblance, but are not equivalent.
$A \dashrightarrow B$ and the dashed box mean that problem B and their relation are not clear yet.
$\textcircled{2}$ is the best known approximation ratio of this problem.
$\dagger$: k is the number of pairs of requirements.

Fig. 1.2. The major problems under parameterized study

a minimum set of vertices which are incident to all edges. 2. MultiWay-Cut asks for a minimum set of vertices (or edges, then also called MultiTerminal-Cut) whose removal breaks all path between any pair of vertices from a given set of terminals. 3. Subset-FVS asks for a minimum set of vertices whose removal breaks all cycles through a given subset of vertices. 4. Multi-Cut asks for a minimum set of vertices (or edges) whose removal breaks all path between each pair of vertices as given. 5. Fuzzy-Clustering asks for a minimum number of edge addition/deletion to make a graph into disjoint union of cliques, where some pairs bear no cost.

1.6 Outline of This Dissertation

This dissertation starts from a comprehensive survey and literature review in Chapter 2, which also contains the formal definitions of the problems studied, as well as the general notations on graph theory, algorithms, and complexity theory. After that, the concrete results are presented in order.

Chapter 3 studies the correlation clustering problem, and develops the first kernelization algorithm for its weighted version, which in sub-quadratic time produces a $2k$-vertex kernel. The algorithm (Section 3.2) is not the only contribution of this chapter, and is preceded by a series of cutting lemmas (Section 3.1), which is a result of a thorough study of the structural specialties of this problem by relating it with graph edge-cuts. Following from a simple observation: a *densely* connected subgraph with very *sparse* connection to outside vertices should make a cluster, the reduction steps used to obtain the kernel are extremely simple. The only structure involved in the reduction is the closed neighborhood of each vertex, on which the applicability can be efficiently checked, based on its internal density and external connectivity. With quantitative measures defined to measure the density and sparsity, the condition and correctness of the reduction immediately follow from the cutting lemmas, and some elementary counting. The kernel size analysis is even simple. In the reduced instance, by the reduction condition, each vertex not participating in any edited pair will force a large amount of editions in its closed neighborhood, and thus on average, each vertex shares at least one half of editing number (each editing involves exactly two vertices). More interestingly, my approach also works for the unweighted version of this problem, —noting that unlike traditional algorithms, a kernelization algorithm for a weighted problem does can *not* directly apply for its unweighted variation, see 2 for explanation— which also substantially differentiates itself to previous ones. A result that matches the best kernel bound for the unweighted version is described in Section 3.4, which also considers the more general real-weighted version of the problem. On this case, my techniques are still applicable, and lead to a simple kernelization algorithm that constructs a kernel of at most $4k$ vertices. As an intensively studied problems by many researchers, many techniques have been applied on this problem, especially the *crown reduction* and *modular decomposition*. Compared to previous work in literature, my work outperforms not only in the kernel size, but more importantly in efficiency and conceptual simplicity.

Chapter 4 turns to the hierarchical clustering problem, a famous generalization of correlation clustering. The lens used in this dissertation to view this problem is ostensibly different with that used by previous work in literature, namely, it is based on the distance matrix, from which multiple weighted graphs are defined, and thus a graph-theoretic approach can be applied. Details of the formulation are presented in Section 4.1, where the cutting lemmas are also translated into the new language. At the outset (Section 4.2) of the kernelization algorithm, as a demonstration of the power of the cutting lemmas, a $4k$-element kernel is derived by translating the reduction rules and analysis used in Chapter 3. This result substantially improves previous $M \cdot k$ kernel, replacing the multiplicative factor M by a constant 4. Noting that the hierarchical clustering problem contains the correlation clustering problem as a special and simplified case, so the former has been widely believed to be "harder" than the latter in the intuitive sense, and particularly, an M factor was taken for granted. Inspired by this new evidence, which casts doubt on the base under the hardness claim, its parameterized complexity is studied, ending with a non-standard but interesting outcome. Instead of a concrete algorithm, Section 4.3 shows that *any* branch-and-search based parameterized algorithm for the correlation clustering problem can be adapted to the hierarchical clustering problem, with the same time complexity up to a polynomial factor. In addition to the concrete results themselves, one important contribution of this dissertation on this problem is: the hierarchical clustering is not necessarily harder than cluster editing, at least from the aspect of parameterized (exact) computation.

Chapter 5 establishes a comprehensive study on the parameterized complexity of the feedback vertex set problem (FVS) on undirected graphs. In particular, a variation of the problem, the disjoint feedback vertex set problem (disjoint-FVS), to which the FVS can be easily reduced, is examined. The formal definition and details are given in Section 5.1, where a $4k$ kernel is presented. While in principle, the reductions rules presented here to obtain the kernel are only a generalization of what have been previously known and used in literature, a brand-new technique is proposed to analyze the size of the reduced instance, and the new bound for kernel size ensues. Then Section 5.2 is focused on instances having a special topological structure that is closely related to the maximum genus of the graph, and manages to design a polynomial time algorithm to solve such instances. Afterward, Section 5.3 proposes a new branch-and-search process on disjoint-FVS, which effectively reduces a given graph to a graph with the special structure. To precisely evaluates the efficiency of the branch-and-search process, it also introduces a new branch-and-search measure. These algorithmic, combinatorial, and topological structural studies finally bring an $\mathcal{O}^*(3.83^k)$-time parameterized algorithm for the general FVS problem, improving the previous best algorithm of time $\mathcal{O}^*(5^k)$ for the problem.

Finally, after a brief summary in Section 6.1, this dissertation closes with possible directions for future work in Section 6.2. Set around the problems enlisted in Figure 1.2, several possible projects are mentioned, among which the emphasis is placed on two important projects: "kernelization of Multiway Cut (MultiTerminal Cut)" and

“2-approximation of correlation clustering ”. Possible applications of new techniques reported in this dissertation are also introduced there.

2. LITERATURE REVIEW AND DEFINITIONS

The purpose of this chapter is twofold: to formally define the framework as well as the problems studied in this dissertation; and to provide a comprehensive literature review of them[1].

2.1 General References and Notations

The general references include: Bondy and Murty [32] and Diestel [77] for graph theory; Bang-Jensen and Gutin [17] for digraph theory; Cormen et al. [59] and Kleinberg and Tardos [144] for algorithmic techniques; Schrijver [177] for combinatorial optimization; Arora and Barak [14] for PCP theory and complexity theory; Garey and Johnson [107] for NP-completeness and a list of NP-complete problems, from which my notations will follow. The only monograph devoted to tournaments was Moon [159], which is a little bit obsolete, and its notations will not be used here.

For any set A, denote by $|A|$ the cardinality of the set. Unless specified otherwise, graphs are always assumed to be undirected and simple. A graph $G = (V, E)$ is represented as the pair of vertex set V and edge set E, whose sizes are denoted by $n = |V|$ and $m = |E|$ respectively. A graph is a *complete graph* if each pair of vertices

[1]When a paper is available in both conference and journal formats, I will consistently refer the journal version. On one hand, to fully explain a deep algorithmic technique as well as provide proofs with the standard of mathematical rigor, a theoretical paper is usually very long and dense. On the other hand, all conferences proceedings impose hard limits on pages (e.g. 12 pages of LNCS or 10 pages of ACM proceedings), which seldom accommodate full details. Moreover, the technical bugs have far larger probability to escape the one-round reviews of conference papers than the thorough refereeing procedures of journals.
However, each coin has two sides. A journal version might be prepared and submitted many years after the conference version has been reported, especially when significant extensions are required, e.g. [61] and [149]. The notoriously long reviewing periods of journals also impede the appearance time, e.g. [175]. As a result, the date of the publications do not necessarily reflect when the results are actually obtained, and it is not uncommon for an algorithm published this year has been supplanted by others publicized a couple of years ago. The readers should keep this fact in mind when reading through the references.

are connected by an edge. A *clique* in a graph G is a subgraph G' of G such that G' is a complete graph. By definition, a clique of h vertices contains $\binom{h}{2} = h(h-1)/2$ edges. If two vertices v and w are not adjacent, then we say that the edge $[v, w]$ is *missing*, and call the pair $\{v, w\}$ an *anti-edge*. The total number of anti-edges in a graph of n vertices and m edges is $n(n-1)/2 - m$. The subgraph of the graph G *induced* by a vertex subset X is denoted by $\mathsf{G}[\mathsf{X}]$. For a vertex v in a graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$, denote by $\mathsf{N}(v)$ the set of *neighbors* of v, and let $\mathsf{N}[v] = \mathsf{N}(v) \cup \{v\}$ be the *closed neighborhood* of v. For a vertex subset X, $\mathsf{N}[\mathsf{X}] = \bigcup_{v \in \mathsf{X}} \mathsf{N}[v]$, and $\mathsf{N}(\mathsf{X}) = \mathsf{N}[\mathsf{X}] \backslash \mathsf{X}$. The number of neighbors of a vertex v in the graph G is called its *degree*, and denoted by $d_{\mathsf{G}}(v) = |\mathsf{N}(v)|$, where the subscript G is usually omitted when it is clear from the context which graph is being referred to.

For a graph G and an edge subset E' in G, denote by $\mathsf{G} - \mathsf{E}'$ the graph G with the edges in E' removed (the end vertices of these edges are not removed). Similarly, denote by $\mathsf{G} + \mathsf{E}'$ the graph G with the edges in $\mathsf{E}' \subseteq \mathsf{V}^2$ inserted.

Following the recent convention in the literature in exact and parameterized algorithms, I will denote by $\mathcal{O}^*(f(k))$ the complexity $\mathcal{O}(f(k)n^{\mathcal{O}(1)})$ for a super-polynomial function f.

2.2 Parameterized (Exact) Computation

In computation, a decision problem is defined as a subset of language L^* for some finite alphabet L. A problem is *parameterized* when an integer parameter k is attached to it, that is, a *parameterized problem* is a subset of $\mathsf{L}^* \times \mathbb{N}$. A parameterized problem $\mathsf{Q} \subseteq \mathsf{L}^* \times \mathbb{N}$ is classified as *fixed-parameter tractable (FPT)* if there exists a deterministic algorithm A, such that for any given instance $(x, k) \in \mathsf{L}^* \times \mathbb{N}$, A can in time $\mathcal{O}(f(k) \cdot p(|x|))$ determine whether $(x, k) \in \mathsf{Q}$ or not, where p is a polynomial function, and $f : \mathbb{N} \to \mathbb{N}$ is any computable function. Such an algorithm A will be called an FPT algorithm.

Searching for better exact algorithms for NP-hard problems (than the trivial exhaustive search) has attracted interests from researchers even before the definition of NP-completeness was given. The most notable example is Bellman's $\mathcal{O}(2^n)$ time algorithm for traveling salesman problem [20]. Albeit parameterized approximation algorithms began to receive more and more interests recently, parameterized computation is widely considered a new approach of exact algorithms, and most known results are exact, — exceptions exist, such as [33, 157].

Overview. Only a couple of independent parameterized algorithms were known before 1990s. Afterward this line of research was boosted by development of more and more powerful computers, which made algorithms with moderately exponential running time practical. The study of parameterized computation was systematized by Downey and Fellows and their colleagues in a series of important papers [1, 39, 79–84] published in 1990s. Finally, in 1999 they collected those material into a groundbreaking monograph [85]. Parameterized algorithms are also closely related to (non-parameterized) exact algorithms, e.g. [96, 98]. Woeginger surveyed earlier results in two papers [189,190], and recently Fomin and Kratsch wrote a textbook [99] on this topic.

With the theoretic framework built, further studies had a solid base and can go easily. Parameterized computation has forked into two branches, parameterized complexity theory and parameterized algorithms. The first branch was focused on further characterizing the complexity classes, and especially connecting them with traditional complexity classes. One notable success was achieved when the fixed-parameter tractability was studied by relating to the approximability of the problems [38, 53]. The second branch took the positive direction, designing algorithmic techniques and applying them to solve more problems. In 2006, two comprehensive surveys on these two directions were published by Flum and Grohe [95] and Niedermeier [164] respectively. The dissertation is only concerned with the second direction, and particularly new techniques for algorithmic design and their applications. The remainder of this section will be a short overview of the progress of parameterized algorithms.

Branch-and-bound. As mentioned in Section 1.2, a kernel directly implies an FPT algorithm, by applying any trivial brute-force search algorithm on it. For instance, the only known parameterized algorithm for the edge clique cover problem was obtained by applying brute-force search on a 2^k kernel [109], similarly is the first $\mathcal{O}^*(c^k)$ time algorithm[2] for the FVS problem [64]. On the on hand, the algorithms are not restricted to trivial ones, and in most cases, they perform far better. Branch-and-bound is the most universal technique for exact algorithms. For many NP-hard problems, the first algorithms better than the trivial $\mathcal{O}(2^n)$ bound were attributed to this technique. The basic idea of branch-and-bound is to simulate each nondeterministic decision with a *branching*, while discard (*pruning*) a branch as soon as it is determined to be not optimal, by *bounding*. This idea naturally fits into the framework of parameterized computation. With the extra parameter k at our disposal, we can always prune a branch at the moment it uses up its quota, k, and then the depth of a branch can be bounded in some way. Such results (characterized as *branch-on-kernel* mode) are too voluminous to be enlisted comprehensively, therefore I only give two representative examples on vertex cover problem by Chen and colleagues [49, 54], and refer interested readers to Niedermeier [164] and references

[2]see Section 2.5 for details.

therein, which devotes a whole chapter (in the name "Depth-Bounded Search Trees", naturally) to this topic. As a final remark, results of this type, i.e. algorithms purely based on branch-on-kernel, are not popular anymore and seldom found in modern literature, however, it is still a important technique and usually works as a step of a more complicated and involved algorithm.

The time analysis of a branch-and-bound algorithm is not as simple as its procedure. Most of the search tree sizes are computed using recurrence equations, which is a classic method in algorithm analysis [110], and the 72-page paper by Kullmann [146] can be considered as a treatise on it. With more and more branching and bounding rules introduced, such analyses turned to be extremely involved. Some papers were totally devoted to the analyses, e.g. [55]. Results of this type usually had strange numbers as the base, such as $\mathcal{O}(1.2852^k)$ for vertex cover by Chen et al. [54], and $\mathcal{O}(2.6494^k)$ for set splitting by Lokshtanov and Sloper [154]. Actually, with the increase of complication of branching, the overhead soonly dominates, and thus algorithms exploiting this technique, e.g. [56], are of only theoretical merit.

As a new way to conduct and analyze branch-and-bound algorithms, the technique *measure and conquer* was first proposed for non-parameterized algorithms by Fomin et al., resulting in some breakthroughs for notoriously hard problems like independent set and domination [97,98]. It was immediately adpoted by parameterized computation community and is a part of many important results, such as [51,57,187]. As a tentative exploration, van Rooij and Bodlaender even tried to automatically generate new measures [186].

Iterative compression. As suggested by the name, this technique tries to construct a smaller solution out of a known feasible solution, if such one exists. Originally designed by Reed et al. to give the first FPT algorithm for the odd cycle transversal problem [170], this was later shown to be extremely powerful. Indeed, immediately after its apprearance, dozens of papers were published simply applying iterative compression to new problems, as surveyed by Guo et al. [116]. The earliest applications of this technique contain no more than trivial adaption, while later results, usually combining iterative compression with other non-trivial techniques, did bring deep and wide influence. The most notable example should be the Chen et al.'s algorithm for the directed feedback vertex set problem [57], which settled one of the longest open problems in parameterized computation in a positive way. As a remark, Chen et al.'s algorithm also involved measure and conquer.

Other parameters. Other than the solution size, the second most used parameter is tree-width (branch-width, clique-width, etc.). These concepts were formulated and studied as a part of the seminal project "graph minors" by Robertson and Seymour, and scattered in the series of papers [172–174]. The basic idea is very similar to the Lipton-Tarjan separator theorem [151, 152], which finds a small set of vertices wich separates the graph into two balanced parts, as well its follower, Baker's outplanar decomposition [16]. This line of study turned to be most successfully in sparse graphs, on which it provides a general way to design subexponential parameterized algorithms and also polynomial-time approximation schemas, such as $\mathcal{O}(2^{15.13\sqrt{k}})$ for dominating set [7, 102, 136], and $\mathcal{O}(2^{4.5\sqrt{k}})$ for vertex cover problem [10, 101], both on planar graphs. This work was generalized into graphs excluding fixed minor. Finally, this study of algorithmic graph minor theory is now widely know as *bidimensionaltiy theory* [68–76].

Very recently, some efforts have been paid on the utilization of other parameters. One scheme is directly inspired by the trivial $n/4$ lower bound of independent set and $n/2$ of MAX-SAT. This direction, called *parameter above a tight lower bound,*, was investigate widely by Gutin and Yeo and their colleagues, and has resulted in many results [12,60,121–126]. The other direction went even farther, that is, it looked for a set of parameters, instead of a single one. This line of research is called *multivariate parameters*, and Niedermeier recently surveyed the latest progress [165].

2.3 Kernelization Algorithms

Albeit the word "algorithm" appear in the name, a kernelization algorithm does not solve a problem by returning a solution, as a regular algorithm would do. Given a parameterized problem $(x, k) \in Q \subseteq L^* \times \mathbb{N}$, a kernelization algorithm A in time $p(|x| + k)$ transforms (x, k) into another instance (x', k') *of the same problem*, where p and q are both polynomial functions, such that

$$k' \leqslant k, \qquad |x'| \leqslant q(k')$$

and x and x' are *equivalent*. Here by equivalent we mean $(x, k) \in Q$ iff $(x', k') \in Q$, and the optimal solutions of x and x' can be traceable to each other. The new instance (x', k') is called the kernel, and $q(k')$ is the size of the kernel[3].

There is a very important and subtle point on the definition of kernelization algorithms that has been widely and undeservedly ignored. Conventionally, given an instance, an algorithm returns a solution to this instance. Thus, we can always feed an unweighted instance into a algorithm designed for its weighted version, by trivially assigning weights (most time uniform unit weights will suffice). This practice, although inefficient, is guaranteed to work in principle and a corrected solution can always be expected. In this sense kernelization algorithms, whose return are reduced

[3]This definition is different from that in literature, by restricting the kernel size $q()$ to be polynomial. See below for the explanation.

instances instead of final results, are distinct from others. The instance returned by a kernelization algorithm for a weighted problem is normally weighted, even the input instance is unweighted. In other words, the input and output instances are different and thus it violates the definition of kernelization algorithms. Thus far there are only very few studies on the kernelization algorithm on weighted problems, not to mention that algorithms applicable for both unweighted and weighted versions for the same problem.

(Pre-)History. The earlist work on data reduction can be traced back to the 1950's, when Quine [168] considered the simplification of truth functions.

It is both natural and easy to do reduction, especially for prohibitively hard problems. The satisfiability (sat) problems asks whether or not there is an assignment to a formula in conjunctive normal form (CNF-formula) such that it returns true, i.e. is it satisfiable. This requires each clause to be satisfied, and hence there is no choice for the unit-literal clauses which has only one satisfactory assignment. More specifically, the literal in a unit-literal clause must be assigned accordingly. The other observation is that if all occurrences of a variable are in the same form, all positive or all negative, assigning "true"or "false" to it will satisfy all clauses containing it without sacrificing possibilities of satisfying other clauses. The two reductions described here are only a tip of the iceberg, and there are large amount of similar reductions proposed and applied only for sat problem, which show significant improvement in practice. There are similar results for other problems, and actually, all heuristic algorithms have such reduction steps in essence.

Kernelization algorithms are more than reductions. Above operations might reduce the instances, nevertheless, they are not kernelization algorithms. They do not satisfy the definition of kernelization algorithms by one imporant element missing, which is, a provable bound on the kernel size. The study of data reductions in the systematics sense, that is, kernelization algorithms, was first carried out after the introduction and development of parameterized computation. In this sense, the history of study of kernelization started from 1990's. For more details on the evolution of kernelization, refer to the textbook of Niedermeier [164], as well as the surveys by Guo and Niedermeier [117] and Bodlaender [27], and also the references therein.

Relation to parameterized algorithms. If we drop the condition for the kernel size to be polynomial, a problem has such a kernel if and only if it is fixed-parameter tractable. This well-known equivalence was established by Cai et al. [40]. This theorem, though of theoretical importance and a fundamental position in the theory, has no any practical merit, as no kernel of exponential size derived from a parameterized algorithm (the proof in [40] is constructive!) is interesting. On this ground, the study of super-polynomial kernels is not of its own interest. This explains the requirement of kenels to be of polynomial size in the definition given at the beginning of this section.

The connections between kernelization algorithms and parameterized algorithms are not limited to the theoretical sense. Indeed, they are frequently used together to obtain the best speed-up, as the general method proposed by Niedermeier and Rossmanith [166]. However, a kernelization algorithm also has overhead, and if it is invoked too frequently, the overhead will overshadow its benefit. This situation is very similar with the branch-and-bound algorithms. Thus, how to adjust the invocations of kernelization algorithms in a parameterized algorithms is a practical problem, which should be investigated with an experimental approach, and in real applications a flexible way might be the best. Some preliminary results on this include [8, 115].

Concrete results. Since most parameterized algorithms contain kernelization as a step, there are numerous results on variant problems. Here I trace the development of two of the most important problems, vertex cover and FVS.

As the best studied problem, vertex cover has attracted most attention from the beginning, and the first kernel was reported by Buss and Goldsmith in 1993 [37]. Their quadratic kernel was obtained by reductions on vertices of degree 1 and degree $> k$, which are justified by two very simple observations: For a 1-degree vertex, its neighbor is always a better option; While for a $(> k)$-degree vertex, there is no other choice other than putting it into the solution. I remark similar observations are able to give polynomial kernels for many problems, as surveyed in [164].

Above reductions are of a local feature, that is, they can be applied locally, and therefore are very easy to be implemented. Whereas to further improve the kernel size, local techniques seem to not work, and global structures have to be considered, which obviously take more time. Inspired by a famous theorem of Nemhauser and Trotter [162], Chen et al. [54] presented the first $2k$ kernel, which, however, is not efficient enough for some instances [48]. Later, Fellows applied the *crown reduction* to obtain a $3k$ kernel [92], which turned to be very efficient in experiments [2]. These two approaches, originally considered orthogonal, were later shown to be closely connected by Chlebík and Chlebíková [58], and now it is well known that the NT thoerem is really equivalent to the strong crown reduction.

FVS is harder than vertex cover in all measures, including kernelization. The first polynomial kernel, reported by Burrage et al. [36], has a degree as large as

11! This was improved by Bodlaender into cubic [25], and further to quaratic by Thomassé [185].

The early results on kernelization algorithms were surveyed by Guo and Niedermeier [117].

Attacks on planar graphs. Once a problem is shown to be fixed-parameter intractable, it is meaningless to search for kernelization algorithms. Thus, dominating set problem, known to be $W[2]$-hard [85], is not expected to have kernels in general graphs. Fortunately, this hardness does not carry to its planar version, while the NP-hardness does [107]. Alber, et al. managed to give a linear kernel for planar dominating set problem [9], and shortly after its appearance the kernel size $335k$ was improved to $67k$ by Chen et al. via more careful analysis [50]. Compared to the result itself, the technique, adapted by other researchers and shown to be extremely general, turned out to be far more important. Basically, it consists of two steps: First construct reduction rules based on the properties of dominating set problem; Second analyze the kernel size with help of topological strucutre of planar graphs.

As expected, the analyses technique in the second step is the essence of this work. The two steps are well separated: The first step is specialized for dominating set problem, without any properties of planarity; While the second step mainly uses topological properties of planar graphs. This separability enables it work also for other planar problems. To give such a kernel, one only needs to design reduction rules with the properties of the specific problem. Within this framework, numerous problems were shown to admit linear kernels on planar graphs, including connected dominating set [153], induced matching [135], full-degree spanning tree [119], cycle packing [31]. There was also an immature attempt to formalize this technique [118], which, unfortunately, heavily relies on intuition at several critical places, therefore, a systematic theory on this technique satisfying the standard of mathematical rigor is still at large.

Generalization. On one hand, earlier attempts for general technique did not end with success, on the other hand, more and more concrete results kept being proposed. For years people became more and more eager for such a theory. This thirst was only quenched by a couple of positive results published in 2009 and thereafter. Thus far we have three major results in this category, while more are expected.

The first success was Kratsch's proof of existence of polynomial kernels for problems in some complexity classes[4] [145]. Such classes were orginally defined in the study of approximation algorithms, while later Cai and Chen [38] built connection between them and FPT by showing problems in them are always FPT.

The scecond result seems more interesting. Bodlaender et al. showed polynomial kernels for problems satisfying certain conditions [29]. More specifically, it set two set of conditions, for which linear kernels and quadratic kernels follow directly. Their results were not limited to planar graphs, instead, it considered all graphs embeddable into fixed surfaces whose genuses are bounded. Moreover, the conditions were related to Courcelle's logical formulation and Robertson and Seymour's Minor theory.

The third result, very close to the second one, should also be stamped in 2009. In the framework of bidimensionaltiy theory, Fomin et al. [100] studied the graphs which avoid a fixed minor, and showed many bidimensional problems have small kernels on those graphs. Bidimensionality theory has been extensively studied and its applications on parameterized algorithms and approximation algorithms have been very well-known for a long time, whereas, for a long time, it have no applications

[4]MIN $F^{+}\Pi_1$ and MAX NP

in kernelization algorithms found. Given that they three categories are so closely related, Demaine et al. conjectured such applications [70]. [100] actually confirmed this conjecture, and consequently made bidimensionality theory more complete..

Thus far, this line of research, still at its incipient stage, has not provided any benefit for the design of concrete kernelization algorithms. In this sense, it becomes interesting on how to make connections between the theoretical results and those concrete ones. The reader is referred to the excellent survey by Bodlaender [27] .

Lower bounds. Common sense holds that for studies on algorithms, the negative direction is always (far) harder than the positive one. This general principle also holds for kernelization algorithms. Therefore, it is not strange that such studies are left behind the algorithmic techniques. Here the negative results are only concerned with the existence of polynomial kernels, or more concrete bound (note that the existence of kernels is equivalent to the identity of FPT, and therefore is not of independent interest).

Again, lower bounds of kernelization can be related to the study of approximation, or more specifically, (in)approximibility. If a problem admits a linear kernel[5] of size $\mathsf{c}\mathsf{k}$, this kernel can be returned directly as an approximation solution with ratio c. Thus, directly following the inapproximability results in literature, we have lower bound for kernel size. The most famous result of this type is again on vertex cover problem, which can be approximated with any constant ratio bettern than 2 [143], assuming Unique Game Conjecture [142]. Thus, the $2\mathsf{k}$ kernel given above is already optimal and cannot be improved.

There is another way to provide lower bounds in a problem-specific manner, which is based on the duality relations between problems. The duality relations

[5]the definition and controvesies of linear kernels are explained later.

have been well-known in algorithmic study for a long time, and directly involved in several important algorithms, among which the most famous one should be the duality of linear programming [62, 163, 176]. Chen et al. [50] defined "parametric duality", which is a parametric version of the duality relation, such that there is a linear relation between the sizes of the solutions of the problems. Based on this, they proposed a new approach for lower bounds of kernelization. Note that for any NP-hard problem Q, and any given kernelization algorithm A of it, there must be some instance I which A cannot handle, unless **P=NP**. Moreover, if two problems are parametric duals of each other and both have kernelization algorithms, we can try both algorithms on an instance of one problem, for it is also an instance of its dual problem. This means there must be instances which can resist the attacks from both kernelization algorithms, that is, there must be a gap between two sides. Then any kernelization algorithm for a problem gives a lower bound for its dual problem. One important concrete result is the $4/3 \cdot \mathsf{k}$ bound for planar vertex cover problem (note the $2\mathsf{k}$ bound for vertex cover given above does not carry to its planar version).

A more promising result was only recently proposed by Bodlaender [28]. They showed problems and/or-compositional and satisfying specific conditions have no polynomial kernels. This result was based on a recent result in complexity theory of Fortnow and Santhaman [103].

Kernel size analysis. The proof of the ratio of an approximation algorithm is usually very invovled, and it becomes more prohibitive if the tight analysis is asked. Some approximation algorithm was later proved to be of better bound, among which the most famous one was given by Chen et al. [52], which improved the analysis and showed a tight ratio for Johnson's approximation algorithm for MAX SAT [132]. There are still many approximation algorithms whose tight analyses are still open, e.g. shortest superstring problem [21]. This situation is very similar for the kernelization algorithms, where the analysis of the kernel size is normally the hardest in such an algorithm. Only few kernels came with a tight examples.

The analysis of Fellows' algorithm on vertex cover problem based on crown reduction might receive most attention. The original bound given was $3k$ [92], however, after opened for a long time, which was fianlly settled to be $2k$ [58].

Mathematical tools specialized for kernel size analyses might also be interesting, while no such results available yet.

2.4 Clustering

Classifying objects is one of the innate abilities of us, and also one of the most important activities in human life. Simply speaking, a cluster is a set of entities which are alike, and entities from different clusters are not alike [90]. In applications, some prior knowledge on the final results might or might not be available, and accordingly, they are called supervised and unsupervised classifications. Both of which are well studied, while this dissertation will be concerned only the second one, that is, unsupervised classifications without any prior knowledge. which is also widely known as clustering.

This clustering and related problems are really universal, and can be found in literature of almost all discplines. To indicate how popular the stuies on clustering are, one only need to search for the venues where papers titling "cluster analysis" were published. The number of journals is at least 3000. The classic textbook dedicated to clustering was by Everitt et al. [90]. and a comprehensive algorithmic-biased survey of earlier work on clustering can be found in Jain, Murty and Flynn [131]. More recent progress are concluded in several textbooks and treatises [106, 148, 156].

Background. As just mentioned, this problem arose naturally in almost everywhere, and therefore has been studied in many different research communities. At the beginning, the communications between them were very scarce, and in such an "unsupervised" time, it was not uncommon for one work conducted by several groups independently and published in different journals without knowing each other for many years. With the same reason, they named it differently, such as *numerical taxonomy* in biology and ecology [181–183], *unsupervised learning* in artificial intelligence and machine learning [86], *segmentation* in computer vision and medical image processing [191,192]. Somehow surprisingly, efforts from totally different background brought very similar outcomes. Moreover, nowadays, people have realized the existence of each other, and want to have further studies coordinated, and as a result, a new research area specialized on clustering problem has emerged.

