Approximation Algorithms for Partition and Design Problems in Networks

Liang ZHAO

-------------------------------Errata -------------------------------

p36 lines 6 and 7, p38 line -8, p41 lines 6 and 7,
"E ¥cap 2^W" should be "¥{e ¥cap W | e ¥in E¥} - ¥{¥emptyset¥};

p36 line 8, p41 line 8, "e ¥cap 2^W" should be "e ¥cap W"

p69, subsection 3.5.1, there is a major bug on the conclusion,
since "the Lemmas 3.1, 3.2 CANNOT be extended to MPP in a
straightforward manner," thus it is unknown if the performance
guarantees obtained for MPP-NTs hold for their target split versions.
Thanks for an anonymous referee of DAM for pointing out this.

APPROXIMATION ALGORITHMS FOR PARTITION AND DESIGN PROBLEMS IN NETWORKS

Liang ZHAO

Department of Applied Mathematics and Physics
Graduate School of Informatics
Kyoto University
Kyoto 606-8501, Japan

March, 2002

Doctoral Dissertation

submitted to Graduate School of Informatics, Kyoto University

in partial fulfillment of the requirement for the degree of

DOCTOR OF INFORMATICS

(Applied Mathematics and Physics)

Preface

Reliability is one of the most important properties of networks, which has been studied as a subject in combinatorial optimizations for many years. This thesis treats two kinds of problems that are related to network reliability: the *multiway partition problem* (MPP) considers to estimate the reliability of a given network, and the *survivable network design problem* (SNDP) tries to construct reliable networks at low costs.

A family of nonempty, pairwise-disjoint subsets of a set V whose union is V is called *k-partition* of V. Given a system (V, T, f, k), where V is a finite set, $T \subseteq V$, $f : 2^V \to \mathbf{R}$ is a submodular function and k is an integer, the general MPP asks to find a k-partition $\mathcal{P} = \{V_1, V_2, \ldots, V_k\}$ of V satisfying $V_i \cap T \neq \emptyset$ for all i that minimizes $f(V_1) + f(V_2) + \cdots + f(V_k)$. As an application, the next network problem can be formulated as MPP. Suppose that we are given a network that consists of two kinds of nodes: *terminals* and *nonterminals*, and links between pairs of nodes. Terminals never fail, whereas nonterminals and links may fail independently. Given the failure rates, find a set of nonterminals and links with the maximum failure probability whose failure leaves the terminals with at least k pairwise-disconnected groups. Besides this problem in network reliability analysis, MPP also appears in numerous applications such as VLSI design, parallel computing and others.

On the other hand, SNDP arises in designing reliable networks at low costs. Analogously, suppose that we are given a network with two kinds of nodes, terminals and nonterminals, and links between pairs of nodes. Nonterminals and links have nonnegative weights (e.g., the installation costs). Let us use "nonterminal/link" to denote a member in the set of nonterminals and links. Problem SNDP is: given integers r_{st} (*connectivity requirement*) for each pair of terminals s and t, find a minimum weight subnetwork in which there are at least r_{st} nonterminal/link disjoint paths between (each pair of) terminals s and t, where *nonterminal/link disjointness* means that no two of paths share a common nonterminal or link. Notice that, two terminals s and t are still connected after the failure of $r_{st} - 1$ nonterminal/links in a feasible solution. In the terminology of reliability, they are said to be able to "survive" the failure of at least $r_{st} - 1$ nonterminal/links. This is where the name of this problem came from. SNDP captures a generalization of many familiar problems such as *shortest path*, *minimum spanning tree*, *Steiner tree* (edge weighted or node weighted), *minimum k edge-connected spanning* *subgraph*. It has many applications in communication systems, transportation systems and others.

Unfortunately, both MPP and SNDP are NP-hard. Thus it is generally hard to obtain an optimal solution. So far, a lot of approaches, exact or approximate, have been developed. However, most of them are either inefficient or are designed for only restricted cases. Especially, they take no or little consideration on the existence of nonterminals. Before this work, little was known for our general formulations despite the practical importance.

In this thesis, we aim to design *approximation algorithms* that can find near-optimal solutions efficiently. The structure of the thesis goes as follows.

Chapter 1 provides an introduction with applications. After that, we present a unified framework for developing and analyzing approximation algorithms for MPP in Chapter 2 and 3. Chapter 4 treats SNDP, which is formulated (actually, generalized) by employing a requirement function. An efficient primal-dual method based approximation algorithm is proposed. Another approach for SNDP is given in Chapter 5, which utilizes approximation algorithms for "simpler" versions of SNDP. Finally in Chapter 6, we conclude the results obtained in this thesis. References and index are given at the end.

We are aware that many unknown facts remain in this exciting field. In particular, how far we can go is always a challenging question when considering approximation algorithms. It is deeply hoped that the new results obtained in this thesis would be helpful to stimulate future researches, as an old saying goes, "This is not an end, but a beginning."

Kyoto, March 2002

Liang Zhao

Acknowledgement

First of all, I would like to thank my parents Yingzhou Zhao and Jiyi Xiang. Everything began from them. I also want to thank my brothers Jing and Qing, and their family. Thanks for caring me so long.

I am greatly indebted to my advisor, Professor Toshihide Ibaraki. He gave me the opportunity to study in Kyoto University. During my near six years stay as a foreign student, he always gives me invaluable advice and encouragement, and helps me a lot both in study and in finance. Even in his very busy moment, he takes time to supervise my study, in particular, to supervise this dissertation. He commented very carefully on the the manuscript, which significantly improved the accuracy and quality.

I am deeply grateful to Professor Hiroshi Nagamochi for his heartful and earnest guidance and discussion. Without his guidance and help, none of the work could have been possible. I have learned so much, from methodology of doing research to the study of Japanese, from him that I could not forget at all.

I am indebted to Hirotaka Ono for his continuous, not discrete, helps and discussions throughout both my master's and doctoral studies. I also thank other members in the set of Ibaraki Lab., especially, to Professor Mutsunori Yagiura, Yoshitaka Nakao, Professor Koji Nonobe and Professor Toshimasa Ishii. They and Jie Huang and Huiyu Wang have given me so much helps and pleasure during my studies in Ibaraki Lab.

I am also thankful to Professor Masao Fukushima and Professor Kazuo Iwama for serving on my dissertation committee.

I want to express my special thanks to Guangyao Luo — my mathematics teacher in high school, for his showing me the great world of mathematics and computer science.

Finally, I want to express my heartiest appreciation to my wife Xiaqing Zheng. Without her support, kindness and encouragement, this dissertation would not exist.

Contents

1 **Introduction** **1**

1.1 Notations and definitions . . . 1

1.2 Multiway partition problem (MPP) . . . 4

1.2.1 Formulation and applications . . . 4

1.2.2 More applications . . . 8

1.2.3 Historical background . . . 13

1.3 Survivable network design problem (SNDP) . . . 14

1.3.1 Formulation and applications . . . 14

1.3.2 More applications . . . 16

1.4 Preliminary of set functions in hypergraphs . . . 18

2 **Greedy Splitting Algorithm for MPP** **21**

2.1 Introduction . . . 21

2.1.1 Survey on MPP-NT (MPP with no target) . . . 21

2.1.2 Survey on general MPP . . . 26

2.2 Greedy splitting algorithm (GSA) for MPP-NT . . . 28

2.2.1 Greedy splitting approach . . . 28

2.2.2 Algorithm description and main lemma . . . 29

2.2.3 Performance analysis: part I . . . 32

2.2.4 Performance analysis: part II . . . 35

2.3 Modified GSA (M-GSA) for general MPP . . . 38

2.3.1 Algorithm description and main lemma . . . 38

2.3.2 Performance analysis . . . 40

2.4 Tight examples . . . 42

3 **Greedy Splitting of Higher Accuracy for MPP** **45**

3.1 Introduction . . . 45

3.2 Greedy splitting algorithm 2 (GSA2) . . . 45

3.2.1 Algorithm description and main lemma for odd k . . . 45

3.2.2 Algorithm description and main lemma for even k 53
3.3 Performance analysis of GSA2 57
3.3.1 Polynomial time implementation 57
3.3.2 Performance guarantee: part I 58
3.3.3 Performance guarantee: part II 59
3.4 Tight examples 66
3.5 Modified GSA2 (M-GSA2) for general MPP and remarks 69
3.5.1 Modified GSA2 (M-GSA2) for general MPP 69
3.5.2 Remarks 69

4 A Primal-Dual Approximation Algorithm for SNDP 73
4.1 Introduction 73
4.1.1 Network design problem with requirement function (NDRF) 73
4.1.2 A survey on SNDP 74
4.1.3 Preliminary 76
4.2 Primal-dual approximation algorithm for NDRF 76
4.3 Proof of performance guarantee 80
4.4 Implementation for SNDP 85
4.5 Remarks 88

5 Another Approach for SNDP 91
5.1 Introduction 91
5.2 Algorithms for SNDP 92
5.2.1 Approximating SNDP by employing algorithm for SNDP-G 92
5.2.2 Approximating SNDP by employing algorithm for SNDP-G$^+$ 94
5.3 Extensions to NDRF 94
5.3.1 Approximating NDRF by employing algorithm for NDRF-G 95
5.3.2 Approximating NDRF by employing algorithm for NDRF-G$^+$ 96

6 Conclusion 99
6.1 Approximation algorithms for MPP 99
6.2 Approximation algorithms for SNDP 101

Bibliography 103

Index 113

List of Figures

1.1 A (connected) graph and a path (s,t-path). 1
1.2 A spanning tree of the graph in Figure 1.1. 2
1.3 A target-split cut of the graph in Figure 1.1. 5
1.4 A computer network, where computers are connected by links and hubs. . . . 8
1.5 The graph model for Figure 1.4, where nonterminals and edges have weights. 9
1.6 The bipartite graph for Figure 1.5, where only nonterminals have weights. . . 9
1.7 Hypergraph and notations Γ and δ. 10
1.8 The hypergraph model for Figure 1.6, where hyperedges have weights. 11
1.9 Hyperedge disjoint (s,t-) paths in a hypergraph and their corresponding nonterminal disjoint (s,t-) paths in the corresponding bipartite graph. 14
1.10 Graph and Steiner tree. 15
1.11 Unicast and multicast in communication network. 17
1.12 Minimum connected spanning subgraph in hypergraph may not be a tree. There are two paths ($b, \{a, b, c\}, c$ and $b, \{d, b, c\}, c$) between vertices b and c. . 18

2.1 Splitting a p_i-partition to obtain a p_{i+1}-partition. Greedy algorithm tries to minimize the weight increase. 29
2.2 Greedy splitting algorithm (GSA) for MPP-NT 30
2.3 Modified greedy splitting algorithm (M-GSA) for MPP. 39
2.4 A tight example for GSA applied to k-PPSS. 43
2.5 A tight example for GSA applied to k-PPG. 44

3.1 Greedy splitting algorithm 2 (GSA2) for MPP-NT for odd k 46
3.2 Illustration of sub-case (1a) for k-PPG. 49
3.3 Greedy splitting algorithm 2 (GSA2) for MPP-NT for even k. 53
3.4 Tight example for GSA2 applied to k-PPSS. 66
3.5 Tight example for GSA2 applied to k-PPSSS. 67
3.6 Tight example for GSA2 applied to k-PPG. 68
3.7 Lemma 3.2 fails if we compute a minimum 3-partition at first. 70
3.8 An example for which main lemmas cannot be extended. 71

4.1 ALGO-IP approximation algorithm for (IP). 77
4.2 PRIMAL-DUAL: primal-dual approximation algorithm for $(\mathrm{IP})_i$ 79
4.3 It holds that $a \in \Delta_B(C_a \cap C_b) \cup \Delta_B(C_a \cup C_b)$. 83
4.4 Construct a tree from a laminar set family by the set inclusion relationship. . 83
4.5 $\mathcal{D}$ transforms a bipartite graph into a directed graph, where edges (u, u^c) have capacity one, and other edges have capacity ∞. 86
4.6 A tight example for Lemma 4.2. 89

5.1 Construction of the edge weight function $\bar{w}$ from nonterminal weight function w. 92
5.2 Algorithm SNDP-G-TO-SNDP. 93
5.3 Algorithm SNDP-G$^+$-TO-SNDP 94
5.4 Algorithm NDRF-G-TO-NDRF 96
5.5 Construction of the weight function $\tilde{w}$ from the weight function w. 96
5.6 Algorithm NDRF-G$^+$-TO-NDRF 97

List of Tables

2.1 Complexities of MPP-NTs 25
2.2 Previous best-known approximation results for MPP-NT 26

6.1 Approximation results obtained in this thesis for MPP-NT. 99
6.2 Performance guarantees of algorithm GSA2 for MPP-NT (assuming the existence of polynomial time implementations). 100
6.3 Approximation results obtained in this thesis for general MPP. 101

Chapter 1

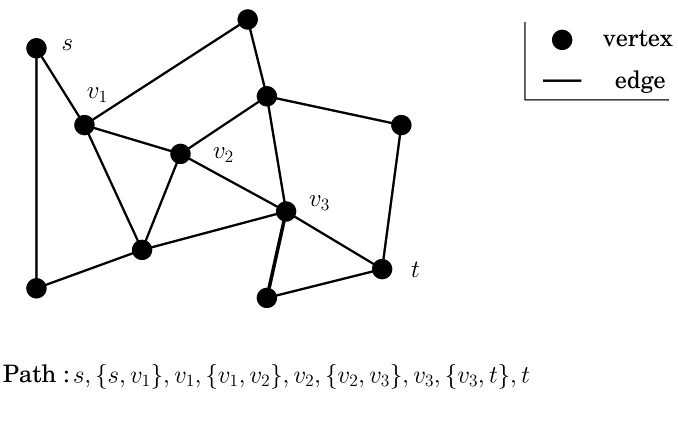
Introduction

1.1 Notations and definitions

Let us start with graph.

A *graph* is a pair $G = (V, E)$ of a *vertex set* V and an *edge set* E, where edges are two-elements subsets of V. We use $e = \{u, v\}$ to denote an edge e with two *endpoints* $u, v \in V$. In general, we allow the existence of *multiple edges*, i.e. edges having the same endpoints. Thus the set E of edges should be viewed as a set that allows multiple members. The notation $e = \{u, v\}$ is used without confusion. Graph is called *network* if vertices or edges, or both, have weights.

Let $G = (V, E)$ be a graph. A *path* (*s,t-path*) between two vertices s and t is a sequence $v_0 = s$, e_1, v_1, …, v_{i-1}, e_i, v_i, …, e_j, $v_j = t$, where v_i (resp. e_i) are distinct vertices (resp. edges) such that $e_i = \{v_{i-1}, v_i\}$ holds for $i = 1, \ldots, j$. If there is a path between each pair of vertices, we say that graph G is *connected*. Two vertex subsets $V_1, V_2 \subseteq V$ are said to be *disconnected* if there is no path between a vertex in V_1 and a vertex in V_2. A (connected) graph and a path in it are illustrated in Figure 1.1.



Path : $s, \{s, v_1\}, v_1, \{v_1, v_2\}, v_2, \{v_2, v_3\}, v_3, \{v_3, t\}, t$

Figure 1.1: A (connected) graph and a path (s,t-path).

Graph $G' = (V', E')$ is *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E \cap 2^{V'}$ hold (recall that $2^{V'} \triangleq \{S \mid S \subseteq V'\}$). It is *spanning* if $V' = V$. The subgraph of G *induced* by a vertex subset $S \subseteq V$ is defined as $G[S] \triangleq (S,\, E \cap 2^S\,)$. A *connected component*, or *component* in short, is a maximal connected subgraph of G, i.e., a connected subgraph of G such that adding any vertex or edge not in it (but in G) violates the connectedness[1]. Obviously, a component must be an induced subgraph such that contains at least one vertex.

A *tree* is such a graph that there is a unique path between each pair of vertices (hence it is connected). A *forest* is a graph such that there is at most one path between each pair of vertices. It is clear that a forest is a collection of trees, where each tree forms a component. If a spanning subgraph of a graph is tree (resp. forest), we call it *spanning tree* (*spanning forest*). A spanning tree of the graph in Figure 1.1 is shown in Figure 1.2.

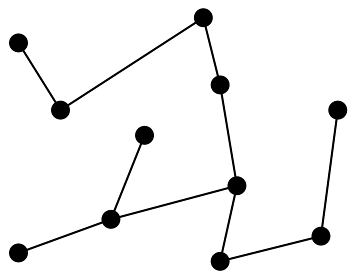


Figure 1.2: A spanning tree of the graph in Figure 1.1.

We need more notations. An *optimization problem* asks to minimize or maximize a function under given constraints. We consider minimization problem, which can be written as

$$\begin{array}{rl} \text{minimize} & f(x) \\ \text{subject to} & x \in \Omega, \end{array}$$

where f, x, $f(x)$ and Ω are called *objective function*, *solution*, *objective value* and *feasible region* (usually specified implicitly), respectively. A solution x is said to be *feasible* if $x \in \Omega$; otherwise it is *infeasible*. A feasible solution x^* satisfying $f(x^*) \leq f(x)$ for all $x \in \Omega$ is called an *optimal solution*, and $f(x^*)$ is called the *optimal value* or *optimum*. It is clear that the optimum is unique (if one exists), whereas optimal solutions may not be unique.

A *combinatorial optimization problem* (COP) is an optimization problem in which the feasible region has a combinatorial structure, typically defined by a subset of 2^V for a finite set V. For instance, given a graph, the problem of finding a minimum subset of edges whose removal leaves the graph with k components (*k-way cut problem*), and the problem of finding a minimum spanning tree, are all COPs. Generalizing these two problems, this thesis considers

[1]In general, we say that a set is *maximal* (resp. *minimal*) with respect to a property if it satisfies the property but none does its superset (resp. subset) with one more (resp. less) element.

two types of COPs: the *multiway partition problem* (MPP) and the *survivable network design problem* (SNDP). Precise formulations will be given in Sections 1.2 and 1.3, respectively.

The minimum spanning tree problem is well known, and can be solved efficiently. However, as will be seen later, MPP and SNDP are *NP-hard.* NP-hard problems are strongly believed to have no polynomial time algorithms (see [GJ 79, Hoc 97]; this is the famous unsolved $P \neq NP$ conjecture), where an algorithm is *polynomial time*, or have *polynomial running time*, if it can complete the computation in polynomial time of the input size. Thus it is reasonable to sacrifice optimality and settle for a "good" feasible solution that can be found efficiently (i.e., in polynomial time). While there are algorithmic tools for solving NP-hard problems exactly (which do not guarantee polynomial running time), polynomial time approximate methods, called *heuristics* (or *meta-heuristics*) and *approximation algorithms* are often used. In this thesis, we take the latter approach: approximation algorithm.

Definition 1.1. *Let $\mathcal{P}$ be a minimization problem for which each instance I has its optimum $opt(I) \geq 0$. A polynomial time algorithm $\mathcal{A}$ is said ρ-approximation algorithm for $\mathcal{P}$ if it delivers a feasible solution of objective value at most $\rho \cdot opt(I)$ for any instance I of $\mathcal{P}$, where the value ρ (≥ 1), independent on I, is called the performance guarantee.*

By this definition, an approximation algorithm always has polynomial running time. Other terms for performance guarantee include *approximation factor*, *approximation ratio*, etc.

In considering COPs, the next definition contains important concepts. Let $\mathbf{R}$ (resp. $\mathbf{R}^+$) denote the set of real numbers (resp. set of nonnegative real numbers).

Definition 1.2. *A set function $f : 2^V \to \mathbf{R}$ is said*

- **symmetric** *if $f(X) = f(V - X)$ holds for all $X \subseteq V$;*
- **monotone** (*nondecreasing*) *if $f(X) \leq f(Y)$ holds for all $X \subseteq Y \subseteq V$;*
- **submodular** (*resp.* **supermodular**) *if $f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$ (resp. $f(X) + f(Y) \leq f(X \cap Y) + f(X \cup Y)$) holds for all $X, Y \subseteq V$;*
- **weakly supermodular** *if $f(X) + f(Y) \leq \max\{f(X \cap Y) + f(X \cup Y),\ f(X - Y) + f(Y - X)\}$ holds for all $X, Y \subseteq V$.*

It is trivial to see the next properties.

Proposition 1.1. *Let f, $g : 2^V \to \mathbf{R}$ be two set functions.*

1. *Function $-f$ is submodular if and only if f is supermodular.*
2. *Function $\bar{f}$ defined by $\bar{f}(S) \triangleq f(V - S)$ for all $S \subseteq V$ is submodular (resp. supermodular) if and only if so is f.*

3. *If functions f and g are submodular (resp. supermodular), then function $f+g$ defined by $(f+g)(S) \triangleq f(S)+g(S)$ for all $S \subseteq V$ is submodular (resp. supermodular). In particular, $f+c$ is submodular (resp. supermodular) for any constant $c \in \mathbf{R}$.* $\square$

For further studies, we refer the readers to [DMM 97] for a recent comprehensive annotated bibliographies for combinatorial optimization. Among many good books in this field, there are two up-to-date textbooks [CCPS 98, KV 00] that cover many of the latest progresses, as well as fundamental and classical results. For classic books on NP-hard problems and computational complexity, see [GJ 79, Pap 94]. For exact-algorithm approaches for (NP-hard) COPs, see [Bel 57, JRT 95, Iba 87, LB 96]. For approximation algorithms (resp. heuristics), see [GGU 72, Joh 74, Hoc 97, KV 00] ([Ree 93, OL 96]). Submodular functions, their applications and related topics are found in [Fra 97, Fuj 91, Lov 83, Nar 97].

1.2 Multiway partition problem (MPP)

Let us first introduce the multiway partition problem (MPP).

1.2.1 Formulation and applications

Problem 1.1 (MPP). *Given a system (V,T,f,k), where V is a finite set, $T \subseteq V$ called target set, $f: 2^V \to \mathbf{R}$ is a submodular function and $k \geq 1$ is an integer, MPP is defined by*

$$\begin{aligned} \text{minimize} \quad & f(V_1)+f(V_2)+\cdots+f(V_k) \\ \text{subject to} \quad & V_1 \cup V_2 \cup \cdots \cup V_k = V, && (1.1) \\ & V_i \cap V_j = \emptyset, \quad 1 \leq i \neq j \leq k, && (1.2) \\ & V_i \cap T \neq \emptyset, \quad i = 1,2,\ldots,k. && (1.3) \end{aligned}$$

Without loss of generality we assume $|T| \geq k$, since otherwise there is no feasible solution. We also assume that function f is given by an oracle which returns $f(S)$ for a given $S \subseteq V$.

A family $\mathcal{P} = \{V_1, V_2, \ldots, V_k\}$ of nonempty, pairwise-disjoint subsets of V whose union is V is called *k-partition of V*. If V is identified without confusion, we may say "k-partition" instead of "k-partition of V". The *weight* of $\mathcal{P}$ is $f(\mathcal{P}) \triangleq f(V_1)+f(V_2)+\cdots+f(V_k)$. A k-partition that satisfies (1.3) (i.e., a feasible solution of MPP) is called *target-split* (of V, with respect to T). Notice that, when $T = V$, MPP reduces to the problem of finding a minimum weight k-partition of V.

MPP has many practical applications. Let us see one in the following. Consider a connected communication network that consists of two types of elements: nodes and links between pairs of nodes. Due to the failures of links, a specified group of nodes (*targets*) can possibly be separated into pairwise-disconnected subgroups. Let V and $T \subseteq V$ denote the set of nodes and the set of targets, respectively. To estimate the reliability with respect to T, one may want to know the maximum probability of such failure of links that disconnect some k targets from each other, $2 \leq k \leq |T|$. Such probability is called *robustness index*. It can be said that the smaller the robustness index, the more reliable the network.

Let us consider a formulation for finding the robustness index. Suppose that the failure rate (probability) of each link is known in advance, and links fail independently. The failure probability of a link subset E' is then the *product* of failure rates of links in E'. As it is often convenient to use *sum* instead of product, we give each link e of failure rate $p(e)$ a weight of $w(e) = -\ln p(e)$ (note $w(e) \geq 0$). The weight of a link subset E' is given by

$$\sum_{e \in E'} w(e) = -\sum_{e \in E'} \ln p(e) = -\ln\left(\prod_{e \in E'} p(e)\right),$$

which is large if and only if the failure probability $\prod_{e \in E'} p(e)$ of E' is small. Let vertices and edges represent nodes and links respectively. The problem of finding the robustness index for a k is then formulated as the following problem.

Problem 1.2 (Target split problem). *Given a graph $G = (V, E)$ with a target set $T \subseteq V$ and an edge weight function $w : E \to \mathbf{R}^+$, find a minimum weight subset of edges whose removal leaves T with at least k nonempty pairwise-disconnected subsets.*

We call a subset of edges *k-way cut* if its removal leaves the graph with at least k components. Obviously, a feasible solution of target split problem, called *target-split cut*, must be a k-way cut. Alternatively, target-split cut is a k-way cut whose removal leaves the graph with at least k (may not be all) components that contain at least one target. An example of target-split cut for the graph in Figure 1.1 is illustrated in Figure 1.3.

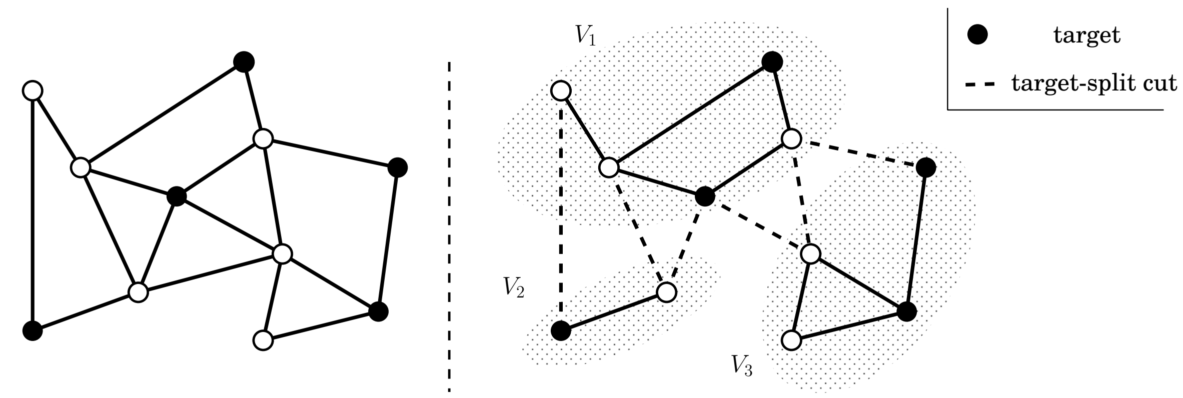


Figure 1.3: A target-split cut of the graph in Figure 1.1.

Clearly, instead of the original definition of robust index, the weight of a minimum target-split cut can be used to estimate the reliability between targets: the larger the more robust.

Sometimes it is called as the robustness index (instead of the corresponding probability).

Before we give the MPP formulation for target split problem, let us see two special cases and their applications. One special case with $T = V$ (i.e., the problem of finding a minimum weight k-way cut) is called *k-way cut problem.* Two decades ago, Pulleyblank [Pul 82] (see [BG 97]) formulated it in connection with generating the cutting plane for solving the traveling salesman problem. After that, numerous applications of the k-way cut problem have been found, e.g., in network reliability analysis for which we have seen an example. See also [BG 97, Col 87, Cun 85, GH 88, Stoe 92, Tit 99].

We have another application of k-way cut problem. A family DST of spanning trees is *disjoint* if no two of them share a common edge. The maximum number of disjoint spanning trees is considered as a measure of network invulnerability, see e.g., [Gus 83]. Let

$$\sigma(G) \triangleq \min_{k\geq 2} \min\Big\{\frac{|E'|}{k-1} \;\Big|\; E' \text{ is a } k\text{-way cut of } G\Big\}.$$

Let DST denote a family of disjoint spanning trees. The next equality is due independently to Tutte [Tut 61] and Nash-Williams [NW 61] (see also [CCPS 98], Exercise 8.46).

$$\max\{|DST|\} = \lfloor \sigma(G) \rfloor. \tag{1.4}$$

A natural generalization to edge-weighted graph was considered by Cunningham [Cun 85]. He introduced the *strength* for a graph G with a nonnegative edge weight function w as

$$\sigma(G,w) = \min_{k\geq 2} \min\Big\{\frac{\sum_{e\in E'} w(e)}{k-1} \;\Big|\; E' \text{ is a } k\text{-way cut of } G\Big\}. \tag{1.5}$$

Thus $\sigma(G) = \sigma(G,1)$ holds. Obviously, $\sigma(G)$ (hence $\max\{|DST|\}$) and $\sigma(G,w)$ can be found by solving k-way cut problems for $k = 2,\ldots,n$, where n is the number of vertices.

The k-way cut problem also has applications in parallel computing, in which one wants to assign n computation tasks to k processors. It is assumed that, some pairs of tasks must communicate during the computation, and communications take costs (e.g. network traffics). If such a pair of tasks are assigned to the same processor, then only a negligible cost will be incurred. On the other hand, if they are assigned to different processors, then a large communication cost is incurred. Let vertices denote computation tasks, and edges denote the need of communication between two tasks. The communication cost of two tasks, if assigned to different processors, is associated with the corresponding edge. We see that the problem of finding an assignment of tasks to processors that minimizes the communication cost is nothing but a k-way cut problem. See also [LKP 90, Ston 77].