The earliest efforts were on the formulations. Note that given any two solutions for a clustering problem, there are no ways to deterministically tell which solution is the better. Thus the first task must be deciding a criterion, such that algorithms can be designed and judged according to such a criterion. Indeed, severl criteria have been proposed, each with its merit as well as weakness, and it is believed that there is no a best one in them [137].

In literature, according to the structure of resulted clusters, clustering is generally classified as partitional and hierarchical [131].

Correlation clustering. Basically, the *partitional clustering* asks for partition given objects, such that some conditions are satisfied [193]. One notable model of partitional clustering is the the correlation clustering problem, whose objective is to minimize the dissimilarities between objects in the same group, and the similarities between objects of different groups. The formal definition is:

> correlation clustering: Given (G, k), where $G = (V, E)$ is an undirected graph and k is an integer, is it possible to transform G into a union of disjoint cliques by edge deletions and/or edge insertions such that the total number of the inserted edges and deleted edges is bounded by k?

The strucutre of an objective graph consists of disjoint union of cliques, and in this sense, this is a very well-structured problem. Thanks to its extreme simplicity, it received most interests, and many algorithms were reported [131,180,193].

Hierarchical clustering. Unlike correlation clustering, the results of hierarchical clustering is a hierarchical tree[6], such that all objects are the leaves at level 0, and two objects with distance d first meet at the level d [90,131,193]. Since the distances are usually given in the form of a matrix, where i-row j-column element D_{ij} is the distance between object i and j. Such a matrix is called *proximity matrix* or *distance matrix*, this problem is usually formulated with language of matrix:

> hierachical clustering: Given (X, D, k), where X is a set of n elements, D is a $n \times n$ integer matrix with values between 0 and $M + 1$, and k is an integer, is there an ultrametric distance matrix D' such that $d(D, D') \leqslant k$?

where *ultrametric* means $D_{ij} \leqslant \max(D_{il}, D_{jl})$ for all triples $i, j, l \in X$ and $d(D, D') = \sum_{1 \leqslant i < j \leqslant n} |D_{ij} - D'_{ij}|$. It is easy to see that correlation clustering problem is the special case of hierachical clustering problem, when $M = 1$.

Naturally, there are two directions to solve the problem, i.e. top-down and bottom-up. Agglomerative hierarchical clustering starts from the bottom, and itera-

[6]Some variations of hierarchical clustering have another requirement, the hierarchical tree be binary [120].

tively conducts merging operations. At the beginning of an agglomerative hierarchical clustering algorithm, there are n clusters, each of which consists of a single object. They correspond to the 0-level of the hierarchical tree. Before the i-th iteration, all clusters at and below i-level have been settled, and the algorithm merges clusters at i-level into clusters of $(i+1)$-level. Consequently, the whole hierarchical tree is constructed after M iterations. Algorithms in this category include single linkage algorithm [90, 131, 161], "BIRCH (Balanced Iterative Reducing and Clustering using Hierarchiies)" of Zhang et al. [194], "CURE (Clustering using Representatives)" of Guha et al. [112], and "ROCK (Clustering using Representatives)" of Guha et al. [111].

On the opposite way, divisive hierarchical clustering starts from the top, and iteratively conducts operations of division. One example is "DIANA (Divisiave ANAlysis)" of Kaufman and Rousseeuw [141]. Note that algorithms for the correlation clustering problem can be used for the iterative step of divisive hierarchical clustering algorithms.

All above algorithms are of heuristic nature, and only very recently, the systematic studies of hierarchical clustering, including approximation and exact algorithms, were started by some computer scientists.

Complexity and approximation. The hardness of hierarchical clustering problem has been known for a long time. Specifically, it was shown to be NP-hard by Křivánek and Morávek in 1986 [147], and APX-hard by Agarwala et al. in 1999 [3].

Comparatively, the hardness results of correlation clustering turned to be far more complicated (note that any hardness result on correlation clustering problem, as a special case of the hierarchical clustering problem, directly applies for the later). The NP-hardness of correlation clustering was only settled in 2004 by two groups from different areas [18,179]. This is also a concrete example of how this problem was studied by researchers by different background without knowing each other. Immediately, the APX-hardness was also settled, by Charikar, Guruswami, and Wirth [45].

For polynomial-time approximation algorithms of correlation clustering, The best result is a randomized approximation algorithm of expected approximation ratio 3 by Ailon, Charikar, and Newman [6]. Ailon and Charikar then generalized their approach in [6] to hierarchical clustering problem, ending with a randomized approximation algorithm of expected approximation ratio $M+2$ [5]. Both were later derandomized by van Zuylen and Williamson [188], with the same approximation guarantees.

Parameterized and kernelization algorithms. The parameterized study on unweighted correlation clustering[7] was first taken by Gramm et al. [108], whose results included an $\mathcal{O}(2.27^k + n^3)$ algorithm and a kernel of $\mathcal{O}(k^2)$ vertices. This result was immediately improved by a successive sequence of studies on kernelization algorithms that produce kernels of size $24k$ by Fellows et al. [93], and of size $4k$ by Guo et al. [113]. For the weighted version of this problem, to my best knowledge, the only work was done by Böcker et al. [23], in which an $\mathcal{O}(1.82^k)$ was given. Böcker et al. [23] also porposed a "quadratic kernel", which, however, satisfied neither the definition of this dissertation (given in Section 2.3) nor any previous literature, because there is no guarantee that the kernel is still an instance of weighted correlation clustering problem, instead, it becomes a far harder problem. Based on these algorithms, experimental studies were also carried out, such as [24, 65].

[7]In parameterized computation community, this name cluster editing is preferred and more widely used. Since this dissertation is studying this problem as a variant of clustering problems, the terminology of clustering community suits better.

Only recently, Guo et al. [115] reported the first kernel of $\mathcal{O}(M \cdot k)$ objects for hierarchical clustering, by generalizing their kernelization algorithm in [113]. In the same paper, they also provided an $\mathcal{O}(3^k)$ parameterized algorithm based on trivial branching. However, their kernelization algorithm suffers from the high time complexity and therefore, in the experiments they conducted, they had to use another kernelization algotihm with kernel size $\mathcal{O}(k^2)$ rather than the $\mathcal{O}(M \cdot k)$ one (note that, naturally, $k \gg M$ in any non-trivial instances).

Known results on these two clustering problems are summarized as in Table 2.1.

Table 2.1
Previous results for the correlation clustering and M-hierarchical clustering problems

Approaches	correlation clustering	hierarchical clustering
approximation ratio	$2.5^\dagger$ [6] 2.5 [188]	$M+2^\dagger$ [5] $M+2$ [188]
exact	$\mathcal{O}^*(1.62^k)$ [22]	$\mathcal{O}^*(3^k)$ [115]
kernelization	$4k$ [113]	$(2M+4)k$ [115]

†: randomized approximation

2.5 Feedback Sets

The general definition for this family of problems is: Given a (directed or undirected, or with further restriction) graph, find the minimum number of vertices (arcs) whose removal leaves the graph acyclic.

Background of FVS. The origin of these problems was on the study of operating systems, within which the first formulation, DFVS, was proposed in 1960's. Later, similar applications in database and VLSI design were reported, and its undirected counterpart FVS was also considered. Although lots of efforts were put, the algorithmic study did not go well at the beginning. These two problem were in the Karp's list of NP-hard problems [139], and since it is at least as hard as vertex cover [139], it is APX-hard [78]. Earlier results were concluded in a comprehensive survey by Festa et al. [94].

Unlike other famous problems, there are no trivial approximation algorithms for FVS. For FVS, the first approximation algorithms appeared only in 1999, when two independent results were reported by Bafna, Berman, and Fujito [15] and Even et al. [89]. Both results have the same approximation ratio of 2, which is the best one can expect assuming Unique Gmae Conjecture as well as Khot's inapproximability result on vertex cover [142, 143]. For DFVS, this is even worse. The best approximation obtained ratio was $\mathcal{O}(\log k \log \log k)$, independently reported by Seymour [178], and Even et al. [88].

Parameterized algorithms of FVS. The FPT theory shed some light, and the situation drastically changed thereafter. Downey and Fellows first showed its fixed-parameter tractability. Albeit the time complexity of their algorithm is terrible ($\mathcal{O}(17(k^4)!n^{\mathcal{O}(1)})$), it triggered an explosive studies on this topic, as enlisted in Table 2.2.

Unfortunately, on the directed side, the study of parameterized computation was stucked, and for two decades, it withstood vehement attacks and no algorithmic techniques applied for DFVS. Only in 2008, it was finnaly shown to be in FPT by two groups [57]. Given that this algorithm is not satisfactory in pactice, work is still in progress to further improve it.

The study on kernelization algorithms of FVS is also fruitful, starting from the first polynomial kernel of size $\mathcal{O}(k^{11})$ by Burrage et al. [36], $\mathcal{O}(k^3)$ by Bodlaender [25],

Table 2.2
The history of deterministic parameterized algorithms for FVS

Authors	**Complexity**	**Year**
Downey and Fellows [80]	$\mathcal{O}(17(k^4)!n^{\mathcal{O}(1)})$	1992
Bodlaender [26]		1994
Downey and Fellows [85]	$\mathcal{O}((2k+1)^k n^2)$	1999
Raman et al. [169]	$\mathcal{O}(\max\{12^k, (4\log k)^k\} n^{2.376})$	2002
Kanj et al. [134]	$\mathcal{O}((2\log k + 2\log\log k + 18)^k n^2)$	2004
Raman et al. [169]	$\mathcal{O}((12\log k/\log\log k + 6)^k n^{2.376})$	2006
Guo et al. [114]	$\mathcal{O}((37.7)^k n^2)$	2006
Dehne et al. [64]	$\mathcal{O}((10.6)^k n^3)$	2005
Chen et al. [51]	$\mathcal{O}(5^k k n^2)$	2008

and finally $4k^2$ by Thomassé [185]. The quadratic result is tight somehow, for Dell and van Melkebeek recently showed that it is unlikely for FVS admits a kernel with less than $\mathcal{O}(k^2)$ edges [27, 66]. It is still open for the existence of polynomial kernels for DFVS.

There were also studies on randomized parameterized algorithms and exact (non-parameterized) algorithms. Becker, Bar-Yehuda, and Geiger reported an $\mathcal{O}(4^k)$ randomized algorithm [19]. Since the complement of an FVS is a forest, and then the removal of FVS corresponds a maximum induced forest. With this observation, Fomin et al. proposed an $\mathcal{O}(1.7548^n)$ exact algorithm by growing the forest [96].

FAST. In spite of the trivial fact that the FAST problem was a special case of FAS problem, FAST was formulated in totally different . Historically, since it is defined on tournament, it was briefly studied from the combinatorial aspect by Moon [159, 160], who spent the whole professinoal life on tournaments and related problems.

The algorithmic study was first mdad by Dwork et al., as a formulation for their revolutionary concept *meth-search engine* [87]. Their work actually successfully popularized this topic, and started a exponentially increasing number of publications [6, 13, 91, 140, 188].

Relation of FAST to correlation clustering. The objective of both problems is ask for minimum number of adjustment to a graph/digraph to make it transitive. Moreover, for both problems, the forbidden structures are the (directed) triangles, such that a graph satisfy the specified condition if and only if it contains no such triangles. This relation was reflected in the “one stone two birds” approximation algorithm for both FAST and correlation clustering of Ailon et al [6].

For the weighted version, if the weights are arbitrary, or more specifically, 0 is allowed, both problems become far harder. Actually, FAS is the FAST with 0-weight arcs, and was notoriously hard, whose identity of FPT was recently solved [57]. Similarly, when 0-weight edges are allowed, correlation clustering becomes fuzzy clustering (also called correlation clustering with “no care” edges), which is computationally equivalent to the multi-cut problem, which was also shown to be in FPT very recently [34, 158].

3. CORRELATION CLUSTERING[⋆]

The main result of the this chapter is a general kernelization algorithm for the *correlation clustering* problem, working for both unweighted and weighted versions. This algorithm not only significantly simplifies and improves previous result on unweighted version, but more importantly, gives the first polynomial kernel for the weighted correlation clustering problem [1]:

Theorem 3.1. *There is a polynomial-time kernelization algorithm for the (weighted) correlation clustering problem that produces a kernel that contains at most* $2k$ *vertices.*

To make this chapter self-contained, we recall some definitions here. Let $G = (V, E)$ be an undirected graph, and let V^2 be the set of all unordered pairs of vertices in G (thus, for two vertices v and w, $\{v, w\}$ and $\{w, v\}$ are regarded as the same pair). Let $wt : V^2 \to \mathbb{N} \cup \{+\infty\}$ be a *weight function*, where $\mathbb{N}$ is the set of positive integers. The *weight* of an edge $[v, w]$ in G is defined to be $wt(v, w)$. If vertices v and w are not adjacent, and we add an edge between v and w, then we say that we *insert an edge* $[v, w]$ *of weight* $wt(v, w)$. The weighted correlation clustering problem is formally defined as follows:

> Weighted correlation clustering: Given (G, wt, k), where $G = (V, E)$ is an undirected graph, $wt : V^2 \to \mathbb{N} \cup \{+\infty\}$ is a weight function, and k is an integer, is it possible to transform G into a union of disjoint cliques by edge deletions and/or edge insertions such that the weight sum of the inserted edges and deleted edges is bounded by k?

[⋆] Reprinted with permission from *Cluster editing: kernelization based on edge cuts* by Y. Cao and J. Chen, 2011, Algorithmica, doi:10.1007/s00453-011-9595-1, Copyright 2011 by Springer-Verlag.

[1] Böcker et al. reported a quadratic (vertex) kernel in [23], however, it does not satisfy the definition of Section 2.3, or any other accepted definitions. In particular, their kernelization algorithm introduces “no-care” edges, and the output of it does *not* remain an valid instance of weighted correlation clustering problem, instead, it becomes a far harder problem.

And the unweighted version can be treated as $wt(\{v, w\}) = 1$ for each pair of vertices $v, w \in V$.

A more general version of the weighted correlation clustering problem is defined with real weights, that is, the weight function wt takes values in $\mathbb{R}_{\geqslant 1} \cup \{+\infty\}$, where $\mathbb{R}_{\geqslant 1}$ is the set of all real numbers larger than or equal to 1, and correspondingly the parameter k is a positive real number. Our techniques are also applicable for this more general version and yield a polynomial-time kernelization algorithm for this version with a kernel of at most $4k$ vertices.

We would like to remark on the techniques we have used in this research:

1. the cutting lemmas are of potential use for future work on kernelizations and algorithms;

2. both the idea and process are very simple with efficient implementations. Indeed, there is a single reducible condition on which a series of reduction steps are applied in order. The reducible condition and the reduction steps are applicable to both weighted and unweighted versions;

3. the reduction process to obtain the above kernel results is independent of the parameter k, and therefore is more general and applicable;

4. compared to the approach based on critical cliques (i.e., simple series modules), our approach has the following advantages:

   (a) our approach is applicable to the weighted versions of the problem, while it seems quite difficult to generalize the techniques based on modular decomposition to handle weights; and

   (b) our approach has a single-pass reduction while the methods based on modular decomposition require iterations and re-constructions.

3.1 Cutting Lemmas

Let $G = (V, E)$ be a graph, and let $S \subseteq V^2$. Denote by $G \triangle S$ the graph obtained from G as follows: for each pair $\{v, w\}$ in S, if $[v, w]$ is an edge in G, then remove the edge $[v, w]$ in the graph, while if $\{v, w\}$ is an anti-edge, then insert the edge $[v, w]$ into the graph. The set S is a *solution* to the graph G if the graph $G \triangle S$ is a union of disjoint cliques. The *weight* of a set $S \subseteq V^2$ of vertex pairs is defined as $wt(S) = \sum_{\{v,w\} \in S} wt(v, w)$. In particular, a set E' of edges in G defines a set of vertex pairs in a natural way, so the *weight* of the edge set E' is defined as $wt(E') = \sum_{[v,w] \in E'} wt(v, w)$. For an instance (G, wt, k) of the correlation clustering problem, denote by $\text{opt}(G)$ the weight of an optimal (i.e., minimum weighted) solution to the graph G.

For two vertex subsets X and Y, denote by $P(X, Y)$ the set of all vertex pairs $\{v, w\}$ where $v \in X$ and $w \in Y$, by $P_E(X, Y)$ the set of edges $[v, w]$ where $v \in X$ and $w \in Y$, and by $P_A(X, Y)$ the set of anti-edges $\{v, w\}$ where $v \in X$ and $w \in Y$. For a vertex subset X, define $\overline{X} = V \backslash X$, and the edge set $P_E(X, \overline{X})$ is called the *cut of* X. The weight of the cut of X is denoted by $\gamma(X) = wt(P_E(X, \overline{X}))$. Obviously, $\gamma(X) = \gamma(\overline{X})$. As a shorthand, $S(X, Y) = S \cap P(X, Y)$ is the set of pairs in S in which one vertex is in X and the other vertex is in Y.

Behind all of the following lemmas is a very simple observation: as a hereditary property, any induced subgraph in a cluster graph is also a cluster graph. Therefore, a solution S to the graph G restricted to an induced subgraph G' of G (i.e., the pairs of S in which both vertices are in G') is also a solution to the subgraph G'.

Lemma 3.2. *Let $\mathcal{P} = \{V_1, V_2, \ldots, V_p\}$ be a vertex partition of a graph G, and let $E_{\mathcal{P}}$ be the set of edges in G whose two ends belong to two different parts in $\mathcal{P}$. Then $\sum_{i=1}^{p} opt(G[V_i]) \leqslant opt(G) \leqslant wt(E_{\mathcal{P}}) + \sum_{i=1}^{p} opt(G[V_i])$.*

Proof. Let S be an optimal solution to the graph G. For $1 \leqslant i \leqslant p$, let S_i be the subset of S such that each pair in S_i has both its vertices in V_i. As noted above, the set S_i is a solution to the graph $G[V_i]$, which implies $\text{opt}(G[V_i]) \leqslant wt(S_i)$. Thus,

$$\sum_{i=1}^{p} \text{opt}(G[V_i]) \leqslant \sum_{i=1}^{p} wt(S_i) \leqslant wt(S) = \text{opt}(G).$$

Moreover, if we remove all edges in $E_{\mathcal{P}}$ then apply an optimal solution S_i' to each induced subgraph $G[V_i]$, we will obviously end up with a union of disjoint cliques. Therefore, these operations make a solution to the original graph G whose weight is

$$wt(E_{\mathcal{P}}) + \sum_{i=1}^{p} wt(S_i') = wt(E_{\mathcal{P}}) + \sum_{i=1}^{p} \text{opt}(G[V_i]).$$

This gives immediately $\text{opt}(G) \leqslant wt(E_{\mathcal{P}}) + \sum_{i=1}^{p} \text{opt}(G[V_i])$. □

Lemma 3.2 directly implies the following corollaries. First, if there is no edge between two different parts in the vertex partition $\mathcal{P}$, then Lemma 3.2 gives

Corollary 3.3. *Let G be a graph with connected components $G_1, \ldots, G_p$, then opt$(G) = \sum_{i=1}^{p}$ opt(G_i), and every optimal solution to the graph G is a union of optimal solutions to the subgraphs $G_1, \ldots, G_p$.*

When $p = 2$, i.e., $\mathcal{P} = \{X, \overline{X}\}$ happens to be a bipartition, the edge set $E_{\mathcal{P}}$ becomes the cut $P_E(X, \overline{X})$, and $wt(P_E(X, \overline{X})) = \gamma(X)$. Lemma 3.2 gives

Corollary 3.4. *Let $X \subseteq V$ be a vertex subset, then*

$$\textit{opt}(G[X]) + \textit{opt}(G[\overline{X}]) \leqslant \textit{opt}(G) \leqslant \textit{opt}(G[X]) + \textit{opt}(G[\overline{X}]) + \gamma(X).$$

Corollary 3.4 enables us to derive a lower bound for the weight of a cut in a graph in terms of an optimal solution to the graph, as shown in the following lemma.

Lemma 3.5. *Let* G *be a graph, and let* S *be an optimal solution to* G*. For any subset* X *of vertices in* G*, if we let* $\mathsf{S}(\mathsf{X},\overline{\mathsf{X}})$ *be the subset of* S *in which each vertex pair contains exactly one vertex in* X*, then* $wt(\mathsf{S}(\mathsf{X},\overline{\mathsf{X}})) \leqslant \gamma(\mathsf{X})$*.*

Proof. The optimal solution S can be divided into three disjoint parts: the subset $\mathsf{S}(\mathsf{X})$ of pairs in which both vertices are in X, the subset $\mathsf{S}(\overline{\mathsf{X}})$ of pairs in which both vertices are in $\overline{\mathsf{X}}$, and the subset $\mathsf{S}(\mathsf{X},\overline{\mathsf{X}})$ of pairs in which exactly one vertex is in X. By Corollary 3.4,

$$\text{opt}(\mathsf{G}) = wt(\mathsf{S}(\mathsf{X}))+wt(\mathsf{S}(\overline{\mathsf{X}}))+wt(\mathsf{S}(\mathsf{X},\overline{\mathsf{X}})) \leqslant \text{opt}(\mathsf{G}[\mathsf{X}])+\text{opt}(\mathsf{G}[\overline{\mathsf{X}}])+\gamma(\mathsf{X}). \quad (3.1)$$

Again, since $\mathsf{S}(\mathsf{X})$ is a solution to the induced subgraph $\mathsf{G}[\mathsf{X}]$ and $\mathsf{S}(\overline{\mathsf{X}})$ is a solution to the induced subgraph $\mathsf{G}[\overline{\mathsf{X}}]$, we have $wt(\mathsf{S}(\mathsf{X})) \geqslant \text{opt}(\mathsf{G}[\mathsf{X}])$ and $wt(\mathsf{S}(\overline{\mathsf{X}})) \geqslant \text{opt}(\mathsf{G}[\overline{\mathsf{X}}])$, which combined with (3.1) gives immediately $wt(\mathsf{S}(\mathsf{X},\overline{\mathsf{X}})) \leqslant \gamma(\mathsf{X})$. □

Similarly we have the following lemmas.

Lemma 3.6. *Let* X *be a subset of vertices in a graph* $\mathsf{G} = (\mathsf{V},\mathsf{E})$*, and let* S *be any optimal solution to* G*. Let* $\mathsf{S}(\mathsf{V},\overline{\mathsf{X}})$ *be the set of pairs in* S *in which at least one vertex is in* $\overline{\mathsf{X}}$*. Then opt*$(\mathsf{G}) \geqslant$ *opt*$(\mathsf{G}[\mathsf{X}]) + wt(\mathsf{S}(\mathsf{V},\overline{\mathsf{X}}))$*.*

Proof. The optimal solution S is divided into two disjoint parts: the subset $\mathsf{S}(\mathsf{X})$ of pairs in which both vertices are in X, and the subset $\mathsf{S}(\mathsf{V},\overline{\mathsf{X}})$ of pairs in which at least one vertex is in $\overline{\mathsf{X}}$. The set $\mathsf{S}(\mathsf{X})$ is a solution to the induced subgraph $\mathsf{G}[\mathsf{X}]$. Therefore, $wt(\mathsf{S}(\mathsf{X})) \geqslant \text{opt}(\mathsf{G}[\mathsf{X}])$. This gives

$$\text{opt}(\mathsf{G}) = wt(\mathsf{S}) = wt(\mathsf{S}(\mathsf{X})) + wt(\mathsf{S}(\mathsf{V},\overline{\mathsf{X}})) \geqslant \text{opt}(\mathsf{G}[\mathsf{X}]) + wt(\mathsf{S}(\mathsf{V},\overline{\mathsf{X}})),$$

which proves the lemma. □

Lemma 3.7. *Let* X *be a subset of vertices in a graph* G*, and let* B_X *be the set of vertices in* X *that are adjacent to vertices in* $\overline{X}$*. For any optimal solution* S *to* G*, if we let* $S(B_X)$ *be the set of pairs in* S *in which both vertices are in* B_X*, then* $opt(G) + wt(S(B_X)) \geqslant opt(G[X]) + opt(G[\overline{X} \cup B_X])$.

Proof. The optimal solution S can be divided into three disjoint parts: the subset $S(X)$ of pairs in which both vertices are in X, the subset $S(\overline{X})$ of pairs in which both vertices are in $\overline{X}$, and the subset $S(X, \overline{X})$ of pairs in which one vertex is in X and the other vertex is in $\overline{X}$. We also denote by $S(B_X, \overline{X})$ the subset of pairs in S in which one vertex is in B_X and the other vertex is in $\overline{X}$. Since $S(X)$ is a solution to the induced subgraph $G[X]$, we have

$$\begin{aligned}
\mathrm{opt}(G) + wt(S(B_X)) &= wt(S(X)) + wt(S(\overline{X})) + wt(S(X, \overline{X})) + wt(S(B_X)) \\
&\geqslant \mathrm{opt}(G[X]) + wt(S(\overline{X})) + wt(S(X, \overline{X})) + wt(S(B_X)) \\
&\geqslant \mathrm{opt}(G[X]) + wt(S(\overline{X})) + wt(S(B_X, \overline{X})) + wt(S(B_X)).
\end{aligned}$$

The last inequality holds true because $B_X \subseteq X$, so $S(B_X, \overline{X}) \subseteq S(X, \overline{X})$. Since $S' = S(\overline{X}) \cup S(B_X, \overline{X}) \cup S(B_X)$ is the subset of pairs in S in which both vertices are in the induced subgraph $G[\overline{X} \cup B_X]$, S' is a solution to the induced subgraph $G[\overline{X} \cup B_X]$. This gives

$$wt(S') = wt(S(\overline{X})) + wt(S(B_X, \overline{X})) + wt(S(B_X)) \geqslant \mathrm{opt}(G[\overline{X} \cup B_X]),$$

which implies the lemma immediately. $\square$

The above results that reveal the relations between the structures of the correlation clustering problem and graph edge-cuts not only form the basis for our kernelization results presented in this dissertation, but also are of their own importance and interests.

3.2 The Reduction Steps

Obviously, the number of distinct vertices included in the vertex pairs in a solution S of k pairs is upper bounded by $2k$. Thus, if we can also bound the number of vertices that are not included in S, we get a kernel. For such a vertex v, the clique containing v in $G \triangle S$ must be $G[N[v]]$. Inspired by this observation, our approach is to check the neighborhood $N[v]$ for each vertex v in the input graph G.

Intuitively, if an induced subgraph is very "dense inherently", while is also "loosely connected to outside" (*i.e.* there are relatively fewer edges in the cut of this subgraph), then the subgraph might be cut off and solved separately. By the cutting lemmas, the cost of a solution obtained as such should not be too far away from that of an optimal solution. Actually, we will figure out the conditions under which they are equal.

The subgraph we are considering is $G[N[v]]$ for some vertex v. In terms of the density, a simple fact is that the fewer edges are missing from a subgraph, the denser it is. Therefore, to measure the density of $G[N[v]]$, we introduce the *deficiency* $\delta(v)$ of $N[v]$ as the total weight of anti-edges in $G[N[v]]$, which is formally defined as $\delta(v) = wt(\{\{x, y\} \mid x, y \in N(v), [x, y] \notin E\})$. For the connection of $N[v]$ to outside, the most natural measurement should be the weight $\gamma(N[v])$ of the cut $P_E(N[v], \overline{N[v]})$.

Suppose that $N[v]$ exclusively forms a clique, i.e. v is stable. Then anti-edges of total weight $\delta(v)$ have to be inserted to make $N[v]$ a clique, and edges of total weight $\gamma(N[v])$ have to be deleted to make $N[v]$ disjoint. Note that each inserted edge involves two distinct vertices in $N[v]$, while each deleted edge touches only one. Based on this observation, we introduce the following important definition.

Definition The *stable cost* of a vertex v is defined as $\rho(v) = 2\delta(v) + \gamma(N[v])$. The neighborhood $N[v]$ is *reducible* if $\rho(v) < |N[v]|$.

We now describe three reduction rules on the neighborhood $N[v]$ of a vertex v such that $N[v]$ is reducible. In fact, the reducibility of $N[v]$ is the only reduction condition we need on which the three reduction rules are applied in order.

Lemma 3.8. *For any vertex v such that $N[v]$ is reducible, there is an optimal solution S^* to G such that the vertex set $N[v]$ is entirely contained in a single clique in the graph $G \triangle S^*$.*

Proof. Let S be an optimal solution to the graph G, and here we only consider the case where $N[v]$ is not entirely contained in a single clique in $G \triangle S$. Take any clique C intersecting $N[v]$, let $X = C \cap N[v]$ and $Y = N[v] - X$. Then $X \neq \emptyset$ and $Y \neq \emptyset$. Note that we do not assume that Y is in a single clique in $G \triangle S$.

Inserting all missing edges in $G[N[v]]$ will transform it into a clique. Therefore, opt$(G[N[v]]) \leqslant \delta(v)$. Combining this with Corollary 3.4, we get

$$\begin{aligned} \text{opt}(G) &\leqslant \text{opt}(G[N[v]]) + \text{opt}(G[\overline{N[v]}]) + \gamma(N[v]) \\ &\leqslant \delta(v) + \text{opt}(G[\overline{N[v]}]) + \gamma(N[v]) \\ &= \text{opt}(G[\overline{N[v]}]) + \rho(v) - \delta(v). \end{aligned} \tag{3.2}$$

Obviously, $wt(S(V, N[v])) \geqslant wt(S(X, Y))$ because $X \subseteq V$ and $Y \subseteq N[v]$. Since the solution S places the sets X and Y in different cliques, all edges between X and

Y must be deleted, which means $\mathsf{S}(\mathsf{X},\mathsf{Y}) = \mathsf{P}(\mathsf{X},\mathsf{Y})$. Finally, by the definition of $\delta(\nu)$ and fact that both X and Y are subsets of $\mathsf{N}[\nu]$, the weight sum of all anti-edges between X and Y is at most $\delta(\nu)$. Thus, we have $wt(\mathsf{S}(\mathsf{X},\mathsf{Y})) + \delta(\nu) \geqslant wt(\mathsf{P}(\mathsf{X},\mathsf{Y}))$. Now by Lemma 3.6,

$$\begin{aligned} \mathrm{opt}(\mathsf{G}) &\geqslant \mathrm{opt}(\mathsf{G}[\overline{\mathsf{N}[\nu]}]) + wt(\mathsf{S}(\mathsf{V},\mathsf{N}[\nu])) \\ &\geqslant \mathrm{opt}(\mathsf{G}[\overline{\mathsf{N}[\nu]}]) + wt(\mathsf{S}(\mathsf{X},\mathsf{Y})) \qquad (3.3) \\ &\geqslant \mathrm{opt}(\mathsf{G}[\overline{\mathsf{N}[\nu]}]) + wt(\mathsf{P}(\mathsf{X},\mathsf{Y})) - \delta(\nu). \end{aligned}$$

Combining (3.2) and (3.3), and noting that $\mathsf{N}[\nu]$ is reducible and that the weight of each vertex pair is at least 1, we get

$$|\mathsf{X}| \cdot |\mathsf{Y}| \leqslant wt(\mathsf{P}(\mathsf{X},\mathsf{Y})) \leqslant \rho(\nu) < |\mathsf{N}[\nu]| = |\mathsf{X}| + |\mathsf{Y}|. \qquad (3.4)$$

This can hold true only when $|\mathsf{X}| = 1$ or $|\mathsf{Y}| = 1$. In both cases, we have $|\mathsf{X}| \cdot |\mathsf{Y}| = |\mathsf{X}| + |\mathsf{Y}| - 1$. Combining this with (3.4), and noting that all the quantities are integers, we must have

$$wt(\mathsf{P}(\mathsf{X},\mathsf{Y})) = \rho(\nu),$$

which, when combined with (3.2) and (3.3), gives

$$\mathrm{opt}(\mathsf{G}) = \mathrm{opt}(\mathsf{G}[\overline{\mathsf{N}[\nu]}]) + \rho(\nu) - \delta(\nu) = \mathrm{opt}(\mathsf{G}[\overline{\mathsf{N}[\nu]}]) + \gamma(\mathsf{N}[\nu]) + \delta(\nu). \qquad (3.5)$$

Note that $\gamma(\mathsf{N}[\nu]) + \delta(\nu)$ is the minimum cost to insert edges into and delete edges from the graph G to make $\mathsf{N}[\nu]$ a disjoint clique. Therefore, Equality (3.5) shows that if we first apply edge insert/delete operations of minimum weight to make $\mathsf{N}[\nu]$ a disjoint clique, then apply an optimal solution to the induced subgraph $\mathsf{G}[\overline{\mathsf{N}[\nu]}]$, then we have an optimal solution S^* to the graph G. This completes the proof of the lemma because the optimal solution S^* has the vertex set $N[v]$ entirely contained in a single clique in the graph $G \triangle S^*$. □

By Lemma 3.8, the optimal solution S^* inserts a collection E_0 of edges of total weight $\delta(v)$ for anti-edges in the subgraph $G[N[v]]$ to make it a clique. Therefore, the remaining vertex pairs in $S^* - E_0$ make an optimal solution to the resulting graph $G \triangle E_0$. This gives the rule for our first reduction step:

Rule 3.1. *For a vertex v such that $N[v]$ is reducible, insert an edge for each anti-edge in $G[N[v]]$ to make $G[N[v]]$ a clique, and decrease the parameter k by $\delta(v)$.*

Remark. If $N[v]$ is reducible, then after applying Rule 1, $N[v]$ remains reducible.

After Rule 3.1 inserted edges to make the induced subgraph $G[N[v]]$ a clique, we use the following rule to remove the vertices in $N(N[v])$ that are loosely connected to $N[v]$.

Rule 3.2. *Let v be a vertex such that $N[v]$ is reducible on which Rule 1 has been applied. For each vertex x in $N(N[v])$, if $wt(P_E(x, N[v])) \leqslant wt(P_A(x, N[v]))$, then delete all edges in $P_E(x, N[v])$ and decrease the parameter k by $wt(P_E(x, N[v]))$.*

Remark. Similar to Rule 1, it is easy to verify that if $N[v]$ is reducible, then after applying Rule 2, $N[v]$ remains reducible.

Lemma 3.9. *Rule 3.2 is safe.*

Proof. By Lemma 3.8, there is an optimal solution S to the graph G such that $N[v]$ is entirely contained in a single clique C in the graph $G \triangle S$. We first prove, by contradiction, that the clique C containing $N[v]$ in the graph $G \triangle S$ contains at most one vertex not in $N[v]$. Suppose, on the contrary, that there are r ($\geqslant 2$) vertices

$u_1, \ldots, u_r$ not in $N[v]$ that are in C. For $1 \leqslant i \leqslant r$, let $c_i = wt(P_E(u_i, N[v]))$ and $c'_i = wt(P(u_i, N[v]))$. Note that $c'_i \geqslant |N[v]|$ and $\sum_{i=1}^{r} c_i \leqslant \gamma(N[v])$. Then in the optimal solution S to G, the total weight of the edges inserted between $N[v]$ and $\overline{N[v]}$ is at least

$$\begin{aligned}
\sum_{i=1}^{r}(c'_i - c_i) &\geqslant \sum_{i=1}^{r}(|N[v]| - c_i) \\
&= r|N[v]| - \sum_{i=1}^{r} c_i \\
&\geqslant r|N[v]| - \gamma(N[v]) \\
&\geqslant 2|N[v]| - \gamma(N[v]) \\
&> 2|N[v]| - |N[v]| \\
&= |N[v]| \\
&> \gamma(N[v]).
\end{aligned}$$

Herein, we have used the fact $|N[v]| > \gamma(N[v])$ (this is because by the conditions of the rule, $\rho(v) = 2\delta(v) + \gamma(N[v]) < |N[v]|$). But this contradicts Lemma 3.5.

Therefore, for the optimal solution S, there is at most one vertex x that is not in $N[v]$ but in the clique C containing $N[v]$ in the graph $G \triangle S$. We can assume that the vertex x satisfies the condition $wt(P_E(x, N[v])) > wt(P_A(x, N[v]))$: otherwise, instead of inserting edges for all anti-edges in $P_A(x, N[v])$ to make $N[v] \cup \{x\}$ a clique, we delete all edges in $P_E(x, N[v])$ and will get another optimal solution S' that makes the subgraph $G[N[v]]$ a separated clique in the objective graph $G \triangle S'$. In consequence, for a vertex x in $N(N[v])$ with $wt(P_E(x, N[v])) \leqslant wt(P_A(x, N[v]))$, we can always assume that x is not in the clique containing $N[v]$ in the graph $G \triangle S$ for the optimal solution S. In particular, deleting all edges in $P_E(x, N[v])$ for such a vertex x is always safe. $\square$

After Rules 1-2 are applied, the subgraph $N[v]$ has a very simple structure, which is characterized by the following lemma:

Lemma 3.10. *Let* v *be a vertex such that* $N[v]$ *is reducible on which Rules 1-2 have been applied. Then there is at most one vertex* x *that was originally in* $N(N[v])$ *and now is still adjacent to* $N[v]$*.*

Proof. By the condition in Rule 2, any vertex x in $N(N[v])$ that is still adjacent to $N[v]$ after the application of Rule 2 must satisfy $wt(P_E(x, N[v])) > wt(P_A(x, N[v]))$. Since $wt(v, w) \geqslant 1$ for all vertex pairs $\{v, w\}$,

$$|N[v]| \leqslant wt(P(x, N[v])) = wt(P_E(x, N[v])) + wt(P_A(x, N[v])).$$

Thus, the vertex x satisfies the condition $wt(P_E(x, N[v])) > |N[v]|/2$.

To prove the lemma, suppose on the contrary that there are two vertices x_1 and x_2 that were originally in $N(N[v])$ and are still adjacent to $N[v]$ after the application of Rules 1-2. By the above discussion, we have $wt(P_E(x_1, N[v])) > |N[v]|/2$ and $wt(P_E(x_2, N[v])) > |N[v]|/2$. This gives

$$\gamma(N[v]) \geqslant wt(P_E(x_1, N[v])) + wt(P_E(x_2, N[v])) > |N[v]|.$$

But this contradicts the assumption that $N[v]$ is reducible, that is, $\rho(v) = 2\delta(v) + \gamma(N[v]) < |N[v]|$. $\square$

By Lemma 3.10, after Rules 1-2 are applied, the structure of the subgraph $G[N[v]]$ must be in one of the following two cases: (1) no vertex in $N(N[v])$ is adjacent to $N[v]$. In this case, $G[N[v]]$ is a separated clique – by Corollary 3.3, we can simply remove the clique and work on the rest of the graph; and (2) there is one vertex x in $N(N[v])$ that is still adjacent to $N[v]$ – this case will be handled by the following reduction step (note that Rules 1-2 do not change the values of $wt(P_E(x, N[v]))$ and $wt(P_A(x, N[v]))$).

Rule 3.3. *Let v be a vertex such that $N[v]$ is reducible on which Rules 1-2 have been applied. If there is a vertex x in $N(N[v])$ that is still adjacent to $N[v]$, then contract $N[v]$ into a single vertex v', add an edge $[v', x]$ of weight $wt(P_E(x, N[v])) - wt(P_A(x, N[v]))$, set the weight of each anti-edge $\{v', u\}$, where $u \neq x$, to $+\infty$, and decrease the parameter k by $wt(P_A(x, N[v]))$.*

Lemma 3.11. *Rule 3.3 is safe.*

Proof. Let G' be the graph obtained by applying Rule 3.3 on a graph G. First note that because of Rule 2, we must have $wt(P_E(x, N[v])) > wt(P_A(x, N[v]))$. Therefore, the new edge $[v', x]$ in the graph G' has a valid weight $wt(P_E(x, N[v])) - wt(P_A(x, N[v])) \geqslant 1$.

From the proofs of Lemmas 3.9-3.10, we can assume that there is an optimal solution S to the graph G such that either $N[v]$ or $N[v] \cup \{x\}$ is a separated clique in the graph $G \triangle S$. To prove the safeness of Rule 3.3, we only need to verify that the optimal solution S to the graph G has a weight bounded by k if and only if the reduced graph G' has a solution of weight bounded by $k - wt(P_A(x, N[v]))$.

If $N[v] \cup \{x\}$ becomes a separated clique in $G \triangle S$, then the solution S must consist of a set I_x of edge insertions of total weight $wt(P_A(x, N[v]))$ to the anti-edges between x and $N[v]$, plus the set $S \setminus I_x$ of other vertex pairs. Because the induced subgraph $G[N[v]]$ has already become a clique before Rule 3 is applied, no vertex pair in the set $S \setminus I_x$ contains vertices in $N[v]$. Thus, each vertex pair in $S \setminus I_x$ contains at most one vertex (i.e., the vertex x) in $N[v] \cup \{x\}$. This implies that the set $S \setminus I_x$ will make the graph $G' \triangle (S \setminus I_x)$ a union of disjoint cliques, which is equal to the graph $G \triangle S$ minus the clique made by $N[v] \cup \{x\}$ and plus the edge $[v', x]$. Therefore, if the optimal solution S to the graph G has a weight $wt(S)$ bounded by k, then the graph G' has a solution $S \setminus I_x$ of weight bounded by $wt(S \setminus I_x) = wt(S) - wt(I_x) \leqslant k - wt(P_A(x, N[v]))$.

Similarly, if $N[v]$ becomes a separated clique in $G \triangle S$, then the solution S must consist of a set D_x of edge deletions of total weight $wt(P_E(x, N[v]))$ to separate x and $N[v]$, plus the set $S \setminus D_x$ of other vertex pairs of total weight $wt(S) - wt(P_E(x, N[v]))$. The collection $S \setminus D_x$ plus deleting the edge $[v', x]$ is a solution S' to the graph G' such that $G' \triangle S'$ is the graph $G \triangle S$ minus the clique made by $N[v]$ and plus the isolated vertex v'. The weight of the collection S' is equal to

$$\begin{aligned} & wt(S \setminus D_x) + wt(v', x) \\ = \quad & [wt(S) - wt(P_E(x, N[v]))] + [wt(P_E(x, N[v])) - wt(P_A(x, N[v]))] \\ = \quad & wt(S) - wt(P_A(x, N[v])). \end{aligned}$$

This again proves that if the optimal solution S to the graph G has a weight $wt(S)$ bounded by k, then the graph G' has a solution S' of weight bounded by $k - wt(P_A(x, N[v]))$.

For the proof for the other direction, suppose that the graph G' has a solution S' of weight bounded by $k - wt(P_A(x, N[v]))$. Since $wt(v', w) = +\infty$ for all vertices $w \neq x$, the graph $G' \triangle S'$ must either have the single vertex v' as a separated clique or have the edge $[v', x]$ as a separated clique. Now the rest of the proof for this direction proceeds in a way similar to that for the other direction. If $G' \triangle S'$ has the vertex v' as a separated clique, then S' minus the edge deletion $[v', x]$ and plus the edge deletions for the edges between x and $N[v]$ makes a solution to the graph G, whose weight is bounded by

$$[k - wt(P_A(x, N[v])] - [wt(P_E(x, N[v])) - wt(P_A(x, N[v]))] + wt(P_E(x, N[v])) = k.$$

On the other hand, if $G' \triangle S'$ has the edge $[v', x]$ as a separated clique, then S' plus the edge insertions for the anti-edges between x and $N[v]$ makes a solution to the graph G, whose weight is bounded by

$$[k - wt(P_A(x, N[v])] + wt(P_A(x, N[v])) = k.$$

This completes the proof for the direction that if the graph G' has a solution of weight bounded by $k - wt(P_A(x, N[v]))$, then the graph G has a solution of weight bounded by k.

In summary, Rule 3 is safe and the lemma is proved. □

3.3 The Kernelization Algorithm

Now we are ready to describe our kernelization algorithm, which is simply an application of the three reduction rules Rule 1, Rule 2, and Rule 3 in order.

The Kernelization Algorithm. For each vertex v such that $N[v]$ is reducible

1. insert edges to make $G[N[v]]$ a clique and decrease k by $\delta(v)$;

2. for each vertex u in $N(N[v])$ with $wt(P_E(u, N[v])) \leqslant wt(P_A(u, N[v]))$, delete all edges in $P_E(u, N[v])$ and decrease k by $wt(P_E(u, N[v]))$;

3. if $N[v]$ becomes a separated clique, then remove the clique; otherwise let x be the unique vertex in $N(N[v])$ that is still adjacent to $N[v]$, contract

$N[v]$ into a single vertex v', make an edge $[v', x]$ of weight $wt(P_E(x, N[v])) - wt(P_A(x, N[v]))$, set the weight of each anti-edge $\{v', u\}$, where $u \neq x$, to $+\infty$, and decrease k by $wt(P_A(x, N[v]))$.

Note the reduction condition in the Kernelization Algorithm does not depend on the parameter k. The analysis of time complexity is omitted here, and it can be easily verified this algorithm has its running time bounded by $\mathcal{O}(n^3)$. The following theorem implies our main Theorem 3.1 directly.

Theorem 3.12. *Let* (G, wt, k) *be an instance of the weighted correlation clustering problem such that no vertex* v *in* G *has a reducible neighborhood* $N[v]$*. If the graph* G *has more than* $2k$ *vertices, then no solution to the graph* G *has its weight bounded by* k*.*

Proof. Let S be an optimal solution to the graph $G = (V, E)$. For each vertex pair $\{v, w\}$ in S, we divide the cost $wt(v, w)$ into two halves and distribute them evenly to the two vertices v and w. By this procedure, each vertex v gets a "cost" $\text{cost}(v) = \frac{1}{2}\sum_{\{v,w\}\in S} wt(v, w)$. Obviously, the total weight of S is equal to $\sum_{v\in V} \text{cost}(v)$.

For a vertex v not contained in any pair in the solution S, the neighborhood $N[v]$ becomes a separated clique in the graph $G \triangle S$. Note that for any two vertices v and w of distance 2 (i.e., the vertices v and w are not adjacent but have a common neighbor), at most one of v and w is not contained in any pair in the solution S: otherwise v and w would have to belong to the same clique in $G \triangle S$ because of their common neighbor but the edge $[v, w]$ would be missing.

Let $Z_S = \{v_1, v_2, \ldots, v_r\}$ be the set of vertices in the graph G that are not contained in any pair in the solution S. Then, for any two vertices v_i and v_j in Z_S, either v_i and v_j are adjacent or the distance between v_i and v_j is larger than 2. If the distance between v_i and v_j is larger than 2, then the two neighborhoods

$N[v_i]$ and $N[v_j]$ have no common vertex. If the vertices v_i and v_j are adjacent, then $v_i \in N[v_j]$ and $v_j \in N[v_i]$, and both $N[v_i]$ and $N[v_j]$ will become separated cliques in the graph $G \triangle S$. Therefore, in this case, we must have $N[v_i] = N[v_j]$. As a result, any two of the neighborhoods in the collection $\{N[v_1], N[v_2], \ldots, N[v_r]\}$ are either the same or mutually disjoint. Thus, without loss of generality, we can assume that all neighborhoods in $\{N[v_1], N[v_2], \ldots, N[v_r]\}$ are pairwise disjoint (otherwise, we can simply remove duplicated copies of the neighborhoods in the collection). Let $N_S = N[v_1] \cup N[v_2] \cup \cdots \cup N[v_r]$.

The total cost in the solution S to make a neighborhood $N[v_i]$ a separated clique is $\delta(v_i) + \gamma(N[v_i])$, where the cost $\delta(v_i)$ is on the anti-edges in $G[N[v_i]]$ that have both ends in $N[v_i]$, and the cost $\gamma(N[v_i])$ is on the edges in $P_E(N[v_i], \overline{N[v_i]})$ that have one end in $N[v_i]$ and the other end not in $N[v_i]$. Therefore, if we count the cost assigned to the vertices in $N[v_i]$, then the total cost of the vertices in $N[v_i]$ is $\delta(v_i) + \gamma(N[v_i])/2 = \rho(v_i)/2$, which is at least $|N[v_i]|/2$ because the neighborhood $N[v_i]$ is not reducible. From this analysis, we get

$$\sum_{v \in N_S} \mathsf{cost}(v) = \sum_{i=1}^{r} \sum_{v \in N[v_i]} \mathsf{cost}(v) \geqslant \sum_{i=1}^{r} |N[v_i]|/2 = |N_S|/2. \tag{3.6}$$

On the other hand, each w of the vertices not in the set N_S is contained in at least one pair in the solution S and therefore bears cost at least $1/2$. This gives

$$\sum_{v \in V - N_S} \mathsf{cost}(v) \geqslant |V - N_S|/2. \tag{3.7}$$

Combining (3.6) and (3.7), we conclude that the total cost of the optimal solution S is

$$\sum_{v\in V}\mathsf{cost}(v) = \sum_{v\in N_S}\mathsf{cost}(v) + \sum_{v\in V-N_S}\mathsf{cost}(v) \geqslant |N_S|/2 + |V-N_S|/2 = |V|/2.$$

Therefore, if $|V| > 2k$, then the graph G has no solution of weight bounded by k. □

3.4 On Unweighted and Real-Weighted Versions

We now show how to adapt the Kernelization Algorithm in the previous section to handle the unweighted and real-weighted versions of the correlation clustering problem. Only relatively minor modifications are needed, and we will be focused on the discussions of these modifications.

The unweighted version. The unweighted version of the correlation clustering problem is equivalent to the weighed version when we assume $wt(v, w) \equiv 1$ for all pairs $\{v, w\}$ of vertices in the graph G. Since the weight function wt also takes positive integral values, most results for the weighted version also hold true for the unweighted version. In particular, the results in Section 2 and Lemmas 3.8-3.10 all remain valid. However, Rule 3.3 is no longer valid, which may introduce an edge $[v', x]$ of weight $wt(P_E(x, N[v])) - wt(P_A(x, N[v]))$ that is larger than 1. Thus, Rule 3 may transform an instance for the unweighted version into an instance that is not valid for the unweighted version.

This can be easily circumvented, by replacing Rule 3.3 by the following new rule:

Rule 3.3 (U). *Let v be a vertex such that $N[v]$ is reducible on which Rules 1-2 have been applied. If a vertex x in $N(N[v])$ is still adjacent to $N[v]$, then let e =*

$|P_E(x, N[v])|$ *and* $a = |P_A(x, N[v])|$*, replace* $N[v]$ *by a complete graph* K_{e-a} *in which all vertices are adjacent to* x*, and decrease* k *by* a*.*

The correctness of Rule 3 (U) can be proved by an argument similar to that in the proof of Lemma 3.11. In particular, it can be proved that the original graph has a solution that consists of no more than k edge operations if and only if the reduced graph has a solution that consists of no more than $k - a$ edge operations. Therefore, Rule 3 (U) is safe for the unweighted version of the CORRELATION CLUSTERING problem. Moreover, with the new rule, the proof of Theorem 3.12 can be applied without any change to the unweighted version. Therefore, the Kernelization Algorithm presented in Section 4, with Step 3 replaced by Rule 3 (U), constructs a kernel of at most $2k$ vertices for the unweighted version of the correlation clustering problem.

The real-weighted version. Further care is required when we extend our algorithm to the real-weighted version of the correlation clustering problem. The first problem is that, with non-integral values, the relations given in (3.4) no longer imply the equalities in (3.5). In particular, $\rho(v) < |N[v]|$ no longer implies $\rho(v) \leqslant |N[v]| - 1$. This can be fixed by changing the definition of the reducible neighborhood $N[v]$ as follows.

Definition [Reducible Neighborhood for the Real-Weighted Version] The neighborhood $N[v]$ of a vertex v is *reducible* if $\rho(v) \leqslant |N[v]| - 1$.

Then the relations in (3.4) become

$$|X| \cdot |Y| \leqslant wt(P(X, Y)) \leqslant \rho(v) \leqslant |N[v]| - 1 = |X| + |Y| - 1. \tag{3.8}$$

This again gives $|X| \cdot |Y| = |X| + |Y| - 1$ (note that $|X|$ and $|Y|$ are positive integers), so $wt(P(X,Y)) = \rho(v)$ and the equalities in (3.5) become true. As a consequence, for the real-weighted version of the problem under the new definition of reducible neighborhood, Lemma 3.8 is valid, and Rule 1 is safe .

Rule 2, Lemma 3.9, and Lemma 3.10 remain valid for the real-weighted version.

Now consider Rule 3. By the conditions of Rule 3, we have $wt(P_E(x, N[v])) > wt(P_A(x, N[v]))$. However, the value $wt(P_E(x, N[v])) - wt(P_A(x, N[v]))$ can be smaller than 1 for the real-weighted version thus it may not be a valid weight value to be assigned to the edge $[v'x]$ (recall that we require all weights be at least 1 for the weight function wt). This can be fixed by the following modification:

Rule 3.3 (R)**.** *Let v be a vertex such that $N[v]$ is reducible on which Rules 1-2 have been applied. If a vertex x in $N(N[v])$ is still adjacent to $N[v]$, then let $w_e = wt(P_E(x, N[v]))$, $w_a = wt(P_A(x, N[v]))$, and*

- *if $w_e - w_a \geqslant 1$, then contract $N[v]$ into a single vertex v', add an edge $[v', x]$ of weight $w_e - w_a$, set the weight of each anti-edge $\{v', u\}$, where $u \neq x$, to $+\infty$, and decrease k by w_a;*
- *if $w_e - w_a < 1$, then replace $N[v]$ by two vertices v' and v'', add an edge $[v', x]$ of weight 2, and an edge $[v', v'']$ of weight $2 - (w_e - w_a)$, set the weight of each of the anti-edges $\{v', u\}$, where $u \neq x, v''$, and $\{v'', w\}$, where $w \neq v'$, to $+\infty$, and decrease k by $w_e - 2$.*

Note that we must have $w_e \geqslant 2$: otherwise, from $w_a < w_e < 2$, and by the requirement that the weight of each vertex pair be at least 1, the neighborhood $N[v]$ must consist of exactly two vertices v and w such that $wt(x, w) = w_e$ and $wt(x, v) = w_a$. This would imply $\delta(v) = 0$ and $\gamma(N[v]) = w_e$. As a consequence, $\rho(v) = w_e > 1 = |N[v]| - 1$, which would contradict the assumption that $N[v]$

is reducible (recall that applying Rules 1-2 to a reducible $N[v]$ cannot make $N[v]$ non-reducible). Therefore, in any case, Rule 3 (R) does not increase the value of the parameter k. Moreover, it can be easily verified that the new weight function has value at least 1 on each vertex pair in the reduced graph. Therefore, on an instance (G, wt, k) of the real-weighted version, Rule 3 (R) produces a valid instance (G', wt', k') for the real-weighted version with $k' \leqslant k$.

To verify that Rule 3 (R) is safe for the real-weighted version, we need to prove that the graph G has a solution of weight bounded by k if and only if the graph G' has a solution of weight bounded by k'. The proof is identical to that of Lemma 3.11 when $w_e - w_a \geqslant 1$. For the case $w_e - w_a < 1$, note that because of the way we assigned weights to the new vertex pairs in the reduced graph G', in an optimal solution to the graph G', either the edge $[x, v']$ or the edge $[v', v'']$ must make a separated clique. Now following a similar idea as that given in the proof of Lemma 3.11, we can prove that the original graph G has a solution of weight bounded by k if and only if the reduced graph G' has a solution of weight bounded by $k-(w_e-2)$. More specifically, we can verify that (1) the original graph G has an optimal solution of weight bounded by k in which $N[v] \cup \{x\}$ makes a separated clique if and only if the new graph G' has an optimal solution of weight bounded by $k - (w_e - 2)$ in which the edge $[x, v']$ makes a separated clique, and (2) the original graph G has an optimal solution of weight bounded by k in which $N[v]$ makes a separated clique if and only if the new graph G' has an optimal solution of weight bounded by $k - (w_e - 2)$ in which the edge $[v', v'']$ makes a separated clique.

Finally, a result for the real-weighted version similar to Theorem 3.12 for the integral-weighted version can be proved similarly. For the completeness, we present the detailed proof for this result as follows.

Theorem 3.13. *Let* $(\mathsf{G}, \mathsf{wt}, \mathsf{k})$ *be an instance for the real-weighted version of the correlation clustering problem such that no vertex* ν *in* G *has a reducible neighborhood* $\mathsf{N}[\nu]$. *If the graph* G *has at least* $4\mathsf{k}$ *vertices, then* G *has no solution of weight bounded by* k.

Proof. Let S be an optimal solution to the graph G, and let $Z_S = \{\nu_1, \nu_2, \ldots, \nu_r\}$ be the set of all vertices in G that do not appear in any pair in S. As we proved in Theorem 3.12, for two vertices ν_i and ν_j in Z_S, either $N[\nu_i] \cap N[\nu_j] = \emptyset$, or $N[\nu_i] = N[\nu_j]$. Therefore, without loss of generality, we can assume that $N[\nu_i] \cap N[\nu_j] = \emptyset$ for all $i \neq j$. The set $N[\nu_1] \cup N[\nu_2] \cup \cdots \cup N[\nu_r]$ contains all vertices that do not appear in any pair in the solution S (plus perhaps some vertices that appear in pairs in S).

Again, we divide the weight of each vertex pair $\{\nu, w\}$ in the solution S into two halves and distribute them equally to the vertices ν and w, and then count the costs on all vertices. For each vertex ν_i in Z_S, since $N[\nu_i]$ will make a separated clique in the graph $G \triangle S$, the sum of the costs on the vertices in $N[\nu_i]$ is equal to $\delta(\nu) + \gamma(N[\nu_i])/2 = \rho(\nu_i)/2$. Since the neighborhood $N[\nu_i]$ is not reducible, we have $\rho(\nu_i) > |N[\nu_i]| - 1$. Therefore, the sum of the costs on the vertices in $N[\nu_i]$ is larger than $(|N[\nu_i]| - 1)/2$. In consequence, the average cost on each vertex in $N[\nu_i]$ is larger than $(|N[\nu_i]| - 1)/(2|N[\nu_i]|)$. Since $|N[\nu_i]| \geqslant 2$ (otherwise ν_i would be an isolated vertex and $N[\nu_i]$ would be reducible), we conclude that the average cost on each vertex in $N[\nu_i]$ is larger than $1/4$. Extending this to all neighborhoods, we conclude that the average cost on each vertex in the set $N[\nu_1] \cup N[\nu_2] \cup \cdots \cup N[\nu_r]$ is larger than $1/4$.

For each vertex w not in the set $N[\nu_1] \cup N[\nu_2] \cup \cdots \cup N[\nu_r]$, w appears in at least one pair in the solution S. Therefore, the cost on the vertex w is at least $1/2$. Combining this fact with the above analysis, we derive immediately that the weight of the solution S is larger than $n/4$, where n is the number of vertices in the graph G. Therefore, if $n \geqslant 4k$, then the graph G has no solution of weight bounded by k. □

Corollary 3.14. *There is a polynomial-time kernelization algorithm for the real-weighted version of the correlation clustering problem that produces a kernel that contains at most* $4k - 1$ *vertices.*

3.5 Discussion

An interesting observation is that for the unweighted version, by the definition of simple series modules [63], all of the following are exactly the same:

$$N[u] = N[M], \quad \delta(u) = \delta(M), \quad \text{and} \quad \gamma(N[u]) = \gamma(N[M]),$$

where M is the simple series module containing the vertex u, and $\delta(M)$ is a natural generalization of the definition $\delta(v)$. Therefore, it does not matter if we use the module or any vertex in the module: every vertex in a simple series module is a full representative for the module. This observation shows that previous kernelization algorithms can be significantly simplified by avoiding modular decompositions. More importantly, this enables our approach to handle the weighted versions.

The kernel size analysis is tight, and there are graphs with $2k$ vertices whose optimal solutions have size exactly k. For any integer $k > 1$, we can take a cycle of $2k$ vertices, where each of the $2k$ edges has unit weight, and it is easy to verify the only optimal solution is to remove half of the edges.

As a final remark, I also would like to compare this result with the 2-approximation algorithm for vertex cover problem. Albeit there had been long known how to obtain such an algorithm for the unweighted version (greedy, or from a maximum matching, say), they enjoyed only very limited applications. Situations only changed after Hochbaum developed the first algorithm for the weighted version based on linear program [127] (See also [128]). Contrary to previous specialized techniques, Hochbaum's algorithm not only works for both unweighted and weighted versions of vertex cover, but is later shown to be easily generalized to variations of vertex cover, including partial vertex cover, capacitated vertex cover, and other generalizations, and consequently brings numerous results [127, 129, 130].