Other applications of k-way cut problem include VLSI layout. In VLSI layout, a graph represents the connection diagram of a circuit, in which vertices and edges represent circuit modules and interconnections between pairs of modules, respectively. One wants to find a layout of the circuit into k layers at the minimum layout cost. Similarly as the previous processor assignment problem, if two modules, corresponding to the endpoints of an edge, is assigned to different layers, then a cross-layer cost occurs; otherwise (i.e., they are assigned to the same layer), only a small inner-layer cost occurs which can be omitted. Let each edge be given the cross-layer cost. It is then easy to see that the problem of finding a layout minimizing the layout cost can be formulated as k-way cut problem. See also [CO 96, Len 90].

Another special case of the target split problem asks to find a minimum weight k-way cut whose removal disconnects all k specified target vertices (i.e., $|T| = k$). It is often called the *multiterminal cut problem* . This problem also has applications in network reliability analysis and others, see [DJPSY 83]. Some other closely-related variants and applications of target split problem can be found in [Hu 69, Ston 77].

We now show that the target split problem can be formulated as MPP. For this, given a graph $G = (V, E)$ with a target set $T \subseteq V$ and a weight function $w : E \to \mathbf{R}^+$, we denote the weight of an edge subset $E' \subseteq E$ by

$$w(E') \triangleq \sum_{e \in E'} w(e).$$

Define a function $w_{\rm ex} : 2^V \to \mathbf{R}^+$ by

$$w_{\rm ex}(S) \triangleq \sum_{\emptyset \neq e \cap S \neq e} w(e), \quad S \subseteq V,$$

i.e., $w_{\rm ex}(S)$ is the weight of edges that have one but not all endpoints in S. Function $w_{\rm ex}$ is called the *cut function*, which is submodular (see Proposition 1.4). Let us show that target split problem can be formulated as MPP with system $(V, T, w_{\rm ex}, k)$, where the latter is called the *k-target-split problem in graphs* (k-TPG) in this thesis. For this, we consider a mapping from k-partition to k-way cut, where $\mathcal{P} = \{V_1, V_2, \ldots, V_k\}$ is mapped to $E_{\mathcal{P}} \triangleq \{\{u, v\} \in E \mid \exists\, i \neq j,\ u \in V_i,\ v \in V_j\}$. (In Figure 1.3, take $\mathcal{P} = \{V_1, V_2, V_3\}$, $E_{\mathcal{P}}$ is nothing but the target-split cut.) It then holds that

$$\begin{aligned} w_{\rm ex}(\mathcal{P}) &= \sum_{i=1}^{k} w_{\rm ex}(V_i) = \sum_{i=1}^{k} \sum_{\emptyset \neq e \cap V_i \neq e} w(e) \\ &= \sum_{e} \sum_{i : \emptyset \neq e \cap V_i \neq e} w(e) = 2 \sum_{e \in E_{\mathcal{P}}} w(e) = 2w(E_{\mathcal{P}}). \end{aligned}$$

(Note that each edge has exactly two endpoints). Thus, from a feasible solution $\mathcal{P}$ of k-TPG, we can construct a feasible solution $E_{\mathcal{P}}$ of target split problem with half the weight. On the other hand, it is not difficult to see that a minimal target-split cut must be a k-way cut $E_{\mathcal{P}'}$ for some k-partition $\mathcal{P}'$ of V that is feasible to k-TPG. Hence we conclude that the target split problem can be formulated as k-TPG, i.e., MPP with system $(V, T, w_{\rm ex}, k)$.

In particular, the k-way cut problem can be formulated as MPP with system $(V, V, w_{\rm ex}, k)$. We call the latter problem *k-partition problem in graphs* (k-PPG). We also denote the k-TPG problem with $|T| = k$ by k-MCP, since it is equivalent to the multiterminal cut problem. In the following, we will not distinguish between k-way cut problem and k-PPG, nor between multiterminal cut problem (resp. target split problem) and k-MCP (resp. k-TPG).

1.2.2 More applications

In the previous graph models of applications, there is only one kind of vertices. Sometimes such models may be too simple to handle real networks. For instance, consider a computer network such as the Internet. We know that computers are not connected simply by links. Instead, they are connected by links and *hubs*[2]. While a link is incident to only two of computer/hubs (i.e., two computers, or two hubs, or one computer and one hub), a hub connects two or more computer/hubs by links. This is illustrated in Figure 1.4. This fact suggests the necessity of new models that consist of two (or more) kinds of vertices.

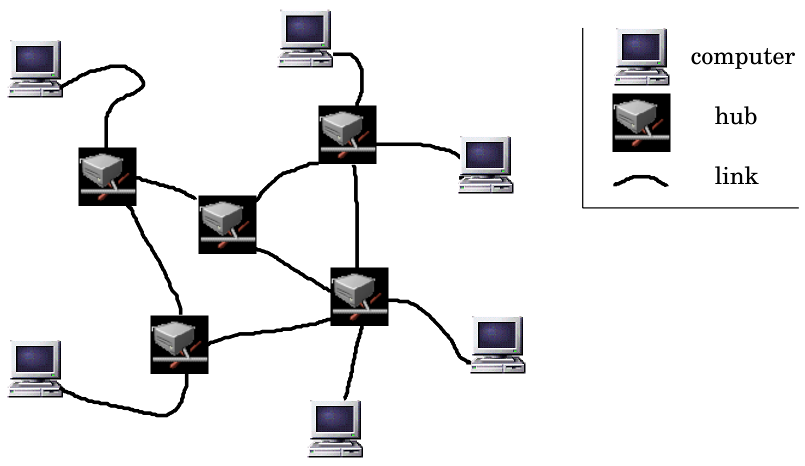


Figure 1.4: A computer network, where computers are connected by links and hubs.

Possibly hubs may fail just as links. Again, suppose that the failure rate of each hub/link is known in advance and they fail independently. Let terminals, nonterminals and edges represent computers, hubs and links, respectively. Associate each nonterminal/edge of failure rate p with weight $-\ln p \geq 0$. We again obtain a graph (see Figure 1.5). However, it is different from the previous model. There are two kinds of vertices: terminals and nonterminals, and nonterminals have weights as well as edges.

For estimating the reliability, we want to determine a minimum weight set of nonterminal/edges whose removal leaves the terminals with (at least) k pairwise-disconnected subsets. Note that k-PPG (in general, k-TPG) is a special case with empty nonterminal set. We claim that this generalized problem can also be formulated as MPP.

[2]We use "hub" to denote a network element in connections between interested computers but other than a link or an interested computer. It may or may not be a real hub as used in communication terminology.

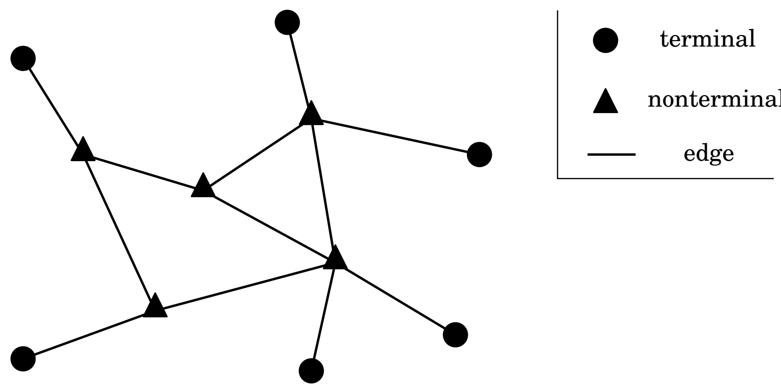


Figure 1.5: The graph model for Figure 1.4, where nonterminals and edges have weights.

Let us first consider two formulations. The first is bipartite-graph based. Formally, we replace each edge $e = \{s, t\}$ of weight w_e by two edges $\{s, u_e\}$ and $\{u_e, t\}$, introducing a new *nonterminal* u_e of weight $w(e)$ (edges $\{s, u_e\}$ and $\{u_e, t\}$ are not given weights). Observe that, only nonterminals have weights and no two terminals are adjacent[3] in the resulting graph. Next, we replace each edge $e' = \{s', t'\}$ between two *nonterminals* s' and t' by two edges $\{s', v'_{e'}\}$ and $\{v'_{e'}, t'\}$, introducing a new *terminal* $v'_{e'}$. After the second replacement, the graph becomes *bipartite*, i.e., each edge has one endpoint of terminal and another endpoint of nonterminal. Notice that only nonterminals have weights. See Figure 1.6.

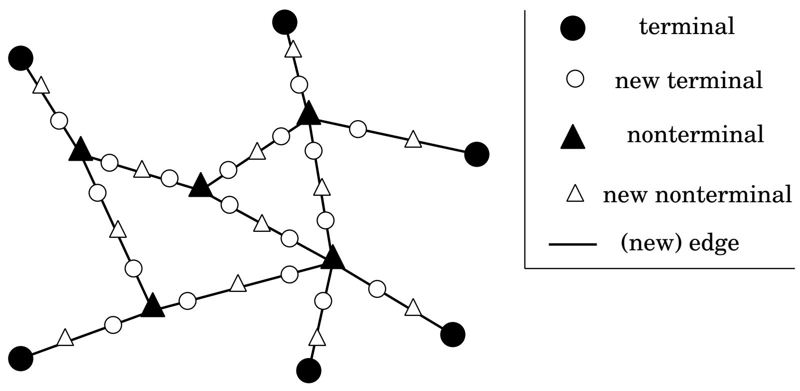


Figure 1.6: The bipartite graph for Figure 1.5, where only nonterminals have weights.

We now formulate the above problem by using the resulting bipartite graph.

Problem 1.3 (Target split problem in hypergraphs of type 1). *Given a bipartite graph G with a terminal set V, a nonterminal set U, a target set $T \subseteq V$, a set E of edges between V and U and a weight function $w : U \to \mathbf{R}^+$, find a minimum weight subset of nonterminals whose removal leaves T with at least k nonempty pairwise-disconnected subsets.*

[3]We allow the existence of edges between terminals in the input. It is not important by this replacement.

Let us use $G = (V, U, E)$ to denote the bipartite graph with terminal set V, nonterminal set U and edge set E between V and U. Before we give another formulation which explains the name of Problem 1.3, let us introduce the notation of hypergraph.

Hypergraph is a generalization of graph. A *hypergraph* is a pair $H = (V, E)$ of a *vertex set* V and a *hyperedge set* E, where hyperedges are nonempty subsets (of any cardinality) of V (see Figure 1.7). The *degree* of a hyperedge is defined as its cardinality. A hyperedge $e = \{v_1, \ldots, v_k\}$ may also be treated as the set $\{v_1, \ldots, v_k\}$ of vertices, where each v_i is called *endpoint* of e. Hence graphs are special hypergraphs, in which each hyperedge (i.e., edge) has degree 2. Again, we note that the set E of hyperedges should be viewed as a set that allows multiple members, since we allow the existence of multiple hyperedges.

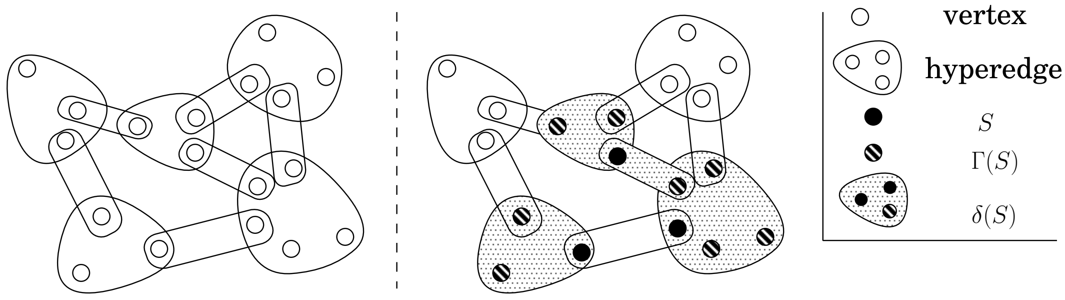


Figure 1.7: Hypergraph and notations Γ and δ.

We generalize notations for graph to hypergraph. Let $H = (V, E)$ be a hypergraph. A *subgraph* of H is a hypergraph $H' = (V', E')$ that satisfies $V' \subseteq V$ and $E' \subseteq E \cap 2^{V'}$. The subgraph of H *induced* by a vertex subset S is defined as $H[S] \triangleq (S,\ E \cap 2^S)$. A *path* (*s*,*t*-*path*) between two vertices s and t is a sequence $v_0 = s$, e_1, v_1, …, v_{i-1}, e_i, v_i, …, e_j, $v_j = t$, where v_i (resp. e_i) are distinct vertices (resp. hyperedges) such that v_{i-1}, $v_i \in e_i$ holds for $i = 1$, …, j. If there is a path between any two vertices, we say that H is *connected*. On the other hand, two vertex subsets V_1 and V_2 are *disconnected* if there is no path between a vertex in V_1 and a vertex in V_2.

We further introduce some notations. The set of *neighbors* of a vertex subset $S \subseteq V$ is

$$\Gamma(S) \ \triangleq\ \{v \in V - S \mid \exists e \in E,\ v \in e,\ e \cap S \neq \emptyset\}.$$

The set of hyperedges *incident* to S is defined as

$$\delta(S) \ \triangleq\ \{e \in E \mid \emptyset \neq e \cap S \neq e\},$$

which is called *cut* if $\emptyset \neq S \subset V$ holds. For a singleton set $\{v\}$, we use $\Gamma(v)$ and $\delta(v)$ instead of $\Gamma(\{v\})$ and $\delta(\{v\})$, respectively. The number of neighbors of vertex v (i.e., $|\Gamma(v)|$) is called the *degree* of v. We illustrate the definitions in Figure 1.7.

Now the name of Problem 1.3 becomes clear, since we have an alternative description by hypergraph. More precisely, using the notation Γ with respect to bipartite graph G, we leave terminals unchanged and replace each nonterminal u and all edges incident to it by a hyperedge $e_u = \Gamma(u)$ of the same weight as u. Then we obtain a hypergraph $H = (V, \tilde{E})$, where $\tilde{E} = \{e_u = \Gamma(u) \mid u \in U\}$. This is illustrated in Figure 1.8.

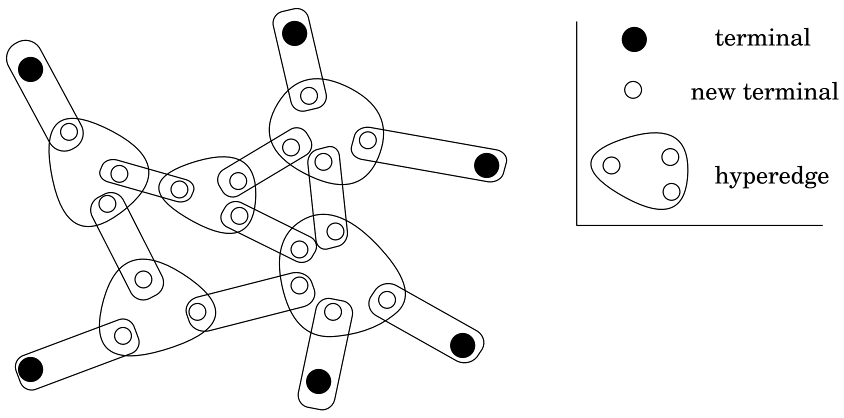


Figure 1.8: The hypergraph model for Figure 1.6, where hyperedges have weights.

With this interpretation, Problem 1.3 can be written as follows, where we use E instead of $\tilde{E}$ to denote the set of hyperedges for simplicity.

Problem 1.4 (Target split problem in hypergraphs of type 1). *Given a hypergraph $H = (V, E)$ with a vertex set V, an hyperedge set E, a target set $T \subseteq V$ and a weight function $w : E \to \mathbf{R}^+$, find a minimum weight subset of hyperedges whose removal leaves T with at least k nonempty pairwise-disconnected subsets.*

Similarly as before, we call a set of hyperedges *k-way cut* if its removal leaves the hypergraph with at least k components (i.e., maximal connected subgraphs). If there are at least k (may not be all) components that contain at least one target, it is called *target-split cut.*

Now we are ready to show the promised claim that Problem 1.3 can be formulated as MPP. We use the equivalent formulation of Problem 1.4.

Suppose that we are given a hypergraph $H = (V, E)$ with a target set $T \subseteq V$ and a weight function $w : E \to \mathbf{R}^+$. Let $w(E') \triangleq \sum_{e \in E'} w(e)$ denote the weight of a hyperedge subset $E' \subseteq E$. Let a function $w_{\text{in}} : 2^V \to \mathbf{R}^+$ be defined by

$$w_{\text{in}}(S) \triangleq \sum_{e \subseteq S} w(e) \quad S \subseteq V.$$

That is, $w_{\text{in}}(S)$ is the weight of hyperedges whose endpoints are all in S. Function w_{in} is supermodular (see Proposition 1.4). Hence function $\frac{w(E)}{k} - w_{\text{in}}$ is submodular by Proposition 1.1. We claim that Problem 1.4 can be formulated as MPP with system $(V, T, \frac{w(E)}{k} - w_{\text{in}}, k)$. For this, let $\mathcal{P} = \{V_1, V_2, \ldots, V_k\}$ be a k-partition of V. We see that the weight of $\mathcal{P}$ (with respect to function $\frac{w(E)}{k} - w_{\text{in}}$) is

$$\begin{aligned}\sum_{i=1}^{k}\left(\frac{w(E)}{k} - w_{\text{in}}\right)(V_i) &= w(E) - \sum_{i=1}^{k} w_{\text{in}}(V_i) = w(E) - \sum_{i=1}^{k}\sum_{e \subseteq V_i} w(e) \\ &= \sum_{e} \{w(e) \mid \exists i \neq j,\ e \cap V_i \neq \emptyset \neq e \cap V_j\}. \qquad (1.6)\end{aligned}$$

This is nothing but the weight of k-way cut $E_{\mathcal{P}} = \{w(e) \mid \exists i \neq j,\ e \cap V_i \neq \emptyset \neq e \cap V_j\}$. On the other hand, given a k-way cut E', there is a k-partition $\mathcal{P}'$ whose weight is no more than the weight of E'. Speaking precisely, $\mathcal{P}'$ consists of the vertex subsets of the components obtained by removing E' (if there are more than k components, simply bound some as one). Thus we obtain the next proposition.

Proposition 1.2. *The target split problem in hypergraphs of type 1* (Problem 1.4) *can be formulated as MPP with system* $(V, T, \frac{w(E)}{k} - w_{\text{in}}, k)$.

In this thesis, we call it *k-target-split problem in hypergraphs of type 1* (k-TPH-T1), and call the special case of $T = V$ *k-partition problem in hypergraphs of type 1* (k-PPH-T1).

In practice, there are two other types of k-target-split problems in hypergraphs. We denote them by k-TPH-T2 (type 2) and k-TPH-T3 (type 3) respectively. They differ from k-TPH-T1 by the objective function. Let $\mathcal{P} = \{V_1, V_2, \ldots, V_k\}$ be a k-partition of V. In the objective function of k-TPH-T1, the weight of a hyperedge e is counted only *once* if

$$p_e \triangleq \left|\{i \mid e \cap V_i \neq \emptyset\}\right| \geq 2;$$

otherwise it is not counted. In k-TPH-T2 (resp., k-TPH-T3), the weight of e is counted p_e (resp., $p_e - 1$) times (if $p_e \geq 2$). Notice that, when restricted to graphs, these three types of problems (k-TPH-T1, k-TPH-T2 and k-TPH-T3) reduce to the same problem k-TPG. Even in hypergraphs, there is no difference between them if $k = 2$.

We claim that, k-TPH-T2 and k-TPH-T3 can also be formulated as MPP. For this, let a function (the cut function) $w_{\text{ex}} : 2^V \to \mathbf{R}^+$ be defined by

$$w_{\text{ex}}(S) \triangleq \sum_{\emptyset \neq e \cap S \neq e} w(e), \qquad S \subseteq V.$$

(It was first introduced in considering k-TPG). It is not hard to see the next proposition.

Proposition 1.3. *Problems k-TPH-T2 and k-TPH-T3 can be formulated as MPP with systems (V, T, w_{ex}, k) and $(V, T, w_{\text{ex}} + w_{\text{in}} - \frac{w(E)}{k}, k)$, respectively.*

(A proof for submodularity can be found in Proposition 1.4). We denote the cases of $T = V$ by k-PPH-T2 and k-PPH-T3, respectively (which generalize k-PPG).

1.2.3 Historical background

Let us give a brief review on the historical background of the studies on MPP. A survey on complexity results and (approximation) algorithms is left to the next chapter.

To our knowledge, the first MPP is k-PPH-T2. It was considered by Lawler as early as 1970s, even prior to Pulleyblank's formulation [Pul 82] of k-PPG. Lawler [Law 62] first formulated a k-partition problem in hypergraphs using a monotone weight function on the vertex set. Later he [Law 73] gave the first precise definition of k-PPH-T2 (using the cut function). In the same article, several applications were suggested, which includes network analysis (where k-PPG was considered), information storage and retrieval, numerical taxonomy, and packaging of electronic circuits. However, at that time, he was unable to give good algorithms, which may be because the NP-hardness that was discovered later.

Several years later in 1983, Dalhaus, Johnson, Papadimitriou, Seymour and Yannakakis [DJPSY 83] considered k-MCP (in graphs). They showed the NP-hardness of k-MCP for any (fixed) $k \geq 3$. This result hence implies the NP-hardness of all the target-split versions of problems mentioned above.

The complexity of k-PPG was open for several years until Goldschmidt and Hochbaum [GH 88] showed that it is NP-hard even for unit edge weights. In the same article, they also showed that, however, it is solvable in polynomial time for any fixed k. This is quite different from k-MCP and k-TPG. Obviously, the NP-hardness of k-PPG implies the NP-hardness of all the problems mentioned above.

Problem k-TPH-T3 (including k-PPH-T3) is rather new. It arises from circuit partitioning in VLSI design. Circuit partitioning is a task of dividing a circuit into smaller components, with the objective to minimize the cost of connecting different components. It is known that, instead of graphs, hypergraphs are more natural representations of circuits in the layout process of VLSI design (see [Len 90]). For this, Chopra and Owen [CO 96, CO 99] considered the next problem. Given a hypergraph $H = (V, E)$ with a set $A \subseteq V$ and a weight function $w : E \to \mathbf{R}^+$, the *A-partition problem on hypergraphs* asks to find a $|A|$-partition $\mathcal{P} = \{V_1, V_2, \ldots, V_{|A|}\}$ satisfying $|A \cap V_i| = 1$ for all i, and minimizing $\sum_{e \in E} n_e w(e)$, where n_e, called the *cutsize* of e (see [Don 88]), is defined by

$$n_e \triangleq |\{i \mid e \cap V_i \neq \emptyset\}| - 1.$$

Clearly, this problem is a special case of k-TPH-T3, where $T = A$ and $k = |A|$ hold. It generalizes k-MCP, thus is NP-hard for any $k \geq 3$. Chopra and Owen considered several formulations with practical experiments, but approximation result was left open.

There is little literature on k-PPH-T1 and k-TPH-T1. Narayanan, Roy and Patkar [NRP 96] noticed a connection between k-PPH-T1 and k-PPG, which shows the NP-hardness of k-PPH-T1. In the same article, they also considered to approximate it from a theoretical view of point, with no mention on applications. An application in VLSI layout was presented by Hadley [Had 95]. We have seen an application in network reliability analysis before.

1.3 Survivable network design problem (SNDP)

1.3.1 Formulation and applications

In the previous section, we have seen how to formulate the problem of estimating the reliability of given networks as MPP. The converse question is, what is the least cost network that provides enough reliability? *Survivable network design problem* (SNDP) considers such questions. Before we formulate it, more notations are needed.

Paths in a hypergraph are said *hyperedge disjoint* if no two of them share a common hyperedge. Let $G = (V, U, E)$ be a bipartite graph, where U is the set of nonterminals. Paths in G (in the sense of hypergraph) are said *nonterminal disjoint*, or *U-disjoint*, if no two of them share a common nonterminal. Notice that they reduce to hyperedge disjoint paths in the corresponding hypergraph. This is illustrated in Figure 1.9.

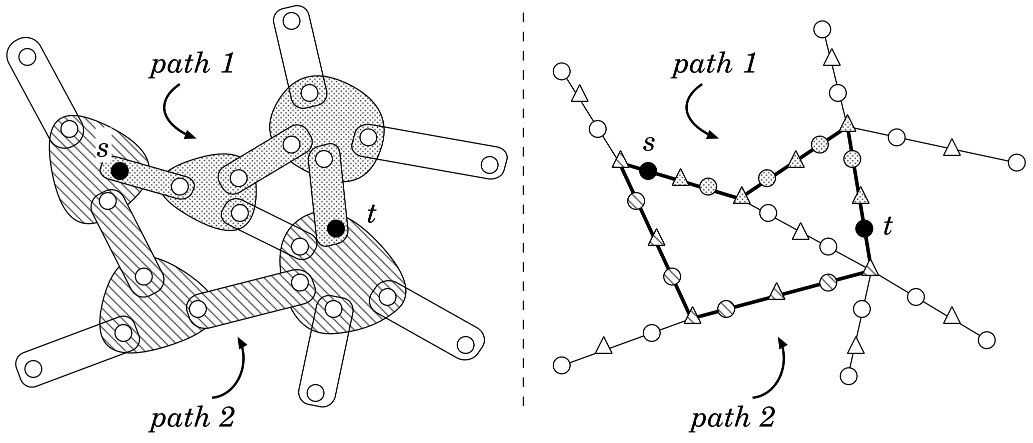


Figure 1.9: Hyperedge disjoint (s,t-) paths in a hypergraph and their corresponding nonterminal disjoint (s,t-) paths in the corresponding bipartite graph.

We first give a definition of SNDP.

Problem 1.5 (SNDP: hypergraph formulation). *Given a hypergraph $H = (V, E)$ with a weight function $w : E \to \mathbf{R}^+$ and connectivity requirements r_{st} for each pair of distinct vertices $s, t \in V$, find a minimum weight subset $E^* \subseteq E$ such that there are at least r_{st} hyperedge disjoint paths between each pair of distinct vertices s and t in the subgraph $H^* = (V, E^*)$ of H.*

Equivalently, we have an alternative description with the bipartite graph model.

Problem 1.6 (SNDP: bipartite graph formulation). *Given a bipartite graph $G = (V, U, E)$ with a weight function $w : U \to \mathbf{R}^+$ and connectivity requirements r_{st} for each pair of distinct terminals $s, t \in V$, find a minimum weight subset $U^* \subseteq U$ such that there are at least r_{st} nonterminal disjoint paths between each pair of distinct terminals $s, t \in V$ in the subgraph $G^* = G[T \cup U^*]$ of G.*

Let us see some applications. To avoid confusion, by "SNDP" we mean the general SNDP as formulated in Problem 1.5 (hypergraph model) or Problem 1.6 (bipartite graph model). On the other hand, by "SNDP in graphs" (SNDP-G), we refer the graph-restricted problem of SNDP formulated in Problem 1.5.

Some special cases of SNDP-G are well known, e.g., the *shortest path problem* (SPP). Given a graph with nonnegative edge weights (distances) and two vertices s and t, SPP asks to find a minimum weight path between s and t. It is easy to see that SPP can be formulated as SNDP-G with $r_{st} = 1$ (other requirements are zero). In fact, a minimal optimal solution (if all edges have positive weights, any optimal solution) of this SNDP-G must be a shortest s,t-path, and vice versa. We note that SPP can be solved by Dijkstra's algorithm ([Dij 59]) in $O(m + n \log n)$ time ([FT 87]), where n (resp., m) is the number of vertices (resp., edges).

Another example is the *minimum spanning tree problem* (MST). Given a graph with nonnegative edge weights, MST asks to find a minimum weight spanning tree. Problem MST can be formulated as SNDP-G with $r_{st} = 1$ for all pairs of vertices s and t. It is known that MST can be solved in $O(m + n \log n)$ time ([FT 87]).

A variant of MST, the *Steiner tree problem* (STP), is however NP-hard. Given a graph

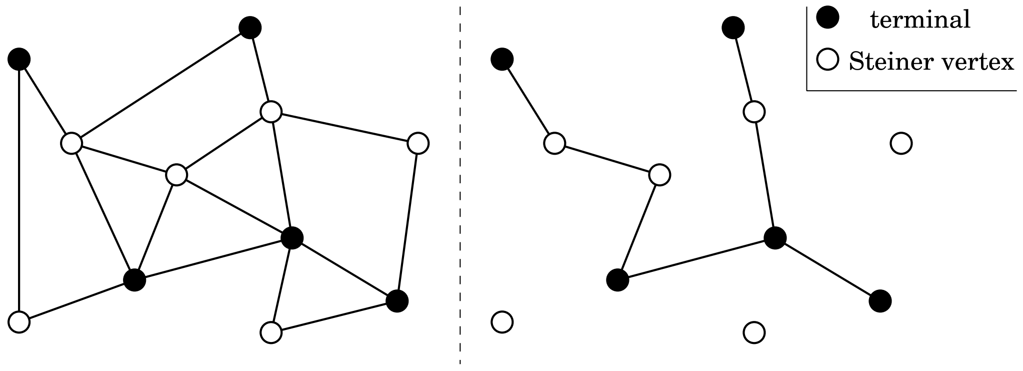


Figure 1.10: Graph and Steiner tree.

with nonnegative edge weights and a terminal set $T \subseteq V$, STP asks to find a minimum weight subset of edges that connects all vertices in T, where vertices in $V - T$ (called *Steiner* *vertices*) can also be used. This is illustrated in Figure 1.10. Notice that a minimal solution must be a tree. Obviously, STP can also be formulated as SNDP-G, in which $r_{st} = 1$ holds for $s, t \in T$ (0 otherwise). Karp [Karp 72] has shown the NP-hardness of STP even for unit edge weights. This result hence implies the NP-hardness of SNDP-G and SNDP.