4. HIERARCHICAL CLUSTERING

It is natural to try to apply techniques presented in Chapter 3 to other clustering problems. This chapter is devoted to the famous hierarchical clustering problem. Following the same basic reasoning, but with more complicated reduction rules, I obtain a $4k$ element kernel.

Theorem 4.1. *There is an* $\mathcal{O}(M \cdot n^3)$ *time kernelization algorithm for the hierarchical clustering problem that produces a kernel with at most* $4k$ *elements.*

The second result here is a direct generalization of the currently best parameterized algorithm for correlation clustering to hierarchical clustering with the same time complexity up to a polynomial factor. These two results together tell us from the aspect of parameterized (exact) computation., the hierarchical clustering is not necessarily harder than correlation clustering.

Theorem 4.2. *There is an* $\mathcal{O}^*(1.62^k)$ *time parameterized algorithm for hierarchical clustering problem, which either returns a solution with cost at most* k*, or correctly reports no such solution exists.*

These techniques are also general enough to be applied for other clustering problems, e.g. the clustering aggregation problem, the details are omitted here.

4.1 Preliminaries

A nonnegative square matrix D of order n is called *ultrametric* if for any triplet $1 \leqslant i, j, k \leqslant n$, of the three pairwise distances, either all are the same, or two are the same while the other is strictly smaller. This can be concisely characterized by $D_{ij} \leqslant \max(D_{ik}, D_{jk})$. Immediately following from the definition are two observations:

$$D_{ik} = \max(D_{ij}, D_{jk}) \quad \text{if } D_{ij} \neq D_{jk}, \tag{4.1}$$

$$\text{and} \qquad D_{ik} \leqslant D_{ij} \quad \text{if } D_{ij} = D_{jk}. \tag{4.2}$$

It is also common to characterize ultrametric matrices as *conflict triples* free, here by a conflict triple I mean three elements $1 \leqslant i < j < k \leqslant n$ which satisfies $D_{ij} \leqslant D_{ik} < D_{jk}$. Let S be an $n \times n$ matrix whose elements are integers from $[-M, M]$, and $D + S$ is the normal addition of matrices.[1] S is a *solution* to matrix D if $D + S$ is ultrametric, and by definition, the cost of S is $\texttt{cost}(S) = \sum_{1 \leqslant i < j \leqslant n} |S_{ij}|$. For a matrix D, denote by $\texttt{opt}(D)$ the cost of an optimal solution to D, i.e. the minimum cost of a solution over all solutions to D.

Denote by $[n]$ the set of positive integers $\{1, 2, \ldots, n\}$. Given any pair of index subsets $I, J \subseteq [n]$, $D|_{I,J}$ is the $|I| \times |J|$ submatrix of D determined by the row index set I and column index set J, and particularly, I write $D|_I$ as a shorthand for $D|_{I,I}$ whose rows and columns are both indexed by I. By definition of ultrametric, in the objective matrix $D' = D + S$, the submatrix $D'|_I$ for any subset $I \subseteq [n]$ is also ultrametric. In terms of this hereditary property, the submatrix $D|_I$ for each index set I can be viewed as an instance of hierarchical clustering (on which $\texttt{opt}(D|_I)$ is defined naturally). Moreover, the submatrix $S|_I$ of a solution matrix S to D is also a solution to $D|_I$, nevertheless the optimality does not transfer in general.

As previously mentioned, the hierarchical clustering problem degenerates to the correlation clustering problem when $M = 1$. It may help to understand the relation

[1]In this dissertation I directly use the distance matrix as the base of our operation, instead the $n(n-1)/2$ vector by previous authors, because this will make our description easier. Observe that when counting the cost, only the upper-triangle of the (symmetric) solution matrix is counted.

between these two problems by observing that the hierarchical clustering problem does not admit a weight function in a natural way, and its special case of $M = 1$ corresponds to the unweighted version of correlation clustering (i.e. $wt(*) = 1$, recall the definition in Chapter 3). To save us from repetitive exclusions of trivial case $M = 1$, in the following I will always assume $M > 1$.

The first challenge that presents itself here is how to formulate distance matrices into graphs. Previous work on the correlation clustering problem is (almost) always conducted via a graph-theoretic approach. Unfortunately, in general we are not able to define *one* natural graph out of the distance matrix D, preserving the information in D. Known theoretical studies on this problem, unanimously making use of results of the correlation clustering problem, have to jettison some information to make an instance of correlation clustering, which consequently induces a loss of M factor. As an example, by defining a graph G_D from D where vertices are those elements, and an edge is present between a pair of vertices if their distance is at most M, and then applying a kernelization algorithm for the correlation clustering problem on G_D, a kernel of $\mathcal{O}(M \cdot k)$ elements can be easily derived. Information discarded in this formulation is the distances smaller than $M + 1$, which are indistinguishably represented as edges, and therefor it introduces a multiplicative factor M, which can be informally considered as an integrality gap due to relaxation..

As it turns out, this sacrifice is not really necessary, and the fundamental observation sparing us from this loss is: There are M natural graphs defined on the same vertex set V, one for each level, where an edge is present between a pair of elements if and only if their distance is below that level. More specifically, for the ground distance matrix D and any positive integer $t \in [M]$, the graph $G_D^t = (V, E_D^t)$ at level t is defined as follows: $E_D^t = \{(u, v) \,|\, D_{uv} \leqslant t\}$. The subscript D will be omitted when it is clear from the context which distance matrix is being referred to, so is the superscript t if it happens to be M. One can check that the $M = 1$ case stated above is the graph for $t = M = 1$. Moreover, the graphs considered in this paper are weighted such that the values of difference are respected, that is, to graphs from a distance matrix D, I assign weight to each pair of edges/anti-edges to represent their distance. Now for each $t \in [M]$, I define the weight function wt^t for G^t as follows: for each pair of elements $u, v \in V$ (not necessarily adjacent in G^t),

$$wt^t(uv) = \begin{cases} t + 1 - D_{uv} & \text{if } D_{uv} \leqslant t, \\ D_{uv} - t & \text{otherwise.} \end{cases} \tag{*}$$

I remark the weight function is well defined: It is easy to check that wt^t always gives positive integers, and its range is actually $[M]$. The two cases correspond to edges and anti-edges respectively. In particular, for the G^M, an edge is present between each pair of elements of distance $d \leqslant M$, with weight $M + 1 - d$, and no edge between distance-$(M + 1)$ pairs, with weight 1 $((M + 1) - M)$. The final remark on the graph formulation is: instead of statically fixing a graph fixed at the beginning, I will dynamically maintain graphs with the algorithmic procedure that evolves the matrix. With this definition at hand, I can immediately have:

Observation 1. *Let D be a distance matrix, each of the M graphs defined as above consists of a disjoint union of cliques if and only if D is ultrametric.*

Thus, a solution to a hierarchical clustering instance, corresponds to M solutions to correlation clustering instances, each from a graph G^t ($t \in [M]$). G^t will also be called the t-*perspective graph*, on which some definitions are given as follows, which are natural generalizations of standard terminologies from graph theory. A t-*clique* is a subset of elements which are pairwise connected in G^t, i.e. the distance between each pair of elements in them is at most t. Now Observation 1 can be rephrased as:

A distance matrix D is ultrametric, iff for each $t \in [M]$, the t-perspective graph G_D^t consists of a disjoint union of t-cliques. To t-*split* two disjoint subsets of elements X and Y is to increase the distance between each pair of elements in different parts to at least $t+1$, and more specifically, for each pair of $x \in X$ and $y \in Y$ such that $D_{xy} \leqslant t$, set it to $t+1$, while keep others unchanged. Similarly, to t-*merge* a subset V of elements is to decrease the distance between each pair of elements in V to lower than t, and more specifically, if $D_{xy} > t$ where $x, y \in V$, set it to t. A *cut* in a graph is defined by a subset X of vertices, and its weight is the total weight of all edges lying between X and $\overline{X}$. In different perspective graphs, the same subset X have different cuts, and I denote by $\gamma^t(X)$ the cut of X in the t-perspective graph G^t, that is

$$\gamma^t(X) = \sum_{\substack{u \in X, v \notin X \\ uv \in E^t}} \pi(uv) = \sum_{\substack{u \in X, v \notin X \\ D_{uv} \leqslant t}} (t + 1 - D_{uw}), \tag{4.3}$$

In the remainder of this section, I will only work on the M-perspective graph, where the weight of a missing edge is always 1. For the simplicity, wherever not specified otherwise, the clique, split and merge are always M-clique, M-split and M-merge.

Lemma 4.3. *Let* D *be the distance matrix on element set* V*,* $\mathcal{P} = \{V_1, V_2, \ldots, V_p\}$ *be a partition of* V*, and let* $E_{\mathcal{P}}$ *be the set of edges in* G *whose two ends belong to two different parts in* $\mathcal{P}$*. Then* $\sum_{i=1}^p \mathtt{opt}(D|_{V_i}) \leqslant \mathtt{opt}(D) \leqslant wt(E_{\mathcal{P}}) + \sum_{i=1}^p \mathtt{opt}(D|_{V_i})$.

Proof. Let S be an optimal solution to D. As noted above, for $1 \leqslant i \leqslant p$, $S|_{V_i}$ is a solution to the submatrix $D|_{V_i}$, which implies $\mathtt{opt}(D|_{V_i}) \leqslant \mathtt{cost}(S|_{V_i})$. Thus

$$\sum_{i=1}^{p} \mathtt{opt}(D|_{V_i}) \leqslant \sum_{i=1}^{p} \mathtt{cost}(S|_{V_i}) \leqslant \mathtt{cost}(S) = \mathtt{opt}(D).$$

Moreover, if I increase all inter-part distances to $M+1$, that is, split all parts by removing all edges in $E_{\mathcal{P}}$ from G, then apply an optimal solution S'_i to each submatrix $D|_{V_i}$, I will end up with a solution. Therefore, these operations make a solution to the original distance matrix D whose cost is

$$wt(E_{\mathcal{P}}) + \sum_{i=1}^{p} \mathtt{cost}(S'_i) = wt(E_{\mathcal{P}}) + \sum_{i=1}^{p} \mathtt{opt}(D|_{V_i}),$$

which is no less than $\mathtt{opt}(D)$. Now both inequalities are verified. □

Lemma 4.3 directly implies the following corollaries. First, if there is a partition such that all inter-part pairs have distance $M+1$, then $wt(E_{\mathcal{P}}) = 0$ and Lemma 4.3 gives

Corollary 4.4. *Let D be the distance matrix on element set V, $\mathcal{P} = \{V_1, V_2, \ldots, V_p\}$ be a partition of V. If $D_{uv} = M+1$ for all pairs of u and v which are in different parts of $\mathcal{P}$, then $\mathtt{opt}(D) = \sum_{i=1}^{p} \mathtt{opt}(D|_{V_i})$, and every optimal solution to D is a union of optimal solutions to the submatrices $D|_{V_1}, \ldots, D|_{V_p}$.*

When $p = 2$, i.e. the element partition is $\mathcal{P} = \{X, \overline{X}\}$, the edge set $E_{\mathcal{P}}$ has weight $\gamma(X)$. Lemma 4.3 gives

Corollary 4.5. *Let $X \subset V$ be a subset of elements, then*

$$\mathtt{opt}(D|_X) + \mathtt{opt}(D|_{\overline{X}}) \leqslant \mathtt{opt}(D) \leqslant \mathtt{opt}(D|_X) + \mathtt{opt}(D|_{\overline{X}}) + \gamma(X).$$

This corollary suggests a lower bound for the cost of an optimal solution in some bordering parts, as shown in the following lemma.

Lemma 4.6. . *Let S be an optimal solution to distance matrix D. For any subset of elements X, $\mathtt{cost}(S|_{X,\overline{X}}) \leqslant \gamma(X)$.*

Proof. The optimal solution S can be divided into three disjoint parts: $\mathsf{S}|_{\mathsf{X}}$, $\mathsf{S}|_{\overline{\mathsf{X}}}$, and $\mathsf{S}|_{\mathsf{X},\overline{\mathsf{X}}}$. By Corollary 4.5,

$$\mathtt{opt}(\mathsf{D}) = \mathtt{cost}(\mathsf{S}|_{\mathsf{X}}) + \mathtt{cost}(\mathsf{S}|_{\overline{\mathsf{X}}}) + \mathtt{cost}(\mathsf{S}|_{\mathsf{X},\overline{\mathsf{X}}}) \leqslant \mathtt{opt}(\mathsf{D}|_{\mathsf{X}}) + \mathtt{opt}(\mathsf{D}|_{\overline{\mathsf{X}}}) + \gamma(\mathsf{X}) \quad (4.4)$$

Again, since $\mathsf{S}|_{\mathsf{X}}$ is a solution to the submatrix $\mathsf{D}|_{\mathsf{X}}$ and $\mathsf{S}|_{\overline{\mathsf{X}}}$ is a solution to the submatrix $\mathsf{D}|_{\overline{\mathsf{X}}}$, I have $\mathtt{cost}(\mathsf{S}|_{\mathsf{X}}) \geqslant \mathtt{opt}(\mathsf{D}|_{\mathsf{X}})$ and $\mathtt{cost}(\mathsf{S}|_{\overline{\mathsf{X}}}) \geqslant \mathtt{opt}(\mathsf{D}|_{\overline{\mathsf{X}}})$, which combined with (4.4) give immediately $\mathtt{cost}(\mathsf{S}|_{\mathsf{X},\overline{\mathsf{X}}}) \leqslant \gamma(\mathsf{X})$. □

In a matrix with largest distance d, it does not make sense to increase any distance to higher than d. This observation can be formalized as the following lemma.

Lemma 4.7. *In the objective matrix* $\mathsf{D}' = \mathsf{D} + \mathsf{S}$ *for any optimal solution* S*, all distances of* D' *are no more than* $\mathsf{d} = \max_{1\leqslant i<j\leqslant n} \mathsf{D}_{ij}$*, that is, the largest distance in* D*.*

Proof. I prove by contradiction. Assume $\mathsf{d}' = \max_{1\leqslant i<j\leqslant n} \mathsf{D}'_{ij} > \mathsf{d}$, then I claim the following is a better solution:

$$\mathsf{S}'_{ij} = \begin{cases} \mathsf{S}_{ij} & \text{if } \mathsf{D}'_{ij} < \mathsf{d}', \\ \mathsf{S}_{ij} - 1 & \text{if } \mathsf{D}'_{ij} = \mathsf{d}'. \end{cases}$$

Obviously $\mathtt{cost}(\mathsf{S}') < \mathtt{cost}(\mathsf{S})$, as the only modifications happen on positive numbers, and I decrease them each by 1.

Now it remains to verify that S' is also a solution to D. Applying solutions S and S' to D, I get two different matrices. From each matrix, I can define M perspective graphs, and by the above construction, the only levels at which two perspective graphs are different are $(\mathsf{d}'-1)$ and d'. By Observation **??**, I only need to verify the

$(d'-1)$- and d'-perspective graphs for $D+S'$ are both disjoint unions of cliques. At level d', the whole set V becomes a single d'-clique. At level $d'-1$, all components are either $d'-1$-cliques in $D+S$, or d'-cliques. $\square$

With the help of Lemma 4.7, I can generalize Corollary 4.4 into the cases where the largest distance is not $M+1$.

Lemma 4.8. *Let* D *be the distance matrix on element set* V*,* $\mathcal{P} = \{V_1, V_2, \ldots, V_p\}$ *be a partition of* V*. If for each pair of elements* u *and* v *which are in different parts of* $\mathcal{P}$*, their distance* D_{uv} *is the maximum over the whole matrix* D*, then* $\mathtt{opt}(D) = \sum_{i=1}^{p} \mathtt{opt}(D|_{V_i})$*, and every optimal solution to* D *is a union of optimal solutions to the submatrices* $D|_{V_1}, \ldots, D|_{V_p}$*.*

Based on this corollary, if the largest distance $M' = \max_{1 \leqslant i < j \leqslant n} D_{ij}$ of D is less than $M+1$, I can treat it as an instance of $M'-1$ hierarchical clustering, and then solve it. Thus, in this paper, without loss of generality, I always assume there exists at least one pair of i and j such that $D_{ij} = M+1$.

4.2 The Kernelization Algorithm

For element $v \in V$, denote by $N^t[v] = \{u \mid D_{uv} \leqslant t\}$ those elements with distance to v upper bounded by t (recall that v itself is implicitly included because $D_{vv} = 0$), which form the closed neighborhood of v in G^t. In particular, $N^M[v] = \{u \mid D_{uv} \leqslant M\} = \{u \mid D_{uv} \neq M+1\}$.

A trivially simple but important fact about a solution S of cost $\leqslant k$ to D is: at most $2k$ different elements have some of their distances to other elements changed. As a consequence, if I am also able to bound the number of elements that are not affected by S, I get a kernel. For such an unaffected element v, the v-th row of S

consists of only 0's, then in the ultrametric matrix $D' = D + S$, for any neighbor $u \in N[v]$ and another different element w, the distance D'_{uw} has to satisfy

$$D'_{uw} \begin{cases} \leqslant \max(D_{vu}, D_{vw}) \leqslant M & \text{if } u, w \in N[v]; \\ = \max(D_{vu}, D_{vw}) = M + 1 & \text{if } u \in N[v], w \notin N[v]. \end{cases} \tag{4.5}$$

This is necessary (but not sufficient) conditions for the element v to be immune to a solution S, and if (4.5) is not satisfied by D, then $D|_{N[v]}$ has to be modified by S. To measure the cost of required modification, in addition to previously defined $\gamma^t(N[v])$, I define

$$\delta^t(v) = \sum_{\substack{D_{uv}, D_{vw} \leqslant t \\ D_{uw} > t}} (D_{uw} - t), \tag{4.6}$$

and $\rho^t(v) = 2\delta^t(v) + \gamma^t(N[v])$. $N^t[v]$ is said to be *reducible (at level* t*)* if $\rho^t(v) < |N^t[v]|$. Again, the superscript t is dropped when $t = M$.

The proofs in this section, although many non-trivial calculations involved, follow from a very simple observation, that is, by explicitly constructing a solution S, I can exclude all possibilities ending with solutions of a cost higher than $\mathtt{cost}(S)$. I describe two reduction rules on the neighborhood $N[v]$ of an element v such that $N[v]$ is reducible. In fact, the reducibility of $N[v]$ is the only reduction condition I need on which the three reduction rules are applied in order. The first one tells that an reducible neighborhood can be put into a single M-clique.

Lemma 4.9. *For any element* v *such that* $N[v]$ *is reducible, there is an optimal solution* S^* *to* D *such that the maximum distance in* $(D + S^*)|_{N[v]}$ *is upper bounded by* M.

Proof. Let S be an optimal solution to D, and $D' = D + S$. Since the neighborhood $N[v]$ is reducible, I have $\rho(v) < |N[v]|$. Suppose that $N[v]$ does not form a single clique in $G_{D'}$, then it can be divided into disjoint subsets such that each is contained in different M-cliques, let them be $N[v] = C_1 \cup C_2 \cup \cdots \cup C_r$, where $C_i \neq \emptyset$ for $1 \leqslant i \leqslant r$. Note that C_i $(1 \leqslant i \leqslant r)$ itself does not induce a clique in $G_{D'}$, instead, it is the intersection of such a clique and $N[v]$.

Our first step is to construct a solution to the instance induced by submatrix $D|_{N[v]}$, and I obtain it by modifying the known solution $S|_{N[v]}$:

$$S'_{uw} = \begin{cases} S_{uw} & \text{if } \exists i, u, w \in C_i, \\ S_{uw} - 1 & \text{otherwise.} \end{cases} \tag{4.7}$$

That is, values of S' are the same as those of S only with the exception of inter-part pairs. In particular, all inter-part pairs have distance $M + 1$ in $(D + S)|_{N[v]}$, and M in $D|_{N[v]} + S'$ (note S' is a $|N[v]| \times |N[v]|$ matrix).

Claim 1. *S' is a solution to $D|_{N[v]}$, with cost at most $\mathtt{cost}(S|_{N[v]}) - \sum_{1 \leqslant i < j \leqslant r} |C_i||C_j| + 2\delta(v)$.*

Proof of Claim 1. Observe $N[v]$ forms a M-clique in M-perspective graph of $D|_{N[v]} + S'$. For each part C_i, $D|_{C_i} + S'|_{C_i} = D|_{C_i} + S|_{C_i}$, which is upper bounded by M by the partition of $N[v]$ defined above; and all inter-part distance is exactly M by (4.7). Thus $N[v]$ forms a single M-clique. Furthermore, by Lemma 4.8, and noting all inter-part distances are M, each C_i can be solved spearately, for which I simply use $S'|_{C_i}$. Thus, S' is a solution to $D|_{N[v]}$.

To calculate the cost of S', I only need to care about those positions different from S, that is, those inter-part ones. Let δ_1 be the number of those inter-part pairs with distances $M+1$ in $D|_{N[v]}$, which is obviously a subset of pairs defined in $\delta(v)$, and therefore $\delta_1 \leqslant \delta(v)$, then

$$\begin{aligned} \mathtt{cost}(S') &= \mathtt{cost}(S|_{N[v]}) - (\sum_{1\leqslant i<j\leqslant r} |C_i||C_j| - \delta_1) + \delta_1 \\ &= \mathtt{cost}(S|_{N[v]}) - \sum_{1\leqslant i<j\leqslant r} |C_i||C_j| + 2\delta_1 \qquad (4.8) \\ &\leqslant \mathtt{cost}(S|_{N[v]}) - \sum_{1\leqslant i<j\leqslant r} |C_i||C_j| + 2\delta(v). \end{aligned}$$

This completes the proof of Claim 1. $\square$

According to Corollary 4.5, $\mathtt{opt}(D) \leqslant \mathtt{opt}(D|_{N[v]}) + \mathtt{opt}(D|_{\overline{N[v]}}) + \gamma(N[v])$, and noting $\mathtt{opt}(D|_{N[v]}) \leqslant \mathtt{cost}(S')$, I have

$$\begin{aligned} \mathtt{opt}(D) &\leqslant \mathtt{cost}(S') + \mathtt{opt}(D|_{\overline{N[v]}}) + \gamma(N[v]) \\ &\leqslant \mathtt{cost}(S|_{N[v]}) - \sum_{1\leqslant i<j\leqslant r} |C_i||C_j| + 2\delta(v) + \mathtt{opt}(D|_{\overline{N[v]}}) + \gamma(N[v]) \qquad (4.9) \\ &= \mathtt{cost}(S|_{N[v]}) + \mathtt{opt}(D|_{\overline{N[v]}}) + \rho(v) - \sum_{1\leqslant i<j\leqslant r} |C_i||C_j|. \end{aligned}$$

On the other hand, $\mathtt{opt}(D) = \mathtt{cost}(S)$ by the optimality of S, I have

$$\mathtt{opt}(D) \geqslant \mathtt{cost}(S|_{\overline{N[v]}}) + \mathtt{cost}(S|_{N[v]}) \geqslant \mathtt{opt}(D|_{\overline{N[v]}}) + \mathtt{cost}(S|_{N[v]}). \qquad (4.10)$$

Combining (4.9) and (4.10), and noting that $N[v]$ is reducible, I get

$$\sum_{1\leqslant i<j\leqslant r} |C_i||C_j| \leqslant \rho(v) < |N[v]| = \sum_{1\leqslant i\leqslant r} |C_i|. \qquad (4.11)$$

This can hold true only when $r = 2$, and one of $|C_1|, |C_2|$ is 1. In both cases, I have $|C_1| \cdot |C_2| = |C_1| + |C_2| - 1$. Combining this with (4.11), and noting that all the quantities are integers, it must be $\rho(v) = \sum_{1 \leqslant i < j \leqslant r} |C_i||C_j| = |N[v]| - 1$, and

$$\texttt{opt}(D) = \texttt{cost}(S') + \texttt{opt}(D|_{\overline{N[v]}}) + \gamma(N[v]). \tag{4.12}$$

Therefore (4.12) shows that if I first M-split V into $N[v]$ and $\overline{N[v]}$ with cost $\gamma(N[v])$, apply S' to $N[v]$ with S', and then apply an optimal solution to the submatrix indexed by $\overline{N[v]}$, then I have an optimal solution S^* to the whole matrix D. This completes the proof of the lemma because the optimal solution S^* has the element set $N[v]$ entirely contained in a single M-clique in the M-perspective graph of $D + S^*$. □

By Lemma 4.9, the optimal solution S^* merges $N[v]$ into a M-clique by decreasing all distance $M + 1$ in $D|_{N[v]}$ to M, with a total cost $\delta(v)$. Therefore, the remaining pairs in S^* make an optimal solution to the resulting matrix. This gives the rule for our first reduction rule:

Rule 4.1. *For an element v such that $N[v]$ is reducible, decrease $D|_{N[v]}$ to at most M, and decrease the parameter k by $\delta(v)$.*

After Rule 4.1, no distance in $D|_{N[v]}$ is $M+1$, and therefore $\delta(v) = 0$ and $\rho(v) = \gamma(N[v])$. Now I turn to $D|_{N[v],\overline{N[v]}}$. Particularly, I am interested in those elements whose distance to $N[v]$ is mostly $M + 1$. Note that the definition of reducibility guarantees that almost all elements in $\overline{N[v]}$ have to fall into this category.

Rule 4.2. *Let v be an element such that $N[v]$ is reducible on which Rule 4.1 has been applied. For each element $x \notin N[v]$, if $\sum_{u \in N[v]}(M + 1 - D_{xu}) \leqslant |N[v]|/2$, then M-split x from $N[v]$ and decrease k accordingly.*

Lemma 4.10. *Rule 4.2 is safe.*

Proof. By Lemma 4.9, there is an optimal solution S to D such that $N[v]$ is entirely contained in a single M-clique C in $D' = D + S$. I first prove, by contradiction, that C containing $N[v]$ in $G_{D'}$ contains at most one element not in $N[v]$. Suppose, on the contrary, that there are r elements $u_1, \ldots, u_r$ not in $N[v]$ that are in C, where $r \geqslant 2$. For $1 \leqslant i \leqslant r$, let c_i be the total cost of M-splitting u_i from $N[v]$; and let m_i be the total cost of M-merging u_i to $N[v]$. Note that $c_i + m_i \geqslant |N[v]|$ and $\sum_{i=1}^{r} c_i \leqslant \gamma(N[v])$. Then in the optimal solution S to D, the total cost on decreasing distance between pairs of $N[v]$ and $\{u_1, \ldots, u_r\}$ is at least

$$\begin{aligned}
\sum_{i=1}^{r} m_i &= \sum_{i=1}^{r} (c_i + m_i - c_i) = \sum_{i=1}^{r} (c_i + m_i) - \sum_{i=1}^{r} c_i \\
&\geqslant \sum_{i=1}^{r} |N[v]| - \gamma(N[v]) = r|N[v]| - \gamma(N[v]) \\
&> 2|N[v]| - |N[v]| = |N[v]| \\
&> \gamma(N[v]).
\end{aligned}$$

Herein, I have used the fact $|N[v]| > \gamma(N[v])$ (this is because by the conditions of the rule, $\rho(v) = 2\delta(v) + \gamma(N[v]) < |N[v]|$). But this contradicts Corollary 4.6.

Therefore, there is at most one element x out of $N[v]$ that is in the M-clique C containing $N[v]$ in $D + S$. I can assume that the element x satisfies the condition $\sum_{u \in N[v]} (M + 1 - D_{xu}) > |N[v]|/2$: otherwise, instead of decreasing $D|_{x,N[v]}$ to M to make $N[v] \cup x$ a M-clique, I increase all distances in $D|_{x,N[v]}$ to $M + 1$ and will get another optimal solution S' that makes the subset of elements $N[v]$ a separated M-clique in $D + S'$. In consequence, for an element x not in $N[v]$ with $\sum_{u \in N[v]} (M + 1 - D_{xu}) \leqslant |N[v]|/2$, I can always assume that x is not in the M-clique containing $N[v]$

in $D + S$ for the optimal solution S. In particular, increasing the distance between x and $N[v]$ to $M + 1$ for such an element x is always safe. □

After Rule 4.1-4.2 are applied, the $D|_{N[v],\overline{N[v]}}$ has a very simple structure, which is characterized by the following lemma:

Lemma 4.11. *Let v be an element such that $N[v]$ is reducible on which Rules 4.1-4.2 have been applied. Then there is at most one element $x \in \overline{N[v]}$ such that $D|_{N[v],x}$ have values not $M + 1$.*

Proof. By the condition of Rule 4.2, any element x not in $N[v]$ that still has distance smaller than $M + 1$ to some elements of $N[v]$ after the application of Rule 4.2 must satisfy $\sum_{u \in N[v]}(M + 1 - D_{xu}) > |N[v]|/2$. To prove the lemma, suppose on the contrary that there are two such elements, let them be x and y, then I have

$$\gamma(N[v]) \geqslant \sum_{u \in N[v]} (M + 1 - D_{xu}) + \geqslant \sum_{u \in N[v]} (M + 1 - D_{yu}) > |N[v]|.$$

But this contradicts the assumption that $N[v]$ is reducible, that is, $\rho(v) = 2\delta(v) + \gamma(N[v]) < |N[v]|$. □

Hereafter I will call this only element x as the *pendent element w.r.t.* v. Now it is ready to describe our kernelization algorithm, which is simply an application of the above-mentioned rules in order.

The kernelization algorithm. For each element v such that $N[v]$ is reducible

1. decrease value $M + 1$ in $D|_{N[v]}$ to M and decrease k accordingly;
2. for each element $x \notin N[v]$ such that $\sum_{u \in N[v]}(M + 1 - D_{xu}) \leqslant |N[v]|/2$, set all values in $D|_{N[v],x}$ to $M + 1$ and decrease k accordingly;

Note that: 1) there is only one condition for the rules of the whole Algorithm, and it is only checked once; and 2) the reduction condition does not depend on the parameter k.

This kernelization algorithm is applied in an iterative way, that is, I start from the highest level M, and then for each isolated element set split in last run, re-apply it at the level $M-1$, and so on, until there is no more isolation operation and stop. Therefore, the kernel consists of some isolated element sets which each forms an independent instance of hierarchical clustering problem with a different value of M. To analyze the size of the final kernel, I simply count the relation between each set and the cost of the modifications required to make the matrix ultrametric. Because our counting will not dependent on the value of M, this ratio holds for all subsets which form independent instances, and therefore to the whole set.

Lemma 4.12. *Let (V, D, k) be an instance of the hierarchical clustering problem on which our kernelization algorithm has been applied. If V has more than $4k$ elements, then no solution to D has its cost upper bounded by $\leqslant k$.*

Proof. Let matrix S be an optimal solution to the distance matrix D. For each element pair $\{v, w\}$ in V, I divide the cost $|S_{vw}|$ into two halves and distribute them evenly to the two elements v and w. By this procedure, each element v gets a "cost" $\mathsf{cost}(v) = \frac{1}{2}\sum_{u \in V-v} |S_{uv}|$. Obviously the total cost of S is equal to $\sum_{v\in V} \mathsf{cost}(v)$. In this proof the costs are counted on each elements, and particular interest will be paid to those elements with 0 cost.

For any two elements u, v with distance $D_{uv} = M + 1$, if there exists another element w such that both $D_{uw} \leqslant M$ and $D_{vw} \leqslant M$, then at most one of u, v can has 0 cost: to make u, v, and w ultrametric, at least one of u and v has to share some cost. Let $Z_S = \{v_1, v_2, \ldots, v_r\}$ be the set of elements with 0 cost in S. Then for each two elements v_i and v_j in Z_S, either their distance is $M+1$ and any other element has distance $M+1$ to at least one of them; or their distance is $\leqslant M$ and any other element has distance $M+1$ to v_i if and only if it has distance $M+1$ to v_j. As a result, any two of the neighborhood $\{N[v_1], N[v_2], \dots, N[v_r]\}$ in G are either the same or mutually disjoint. Thus, without loss of generality, it can be assumed that all neighborhoods in $\{N[v_1], N[v_2], \dots, N[v_r]\}$ are pairwise disjoint (otherwise, I can simply remove duplicated copies of the neighborhoods in the collection). Let $N_S = N[v_1] \cup N[v_2] \cup \cdots \cup N[v_r]$.

Without changing any value in the column indexed by v_i, a solution S has to decrease distance $M+1$ between any pair of elements in $N[v_1]$ to $\leqslant M$, and increase distance $\leqslant M$ between any element in $N[v_1]$ and $\overline{N[v_1]}$ to $M+1$. These operations induce cost at least $\delta(v_i) + \gamma(v_i)$, and counted on elements, it is in total $\delta(v_i) + \gamma(v_i)/2 = \rho(v)/2$. If $N[v_i]$ is not reducible, then the cost is at least $|N[v_i]|/2$. If $N[v_i]$ is reducible, then by Lemma 4.11 there can be at most one element x whose distances to $N[v_i]$ are not all $M+1$. According to Rule 4.2, in this case I have $\rho(v_i) > |N[v_i]|/2$, and thus the cost is strictly larger than $|N[v_i]|/4$. From this analysis, I get

$$\sum_{v \in N_S} \mathsf{cost}(v) = \sum_{i=1}^{r} \sum_{v \in N[v_i]} \mathsf{cost}(v) \geqslant \sum_{i=1}^{r} |N[v_i]|/4 = |N_S|/4. \tag{4.13}$$

On the other hand, each w of the elements not in the set N_S has cost at least one non-zero value in the column $S|_{V-w,w}$, and therefore bears cost at least $1/2$. This gives

$$\sum_{v \in V - N_S} \mathsf{cost}(v) \geqslant |V - N_S|/2. \tag{4.14}$$

Putting (4.13) and (4.14) together, it can be concluded the total cost of the optimal solution S is

$$\sum_{v\in V}\mathsf{cost}(v) = \sum_{v\in N_s}\mathsf{cost}(v) + \sum_{v\in V-N_S}\mathsf{cost}(v) \geqslant |N_s|/4 + |V-N_S|/2 \geqslant |V|/4.$$

Therefore, if $|V| > 4k$, then D has no solution of cost bounded by k. □

4.3 The Parameterized Algorithm

An example. Inspired by the formulation and usage of perspective graphs in last section, one might want to solve the hierarchical clustering problem in a level-by-level way, that is, given an algorithm $\mathcal{A}$ for the correlation clustering problem, it is applied to the M-perspective graph G^M, and then on each sub-instances of level $M-1$ it makes, and continue till level 1. However, this greedy approach does not work, as shown in the following counter-example (Figure 4.1). This graph is actually a $K_5 \times K_1$, where the K_5 (at the bottom) has pairwise distance 1, while the K_1 vertex x (at the top) is connected to three vertices of the K_5 with edges of distance $M+1$ (red edges), while the others two with M (blue edge) and 1 (black edge) respectively.

It is easy to verify that the (only) optimal solution should be increasing both the black edge and the blue edge to $M+1$, with total cost $M+1$. On the other hand, it is even clearer that the G, where the three red edges are missed and the black edge has weight M, has only one optimal solution which adds the three missed edges back, with cost 3. Now in the resulted distance matrix, x has distance M to four elements in the K_5, and distance 1 to the other element. When M is large enough, the optimal cost of this new instance is $M-1$ (still increasing the distance of the black edge M), which is a lower bound for the cost of any solution further obtained. Hence, the

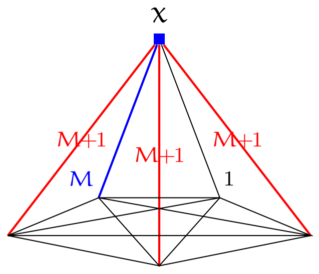


Fig. 4.1. An instance of hierarchical clustering that cannot be solved greedily

solution returned by this level-by-level approach has cost $\geqslant 3 + (M - 1) = M + 2$, which is strictly larger than the optimal cost $M + 1$ this instance admits. In other words, we lose the chance of obtaining optimality with the first step.

This counter-example does not rule out the possibility of using algorithms for the correlation clustering problem to the hierarchical clustering problem. One commonality between the correlation clustering problem and hierarchical clustering problem is both of them can be characterized as being *conflict-triple/triangle* free. Recall that a conflict triple is a triple of elements whose three pairwise distances satisfy $d_1 \leqslant d_2 < d_3$. To make a distance matrix ultrametric, we have to break each conflict triple, which can only be done by modifying at least one of the three distances. Following from this observation, it is a good exercise to design an $\mathcal{O}^*(3^k)$ FPT algorithm. From the viewpoint of t-perspective graphs, there is exactly one edge missed (between the pair of elements with the largest distance d_3) from G^t for each $d_2 \leqslant t < d_3$. This strucutre, two edges present and one edge missed in an subgraph induced by three vertices, is exactly the conflict-triangle widely used in the studies of correlation clustering problem. Here the objective is to break each conflict-triangle, which can only be done by reversing at least one of the three pairs.

For correlation clustering problem, there have been several improved results published, all of which follow the same basic observation, that is, branching on the conflict-triangles. With the help of more careful branching steps and more complicated analysis techniques, the best published algorithm for the correlation clustering problem takes time $\mathcal{O}^*(1.62^k)$ [22]. On the other hand, there has been no non-trivial FPT algorithm proposed for the hierarchical clustering problem.

Instead of adapting one particular algorithm for the correlation clustering problem to the hierarchical clustering problem, I go one step further, that is, I show any FPT algorithm for the correlation clustering problem, provided it is based on branching on breaking conflict-triangles, can be adapted to the hierarchical clustering problem, with the same time complexity as far as the exponential part is concerned! Indeed, what I will show is a *meta algorithm*, which takes, in addition to an instance $I_H = (V, D, k)$ of the hierarchical clustering problem, an algorithm for the correlation clustering problem, and returns an optimal solution to I_H.

The meta algorithm is described in Figure 4.2. To be qualified as the main ingredient in this meta-algorithm, the algorithm $\mathcal{A}$ has to satisfy some stipulations.

A $\mathcal{A}$ can be described as a searching tree $\mathcal{T}$, whose leaves correspond to all solutions (many-to-one), while internal nodes correspode to partial solutions. Here by a partial solution, we mean a set of *irreversible* edge insertions and/or deletions each breaks some conflict-triangle, and is set to be permament (its subnodes cannot insert a deleted edge back or delete an inserted edge).

B $\mathcal{A}$ transverses $\mathcal{T}$ by following every branch of $\mathcal{T}$, and stops only when it reaches a leaf, or a partial solution whose cost already exceeds k.

C The running time of $\mathcal{A}$ is measured by the number of nodes it transverses.

```
**Algorithm Meta-HC**$(V, D, k, \mathcal{A})$
INPUT: A set of elements $V$, $|V| \times |V|$ distance matrix $D$, integer $k$,
        and algorithm $\mathcal{A}$ for correlation clustering problem
OUTPUT: An $|V| \times |V|$ matrix $S$ such that $D + S$ is ultrametic and $\texttt{cost}(S) \leqslant k$, if such
a matrix exists.
0   $M = \max_{1 \leqslant i < j \leqslant n}(D_{ij}) - 1$;
1   construct the $M$-perspective graph $G^M$ as well as the weight function $wt^M$ as
    defined in Section 4.1;
2   for each solution $S$ returned by $\mathcal{A}(G^M, wt^M, k)$ do
3     if $M == 1$ then return $S$;
4     $D' = D + S$, $k' = k - \texttt{cost}(S)$;
5     for each pair of $i$ and $j$ in $1..n$ do
6       if $D'_{ij} = M + 1$ then set $D'_{ij} = M^{\dagger}$;
7     return $S +$ Meta-HC$(V, D', k', \mathcal{A})$.
```

†: This (tricky) step is used to simplify the presentation of this algorithm. In particular, after each iteration, we do not break it into pieces of subintances and solve them independently, instead, we still treat it as a single instance. Note that if there is no conflict triangle at level $M + 1$, edges with distance $M + 1$ totally partition the elements. By uniformally decreasing them to M, they still induce a total partition, and thus will not make new triangle conflicts. Indeed, the instance remains equivalent according to Lemma 4.4.

Fig. 4.2. A meta-algorithm for hierarchical clustering

With the meta-algorithm and stipulations, we are ready to present the main result of this section:

Theorem 4.13. *Let $\mathcal{A}$ be an algorithm on weighted correlation clustering problem that is based on breaking conflict triangles and satisfies all the three stipulations stated above. Then the algorithm **Meta-HC** using the algorithm $\mathcal{A}$ solves the hierarchical clustering problem with the same time complexity as $\mathcal{A}$, up to a polynomial factor.*

Proof. I first show the correctness of Algorithm **Meta-HC**, that is, given an instance I_H of the hierarchical clustering problem, if it admits solutions with cost no more than k, **Meta-HC** can always find one. Denote by $\mathcal{S}$ the set of solutions to I_H with cost no more than k.

The first observation is, for any solution $S \in \mathcal{S}$, it has to break all conflict triangles in G^t at each level t, and as an example, we start from the M-perspective graph G. Note that the distance between a pair of elements is changed to break the conflict triangle(s) in G only if $D_{uv} \leqslant M$ and $D_{uv} + S_{uv} > M$, or vice versa. Hence any pair of elements u and v such that $S_{uv} = 0$ is not counted, neither are those with $D_{uv} \leqslant M$ and $D_{uv} + S_{uv} \leqslant M$. For those pairs which are counted, if $S_{uv} > 0$ and $D_{uv} + S_{uv} = M + 1$, we count S_{uv}; while $S_{uv} < 0$ and $D_{uv} = M + 1$, we count 1. By this way, we get the cost paid by S to break all conflict triangles in G, and which is obviously $\leqslant \texttt{cost}(S) \leqslant k$. By **Stipulation B**, this node must be transversed by $\mathcal{A}$. Now that level M is conflict triangle free, we can turn to each submatrix, and similarly distribute cost of S to them, level by level. The total costs of the whole procedure is exactly $\texttt{cost}(S)$, if and only if there is no *counteracting operations* on any pair. By counteracting operations on a pair of elements u and v we mean decreasing D_{uv} at some step and later increasing it, or the inverse. Assume such counteracting operations, "decrease-then-increase", have been applied on D_{uv}. If t is the level it gets last decreased, then by above distribution, its new value must be t, and thus it cannot be increased by later operations conducted at levels lower than t. Similarly for the counteracting operations "increase-then-decrease", if t is the level it gets increased, then after that u and v will be in different subinstances in lower levels, and thus no operations can be applied to them.

Now it remains to show the time complexity, for which we will only concern us with the exponential part. This directly following from the fact that we transverse the same tree and same number of nodes, as well as the **Stipulation C**. Therefore, **Meta-HC** takes the same time as $\mathcal{A}$ up to a polynomial factor. $\square$

Particularly, the algorithm given in [22] is such a branching algorithm of running time $\mathcal{O}(1.62^k)$, and thus can be used in my meta algorithm. Now Theorem 4.2 comes as a direct corollary of Theorem 4.13 and Theorem 7 in [23].

4.4 Discussion

Inspired by the results presented above, one immediate question is: How about the approximation? Currently, the best approximation ratio of approximation algorithms for the hierarchical clustering problem is $M + 2$ [5, 188], while the correlation clustering problem admits a 2.5 approximation algorithm [6, 188]. Rooted from the same idea, these two algorithms follow the same greedy procedure, where the M inevitably make its role similar as a weight. As illustrated in the following example (a similar tight example for the correlation clustering problem can be found in the first author's dissertation [4]):

> a $(n - 2)$-clique with distance 1 for each pair, and two other vertices x, y such that the distances between x to the clique are 1; the distances between y to the clique are M, while $d(x, y) = M + 1$,

the approximation ratio $(M + 2)$ in Ailon and Charikar's algorithm[2] is (asymptotically) tight, and thus to achieve a better approximation ratio new algorithm has to be designed. Now the question is: Can it be designed in a way that the levels play a role more obedient, and therefore the hierarchical structure is also compressed in the analysis.

[2] I remark here that the analysis of [5] is very subtle and involved. Indeed, the original analysis given in the conference version (FOCS 2005) contains a flaw, which is later fixed in the full version.

5. FEEDBACK VERTEX SET[⋆]

The main thrust of this chapter is an improved parameterized algorithm for the feedback vertex set (FVS) problem, based on a small kernel for a variation of it. Recall that an FVS in a graph is a set of vertices whose removal breaks all cycles.

My approach, as some of the previous ones, is to study a variation of the FVS problem, the *disjoint feedback vertex set* problem (disjoint-FVS), which asks for an FVS in a graph G that has no overlap with a given FVS. This is a natural step to solve the FVS problem in the framework of iterative compression. First I show that disjoint-FVS admits a small kernel (Theorem 5.1), and can be solved in polynomial time when the graph has a special topological structure that is closely related to the maximum genus of the graph. I then propose a simple branch-and-bound process on disjoint-FVS, and introduce a new branch-and-bound measure. The branch-and-bound process effectively reduces a given graph to a graph with the special structure, and the new measure more precisely evaluates the efficiency of the branch-and-bound process. These algorithmic, combinatorial, and topological structural studies enable an $\mathcal{O}^*(3.83^k)$-time parameterized algorithm for the general FVS problem, improving the previous best algorithm of time $\mathcal{O}^*(5^k)$ for the problem.

Theorem 5.1. *There is a polynomial-time kernelization algorithm for the* DISJOINT-FVS *problem that produces a kernel that contains at most* $4k$ *vertices.*

Theorem 5.2. *The FVS problem is solvable in time* $\mathcal{O}^*(3.83^k)$.

[⋆] Reprinted with permission from *On feedback vertex set: New measure and new structures,* by Y. Cao, J. Chen, and Y. Liu, 2010, Algorithm Theory-SWAT 2010, pp. 93-104, Copyright 2010 by Springer-Verlag.

5.1 Disjoint-FVS and Its Kernel

Let me start with a precise definition of the first problem.

disjoint-FVS. Given a graph $G = (V, E)$, an FVS F in G, and a parameter k, either construct an FVS F' of size k in G such that $F' \cap F = \emptyset$, or report that no such an FVS F' exists.

It has to be "no" if the subgraph induced by F contains a cycle, and thus I can always assume F induces a forest. Let $V_1 = V \setminus F$, which, by definition, also induces a forest, and denote an FVS entirely contained in V_1 by V_1-*FVS*. Therefore, an instance of disjoint-FVS can be written as $(G; V_1, V_2; k)$, where $(V_1, V_2 = F)$ is a partition of the vertex set of the graph G both inducing forests, and looks for a V_1-FVS of size k in the graph G. Recall that $d_G(v)$ ($d_{G[V_1]}(v)$ resp.) is the degree of the vertex v in the original graph G (the induced subgraph $G[V_1]$ resp.).

Given an instance $(G; V_1, V_2; k)$, apply the following two simple rules:

Rule 5.1. *Remove all vertices v with $d_G(v) \leqslant 1$;*

Rule 5.2. *For a vertex v in V_1 with $d_G(v) = 2$,*

- *if the two neighbors of v are in the same connected component of $G[V_2]$, then include v into the objective V_1-FVS, $G = G - v$, and $k = k - 1$;*
- *otherwise, move v from V_1 to V_2: $V_1 = V_1 \setminus \{v\}$, $V_2 = V_2 \cup \{v\}$.*[1]

Note that the second case in Rule 5.2 includes the case where one or both neighbors of v are not in V_2.

[1]Readers who are familiar with previous algorithms for this problem may be curious about the way I handle degree-2 vertices here. When a vertex v is excluded from the objective FVS, most previous works (e.g., [51, 64]) "smoothen" v (i.e., replacing v and the two edges incident to v with a new edge connecting the two neighbors of v). The difference here is that I am focused on kernelization that bounds the number of vertices in V_1. During the kernelization process, new degree-2 vertices can be created in the set V_1 but cannot be ignored when counting the number of vertices in V_1.

The correctness of Rule 5.1 is trivial: no degree-0 or degree-1 vertices can be contained in any cycle. On the other hand, although Rule 5.2 is also easy to verify for the unrestricted FVS problem [51] (because any cycle containing a degree-2 vertex v must also contain the two neighbors of v), it is much less obvious for the disjoint-FVS problem – the two neighbors of the degree-2 vertex v may not be in V_1 and cannot be included in the objective V_1-FVS. For this, I have the following lemma.

Lemma 5.3. *Rule 5.2 is safe. In particular, for any degree-2 vertex v in the set V_1 whose two neighbors are not in the same connected component of $G[V_2]$, there is a minimum V_1-FVS that does not contain v.*

Proof. In the first case, v and some vertices in V_2 form a cycle. Therefore, in order to break this cycle, the vertex v must be contained in the objective V_1-FVS.

For the second case, it suffices to show that if the graph G has a V_1-FVS F' , then G has a V_1-FVS of size at most $|F'|$ that does not contain v. If one u_1 of the neighbors of v is in V_1, then the set $(F' \setminus \{v\}) \cup \{u_1\}$ will be such a V_1-FVS. Thus, I can assume that the two neighbors u_1 and u_2 of v are in two different connected components in $G[V_2]$. Since $G \setminus F'$ is acyclic, there is either no path or a unique path in $G \setminus F'$ between u_1 and u_2. If there is no path, then adding v to $G \setminus F'$ does not create any cycle, and hence the set $F' \setminus \{v\}$ is a V_1-FVS of size $|F'| - 1$ that does not contain v. If there is a unique path P, then it must contain at least one vertex w in V_1 (since u_1 and u_2 are in different connected components in $G[V_2]$). Removing w will break this unique path, and the situation becomes the same as above, and thus the set $(F' \setminus \{v\}) \cup \{w\}$ is a V_1-FVS of the same size as F' but does not contain v. □

As a caveat, the second case of Rule 5.2 cannot be applied *simultaneously* on more than one vertex in V_1. For example, let v_1 and v_2 be two degree-2 vertices in V_1 that are both adjacent to two vertices u_1 and u_2 in V_2. Then it is obvious that one cannot move both v_1 and v_2 to V_2. In fact, if I first apply the second case of Rule 5.2 on v_1, then the first case of Rule 5.2 will become applicable on v_2.

Definition An instance $(G; V_1, V_2; k)$ of the disjoint-FVS problem is V_1-*irreducible* if none of the Rules 5.1-5.2 can be applied on vertices in the set V_1 (as a result, all vertices in V_1 must have degree larger than 2). An instance $(G; V_1, V_2; k)$ is *nearly* V_1-*irreducible* if in the set V_1 there is at most one vertex of degree 2 and all other vertices in V_1 are of degree larger than 2.

For an instance $(G; V_1, V_2; k)$ that is (nearly) V_1-irreducible, in case there is no ambiguity, I will simply say that the graph G is (nearly) V_1-irreducible, respectively. In the following, I show that a nearly V_1-irreducible instance necessarily has a small size.

I start with a simple branch-and-bound algorithm **FindFVS** for nearly V_1-irreducible instances of the disjoint-FVS problem, as given in Figure 5.1. The algorithm is similar to the one presented in [51], but gives degree-2 vertices a higher priority when selecting a vertex for branching. The basic step of the algorithm is to pick a vertex v in V_1 and branch on either including or excluding v in the objective V_1-FVS F. Note that in certain situations, the algorithm directly takes one of the two actions in the branching (see the footnotes in the algorithm). The correctness of these actions are ensured by Lemma 5.3, which, in consequence, ensures the correctness of the algorithm.

Note that since I will use this algorithm to count the number of vertices in the set V_1, Rules 5.1-5.2 are *not* applied on vertices of degree less than 3 in the set V_1 that are generated during the process of the algorithm – I only assume that the input instance $(G; V_1, V_2, k)$ is nearly V_1-irreducible.

```
Algorithm FindFVS
INPUT: a nearly $V_1$-irreducible instance $(G; V_1, V_2; k)$ of disjoint-FVS.
OUTPUT: a $V_1$-FVS $F$ of size $\leqslant k$ in $G$, or report that no such $V_1$-FVS exists.
1.   $F = \emptyset$;
2.   while $|V_1| > 0$ and $k \geqslant 0$ do
3.     if there are vertices in $V_1$ that have degree 2 in $G$
4.     then let $v$ be such a vertex
5.     else let $v$ be a vertex in $V_1$ that has degree $\leqslant 1$ in the induced subgraph $G[V_1]$
6.     branching
7.         case 1: \\ $v$ is in the objective $V_1$-FVS $F$.
8.             † add $v$ to $F$ and delete $v$ from $G$;  $k = k - 1$;
9.         case 2: \\ $v$ is not in the objective $V_1$-FVS $F$.
10.            ‡ move $v$ from $V_1$ to $V_2$;
11.  if $|V_1| = 0$ then return $F$ else return "no $V_1$-FVS of size $\leqslant k$".

† this action will not be taken if $d_G(v) = 2$ and the two neighbors of $v$ are
  not in the same connected component of $G[V_2]$.
‡ this action will not be taken if two neighbors of $v$ are in the same connected
  component of $G[V_2]$.
```

Fig. 5.1. A simple branch-and-bound algorithm for disjoint-FVS

Lemma 5.4. *Each execution of steps 6-10 of the algorithm* **FindFVS** *results in a nearly* V_1*-irreducible instance.*

Proof. Since the input instance is nearly V_1-irreducible, it suffices to prove that on a nearly V_1-irreducible instance, the execution of steps 6-10 of the algorithm produces a nearly V_1-irreducible instance.

Steps 6-10 either delete the vertex v from the graph (case 1) or move v from set V_1 to set V_2 (case 2). Moving v from V_1 to V_2 does not change the degree of any vertex remaining in the set V_1. Therefore, if the branching action of steps 6-10 is to move v from V_1 to V_2, then the resulting instance is also nearly V_1-irreducible. Note that by the second footnote in the algorithm, the action of steps 9-10 will not be taken if two neighbors of v are in the same connected component in the induced subgraph

$G[V_2]$. This ensures that steps 9-10 produces a valid instance of the disjoint-FVS problem.

Now consider the action of steps 7-8 in the algorithm that deletes the vertex v from the graph. If $d_G(v) = 2$ and the two neighbors of v are in the same connected component of $G[V_2]$, or if v has degree 0 in $G[V_1]$ (i.e., $d_{G[V_1]}(v) = 0$), then deleting v does not affect the degree of any vertex remaining in the set V_1. Therefore, in these cases the action of steps 7-8 in the algorithm produces a nearly V_1-irreducible instance. Note that by the first footnote in the algorithm, if $d_G(v) = 2$ and the two neighbors of v are not in the same connected component of $G[V_2]$, then the action of steps 7-8 of the algorithm will not be taken. Therefore, the only remaining case I need to examine is that $d_G(v) \geqslant 3$ and $d_{G[V_1]}(v) \geqslant 1$. By step 5 of the algorithm, in this case, I must have $d_{G[V_1]} = 1$. Let w be the unique neighbor of v in $G[V_1]$. By the way I picked the vertex v and the assumption $d_G(v) \geqslant 3$, no vertex in V_1 has degree 2 in G. In particular, $d_G(w) \geqslant 3$. Therefore, deleting the vertex v can result in at most one degree-2 vertex in V_1 (i.e., w) and will keep all other vertices in V_1 with degree at least 3. Thus, in this case the action of steps 7-8 of the algorithm again produces a nearly V_1-irreducible instance. $\square$

Now I am ready for the main result in this section. A *computational path* of the algorithm **FindFVS** is a sequence of in-order executions of the algorithm that in steps 6-10 executes the action of either case 1 or case 2 (but not both).

Theorem 5.5. *Let* $(G; V_1, V_2; k)$ *be a nearly* V_1*-irreducible instance of the disjoint-FVS problem, and let* τ_1 *and* τ_2 *be the number of connected components in the induced subgraphs* $G[V_1]$ *and* $G[V_2]$*, respectively. Let* δ_2 *be the number of vertices in* V_1 *that have degree* 2 *in* G*. If* $|V_1| > \delta_2 + 2k + \tau_2 - \tau_1 - 1$*, then there is no* V_1*-FVS of size bounded by* k *in the graph* G*.*

Table 5.1
Moving the vertex v from V_1 to V_2

degree of v	neighbors of v	δ_2'	k'	τ_1'	τ_2'	V_1'
$d_G(v) = 2$	$w_1, w_2 \in V_1$	$\delta_2 - 1$	k	$\tau_1 + 1$	$\tau_2 + 1$	$V_1 - \{v\}$
with neighbors	$w_1, w_2 \in V_2$	$\delta_2 - 1$	k	$\tau_1 - 1$	$\tau_2 - 1$	$V_1 - \{v\}$
w_1 and w_2	$w_1 \in V_1$, $w_2 \in V_2$	$\delta_2 - 1$	k	τ_1	τ_2	$V_1 - \{v\}$
$d_G(v) \geqslant 3$	$\lvert N(v) \cap V_1\rvert = 0$	δ_2	k	$\tau_1 - 1$	$\leqslant \tau_2 - 2$	$V_1 - \{v\}$
$d_G(v) \geqslant 3$	$\lvert N(v) \cap V_1\rvert = 1$	δ_2	k	τ_1	$\leqslant \tau_2 - 1$	$V_1 - \{v\}$

Table 5.2
Deleting the vertex v from G

degree of v	neighbors of v	δ_2'	k'	τ_1'	τ_2'	V_1'
$d_G(v) = 2$	$w_1, w_2 \in V_1$					
with neighbors	$w_1, w_2 \in V_2$	$\delta_2 - 1$	$k - 1$	$\tau_1 - 1$	τ_2	$V_1 - \{v\}$
w_1 and w_2	$w_1 \in V_1$, $w_2 \in V_2$					
$d_G(v) \geqslant 3$	$\lvert N(v) \cap V_1\rvert = 0$	δ_2	$k - 1$	$\tau_1 - 1$	τ_2	$V_1 - \{v\}$
$d_G(v) \geqslant 3$	$\lvert N(v) \cap V_1\rvert = 1$	$\leqslant \delta_2 + 1$	$k - 1$	τ_1	τ_2	$V_1 - \{v\}$

Proof. I prove the theorem by induction on the number $|V_1|$ of vertices in the set V_1. The initial case $|V_1| = 1$ is easy to verify and thus omitted (in this case $\tau_1 = 1$ must hold true, and then the condition $|V_1| > \delta_2 + 2k + \tau_2 - \tau_1 - 1$ implies $\delta_2 + 2k + \tau_2 \leqslant 2$).

For the general case of $|V_1| > 1$, let $(G; V_1, V_2; k)$ be a nearly V_1-irreducible instance of disjoint-FVS and suppose that the graph G has a V_1-FVS of size bounded by k. Since the algorithm **FindFVS** solves the disjoint-FVS problem correctly, there is a computational path $\mathcal{P}$ of the algorithm that returns a V_1-FVS F with $|F| \leqslant k$. Consider how the path $\mathcal{P}$ changes the values of an instance when it executes (correctly) the action of one of the cases in steps 6-10 in the algorithm. Let $|V_1|$, δ_2, k, τ_1, and τ_2 be the values before the execution of steps 6-10, and let $|V_1'|$, δ_2', k', τ_1', and τ_2' be the corresponding values after the execution of steps 6-10. The relations between these values are summarized in Tables 5.1 and 5.2, where many are obvious. Given below are explanations for some less obvious ones in the figure.

First consider the case where the computational path $\mathcal{P}$ takes the action of case 2 in the algorithm, i.e., moving the vertex v from V_1 to V_2. See Table 5.1.

If $d_G(v) = 2$ and both neighbors w_1 and w_2 of v are in V_2, and if v is moved from V_1 to V_2 (see the 3rd line in Table 5.1), then by the second footnote in the algorithm, w_1 and w_2 must belong to two different connected components of the induced subgraph $G[V_2]$. Therefore, moving v from V_1 to V_2 must decrease τ_1 by 1 (because v by itself makes a connected component in $G[V_1]$) and merge the two connected components of $G[V_2]$ into one (i.e., $\tau_2' = \tau_2 - 1$).

If $d_G(v) \geqslant 3$ and v has no neighbor in V_1, and if v is moved from V_1 to V_2 (see the 5th line in Table 5.1), then all neighbors of v (there are at least 3) are in different connected component of $G[V_2]$. Therefore, moving v from V_1 to V_2 decreases the value τ_1 by 1 (i.e., $\tau_1' = \tau_1 - 1$) and merges at least three connected components of $G[V_2]$ into one (i.e., $\tau_2' \leqslant \tau_2 - 2$).