A special case of SNDP-G with connectivity requirements in the set of $\{0, 1\}$ generalizes STP. It is often called the *generalized Steiner tree problem* or *Steiner forest problem* (notice that a minimal solution must be a forest). In this thesis, we refer it as *0-1 SNDP-G*.

Another important application is the *minimum k edge-connected spanning subgraph problem*, where the connectivity requirements are k for all pairs of distinct vertices. This problem is extensively studied in designing reliable networks. Not surprisingly, it is also NP-hard, even for $k = 2$ and graphs with unit edge weights (see, e.g., [Fer 98]).

Problem SNDP-G (with arbitrary connectivity requirements) is also called the *generalized Steiner network problem*. Notice that, each pair of vertices s and t are still connected after the removal of $r_{st} - 1$ edges in a feasible solution. In the words of network reliability, they are said to be able to "survive" the failure of at least $r_{uv} - 1$ edges. This is where the name (*survivable network design*) came from. For more about SNDP-G, we refer the readers to [GW 97, Wil 01].

1.3.2 More applications

We have seen some applications of SNDP-G. They are fundamental and important in both practice and theory combinatorial optimizations. However, as stated before, the graph model with only one kind of vertices may be too simple to fulfill the needs of applications. This motivated our study on hypergraph-based SNDP.

Let us first see an application, the *multicasting* in communication system. Consider a computer network that consists of computers, links and hubs (see Figure 1.4). A special computer (called the *server*) wants to send the *same data* to some other special computers (called *clients*). This is the case of Internet/intranet radio, TV, or other streaming broadcast systems. In the traditional method, called *unicast*, the server is required to send the data *once per client*. This is illustrated in Figure 1.11.

Since the same data is sent as many time as the number of clients, this method of unicast may result in large, unaffordable network traffics in total. In order to do it more efficiently, *multicast* is proposed recently. By multicasting, the server only needs to send the data *once*. It is the responsibility of hubs in the multicasting route to send (if needed, copy and send) the data to the next hub until it reaches the clients. This is also illustrated in Figure 1.11. It is clear that multicast is more efficient in reducing the network traffic. In fact, it is considered as one of the core technologies in networks of the next generation, see [KK 98].

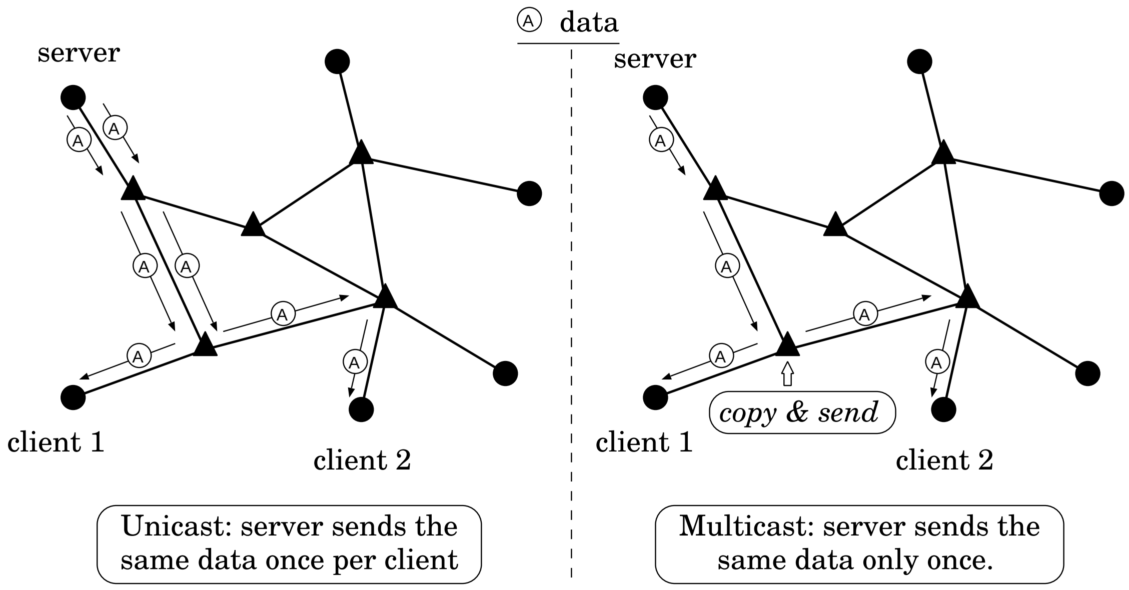

A data
server
A
client 1
client 2
Unicast: server sends the same data once per client
server
copy & send
client 1
client 2
Multicast: server sends the same data only once.

Figure 1.11: Unicast and multicast in communication network.

From the viewpoint of efficiency, it is of our interest to find an optimal multicasting route. Obviously, this is a problem of finding a minimum subnetwork to connect given nodes in a network. We formulate it as the next *multicasting problem.*

Problem 1.7 (Multicasting problem). *Given a graph $G = (V, E)$ with a server $s \in V$ and a set $T \subseteq V$ of clients, where other vertices and edges have nonnegative weights, find a minimum weight subgraph of G that contains at least one s,t-path for each client $t \in T$.*

In practice, the weight may represent transmission or switching time, or simply one if we are interested in minimizing the number of intermediate hub/links.

Now let us show that it can be formulated as SNDP. Similarly as before (see Subsection 1.2.2), by treating edges as nonterminals and inserting terminals between two adjacent nonterminals, we can have a bipartite graph model in which the server and clients are terminals and only nonterminals have weights. An optimal multicasting is thus obtained by solving the SNDP problem with $r_{st} = 1$ for each $t \in T$ (0 otherwise). Obviously, a minimal solution must be a tree, which is often called the *multicast tree* or *distribution tree.*

The multicasting problem also has applications in designing reliable networks. Suppose that we have some computers that need to be connected. Assume that we know in advance the places suitable for installing hub/links and the cost for each installation. The problem is: from given candidate places for installation, find a minimum cost plan that connects the computers. Giving each nonterminal/edge (i.e., candidate) with the installation cost, we see that this problem can be formulated as SNDP with $r_{st} = 1$ for distinct computers s and t. This is equivalent to the multicasting problem.

Using hypergraph model, multicasting problem can be described as to find a minimum weight set of hyperedges that connects given vertices. Thus it captures a hypergraph generalization of STP (Steiner tree problem). We note that, while a minimal solution of STP is a tree, it is not true for this generalization (in hypergraph). In fact, even to determine the existence of a spanning tree (i.e., a spanning subgraph in which there is a unique path between each pair of vertices) in a general hypergraph is known to be NP-complete ([TZ 94, War 98]). See Figure 1.12 for an illustration.

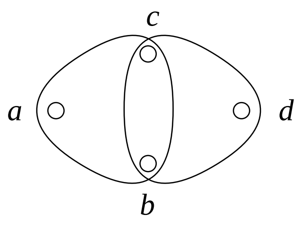


Figure 1.12: Minimum connected spanning subgraph in hypergraph may not be a tree. There are two paths ($b, \{a, b, c\}, c$ and $b, \{d, b, c\}, c$) between vertices b and c.

In this thesis, we consider the general SNDP problem. Note that, any two vertices s and t are still connected after the removal of at most $r_{st} - 1$ hyperedges in a feasible solution.

1.4 Preliminary of set functions in hypergraphs

Let us see several submodular or supermodular functions defined in hypergraphs, which are heavily used in this thesis.

Proposition 1.4. *Let $H = (V, E)$ be a hypergraph with a vertex set V, a hyperedge set E and a nonnegative hyperedge weight function $w : E \to \mathbf{R}^+$. Let two functions $w_{\text{ex}} : 2^V \to \mathbf{R}^+$ and $w_{\text{in}} : 2^V \to \mathbf{R}^+$ be defined by*

$$w_{\text{ex}}(S) \triangleq \sum_{e \in \delta(S)} w(e) = \sum_{\emptyset \neq e \cap S \neq e} w(e), \tag{1.7}$$

$$w_{\text{in}}(S) \triangleq \sum_{e \subseteq S} w(e) \tag{1.8}$$

for all sets $S \subseteq V$. Then function w_{ex} is symmetric and submodular, whereas functions $w_{\text{ex}} + w_{\text{in}}$ and w_{in} are submodular and supermodular, respectively. In particular, function $|\delta| : 2^V \to \mathbf{R}^+$ defined by $|\delta|(S) = |\delta(S)|$ for all sets $S \subseteq V$ is symmetric and submodular.

Proof. The submodularity of w_{ex} is well known, see, e.g., [KW 96]. We give a proof for completeness. Let X and Y be two subsets of V. We have

$$\begin{aligned}
& w_{\mathrm{ex}}(X) + w_{\mathrm{ex}}(Y) - w_{\mathrm{ex}}(X \cap Y) - w_{\mathrm{ex}}(X \cup Y) \\
= \; & \sum_{\emptyset \neq e \cap X \neq e} w(e) \; + \sum_{\emptyset \neq e \cap Y \neq e} w(e) \; - \sum_{\emptyset \neq e \cap (X \cap Y) \neq e} w(e) \; - \sum_{\emptyset \neq e \cap (X \cup Y) \neq e} w(e) \\
= \; & \sum \Big\{ w(e) \mid e \cap (X - Y) \neq \emptyset \neq e \cap (Y - X), \; e \cap (X \cap Y) = \emptyset \Big\} \\
& + \sum \Big\{ w(e) \mid e \cap (X - Y) \neq \emptyset \neq e \cap (Y - X), \; e \subseteq (X \cup Y) \Big\} \; \geq \; 0 .
\end{aligned}$$

Thus function w_{ex} is submodular. Clearly it is also symmetric. Let us next show that functions $w_{\mathrm{ex}} + w_{\mathrm{in}}$ and w_{in} are submodular and supermodular, respectively. Notice that, by definition,

$$(w_{\mathrm{ex}} + w_{\mathrm{in}})(S) \; = \sum_{e \cap S \neq \emptyset} w(e) \; = \; w_{\mathrm{in}}(V) - w_{\mathrm{in}}(V - S) \quad \forall S \subseteq V.$$

Thus, by Proposition 1.1, it is sufficient to show that function w_{in} is supermodular. This can be seen by

$$\begin{aligned}
& w_{\mathrm{in}}(X) + w_{\mathrm{in}}(Y) - w_{\mathrm{in}}(X \cap Y) - w_{\mathrm{in}}(X \cup Y) \\
= \; & \sum_{e \subseteq X} w(e) \; + \sum_{e \subseteq Y} w(e) \; - \sum_{e \subseteq (X \cap Y)} w(e) \; - \sum_{e \subseteq (X \cup Y)} w(e) \\
= \; & - \sum \Big\{ w(e) \mid e \cap (X - Y) \neq \emptyset \neq e \cap (Y - X), \; e \subseteq (X \cup Y) \Big\} \; \leq \; 0 .
\end{aligned}$$

Therefore functions $w_{\mathrm{ex}} + w_{\mathrm{in}}$ and w_{in} are submodular and supermodular, respectively. Taking the case of weight function $w' \equiv 1$, we see that $|\delta| = w'_{\mathrm{ex}}$ is symmetric and submodular. $\square$

Finally, we note that function w_{ex} is called the *cut function*.

Chapter 2

Greedy Splitting Algorithm for MPP

2.1 Introduction

We first give a survey on MPP. Due to the difference in complexities, we review two kinds of problems individually. One is to find a minimum weight k-partition, called *MPP with no target* (MPP-NT). The other is *MPP with target*, which is simply (the general) MPP.

2.1.1 Survey on MPP-NT (MPP with no target)

Let V be a finite set. If function $f : 2^V \to \mathbf{R}$ is submodular, we call the pair (V, f) *submodular system*. Analogous notations are used for *nonnegative submodular systems*, *symmetric submodular system*, etc. Recall that MPP-NT has the next formulation.

Problem 2.1 (MPP-NT). *Given a system (V, f, k), where (V, f) is a submodular system and $k \geq 1$ is an integer, find a family $\mathcal{P} = \{V_1, V_2, \ldots, V_k\}$ such that*

$$\begin{aligned}
\text{minimize} \quad & f(V_1) + f(V_2) + \cdots + f(V_k) \\
\text{subject to} \quad & V_1 \cup V_2 \cup \cdots \cup V_k = V, && (2.1) \\
& V_i \cap V_j = \emptyset, \quad 1 \leq i \neq j \leq k, && (2.2) \\
& V_i \neq \emptyset, \qquad i = 1, 2, \ldots, k. && (2.3)
\end{aligned}$$

Recall that a family $\mathcal{P} = \{V_1, V_2, \ldots, V_k\}$ satisfying (2.1) – (2.3) is called k-partition (of V). We assume that function f is given by an oracle that returns $f(S)$ for a given set $S \subseteq V$.

Observation 2.1. *If function f is changed to $f + c$, where $c \in \mathbf{R}$ is a constant, then the weight of a k-partition increases (or decreases) by kc.*

Notice that kc is not dependent on the k-partition. Hence an optimal solution with respect to f is also an optimal solution with respect to $f + c$. However, this is inappropriate for defining performance guarantee and may not be interesting in practical applications (e.g., a bigger c yields a better guarantee). Throughout this thesis, we assume that the optimum is nonnegative. We consider the next problem classes.

Problem 2.2 (k-PPSS). *The k-partition problem in submodular systems (k-PPSS) is to find a minimum k-partition of V for a given nonnegative submodular system (V, f).*

In short, k-PPSS is MPP-NT in nonnegative submodular system.

Problem 2.3 (k-PPSSS). *The k-partition problem in symmetric submodular systems (k-PPSSS) is k-PPSS with symmetric function f.*

Problem 2.4 (k-PPMSS). *The k-partition problem in monotone submodular systems (k-PPMSS) is k-PPSS with monotone function f.*

In Chapter 1, we have seen k-PPH-T2 (k-partition problem in hypergraphs of type 2), and k-PPG (k-partition problem in graphs).

Problem 2.5 (k-PPH-T2, k-PPG). *Given a hypergraph $H = (V, E)$ with a weight function $w : E \to \mathbf{R}^+$, problem k-PPH-T2 is k-PPSSS with system (V, w_{ex}, k), where w_{ex} is the cut function of H. Problem k-PPG is k-PPH-T2 restricted to graphs.*

(Notice that, by Proposition 1.4, a cut function is nonnegative, symmetric and submodular.)

Obviously, we have the next inclusion relationship among the above problem classes.

$$k\text{-PPG} \subset k\text{-PPH-T2} \subset k\text{-PPSSS} \subset k\text{-PPSS}. \tag{2.4}$$

Later we will see that, in a sense slightly different from the above,

$$k\text{-PPG} \subset k\text{-PPMSS} \subset k\text{-PPSS}.$$

Our study starts from k-PPG. As noted in Chapter 1, it is NP-hard even for unit edge weights ([GH 88]). On the other hand, it is polynomial time solvable for any fixed k. More precisely, the running time is $O(n^{k^2/2-3k/2+4}F(m,n))$ for a graph with m edges and n vertices ([GH 88]), where $F(m,n)$ stands for the running time of a maxflow algorithm (for network flow theory, see e.g., [AMO 93]). It is known that $F(m,n) = O(mn\log(n^2/m))$ ([GT 88]).

Afterwards, Karger and Stein [KS 96] developed a randomized algorithm that can find an optimal solution of k-PPG with high probability in $O(n^{2(k-1)}\log^3 n)$ time. Kamidoi, Wakabayasi and Yoshida [KWY 97] proposed another deterministic algorithm and claim (with no full proof) that it runs in $O(n^{2(k-2)}F(m,n))$ time. Goldschmidt and Hochbaum [GH 90] found a linear time algorithm which "almost certainly" solve k-PPG in a kind of "random graphs". Dalhaus *et al.* [DJPSY 83] showed an $O(n^{O(k)})$ time algorithm for k-PPG in planar graphs, where a graph is planar if it can be drawn in the plane such that no pair of edges intersect. This result is later improved by Hartvigsen [Har 93] to $O(n^{2k-1})$.

For small k, there are more efficient algorithms. Problem 2-PPG, more commonly known as the *minimum cut problem*, is shown to be solvable in $O(mn + n^2 \log n)$ time due to Nagamochi and Ibaraki [NI 92], later simplified by Stoer and Wagner [SW 94]. An algorithm due to Hao and Orlin [HO 94] also solves 2-PPG in $O(mn \log(n^2/m))$ time. Random algorithms in [KS 96] and [Kar 96] can find a minimum cut with high probability in $O(n^2 \log^3 n)$ and $O(m \log^3 n)$ time, respectively. There are reports [CGKLS 97, JRT 00] on practical performance of these algorithms. It is also known that an algorithm due to Gabow [Gab 95] can solve 2-PPG in graphs with unit edge weights in $O(m + \lambda^2 n \log(n/\lambda))$ time, where λ is the size of a minimum cut (called *edge connectivity*; observe that $\lambda \leq 2m/n$ holds).

For $k \in \{3\}, \{3, 4\}$, Kapoor [Kap 96] and Kamidoi, Wakabayashi and Yoshida [KWY 02] showed that k-PPG can be solved in $O(n^k F(m, n))$ time. An improved $O(mn^3)$ time algorithm for 3-PPG is found by Burlet and Goldschmidt [BG 97]. For $k \in \{3, 4\}$, Nagamochi and Ibaraki [NI 00] proposed improved $O(n^{k-1} F(m, n))$ time algorithms. Later the results are extended to $k \in \{5, 6\}$ by Nagamochi, Katayama and Ibaraki [NKI 00]. Using their idea, Levine [Lev 00] found an $O(mn^{k-2} \log^3 n)$ time random algorithm (of Monte Carlo type) for $k \in \{3, 4, 5, 6\}$. For planar graphs, Hochbaum and Shmoys [HS 85] showed that 3-PPG with unit edge weights can be solved in $O(n^2)$ time. Later it is improved to $O(n \log n)$ by He [He 91]. For $k \in \{3\}$, $\{3, 4\}$ and $\{5, 6\}$, k-PPG in planar graphs with arbitrary edge weights can be solved in $O(n^k)$ time ([Har 93], [NI 00] and [NKI 00]).

From the NP-hardness of k-PPG, it is clear that k-PPH-T2, k-PPSSS and k-PPSS are also NP-hard. Queyranne [Que 99] claimed (where no full proof is available) that, for any fixed k, k-PPSSS is solvable in $O(|V|^{k^2} \theta)$ time, where θ is the time bound of the value oracle for f. Notice that, for k-PPH-T2 in which f is the cut function, we can estimate $f(S) = \sum_{\emptyset \neq e \cap S \neq e} w(e)$ in $\theta = O(D)$ time for all sets $S \subseteq V$ in a straightforward manner, where D is the sum of the degrees of hyperedges,

On the other hand, no similar result is known for k-PPSS except for $k = 2$. Problem 2-PPSS is shown to be solvable in $O(|V|^3 \theta)$ time due to Queyranne [Que 95]; see also [NI 98]. Of course, this also holds for 2-PPSSS. In particular, 2-PPH-T2 can be solved in $O(n^3 D)$ time. The last result is later improved by Klimmek and Wagner [KW 96] to $O(n^2 \log n + nD)$ (notice that we can assume $n \leq D$ without loss of generality). Another algorithm for 3-PPSSS is obtained in [NI 00].

In Chapter 1, we have introduced other two problems in hypergraphs: k-PPH-T1 and k-PPH-T3. Let $H = (V, E)$ be a hypergraph, and $\mathcal{P} = \{V_1, V_2, \ldots, V_k\}$ be a k-partition of V. The objective function in k-PPH-T1 (resp., k-PPH-T3) counts the weight of a hyperedge e

once (resp., $p_e - 1$ times) if $p_e = |\{i \mid e \cap V_i \neq \emptyset\}| \geq 2$. We have shown in Proposition 1.3 that, k-PPH-T1 and k-PPH-T3 can be formulated as MPP with systems $(V, V, \frac{w(E)}{k} - w_{\text{in}}, k)$ and $(V, V, w_{\text{ex}} + w_{\text{in}} - \frac{w(E)}{k}, k)$, respectively. (Functions w_{ex} and w_{in} are defined in Proposition 1.4.)

Problem 2.6 (k-PPH-T1, k-PPH-T3). *Given a hypergraph $H = (V, E)$ with a weight function $w : E \to \mathbf{R}^+$, problems k-PPH-T1 and k-PPH-T3 are MPP-NT with systems $(V, \frac{w(E)}{k} - w_{\text{in}}, k)$ and $(V, w_{\text{ex}} + w_{\text{in}} - \frac{w(E)}{k}, k)$, respectively.*

Currently, there is no result of showing polynomial time algorithm or the NP-hardness for these two problems even for fixed $k \geq 3$. As noted in Chapter 1, in graphs, they reduce to k-PPG. Also we have 2-PPH-T1 = 2-PPH-T3 = 2-PPH-T2.

No literature is available for k-PPMSS so far. We are not aware much of its practical applications. Considering the complexity of k-PPMSS, recall that we claimed that k-PPG is a special case of k-PPMSS. Let us see this in the following.

Given a hypergraph $H = (V, E)$ with a nonnegative hyperedge weight function w, k-PPH-T3 employs function $w_{\text{ex}} + w_{\text{in}} - \frac{w(E)}{k}$, which is monotone and submodular, but may not be nonnegative. Nevertheless, since it is equivalent to employing function $w_{\text{ex}} + w_{\text{in}}$, we see that k-PPH-T3 is a special case of k-PPMSS. Hence we obtain the next inclusion.

$$k\text{-PPG} \subset k\text{-PPH-T3} \subset k\text{-PPMSS} \subset k\text{-PPSS}. \tag{2.5}$$

This shows that k-PPMSS is also NP-hard. We note that, for applications, the rank function of a polymatroid is nonnegative, monotone and submodular. Furthermore, a nonnegative, monotone and submodular function f (with $f(\emptyset) = 0$) is the rank function of certain polymatroid. Thus k-PPMSS can be viewed as multiway partition problem in polymatroids, which suggests potential applications since polymatroid is a large class of systems. We will not go further details on polymatroid and its rank function, see e.g., [KV 00].

We now summarize the complexities of MPP-NTs in Table 2.1.

From Table 2.1, we see that, even if a problem (e.g. k-PPG) is solvable for fixed k, the time complexity grows exponentially with respect to k. Thus it is of great interest to design efficient approximation algorithms whose running time are also polynomials with respect to k. The history of studies on approximation algorithms for these problems is rather short, actually, only one decade.

Saran and Vazirani [SV 91] proposed the first approximation algorithm for k-PPG. Actually they provided two. The first is based on the so-called *cut-tree* (or *Gomory-Hu tree*) structure of graphs (see [GH 61]) and runs in $O(nF(m,n))$ time. The second is a greedy algorithm that splits the graph in $k-1$ phases, where in each phase it increases the size of partition by one. By a sophisticated proof using cut-tree structure, they showed that both algorithms achieve the same performance guarantee of $2 - \frac{2}{k}$. Since at that time, people did

Table 2.1: Complexities of MPP-NTs

Problem	Arbitrary k	Time complexity for fixed k
k-PPG	NP-hard [GH 88]	$O(n^{k^2})$* [GH 88] (better in special cases)
k-PPSSS	$\downarrow$	$O(\lvert V\rvert^{k^2}\theta)$† [Que 99]‡
k-PPSS	$\downarrow$	for $k=2$, $O(\lvert V\rvert^3\theta)$ [Que 95]
k-PPH-T1	NP-hard	for $k=2$, same as 2-PPH-T2
k-PPH-T2	NP-hard	for $k=2$, $O(n^2\log n + nD)$§ [KW 96]
k-PPH-T3	NP-hard	for $k=2$, same as 2-PPH-T2
k-PPMSS	$\downarrow$	for $k=2$, see k-PPSS

*n: number of vertices.

†θ: time bound of the value oracle for function f.

‡No full proof available.

§D: sum of degrees of hyperedges.

not know how to compute a minimum cut efficiently, they were unable to give an efficient implementation for the greedy algorithm.

Later Kapoor [Kap 96] pointed out that the greedy algorithm in [SV 91] can be implemented by $O(k)$ minimum cut computations in $O(mn + n^2\log n)$ time per phase. Therefore it yields a total running time of $O(kn(m + n\log n))$. He further claimed that, for any fixed $j \geq 2$, k-PPG can be approximated within factor $2 - \frac{j+1}{k} + \frac{j-1}{k^2} + O(\frac{j}{k^3})$ in polynomial time. However, the correctness proof is not complete, since it contains a lemma which is not valid in general (we will discuss it in detail in Chapter 3).

Recently, Naor and Rabani [NR 01] proposed a 2-approximation algorithm that is based on an LP (*linear programming*; see e.g., [Sch 98]) relaxation. While their algorithm does not give an improved guarantee nor a faster running time, it is of interest in the sense that all the other algorithms for k-PPG are combinatorial. We now know that approximation results for many problems can be improved if good LP relaxations were found. We will see an example k-MCP in the next subsection.

For k-PPH-T1 and k-PPH-T3, Narayanan, Roy and Patkar [NRP 96] considered approximation algorithms via the so-called *principal partition* structure of graph (see [Nar 91]). Using a complicated proof (without running time analysis), they showed the performance guarantees of $d_{\max}(1 - \frac{1}{n})$ and $2 - \frac{2}{n}$ for k-PPH-T1 and k-PPH-T3 respectively, where $d_{\max}$ is the maximum degree of hyperedges and n is the number of vertices.

Several years later, Queyranne [Que 99] pointed out that, the greedy algorithm in [Kap 96, SV 91] can be extended for approximating k-PPSSS. He claimed that a performance guarantee of $2 - \frac{2}{k}$ can be obtained. We note that Queyranne's proof also uses a lower bound derived from the cut-tree structure (for symmetric submodular systems; see [Que 93, GR 95]), which is rather complicated and does not work for general submodular systems (see [Ben 95]). As will be seen later, our approach does not need the cut-tree structure and works for any submodular system. In a much simpler way, we will show the same or improved results compared with those obtained so far by [Kap 96, NRP 96, SV 91, Que 99].

We summarize the previous best-known approximation results in the next table. (None was known for k-PPMSS and k-PPSS.) New results obtained in this thesis will be summarized later in Table 6.1 in Chapter 6.

Table 2.2: Previous best-known approximation results for MPP-NT

Problem	Reference	Guarantee	Running time	Proved by
k-PPG	[Kap 96, SV 91]	$2-2/k$	$O(kn(m+n\log n))^*$	cut-tree
k-PPH-T1	[NRP 96]	$d_{\max}(1-1/n)^\dagger$	N/A	principal partition
k-PPH-T2	[Que 99]	$2-2/k$	$O(kn^3D)^\ddagger$	see k-PPSSS
k-PPH-T3	[NRP 96]	$2-2/n$	N/A	principal partition
k-PPSSS	[Que 99]	$2-2/k$	$O(k\|V\|^3\theta)^\S$	cut-tree

$^*m$ (resp. n): number of edges (resp. vertices).

$^\dagger d_{\max}$: maximum degree of hyperedges.

$^\ddagger D$: sum of degrees of hyperedges.

$^\S\theta$: time bound of the value oracle for function f.

2.1.2 Survey on general MPP

Recall that the general MPP is formulated as the next problem.

Problem 2.7 (MPP). *Given a system (V,T,f,k), where V is a finite set, $T\subseteq V$ called target set, $f:2^V\to\mathbf{R}$ is a submodular function and k is an integer, MPP is defined by*

$$\begin{array}{lll} \text{minimize} & f(V_1)+f(V_2)+\cdots+f(V_k) & \\ \text{subject to} & V_1\cup V_2\cup\cdots\cup V_k=V, & (2.6)\\ & V_i\cap V_j=\emptyset,\quad 1\le i\ne j\le k, & (2.7)\\ & V_i\cap T\ne\emptyset,\quad i=1,2,\ldots,k. & (2.8)\end{array}$$

Recall that a feasible solution of MPP, i.e. a k-partition of V satisfying (2.8), is called *k-target-split* (with respect to T). We assume that $|T|\ge k$ holds and the function f is given by an oracle. We further assume that the optimum is nonnegative. Similarly as before, we consider the next problem classes.