If $d_G(v) \geqslant 3$ and $|N(v) \cap V_1| = 1$, and v is moved from V_1 to V_2 (see the 6th line in Table 5.1), then the value τ_1 is unchanged (i.e., $\tau_1' = \tau_1$), and again by the second footnote in the algorithm, the value τ_2 is decreased by at least 1 (i.e., $\tau_2' \leqslant \tau_2 - 1$).

Now consider the case where the computational path $\mathcal{P}$ takes the action of case 1 in the algorithm, i.e., deleting the vertex v from the graph G. See Table 5.2. First note that by the first footnote in the algorithm, if the vertex v has degree 2 and if the two neighbors of v do not belong to the same connected component of $G[V_2]$, then the action of case 1 in the algorithm is not taken. In particular, the action of case 1 in the algorithm is not applicable under the conditions of the 2nd line and the 4th line in Table 5.2.

If $d_G(v) \geqslant 3$ and if v has no neighbors in V_1 (see the 5th line in Table 5.2), then deleting v only decreases the value τ_1 by 1 (i.e., $\tau_1' = \tau_1 - 1$).

Finally, if $d_G(v) \geqslant 3$ and $|N(v) \cup V_1| = 1$ (see the 6th line in Table 5.2). Let w be the unique neighbor of v in V_1. Then, deleting v may create at most one degree-2 vertex (i.e., w) in the set V_1 (i.e., $\delta_2' \leqslant \delta_2 + 1$), while unchanging the values of τ_1 and τ_2.

This verifies all relations in Tables 5.1 and 5.2.

Let $(G'; V_1', V_2'; k')$ be the instance produced by the computational path $\mathcal{P}$ on the nearly V_1-irreducible instance $(G; V_1, V_2; k)$. By the assumption, the graph G has a V_1-FVS of size bounded by k. Since we also assume that the computational path $\mathcal{P}$ is correct, the graph G' must have a V_1'-FVS of size bounded by k'. Since $|V_1'| = |V_1| - 1$ and by Lemma 5.4, the instance $(G'; V_1', V_2'; k')$ is nearly V_1'-irreducible, we can apply the induction hypothesis on the instance $(G'; V_1', V_2'; k')$, which gives $|V_1'| \leqslant \delta_2' + 2k' + \tau_2' - \tau_1' - 1$. This gives

$$|V_1| = |V_1'| + 1 \leqslant \delta_2' + 2k' + \tau_2' - \tau_1' - 1 + 1.$$

Using this inequality to examine each situation in Tables 5.1 and 5.2, we can easily verify that the inequality

$$|V_1| \leqslant \delta_2 + 2k + \tau_2 - \tau_1 - 1$$

holds true. Therefore, if $|V_1| > \delta_2 + 2k + \tau_2 - \tau_1 - 1$, then the graph G has no V_1-FVS of size bounded by k. $\square$

Since a V_1-irreducible instance is also nearly V_1-irreducible in which $\delta_2 = 0$, we get immediately

Corollary 5.6. *Let* $(G; , V_1, V_2; k)$ *be a* V_1*-irreducible instance of the disjoint-FVS problem. If* $|V_1| > 2k + \tau_2 - \tau_1 - 1$*, then there is no* V_1*-FVS of size bounded by* k *in the graph* G*.*

Note that for disjoint-FVS instances that have been considered in the literature, e.g., [51,64], it is always assumed that $|V_2| = k+1$. By the simple fact that $\tau_2 \leqslant |V_2|$ and $\tau_1 > 0$, we have $2k+\tau_2-\tau_1-1 \leqslant 3k-1$. Therefore, in this case, a V_1-irreducible instance $(G; V_1, V_2; k)$ will have no desired V_1-FVS unless the set V_1 contains no more than $3k - 1$ vertices, and the vertices in the whole graph is upper bounded by $(3k - 1) + (k + 1) = 4k$, which implies the Theorem 5.1. This improves the previous best upper bound of $4k$ on the size of V_1, as presented in [64]. In fact, the bound given in Corollary 5.6 is *tight*, which can be seen as follows. Consider the graph G in Figure 5.2, which consists of $2k + 1$ vertices w_1, w_2, v_1, v_2, ..., v_{2k-1}, where $k \geqslant 2$ is an arbitrary positive integer. The vertices of G are partitioned into two sets $V_1 = \{v_1, v_2, \ldots, v_{2k-1}\}$ and $V_2 = \{w_1, w_2\}$, and $(G; V_1, V_2; k)$ is a V_1-irreducible instance of the disjoint-FVS problem. Note that $\tau_1 = \tau_2 = 1$. We have $|V_1| = 2k - 1 = 2k + \tau_2 - \tau_1 - 1$, while the graph G has a V_1-FVS F of k vertices: $F = \{v_1, v_3, v_5, \ldots, v_{2k-1}\}$.

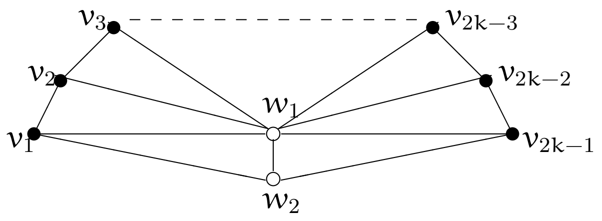


Fig. 5.2. An example showing the tightness of Corollary 5.6

Finally, I remark that my kernelization result was obtained based on a branch-and-bound algorithm, i.e., **FindFVS**, for the problem, instead of on the analysis of the resulting structures after applications of reduction rules. This technique, to my best knowledge, had not been used in the literature of kernelization.

5.2 A Polynomial-Time Solvable Case for Disjoint-FVS

This section examines a special class of instances for the DISJOINT-FVS problem. This approach is closely related to the classical study on graph maximum genus embeddings [47, 104]. However, the study on graph maximum genus embeddings that is related to my approach is based on general spanning trees of a graph, while my approach must be restricted to only spanning trees that are constrained by the vertex partition (V_1, V_2) of an instance $(G; V_1, V_2; k)$ of disjoint-FVS.

Since the induced subgraph $G[V_2]$ is a forest, there exists a spanning tree T of the graph G that contains $G[V_2]$[2]. Such a spanning tree will be called a $T_{G[V_2]}$-*tree*. By the construction, every edge in $E(G) - E(T)$ has at least one end in V_1. Two edges in $E(G) - E(T)$ are V_1-*adjacent* if they have a common end in V_1. A V_1-*adjacency matching* in $E(G) - E(T)$ is a partition of the edges in $E(G) - E(T)$ into groups of one or two edges, called *1-groups* and *2-groups*, respectively, such that two edges in the same 2-group are V_1-adjacent. A *maximum* V_1-*adjacency matching* in $E(G) - E(T)$ is a V_1-adjacency matching in $E(G) - E(T)$ that maximizes the number of 2-groups, and equivalently minimize the total number of groups.

Definition Let $(G; V_1, V_2; k)$ be an instance of the disjoint-FVS problem. The V_1-*adjacency matching number* $\mu(G, T)$ of a $T_{G[V_2]}$-tree T in G is the number of 2-groups in a maximum V_1-adjacency matching in $E(G) - E(T)$. The V_1-*adjacency matching*

[2]Actually, by slightly adapting Kruskal's algorithm, this can be constructed in time $O(m\alpha(n))$, where $\alpha(n)$ is the inverse of Ackermann function [46, 184].

number $\mu(G)$ of the graph G is the largest $\mu(G, T)$ over all $T_{G[V_2]}$-trees T in it.

An instance $(G; V_1, V_2; k)$ of disjoint-FVS is *3-regular*$_{V_1}$ if every vertex in the set V_1 has degree exactly 3. Let $f_{V_1}(G)$ be the size of a minimum V_1-FVS for G. Let $\beta(G)$ be the *Betti number* of G that is the total number of edges in $E(G) - E(T)$ for any spanning tree T in G. Note that the edge set $E(G) - E(T)$ forms a basis of the *fundamental cycles* for the graph G such that *every* cycle in G contains at least one edge in $E(G) - E(T)$. In this sense, $\beta(G)$ is the number of fundamental cycles in the graph G [104]. The following lemma is a nontrivial generalization of a result in [150] (the result in [150] is a special case for Lemma 5.7 in which all vertices in the set V_2 have degree 2).

Lemma 5.7. *For any 3-regular*$_{V_1}$ *instance* $(G; V_1, V_2; k)$ *of the disjoint-FVS problem, we have* $f_{V_1}(G) = \beta(G) - \mu(G)$*. Moreover, a minimum* V_1*-FVS of the graph* G *can be constructed in linear time from a* $T_{G[V_2]}$*-tree whose* V_1*-adjacency matching number is* $\mu(G)$*.*

Proof. First note that a maximum V_1-adjacency matching in $E(G) - E(T)$ for a $T_{G[V_2]}$-tree T can be constructed in linear time, as follows. Let G^T be the graph induced by the edge set $E(G) - E(T)$ (i.e., the vertex set of G^T consists of the ends of the edges in $E(G) - E(T)$, and the edge set of G^T is $E(G) - E(T)$). Since each vertex in V_1 has degree 3 and T is a spanning tree in G, each vertex in G^T has degree bounded by 2. Thus, each connected component of G^T is either a simple path or a simple cycle. Therefore, a maximum V_1-adjacency matching in $E(G) - E(T)$ can be constructed trivially by maximally pairing the edges in each connected component of G^T.

Let T be a $T_{G[V_2]}$-tree such that there is a V_1-adjacency matching M in $E(G) - E(T)$ that contains $\mu(G)$ 2-groups. Let U be the set of edges that are in the 1-groups in M. We construct a V_1-FVS F as follows: (1) for each edge e in U, arbitrarily pick an end of e that is in V_1 and include it in F; and (2) for each 2-group of two V_1-adjacent edges e_1 and e_1 in M, pick the vertex in V_1 that is a common end of e_1 and e_2 and include it in F. Note that every cycle in the graph G contains at least one edge in $E(G) - E(T)$, while now every edge in $E(G) - E(T)$ has at least one end in F. Therefore, F is an FVS. By the above construction, F is a V_1-FVS. The number of vertices in F is equal to $|U| + \mu(G)$. Since $|U| = |E(G) - E(T)| - 2\mu(G) = \beta(G) - 2\mu(G)$, we have $|F| = \beta(G) - \mu(G)$. This concludes that

$$f_{V_1}(G) \leqslant \beta(G) - \mu(G). \tag{5.1}$$

Now consider the other direction. Let F be a minimum V_1-FVS for the graph $G = (V, E)$, i.e., $|F| = f_{V_1}(G)$. Let $\overline{F} = V \setminus F$, then the induced subgraph $G[\overline{F}]$ is a forest, and thus, there is a spanning tree T in G that contains the entire subgraph $G[\overline{F}]$. We construct a V_1-adjacency matching in $E(G) - E(T)$ and show that it contains at least $\beta(G) - |F|$ 2-groups. Since T contains $G[\overline{F}]$, each edge in $E(G) - E(T)$ has at least one end in F. Let E_2 be the set of edges in $E(G) - E(T)$ that have their both ends in F, and let E_1 be the set of edges in $E(G) - E(T)$ that have exactly one end in F.

Claim. Each end of an edge in E_2 is shared by exactly one edge in E_1.

In particular, no two edges in E_2 share a common end.

To prove the above claim, first note that since T is a spanning tree in G, each vertex in $F \subseteq V_1$, which has degree 3 in G, can be incident to at most two edges in $E(G) - E(T) = E_1 \cup E_2$. In particular, if u is an end of an edge $[u, v]$ in E_2 (i.e., $u, v \in F$), then there is at most one other edge in $E_1 \cup E_2$ that is incident to u. Now assume to the contrary of the claim that the vertex u is not shared by an edge in

E_1. Then for the other two edges e_1 and e_2 in G that are incident to u, either both e_1 and e_2 are in T or exactly one of e_1 and e_2 is in E_2. If both e_1 and e_2 are in T, then every edge in $E(G) - E(T)$ (including $[u, v]$) has at least one end in $F \setminus \{u\}$. Similarly, if exactly one $[u, w]$ of the edges e_1 and e_2 is in E_2, where w is also in F, then again every edge in $E(G)-E(T)$ (including $[u, v]$ and $[u, w]$) has at least one end in $F \setminus \{u\}$. Thus, in either case, $F \setminus \{u\}$ would make a smaller V_1-FVS, contradicting the assumption that F is a minimum V_1-FVS. This proves the claim.

Suppose that there are m_2 vertices in F that are incident to two edges in $E(G)-E(T)$. Thus, each of the rest $|F| - m_2$ vertices in F is incident to at most one edge in $E(G) - E(T)$. By counting the total number of incidencies between the vertices in F and the edges in $E(G) - E(T)$, we get

$$2|E_2| + |E_1| = 2|E_2| + (\beta(G) - |E_2|) \leqslant 2m_2 + (|F| - m_2),$$

or equivalently,

$$m_2 - |E_2| \geqslant \beta(G) - |F|. \tag{5.2}$$

Now we construct a V_1-adjacency matching in $E(G)-E(T)$, as follows. For each edge e in E_2, by the above claim, we can make a 2-group that consists of e and an edge in E_1 that shares an end in V_1 with e (note that this grouping will not put an edge in E_1 in two different 2-groups because if the edge e in E_2 shares an end with an edge e' in E_1, then e' cannot share an end with any other edges in E_2). Besides the ends of the edges in E_2, there are $m_2 - 2|E_2|$ vertices in F that are incident to two edges in E_1. For each v of these vertices, we make a 2-group that consists of the two edges in E_1 that are incident to v. Note that this construction of 2-groups never re-uses any edges in $E(G) - E(T)$ more than once. Therefore, the construction gives

$|E_2|+(m_2-2|E_2|) = m_2-|E_2|$ disjoint 2-groups. We then make each of the rest edges in $E(G)-E(T)$ a 1-group. This gives a V_1-adjacency matching in $E(G)-E(T)$ that has $m_2-|E_2|$ 2-groups. By Inequality (5.2) and by definition, we have

$$\mu(G) \geqslant \mu(G,T) \geqslant m_2 - |E_2| \geqslant \beta(G) - |F| = \beta(G) - f_{V_1}(G). \tag{5.3}$$

Combining (5.1) and (5.3), we conclude with $f_{V_1}(G) = \beta(G) - \mu(G)$. □

By Lemma 5.7, in order to construct a minimum V_1-FVS for a 3-regular$_{V_1}$ instance $(G; V_1, V_2, k)$ of disjoint-FVS, we only need to construct a $T_{G[V_2]}$-tree in the graph G whose V_1-adjacency matching number is $\mu(G)$. The construction of an unconstrained maximum adjacency matching in terms of general spanning trees has been considered by Furst, Gross and McGeoch in their study of graph maximum genus embeddings [104]. I follow a similar approach, based on cographic matroid parity, to construct a $T_{G[V_2]}$-tree in G whose V_1-adjacency matching number is $\mu(G)$. I start with a quick review on the related concepts in matroid theory. More detailed discussion on matroid theory can be found in [155].

A *matroid* is a pair $(E, \mathfrak{I})$, where E is a finite set and $\mathfrak{I}$ is a collection of subsets of E that satisfies the following properties (note that the collection $\mathfrak{I}$ may not be explicitly given but is defined in terms of certain subset properties):

(1) If $A \in \mathfrak{I}$ and $B \subseteq A$, then $B \in \mathfrak{I}$;

(2) If $A, B \in \mathfrak{I}$ and $|A| > |B|$, then there is an element $a \in A \setminus B$ such that $B \cup \{a\} \in \mathfrak{I}$.

The *matroid parity* problem is stated as follows: given a matroid $(E, \mathfrak{I})$ and a perfect pairing $\{[a_1, \overline{a}_1], [a_2, \overline{a}_2], \ldots, [a_n, \overline{a}_n]\}$ of the elements in the set E, find a largest subset M in $\mathfrak{I}$ such that for all i, $1 \leqslant i \leqslant n$, either both a_i and $\overline{a}_i$ are in M, or neither of a_i and $\overline{a}_i$ is in M.

Each connected graph G is associated with a *cographic matroid* $(E_G, \mathfrak{I}_G)$, where E_G is the edge set of G, and an edge set S is in $\mathfrak{I}_G$ if and only if $G - S$ is connected. It is well-known that matroid parity problem for cographic matroids can be solved in polynomial time [155]. The fastest known algorithm for cographic matroid parity problem is by Gabow and Stallmann [105], which runs in time $\mathcal{O}(mn \log^6 n)$.

The following explains how to reduce the problem to the cographic matroid parity problem. Let $(G; V_1, V_2; k)$ be a 3-regular$_{V_1}$ instance of the disjoint-FVS problem. Without loss of generality, we make the following assumptions: (1) the graph G is connected (otherwise, we simply work on each connected component of G); and (2) for each vertex v in V_1, there is at most one edge from v to a connected component in $G[V_2]$ (otherwise, we can directly include v in the objective V_1-FVS).

Recall that two edges are V_1-*adjacent* if they share a common end in V_1. For an edge e in G, denote by $d_{V_1}(e)$ the number of edges in G that are V_1-adjacent to e (note that an edge can be V_1-adjacent to the edge e from either end of e).

The *labeled subdivision* G_2 of the graph G is constructed as follows:

1. shrink each connected component of $G[V_2]$ into a single vertex; let the resulting graph be G_1;
2. assign each edge in G_1 a distinguished label;
3. for each edge labeled e_0 in G_1, suppose the edges V_1-adjacent to e_0 are labeled by e_1, e_2, …, e_d (in arbitrary order), where $d = d_{V_1}(e_0)$; subdivide e_0 into d *segment edges* by inserting $d-1$ degree-2 vertices in e_0, and label the segment edges by (e_0e_1), (e_0e_2), …, (e_0e_d). Let the resulting graph be G_2. The segment edges (e_0e_1), (e_0e_2), …, (e_0e_d) in G_2 are said to be *from* the edge e_0 in G_1.

There are a number of interesting properties for the graphs constructed above. First, each of the edges in the graph G_1 corresponds uniquely to an edge in G that has at least one end in V_1. Thus, without creating any confusion, we will simply say that the edge is in the graph G or in the graph G_1. Second, because of the assumptions we made on the graph G, the graph G_1 is a simple and connected graph. In consequence, the graph G_2 is also a simple and connected graph. Finally, because each edge in G_1 corresponds to an edge in G that has at least one end in V_1, and because each vertex in V_1 has degree 3, every edge in G_1 is subdivided into at least two segment edges in G_2.

Now in the labeled subdivision graph G_2, pair the segment edge labeled $(e_0 e_i)$ with the segment edge labeled $(e_i e_0)$ for all segment edges (note that $(e_0 e_i)$ is a segment edge from the edge e_0 in G_1 and that $(e_i e_0)$ is a segment edge from the edge e_i in G_1). By the above remarks, this is a perfect pairing $\mathcal{P}$ of the edges in G_2. Now with this edge pairing $\mathcal{P}$ in G_2, and with the cographic matroid $(E_{G_2}, \mathfrak{I}_{G_2})$ for the graph G_2, we call Gabow and Stallmann's algorithm [105] for the cographic matroid parity problem. The algorithm produces a maximum edge subset M in $\mathfrak{I}_{G_2}$ that, for each segment edge $(e_0 e_i)$ in G_2, either contains both $(e_0 e_i)$ and $(e_i e_0)$, or contains neither of $(e_0 e_i)$ and $(e_i e_0)$.

Lemma 5.8. *From the edge subset M in $\mathfrak{I}_{G_2}$ constructed above, a $T_{G[V_2]}$-tree for the graph G with a V_1-adjacency matching number $\mu(G)$ can be constructed in time $O(m\alpha(n))$, where n and m are the number of vertices and the number of edges, respectively, of the original graph G.*

Proof. Suppose that the edge subset M consists of the edge pairs $\{[(e_1 e_1'), (e_1' e_1)], \ldots, [(e_h e_h'), (e_h' e_h)]\}$ in G_2. Since $M \in \mathfrak{I}_{G_2}$, $G_2 - M$ is connected. Thus, for each edge e_i in G_1, there is at most one segment edge in M that is from e_i. Therefore, the edge subset M corresponds to an edge subset M' of exactly $2h$ edges in G_1 (thus exactly $2h$ edges in G): $M' = \{e_1, e_1'; \ldots, e_h, e_h'\}$, where for $1 \leqslant i \leqslant h$, the edges e_i and e_i' are V_1-adjacent. Since $G_2 - M$ is connected, it is easy to verify that the graph $G_1 - M'$ (thus the graph $G - M'$) is also connected. Also note that the graph $G - M'$ contains the induced subgraph $G[V_2]$ because no edge in G_1 has its both ends in V_2. Therefore, by we can construct, in time $O(m\alpha(n))$, a $T_{G[V_2]}$-tree T_1 for the graph $G - M'$, which is also a $T_{G[V_2]}$-tree for the graph G. Now if we make each pair $[e_i, e_i']$ a 2-group for $1 \leqslant i \leqslant h$, and make each of the rest edges in $E(G) - E(T_1)$ a 1-group, we get a V_1-adjacency matching with h 2-groups in $E(G) - E(T_1)$.

To complete the proof of the lemma, we only need to show that $h = \mu(G)$. For this, it suffices to show that no $T_{G[V_2]}$-tree can have a V_1-adjacency matching with more than h 2-groups. Let T_2 be a $T_{G[V_2]}$-tree with q 2-groups $[e_1, e_1']$, …, $[e_q, e_q']$ in $E(G) - E(T_2)$. Since T_2 is entirely contained in $G - \cup_{i=1}^{q}\{e_i, e_i'\}$, $G - \cup_{i=1}^{q}\{e_i, e_i'\}$ is connected. In consequence, the graph $G_1 - \cup_{i=1}^{q}\{e_i, e_i'\}$ is also connected. From this, it is easy to verify that the graph $G_2 - \cup_{i=1}^{q}\{(e_i e_i'), (e_i' e_i)\}$ is also connected. Therefore, the edge subset $\{(e_1 e_1'), (e_1' e_1); \ldots, (e_q e_q'), (e_q' e_q)\}$ is in $\mathfrak{I}_{G_2}$. Now since M is the the solution of the matroid parity problem for the cographic matroid $(E_{G_2}, \mathfrak{I}_{G_2})$ and since M consists of h edge pairs, we must have $h \geqslant q$. This completes the proof of the lemma. $\square$

Now it is ready to present the main result in this section.

Theorem 5.9. *There is an $\mathcal{O}(n^2 \log^6 n)$-time algorithm that on a 3-regular$_{V_1}$ instance $(G; V_1, V_2; k)$ of the* DISJOINT-FVS *problem, either constructs a V_1-FVS of size bounded by k, if such a V_1-FVS exists, or reports correctly that no such a V_1-FVS exists.*

Proof. For the 3-regular$_{V_1}$ instance $(G; V_1, V_2; k)$ of DISJOINT-FVS, we first construct the graph G_1 in linear time by shrinking each connected component of $G[V_2]$ into a single vertex. Note that since each vertex in V_1 has degree 3, the total number of edges in G_1 is bounded by $3|V_1|$. From the graph G_1, we construct the labeled subdivision graph G_2. Again since each vertex in V_1 has degree 3, each edge in G_1 is subdivided into at most 4 segment edges in G_2. Therefore, the number n_2 of vertices and the number m_2 of edges in G_2 are both bounded by $\mathcal{O}(|V_1|) = \mathcal{O}(n)$. From the graph G_2, we apply Gabow and Stallmann's algorithm [105] on the cographic matroid $(E_{G_2}, \mathfrak{I}_{G_2})$ that produces the edge subset M in $\mathfrak{I}_{G_2}$ in time $\mathcal{O}(m_2 n_2 \log^6 n_2) = \mathcal{O}(n^2 \log^6 n)$. By Lemma 5.8, from the edge subset M, we can construct in time $\mathcal{O}(m\alpha(n))$ a $T_{G[V_2]}$-tree T for the graph G whose V_1-adjacency matching number is $\mu(G)$. Finally, by Lemma 5.7, from the $T_{G[V_2]}$-tree T, we can construct a minimum V_1-FVS F in linear time. Now the solution to the 3-regular$_{V_1}$ instance $(G; V_1, V_2; k)$ of disjoint-FVS can be trivially derived by comparing the size of F and the parameter k. Summarizing all these steps gives the proof of the theorem. $\square$

Theorem 5.9 and Lemma 5.3 together immediately imply

Corollary 5.10. *There is a polynomial time algorithm that on an instance* $(G; V_1, V_2; k)$ *of disjoint-FVS where all vertices in* V_1 *have degree bounded by 3, either constructs a* V_1*-FVS of size bounded by* k*, if such an FVS exists, or reports correctly that no such a* V_1*-FVS exists.*

I remark that Corollary 5.10 is the best possible in terms of the maximum vertex degree in the set V_1. This can be reasoned as follows. Rizzi [171] proved that the FVS problem on graphs of maximum vertex degree 4 is NP-hard (in fact, he proved that the problem is APX-hard, but it is easy to verify that his reductions are also valid for the proof of NP-hardness of the problem). Given an instance G of the FVS problem on graphs of maximum vertex degree 4, we add a degree-2 vertex to the middle of each edge in G. Let the new graph be G'. Let V_1 be the set of vertices in G' that correspond to the original vertices in G, and let V_2 be the set of new degree-2 vertices in G'. Now it is rather straightforward to see that a minimum V_1-FVS in G' corresponds to a minimum FVS in the original graph G. Moreover, the maximum vertex degree in the set V_1 in G' is bounded by 4. This proves that the disjoint-FVS problem is NP-hard even when restricted to graphs in which the maximum vertex degree in the set V_1 is 4.

5.3 An Improved Algorithm for Disjoint-FVS

Now come back to the general disjoint-FVS problem without degree restriction. Before presenting the algorithm in Figure 5.3, I need to explain the terminologies used in it. A vertex v in the set V_1 is a *nice* V_1*-vertex* if v is of degree 3 and all its three neighbors belong to the set V_2. I denote by p the number of nice V_1-vertices in V_1, and, as before, by τ_2 the number of connected components in $G[V_2]$. As a slight abuse of the set union operation in step 4, the union $\{w\} \cup \textbf{Feedback}(G - w, V_1 \setminus \{w\}, V_2, k - 1)$ is interpreted as a 'No' when $\textbf{Feedback}(G - w, V_1 \setminus \{w\}, V_2, k - 1)$ is 'No'. Step 5 simply applies the second case of the Rule 5.2 (note the first case does not apply after step 4). Finally, in step 7, I assume that I have picked an (arbitrary) vertex in each tree of $G[V_1]$ and designated it as the root of this tree so that each tree is rooted, in which a *lowest parent* w is a vertex that has children and all its children are leaves.

This section will be devoted to establish the correctness of this algorithm and bound its running time. I start with the following lemma that, not only justifies

```
Algorithm Feedback($G, V_1, V_2, k$)
INPUT: an instance $(G; V_1, V_2; k)$ of disjoint-FVS.
OUTPUT: a $V_1$-FVS $F$ of size $\leqslant k$ in $G$ if such a $V_1$-FVS exists, or 'No' otherwise.

0.   $p$ = number of nice $V_1$-vertices; $\tau_2$ = number of connected components in $G[V_2]$.
1.   if $(k < 0)$ or ($k = 0$ and $G$ is not a forest) or $(2p \geqslant 2k + \tau_2)$ then return 'No';
2.   if ($k \geqslant 0$ and $G$ is a forest) or $(p = |V_1|)$ then solve it in polynomial time;
3.   if a vertex $w \in V_1$ has degree $\leqslant 1$ then return Feedback($G - w, V_1 \setminus \{w\}, V_2, k$);
4.   if a vertex $w \in V_1$ has two neighbors in the same connected component in $G[V_2]$
     then return $\{w\} \cup$ Feedback($G - w, V_1 \setminus \{w\}, V_2, k - 1$);
5.   if a vertex $w \in V_1$ has degree 2 then
       return Feedback($G', V_1 \setminus \{w\}, V_2, k$); \\ add an edge between neighbors of $w$
6.   if a non-nice $V_1$-vertex $w$ satisfies $|N(w) \cap V_1| \leqslant 1$, and $|N(w) \cap V_2| \geqslant 3$ then
6.1    $F_1$ = Feedback($G - w, V_1 \setminus \{w\}, V_2, k - 1$);
6.2    if $F_1 \neq$ 'No' then return $F_1 \cup \{w\}$
6.3    else return Feedback($G, V_1 \setminus \{w\}, V_2 \cup \{w\}, k$);
7.   pick a lowest parent $w$ in any tree in $G[V_1]$ and let $v$ be a child of $w$;
7.1    $F_1$ = Feedback($G - w, V_1 \setminus \{w, v\}, V_2 \cup \{v\}, k - 1$);
7.2    if $F_1 \neq$ 'No' then return $F_1 \cup \{w\}$
7.3    else return Feedback($G, V_1 \setminus \{w\}, V_2 \cup \{w\}, k$).
```

Fig. 5.3. An algorithm for disjoint-FVS

step 1, but also reveals the composite measure (solution size coupled with number of connected components in $G[V_2]$,) I will be using to analyze the time complexity.

Lemma 5.11. *If* $2p \geqslant 2k + \tau_2$*, then there is no* V_1*-FVS of size bounded by* k *in* G*.*

Proof. Suppose that F is a V_1-FVS such that $|F| = k' \leqslant k \leqslant p$ ($k \leqslant p$ follows from the condition). Let $V_1' \subseteq V_1 \setminus F$ be the set of nice V_1-vertices that are not in F, and $p' = |V_1'|$, then $p' \geqslant p - k'$. By definition of F, the subgraph $G' = G[V_2 \cup V_1']$ induced by the vertex set $V_2 \cup V_1'$ is a forest, which means

$$|V_2| + p' = |V(G')| > |E(G')| = |V_2| - \tau_2 + 3p',$$

and $2k + \tau_2 > 2k + 2p' \geqslant 2k + 2(p - k') \geqslant 2p$, which contradicts the condition. □

Lemma 5.12. *The algorithm* **Feedback** *solves the disjoint-FVS problem correctly.*

Proof. Steps 1-2 serve as the exit conditions, whose correctness follows from Lemma 5.11, Corollary 5.10, and other trivial facts. Step 3-5 are simply paraphrase of the reduction rules (Rules 5.1-5.2), and thus justified by Lemma 5.3.