Problem 2.8 (k-TPSS). *Given a nonnegative submodular system (V, f) with a target set $T \subseteq V$, the k-target-split problem in submodular systems (k-TPSS) asks to find a minimum k-target-split of V.*

Problem 2.9 (k-TPSSS). *The k-target-split problem in symmetric submodular systems (k-TPSSS) is k-TPSS with symmetric function f.*

Problem 2.10 (k-TPMSS). *The k-target-split problem in monotone submodular systems (k-TPMSS) is k-TPSS with monotone function f.*

Problem 2.11 (k-TPH-T2, k-TPG). *Given a hypergraph $H = (V, E)$ with a target set $T \subseteq V$ and a weight function $w : E \to \mathbf{R}^+$, problem k-TPH-T2 is k-TPSSS with system (V, T, w_{ex}, k), where w_{ex} is the cut function of H. Problem k-TPG is k-TPH-T2 restricted to graphs.*

Problem 2.12 (k-TPH-T1, k-TPH-T3). *Given a hypergraph $H = (V, E)$ with a target set $T \subseteq V$ and a weight function $w : E \to \mathbf{R}^+$, problems k-PPH-T1 and k-PPH-T3 are MPP with systems $(V, T, \frac{w(E)}{k} - w_{\text{in}}, k)$ and $(V, T, w_{\text{ex}} + w_{\text{in}} - \frac{w(E)}{k}, k)$, respectively.*

We have the next inclusion among the above problem classes.

$$k\text{-TPG} \subset k\text{-TPH-T2} \subset k\text{-TPSSS} \subset k\text{-TPSS} \tag{2.9}$$
$$k\text{-TPG} \subset k\text{-TPH-T3} \subset k\text{-TPMSS} \subset k\text{-TPSS}. \tag{2.10}$$

Recall that k-TPG with $|T| = k$ is called k-MCP (multiterminal cut problem). Clearly, k-PPG can be solved by at most $\binom{|V|}{|T|} = O(n^k)$ k-MCP computations, by considering all possible combinations of targets. This suggests that k-MCP may be more difficult than k-PPG, in the sense of existence of polynomial time algorithms. In fact, Dalhaus *et al.* [DJPSY 83] showed that, unless P=NP, there is no approximation algorithm for 3-MCP with performance guarantee better than $1+\epsilon$ for some constant $\epsilon > 0$. They also showed that, k-MCP in planar graphs is solvable in $O(n^{O(k)})$ time, which implies that k-PPG (and k-TPG) in planar graphs is solvable in $O(n^{O(k)})$ time. They gave a combinatorial approximation algorithm for k-MCP with a performance guarantee of $2 - \frac{2}{k}$.

In the next fifteen years since [DJPSY 83], there was no better result for k-MCP until a revolutionary algorithm was proposed by Călinescu, Karloff and Rabani [CKR 98]. They proposed a novel geometric relaxation, and use it to obtain a $(\frac{3}{2} - \frac{1}{k})$-approximation algorithm. Later Cunningham and Tang [CT 99] showed that, for $k = 3$, the approximation guarantee $\frac{3}{2} - \frac{1}{3} = \frac{7}{6}$ can be improved to $\frac{12}{11}$, which is the best possible as long as the geometric relaxation is used. The same result is independently obtained by Karger, Klein, Stein, Thorup and Young [KKSTY 99]. Karger et al. further showed that, for $k = 4$, 5, ≥ 6, the performance guarantee can be improved to 1.1539, 1.2161 and 1.3438, respectively. These are the best known results for k-MCP. For studies related to k-MCP, see [BTV 99, CO 96, CO 99, CR 91, EFS 99, ES 94, FK 00, GVY 94, Har 98, NZ 97, Yeh 01]. We note that, the geometric relaxation in [CKR 98] exploits the fact that there are exactly k targets, which is not valid for the general k-TPG problem.

Problem 2-MCP is well-known as the *minimum s,t-cut problem* for $T = \{s, t\}$. It is well-known that this problem can be solved by a single maximum flow computation ([FF 56]), as the famous *maxflow-mincut theorem*. Numerous applications have been found for maxflow/mincut problem (see [AMO 93]). It has been extensively studied and various algorithms have been developed so far (see also [AMO 93]). It can be solved in $O(mn \log(n^2/m))$ time in a general graph ([GT 88]), in $O(\min\{m^{\frac{1}{2}}, n^{\frac{2}{3}}\}n)$ time in a graph with unit edge weights ([ET 75]), and in $O(n \log n)$ time in a planar graph ([AMO 93]), where n and m are the numbers of vertices and edges respectively.

Little is known for problems in hypergraph. Garg, Vazirani and Yannakakis [GVY 94] considered a *node weighted* version of k-MCP, where, instead of edges, vertices have nonnegative weights. It is easy to see that this problem is equivalent to k-TPH-T1 with $|T| = k$. Garg et al. [GVY 94] gave a $(2 - \frac{2}{k})$-approximation algorithm. Their algorithm is also based on LP relaxation and exploits the fact that there are exactly $|T| = k$ targets, hence does not apply to the general k-TPH-T1 problem. No more literature is known.

The literature is even poorer for problems in general submodular systems. The only known result is that 2-TPSS (hence 2-TPSSS and 2-TPMSS), can be solved by polynomial time calls to the value oracle for function f (see Lemma 2.3).

While all the previous literatures treat k-MCP (in graph or hypergraph), Maeda, Nagamochi and Ibaraki [MNI 93] considered the first target split problem k-TPG, in which they showed that it can be approximated within a factor of $2 - \frac{2}{k}$. As far as we know, there is no more result available for other target split versions of problems.

2.2 Greedy splitting algorithm (GSA) for MPP-NT

In this section, we present a framework for approximating MPP-NT. The idea is to greedily increase the size of partition and use submodularity to derive performance guarantee.

2.2.1 Greedy splitting approach

We first give a general description of the greedy splitting approach. The idea is natural: if it is hard to get an optimal solution, let us greedily construct a near-optimal one in several phases. We start with $\mathcal{P}_1 = \{V\}$. In the i^{th} phase, a p_{i+1}-partition $\mathcal{P}_{i+1}$ is constructed by splitting some member(s) of the previously obtained p_i-partition $\mathcal{P}_i$, where of course $p_{i+1} > p_i$

holds. This is illustrated in Figure 2.1. The splitting process stops as soon as a k-partition is obtained.

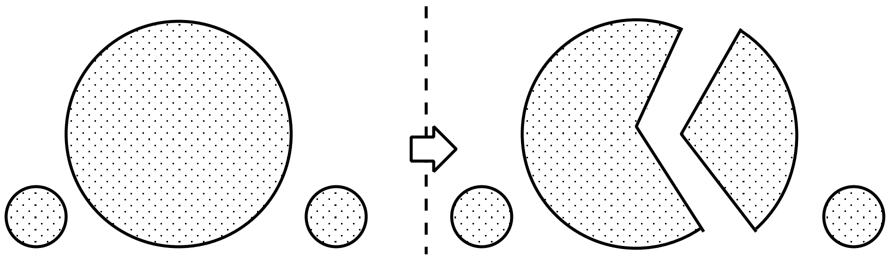


Figure 2.1: Splitting a p_i-partition to obtain a p_{i+1}-partition. Greedy algorithm tries to minimize the weight increase.

Since it is desired to get a solution with weight as small as possible, we want to find such a way of splitting that minimizes the weight increase, or *splitting weight* that is $f(\mathcal{P}_{i+1}) - f(\mathcal{P}_i)$ in each phase i. Actually we do not know how to do this efficiently for arbitrary numbers of p_{i+1} and p_i. Nevertheless, we can do it in some cases, e.g., when $p_{i+1} - p_i \equiv 1$ holds. This is the subject of the next subsection.

2.2.2 Algorithm description and main lemma

We first observe that 2-PPSS and a variant of it can be solved efficiently.

Theorem 2.1 (Queyranne [Que 95]). *Given a symmetric submodular system (V, g), where $|V| \geq 2$ holds, a nonempty proper subset S^* of V such that $g(S^*)$ is minimum can be found in $O(|V|^3\theta_g)$ time, where θ_g is the time bound of the oracle for g.* □

Theorem 2.2. *Given a submodular system (V, f) and a set $W \subseteq V$ satisfying $|W| \geq 2$, a nonempty proper subset S^* of W such that $f(S^*) + f(W - S^*)$ is minimum can be found in $O(|W|^3\theta)$ time, where θ is the time bound of the oracle for f.*

Proof. Consider a system (W, g), where function $g\colon 2^W \to \mathbf{R}$ is defined by $g(S) = f(S) + f(W - S)$ for all sets $S \subseteq W$. Notice that function g is symmetric and submodular, and we can estimate $g(S)$ in at most 2θ time for any set $S \subseteq W$ (by two calls to the oracle for f). Obviously, $f(S^*) + f(W - S^*)$ is minimum if and only if $g(S^*)$ is minimum. Theorem 2.1 then shows that such a set S^* can be found in $O(|W|^3\theta)$ time. □

We now present the *greedy splitting algorithm* (GSA) in Figure 2.2. (Note that "argmin" means the value of argument that minimizes the objective.)

GSA contains $k - 1$ phases. It starts with $\mathcal{P}_1 = \{V\}$. In the i^{th} phase, we compute a pair (S_i, W_i) that minimizes the splitting weight $f(S) + f(W - S) - f(W)$ over all S and W

Input: A submodular system (V, f) and an integer $k \geq 1$.
Output: A k-partition $\mathcal{P}_k$ of V.

```
1  $\mathcal{P}_1 \leftarrow \{V\}$
2  for $i = 1, \ldots, k-1$ do
3      $(S_i, W_i) \leftarrow \text{argmin}\,\{f(S) + f(W-S) - f(W) \mid \emptyset \neq S \subset W,\ W \in \mathcal{P}_i\}$
4      $\mathcal{P}_{i+1} \leftarrow (\mathcal{P}_i - \{W_i\}) \cup \{S_i, W_i - S_i\}$
5  end /* for */
```

Figure 2.2: Greedy splitting algorithm (GSA) for MPP-NT

satisfying $\emptyset \neq S \subset W$ and $W \in \mathcal{P}_i$. We then get $\mathcal{P}_{i+1}$ from $\mathcal{P}_i$ by replacing W_i with S_i and $W_i - S_i$. It is obvious that for all $\ell = 1, 2, \ldots, k$, the weight of ℓ-partition $\mathcal{P}_\ell$ is

$$f(\mathcal{P}_\ell) \ = \ f(V) + \sum_{i=1}^{\ell-1}\Big(f(S_i) + f(W_i - S_i) - f(W_i)\Big). \tag{2.11}$$

Clearly the output $\mathcal{P}_k$ is a k-partition of V. Let us consider the running time of GSA. For any (fixed) $W \subseteq V$, Theorem 2.2 shows that, we can find a nonempty set $S^* \subset W$ such that $f(S^*) + f(W - S^*)$ (hence $f(S^*) + f(W - S^*) - f(W)$) is minimum in $O(|W|^3\theta)$ time. Thus we can execute Line 3 in $\sum_{W \in \mathcal{P}_i} O(|W|^3\theta) = O(\sum_{W \in \mathcal{P}_i} |W|^3\theta) = O(|V|^3\theta)$ time in the i^{th} phase (notice $|V| = \sum_{W \in \mathcal{P}_i} |W|$). This implies that the running time of GSA is $O(k|V|^3\theta)$.

We note that GSA can be made faster. Suppose that, before the i^{th} phase we have computed an S_W for each $W \in \mathcal{P}_i$ that minimizes $f(S) + f(W - S) - f(W)$. Then in the i^{th} phase, Line 3 can be done in $O(k)$ time by choosing the pair (S_i, W_i) that minimizes $f(S_W) + f(W - S_W) - f(W)$. Before leaving the i^{th} phase, in which W_i is replaced by S_i and $W_i - S_i$, we compute an S_W that minimizes $f(S) + f(W - S) - f(W)$ for each $W \in \{S_i, W_i - S_i\}$. This observation does not improve the time bound but it avoids duplicate computations.

We summarize the arguments so far as the next theorem.

Theorem 2.3. *Given a submodular system (V, f) and an integer $k \geq 1$, algorithm GSA finds an ℓ-partition $\mathcal{P}_\ell$ of V for $\ell = 1, 2, \ldots, k$ in a total time of $O(k|V|^3\theta)$, where θ is the time bound of the oracle for f.* □

Note that we can use an efficient 2-partition algorithm directly instead of calling the oracle. In particular for k-PPG, Line 3 can be done in $O(mn + n^2 \log n)$ time if implemented by the minimum cut algorithm of [NI 92] or [SW 94], where m and n are the numbers of edges and vertices respectively. This was first pointed out by Kapoor [Kap 96].

We now consider to derive the performance guarantee of GSA. For this, we first prove a technical lemma.

Lemma 2.1 (Main lemma). *Let $\mathcal{P}_\ell$ be the ℓ-partition of V found by GSA in the $(\ell-1)^{th}$ phase, $1 \leq \ell \leq k$. For any ℓ-partition $\mathcal{P} = \{V_1, V_2, \ldots, V_\ell\}$ of V, it holds that*

$$f(\mathcal{P}_\ell) \;\leq\; \sum_{i=1}^{\ell-1} (f(V_i) + f(V - V_i)) - (\ell - 2) f(V). \tag{2.12}$$

Before going to the proof, notice that the right-hand side of (2.12) varies with the choice of the last member in $\mathcal{P}$ (i.e., numbered ℓ) for a fixed ℓ-partition $\mathcal{P}$.

Proof. We proceed by induction on ℓ.

It is trivial for $\ell = 1$. Suppose that it holds for $\ell - 1$. Consider an ℓ-partition $\mathcal{P} = \{V_1, V_2, \ldots, V_\ell\}$. Since $\mathcal{P}_{\ell-1}$ is an $(\ell-1)$-partition, there must exist $W \in \mathcal{P}_{\ell-1}$ and $V_j, V_h \in \mathcal{P}$ with $j < h$ satisfying $W \cap V_j \neq \emptyset \neq W \cap V_h$ (imaging to put ℓ balls $v_i \in V_i$ into $\ell - 1$ boxes $W' \in \mathcal{P}_{\ell-1}$). We here define an $(\ell-1)$-partition $\mathcal{P}' = \{V_1, \ldots, V_{j-1}, V_{j+1}, \ldots, V_{\ell-1}, V_j \cup V_\ell\}$, in which V_j is merged with V_ℓ and comes to the last. Applying the induction hypothesis on $\mathcal{P}_{\ell-1}$ and $\mathcal{P}'$, we have

$$f(\mathcal{P}_{\ell-1}) \;\leq\; \sum_{1 \leq i \leq \ell-1, i \neq j} (f(V_i) + f(V - V_i)) - (\ell - 3) f(V). \tag{2.13}$$

Thus to show (2.12), it suffices to show

$$f(\mathcal{P}_\ell) - f(\mathcal{P}_{\ell-1}) \;\leq\; f(V_j) + f(V - V_j) - f(V). \tag{2.14}$$

Notice that $\emptyset \neq W \cap V_j \subset W$ holds. Thus pair $(W \cap V_j, W)$ is a splitting candidate for Line 3 of GSA in the $(\ell-1)^{th}$ phase. Hence, by the optimality of $(S_{\ell-1}, W_{\ell-1})$, we have

$$\begin{aligned} f(\mathcal{P}_\ell) - f(\mathcal{P}_{\ell-1}) &= f(S_{\ell-1}) + f(W_{\ell-1} - S_{\ell-1}) - f(W_{\ell-1}) \\ &\leq f(W \cap V_j) + f(W - V_j) - f(W) \\ &\leq f(V_j) + f(W - V_j) - f(W \cup V_j) \\ &\leq f(V_j) + f(V - V_j) - f(V), \end{aligned}$$

where the last two inequalities are from submodularity. □

Remark: Lemma 2.1 holds for any submodular system, not necessarily symmetric or nonnegative. In fact, it is valid for system (V, f) that satisfies the next inequality

$$f(X \cap Y) + f(X - Y) + f(V) \;\leq\; f(Y) + f(V - Y) + f(X) \tag{2.15}$$

for all $X, Y \subseteq V$ with $\emptyset \neq X \cap Y \subset X$. □

Lemma 2.1 does not immediately provide the performance guarantee. Let us find the performance guarantees of GSA for various MPP-NTs in the next two subsections.

2.2.3 Performance analysis: part I

Theorem 2.4. *Given a submodular system (V, f) and an integer $k \geq 2$, where $f(V) \geq 0$ holds, GSA finds a k-partition of V with weight at most $(1+\alpha)(1-\frac{1}{k})opt$, where opt is the optimal value, and α is an arbitrary constant that satisfies $\sum_{i=1}^{k} f(V-V_i) \leq \alpha \sum_{i=1}^{k} f(V_i)$ for all k-partitions $\mathcal{P} = \{V_1, \ldots, V_k\}$ of V.*

Proof. Let $\mathcal{P}^* = \{V_1^*, V_2^*, \ldots, V_k^*\}$ be a minimum k-partition of V satisfying

$$f(V_k^*) + f(V - V_k^*) \;=\; \max_{1\leq i\leq k}\{f(V_i^*) + f(V - V_i^*)\}.$$

It then holds that $f(V_k^*) + f(V - V_k^*) \geq \frac{1}{k}\sum_{i=1}^{k}(f(V_i^*) + f(V - V_i^*))$. Therefore we have

$$\begin{aligned}\sum_{i=1}^{k-1}(f(V_i^*) + f(V - V_i^*)) &\leq (1-\frac{1}{k})\sum_{i=1}^{k}(f(V_i^*) + f(V - V_i^*)) \\ &\leq (1+\alpha)(1-\frac{1}{k})\sum_{i=1}^{k} f(V_i^*) \;=\; (1+\alpha)(1-\frac{1}{k})opt.\end{aligned}$$

On the other hand, by Lemma 2.1 and $f(V) \geq 0$, GSA finds a k-partition with weight at most $\sum_{i=1}^{k-1}(f(V_i^*) + f(V - V_i^*))$. Hence the proof is completed. □

(Note that, in considering performance guarantee, we may have to further assume $opt \geq 0$.)

It is clear that, for symmetric functions (i.e., $f(V-X) = f(X)$ holds for all $X \subseteq V$), we can let $\alpha = 1$ and thus obtain the following corollaries.

Corollary 2.1 (Queyranne [Que 99]). *Problem k-PPSSS in a nonnegative symmetric submodular system (V, f) can be approximated by GSA within a factor of $2 - \frac{2}{k}$ in $O(k|V|^3\theta)$ time, where θ is the time bound of the oracle for f.* □

Corollary 2.2 (Saran, Vazirani [SV 91] and Kapoor [Kap 96]). *Problem k-PPG can be approximated by GSA within a factor of $2 - \frac{2}{k}$ in $O(kn(m + n\log n))$ time, where n and m are the numbers of vertices and edges respectively.* □

Corollary 2.3. *Problem k-PPG in a planar graph with n vertices can be approximated within a factor of $2 - \frac{2}{k}$ in $O(kn^2)$ time.*

Proof. Employ an $O(n^2)$ time minimum cut algorithm for planar graph (see [NI 00]). □

In Chapter 1, we have noted the strength of a graph. Given a graph G with a nonnegative edge weight function w, the strength of G is defined by

$$\sigma(G, w) \;=\; \min_{k\geq 2}\left\{\frac{\text{the optimum of } k\text{-PPG in } G}{k-1}\right\}.$$

By applying GSA to k-PPG in G for $k = n$, we obtain the next result.

Corollary 2.4. *Given a graph G with nonnegative weights on edges, a value σ satisfying*

$$\frac{\sigma}{2-\frac{2}{n}} \le \sigma(G,w) \le \sigma$$

can be determined in $O(n^2(m+n\log n))$ time, where $\sigma(G,w)$, n and m are the strength of G, number of vertices and number of edges, respectively. □

We note that $\sigma(G,w)$ can be find in $O(mn^2(m+n\log n))$ time, see [Cun 85]. Our $(2-\frac{2}{n})$-approximation algorithm is faster by a factor of m.

For k-PPSS we cannot use $\alpha = 1$. Nevertheless, we show that $\alpha = k-1$ is enough.

Lemma 2.2. *For any k-partition $\{V_1,\ldots,V_k\}$ of a submodular system (V,f), it holds that*

$$\sum_{i=1}^{k} f(V-V_i) \le (k-1)\sum_{i=1}^{k} f(V_i) - k(k-2)f(\emptyset).$$

Proof. For any sets $X,Y \subseteq V$ satisfying $X\cap Y=\emptyset$, $f(X\cup Y) \le f(X)+f(Y)-f(\emptyset)$ holds by submodularity. An easy induction then shows that $f(V-V_i) = f(\bigcup_{j\neq i} V_j) \le \sum_{j\neq i} f(V_j) - (k-2)f(\emptyset)$ for all $i=1,\ldots,k$. Hence the lemma. □

Theorem 2.5. *Problem MPP-NT in a submodular system (V,f) satisfying $f(V)\ge 0$ and $f(\emptyset)\ge 0$ can be approximated within a factor of $k-1$ in $O(k|V|^3\theta)$ time, where θ is the time bound of the oracle for f.*

Proof. We first show that any k-partition $\mathcal{P}=\{V_1,V_2,\ldots,V_k\}$ has weight $f(\mathcal{P})\ge 0$ (for the definition of performance guarantee). This can be seen by

$$\begin{aligned} f(\mathcal{P}) &= \sum_{i=1}^{k} f(V_i) \ge f(V_1\cup V_2)+f(\emptyset)+\sum_{i=3}^{k} f(V_i) \\ &\ge f(V_1\cup V_2\cup V_3)+2f(\emptyset)+\sum_{i=4}^{k} f(V_i) \ge \cdots \ge f(V)+(k-1)f(\emptyset). \end{aligned}$$

It is then easy to see the guarantee $k-1$ by Theorem 2.4 and Lemma 2.2 (or applying Lemma 2.1 directly). □

Corollary 2.5. *Problem k-PPSS for a nonnegative submodular system (V,f) can be approximated by GSA within a factor of $k-1$ in $O(k|V|^3\theta)$ time, where θ is the time bound of the oracle for f.* □

Let us next consider k-PPMSS.

Theorem 2.6. *Problem MPP-NT in submodular system (V,f) satisfying $f(\emptyset)\ge 0$ and $f(V)\ge f(S)$ for all $\emptyset\neq S\subset V$ can be approximated by GSA within a factor of $2-\frac{2}{k}$ in $O(k|V|^3\theta)$ time, where θ is the time bound of the oracle for f.*

Proof. Let $\mathcal{P}^* = \{V_1^*, V_2^*, \ldots, V_k^*\}$ be a minimum k-partition of V satisfying

$$f(V_{k-1}^*) + f(V_k^*) \;=\; \max_{1 \le i < j \le k} \{f(V_i^*) + f(V_j^*)\} \ge \frac{2}{k} \sum_{i=1}^{k} f(V_i^*).$$

By Lemma 2.1, we see that GSA finds a k-partition $\mathcal{P}_k$ with weight

$$f(\mathcal{P}_k) \;\le\; \sum_{i=1}^{k-1} (f(V_i^*) + f(V - V_i^*)) - (k-2) f(V). \tag{2.16}$$

By assumption, $f(V) \ge f(V - V_i^*)$ holds for all i. Thus we have

$$\begin{aligned} f(\mathcal{P}_k) \;&\le\; \sum_{i=1}^{k-1} f(V_i^*) + f(V - V_{k-1}^*) \\ &\le\; \sum_{i=1}^{k-1} f(V_i^*) + \sum_{1 \le i \le k, i \ne k-1} f(V_i^*) \\ &=\; 2 \sum_{i=1}^{k} f(V_i^*) - f(V_{k-1}^*) - f(V_k^*) \;\le\; (2 - \frac{2}{k}) \sum_{i=1}^{k} f(V_i^*). \end{aligned}$$

(For the second inequality, see the proof of Lemma 2.2.)

For showing the performance guarantee, we need to show $f(\mathcal{P}^*) = \sum_{i=1}^{k} f(V_i^*) \ge 0$. In fact, we can show a stronger result that $f(S) \ge 0$ holds for all $S \subseteq V$. Notice that, for any set S with $\emptyset \ne S \subset V$, $f(S) + f(V - S) \ge f(V) + f(\emptyset)$ holds by submodularity. Since $f(V) \ge f(V - S)$ holds by assumption, we have $f(V) \ge f(S) \ge f(\emptyset) \ge 0$. Hence the theorem. □

Corollary 2.6. *Problem k-PPMSS can be approximated by GSA within a factor of* $2 - \frac{2}{k}$ *in* $O(k|V|^3\theta)$ *time for any nonnegative monotone submodular system* (V, f)*, where* θ *is the time bound of the oracle for* f*.* □

Our derivation of the performance guarantees of GSA is not only simple and unified, but it also allows us to use approximation algorithms in Line 3 of GSA. Let us see the details in the following.

Assume that $f(S) + f(W - S) - f(W) \ge 0$ holds for any $\emptyset \ne S \subset W \subseteq V$ (which is true if $f(\emptyset) \ge 0$ holds). Suppose that a ρ-approximation algorithm is used in Line 3 of GSA, where ρ is independent on the choice of W. We observe that, by a similar induction as the proof for Lemma 2.1, the ℓ-partition $\mathcal{P}_\ell$, $1 \le \ell \le k$, obtained by this variant of GSA has weight

$$f(\mathcal{P}_\ell) \;\le\; \rho \left(\sum_{i=1}^{\ell-1} \big(f(V_i) + f(V - V_i)\big) - (\ell - 2) f(V) \right).$$

Therefore we have the next theorem.

Theorem 2.7. *The variant of GSA that uses a ρ-approximation algorithm in Line 3 is a $\rho(1+\alpha)(1-\frac{1}{k})$-approximation algorithm for k-PPSS, where α is an arbitrary constant that satisfies $\sum_{i=1}^{k} f(V-V_i) \le \alpha \sum_{i=1}^{k} f(V_i)$ for all k-partitions $\{V_1, \ldots, V_k\}$ of V.* □

(Note that $\alpha \le k-1$ holds in general and $\alpha = 1$ holds for symmetric submodular systems.) As a result, we obtain the next corollary by using the linear time $(2+\epsilon)$-approximation algorithm [Mat 93] for the minimum cut problem in graphs with unit edge weights, where $\epsilon \in (0,1)$ is an arbitrary number.

Corollary 2.7. *Problem k-PPG in a graph with unit edge weights can be approximated within a factor of $(4+\epsilon)(1-\frac{1}{k})$ in $O(k(n+m))$ time, where $\epsilon \in (0,1)$ is a constant, n and m are the numbers of vertices and edges respectively.* □

Obviously, similar results can be obtained for other MPP-NTs.

2.2.4 Performance analysis: part II

In this subsection, we consider to apply GSA to k-partition problems in hypergraphs.

Let $H = (V, E)$ be a hypergraph with a weight function $w : E \to \mathbf{R}^+$. Recall that three types of k-partition problems, k-PPH-T1, k-PPH-T2 and k-PPH-T3, employ objective functions $f_1 = \frac{w(E)}{k} - w_{\text{in}}$, $f_2 = w_{\text{ex}}$ and $f_3 = w_{\text{in}} + w_{\text{ex}} - \frac{w(E)}{k}$, respectively.

First of all, observe that function $f_2 = w_{\text{ex}}$ is nonnegative, symmetric and submodular. Thus k-PPH-T2 is a special case of k-PPSSS. Hence GSA is a $(2-\frac{2}{k})$-approximation algorithm for k-PPH-T2. Let us consider its running time. Recall that w_{ex} can be evaluated in $O(D)$ time, where D is the sum of degrees of hyperedges. Thus GSA can be implemented so that it has $O(kn^3D)$ running time in a hypergraph with n vertices. In the next theorem, we show that a faster implementation is available.

Theorem 2.8. *The k-PPH-T2 problem can be approximated by GSA within a factor of $2-\frac{2}{k}$ in $O(kn(n\log n + D))$ time, where n and D are the number of vertices and the sum of degrees of hyperedges respectively.*

Proof. The guarantee is implied by Corollary 2.1. We next show a faster implementation.

Suppose that a hypergraph $H = (V, E)$ with a weight function w is given. We claim that, for any $W \subseteq V$, a nonempty set $S \subset W$ minimizing $w_{\text{ex}}(S) + w_{\text{ex}}(W-S) - w_{\text{ex}}(W)$ can be found in $O(|W|^2 \log |W| + |W|D)$ time. This implies that Line 3 of GSA can be executed in $O(\sum_{W \in \mathcal{P}_i}(|W|^2 \log |W| + |W|D)) = O(n^2 \log n + nD)$ time in the i^{th} phase. Hence the claimed running time.

Now we show the claimed fact in the following. Denote the set of hyperedges *between* two vertex subsets V_1 and V_2 by

$$E(V_1 : V_2) \triangleq \{e \in E \mid e \cap V_1 \neq \emptyset \neq e \cap V_2,\ e \subseteq V_1 \cup V_1\}.$$

We have

$$\begin{aligned}
w_{\text{ex}}(S) + w_{\text{ex}}(W-S) - w_{\text{ex}}(W) &= \sum_{e\in\delta(S)} w(e) + \sum_{e\in\delta(W-S)} w(e) - \sum_{e\in\delta(W)} w(e) \\
&= 2\sum_{e\in E(S:W-S)} w(e) + \sum_{e\in\delta(S)\cap\delta(W-S)\cap\delta(W)} w(e) \\
&= 2\left(\sum_{e\in E(S:W-S)} w(e) + \sum_{e\in\delta(S)\cap\delta(W-S)\cap\delta(W)} \frac{w(e)}{2}\right).
\end{aligned}$$

Thus minimization of $w_{\text{ex}}(S) + w_{\text{ex}}(W-S) - w_{\text{ex}}(W)$ can be reduced to 2-PPH-T2 (i.e., minimum cut problem) in hypergraph $H[W] = (W, E\cap 2^W)$ with a modified weight function $w' : E\cap 2^W \to \mathbf{R}^+$ defined by

$$w'(e\cap 2^W) = \begin{cases} w(e) & \text{if } e\subseteq W, \\ \frac{w(e)}{2} & \text{if } e\in\delta(W). \end{cases}$$

(Notice that there may exist multiple hyperedges.)