Step 6 of the algorithm is correct because it simply branches on either including or excluding the vertex w in the objective V_1-FVS. Note that after passing steps 3-5, all vertices in the set V_1 have degree at least 3, and after passing steps 3-6, each vertex in the set V_1 either is a nice V_1-vertex or has at least one neighbor in V_1. In particular, after steps 3-6, if a leaf v in $G[V_1]$ is not a nice V_1-vertex, then v has exactly two neighbors in V_2 that belong to two different connected components of $G[V_2]$. Now consider step 7. As remarked above (also noting step 2), at this point there must be a tree with more than one vertex in the induced subgraph $G[V_1]$. Therefore, we can always find a lowest parent w in a tree in $G[V_1]$. Step 7 branches on this lowest parent w. In case w is included in the objective V_1-FVS, w is deleted from the graph, and the parameter k is decreased by 1. Note that after the vertex w is deleted, the child v of w becomes of degree 2 with its two neighbors in two different connected components of $G[V_2]$. By Lemma 5.3, the vertex v can be excluded from the objective V_1-FVS. Thus, it is safe to move the vertex v from V_1 to V_2. This verifies the correctness of steps 7.1-7.2. Step 7.3 is simply to exclude the vertex w from the objective V_1-FVS.

Observe that before making recursive calls, each of the steps 3-7 decreases the number of vertices in the set V_1 by at least 1. Therefore, the algorithm must terminate in a finite number of steps. Summarizing all the above discussion, we conclude with the correctness of the algorithm **Feedback**(G, V_1, V_2, k). □

On the running time of the algorithm **Feedback**, I interpret the recursive execution of the algorithm as a search tree $\mathcal{T}$, and analyze its complexity by counting the number of leaves in it. The measure used in the analysis is defined as $\mu = 2(k-p)+\tau_2$, and let $T(\mu)$ be the number of leaves in the search tree $\mathcal{T}$ for the algorithm on the input (G, V_1, V_2, k).

Theorem 5.13. *The algorithm* **Feedback**(G, V_1, V_2, k) *correctly solves the disjoint-FVS problem in time* $\mathcal{O}^*(2^{k+\tau_2/2})$*, where* τ_2 *is the number of connected components in the induced subgraph* $G[V_2]$*.*

Proof. The correctness of the algorithm is given by Lemma 5.12. Therefore, it suffices to analyze the complexity of the algorithm. In particular, we consider the value $T(\mu)$.

Each of the steps 1-5 of the algorithm proceeds without branching. However, we must be careful to verify that these steps do not *increase* the value of the measure μ. Step 3 does not change the values of k, p, and τ_2, thus neither that of μ. Step 4 does not changes the value τ_2, but decreases the value k by 1. Moreover, step 4 *may* also decrease the value p by at most 1 (in case the vertex w is a nice V_1-vertex). Overall, step 4 does not increase the value $\mu = 2(k-p)+\tau_2$. Step 5 does not change the value of k. Moreover, it will never decrease the value of p or increase the value of τ_2. Note that step 5 may increase the value of p (e.g., a neighbor of w in V_1 may become a nice V_1-vertex after smoothening w) or decrease the value of τ_2 (e.g., when the two neighbors of w are in two different connected components in $G[V_2]$). In any case, step 5 does not increase the value $\mu = 2(k-p)+\tau_2$.

Now we study the branching steps. First consider step 6. The branch of steps 6.1-6.2 decreases the value k by 1 and does not change the value of τ_2. Moreover, the steps may increase the value of p (e.g., a neighbor of w in V_1 may become a nice V_1-vertex after deleting w from the graph) but will never decrease the value of p.

Therefore, the branch of steps 6.1-6.2 will decrease the value $\mu = 2(k - p) + \tau_2$ by at least 2. On the other hand, because w has at least three neighbors in V_2, step 6.3 will decrease the value of τ_2 by at least 2, while neither changing the value of k nor decreasing the value of p. Thus, step 6.3 also decreases the value $\mu = 2(k - p) + \tau_2$ by at least 2. In summary, if step 6 is executed in the algorithm, then the function $T(\mu)$ satisfies the recurrence relation $T(\mu) \leqslant 2T(\mu - 2)$.

Similarly, the branch of steps 7.1-7.2 deletes the vertex w from the graph and decreases the value of k by 1. As aforementioned, since the algorithm has passes steps 3-6, the leaf v has exactly three neighbors: one is w and the other two are in two different connected components in $G[V_2]$. Therefore, after deleting w from the graph, moving the degree-2 vertex v from set V_1 to set V_2 decreases the value of τ_2 by 1. Also note that in this branch, the value of p is not changed (because of step 6, the vertex w cannot have a neighbor that is a leaf in $G[V_1]$ but has three neighbors in V_2). In summary, the branch of steps 7.1-7.2 decreases the value $\mu = 2(k-p)+\tau_2$ by at least 3. Now consider step 7.3 that moves the vertex w from set V_1 to set V_2. I break this case into two subcases:

Subcase 7.3.1. The vertex w has at least one neighbor in V_2. Then moving w from V_1 to V_2 neither changes the value of k nor increases the value of τ_2. On the other hand, it creates at least one new nice V_1-vertex (i.e., the vertex v) thus increases the value of p by at least 1. Therefore, in this subcase, step 7.3 increases the value of $\mu = 2(k - p) + \tau_2$ by at least 2.

Subcase 7.3.2. The vertex w has no neighbor in V_2. Because the degree of w is larger than 2 and w is a lowest parent in $G[V_1]$, w has at least two children in V_1, each is a leaf in $G[V_1]$ with exactly two neighbors that are in two different connected components of $G[V_2]$. Note that after moving w from V_1 to V_2, all children of w in $G[V_1]$ will become nice V_1-vertices. Therefore, moving w from V_1 to V_2 increases the value of τ_2 by 1, and increases the value of p by at least 2, with the value of k unchanged. Therefore, in this subcase, step 7.3 increases the value of $\mu = 2(k-p)+\tau_2$ by at least 3.

The conclusion from the above discussion is: If step 7 is executed in the algorithm, then the function $T(\mu)$ satisfies the recurrence relation $T(\mu) \leqslant T(\mu-2) + T(\mu-3)$.

Therefore, the function $T(\mu)$, which is the number of leaves in the search tree $\mathcal{T}$, in the worst case satisfies the recurrence relation $T(\mu) \leqslant 2T(\mu-2)$. Also note that Lemma 5.11, if $\mu = 2(k-p)+\tau_2 \leqslant 0$, then it can be concluded immediately without branching that the input instance is a 'No'. Therefore, $T(\mu) = 1$ for $\mu \leqslant 0$. Now the recurrence relation $T(\mu) \leqslant 2T(\mu-2)$ with $T(\mu) = 1$ for $\mu \leqslant 0$ can be solved using the well-known techniques in parameterized computation (see, for example, [85]), as follows. The characteristic polynomial for the recurrence relation $T(\mu) = 2T(\mu-2)$ is x^2-2, which has a unique positive root $\sqrt{2}$. From this, we derive $T(\mu) = (\sqrt{2})^{\mu} = 2^{\mu/2}$. Moreover, it is fairly easy to see that each computational path in the search tree $\mathcal{T}$ has its time bounded by $\mathcal{O}(n^2 \log^6 n)$, and $\mu/2 = k-p+\tau_2/2 \leqslant k+\tau_2/2$. Therefore, the running time of the algorithm **Feedback**(G, V_1, V_2, k) is $\mathcal{O}(2^{k+\tau_2/2} n^2 \log^6 n)$ $\square$

5.4 Concluding Result: An Improved Algorithm for FVS

The results in previous sections lead to an improved algorithm for the general FVS problem. Following the idea of *iterative compression* proposed by Reed et al. [170], the following problem is formulated:

> FVS reduction: given a graph G and an FVS F of size $k+1$ for G, either construct an FVS of size bounded by k for G, or report that no such an FVS exists.

Lemma 5.14. *The FVS reduction problem can be solved in time $\mathcal{O}^*(3.83^k)$.*

Proof. The proof goes similar to that for Lemma 2 in [3]. Let $G = (V, E)$ be a graph and let F_{k+1} be an FVS of size $k + 1$ in G. Suppose that the graph G has an FVS F'_k of size k, and let the intersection $F_{k+1} \cap F'_k$ be a set F_{k-j} of $k - j$ vertices, for some j, $0 \leqslant j \leqslant k$. Let $F_{j+1} = F_{k+1} \setminus F_{k-j}$ and $F'_j = F'_k \setminus F_{k-j}$. Construct the graph $G' = G - F_{k-j}$. Note that both F_{j+1} and F'_j are FVS for G', and that F_{j+1} and F'_j are disjoint. Thus, if we let $V'_1 = V \setminus F_{k+1}$ and $V'_2 = F_{j+1}$, then F'_j is a solution to the instance (G', V'_1, V'_2, j) of the disjoint-fvs problem. On the other hand, it is also easy to see that any solution to the instance (G', V'_1, V'_2, j) of disjoint-FVS plus the subset F_{k-j} makes an FVS of no more than k vertices for the original graph G.

Therefore, to solve the instance (G, F_{k+1}) for the FVS reduction problem, it suffices to find the subset $F_{k-j} = F_{k+1} \cap F'_k$ of $k - j$ vertices in F_{k+1} for some integer j, $0 \leqslant j \leqslant k$, then to solve the instance (G', V'_1, V'_2, j) for the disjoint-FVS problem. To find the subset F_{k-j} of F_{k+1}, we enumerate all subsets of $k - j$ vertices in F_{k+1} for all $0 \leqslant j \leqslant k$. To solve the corresponding instance (G', V'_1, V'_2, j) for disjoint-FVS derived from the subset F_{k-j} of F_{k+1}, we call the algorithm **Feedback**(G', V'_1, V'_2, j). By Theorem 5.13 (note that $\tau_2 \leqslant |V'_2| = j+1$), the instance (G', V'_1, V'_2, j) for disjoint-FVS can be solved in time $\mathcal{O}^*(2^{j+(j+1)/2}) = \mathcal{O}^*(2.83^j)$. Applying this procedure for every integer j $(0 \leqslant j \leqslant k)$ and all subsets of size $k - j$ in F_{k+1} will successfully find an FVS of size k in the graph G, if such an FVS exists. This algorithm solves the FVS reduction problem in time $\sum_{j=0}^{k} \binom{k+1}{k-j} \cdot \mathcal{O}^*(2.83^j) = \mathcal{O}^*(3.83^k)$. □

Finally, by combining Lemma 5.14 with the iterative compression techniques [51], I obtain the main result of this chapter.

Proof of Theorem 5.2. To determine if a given graph $G = (V, E)$ has an FVS of size bounded by k, we start by applying the polynomial-time approximation algorithm of approximation ratio 2 for the MINIMUM FEEDBACK VERTEX SET problem [15]. This algorithm runs in $\mathcal{O}(n^2)$ time, and either returns an FVS F' of size at most $2k$, or verifies that no FVS of size bounded by k exists. Thus, if no FVS is returned by the algorithm, then no FVS of size bounded by k exists. In the case of the opposite result, we use any subset V' of k vertices in F', and put $V_0 = V' \cup (V \setminus F')$. Obviously, the induced subgraph $G[V_0]$ has an FVS V' of size k. Let $F' \setminus V' = \{v_1, v_2, \ldots, v_{|F'|-k}\}$, and let $V_i = V_0 \cup \{v_1, \ldots, v_i\}$ for $i \in \{0, 1, \ldots, |F'| - k\}$. Inductively, suppose that we have constructed an FVS F_i for the graph $G[V_i]$, where $|F_i| = k$. Then the set $F'_{i+1} = F_i \cup \{v_{i+1}\}$ is an FVS for the graph $G[V_{i+1}]$, and $|F'_{i+1}| = k + 1$.

Now the pair $(G[V_{i+1}], F'_{i+1})$ is an instance for the FVS REDUCTION problem. Therefore, in time $\mathcal{O}^*(3.83^k)$, we can either construct an FVS F_{i+1} of size k for the graph $G[V_{i+1}]$, or report that no such an FVS exists. Note that if the graph $G[V_{i+1}]$ does not have an FVS of size k, then the original graph G cannot have an FVS of size k. In this case, we simply stop and claim the non-existence of an FVS of size k for the original graph G. On the other hand, with an FVS F_{i+1} of size k for the graph $G[V_{i+1}]$, my induction proceeds to the next graph $G[V_{i+1}]$, until we reach the graph $G = G[V_{|F'|-k}]$. This process runs in time $k \cdot \mathcal{O}^*(3.83^k) = \mathcal{O}^*(3.83^k)$ since $|F'| - k \leqslant k$, and solves the FVS problem. $\square$

Theorem 5.2 significantly improves the previous best parameterized algorithm whose running time is bounded by $\mathcal{O}^*(5^k)$ for the FVS problem [51].

6. SUMMARY AND FUTURE RESEARCH

This chapter concludes this dissertation and provide some hints on the future work. My research will continue to be focused on the problems enlisted in Figure 1.2 on Page 12. There are two projects, among others, intriguing me the most. The first one studies the parameterized complexity of the multiway cut problem, emphasizing whether it admits a polynomial kernel or not, which is still open. The second one is not about the kernelization, but the approximation algorithm, and aims for an approximation algorithm for the correlation clustering problem with ratio 2. Moreover, possible applications of new techniques reported in this dissertation, as well as other related problems are also briefly discussed.

6.1 Dissertation Summary

This dissertation studies three problems from the families of clustering and feedback set problems, and develops a very small kernel for each of the problems.

Deviating from previous work based on crown reduction and modular decomposition, this dissertation starts from the relationship between my kernelization algorithm for the correlation clustering problem and graph edge-cuts, which are natural upper bounds for the editing costs on the borders. The only reduction rule is a formalization of an extremely simple observation: a densely connected subgraph with loose connection to others should be a cluster. An easily verified condition, based on edge-cuts, is given to identify such subgraphs. The conceptual simplicity not only enables my algorithm to surpass all previous work theoretically, it also makes the implementation extraordinarily easy.

To reveal the power of the edge-cuts based approach, I further try it on the hierarchical clustering problem, which generalizes the correlation clustering problem. It immediately yields a $4k$-element kernel, significantly improves previous work by doing away with the multiplicative factor from the kernel size. Inspired by this result, I also study the parameterized complexity of the hierarchical clustering problem, and manage to show its equivalency with the correlation clustering problem.

Without new idea on reduction rules involved, though new insights are required to show the applicability of previous rules designed for FVS problem, my $3k$ kernel for disjoint-FVS problem is a result of a brand-new way to do kernel size analysis. This approaches starts from an adapted branch-and-conquer algorithm, which is followed by a extremely straightforward way to count the kernel size. When we are designing a branch-and-conquer algorithm, it is always preferred to dispose of as many elements (vertices and/or edges) as possible in each branching step. Here in my adapted version, I manage to keep the influenced elements minimized, while keeping some properties invariant during the branching steps. Now suppose that the adapted algorithm can find all solution of size at most k in at most $p(k)$ steps, while at each step at most $q(k)$ elements are manipulated, then I can easily say that to be a "yes" instance, the instance cannot be larger than $p(k)q(k)$. The algorithm implies a polynomial kernel! For the disjoint-FVS problem, $p(k) = 2k$ and $p(k) = 1.5$, which gives the claimed bound. The another benefit of this approach to bound kernel sizes is the kernel size should be tight, and actually, it is easily to derive a instance from this algorithm such that it has exactly the size of the bound On the algorithm part, other than presenting a polynomial time algorithm for the 3-regular special case, I also give a proof on the NP-hardness of the cases whose maximum degree is 4. This complexity dichotomy shows a clear picture and provides a base for further study.

6.2 Parameterized Complexity of Multiway Cut

The edge version of the multiway cut problem, also called multiterminal cut, is formally defined as:

> given an undirected graph $G = (V, E)$, a weight function $wt : E \to \mathbb{N}$, a set $T \subseteq V$ of terminals and an nonnegative integer k, find a subset S of edges with total weight at most k, such that after the removal of S, no two terminals are in the same connected component.

The following fact was first observed by Ford and Fulkerson [133], as a corollary of the classic max-flow-min-cut theorem, and later rediscovered several times by different authors.

Lemma 6.1. *Let $G = (V, E)$ be an edge-weighted graph, and $s, t \in V$ be two distinct vertices, there is a minimum s-t cut X such that all other minimum s-t cuts are subsets of X.*

Indeed, any algorithm for max-flow can locate such a cut as byproduct, and therefore it is polynomially foundable (though the original algorithm given in [133] is not polynomially bounded). In the following I will denote such a cut by *max-volume min-cut*, where the volume means the number of vertices in X.

One main concept behind most previous work on this problem is the *isolating cut* defined by Dahlhaus et al. [61], which is a cut separating a terminal from the rest. To simplify the presentation, we will abbreviate an isolating cut for terminal t_i (the i-th terminal in T) as a t_i *cut*, and similarly, a minimum weight isolating cut for a terminal t_i as a *min* t_i *cut*. Note that, for a min t_i cut, the vertex set containing t_i induces one single component, which will be used to represent this cut. In general a t_i cut is not required to separate any other pair of terminals, however, it is possible. For example, in the graph consisting of three vertices and two edges, if all three vertices are terminals, the only isolating cut for the degree-2 vertex happens to also separate the other two terminals.

The most important new concept here is the distance[1] defined as follows:

Definition [Distance] The *distance* from a non-terminal vertex ν to the terminal t_i, denote by $d_i(\nu)$, is the increment of min t cut size after merging ν into t_i. That is

$$d_i(\nu) = \gamma(X_{t_i \oplus \nu}) - \gamma(t_i),$$

where $X_{t_i \oplus \nu}$ is the max-volume min t_i cut after merging ν into t_i, which could be equivalently seen as the max-volume min $\{t_i, \nu\}$-$T \backslash t_i$ cut.

Note that for any terminal t_i, the distance function $d_i(\cdot)$ is defined on all non-terminal vertices, not limited to neighbors of t_i, and $d_i(\cdot)$ is always positive. Obviously, the subgraph induced by $X_{t_i \oplus \nu}$ has to be connected when ν is a neighbor of t_i. This is not true in general when ν is adjacent to t_i. The following lemma implies that t_i cannot be a cut point of subgraph induced by $X_{t_i \oplus \nu}$, and more strongly, if it is connected, it contains exactly two components, one being t_i.

Lemma 6.2. *the subgraph induced by* $X_{t_i \oplus \nu} - t_i$ *is connected.*

Proof. I prove by contradiction. Assume $G[X_{t_i \oplus \nu} - t_i]$ is not connected, I can parition $X = X_{t_i \oplus \nu} - t_i$ into X_1 and X_2 such that $\langle X_1, X_2 \rangle = \emptyset$. Without loss of generality, let $\nu \in X_1$. Then X_2 has to be connected to t_i, otherwise $(X_1 + t_i)$ is a $\{t_i, \nu\}$-$T \setminus t_i$ cut, whose edges is a proper subset of $X_{t_i \oplus \nu}$, which contradicts the minimality of $X_{t_i \oplus \nu}$.

[1]In spite the risk of confusion, I decide to use (redefine) the concept "distance", and I believe this will become clearer with the progress with our explanation. Moreover, please be noted that the other (traditional graph-theoretic) meaning of "distance", the length of the shortest path, is never used in this section.

Noting that t_i is the only min t_i cut, this facts implies $\gamma(X_2 + t_i) > \gamma(t_i)$. Since X_1 and X_2 are disconnected, I have

$$\gamma(X_{t_i \oplus v}) - \gamma(t_i) = \gamma(X_1 + t_i) - \gamma(t_i) + \gamma(X_2 + t_i) - \gamma(t_i) > \gamma(X_1 + t_i) - \gamma(t_i),$$

which means $(X_1 + t_i)$ is a smaller $\{t_i, v\}$-$T \backslash t_i$ cut than $X_{t_i \oplus v}$, and is a contradiction.

□

Some other self-explanatory facts on the distances include:

Lemma 6.3. *Let* $t_i \in T$ *be a terminal, and* $u, v \in V \backslash T$ *be two distinct vertices, then*

I $d_i(v) \leqslant d_i(u)$ *for any* $v \in X_{t_i \oplus u} - t_i$*;*

II if $d_i(u) = \min_{x \in V \backslash T} d_i(x)$*, then* $d_i(u) = d_i(v)$ *for any vertex* $v \in X_{t_i \oplus u} - t_i$*;*

III $X_{t_i \oplus u}$ *and* $X_{t_i \oplus v}$ *coincide only if* $d_i(u) = d_i(v)$*.*

IV $X_{t_i \oplus u}$ *properly contatins* $X_{t_i \oplus v}$ *only if* $d_i(u) > d_i(v)$*.*

In particular, I show that those vertices with distance 1 to a terminal can be grouped.

Lemma 6.4. *Let* (G, w, T, k) *be an instance of multiterminal cut problem where all max-volume minimum weight isolating cuts have been shrinked. If the distance from a vertex* u *to terminal* t_i *is* 1*, then there exists an optimal solution which keeps* $(X_{t_i \oplus u} - t_i)$ *together.*

With the lemma, the branching process can be redesigned to dispose of the vertices of distance 1 in a different way, such that a better branching vector is obtained, which immediately implies a improved algorithm:

Theorem 6.5. *The multiterminal cut problem can be solved in time* $\mathcal{O}(1.84^k)$*.*

which can be even better for small number of terminals, and in particular, for the 3-terminal case:

Corollary 6.6. *The* 3*-terminal cut problem can be solved in time* $\mathcal{O}(1.3563^k)$.

The observation sheds light on the kernelization is: similar to terminals, isolating cuts can also be defined on non-terminal vertices, which separate a non-terminal vertex to all terminals. Most properties of minimum isolating cuts also apply here, and in particular:

Lemma 6.7. *Let* (G, w, T, k) *be an instance of multiterminal cut problem, and* $v \in V \backslash T$ *be a non-terminal vertex. Then there is an optimal solution* S *which keeps max-volume min* v-T *cut* X *together.*

6.3 Approximation of Correlation Clustering

I starts from two simple lemmas.

Lemma 6.8. *There is an optimal solution to make a graph* G *into a single cluster, if and only if each cut is non-sparse.*

Lemma 6.9. *Let* S *be an optimal solution, and* V' *be a clique in the objective graph* $G \triangle S$*, then diameter of* $G[V']$ *is at most 2.*

The equivalency between fuzzy clustering problem and edge multicut problem has been observed and proved by Demaine et al. [67]. However, the proof presented there is flawed. Specifically, the proof of Lemma 4.6 in [67] only showes that all *original* erroneous cycles are broken, while this does not suffice for new erroneous cycles might be introduced during operations. This can be easily explinaed as: any superset of a multicut must be a feasible multicut, but this does *not* hold for clutering. This bug is fixed as follows, and one should note a non-minimal solution of the multicut problem cannot be transformed back. (Reduction from correlation clustering to multicut is omitted here, and interested reader is referred to [67] for the details.)

Lemma 6.10. *Let* G *be an instance of clustering problem, and* $\mathsf{G}', \mathcal{P}$ *be the instance of multict transfomed from* G*. If* E *is a solution to* $\mathsf{G}', \mathcal{P}$*, then there is a subset of* $\phi^{-1}(\mathsf{E})$ *which is a solution of* G*.*

Proof. To show there is no erroneous cycle, it suffices to consider missed edges - either original or introduced by removing edges - since each such a cycle must contain one missed edge. Let uv is a missed edge in the resutled graph, and if it is also missed in the original graph, then each path from u to v in the resulted graph must contain an introduced edge originally non-existent. Let E_1 be those added edges lying in the paths from u to v, $\mathsf{E} - \mathsf{E}_1$ is also a multicut. □

On the linear program formulated in [67], when restricted to the correlation clustering problem, I conjecture that there is always an optimal solution which assigns only integral values (0 or 1) to missed edges, and half-integral values (0, 1/2, or 1) to edges. Then an approximation algorithm of ratio 2 following immediately.

6.4 Other Possible Directions

There are several interesting questions related to the new approaches and the problems studied in this dissertation: further study of this new approach, randomized and algebraic algorithms, and kernelization.

Other applications of the new kernel size analysis technique. The main new ingredient of my $4k$ kernel for the disjoint-FVS problem is the new technique for analysis. As a matter of fact, my $2k$-vertex kernel for the correlation clustering problem can also be analyzed with this technique. It should be interesting to apply this analysis technique to kernelization of other problems, with or without new reduction rules introduced. The first should be those with good branch-and-search algorithms.

Kernelization algorithms for correlation clustering. In Chapter 3, the subgraphs used in the kernelization algorithm are very special, which are only those closed neighborhoods of vertices, however, the observation is general and applicable for any densely connected subgraph. Therefore, it might be possible to build a smaller kernel is trying some more complicated subgraphs. Its reducible condition should be straightforward, while the trouble lies in how to identify them. The more intriguing question is, can a kernel with ck edges be achieved, where c is some constant. This is a linear kernel, which, if possible, should be the first result of this kind. In my linear-vertex kernel, there can still be quadratic number of edges, and this fact is also observed in the kernels of other edge modification problems.

Kernelization of generalizations of problems studied in this dissertation. The problems studied in this dissertation are very important and have many variations and generalizations, which are usually "harder"[2] than themselves. The most widely studied ones include: the fuzzy clustering problems, and the DFVS problem, which are a generalization of the correlation clustering problem and a variation of the FVS problem respectively. After long time of research, both of them were shown to be in FPT very recently, however, both are open for the existence of polynomial kernels. As a final remark, the fuzzy clustering problem is known to be computa-

[2]In an informal and intuitive sense, and common evidences include known approximation lower bounds and parameterized lower bounds.

tionally equivalent to multicut problem [67], which includes the FVS problem as a special case.

REFERENCES

[1] K. R. Abrahamson, R. G. Downey, and M. R. Fellows, *Fixed-parameter tractability and completeness IV: On completeness for W[P] and PSPACE analogues*, Annals of Pure and Applied Logic, 73 (1995), pp. 235–276.

[2] F. N. Abu-Khzam, M. R. Fellows, M. A. Langston, and W. H. Suters, *Crown structures for vertex cover kernelization*, Theory of Computing Systems, 41 (2007), pp. 411–430.

[3] R. Agarwala, V. Bafna, M. Farach, M. Paterson, and M. Thorup, *On the approximability of numerical taxonomy (fitting distances by tree metrics)*, SIAM Journal on Computing, 28 (1999), pp. 1073–1085.

[4] N. Ailon, *Studies in Algorithms*, Princeton University, Princeton, NJ, USA, 2006.

[5] N. Ailon and M. Charikar, *Fitting tree metrics: Hierarchical clustering and phylogeny*, SIAM Journal on Computing, 40 (2011), pp. 1275–1291.

[6] N. Ailon, M. Charikar, and A. Newman, *Aggregating inconsistent information: Ranking and clustering*, Journal of the ACM, 55 (2008), pp. (Article 23) 1–27.

[7] J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, and R. Niedermeier, *Fixed parameter algorithms for* DOMINATING SET *and related problems on planar graphs*, Algorithmica, 33 (2002), pp. 461–493.

[8] J. Alber, F. Dorn, and R. Niedermeier, *Experimental evaluation of a tree decomposition-based algorithm for vertex cover on planar graphs*, Discrete Applied Mathematics, 145 (2005), pp. 219–231.

[9] J. Alber, M. R. Fellows, and R. Niedermeier, *Polynomial-time data reduction for dominating set*, Journal of the ACM, 51 (2004), pp. 363–384.

[10] J. Alber, H. Fernau, and R. Niedermeier, *Parameterized complexity: Exponential speed-up for planar graph problems*, Journal of Algorithms, 52 (2004), pp. 26–56.

[11] S. Albers and P. Weil, eds., *STACS 2008, 25th Annual Symposium on Theoretical Aspects of Computer Science, Bordeaux, France, February 21-23, 2008, Proceedings*, vol. 1 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2008.

[12] N. Alon, G. Gutin, E. J. Kim, S. Szeider, and A. Yeo, *Solving MAX-r-SAT above a tight lower bound*, in Charikar [44], pp. 511–517.

[13] N. Alon, D. Lokshtanov, and S. Saurabh, *Fast FAST*, in Automata, Languages and Programming (ICALP), S. Albers, A. Marchetti-Spaccamela, Y. Matias, S. E. Nikoletseas, and W. Thomas, eds., vol. 5555 of LNCS, Berlin Heidelberg, 2009, Springer-Verlag, pp. 49–58.

[14] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*, Cambridge University Press, New York, NY, USA, 2009.

[15] V. Bafna, P. Berman, and T. Fujito, *A 2-approximation algorithm for the undirected feedback vertex set problem*, SIAM Journal on Discrete Mathematics, 12 (1999), pp. 289–297.

[16] B. S. Baker, *Approximation algorithms for NP-complete problems on planar graphs*, Journal of the ACM, 41 (1994), pp. 153–180.

[17] J. Bang-Jensen and G. Gutin, *Digraphs: Theory, Algorithms and Applications*, Springer Monographs in Mathematics, Springer, London, 2 ed., 2009.

[18] N. Bansal, A. Blum, and S. Chawla, *Correlation clustering*, Machine Learning, 56 (2004), pp. 89–113.

[19] A. Becker, R. Bar-Yehuda, and D. Geiger, *Randomized algorithms for the loop cutset problem*, Journal of Artificial Intelligence Research, 12 (2000), pp. 219–234.

[20] R. Bellman, *Dynamic programming treatment of the travelling salesman problem*, Journal of the ACM, 9 (1962), pp. 61–63.

[21] A. Blum, T. Jiang, M. Li, J. Tromp, and M. Yannakakis, *Linear approximation of shortest superstrings*, Journal of the ACM, 41 (1994), pp. 630–647.

[22] S. Böcker, *A golden ratio parameterized algorithm for cluster editing*, in International Workshop On Combinatorial Algorithms (IWOCA), C. S. Iliopoulos and W. F. Smyth, eds., vol. 7056 of Lecture Notes in Computer Science, Springer, 2011, pp. 85–95.