Therefore, we can minimize $w_{\text{ex}}(S) + w_{\text{ex}}(W-S) - w_{\text{ex}}(W)$ in $O(|W|^2\log|W| + |W|D)$ time by applying the minimum cut algorithm [KW 96]. □

Now let us consider problems k-PPH-T1 and k-PPH-T3.

Functions $f_1 = \frac{w(E)}{k} - w_{\text{in}}$ and $f_3 = w_{\text{in}} + w_{\text{ex}} - \frac{w(E)}{k}$ may not be nonnegative or symmetric. Thus Corollary 2.1 or Theorem 2.4 cannot be applied. Nevertheless, since both Theorem 2.2 and Lemma 2.1 do not require nonnegative or symmetric function, we can still use GSA to find a k-partition in polynomial time (by Theorem 2.2), and estimate the performance guarantees (by Lemma 2.1). Let us see the details in the next two theorems, where we also provide faster implementations.

Theorem 2.9. *The k-PPH-T1 problem can be approximated by GSA within a factor of $(1-\frac{1}{k})\min\{k, d^+_{\max}\}$ in $O(kn(n\log n + D))$ time, where $d^+_{\max}$, n and D are the maximum degree of hyperedges of positive weights, the number of vertices and the sum of degrees of hyperedges, respectively.*

Proof. Suppose that a hypergraph $H = (V, E)$ with a weight function $w : E \to \mathbf{R}^+$ are given.

We first derive the performance guarantee of GSA. For this, let $\mathcal{P}^* = \{V_1^*, V_2^*, \ldots, V_k^*\}$ be an optimal solution satisfying

$$w_{\text{ex}}(V_k^*) = \max_{1\le i\le k}\{w_{\text{ex}}(V_i^*)\} \ge \frac{1}{k}\sum_{i=1}^{k} w_{\text{ex}}(V_i^*).$$

By Lemma 2.1, GSA finds a k-partition whose weight is at most

$$\begin{aligned}
&\sum_{i=1}^{k-1}(f_1(V_i^*)+f_1(V-V_i^*))-(k-2)f_1(V)\\
=\;&\sum_{i=1}^{k-1}\Big((\frac{w(E)}{k}-w_{\rm in}(V_i^*))+(\frac{w(E)}{k}-w_{\rm in}(V-V_i^*))\Big)-(k-2)(\frac{w(E)}{k}-w_{\rm in}(V))\\
=\;&\sum_{i=1}^{k-1}\big(w(E)-w_{\rm in}(V_i^*)-w_{\rm in}(V-V_i^*)\big) \;=\; \sum_{i=1}^{k-1}w_{\rm ex}(V_i^*)\\
\le\;&(1-\frac{1}{k})\sum_{i=1}^{k}w_{\rm ex}(V_i^*)\\
\le\;&(1-\frac{1}{k})\min\{k,d_{\max}^+\}\sum_{i=1}^{k}f_1(V_i^*). \qquad (2.17)
\end{aligned}$$

The last inequality is based on the fact that the weight of a hyperedge e is counted once in the optimum $\sum_{i=1}^{k}f_1(V_i^*)$, whereas it is counted p_e times in $\sum_{i=1}^{k}w_{\rm ex}(V_i^*)$, if $p_e \triangleq |\{i \mid e\cap V_i^* \neq \emptyset\}| \geq 2$ (otherwise it is not counted at all). Notice that we only need to consider hyperedges with positive weights, for which $p_e \le \min\{k, d_{\max}^+\}$ holds.

Since we can evaluate $f_1 = \frac{w(E)}{k} - w_{\rm in}$ in $O(D)$ time, GSA for k-PPH-T1 can be executed in $O(kn^3D)$ time. We give a faster implementation in the following.

Similarly to the proof of Theorem 2.8, we show how to find a nonempty set $S \subset W$ minimizing $f_1(S)+f_1(W-S)-f_1(W)$ in $O(|W|^2\log|W|+|W|D)$ time for any fixed $W\subseteq V$. For this, notice that

$$f_1(S)+f_1(W-S)-f_1(W) \;=\; w_{\rm in}(W)-w_{\rm in}(S)-w_{\rm in}(W-S)+\frac{w(E)}{k}$$

holds. Hence minimization of $f_1(S)+f_1(W-S)-f_1(W)$ can be reduced to 2-PPH-T1 (i.e., minimum cut problem) in hypergraph $H_W \triangleq (W,E_W)$, where $E_W=\{e\in E \mid e\subseteq W\}$ and hyperedges in E_W have the same weights as they have in H. Using the minimum cut algorithm [KW 96], we can minimize it in $O(|W|^2\log|W|+|W|D)$ time. □

Theorem 2.10. *The k-PPH-T3 problem can be approximated within a factor of $2-\frac{2}{k}$ by GSA in $O(kn(n\log n+D))$ time, where n and D are the number of vertices and the sum of degrees of hyperedges, respectively.*

Proof. Suppose that a hypergraph $H=(V,E)$ with a weight function $w: E\to \mathbf{R}^+$ are given. We first derive the performance guarantee. Let $\mathcal{P}^*=\{V_1^*,V_2^*,\ldots,V_k^*\}$ be an optimal solution satisfying $w_{\rm ex}(V_k^*)=\max_{1\le i\le k}w_{\rm ex}(V_i^*)\ge\frac{1}{k}\sum_{i=1}^{k}w_{\rm ex}(V_i^*)$.

By Lemma 2.1, GSA finds a k-partition whose weight is at most

$$\begin{aligned}
&\sum_{i=1}^{k-1}(f_3(V_i^*)+f_3(V-V_i^*))-(k-2)f_3(V)\\
=\;&\sum_{i=1}^{k-1}\left(\big(w_{\rm in}(V_i^*)+w_{\rm ex}(V_i^*)-\frac{w(E)}{k}\big)+\big(w_{\rm in}(V-V_i^*)+w_{\rm ex}(V-V_i^*)-\frac{w(E)}{k}\big)\right)\\
&\qquad\qquad\qquad -(k-2)\left(w_{\rm in}(V)+w_{\rm ex}(V)-\frac{w(E)}{k}\right)\\
=\;&\sum_{i=1}^{k-1}\big(w_{\rm in}(V_i^*)+w_{\rm ex}(V_i^*)+w_{\rm in}(V-V_i^*)+w_{\rm ex}(V-V_i^*)-w(E)\big)\\
=\;&\sum_{i=1}^{k-1}w_{\rm ex}(V_i^*)\;\le\;(1-\frac{1}{k})\sum_{i=1}^{k}w_{\rm ex}(V_i^*)\;\le\;(2-\frac{2}{k})\sum_{i=1}^{k}f_3(V_i^*). \qquad (2.18)
\end{aligned}$$

The last inequality is based on the fact that the weight of a hyperedge e is counted p_e-1 times in the optimum $\sum_{i=1}^{k}f_3(V_i^*)$, whereas it is counted p_e times in $\sum_{i=1}^{k}w_{\rm ex}(V_i^*)$, if $p_e=|\{i\mid e\cap V_i^*\neq\emptyset\}|\geq 2$.

Next, let us consider a fast implementation of GSA. Again, we only need to show how to find a nonempty $S\subset W$ minimizing $f_3(S)+f_3(W-S)-f_3(W)$ in $O(|W|^2\log|W|+|W|D)$ time for any $W\subseteq V$. For this, notice that

$$\begin{aligned}
&f_3(S)+f_3(W-S)-f_3(W)\\
=\;&\big(w_{\rm in}(S)+w_{\rm ex}(S)-\frac{w(E)}{k}\big)+\big(w_{\rm in}(W-S)+w_{\rm ex}(W-S)-\frac{w(E)}{k}\big)\\
&\qquad\qquad\qquad -\big(w_{\rm in}(W)+w_{\rm ex}(W)-\frac{w(E)}{k}\big)\\
=\;&\left(\sum_{e\in E(S:W-S)}w(e)+\sum_{e\in\delta(S)\cap\delta(W-S)\cap\delta(W)}w(e)\right)-\frac{w(E)}{k}
\end{aligned}$$

holds. Therefore, the minimization of $f_3(S)+f_3(W-S)-f_3(W)$ reduces to minimum cut problem in hypergraph $H[W]=(W,E\cap 2^W)$, in which hyperedges have the same weights as in H. Again, using the algorithm [KW 96], we can minimize it in $O(|W|^2\log|W|+|W|D)$ time. □

2.3 Modified GSA (M-GSA) for general MPP

2.3.1 Algorithm description and main lemma

In the previous section, we have discussed algorithm GSA (greedy splitting algorithm) for MPP-NT. We introduce in this section a slightly modified version of GSA (M-GSA) to approximate the general MPP. See Figure 2.3.

Input: A submodular system (V, f), a target set $T \subseteq V$ and an integer k, $1 \le k \le |T|$.
Output: A target-split $\mathcal{P}_k$ of V.

```
1  $\mathcal{P}_1 \leftarrow \{V\}$
2  for $i = 1, \ldots, k-1$ do
3      $(S_i, W_i) \leftarrow \text{argmin}\{f(S) + f(W-S) - f(W) \mid \emptyset \neq S \subset W,\ W \in \mathcal{P}_i,$
                                                  $S \cap T \neq \emptyset \neq (W-S) \cap T\}$
4      $\mathcal{P}_{i+1} \leftarrow (\mathcal{P}_i - \{W_i\}) \cup \{S_i, W_i - S_i\}$
5  end /* for */
```

Figure 2.3: Modified greedy splitting algorithm (M-GSA) for MPP.

Only Line 3 is different from GSA, where we want to minimize $f(S) + f(W-S) - f(W)$ under an additional constraint $S \cap T \neq \emptyset \neq (W-S) \cap T$. By this constraint, we can ensure that the output $\mathcal{P}_k$ is a target-split of V with respect to T.

Let us first consider the minimization of $f(S) + f(W-S)$ for a fixed $W \subseteq V$, under the constraints $\emptyset \neq S \subset W$ and $S \cap T \neq \emptyset \neq (W-S) \cap T$. If $W \subseteq T$, the constraints reduce to $\emptyset \neq S \subset W$ only, which implies that the minimization can be achieved in $O(|W|^3\theta)$ time by Theorem 2.2. Otherwise, as shown soon in Theorem 2.11, it can be achieved by solving the next *partial s, t-partition problem.*

Problem 2.13 (Partial s,t-partition problem). *Given a submodular system (V, f) with a set $W \subseteq V$ and distinct $s, t \in W$, find a set S such that*

$$\begin{array}{ll} \textit{minimize} & f(S) + f(W-S) - f(W) \\ \textit{subject to} & s \in S \subset W,\ t \in W-S. \end{array}$$

Lemma 2.3. *Given a submodular system (V, f), $W \subseteq V$ and distinct $s, t \in W$, a set S^* such that $s \in S^* \subset W$, $t \in W - S^*$ and $f(S^*) + f(W - S^*) - f(W)$ is minimum can be found in $O(|W|^7 \log |W|\theta)$ time, where θ is the time bound of the oracle for f.*

Proof. Consider a system $(W - \{s, t\}, g)$, where function $g : 2^{W-\{s,t\}} \to \mathbf{R}$ is defined by

$$g(S) \;=\; f(S \cup \{s\}) + f(W-(S \cup \{s\})) \quad S \subseteq W - \{s, t\}.$$

We only need to find a set $S' \subseteq W - \{s, t\}$ such that $g(S')$ is minimum and let $S^* = S' \cup \{s\}$. It is easy to verify that function g is submodular. Submodular functions can be minimized in polynomial time ([GLS 88, IFF 00, Sch 00]). In particular, an algorithm in [IFF 00] has a running time of $O(|W|^7 \log |W|\theta)$. Hence the lemma. □

Theorem 2.11. *Given a submodular system (V, f), $T \subseteq V$ and an integer $k \geq 1$, M-GSA has running time of $O(k|T||V|^7 \log |V|\theta)$, where θ is the time bound of the oracle for f.*

Proof. The minimization of $f(S) + f(W - S) - f(W)$ for a fixed $W \subseteq V$ under the constraints $\emptyset \neq S \subset W$ and $S \cap T \neq \emptyset \neq (W - S) \cap T$ can be achieved by applying at most $|T \cap W| - 1$ partial s, t-partition computations. More precisely, we choose an arbitrary $s \in W \cap T$ and compute a minimum partial s, t-partition of W for each $t \in W \cap T - \{s\}$. It is clear that the minimum partial s,t-partition among all t is a required solution. Hence Line 3 of M-GSA can be done in $\sum_{W \in \mathcal{P}_i} O(|T \cap W||W|^7 \log |W|\theta) = O(|T||V|^7 \log |V|\theta)$ time. This implies that the running time of M-GSA is $O(k|T||V|^7 \log |V|\theta)$. $\square$

Remark: Of course, we can get faster implementations in special cases. For instance, it is easy to see that the partial s,t-partition problem for k-TPG reduces to minimum s, t-cut problem. Thus in this case, M-GSA can have $O(k|T|mn \log(n^2/m))$ running time for a graph with n vertices and m edges. $\square$

Consider the performance of M-GSA. It is not difficult to see that analogous results as obtained in the previous section can be obtained in a straightforward manner. Analogously to Lemma 2.1, we obtain the next technique lemma.

Lemma 2.4. *Let $\mathcal{P}_\ell$ be the ℓ-target-split of V found by M-GSA in the $(\ell - 1)^{th}$ phase, $1 \leq \ell \leq k$. For any ℓ-target-split $\{V_1, V_2, \ldots, V_\ell\}$ of V, it holds that*

$$f(\mathcal{P}_\ell) \leq \sum_{i=1}^{\ell-1} (f(V_i) + f(V - V_i)) - (\ell - 2)f(V). \tag{2.19}$$

$\square$

2.3.2 Performance analysis

Similarly as before, we obtain the next theorem as a result of Theorem 2.11 and Lemma 2.4.

Theorem 2.12. *Given a nonnegative submodular system (V, f) with a target set $T \subseteq V$, k-TPSS can be approximated by M-GSA within a factor of $(1+\alpha)(1-\frac{1}{k})$ in $O(k|T||V|^7 \log |V|\theta)$ time, where α is an arbitrary constant satisfying $\sum_{i=1}^k f(V - V_i) \leq \alpha \sum_{i=1}^k f(V_i)$ for all k-target-split $\{V_1, \ldots, V_k\}$ of V, and θ is the time bound of the oracle for f.* $\square$

Corollary 2.8. *Given a nonnegative submodular system (V, f) with a target set $T \subseteq V$, k-TPSS can be approximated by M-GSA within a factor of $k - 1$ in $O(k|T||V|^7 \log |V|\theta)$ time, where θ is the time bound of the oracle for f.* $\square$

Corollary 2.9. *Given a submodular system (V, f) with a target set $T \subseteq V$, where f is nonnegative and symmetric, k-TPSSS can be approximated by M-GSA within a factor of $2 - \frac{2}{k}$ in $O(k|T||V|^7 \log |V|\theta)$ time, where θ is the time bound of the value oracle for f.* $\square$

Theorem 2.13. , *Given a hypergraph $H = (V, E)$ and a target set $T \subseteq V$, k-TPH-T2 can be approximated by M-GSA within a factor of $2 - \frac{2}{k}$ in $O(k|T|m'n' \log(n'^2/m'))$ time, where $m' = 2D + |E|$, $n' = |V| + 2|E|$ and D is the sum of degrees of hyperedges.*

Proof. The guarantee is implied by Corollary 2.9. We give a faster implementation.

Analogously to Theorem 2.8, it is not difficult to see that, the partial s,t-partition problem reduces to 2-TPH-T2 with target set $\{s, t\}$ in hypergraph $H[W] = (W, E \cap 2^W)$ with a modified weight function $w' : E \cap 2^W \to \mathbf{R}^+$ defined by

$$w'(e \cap 2^W) = \begin{cases} w(e) & \text{if } e \subseteq W, \\ \frac{w(e)}{2} & \text{if } e \in \delta(W). \end{cases}$$

Problem 2-TPH-T2 with two targets can be reduced to a maxflow problem ([Law 73]). More precisely, given a hypergraph H'' with n'' vertices and m'' hyperedges, where the sum of degrees of hyperedges is D'', it reduces to a maxflow problem in a (directed) graph with $n''+2m''$ vertices and $2D''+m''$ edges. The theorem is then shown by an easy calculation. □

Combining with the remark of Theorem 2.11, we have the next corollary.

Corollary 2.10 (Maeda, Nagamochi and Ibaraki [MNI 93]). *Problem k-TPG can be approximated by M-GSA within a factor of $2 - \frac{2}{k}$ in $O(k|T|mn \log(n^2/m))$ time, where T is the target set, n and m are the numbers of vertices and edges, respectively.* □

Similar results can be obtained in a straightforward manner.

Theorem 2.14. *Problem MPP in submodular system (V, f) satisfying $f(\emptyset) \geq 0$ and $f(V) \geq f(S)$ for all $\emptyset \neq S \subset V$ can be approximated by M-GSA within a factor of $2 - \frac{2}{k}$ in $O(k|T||V|^7 \log |V|\theta)$ time, where θ is the time bound of the oracle for f.* □

Corollary 2.11. *Given a nonnegative, monotone submodular system (V, f) with a target set $T \subseteq V$, k-TPMSS can be approximated by M-GSA within factor $2 - \frac{2}{k}$ in $O(k|T||V|^7 \log |V|\theta)$ time, where θ is the time bound of the oracle for f.* □

Analogously to the proof of Theorems 2.13, 2.9 and 2.10, we have the next two theorems.

Theorem 2.15. *Given a hypergraph $H = (V, E)$ with a target set $T \subseteq V$, k-TPH-T1 can be approximated by M-GSA within a factor of $(1 - \frac{1}{k}) \min\{k, d^+_{\max}\}$ in $O(k|T|m'n' \log(n'^2/m'))$ time, where $m' = 2D + |E|$, $n' = |V| + 2|E|$, $d^+_{\max}$ is the maximum degree of hyperedges with positive weights, and D is the sum of degrees of hyperedges.* □

Theorem 2.16. *Given a hypergraph $H = (V, E)$ with a target set $T \subseteq V$, k-TPH-T3 can be approximated by M-GSA within a factor of $2 - \frac{2}{k}$ in $O(k|T|m'n' \log(n'^2/m'))$ time, where $m' = 2D + |E|$, $n' = |V| + 2|E|$, and D is the sum of degrees of hyperedges.* □

Again, our proof allows us to use approximate algorithms in Line 3 of M-GSA. Assume that $f(S)+f(W-S)-f(W) \geq 0$ holds for all $\emptyset \neq S \subset W$, $W \in \mathcal{P}_i$ and $S \cap T \neq \emptyset \neq (W-S) \cap T$ (which is true if $f(\emptyset) \geq 0$ holds). It is easy to see the next theorem.

Theorem 2.17. *The variant of M-GSA that uses a ρ-approximation algorithm in Line 3 of M-GSA is a $\rho(1+\alpha)(1-\frac{1}{k})$-approximation algorithm for k-PPSS, where α is an arbitrary constant satisfying $\sum_{i=1}^{k} f(V-V_i) \leq \alpha \sum_{i=1}^{k} f(V_i)$ for all k-target-split $\{V_1, \ldots, V_k\}$ of V.* □

Again, we note that $\alpha \leq k-1$ holds in general, and $\alpha = 1$ holds for symmetric submodular systems. Similar results as in Section 2.2 can also be obtained for other MPPs.

2.4 Tight examples

In this section, we construct tight examples for algorithm GSA (such examples are also tight examples for M-GSA). For this, we need an easy property.

Proposition 2.1. *Let (V, f) and (W, g) be two submodular systems, where V and W are not necessarily disjoint. Define a function $h : 2^{V \cup W} \to \mathbf{R}$ by $h(S) = f(S \cap V) + g(S \cap W)$ for all sets $S \subseteq V \cup W$. Function h is submodular. Furthermore, it is symmetric if so are f and g.*

Proof. We first show the submodularity of h. For any sets $X, Y \subseteq V \cup W$, we have

$$\begin{aligned}
h(X) + h(Y) &= (f(X \cap V) + g(X \cap W)) + (f(Y \cap V) + g(Y \cap W)) \\
&= (f(X \cap V) + f(Y \cap V)) + (g(X \cap W) + g(Y \cap W)) \\
&\geq f((X \cap V) \cap (Y \cap V)) + f((X \cap V) \cup (Y \cap V)) \\
&\quad + g((X \cap W) \cap (Y \cap W)) + g((X \cap W) \cup (Y \cap W)) \\
&= f((X \cap Y) \cap V) + g((X \cap Y) \cap W) \\
&\quad + f((X \cup Y) \cap V) + g((X \cup Y) \cap V) \\
&= h(X \cap Y) + h(X \cup Y).
\end{aligned}$$

Hence function h is submodular.

Now suppose that functions f and g are symmetric. For any sets $X \subseteq V \cup W$, we have

$$\begin{aligned}
h((V \cup W) - X) &= f\big(((V \cup W) - X) \cap V\big) + g\big(((V \cup W) - X) \cap W\big) \\
&= f(V - X) + g(W - X) \;=\; f(X \cap V) + g(X \cap W) \\
&= h(X).
\end{aligned}$$

Hence function h is symmetric. □

Proposition 2.2. *For any $\epsilon > 0$, there exists a nonnegative submodular system for which GSA always finds a k-partition whose weight is at least $k-1-\epsilon$ times of the optimum.*

Proof. Define two submodular systems (V, f) and (W, g) as follows.

Let $V = \{v_0, v_1, \ldots, v_{k-1}\}$, and function f be defined by $f(S) \triangleq |S|$ if $S \subset V$ and $f(V) = 0$. Let $W = \{v_0, w_1, \ldots, w_{k-1}\}$ (note $V \cap W = \{v_0\}$). Let g be the cut function of graph (W, E) for $E = \{\{v_0, w_1\}, \{v_0, w_2\}, \ldots, \{v_0, w_{k-1}\}\}$ of edges of weight $\frac{k-\epsilon}{2}$. See Figure 2.4. Obviously, functions f and g are submodular. Let us consider a system $(V \cup W, h)$, where $h(S) = f(S \cap V) + g(S \cap W)$, $S \subseteq V \cup W$. By Proposition 2.1, function h is submodular (obviously it is also nonnegative).

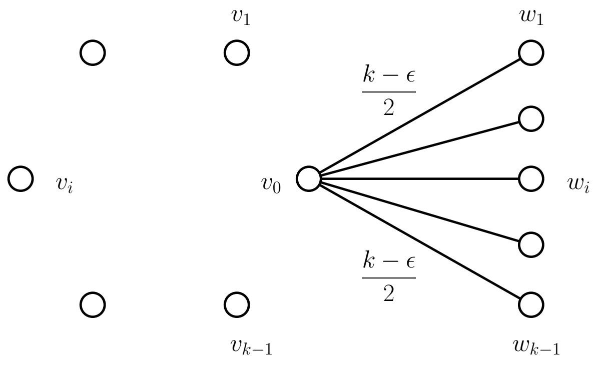


Figure 2.4: A tight example for GSA applied to k-PPSS.

There is a k-partition $\mathcal{P} = \{\{v_1\}, \{v_2\}, \ldots, \{v_{k-1}\}, W\}$ with weight k. On the other hand, since any 2-partition separating some pair of members in V has weight at least k, the minimum 2-partition is $\{\{w_i\}, (V \cup W) - \{w_i\}\}$ of weight $k - \epsilon$ for some i. Therefore GSA first finds a 2-partition $\{\{w_i\}, (V \cup W) - \{w_i\}\}$ for some i. It is then easy to see that the output of GSA is the k-partition $\mathcal{P}' = \{\{w_1\}, \{w_2\}, \ldots, \{w_{k-1}\}, V\}$, whose weight is $(k-1)(k-\epsilon)$. Thus the performance guarantee is at least as bad as

$$\frac{(k-1)(k-\epsilon)}{k} = k - 1 - \frac{(k-1)\epsilon}{k} > k - 1 - \epsilon.$$

□

Proposition 2.3. *For any $\epsilon > 0$, there exists a graph for which GSA always finds a k-way cut whose weight is at least $2 - \frac{2}{k} - \epsilon$ times of the optimum.*

Proof. Recall that k-way cut is a subset of edges whose removal leaves the graph with at least k components. Problem k-PPG is equivalent to find a minimum weight k-way cut.

Let us consider a graph $G = (V, E)$, where $V = \{v_0, v_1, \ldots, v_{k-1}, w_1, w_2, \ldots, w_{k-1}\}$, $E = \{\{v_i, v_j\} \mid 0 \le i < j \le k-1\} \cup \{\{v_0, w_i\} \mid 0 \le i \le k-1\}$. (The subgraph of G induced by vertices $\{v_0, v_1, \ldots, v_{k-1}\}$ is a *clique* of size k). Each edge $\{v_i, v_j\}$ has weight 1, whereas each edge $\{v_0, w_i\}$ has weight $k-1-\epsilon$. The graph is shown in Figure 2.5.

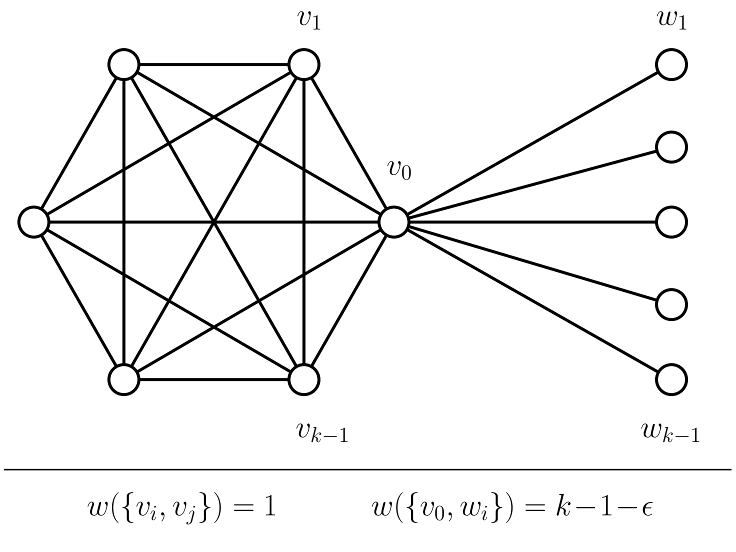


Figure 2.5: A tight example for GSA applied to k-PPG.

It is easy to see that the optimal k-way cut is the set of edges of weight 1, whereas GSA finds a solution consisting of the edges with weight $k-1-\epsilon$. Therefore the performance guarantee of GSA is at least as bad as

$$\frac{(k-1)(k-1-\epsilon)}{k(k-1)/2} = 2 - \frac{2}{k} - \frac{2\epsilon}{k} \geq 2 - \frac{2}{k} - \epsilon.$$

We note that the guarantee $2 - \frac{2}{k}$ is tight even for graphs with unit edge weights. This can be seen by modifying Figure 2.5 such that each vertex w_i and edge $\{v_0, w_i\}$ are replaced by a large clique that is connected to v_0 by exactly $k-1$ edges. □

Observe that the example in Proposition 2.3 also serves as a tight example for GSA applied to some other MPPs as k-PPH-T1, k-PPH-T2, k-PPH-T3 and k-PPSSS.

Chapter 3

Greedy Splitting of Higher Accuracy for MPP

3.1 Introduction

In the previous chapter, we have presented an algorithm GSA (greedy splitting algorithm) for MPP, which greedily increases the size of partition by one in each phase. As a natural extension, one may consider to increase the size of partition more than one in each phase. In this chapter, we consider to increase the size of partition by two or more.

Unlike the case of GSA, increasing the size of partition by two at the smallest weight increase may be difficult to implement. In fact, we do not know how to implement it in polynomial time for an arbitrary submodular system. Nevertheless, we can show some interesting properties on the quality of the obtained solutions. At the end of this chapter, we also give a remark on the extension of greedily increasing the size of partition by three or more.

We first consider MPP-NT (MPP with no target), which is easier than the general MPP problem.

3.2 Greedy splitting algorithm 2 (GSA2)

3.2.1 Algorithm description and main lemma for odd k

Let k be a positive odd integer. We consider an approximation algorithm, called greedy splitting algorithm 2 (GSA2), which is described in Figure 3.1.

GSA2 contains $\frac{k-1}{2}$ phases. It starts with $\mathcal{P}_1 = \{V\}$. In the i^{th} phase, it constructs an $(2i+1)$-partition $\mathcal{P}_{i+1}$ by splitting some member(s) of the previously obtained $(2i-1)$-partition $\mathcal{P}_i$ with the smallest weight increase. There are two ways of splitting (in order to increase the size of partition by two). One is to split two members in $\mathcal{P}_i$ into four. The other is to split one member in $\mathcal{P}_i$ into three. These are considered in Lines 3 and 4 of GSA2, respectively. We choose $\mathcal{P}_{i+1}$ so that the weight increase from $\mathcal{P}_i$ is minimum. This is done in Lines 5–9 of GSA2.

Input: A submodular system (V, f) and an odd integer $k \geq 1$.
Output: A k-partition $\mathcal{P}_{\frac{k+1}{2}}$ of V.

```
1   $\mathcal{P}_1 \leftarrow \{V\}$
2   for $i = 1, \ldots, \frac{k-1}{2}$ do
3       $(S_i^1, W_i^1, S_i^2, W_i^2) \leftarrow \mathrm{argmin}\{\sum_{j=1}^2 (f(S^j) + f(W^j - S^j) - f(W^j)) \mid$
                        $\emptyset \neq S^j \subset W^j,\ j = 1, 2$, for distinct $W^1, W^2 \in \mathcal{P}_i\}$
4       $(T_i^1, T_i^2, W_i) \leftarrow \mathrm{argmin}\{f(T^1) + f(T^2) + f(W - T^1 - T^2) - f(W) \mid$
                        $\{T^1, T^2, W - T^1 - T^2\}$ is a 3-partition of $W \in \mathcal{P}_i\}$
5       if $\sum_{j=1}^2 (f(S_i^j) + f(W_i^j - S_i^j) - f(W_i^j))$
                < $f(T_i^1) + f(T_i^2) + f(W_i - T_i^1 - T_i^2) - f(W_i)$ then
6           $\mathcal{P}_{i+1} \leftarrow (\mathcal{P}_i - \{W_i^1, W_i^2\}) \cup \{S_i^1, W_i^1 - S_i^1, S_i^2, W_i^2 - S_i^2\}$
7       else
8           $\mathcal{P}_{i+1} \leftarrow (\mathcal{P}_i - \{W_i\}) \cup \{T_i^1, T_i^2, W_i - T_i^1 - T_i^2\}$
9       end /* if */
10  end /* for */
```

Figure 3.1: Greedy splitting algorithm 2 (GSA2) for MPP-NT for odd k

Clearly, the output $\mathcal{P}_{\frac{k+1}{2}}$ is a k-partition of V. Let us consider the running time of GSA2. Line 3 can be done in $\sum_{W \in \mathcal{P}_i} O(|W|^3 \theta) = O(|V|^3 \theta)$ time by Theorem 2.2. On the other hand, since we do not know how to find efficiently a minimum 3-partition in an arbitrary submodular system, the complexity of Line 4 is *open* in general. For the purpose of ensuring that GSA2 runs in polynomial time, we assume the next.

Assumption 1. *For any set $W \subseteq V$, a 3-partition $\{T^1, T^2, W - T^1 - T^2\}$ of W that minimizes $f(T^1) + f(T^2) + f(W - T^1 - T^2) - f(W)$ can be found in polynomial time.*

To analyze the performance of GSA2, we show a lemma that is analogous to Lemma 2.1.

Lemma 3.1. *For $\ell = 0, 1, \ldots, \frac{k-1}{2}$, let $\mathcal{P}_{\ell+1}$ be the $(2\ell+1)$-partition of V found by GSA2 in the ℓ^{th} phase. For any $(2\ell+1)$-partition $\mathcal{P} = \{V_1, V_2, \ldots, V_{2\ell+1}\}$ of V, it holds that*

$$f(\mathcal{P}_{\ell+1}) \;\leq\; \sum_{i=1}^{\ell} (f(V_{2i-1}) + f(V_{2i}) + f(V - V_{2i-1} - V_{2i})) - (\ell - 1) f(V). \tag{3.1}$$

Proof. Notice that the right-hand side of (3.1) is dependent on the order of $V_1, V_2, \ldots, V_{2\ell+1}$. More precisely, it is symmetric between V_{2i-1} and V_{2i}, $1 \le i \le \ell$, and the last member of $\{V_1, V_2, \ldots, V_{2\ell+1}\}$ does not appear at all.

The proof is rather long, continuing to the end of this subsection. Similarly as before, we proceed by induction on ℓ. It is trivial for $\ell = 0$. Suppose that it holds for $\ell - 1$. Let us consider a $(2\ell+1)$-partition $\mathcal{P} = \{V_1, V_2, \ldots, V_{2\ell+1}\}$.

We will show that, there always exists a "splitting candidate" that is a candidate in Line 3 or 4 for GSA2. This implies that its splitting weight (the weight increase) is an upper bound on $f(\mathcal{P}_{\ell+1}) - f(\mathcal{P}_\ell)$. We then show how to construct a "nice" $(2\ell-1)$-partition $\mathcal{P}' = \{V'_1, \ldots, V'_{2\ell-1}\}$ from $\mathcal{P}$, for which $\sum_{i=1}^{\ell-1} \left(f(V'_{2i-1}) + f(V'_{2i}) + f(V - V'_{2i-1} - V'_{2i})\right) - (\ell-2)f(V)$ plus the splitting weight of the "splitting candidate" is at most the right-hand side of (3.1). This will prove the lemma by applying the induction hypothesis on $\mathcal{P}_\ell$ and $\mathcal{P}'$. Let us see the details in the following.

We first consider to find the splitting candidate. Since $\mathcal{P}_\ell$ is a $(2\ell-1)$-partition of V, we see that at least one of the next two cases occurs (*imaging to put* $2\ell+1$ *balls* $v_i \in V_i$ *into* $2\ell-1$ *boxes* $W' \in \mathcal{P}_\ell$).

(1) There is a $W \in \mathcal{P}_\ell$ and at least three distinct indices $r, s, t \in \{1, 2, \ldots, 2\ell+1\}$ such that $W \cap V_i \ne \emptyset$ holds for all $i \in \{r, s, t\}$ (*one box obtains at least three balls*).

(2) There are two distinct $W^1, W^2 \in \mathcal{P}_\ell$ and four distinct indices $a, b, p, q \in \{1, 2, \ldots, 2\ell+1\}$ such that $W^1 \cap V_a \ne \emptyset \ne W^1 \cap V_b$, $W^2 \cap V_p \ne \emptyset \ne W^2 \cap V_q$, $W^1 \subseteq V_a \cup V_b$ and $W^2 \subseteq V_p \cup V_q$ hold (*roughly speaking, each of two boxes contains exactly two balls*).

We can easily find splitting candidates in cases (1) and (2). However, it is still unclear how to construct the "nice partition", since, as will be seen later, the nice partitions vary with the choice of indices (i.e., r, s, t and a, b, p, q). Thus we consider the next five sub-cases.

From the symmetry in (3.1), we say V_{2i-1} and V_{2i} "partners" of each other, $1 \le i \le \ell$. Let us first consider two sub-cases derived from case (1).

(1a) There is an index $h \in \{1, \ldots, \ell\}$ satisfying $\{2h-1, 2h\} \subset \{r, s, t\}$ (i.e., two partners appear simultaneously in W). Let $r = 2h-1$ and $s = 2h$.

(1b) Otherwise $|\{2i-1, 2i\} \cap \{r, s, t\}| \le 1$ holds for all $i \in \{1, \ldots, \ell\}$. Let $r \in \{2h-1, 2h\}$ and $s \in \{2h'-1, 2h'\}$ for $1 \le h < h' \le \ell$. By assumption, $W \cap V_j = \emptyset$ holds for all $j \in \{2h-1, 2h, 2h'-1, 2h'\} - \{r, s\}$ (i.e., no partner of V_r or V_s appears in W). By the symmetry of partners, we suppose without loss of generality that $r = 2h$ and $s = 2h'$. Then $W \cap V_{2h-1} = W \cap V_{2h'-1} = \emptyset$ holds.

Case (2) is a little complicated. We first consider the sub-case in which

$$|\{a,b\}\cap\{2h-1,2h\}| \;=\; |\{p,q\}\cap\{2h'-1,2h'\}| \;=\; 1$$

holds for some $1 \le h \ne h' \le \ell$. By symmetry, we assume the next without loss of generality.

(2a) Let $a = 2h$ and $p = 2h'$ for $1 \le h < h' \le \ell$. Then $W^1 \cap V_{2h-1} = W^2 \cap V_{2h'-1} = \emptyset$ holds.

Otherwise, we have the next two sub-cases.

(2b) $|\{a,b\}\cap\{2h-1,2h\}| = |\{p,q\}\cap\{2h'-1,2h'\}| = 2$ holds for $1 \le h \ne h' \le \ell$. Let $h < h'$, $a = 2h-1$, $b = 2h$, $p = 2h'-1$, $q = 2h'$. By the assumption of case (2), $W^1 \subseteq V_{2h-1} \cup V_{2h}$ and $W^2 \subseteq V_{2h'-1} \cup V_{2h'}$ hold.

(2c) $\{|\{a,b\}\cap\{2h-1,2h\}|,\ |\{p,q\}\cap\{2h'-1,2h'\}|\} = \{1,2\}$ hold for some $1 \le h \ne h' \le \ell$. Let $a = 2h-1$, $b = 2h$, $p \in \{2h'-1, 2h'\}$ and $q \notin \{2h'-1, 2h'\}$. Without loss of generality, we assume $p = 2h'$ and $h < h'$. Hence $W^1 \subseteq V_{2h-1} \cup V_{2h}$ and $W^2 \cap V_{2h'-1} = \emptyset$ hold.

Notice that at least one of sub-cases (2a), (2b) and (2c) must occur in case (2).

Now let us show the "splitting candidate" and the corresponding "nice partition" for each sub-case. In constructing the "nice partition"s, the idea is

($*$) *binding members of $\mathcal{P}$ to hide weights that are used by the splitting candidate.*

Let us see the details in the following, where we will explain the above idea.

(1a) Split W into three pieces $W \cap V_{2h-1}$, $W \cap V_{2h}$ and $W - V_{2h-1} - V_{2h}$. Let

$$\mathcal{P}^{1a} \;=\; \{V_1, \ldots, V_{2h-2}, V_{2h+1}, \ldots, V_{2\ell}, V_{2h-1} \cup V_{2h} \cup V_{2\ell+1}\}.$$

Applying the induction hypothesis on $\mathcal{P}_\ell$ and $\mathcal{P}^{1a}$, we have

$$f(\mathcal{P}_\ell) \;\le\; \sum_{1\le i\le \ell,\ i\ne h} \big(f(V_{2i-1}) + f(V_{2i}) + f(V - V_{2i-1} - V_{2i})\big) - (\ell-2)f(V). \tag{3.2}$$

Thus, to prove (3.1), it suffices to show

$$f(\mathcal{P}_{\ell+1}) - f(\mathcal{P}_\ell) \;\le\; f(V_{2h-1}) + f(V_{2h}) + f(V - V_{2h-1} - V_{2h}) - f(V). \tag{3.3}$$

In fact, since $(W \cap V_{2h-1}, W \cap V_{2h}, W)$ is a candidate in Line 4 for GSA2, we have

$$f(\mathcal{P}_{\ell+1}) - f(\mathcal{P}_\ell) \;\le\; f(W \cap V_{2h-1}) + f(W \cap V_{2h}) + f(W - V_{2h-1} - V_{2h}) - f(W). \tag{3.4}$$

Let us show that the right-hand side of (3.3) is at least as large as the right-hand side of (3.4). This will prove the lemma for sub-case (1a). Actually, by submodularity, we have

$$\begin{aligned} & f(V_{2h-1}) + f(V_{2h}) + f(V - V_{2h-1} - V_{2h}) - f(V) \\ \geq\ & f(W \cap V_{2h-1}) + f(W \cup V_{2h-1}) + f(V_{2h}) + f(V - V_{2h-1} - V_{2h}) - f(V) - f(W) \\ \geq\ & f(W \cap V_{2h-1}) + f(W \cap V_{2h}) + f(W \cup V_{2h-1} \cup V_{2h}) + f(V - V_{2h-1} - V_{2h}) - f(V) - f(W) \\ \geq\ & f(W \cap V_{2h-1}) + f(W \cap V_{2h}) + f(W - V_{2h-1} - V_{2h}) - f(W). \end{aligned}$$

Therefore sub-case (1a) is completed.

Let us take an example k-PPG to give an explanation of the idea $(*)$. In sub-case (1a), the splitting candidate is to split W into three pieces $W \cap V_{2h-1}$, $W \cap V_{2h}$ and $W - V_{2h-1} - V_{2h}$. See Figure 3.2. The splitting weight (i.e., $f(W \cap V_{2h-1}) + f(W \cap V_{2h}) + f(W - V_{2h-1} - V_{2h}) - f(W)$ where f is the cut function) is two times of the weight of edge set

$$E(W \cap V_{2h-1} : W \cap V_{2h}) \cup E(W \cap V_{2h-1} : W - V_{2h-1} - V_{2h}) \cup E(W \cap V_{2h} : W - V_{2h-1} - V_{2h})$$

(recall that $E(X : Y)$ is the set of edges between vertex sets X and Y). It is of course no more than two times of the weight of edge set

$$E(V_{2h-1} : V_{2h}) \cup E(V_{2h-1} : V - V_{2h-1} - V_{2h}) \cup E(V_{2h} : V - V_{2h-1} - V_{2h}),$$

which is nothing but $f(V_{2h-1}) + f(V_{2h}) + f(V - V_{2h-1} - V_{2h}) - f(V)$, the right-hand side of (3.3). Therefore we "bind" V_{2h-1} and V_{2h} to V_{2l+1}, and let them be the last member in the constructed nice partition $\mathcal{P}^{1a}$. Analogous explanations are available for other sub-cases.

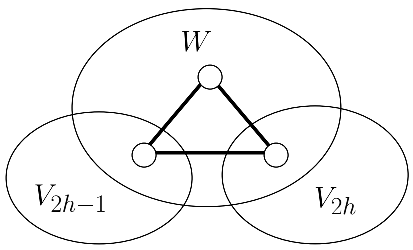


Figure 3.2: Illustration of sub-case (1a) for k-PPG.

Next let us consider sub-case (1b).

(1b) Recall that in sub-case (1b), we have a $W \in \mathcal{P}_\ell$ such that $W \cap V_{2h} \neq \emptyset \neq W \cap V_{2h'}$, $W - V_{2h} - V_{2h'} \neq \emptyset$ and $W \cap V_{2h-1} = W \cap V_{2h'-1} = \emptyset$ hold for $1 \leq h < h' \leq \ell$.

Split W into $W \cap V_{2h}$, $W \cap V_{2h'}$ and $W - V_{2h} - V_{2h'}$. Let

$$\mathcal{P}^{1b} = \{V_1, \ldots, V_{2h-2}, V_{2h+1}, \ldots, V_{2h'-2}, V_{2h'+1}, \ldots, V_{2\ell}, V_{2h-1}, V_{2h'-1}, V_{2h} \cup V_{2h'} \cup V_{2\ell+1}\}.$$

Applying the induction hypothesis on $\mathcal{P}_\ell$ and $\mathcal{P}^{1\mathrm{b}}$, we have

$$\begin{aligned} f(\mathcal{P}_\ell) \;\le\; & \sum_{1\le i\le \ell,\ i\ne h,h'} \Big(f(V_{2i-1}) + f(V_{2i}) + f(V - V_{2i-1} - V_{2i}) \Big) \\ & + \Big(f(V_{2h-1}) + f(V_{2h'-1}) + f(V - V_{2h-1} - V_{2h'-1}) \Big) - (\ell-2) f(V). \end{aligned} \tag{3.5}$$

Thus it suffices to show

$$\begin{aligned} f(\mathcal{P}_{\ell+1}) - f(\mathcal{P}_\ell) \;\le\; & f(V_{2h}) + f(V - V_{2h-1} - V_{2h}) + f(V_{2h'}) \\ & + f(V - V_{2h'-1} - V_{2h'}) - f(V - V_{2h-1} - V_{2h'-1}) - f(V). \end{aligned} \tag{3.6}$$

Since $(W \cap V_{2h}, W \cap V_{2h'}, W - V_{2h} - V_{2h'})$ is a candidate in Line 4 for GSA2, we have

$$f(\mathcal{P}_{\ell+1}) - f(\mathcal{P}_\ell) \;\le\; f(W \cap V_{2h}) + f(W \cap V_{2h'}) + f(W - V_{2h} - V_{2h'}) - f(W). \tag{3.7}$$

To complete the proof for sub-case (1b), we show that the right-hand side of (3.6) is no less than the right-hand side of (3.7). By transferring negative terms, this is equivalent to

$$\begin{aligned} & f(V_{2h}) + f(V - V_{2h-1} - V_{2h}) + f(V_{2h'}) + f(V - V_{2h'-1} - V_{2h'}) + f(W) \;\ge \\ & \quad f(W \cap V_{2h}) + f(W \cap V_{2h'}) + f(W - V_{2h} - V_{2h'}) + f(V - V_{2h-1} - V_{2h'-1}) + f(V). \end{aligned} \tag{3.8}$$

We can prove (3.8) by submodularity as follows.

$$\begin{aligned} (3.8) \;\Longleftarrow\; & f(W \cup V_{2h}) + f(V - V_{2h-1} - V_{2h}) + f(V_{2h'}) + f(V - V_{2h'-1} - V_{2h'}) \\ & \ge\; f(W \cap V_{2h'}) + f(W - V_{2h} - V_{2h'}) + f(V - V_{2h-1} - V_{2h'-1}) + f(V) \\ \Longleftarrow\; & f(W \cup V_{2h} \cup V_{2h'}) + f(V - V_{2h-1} - V_{2h}) + f(V - V_{2h'-1} - V_{2h'}) \\ & \ge\; f(W - V_{2h} - V_{2h'}) + f(V - V_{2h-1} - V_{2h'-1}) + f(V) \\ \Longleftarrow\; & f(W \cup V_{2h} \cup V_{2h'}) + f(V - V_{2h-1} - V_{2h} - V_{2h'-1} - V_{2h'}) \\ & \ge\; f(W - V_{2h} - V_{2h'}) + f(V - V_{2h-1} - V_{2h'-1}). \end{aligned}$$

(Note that the last inequality uses the fact $W \cap V_{2h-1} = W \cap V_{2h'-1} = \emptyset$.)

Similarly we consider sub-case (2a).

(2a) Recall that in sub-case (2a), we have $W^1, W^2 \in \mathcal{P}_\ell$ such that $\emptyset \ne W^1 \cap V_{2h} \subset W^1$, $\emptyset \ne W^2 \cap V_{2h'} \subset W^2$ and $W^1 \cap V_{2h-1} = W^2 \cap V_{2h'-1} = \emptyset$ hold for $1 \le h < h' \le \ell$. Split W^1 into $W^1 \cap V_{2h}$ and $W^1 - V_{2h}$, W^2 into $W^2 \cap V_{2h'}$ and $W^2 - V_{2h'}$. Let $\mathcal{P}^{2\mathrm{a}} =$

$$\{V_1, \ldots, V_{2h-2}, V_{2h+1}, \ldots, V_{2h'-2}, V_{2h'+1}, \ldots, V_{2\ell}, V_{2h-1}, V_{2h'-1}, V_{2h} \cup V_{2h'} \cup V_{2\ell+1}\}.$$

Notice that $\mathcal{P}^{2\mathrm{a}} = \mathcal{P}^{1\mathrm{b}}$. Thus it suffices to show (3.6). Since $(W^1 \cap V_{2h}, W^1, W^2 \cap V_{2h'}, W^2)$ is a candidate in Line 3 for GSA2, we have

$$\begin{aligned} f(\mathcal{P}_{\ell+1}) - f(\mathcal{P}_\ell) \;\le\; & f(W^1 \cap V_{2h}) + f(W^1 - V_{2h}) - f(W^1) \\ & + f(W^2 \cap V_{2h'}) + f(W^2 - V_{2h'}) - f(W^2). \end{aligned}$$

Therefore we only need to show the next inequality to complete the proof.

$$\begin{aligned} & f(V_{2h}) + f(V - V_{2h-1} - V_{2h}) + f(V_{2h'}) + f(V - V_{2h'-1} - V_{2h'}) + f(W^1) + f(W^2) \\ \geq\ & f(W^1 \cap V_{2h}) + f(W^1 - V_{2h}) + f(W^2 \cap V_{2h'}) + f(W^2 - V_{2h'}) \\ & \qquad + f(V - V_{2h-1} - V_{2h'-1}) + f(V). \end{aligned} \tag{3.9}$$

This can be done as follows.

$$\begin{aligned} (3.9) \quad \Longleftarrow \quad & f(W^1 \cup V_{2h}) + f(W^2 \cup V_{2h'}) + f(V - V_{2h-1} - V_{2h}) + f(V - V_{2h'-1} - V_{2h'}) \\ & \geq f(W^1 - V_{2h}) + f(W^2 - V_{2h'}) + f(V - V_{2h-1} - V_{2h'-1}) + f(V) \\ \Longleftarrow \quad & f(V - V_{2h-1}) + f(V - V_{2h'-1}) \geq f(V - V_{2h-1} - V_{2h'-1}) + f(V). \end{aligned}$$

Notice that fact $W^1 \cap V_{2h-1} = W^2 \cap V_{2h'-1} = \emptyset$ is used to obtain the second "$\Longleftarrow$".

(2b) Recall that in sub-case (2b), we have $W^1, W^2 \in \mathcal{P}_\ell$ such that $W^1 \cap V_{2h-1} \neq \emptyset \neq W^1 \cap V_{2h}$, $W^2 \cap V_{2h'-1} \neq \emptyset \neq W^2 \cap V_{2h'}$, $W^1 \subseteq V_{2h-1} \cup V_{2h}$ and $W^2 \subseteq V_{2h'-1} \cup V_{2h'}$ hold for $1 \leq h < h' \leq \ell$. Split W^1 into $W^1 \cap V_{2h-1}$ and $W^1 \cap V_{2h}$, W^2 into $W^2 \cap V_{2h'-1}$ and $W^2 \cap V_{2h'}$. Let $\mathcal{P}^{2b}$ =

$$\{V_1, \ldots, V_{2h-2}, V_{2h+1}, \ldots, V_{2h'-2}, V_{2h'+1}, \ldots, V_{2\ell}, V_{2h-1} \cup V_{2h}, V_{2h'-1} \cup V_{2h'}, V_{2\ell+1}\}.$$

Applying the induction hypothesis on $\mathcal{P}_\ell$ and $\mathcal{P}^{2b}$, we have

$$\begin{aligned} f(\mathcal{P}_\ell) \ \leq \ & \sum_{1 \leq i \leq \ell,\ i \neq h,h'} \Big(f(V_{2i-1}) + f(V_{2i}) + f(V - V_{2i-1} - V_{2i}) \Big) + \Big(f(V_{2h-1} \cup V_{2h}) \\ & + f(V_{2h'-1} \cup V_{2h'}) + f(V - V_{2h-1} - V_{2h} - V_{2h'-1} - V_{2h'}) \Big) - (\ell - 2) f(V). \end{aligned}$$

Thus, it suffices to show

$$\begin{aligned} f(\mathcal{P}_{\ell+1}) - f(\mathcal{P}_\ell) \ \leq \ & \Big(f(V_{2h-1}) + f(V_{2h}) + f(V - V_{2h-1} - V_{2h}) \Big) + \Big(f(V_{2h'-1}) + f(V_{2h'}) \\ & + f(V - V_{2h'-1} - V_{2h'}) \Big) - \Big(f(V_{2h-1} \cup V_{2h}) + f(V_{2h'-1} \cup V_{2h'}) \\ & + f(V - V_{2h-1} - V_{2h} - V_{2h'-1} - V_{2h'}) + f(V) \Big). \end{aligned} \tag{3.10}$$

Since $(W^1 \cap V_{2h}, W^1, W^2 \cap V_{2h'}, W^2)$ is a candidate in Line 3 for GSA2, we have

$$\begin{aligned} f(\mathcal{P}_{\ell+1}) - f(\mathcal{P}_\ell) \ \leq \ & f(W^1 \cap V_{2h-1}) + f(W^1 \cap V_{2h}) - f(W^1) \\ & + f(W^2 \cap V_{2h'-1}) + f(W^2 \cap V_{2h'}) - f(W^2). \end{aligned}$$

Hence we only need to show the next inequality.

$$\begin{aligned} & f(V_{2h-1}) + f(V_{2h}) + f(V - V_{2h-1} - V_{2h}) + f(W^1) \\ & \qquad + f(V_{2h'-1}) + f(V_{2h'}) + f(V - V_{2h'-1} - V_{2h'}) + f(W^2) \\ \geq\ & f(W^1 \cap V_{2h-1}) + f(W^1 \cap V_{2h}) + f(W^2 \cap V_{2h'-1}) + f(W^2 \cap V_{2h'}) + f(V_{2h-1} \cup V_{2h}) \\ & \qquad + f(V_{2h'-1} \cup V_{2h'}) + f(V - V_{2h-1} - V_{2h} - V_{2h'-1} - V_{2h'}) + f(V). \end{aligned} \tag{3.11}$$

In fact, by submodularity, we have

$$
\begin{aligned}
(3.11) \quad \Longleftarrow \quad & f(W^1 \cup V_{2h-1}) + f(V_{2h}) + f(V - V_{2h-1} - V_{2h}) \\
& \quad + f(W^2 \cup V_{2h'-1}) + f(V_{2h'}) + f(V - V_{2h'-1} - V_{2h'}) \\
\geq \ & f(W^1 \cap V_{2h}) + f(W^2 \cap V_{2h'}) + f(V_{2h-1} \cup V_{2h}) \\
& \quad + f(V_{2h'-1} \cup V_{2h'}) + f(V - V_{2h-1} - V_{2h} - V_{2h'-1} - V_{2h'}) + f(V) \\
\Longleftarrow \quad & f(V - V_{2h-1} - V_{2h}) + f(W^2 \cup V_{2h'-1}) + f(V_{2h'}) + f(V - V_{2h'-1} - V_{2h'}) \\
\geq \ & f(W^2 \cap V_{2h'}) + f(V_{2h'-1} \cup V_{2h'}) + f(V - V_{2h-1} - V_{2h} - V_{2h'-1} - V_{2h'}) + f(V) \\
\Longleftarrow \quad & f(V - V_{2h-1} - V_{2h}) + f(V - V_{2h'-1} - V_{2h'}) \\
\geq \ & f(V - V_{2h-1} - V_{2h} - V_{2h'-1} - V_{2h'}) + f(V).
\end{aligned}
$$

Notice that the facts $W^1 \subseteq V_{2h-1} \cup V_{2h}$ and $W^2 \subseteq V_{2h'-1} \cup V_{2h'}$ are used to obtain the second and the third "$\Longleftarrow$", respectively.

Finally we consider sub-case (2c).

(2c) Recall that in sub-case (2c), we have $W^1, W^2 \in \mathcal{P}_\ell$ such that $W^1 \cap V_{2h-1} \neq \emptyset \neq W^1 \cap V_{2h}$, $W^2 \cap V_{2h'} \neq \emptyset \subset W_2$, $W^1 \subseteq V_{2h-1} \cup V_{2h}$ and $W^2 \cap V_{2h'-1} = \emptyset$ hold for $1 \leq h < h' \leq \ell$. Split W^1 into $W^1 \cap V_{2h-1}$ and $W^1 \cap V_{2h}$, W^2 into $W^2 \cap V_{2h}$ and $W^2 - V_{2h'}$. Let $\mathcal{P}^{2c} =$

$$\{V_1, \ldots, V_{2h-2}, V_{2h+1}, \ldots, V_{2h'-2}, V_{2h'+1}, \ldots, V_{2\ell}, V_{2h-1} \cup V_{2h}, V_{2h'-1}, V_{2h'} \cup V_{2\ell+1}\}.$$

Applying the induction hypothesis on $\mathcal{P}_\ell$ and $\mathcal{P}^{2c}$, we have

$$
\begin{aligned}
f(\mathcal{P}_\ell) \ \leq \ & \sum_{1 \leq i \leq \ell,\ i \neq h, h'} \Big(f(V_{2i-1}) + f(V_{2i}) + f(V - V_{2i-1} - V_{2i}) \Big) + \Big(f(V_{2h-1} \cup V_{2h}) \\
& + f(V_{2h'-1}) + f(V - V_{2h-1} - V_{2h} - V_{2h'-1}) \Big) - (\ell - 2) f(V). \qquad (3.12)
\end{aligned}
$$

Thus, it suffices to show

$$
\begin{aligned}
f(\mathcal{P}_{\ell+1}) - f(\mathcal{P}_\ell) \ \leq \ & \Big(f(V_{2h-1}) + f(V_{2h}) + f(V - V_{2h-1} - V_{2h}) \Big) \\
& + \Big(f(V_{2h'}) + f(V - V_{2h'-1} - V_{2h'}) \Big) \\
& - \Big(f(V_{2h-1} \cup V_{2h}) + f(V - V_{2h-1} - V_{2h} - V_{2h'-1}) + f(V) \Big). \qquad (3.13)
\end{aligned}
$$

Since $(W^1 \cap V_{2h}, W^1, W^2 \cap V_{2h'}, W^2)$ is a candidate in Line 3 for GSA2, we have

$$
\begin{aligned}
f(\mathcal{P}_{\ell+1}) - f(\mathcal{P}_\ell) \ \leq \ & f(W^1 \cap V_{2h-1}) + f(W^1 \cap V_{2h}) - f(W^1) \\
& + f(W^2 \cap V_{2h'}) + f(W^2 - V_{2h'}) - f(W^2).
\end{aligned}
$$

Hence we only need to show the next inequality

$$\begin{aligned}
& f(V_{2h-1}) + f(V_{2h}) + f(V - V_{2h-1} - V_{2h}) + f(W^1) \\
& \qquad + f(V_{2h'}) + f(V - V_{2h'-1} - V_{2h'}) + f(W^2) \\
\geq\ & f(W^1 \cap V_{2h-1}) + f(W^1 \cap V_{2h}) + f(W^2 \cap V_{2h'}) + f(W^2 - V_{2h'}) \\
& \qquad + f(V_{2h-1} \cup V_{2h}) + f(V - V_{2h-1} - V_{2h} - V_{2h'-1}) + f(V).
\end{aligned} \tag{3.14}$$

This can be proved as follows.

$$\begin{aligned}
(3.14) \quad \Longleftarrow \quad & f(W^1 \cup V_{2h-1}) + f(V_{2h}) + f(V - V_{2h-1} - V_{2h}) \\
& \qquad + f(W^2 \cup V_{2h'}) + f(V - V_{2h'-1} - V_{2h'}) \\
\geq\ & f(W^1 \cap V_{2h}) + f(W^2 - V_{2h'}) + f(V_{2h-1} \cup V_{2h}) \\
& \qquad + f(V - V_{2h-1} - V_{2h} - V_{2h'-1}) + f(V) \\
\Longleftarrow \quad & f(V - V_{2h-1} - V_{2h}) + f(W^2 \cup V_{2h'}) + f(V - V_{2h'-1} - V_{2h'}) \\
\geq\ & f(W^2 - V_{2h'}) + f(V - V_{2h-1} - V_{2h} - V_{2h'-1}) + f(V) \\
\Longleftarrow \quad & f(V - V_{2h-1} - V_{2h}) + f(V - V_{2h'-1}) \ \geq\ f(V - V_{2h-1} - V_{2h} - V_{2h'-1}) + f(V).
\end{aligned}$$

Notice that the facts $W^1 \subseteq V_{2h-1} \cup V_{2h}$ and $W^2 \cap V_{2h'-1} = \emptyset$ are used to obtain the second and the third "$\Longleftarrow$", respectively. □

3.2.2 Algorithm description and main lemma for even k

For a positive even k, we start with a minimum 2-partition of V, and repeatedly increase the size of the partition by two greedily as we did before. It is described in Figure 3.3, where the same code as in Figure 3.1 is omitted.