[23] S. Böcker, S. Briesemeister, Q. B. A. Bui, and A. Truss, *Going weighted: parameterized algorithms for cluster editing*, Theoretical Computer Science, 410 (2009), pp. 5467–5480.

[24] S. Böcker, S. Briesemeister, and G. W. Klau, *Exact algorithms for cluster editing: Evaluation and experiments*, Algorithmica, (2009).

[25] H. Bodlaender and T. van Dijk, *A cubic kernel for feedback vertex set and loop cutset*, Theory of Computing Systems, 46 (2010), pp. 566–597.

[26] H. L. Bodlaender, *On disjoint cycles*, International Journal of Foundation of Computer Science, 5 (1994), pp. 59–68.

[27] ——, *Kernelization: New upper and lower bound techniques*, in International Workshop on Parameterized and Exact Computation (IWPEC), J. Chen and F. V. Fomin, eds., vol. 5917 of LNCS, Springer, 2009, pp. 17–37.

[28] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin, *On problems without polynomial kernels*, Journal of Computer and System Sciences, 75 (2009), pp. 423–434.

[29] H. L. Bodlaender, F. V. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, and D. M. Thilikos, *(Meta) Kernelization*, CoRR, abs/0904.0727 (2009).

[30] H. L. Bodlaender and M. A. Langston, eds., *Parameterized and Exact Computation, Second International Workshop, IWPEC 2006, Zürich, Switzerland, September 13-15, 2006, Proceedings*, vol. 4169 of Lecture Notes in Computer Science, Springer, 2006.

[31] H. L. Bodlaender, E. Penninkx, and R. B. Tan, *A linear kernel for the k-disjoint cycle problem on planar graphs*, in ISAAC, S.-H. Hong, H. Nagamochi, and T. Fukunaga, eds., vol. 5369 of Lecture Notes in Computer Science, Springer, 2008, pp. 306–317.

[32] J. A. Bondy and U. S. R. Murty, *Graph Theory*, vol. 244 of Graduate Texts in Mathematics, Springer, 2008.

[33] N. Bourgeois, B. Escoffier, and V. T. Paschos, *Approximation of max independent set, min vertex cover and related problems by moderately exponential algorithms*, Discrete Applied Mathematics, 159 (2011), pp. 1954–1970.

[34] N. Bousquet, J. Daligault, and S. Thomassé, *Multicut is FPT*, CoRR, abs/1010.5197 (2010).

[35] A. Buchsbaum, ed., *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, SIAM, 2005.

[36] K. Burrage, V. Estivill-Castro, M. R. Fellows, M. A. Langston, S. Mac, and F. A. Rosamond, *The undirected feedback vertex set problem has a Poly(k) kernel*, in Bodlaender and Langston [30], pp. 192–202.

[37] J. F. Buss and J. Goldsmith, *Nondeterminism within P*, SIAM Journal on Computing, 22 (1993), pp. 560–572.

[38] L. Cai and J. Chen, *On fixed-parameter tractability and approximability of NP optimization problems*, Journal of Computer and System Sciences, 54 (1997), pp. 465–474.

[39] L. Cai, J. Chen, R. G. Downey, and M. R. Fellows, *On the structure of parameterized problems in NP*, Information and Computation, 123 (1995), pp. 38–49.

[40] ——, *Advice classes of parameterized tractability*, Annals of Pure and Applied Logic, 84 (1997), pp. 119–138.

[41] Y. Cao and J. Chen, *Cluster editing: Kernelization based on edge cuts*, in IPEC, V. Raman and S. Saurabh, eds., vol. 6478 of LNCS, Berlin Heidelberg, 2010, Springer-Verlag, pp. 60–71.

[42] ——, *Cluster editing: kernelization based on edge cuts*, Algorithmica, (2011). doi: 10.1007/s00453-011-9595-1.

[43] Y. Cao, J. Chen, and Y. Liu, *On feedback vertex set: New measure and new structures*, in 12th Scandinavian Symposium and Workshops on Algorithm Theory, H. Kaplan, ed., vol. 6139 of LNCS, Berlin Heidelberg, 2010, Springer-Verlag, pp. 93–104.

[44] M. Charikar, ed., *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, SIAM, 2010.

[45] M. Charikar, V. Guruswami, and A. Wirth, *Clustering with qualitative information*, Journal of Computer and System Sciences, 71 (2005), pp. 360–383.

[46] B. Chazelle, *A minimum spanning tree algorithm with inverse-ackermann type complexity*, Journal of the ACM, 47 (2000), pp. 1028–1047.

[47] J. Chen, *Minimum and maximum imbeddings*, in The Handbook of Graph Theory, J. Gross and J. Yellen, eds., LNCS, CRC Press, 2003, pp. 625–641.

[48] ——, *Vertex cover kernelization*, in Kao [138].

[49] ——, *Vertex cover search trees*, in Kao [138].

[50] J. Chen, H. Fernau, I. A. Kanj, and G. Xia, *Parametric duality and kernelization: Lower bounds and upper bounds on kernel size*, SIAM Journal on Computing, 37 (2007), pp. 1077–1106.

[51] J. Chen, F. V. Fomin, Y. Liu, S. Lu, and Y. Villanger, *Improved algorithms for feedback vertex set problems*, Journal of Computer and System Sciences, 74 (2008), pp. 1188–1198.

[52] J. Chen, D. K. Friesen, and H. Zheng, *Tight bound on Johnson's algorithm for Maximum Satisfiability*, Journal of Computer and System Sciences, 58 (1999), pp. 622–640.

[53] J. Chen, X. Huang, I. A. Kanj, and G. Xia, *Polynomial time approximation schemes and parameterized complexity*, Discrete Applied Mathematics, 155 (2007), pp. 180–193.

[54] J. Chen, I. A. Kanj, and W. Jia, *Vertex cover: Further observations and further improvements*, Journal of Algorithms, 41 (2001), pp. 280–301.

[55] J. Chen, I. A. Kanj, and G. Xia, *Labeled search trees and amortized analysis: Improved upper bounds for NP-hard problems*, Algorithmica, 43 (2005), pp. 245–273.

[56] ——, *Improved upper bounds for vertex cover*, Theoretical Computer Science, 411 (2010), pp. 3736–3756.

[57] J. Chen, Y. Liu, S. Lu, B. O'Sullivan, and I. Razgon, *A fixed-parameter algorithm for the directed feedback vertex set problem*, Journal of the ACM, 55 (2008).

[58] M. Chlebík and J. Chlebíková, *Crown reductions for the minimum weighted vertex cover problem*, Discrete Applied Mathematics, 156 (2008), pp. 292–312.

[59] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 3 ed., 2009.

[60] R. Crowston, G. Gutin, M. Jones, and A. Yeo, *Linear-number-of-variables kernel for unit-conflict-free-max-sat parameterized above expectation*, CoRR, abs/1004.0526 (2010).

[61] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis, *The complexity of multiterminal cuts*, SIAM Journal on Computing, 23 (1994), pp. 864–894.

[62] G. B. Dantzig, *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ, 1963.

[63] F. de Montgolfier, *Décomposition modulaire des graphes. Théorie,extensions et algorithmes*, Thése de doctorat, Université Montpellier II, Montpellier, Hérault, France, 2003.

[64] F. K. H. A. Dehne, M. R. Fellows, M. A. Langston, F. A. Rosamond, and K. Stevens, *An $O(2^{O(k)}n^3)$ FPT algorithm for the undirected feedback vertex set problem*, Theory of Computing Systems, 41 (2007), pp. 479–492.

[65] F. K. H. A. Dehne, M. A. Langston, X. Luo, S. Pitre, P. Shaw, and Y. Zhang, *The cluster editing problem: Implementations and experiments*, in Bodlaender and Langston [30], pp. 13–24.

[66] H. Dell and D. van Melkebeek, *Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses*, in STOC, L. J. Schulman, ed., ACM, 2010, pp. 251–260.

[67] E. D. Demaine, D. Emanuel, A. Fiat, and N. Immorlica, *Correlation clustering in general weighted graphs*, Theoretical Computer Science, 361 (2006), pp. 172–187.

[68] E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos, *Bidimensional parameters and local treewidth*, SIAM Journal on Discrete Mathematics, 18 (2004), pp. 501–511.

[69] ——, *Fixed-parameter algorithms for (k, r)-center in planar graphs and map graphs*, ACM Transactions on Algorithms, 1 (2005), pp. 33–47.

[70] ——, *Subexponential parameterized algorithms on bounded-genus graphs and h-minor-free graphs*, Journal of the ACM, 52 (2005), pp. 866–893.

[71] E. D. Demaine, M. Hajiaghayi, and D. M. Thilikos, *The bidimensional theory of bounded-genus graphs*, SIAM Journal on Discrete Mathematics, 20 (2006), pp. 357–371.

[72] E. D. Demaine and M. T. Hajiaghayi, *Equivalence of local treewidth and linear local treewidth and its algorithmic applications*, in SODA, J. I. Munro, ed., SIAM, 2004, pp. 840–849.

[73] ——, *Fast algorithms for hard graph problems: Bidimensionality, minors, and local treewidth*, in Graph Drawing, J. Pach, ed., vol. 3383 of LNCS, Springer, 2004, pp. 517–533.

[74] ——, *Bidimensionality: New connections between FPT algorithms and PTASs*, in Buchsbaum [35], pp. 590–601.

[75] ——, *Graphs excluding a fixed minor have grids as large as treewidth, with combinatorial and algorithmic applications through bidimensionality*, in Buchsbaum [35], pp. 682–689.

[76] E. D. Demaine, M. T. Hajiaghayi, and D. M. Thilikos, *Exponential speedup of fixed-parameter algorithms for classes of graphs excluding single-crossing graphs as minors*, Algorithmica, 41 (2005), pp. 245–267.

[77] R. Diestel, *Graph Theory*, vol. 173 of Graduate Texts in Mathematics, Springer-Verlag, Heidelberg, 2010.

[78] I. Dinur and S. Safra, *On the hardness of approximating minimum vertex cover*, Annals of Mathematics, 162 (2005), pp. 439–485.

[79] R. G. Downey and M. R. Fellows, *Fixed parameter intractability*, in proceedings of the 7th Annual Structural Complexity Conference, 1992, pp. 36–49.

[80] ——, *Fixed parameter tractability and completeness*, in Complexity Theory: Current Research, Cambridge University Press, 1992, pp. 191–225.

[81] ——, *Fixed-parameter tractability and completeness III: Some structural aspects of the W hierarchy*, in Complexity theory: current research, Cambridge University Press, New York, NY, USA, 1993, pp. 191–225.

[82] ——, *Fixed-parameter tractability and completeness I: Basic results*, SIAM Journal on Computing, 24 (1995), pp. 873–921.

[83] ——, *Fixed-parameter tractability and completeness II: On completeness for w[1]*, Theoretical Computer Science, 141 (1995), pp. 109–131.

[84] ——, *Parameterized computational feasibility*, in Feasible Mathematics II, Birkhauser, 1995, pp. 219–244.

[85] ——, *Parameterized Complexity*, Springer, 1999.

[86] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*, John Wiley & Sons, New York, 2 ed., 2001.

[87] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, *Rank aggregation methods for the web*, in WWW '01: Proceedings of the 10th international conference on World Wide Web, New York, NY, USA, 2001, ACM, pp. 613–622. a significantly revised but unpublished version in the name "Rank aggregation revisited" is online-only, available from http://www.eecs.harvard.edu/michaelm/CS222/rank2.pdf.

[88] G. Even, J. Naor, B. Schieber, and M. Sudan, *Approximating minimum feedback sets and multicuts in directed graphs*, Algorithmica, 20 (1998), pp. 151–174.

[89] G. Even, J. Naor, B. Schieber, and L. Zosin, *Approximating minimum subset feedback sets in undirected graphs with applications*, SIAM Journal on Discrete Mathematics, 13 (2000), pp. 255–267.

[90] B. S. Everitt, S. Landau, and M. Leese, *Cluster Analysis*, Wiley, 4 ed., 2001.

[91] U. Feige, *Faster FAST (feedback arc set in tournaments)*, CoRR, abs/0911.5094 (2009).

[92] M. R. Fellows, *Blow-ups, win/win's, and crown rules: Some new directions in FPT*, in Graph-Theoretic Concepts in Computer Science, Springer, 2003, pp. 1–12.

[93] M. R. Fellows, M. A. Langston, F. A. Rosamond, and P. Shaw, *Efficient parameterized preprocessing for cluster editing*, in FCT, LNCS, Berlin Heidelberg, 2007, Springer-Verlag, pp. 312–321.

[94] P. Festa, P. M. Pardalos, and M. G. Resende, *Feedback set problems*, in Handbook of Combinatorial Optimization, Supplement Vol. A, Kluwer Acad. Publ., Dordrecht, 1999, pp. 209–258.

[95] J. Flum and M. Grohe, *Parameterized Complexity Theory*, Springer, 2006.

[96] F. V. Fomin, S. Gaspers, A. V. Pyatkin, and I. Razgon, *On the minimum feedback vertex set problem: Exact and enumeration algorithms*, Algorithmica, 52 (2008), pp. 293–307.

[97] F. V. Fomin, F. Grandoni, and D. Kratsch, *Solving connected dominating set faster than* 2^n, Algorithmica, 52 (2008), pp. 153–166.

[98] ——, *A measure & conquer approach for the analysis of exact algorithms*, Journal of the ACM, 56 (2009).

[99] F. V. Fomin and D. Kratsch, *Exact Exponential Algorithms*, Texts in Theoretical Computer Science. An EATCS Series, Springer, 2011.

[100] F. V. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos, *Bidimensionality and kernels*, in Charikar [44], pp. 503–510.

[101] F. V. Fomin and D. M. Thilikos, *A simple and fast approach for solving problems on planar graphs*, in STACS, V. Diekert and M. Habib, eds., vol. 2996 of LNCS, Springer, 2004, pp. 56–67.

[102] ——, *Dominating sets in planar graphs: Branch-width and exponential speed-up*, SIAM Journal on Computing, 36 (2006), pp. 281–309.

[103] L. Fortnow and R. Santhanam, *Infeasibility of instance compression and succinct PCPs for NP*, Journal of Computer and System Sciences, In Press, Corrected Proof (2010).

[104] M. L. Furst, J. L. Gross, and L. A. McGeoch, *Finding a maximum-genus graph imbedding*, Journal of the ACM, 35 (1988), pp. 523–534.

[105] H. N. Gabow and M. Stallmann, *Efficient algorithms for graphic matroid intersection and parity*, in Automata, Languages and Programming: 12th Colloquium, vol. 194 of LNCS, Springer-Verlag, 1985, pp. 210–220.

[106] G. Gan, C. Ma, and J. Wu, *Data Clustering : Theory, Algorithms, and Applications*, ASA-SIAM series on statistics and applied probability, SIAM, 2007.

[107] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[108] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier, *Graph-modeled data clustering: Exact algorithms for clique generation*, Theory of Computing Systems, 38 (2006), pp. 373–392.

[109] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier, *Data reduction and exact algorithms for clique cover*, ACM Journal of Experimental Algorithmics, 13 (2008).

[110] D. H. Greene and D. E. Knuth, *Mathematics for the Analysis of Algorithms*, Modern Birkhäuser Classics, Birkhäuser Boston, 3 ed., 2007. three volumes.

[111] S. Guha, R. Rastogi, and K. Shim, *Rock: A robust clustering algorithm for categorical attributes*, in ICDE '99: Proceedings of the 15th International Conference on Data Engineering, Washington, DC, USA, 1999, IEEE Computer Society, p. 512.

[112] S. Guha, R. Rastogi, and K. Shim, *Cure: An efficient clustering algorithm for large databases*, Information Systems, 26 (2001), pp. 35–58.

[113] J. Guo, *A more effective linear kernelization for cluster editing*, Theoretical Computer Science, 410 (2009), pp. 718–726.

[114] J. Guo, J. Gramm, F. Hüffner, R. Niedermeier, and S. Wernicke, *Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization*, Journal of Computer and System Sciences, 72 (2006), pp. 1386–1396.

[115] J. Guo, S. Hartung, C. Komusiewicz, R. Niedermeier, and J. Uhlmann, *Exact algorithms and experiments for hierarchical tree clustering*, in Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10), M. Fox and D. Poole, eds., Berlin Heidelberg, 2010, Springer-Verlag, pp. 457–462.

[116] J. Guo, H. Moser, and R. Niedermeier, *Iterative compression for exactly solving NP-hard minimization problems*, in Algorithmics of Large and Complex Networks, J. Lerner, D. Wagner, and K. A. Zweig, eds., vol. 5515 of Lecture Notes in Computer Science, Springer, 2009, pp. 65–80.

[117] J. Guo and R. Niedermeier, *Invitation to data reduction and problem kernelization*, SIGACT News, 38 (2007), pp. 31–45.

[118] J. Guo and R. Niedermeier, *Linear problem kernels for NP-hard problems on planar graphs*, in Automata, Languages and Programming (ICALP), L. Arge, C. Cachin, T. Jurdzinski, and A. Tarlecki, eds., vol. 4596 of LNCS, Springer, 2007, pp. 375–386.

[119] J. Guo, R. Niedermeier, and S. Wernicke, *Fixed-parameter tractability results for full-degree spanning tree and its dual*, in Bodlaender and Langston [30], pp. 203–214.

[120] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge University Press, New York, NY, USA, 1997.

[121] G. Gutin, E. J. Kim, M. Mnich, and A. Yeo, *Ordinal embedding relaxations parameterized above tight lower bound*, CoRR, abs/0907.5427 (2009).

[122] ——, *Betweenness parameterized above tight lower bound*, Journal of Computer and System Sciences, 76 (2010), pp. 872–878.

[123] G. Gutin, E. J. Kim, S. Szeider, and A. Yeo, *A probabilistic approach to problems parameterized above or below tight bounds*, in International Workshop on Parameterized and Exact Computation (IWPEC), J. Chen and F. V. Fomin, eds., vol. 5917 of Lecture Notes in Computer Science, Springer, 2009, pp. 234–245.

[124] G. Gutin, A. Rafiey, S. Szeider, and A. Yeo, *The linear arrangement problem parameterized above guaranteed value*, Theory of Computing Systems, 41 (2007), pp. 521–538.

[125] G. Gutin, L. van Iersel, M. Mnich, and A. Yeo, *All ternary permutation constraint satisfaction problems parameterized above average have kernels with quadratic numbers of variables*, in ESA (1), M. de Berg and U. Meyer, eds., vol. 6346 of Lecture Notes in Computer Science, Springer, 2010, pp. 326–337.

[126] G. Gutin and A. Yeo, *Note on maximal bisection above tight lower bound*, CoRR, abs/1005.2848 (2010).

[127] D. S. Hochbaum, *Approximation algorithms for the set covering and vertex cover problems*, SIAM Journal on Computing, 11 (1982), pp. 555–556.

[128] ——, *Approximating covering and packing problems: set cover, vertex cover, independent set and related problems.*, in Approximation algorithms for NP-hard problems, PWS, Boston, 1996, ch. 3.

[129] ——, *The t-vertex cover problem: Extending the half integrality framework with budget constraints*, in Approximation Algorithms for Combinatorial Optimization (APPROX), K. Jansen and D. S. Hochbaum, eds., vol. 1444 of Lecture Notes in Computer Science, Springer, 1998, pp. 111–122.

[130] ——, *Solving integer programs over monotone inequalities in three variables: A framework for half integrality and good approximations*, European Journal of Operational Research, 140 (2002), pp. 291–321.

[131] A. K. Jain, M. N. Murty, and P. J. Flynn, *Data clustering: A review*, ACM Computing Surveys, 31 (1999), pp. 264–323.

[132] D. S. Johnson, *Approximation algorithms for combinatorial problems*, Journal of Computer and System Sciences, 9 (1974), pp. 256–278.

[133] L. R. F. Jr. and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, New Jersey, 1962.

[134] I. A. Kanj, M. J. Pelsmajer, and M. Schaefer, *Parameterized algorithms for feedback vertex set*, in LNCS, vol. 3162, Springer, 2004, pp. 235–247.

[135] I. A. Kanj, M. J. Pelsmajer, G. Xia, and M. Schaefer, *On the induced matching problem*, in Albers and Weil [11], pp. 397–408.

[136] I. A. Kanj and L. Perkovic, *Improved parameterized algorithms for planar dominating set*, in Mathematical Foundations of Computer Science (MFCS), K. Diks and W. Rytter, eds., vol. 2420 of Lecture Notes in Computer Science, Springer, 2002, pp. 399–410.

[137] R. Kannan, S. Vempala, and A. Vetta, *On clusterings: Good, bad and spectral*, Journal of the ACM, 51 (2004), pp. 497–515.

[138] M.-Y. Kao, ed., *Encyclopedia of Algorithms*, Springer, 2008.

[139] R. M. Karp, *Reducibility among combinatorial problems*, in Complexity of Computer Computations, Plenum, 1972, pp. 85–103.

[140] M. Karpinski and W. Schudy, *Faster algorithms for feedback arc set tournament, Kemeny rank aggregation and betweenness tournament*, in ISAAC (1), O. Cheong, K.-Y. Chwa, and K. Park, eds., vol. 6506 of Lecture Notes in Computer Science, Springer, 2010, pp. 3–14.

[141] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley-Interscience, 9 ed., 1990.

[142] S. Khot, *On the power of unique 2-prover 1-round games*, in STOC, J. Reif, ed., ACM, 2002, pp. 767–775.

[143] S. Khot and O. Regev, *Vertex cover might be hard to approximate to within $2-\epsilon$*, Journal of Computer and System Sciences, 74 (2008), pp. 335 – 349.

[144] J. Kleinberg and É. Tardos, *Algorithm Design*, Addison Wesley, 2005.

[145] S. Kratsch, *Polynomial kernelizations for MIN $F^+\Pi_1$ and MAX NP*, in STACS, S. Albers and J.-Y. Marion, eds., vol. 3 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009, pp. 601–612.

[146] O. Kullmann, *New methods for 3-SAT decision and worst-case analysis*, Theoretical Computer Science, 223 (1999), pp. 1–72.

[147] M. Křivánek and J. Morávek, *NP-hard problems in hierarchical-tree clustering*, Acta Informatica, 23 (1986), pp. 311–323.

[148] D. T. Larose, *Discovering Knowledge in Data: An Introduction to Data Mining*, John Wiley & Sons, 2005.

[149] F. T. Leighton and S. Rao, *Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms*, Journal of the ACM, 46 (1999), pp. 787–832.

[150] D. Li and Y. Liu, *A polynomial algorithm for finding the minimul feedback vertex set of a 3-regular simple graph.*, Acta Mathematica Scientia, 19 (1999), pp. 375–381.

[151] R. J. Lipton and R. E. Tarjan, *A separator theorem for planar graphs*, SIAM Journal on Applied Mathematics, 36 (1979), pp. 177–189.

[152] ——, *Applications of a planar separator theorem*, SIAM Journal on Computing, 9 (1980), pp. 615–627.

[153] D. Lokshtanov, M. Mnich, and S. Saurabh, *Linear kernel for planar connected dominating set*, in TAMC, J. Chen and S. B. Cooper, eds., vol. 5532 of Lecture Notes in Computer Science, Springer, 2009, pp. 281–290.

[154] D. Lokshtanov and C. Sloper, *Fixed parameter set splitting, linear kernel and improved running time*, in ACiD, H. Broersma, M. Johnson, and S. Szeider, eds., vol. 4 of Texts in Algorithmics, King's College, London, 2005, pp. 105–113.

[155] L. Lovász, *The matroid matching problem*, in Algebraic Methods in Graph Theory, Colloquia Mathematica Societatis János Bolyai, Szeged(Hungary), 1978.

[156] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, Cambridge University Press, 2008.

[157] D. Marx and I. Razgon, *Constant ratio fixed-parameter approximation of the edge multicut problem*, Information Processing Letters, 109 (2009), pp. 1161–1166.

[158] ——, *Fixed-parameter tractability of multicut parameterized by the size of the cutset*, CoRR, abs/1010.3633 (2010).

[159] J. W. Moon, *Topics in Tournaments*, Holt, Rinehart and Winston, New York, 1968.

[160] ——, *On maximal transitive subtournaments*, Proceedings of the Edinburgh Mathematical Society, 17 (1971), pp. 345–349.

[161] F. Murtagh, *A survey of recent advances in hierachical clustering algorithms*, Computer Journal, 26 (1983), pp. 354–359.

[162] G. L. Nemhauser and L. E. Trotter, *Vertex packings: Structural properties and algorithms*, Mathematical Programming, 8 (1975), pp. 232–248.

[163] E. D. Nering and A. W. Tucker, *Linear Programming and Related Problems*, Academic Press, Boston, MA, 1993.

[164] R. Niedermeier, *Invitation to Fixed-Parameter Algorithms*, Oxford University Press, 2006.

[165] ——, *Reflections on multivariate algorithmics and problem parameterization*, in STACS, J.-Y. Marion and T. Schwentick, eds., vol. 5 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010, pp. 17–32.

[166] R. Niedermeier and P. Rossmanith, *A general method to speed up fixed-parameter-tractable algorithms*, Information Processing Letters, 73 (2000), pp. 125–129.

[167] M. Patrascu and R. Williams, *On the possibility of faster SAT algorithms*, in Charikar [44], pp. 1065–1075.

[168] W. V. Quine, *The problem of simplifying truth functions*, The American Mathematical Monthly, 59 (1952), pp. 521–531.

[169] V. Raman, S. Saurabh, and C. R. Subramanian, *Faster fixed parameter tractable algorithms for undirected feedback vertex set*, in LNCS, vol. 2518, Springer, 2002, pp. 241–248.

[170] B. A. Reed, K. Smith, and A. Vetta, *Finding odd cycle transversals*, Operations Research Letters, 32 (2004), pp. 299–301.

[171] R. Rizzi, *Minimum weakly fundamental cycle bases are hard to find*, Algorithmica, 53 (2009), pp. 402–424.

[172] N. Robertson and P. D. Seymour, *Graph minors. I. Excluding a forest*, Journal of Combinatorial Theory, Series B, 35 (1983), pp. 39–61.

[173] ——, *Graph minors. II. Algorithmic aspects of tree-width*, Journal of Algorithms, 7 (1986), pp. 309–322.

[174] ——, *Graph minors. XIII. The disjoint paths problem*, Journal of Combinatorial Theory, Series B, 63 (1995), pp. 65–110.

[175] ——, *Graph minors. XVI. Excluding a non-planar graph*, Journal of Combinatorial Theory, Series B, 89 (2003), pp. 43–76.

[176] A. Schrijver, *Theory of Linear and Integer Programming*, Wiley-Interscience series in discrete mathematics and optimization, John Wiley and Sons, 1998.

[177] ——, *Combinatorial Optimization: Efficiency and Polyhedra*, Springer-Verlag, Berlin, 2003.

[178] P. D. Seymour, *Packing directed circuits fractionally*, Combinatorica, 15 (1995), pp. 281–288.

[179] R. Shamir, R. Sharan, and D. Tsur, *Cluster graph modification problems*, Discrete Applied Mathematics, 144 (2004), pp. 173–182.

[180] R. Sharan and R. Shamir, *Center click: A clustering algorithm with applications to gene expression analysis*, in ISMB, P. E. Bourne, M. Gribskov, R. B. Altman, N. Jensen, D. A. Hope, T. Lengauer, J. C. Mitchell, E. D. Scheeff, C. Smith, S. Strande, and H. Weissig, eds., AAAI, 2000, pp. 307–316.

[181] P. H. Sneath, *Numerical taxonomy*, in Bergey's Manual of Systematic Bacteriology, D. J. Brenner, N. R. Krieg, J. T. Staley, and G. M. Garrity, eds., Springer US, 2005, pp. 39–42.

[182] P. H. Sneath and R. R. Sokal, *Numerical Taxonomy*, W.H. Freeman, San Francisco, 1973.

[183] R. R. Sokal and P. H. Sneath, *Principles of Numerical Taxonomy*, W.H. Freeman, San Francisco, 1963.

[184] R. E. Tarjan, *Efficiency of a good but not linear set union algorithm*, Journal of the ACM, 22 (1975), pp. 215–225.

[185] S. Thomassé, *A $4k^2$ kernel for feedback vertex set*, ACM Transactions on Algorithms, 6 (2010).

[186] J. M. M. van Rooij and H. L. Bodlaender, *Design by measure and conquer, a faster exact algorithm for dominating set*, in Albers and Weil [11], pp. 657–668.

[187] J. M. M. van Rooij, J. Nederlof, and T. C. van Dijk, *Inclusion/Exclusion meets measure and conquer*, in ESA, A. Fiat and P. Sanders, eds., vol. 5757 of LNCS, Springer, 2009, pp. 554–565.

[188] A. van Zuylen and D. P. Williamson, *Deterministic pivoting algorithms for constrained ranking and clustering problems*, Mathematics of Operations Research, 34 (2009), pp. 594–620.

[189] G. J. Woeginger, *Exact algorithms for NP-hard problems: A survey*, in Combinatorial Optimization - Eureka, You Shrink!, Springer Berlin/Heidelberg, 2003, pp. 185–207.

[190] ——, *Open problems around exact algorithms*, Discrete Applied Mathematics, 156 (2008), pp. 397–405.

[191] Z. Wu and R. Leahy, *An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 15 (1993), pp. 1101–1113.

[192] D. L. P. C. Xu and J. L. Prince, *Current methods in medical image segmentation*, Annual Review of Biomedical Engineering, 2 (2000), pp. 315–337.

[193] R. Xu and D. C. W. II, *Clustering*, IEEE Press Series on Computational Intelligence, Wiley & IEEE Press, Hoboken, New Jersey, 2009.

[194] T. Zhang, R. Ramakrishnan, and M. Livny, *Birch: An efficient data clustering method for very large databases*, SIGMOD Rec., 25 (1996), pp. 103–114.

VITA

Name:	Cao, Yixin (操宜新)
Address:	Department of Computer Science, Texas A&M University,
	College Station, TX 77843-3112
Email:	yixin@cse.tamu.edu
Education:	Ph.D. in Computer Science, Texas A&M University, 2012
	M.S. in Computer Science, Beijing University of Aeronautics and
	Astronautics, China, 2003
	B.E. in Automation, Harbin Engineering University, China, 2000