Input: A submodular system (V, f) and an even integer $k \geq 2$.
Output: A k-partition $\mathcal{P}_{\frac{k}{2}}$ of V.

```
1        $\mathcal{P}_1 \leftarrow$ a minimum 2-partition of $V$
2        **for** $i = 1, \ldots, \frac{k}{2} - 1$ **do**
3 – 10       (The same code as Lines 3 – 10 in Figure 3.1)
```

Figure 3.3: Greedy splitting algorithm 2 (GSA2) for MPP-NT for even k.

Clearly the output $\mathcal{P}_{\frac{k}{2}}$ is a k-partition of V. In order to obtain a polynomial time algorithm, it also requires that Assumption 1 holds.

We now give a lemma on the performance.

Lemma 3.2. *Let $\mathcal{P}_\ell$ be the 2ℓ-partition of V found by GSA2 in the $(\ell-1)^{th}$ phase, $\ell = 1, 2, \ldots, \frac{k}{2}$. For any 2ℓ-partition $\mathcal{P} = \{V_1, V_2, \ldots, V_{2\ell}\}$ of V, it holds that*

$$\begin{aligned} f(\mathcal{P}_\ell) \;\leq\; & f(V_1) + f(V - V_1) \\ & + \sum_{i=1}^{\ell-1} (f(V_{2i}) + f(V_{2i+1}) + f(V - V_{2i} - V_{2i+1})) - (\ell-1)f(V). \end{aligned} \tag{3.15}$$

Proof. Again, we proceed by induction on ℓ. It is trivial for $\ell = 1$. Suppose that it holds for $\ell - 1$. We consider a 2ℓ-partition $\mathcal{P} = \{V_1, V_2, \ldots, V_{2\ell}\}$, where $\ell \geq 2$ holds.

Referring to the proof for Lemma 3.1, we see that at least one of cases (1) and (2) must happen. Before considering the sub-cases, we redefine the notation of "partners". Notice that (3.15) shows the symmetry between V_{2i} and V_{2i+1} for all i. Thus we say that V_{2i} and V_{2i+1} are partners of each other, $i = 1, 2, \ldots, \ell-1$.

Observation 3.1. *All the sub-cases* (1a), (1b), (2a), (2b) *and* (2c) *in the proof of Lemma 3.1 can happen with the new definition of partner. In each one sub-case of them, the proof can be done by fixing V_1 as the first member in the constructed nice partition.*

Hence we only need to consider such sub-cases that are not covered. First, let us consider case (1). The only sub-case that is not covered is the next.

(1c) $1 \in \{r, s, t\}$ and $|\{W \cap V_{2h}, W \cap V_{2h+1}\}| = 1$ hold for some $h \in \{1, \ldots, \ell-1\}$. Let $r = 1$ and suppose without loss of generality that $s = 2h$ (hence $W \cap V_{2h+1} = \emptyset$) holds. Split W into $W \cap V_1$, $W \cap V_{2h}$ and $W - V_1 - V_{2h}$. Let

$$\mathcal{P}^{1c} \;=\; \{V_{2h+1}, V_2, \ldots, V_{2h-1}, V_{2h+2}, \ldots, V_{2\ell-1}, V_1 \cup V_{2h} \cup V_{2\ell}\}.$$

(Note that $1 < 2h < 2h+1 < 2\ell$.) Applying the induction hypothesis on $\mathcal{P}_{\ell-1}$ and $\mathcal{P}^{1c}$, we have

$$\begin{aligned} f(\mathcal{P}_{\ell-1}) \;\leq\; & f(V_{2h+1}) + f(V - V_{2h+1}) \\ & + \sum_{1 \leq i \leq \ell-1,\ i \neq h} (f(V_{2i}) + f(V_{2i+1}) + f(V - V_{2i} - V_{2i+1})) - (\ell-2)f(V). \end{aligned} \tag{3.16}$$

Therefore, it suffices to show

$$\begin{aligned} f(\mathcal{P}_\ell) - f(\mathcal{P}_{\ell-1}) \;\leq\; & f(V_1) + f(V - V_1) + f(V_{2h}) + f(V - V_{2h} - V_{2h+1}) \\ & - f(V - V_{2h+1}) - f(V). \end{aligned} \tag{3.17}$$

Since $(W \cap V_1, W \cap V_{2h}, W)$ is a candidate in Line 4 for GSA2, we have

$$f(\mathcal{P}_\ell) - f(\mathcal{P}_{\ell-1}) \;\leq\; f(W \cap V_1) + f(W \cap V_{2h}) + f(W - V_1 - V_{2h}) - f(W). \tag{3.18}$$

Hence we only need to show

$$\begin{aligned} & f(V_1)+f(V-V_1)+f(V_{2h})+f(V-V_{2h}-V_{2h+1})+f(W) \\ \leq\ & f(W\cap V_1)+f(W\cap V_{2h})+f(W-V_1-V_{2h})+f(V-V_{2h+1})+f(V). \end{aligned} \tag{3.19}$$

In fact, this can be proved by submodularity as follows.

$$\begin{aligned} (3.19) \quad \Longleftarrow \quad & f(W\cup V_1)+f(V-V_1)+f(V_{2h})+f(V-V_{2h}-V_{2h+1}) \\ & \leq f(W\cap V_{2h})+f(W-V_1-V_{2h})+f(V-V_{2h+1})+f(V) \\ \Longleftarrow \quad & f(W\cup V_1\cup V_{2h})+f(V-V_1)+f(V-V_{2h}-V_{2h+1}) \\ & \leq f(W-V_1-V_{2h})+f(V-V_{2h+1})+f(V) \\ \Longleftarrow \quad & f((W-V_1)\cup V_{2h})+f(V-V_{2h}-V_{2h+1}) \\ & \leq f(W-V_1-V_{2h})+f(V-V_{2h+1}). \end{aligned}$$

Note that the fact $W\cap V_{2h+1}=\emptyset$ is used to obtain the last inequality.

Next, let us consider sub-cases of case (2). Sub-cases that are not covered by (2a), (2b) and (2c) are the following two, where we fix $a=1$ (since $1\in\{a,b,p,q\}$).

(2d) $|\{p,q\}\cap\{2h,2h+1\}|=1$ holds for some $1\leq h\leq \ell-1$. Without loss of generality, let $p=2h$. Thus $W^2\cap V_{2h+1}=\emptyset$ holds.

(2e) $\{p,q\}=\{2h,2h+1\}$ holds for some $1\leq h\leq \ell-1$. Hence $W^2\cap V_{2h}\neq\emptyset\neq W^2\cap V_{2h+1}$ and $W^2\subseteq V_{2h}\cup V_{2h+1}$ hold.

Let us first consider sub-case (2d).

(2d) Split W^1 into $W^1\cap V_1$ and W^1-V_1, W^2 into $W^2\cap V_{2h}$ and W^2-V_{2h}. Let

$$\mathcal{P}^{\rm 2d} \;=\; \{V_{2h+1},V_2,\ldots,V_{2h-1},V_{2h+2},\ldots,V_{2\ell-1},V_1\cup V_{2h}\cup V_{2\ell}\}.$$

Notice that $\mathcal{P}^{\rm 2d}=\mathcal{P}^{\rm 1c}$ holds. Hence we only need to show (3.17). Since $(W^1\cap V_1,W^1,W^2\cap V_{2h},W^2)$ is a candidate in Line 3 for GSA2, we have

$$\begin{aligned} f(\mathcal{P}_\ell)-f(\mathcal{P}_{\ell-1}) \;\leq\; & f(W^1\cap V_1)+f(W^1-V_1)-f(W^1) \\ & + f(W^2\cap V_{2h})+f(W^2-V_{2h})-f(W^2). \end{aligned} \tag{3.20}$$

Therefore we only need to show

$$\begin{aligned} & f(V_1)+f(V-V_1)+f(V_{2h})+f(V-V_{2h}-V_{2h+1})+f(W^1)+f(W^2) \\ \geq\ & f(W^1\cap V_1)+f(W^1-V_1)+f(W^2\cap V_{2h})+f(W^2-V_{2h}) \\ & \qquad + f(V-V_{2h+1})+f(V). \end{aligned} \tag{3.21}$$

This is proved by submodularity as follows.

$$\begin{aligned}
(3.21) \quad \Longleftarrow \quad & f(W^1 \cup V_1) + f(V - V_1) + f(W^2 \cup V_{2h}) + f(V - V_{2h} - V_{2h+1}) \\
& \geq f(W^1 - V_1) + f(W^2 - V_{2h}) + f(V - V_{2h+1}) + f(V) \\
\Longleftarrow \quad & f(W^2 \cup V_{2h}) + f(V - V_{2h} - V_{2h+1}) \geq f(W^2 - V_{2h}) + f(V - V_{2h+1})
\end{aligned}$$

Note that the fact $W^2 \cap V_{2h+1} = \emptyset$ is used to obtain the last inequality.

Finally, we consider sub-case (2e).

(2e) Split W^1 into $W^1 \cap V_1$ and $W^1 - V_1$, W^2 into $W^2 \cap V_{2h}$ and $W^2 \cap V_{2h+1}$. Let

$$\mathcal{P}^{2e} = \{V_{2h} \cup V_{2h+1}, V_2, \ldots, V_{2h-1}, V_{2h+2}, \ldots, V_{2\ell-1}, V_1 \cup V_{2\ell}\}.$$

Applying the induction hypothesis on $\mathcal{P}_{\ell-1}$ and $\mathcal{P}^{2e}$, we have

$$\begin{aligned}
f(\mathcal{P}_{\ell-1}) \leq \; & f(V_{2h} \cup V_{2h+1}) + f(V - V_{2h} - V_{2h+1}) \\
& + \sum_{1 \leq i \leq \ell-1,\ i \neq h} (f(V_{2i}) + f(V_{2i+1}) + f(V - V_{2i} - V_{2i+1})) - (\ell-2)f(V). \quad (3.22)
\end{aligned}$$

Therefore, it suffices to show

$$f(\mathcal{P}_\ell) - f(\mathcal{P}_{\ell-1}) \leq f(V_1) + f(V - V_1) + f(V_{2h}) + f(V_{2h+1}) - f(V_{2h} \cup V_{2h+1}) - f(V). \quad (3.23)$$

Since $(W^1 \cap V_1, W^1, W^2 \cap V_{2h}, W^2)$ is a candidate in Line 3 for GSA2, we have

$$\begin{aligned}
f(\mathcal{P}_\ell) - f(\mathcal{P}_{\ell-1}) \leq \; & f(W^1 \cap V_1) + f(W^1 - V_1) - f(W^1) \\
& + f(W^2 \cap V_{2h}) + f(W^2 \cap V_{2h+1}) - f(W^2). \quad (3.24)
\end{aligned}$$

Therefore we only need to show

$$\begin{aligned}
& f(V_1) + f(V - V_1) + f(V_{2h}) + f(V_{2h+1}) + f(W^1) + f(W^2) \\
\geq \; & f(W^1 \cap V_1) + f(W^1 - V_1) + f(W^2 \cap V_{2h}) + f(W^2 \cap V_{2h+1}) \\
& + f(V_{2h} \cup V_{2h+1}) + f(V). \quad (3.25)
\end{aligned}$$

This can be proved by submodularity as follows.

$$\begin{aligned}
(3.25) \quad \Longleftarrow \quad & f(W^1 \cup V_1) + f(W^2 \cup V_{2h}) + f(V - V_1) + f(V_{2h+1}) \\
& \leq f(W^1 - V_1) + f(W^2 \cap V_{2h+1}) + f(V_{2h} \cup V_{2h+1}) + f(V) \\
\Longleftarrow \quad & f(W^2 \cup V_{2h}) + f(V_{2h+1}) \leq f(W^2 \cap V_{2h+1}) + f(V_{2h} \cup V_{2h+1}).
\end{aligned}$$

Notice that the fact $W^2 \subseteq V_{2h} \cup V_{2h+1}$ is used to obtain the last inequality. □

3.3 Performance analysis of GSA2

3.3.1 Polynomial time implementation

We have noted that, GSA2 has polynomial running time if the next problem, called *partial 3-partition problem*, can be solved in polynomial time (Assumption 1 in Section 3.2).

Problem 3.1 (Partial 3-partition problem). *Given a submodular system (V, f), $W \subseteq V$ ($|W| \geq 3$), find a 3-partition $\{S_1^*, S_2^*, S_3^*\}$ of W that minimizes $f(S_1^*) + f(S_2^*) + f(S_3^*)$.*

Obviously, the partial 3-partition problem reduces to 3-PPSS in submodular system $(W, f|_W)$, where $f|_W$ denotes the restriction of f to W. One may think that, if f is symmetric, then it can be solved by the algorithm [Que 99] for 3-PPSSS. However, this does not work, since function $f|_W$ can lose the symmetric property when restricted to W. An interesting question is thus to find problem classes for which the partial 3-partition problem can be solved in polynomial time. The next proposition shows one of such problem classes: k-PPG.

Proposition 3.1. *Given a graph $G = (V, E)$ with a weight function $w : E \to \mathbf{R}^+$, a 3-partition $\{S_1^*, S_2^*, S_3^*\}$ of a $W \subseteq V$ with $|W| \geq 3$ that minimizes $w_{\text{ex}}(S_1^*) + w_{\text{ex}}(S_2^*) + w_{\text{ex}}(S_3^*)$ can be found in polynomial time.*

Proof. Equivalently, consider to minimize $w_{\text{ex}}(S_1) + w_{\text{ex}}(S_2) + w_{\text{ex}}(S_3) - w_{\text{ex}}(W)$ over all 3-partitions $\{S_1, S_2, S_3\}$ of W. This is nothing but 3-PPG in the reduced subgraph $G[W]$ of G. Thus it can be solved in polynomial time ([GH 88]). □

Since 3-PPG can be solved in $O(mn^3 \log(n^2/m))$ time for a graph with n vertices and m edges ([NI 00]), we have the next corollary.

Corollary 3.1. *GSA2 for k-PPG in a graph with n vertices and m edges can be implemented in $O(kmn^3 \log(n^2/m))$ running time.* □

Problem k-PPG is not the only class for which the partial 3-partition problem can be solved in polynomial time. A slightly generalization, i.e., k-PPH-T2 with $d^+_{\max} \leq 3$, is available. See the next observation.

Observation 3.2. *The partial 3-partition problem for k-PPH-T1 (resp. k-PPH-T3) can be reduced to 3-PPH-T1 (3-PPH-T3). For k-PPH-T2, it reduces to k-PPH-T2 when $d^+_{\max} \leq 3$ holds, where $d^+_{\max}$ is the maximum degree of hyperedges with positive weights.*

This observation can be shown by the same ideas in proofs for Theorems 2.8, 2.9 and 2.10. Notice that the idea in Theorem 2.8 does not work in general. Since the complexities for 3-PPH-T1 and 3-PPH-T3 are still open, the question of whether GSA2 can have polynomial running time or not remains *open* in general for any one of k-PPH-T1, k-PPH-T2 and k-PPH-T3.

3.3.2 Performance guarantee: part I

Assuming that GSA2 has polynomial running time, we derive its performance guarantees.

We first note that, not surprisingly, GSA2 does no worse than GSA. This can be seen by comparing the right-hand side of (2.12) with the right-hand sides of (3.1) and (3.15). Notice that $f(V-X-Y)+f(V) \leq f(V-X)+f(V-Y)$ holds for any disjoint $X, Y \subseteq V$.

In fact, using Lemma 3.1 and Lemma 3.2, we have the next result.

Theorem 3.1. *The performance guarantee of GSA2 is $\lfloor \frac{k}{2} \rfloor$ for k-PPSS, and $2-\frac{2}{k}$ for k-PPSSS $(k \geq 2)$. There are examples indicating that these bounds are tight.*

Proof. First consider k-PPSS. Let $\mathcal{P}^* = \{V_1^*, V_2^*, \ldots, V_k^*\}$ be a minimum k-partition.

Let k be an odd number. By Lemma 3.1, GSA2 finds a k-partition whose weight is at most

$$\sum_{i=1}^{\frac{k-1}{2}} \left(f(V_{2i-1}^*) + f(V_{2i}^*) + f(V - V_{2i-1}^* - V_{2i}^*)\right). \tag{3.26}$$

(Note $f(V) \geq 0$.) Notice that

$$f(V - V_{2i-1}^* - V_{2i}^*) \; = \; f(\bigcup_{j \neq 2i-1, 2i} V_j^*) \; \leq \; \sum_{j \neq 2i-1, 2i} f(V_j^*).$$

(Note $f(\emptyset) \geq 0$.) Hence (3.26) is at most $\frac{k-1}{2} \sum_{i=1}^{k} f(V_i^*)$.

Similarly, let us consider an even k. By Lemma 3.2, GSA2 finds a k-partition whose weight is at most

$$\begin{aligned} & f(V_1^*) + f(V - V_1^*) + \sum_{i=1}^{\frac{k}{2}-1} \left(f(V_{2i-1}^*) + f(V_{2i}^*) + f(V - V_{2i-1}^* - V_{2i}^*)\right) \\ \leq \; & \sum_{i=1}^{k} f(V_i^*) + \sum_{i=1}^{\frac{k}{2}-1} \sum_{i=1}^{k} f(V_i^*) \; = \; \frac{k}{2} \sum_{i=1}^{k} f(V_i^*). \end{aligned} \tag{3.27}$$

Hence the guarantee of GSA2 is $\lfloor \frac{k}{2} \rfloor$ for k-PPSS.

The performance guarantee $2-\frac{2}{k}$ for k-PPSSS is trivial, since we have noted that GSA2 performs at least as good as GSA. Tight examples will be given in Section 3.4. □

Let us now consider k-PPMSS.

Theorem 3.2. *The performance guarantee of GSA2 for MPP-NT in submodular system (V, f) satisfying $f(\emptyset) \geq 0$ and $f(V) \geq f(S)$ for all $\emptyset \neq S \subset V$ is $2-\frac{3}{k}$ for any $k \geq 3$. In particular, this is true for k-PPMSS.*

Proof. Let $\mathcal{P}^* = \{V_1^*, V_2^*, \ldots, V_k^*\}$ be a minimum k-partition with the order $f(V_1^*) \leq f(V_2^*) \leq \cdots \leq f(V_{k-1}^*) \leq f(V_k^*)$. Let $k \geq 3$ be an odd number. By Lemma 3.1, GSA2 finds a k-partition whose weight is at most

$$\begin{aligned}
& \sum_{i=1}^{\frac{k-1}{2}} \big(f(V_{2i-1}^*) + f(V_{2i}^*) + f(V - V_{2i-1}^* - V_{2i}^*)\big) - (\frac{k-1}{2} - 1)f(V) \\
\leq\ & \sum_{i=1}^{k-1} f(V_i^*) + f(V - V_{k-2}^* - V_{k-1}^*) \\
\leq\ & 2\sum_{i=1}^{k} f(V_i^*) - (f(V_{k-2}^*) + f(V_{k-1}^*) + f(V_k^*)) \ \leq\ (2 - \frac{3}{k})\sum_{i=1}^{k} f(V_i^*).
\end{aligned}$$

Similarly, we can show the theorem for an even $k \geq 4$. □

An interesting and somewhat strange result is, while GSA2 improves the guarantee of GSA for k-PPSS and k-PPMSS, it performs as bad as GSA for k-PPSSS. In fact, we later will see that, there exist instances of k-PPSSS for which no guarantee better than $2 - \frac{2}{k}$ can be obtained, even the size of partition is increased by j per phase for any $j \geq 1$.

3.3.3 Performance guarantee: part II

In this subsection, we show that GSA2 performs better than GSA for a special case k-PPH-T2 (hence k-PPG) of k-PPSSS. We also show that improved guarantees can be obtained for k-PPH-T1, and for some cases of k-PPH-T3. We assume that GSA2 runs in polynomial time, which is true for k-PPH-T2 with $d_{\max}^+ \leq 3$ (hence k-PPG) as stated in Corollary 3.1 and Observation 3.2.

Theorem 3.3. *The performance guarantee of GSA2 for k-PPH-T2 is $2 - \frac{3}{k}$ for odd $k \geq 3$, and $2 - \frac{3}{k} + \frac{1}{k^2-k}$ for even $k \geq 2$.*

Proof. Let $H = (V, E)$ be a hypergraph with a weight function $w : E \to \mathbf{R}^+$. Denote the cut function by $f = w_{\text{ex}}$. Let $\mathcal{P}^* = \{V_1^*, V_2^*, \ldots, V_k^*\}$ be a minimum k-partition. Let π denote a permutation of $\{1, \ldots, k\}$, where $\pi(i)$ denotes the number of i in π.

First we consider an odd $k \geq 3$. By Lemma 3.1, for any permutation π, GSA2 finds a k-partition whose weight is at most

$$f_\pi = \sum_{i=1}^{\frac{k-1}{2}} \Big(w_{\text{ex}}(V_{\pi(2i-1)}^*) + w_{\text{ex}}(V_{\pi(2i)}^*) + w_{\text{ex}}(V - V_{\pi(2i-1)}^* - V_{\pi(2i)}^*)\Big). \tag{3.28}$$

To show the theorem for an odd k, we want to show that there is a permutation π^* such that

$$f_{\pi^*} \leq (2 - \frac{3}{k})\sum_{i=1}^{k} w_{\text{ex}}(V_i^*) = (2 - \frac{3}{k})f(\mathcal{P}^*). \tag{3.29}$$

Inequality (3.29) can be proved by considering all permutations and showing that the average value of f_π is at most $(2 - \frac{3}{k})f(\mathcal{P}^*)$. Alternatively, letting

$$\begin{aligned}\Delta_\pi &= 2f(P^*) - f_\pi \\ &= 2w_{\mathrm{ex}}(V^*_{\pi(k)}) + \sum_{i=1}^{\frac{k-1}{2}} \left(w_{\mathrm{ex}}(V^*_{\pi(2i-1)}) + w_{\mathrm{ex}}(V^*_{\pi(2i)}) - w_{\mathrm{ex}}(V - V^*_{\pi(2i-1)} - V^*_{\pi(2i)})\right),\end{aligned}$$

we only need to show that, *the average value of Δ_π is at least $\frac{3}{k}f(\mathcal{P}^*)$*. We prove this fact by showing that, for any hyperedge e (of weight $w(e)$), the average number that $w(e)$ is counted in Δ_π is at least $\frac{3}{k}$ times of the number that $w(e)$ is counted in $f(\mathcal{P}^*)$.

For simplicity, we contract each V_i^* to a single node v_i. This may decrease the degree of e. Let $H|_{\mathcal{P}^*}$ denote the hypergraph obtained after contraction. To avoid confusing, we use "node" to denote v_i in $H|_{\mathcal{P}^*}$. We assume that $H|_{\mathcal{P}^*}$ is complete and simple; i.e., the set of hyperedge in $H|_{\mathcal{P}^*}$ consists of all possible hyperedges, and there are no multiple hyperedges. Otherwise we can realize this assumption by adding zero weight hyperedges and merging multiple hyperedges.

Let the degree of e after contraction be d. We may assume that $d \geq 2$; since otherwise $w(e)$ is not counted in Δ_π or $f(\mathcal{P}^*)$. Recall that $w_{\mathrm{ex}}(S)$ is the sum of weights of hyperedges that have at least one but not all endpoints in S for a set $S \subseteq V$. Thus $w(e)$ is counted, due to the term of $2w_{\mathrm{ex}}(V^*_{\pi(k)})$ in Δ_π, twice if one endpoint of e (in hypergraph $H|_{\mathcal{P}^*}$) has number k. Since $H|_{\mathcal{P}^*}$ has k nodes, we see that the average number (i.e., expected value) that $w(e)$ is counted due to term $2w_{\mathrm{ex}}(V^*_{\pi(k)})$ is $\frac{2d}{k}$.

On the other hand, due to term $\sum_{i=1}^{\frac{k-1}{2}}(w_{\mathrm{ex}}(V^*_{\pi(2i-1)})+w_{\mathrm{ex}}(V^*_{\pi(2i)})-w_{\mathrm{ex}}(V-V^*_{\pi(2i-1)}-V^*_{\pi(2i)}))$ in Δ_π, if $d = 2$ and the two endpoints of e are numbered $\pi(2i-1)$ and $\pi(2i)$ for some i, then $w(e)$ is counted twice. Otherwise, $w(e)$ is not counted if $d = 2$, whereas it is counted p times if $d \geq 3$, where p is the number of pairs of endpoints of e that are numbered $\pi(2i-1)$ and $\pi(2i)$ for some i.

Notice that for each pair of indices $2i-1$ and $2i$, the average number (expected value) that both nodes v_{2i-1} and v_{2i} are endpoints of e is

$$\frac{\binom{k-2}{d-2}}{\binom{k}{d}} = \frac{d(d-1)}{k(k-1)}.$$

Since there are $\frac{k-1}{2}$ pairs of indices, the average number (expected value) that $w(e)$ is counted due to term $\sum_{i=1}^{\frac{k-1}{2}}(w_{\mathrm{ex}}(V^*_{\pi(2i-1)}) + w_{\mathrm{ex}}(V^*_{\pi(2i)}) - w_{\mathrm{ex}}(V - V^*_{\pi(2i-1)} - V^*_{\pi(2i)}))$ is thus

$$2 \cdot \frac{k-1}{2} \cdot \frac{2\cdot(2-1)}{k(k-1)} = \frac{2}{k} \quad \text{if } d = 2;$$

or

$$\frac{k-1}{2} \cdot \frac{d(d-1)}{k(k-1)} = \frac{d(d-1)}{2k} \quad \text{if } d \geq 3.$$

Since $w(e)$ is counted d times in the optimum $f(\mathcal{P}^*)$, we see that the contribution of $w(e)$ to the average value of Δ_π is

$$\frac{1}{2} \cdot \left(\frac{4}{k} + \frac{2}{k}\right) = \frac{3}{k} \quad (d=2) \quad \text{and} \quad \frac{1}{d} \cdot \left(\frac{2d}{k} + \frac{d(d-1)}{2k}\right) \geq \frac{3}{k} \quad (d \geq 3)$$

times of the contribution to $f(\mathcal{P}^*)$. Thus the average value of Δ_π is at least $\frac{3}{k} f(\mathcal{P}^*)$, proving the theorem for odd k.

Similarly, for an even k, let

$$\begin{aligned}\Delta_\pi &= 2f(P^*) - f_\pi \\ &= 2w_{\text{ex}}(V^*_{\pi(k)}) + \sum_{i=1}^{\frac{k}{2}-1} \left(w_{\text{ex}}(V^*_{\pi(2i)}) + w_{\text{ex}}(V^*_{\pi(2i+1)}) - w_{\text{ex}}(V - V^*_{\pi(2i)} - V^*_{\pi(2i+1)})\right).\end{aligned}$$

Let e be a hyperedge whose degree after contraction is $d \geq 2$. Again, the average number that $w(e)$ is counted due to term $2w_{\text{ex}}(V^*_{\pi(k)})$ is $\frac{2d}{k}$. Due to other terms, it is

$$2 \cdot \left(\frac{k}{2} - 1\right) \cdot \frac{2 \cdot (2-1)}{k(k-1)} = \frac{2(k-2)}{k(k-1)} \quad \text{if } d = 2;$$

or

$$\left(\frac{k}{2} - 1\right) \cdot \frac{d(d-1)}{k(k-1)} = \frac{d(d-1)(k-2)}{2k(k-1)} \quad \text{if } d \geq 3.$$

Therefore the contribution of e to the average value of Δ_π is

$$\frac{1}{2} \cdot \left(\frac{4}{k} + \frac{2(k-2)}{k(k-1)}\right) = \frac{3}{k} - \frac{1}{k^2-k} \quad (d=2)$$

and

$$\frac{1}{d} \cdot \left(\frac{2d}{k} + \frac{d(d-1)(k-2)}{2k(k-1)}\right) \geq \frac{3}{k} - \frac{1}{k^2-k} \quad (d \geq 3)$$

times of the contribution to $f(\mathcal{P}^*)$. This proves the theorem for an even k. □

Combining this result with Corollary 3.1, we have the next corollary.

Corollary 3.2. *Problem k-PPG in a graph with n vertices and m edges can be approximated by GSA2 within a factor of* $2 - \frac{3}{k}$ *for any odd* $k \geq 3$, *and within a factor of* $2 - \frac{3}{k} + \frac{1}{k^2-k}$ *for any even* $k \geq 2$, *in* $O(kmn^3 \log(n^2/m))$ *time.* □

Let us now consider k-PPH-T1. We have the next proposition.

Proposition 3.2. *The guarantee of GSA2 for k-PPH-T1 is* $(1-\frac{3}{2k})\min\{k, d^+_{\max}\}$ *for odd* $k \geq 3$, *and* $(1-\frac{3k-4}{2k(k-1)})\min\{k, d^+_{\max}\}$ *for even* $k \geq 2$, *where* $d^+_{\max}$ *is the maximum degree of hyperedges of positive weights.*

Proof. Let $H = (V, E)$ be a hypergraph with a weight function $w : E \to \mathbf{R}^+$. Let $f = \frac{w(E)}{k} - w_{\text{in}}$ be the function employed by k-PPH-T1 (recall that $w_{\text{in}}(S) = \sum_{e \subseteq S} w(e)$). Let $\mathcal{P}^* = \{V_1^*, V_2^*, \ldots, V_k^*\}$ be a minimum k-partition of V with respect to f. Let π denote a permutation of $\{1, \ldots, k\}$, and let $\pi(i)$ be the number of i in π.

First consider an odd $k \geq 3$. By Lemma 3.1, for any permutation π, GSA2 finds a k-partition whose weight is at most

$$\begin{aligned}
f_\pi &= \sum_{i=1}^{\frac{k-1}{2}} \Big(f(V^*_{\pi(2i-1)}) + f(V^*_{\pi(2i)}) + f(V - V^*_{\pi(2i-1)} - V^*_{\pi(2i)})\Big) - \left(\frac{k-1}{2} - 1\right) f(V) \\
&= \sum_{i=1}^{\frac{k-1}{2}} \left(\frac{3w(E)}{k} - w_{\text{in}}(V^*_{\pi(2i-1)}) - w_{\text{in}}(V^*_{\pi(2i)}) - w_{\text{in}}(V - V^*_{\pi(2i-1)} - V^*_{\pi(2i)})\right) \\
&\qquad - \left(\frac{k-1}{2} - 1\right)\left(\frac{w(E)}{k} - w_{\text{in}}(V)\right) \\
&= \sum_{i=1}^{\frac{k-1}{2}} \Big(w(E) - w_{\text{in}}(V^*_{\pi(2i-1)}) - w_{\text{in}}(V^*_{\pi(2i)}) - w_{\text{in}}(V - V^*_{\pi(2i-1)} - V^*_{\pi(2i)})\Big) \\
&= \sum_{i=1}^{\frac{k-1}{2}} \Big(w_{\text{ex}}(V^*_{\pi(2i-1)}) + w_{\text{ex}}(V^*_{\pi(2i)}) - w\big(\{e \in E \mid e \cap V^*_{\pi(2i-1)} \neq \emptyset \neq V^*_{\pi(2i)}\}\big)\Big) \\
&= \sum_{i=1}^{k} w_{\text{ex}}(V_i^*) - w_{\text{ex}}(V^*_{\pi(k)}) - \sum_{i=1}^{\frac{k-1}{2}} w\left(\{e \in E \mid e \cap V^*_{\pi(2i-1)} \neq \emptyset \neq V^*_{\pi(2i)}\}\right). \qquad (3.30)
\end{aligned}$$

Let us consider the average value of f_π. Let

$$\Delta_\pi = \sum_{i=1}^{k} w_{\text{ex}}(V_i^*) - f_\pi = w_{\text{ex}}(V^*_{\pi(k)}) + \sum_{i=1}^{\frac{k-1}{2}} w\left(\{e \in E \mid e \cap V^*_{\pi(2i-1)} \neq \emptyset \neq V^*_{\pi(2i)}\}\right).$$

For each hyperedge e, we consider the average number that $w(e)$ is counted in Δ_π. Let $H|_{\mathcal{P}^*}$ be the hypergraph obtained by contracting each V_i^* to a single node v_i. Assume without loss of generality that $H|_{\mathcal{P}^*}$ is complete and simple.

Suppose that after contraction, e has degree $d \geq 2$. Due to the term of $w_{\text{ex}}(V^*_{\pi(k)})$, $w(e)$ is counted once if one endpoint of e (in $H|_{\mathcal{P}^*}$) is numbered k. Therefore the average number that $w(e)$ is counted due to term $w_{\text{ex}}(V^*_{\pi(k)})$ is $\frac{d}{k}$.

On the other hand, due to other terms in Δ_π, $w(e)$ is counted p times if there are p pairs of endpoints of e that are numbered $\pi(2i-1)$ and $\pi(2i)$ for some i. Thus the average number that $w(e)$ is counted due to term $\sum_{i=1}^{\frac{k-1}{2}} w\left(\{e \in E \mid e \cap V^*_{\pi(2i-1)} \neq \emptyset \neq V^*_{\pi(2i)}\}\right)$ is

$$\frac{k-1}{2} \cdot \frac{\binom{k-2}{d-2}}{\binom{k}{d}} = \frac{k-1}{2} \cdot \frac{d(d-1)}{k(k-1)} = \frac{d(d-1)}{2k}.$$

Since $w(e)$ is counted d times in $\sum_{i=1}^{k} w_{\text{ex}}(V_i^*)$, we see that the contribution of $w(e)$ to the average value of Δ_π is

$$\frac{1}{d}\left(\frac{d}{k} + \frac{d(d-1)}{2k}\right) \geq \frac{3}{2k}$$

times of the contribution to $\sum_{i=1}^{k} w_{\text{ex}}(V_i^*)$. Thus the average value of f_π is at most

$$(1 - \frac{3}{2k}) \sum_{i=1}^{k} w_{\text{ex}}(V_i^*) \leq (1 - \frac{3}{2k}) \min\{k, d^+_{\max}\} f(\mathcal{P}^*),$$

proving the theorem for odd k (recall that $\sum_{i=1}^{k} w_{\text{ex}}(V_i^*) \leq \min\{k, d^+_{\max}\} f(\mathcal{P}^*)$).

We can apply a similar argument for even k. By Lemma 3.2, for any permutation π, GSA2 finds a k-partition whose weight is at most

$$\begin{aligned} f_\pi &= f(V^*_{\pi(1)}) + f(V - V^*_{\pi(1)}) \\ &\quad + \sum_{i=1}^{\frac{k}{2}-1} \left(f(V^*_{\pi(2i)}) + f(V^*_{\pi(2i+1)}) + f(V - V^*_{\pi(2i)} - V^*_{\pi(2i+1)})\right) - (\frac{k}{2} - 1) f(V) \\ &= \sum_{i=1}^{k} w_{\text{ex}}(V_i^*) - w_{\text{ex}}(V^*_{\pi(k)}) - \sum_{i=1}^{\frac{k}{2}-1} w\left(\{e \in E \mid e \cap V^*_{\pi(2i)} \neq \emptyset \neq V^*_{\pi(2i+1)}\}\right). \qquad (3.31) \end{aligned}$$

Let

$$\Delta_\pi = \sum_{i=1}^{k} w_{\text{ex}}(V_i^*) - f_\pi = w_{\text{ex}}(V^*_{\pi(k)}) + \sum_{i=1}^{\frac{k}{2}-1} w\left(\{e \in E \mid e \cap V^*_{\pi(2i)} \neq \emptyset \neq V^*_{\pi(2i+1)}\}\right).$$

Let e be an hyperedge of degree $d \geq 2$ after contraction. The average number that $w(e)$ is counted due to term $w_{\text{ex}}(V^*_{\pi(k)})$ is $\frac{d}{k}$. The average number due to other terms is

$$\left(\frac{k}{2} - 1\right) \frac{d(d-1)}{k(k-1)} = \frac{d(d-1)(k-2)}{2k(k-1)}.$$

Thus the contribution of e to the average value of Δ_π is

$$\frac{1}{d}\left(\frac{d}{k} + \frac{d(d-1)(k-2)}{2k(k-1)}\right) \geq \frac{3k-4}{2k(k-1)}$$

times of the contribution to $\sum_{i=1}^{k} w_{\text{ex}}(V_i^*)$. Therefore the average value of f_π is at most

$$\left(1 - \frac{3k-4}{2k(k-1)}\right) \sum_{i=1}^{k} w_{\text{ex}}(V_i^*) \leq \left(1 - \frac{3k-4}{2k(k-1)}\right) \min\{k, d^+_{\max}\} f(\mathcal{P}^*),$$

proving of the theorem for even k. □

Theorem 3.4. *The performance guarantee of GSA2 for k-PPH-T3 is* $2-\frac{3}{k}$ *(resp.* $2-\frac{3k-4}{k(k-1)}$*) for odd* $k \geq 3$ *(even* $k \geq 2$*) if* $d^+_{\max} = 2$*. Otherwise, it is* $2-\frac{2}{k}$ *for any* $k \geq 2$*, where* $d^+_{\max}$ *is the maximum degree of hyperedges with positive weights.*

Proof. Again, let $H = (V, E)$ be a hypergraph with a weight function $w : E \to \mathbf{R}^+$. Let $f = w_{\text{in}} + w_{\text{ex}} - \frac{w(E)}{k}$ be the function employed by k-PPH-T3. Let $\mathcal{P}^* = \{V_1^*, V_2^*, \ldots, V_k^*\}$ be a minimum k-partition of V with respect to f. Let π denote a permutation of $\{1, \ldots, k\}$, and let $\pi(i)$ be the number of i in π.

First consider an odd number $k \geq 3$. By Lemma 3.1, for any permutation π, GSA2 finds a k-partition whose weight is at most

$$\begin{aligned}
f_\pi &= \sum_{i=1}^{\frac{k-1}{2}} \Big(f(V^*_{\pi(2i-1)}) + f(V^*_{\pi(2i)}) + f(V - V^*_{\pi(2i-1)} - V^*_{\pi(2i)})\Big) - \left(\frac{k-1}{2} - 1\right) f(V) \\
&= \sum_{i=1}^{\frac{k-1}{2}} \Bigg(w_{\text{in}}(V^*_{\pi(2i-1)}) + w_{\text{ex}}(V^*_{\pi(2i-1)}) + w_{\text{in}}(V^*_{\pi(2i)}) + w_{\text{ex}}(V^*_{\pi(2i)}) \\
&\qquad + w_{\text{in}}(V - V^*_{\pi(2i-1)} - V^*_{\pi(2i)}) + w_{\text{ex}}(V - V^*_{\pi(2i-1)} - V^*_{\pi(2i)}) - \frac{3w(E)}{k}\Bigg) \\
&\qquad\qquad - \left(\frac{k-1}{2} - 1\right)\left(1 - \frac{1}{k}\right) w(E) \\
&= \sum_{i=1}^{\frac{k-1}{2}} \Bigg(w_{\text{in}}(V^*_{\pi(2i-1)}) + w_{\text{ex}}(V^*_{\pi(2i-1)}) + w_{\text{in}}(V^*_{\pi(2i)}) + w_{\text{ex}}(V^*_{\pi(2i)}) \\
&\qquad + w_{\text{in}}(V - V^*_{\pi(2i-1)} - V^*_{\pi(2i)}) + w_{\text{ex}}(V - V^*_{\pi(2i-1)} - V^*_{\pi(2i)}) - w(E)\Bigg) \\
&= \sum_{i=1}^{\frac{k-1}{2}} \Big(w_{\text{ex}}(V^*_{\pi(2i-1)}) + w_{\text{ex}}(V^*_{\pi(2i)}) - w(E(V^*_{\pi(2i-1)} : V^*_{\pi(2i)}))\Big) \\
&= \sum_{i=1}^{k} w_{\text{ex}}(V_i^*) - w_{\text{ex}}(V^*_{\pi(k)}) - \sum_{i=1}^{\frac{k-1}{2}} w\left(E(V^*_{\pi(2i-1)} : V^*_{\pi(2i)})\right).
\end{aligned} \tag{3.32}$$

(Recall that $E(X : Y) \triangleq \{e \in E \mid e \subseteq X \cup Y,\ e \cap X \neq \emptyset \neq e \cap Y\}$ for any $X, Y \subseteq V$.)

Let us consider the average value of f_π. Let

$$\Delta_\pi = \sum_{i=1}^{k} w_{\text{ex}}(V_i^*) - f_\pi = w_{\text{ex}}(V^*_{\pi(k)}) + \sum_{i=1}^{\frac{k-1}{2}} w\left(E(V^*_{\pi(2i-1)} : V^*_{\pi(2i)})\right).$$

Let e be an hyperedge of degree d after contracting each V_i^* to a node v_i. The average number that $w(e)$ is counted due to the term of $w_{\text{ex}}(V^*_{\pi(k)})$ is $\frac{d}{k}$. The average number due to other terms is $(\frac{k-1}{2})\frac{2}{k(k-1)} = \frac{1}{k}$ if $d = 2$, and 0 otherwise. Thus the contribution of $w(e)$ to the average value of Δ_π is $\frac{3}{2k}$ (for $d = 2$) or $\frac{1}{k}$ (for $d \geq 3$) times the contribution to $\sum_{i=1}^{k} w_{\text{ex}}(V_i^*)$.

Therefore, the average value of f_π is at most

$$(1-\frac{3}{2k})\sum_{i=1}^{k} w_{\mathrm{ex}}(V_i^*) \ \le\ (2-\frac{3}{k})f(\mathcal{P}^*) \quad \text{if } d^+_{\max}=2;$$

or

$$(1-\frac{1}{k})\sum_{i=1}^{k} w_{\mathrm{ex}}(V_i^*) \ \le\ (2-\frac{2}{k})f(\mathcal{P}^*) \quad \text{if } d^+_{\max}\ge 3.$$

(Note that we only need to consider hyperedges of positive weight.) Hence the theorem for odd k has been proved.

We can apply a similar argument for an even k. By Lemma 3.2, for any permutation π, GSA2 finds a k-partition whose weight is at most

$$\begin{aligned} f_\pi &= f(V^*_{\pi(1)}) + f(V - V^*_{\pi(1)}) \\ &\quad + \sum_{i=1}^{\frac{k}{2}-1}\Big(f(V^*_{\pi(2i)}) + f(V^*_{\pi(2i+1)}) + f(V - V^*_{\pi(2i)} - V^*_{\pi(2i+1)})\Big) - (\frac{k}{2}-1)f(V) \\ &= \sum_{i=1}^{k} w_{\mathrm{ex}}(V_i^*) - w_{\mathrm{ex}}(V^*_{\pi(k)}) - \sum_{i=1}^{\frac{k}{2}-1} w\left(E(V^*_{\pi(2i)} : V^*_{\pi(2i+1)})\right). \end{aligned} \tag{3.33}$$

Let

$$\Delta_\pi \ =\ \sum_{i=1}^{k} w_{\mathrm{ex}}(V_i^*) - f_\pi \ =\ w_{\mathrm{ex}}(V^*_{\pi(k)}) + \sum_{i=1}^{\frac{k}{2}-1} w\left(E(V^*_{\pi(2i)} : V^*_{\pi(2i+1)})\right).$$

Let e be an hyperedge of degree $d \ge 2$ after contraction. The average number that $w(e)$ is counted due to the term of $w_{\mathrm{ex}}(V^*_{\pi(k)})$ is $\frac{d}{k}$. The average number due to other terms is $(\frac{k}{2}-1)\frac{2}{k(k-1)} = \frac{k-2}{k(k-1)}$ if $d=2$, and 0 otherwise. Thus the contribution of $w(e)$ to the average value of Δ_π is $\frac{1}{k} + \frac{k-2}{k(k-1)} = \frac{3k-4}{2k(k-1)}$ (for $d=2$) or $\frac{1}{k}$ (for $d \ge 3$) times the contribution to $\sum_{i=1}^{k} w_{\mathrm{ex}}(V_i^*)$.

Therefore, the average value of f_π is at most

$$\left(1-\frac{3k-4}{2k(k-1)}\right)\sum_{i=1}^{k} w_{\mathrm{ex}}(V_i^*) \ \le\ \left(2-\frac{3k-4}{k(k-1)}\right)f(\mathcal{P}^*) \quad \text{if } d^+_{\max}=2;$$

or

$$\left(1-\frac{1}{k}\right)\sum_{i=1}^{k} w_{\mathrm{ex}}(V_i^*) \ \le\ \left(2-\frac{2}{k}\right)f(\mathcal{P}^*) \quad \text{if } d^+_{\max}\ge 3.$$

Hence the theorem for even k has been proved. □

3.4 Tight examples

Let us see that the performance guarantees of GSA2 obtained so far are tight.

Proposition 3.3. *For any $k \geq 4$ and $\epsilon > 0$, there exists a nonnegative submodular system for which GSA2 always finds a k-partition of weight at least $\lfloor \frac{k}{2} \rfloor - \epsilon$ times of the optimum.*

Proof. First consider an odd $k \geq 5$. Define two systems (V, f) and (W, g_o) as follows.

Let $V = \{v_0, v_1, \ldots, v_{k-1}\}$. Define f by $f(S) \triangleq |S|$ if $S \subset V$ and $f(V) = 0$. Let $W = \{v_0, w_1, \ldots, w_{k-1}\}$, where $V \cap W = \{v_0\}$ holds. Let g_o be the cut function of graph (W, E_o) for $E_o = \{\{v_0, w_1\}, \{v_0, w_2\}, \ldots, \{v_0, w_{k-1}\}\}$ of edges with weight $\frac{k-\epsilon}{4}$. Obviously, both f and g_o are submodular. Let us consider submodular system $(V \cup W, h_o)$, where $h_o(S) = f(S \cap V) + g_o(S \cap W)$, $S \subseteq V \cup W$. See Figure 3.4.

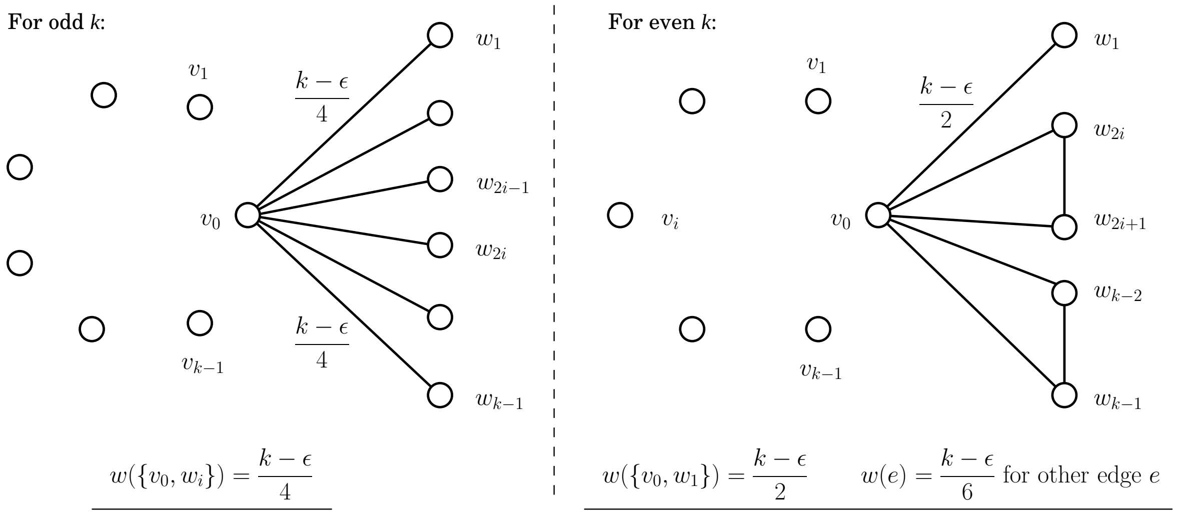


Figure 3.4: Tight example for GSA2 applied to k-PPSS.

There is a k-partition $\mathcal{P} = \{\{v_1\}, \{v_2\}, \ldots, \{v_{k-1}\}, W\}$ whose weight is k. On the other hand, since any 3-partition separating some members in V has weight at least k, the minimum 3-partition is $\{\{w_i\}, \{w_j\}, (V \cup W) - \{w_i, w_j\}\}$ for some $i \neq j$, whose weight is $k - \epsilon$. It is then easy to see that, GSA2 finally outputs k-partition $\mathcal{P}' = \{\{w_1\}, \{w_2\}, \ldots, \{w_{k-1}\}, V\}$, whose weight is $\frac{k-1}{2}(k - \epsilon)$. Thus the performance guarantee is at least as bad as

$$\frac{(k-1)(k-\epsilon)}{2k} = \frac{k-1}{2} - \frac{(k-1)\epsilon}{2k} > \lfloor \frac{k}{2} \rfloor - \epsilon.$$

Let $k \geq 4$ be an even number. Consider the cut function g_e of graph (W, E_e) for $E_o = \{\{v_0, w_1\}, \ldots, \{v_0, w_{k-1}\}\} \cup \{\{w_2, w_3\}, \ldots, \{w_{k-2}, w_{k-1}\}\}$, where edge $\{v_0, w_1\}$ has weight $\frac{k-\epsilon}{2}$, and other edges have weights $\frac{k-\epsilon}{6}$. Construct a submodular system $(V \cup W, h_e)$, where $h_e(S) = f(S \cap V) + g_o(S \cap W)$, $S \subseteq V \cup W$. This is also shown in Figure 3.4.

The k-partition $\mathcal{P} = \{\{v_1\}, \{v_2\}, \ldots, \{v_{k-1}\}, W\}$ has weight k. However, GSA2 outputs a k-partition $\mathcal{P}' = \{\{w_1\}, \{w_2\}, \ldots, \{w_{k-1}\}, V\}$ with weight $\frac{k}{2}(k-\epsilon)$. Thus the performance guarantee is at least as bad as

$$\frac{k(k-\epsilon)}{2k} = \frac{k}{2} - \frac{\epsilon}{2} > \lfloor \frac{k}{2} \rfloor - \epsilon.$$

□

Proposition 3.4. *For any $k \geq 4$ and $\epsilon > 0$, there exists a nonnegative symmetric submodular system for which GSA2 always finds a k-partition of weight at least $2 - \frac{2}{k} - \epsilon$ times of the optimum.*

Proof. We define two submodular systems (V, f) and (W, g).

Let $V = \{v_0, v_1, \ldots, v_{k-1}\}$, and $f(S) \triangleq \min\{|S|, |V - S|\}$ for all $S \subseteq V$. It is not hard to see that f is symmetric and submodular. Let $W = \{v_0, w_1, \ldots, w_{k-1}\}$, where $V \cap W = \{v_0\}$ holds. Let g be the cut function of graph (W, E), where $E = \{\{v_0, w_1\}, \{v_0, w_2\}, \ldots, \{v_0, w_{k-1}\}\}$ consists edges of weights $1 - \frac{\epsilon}{4}$. Function g is also symmetric and submodular. Let us consider a system $(V \cup W, h)$, where $h(S) = f(S \cap V) + g(S \cap W)$, $S \subseteq V \cup W$. See Figure 3.5. By Proposition 2.1, function h is symmetric and submodular (and also nonnegative).

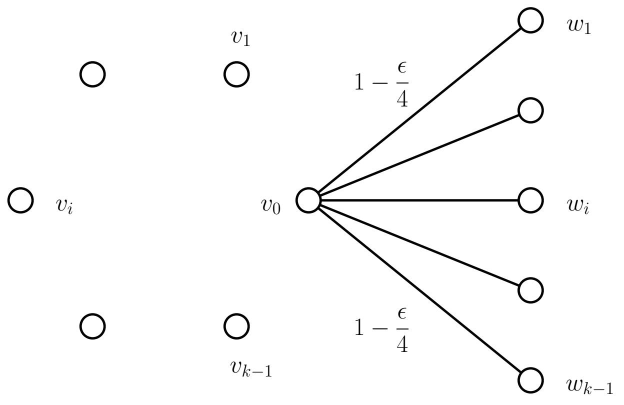


Figure 3.5: Tight example for GSA2 applied to k-PPSSS.

There is a k-partition $\mathcal{P} = \{\{v_1\}, \{v_2\}, \ldots, \{v_{k-1}\}, W\}$, whose weight is k. However, it is easy to see that GSA2 outputs a k-partition $\mathcal{P}' = \{\{w_1\}, \{w_2\}, \ldots, \{w_{k-1}\}, V\}$ of weight $(k-1)(2 - \frac{\epsilon}{2})$. Thus the performance guarantee is at least as bad as

$$\frac{(k-1)(2-\frac{\epsilon}{2})}{k} = 2 - \frac{2}{k} - \frac{(k-1)\epsilon}{2k} > 2 - \frac{2}{k} - \epsilon.$$

□

Proposition 3.5. *For any $\epsilon > 0$, there exists a graph for which GSA2 always finds a k-partition (or k-way cut) of weight at least $2 - \frac{3}{k} - \epsilon$ times of the optimum for odd $k \geq 5$, and $2 - \frac{3}{k} + \frac{1}{k^2-k} - \epsilon$ for even $k \geq 4$.*

Proof. First consider an odd $k \geq 5$.

Let $V = \{v_0, v_1, \ldots, v_{k-1}\}$ and $W = \{v_0, w_1, \ldots, w_{k-1}\}$. Consider a graph $(V \cup W, E_o)$ for $E_o = \{\{v_i, v_j\} \mid i \neq j\} \cup \{\{v_0, w_1\}, \{v_0, w_2\}, \ldots, \{v_0, w_{k-1}\}\}$. Each edge $\{v_i, v_j\}$ has weight 1, whereas each edge $\{v_0, w_i\}$ has weight $\frac{2k-3-\epsilon}{2}$. See Figure 3.6.

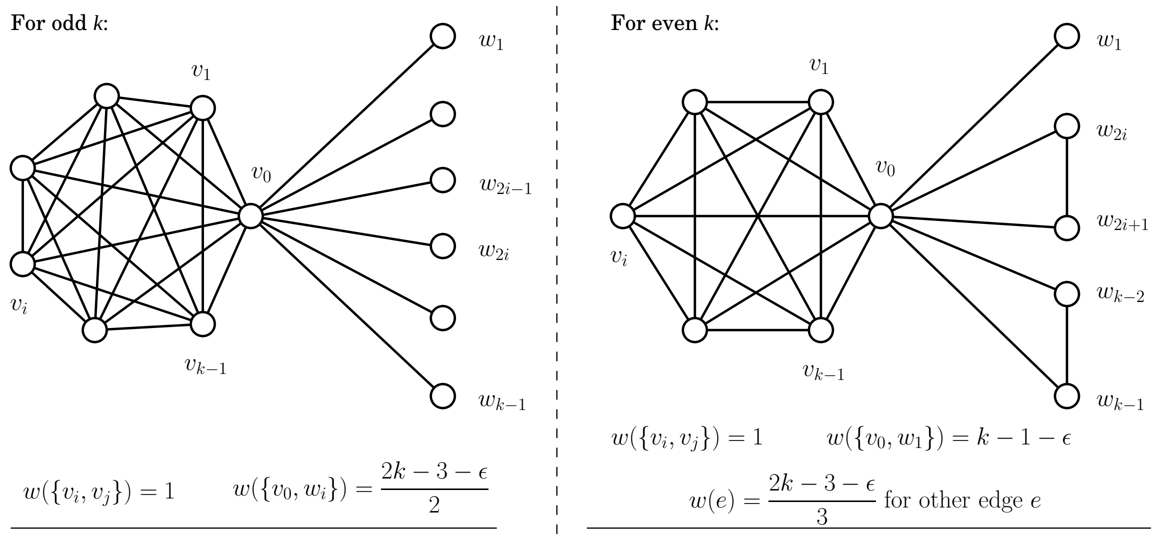


Figure 3.6: Tight example for GSA2 applied to k-PPG.

There is a k-partition $\mathcal{P} = \{\{v_1\}, \{v_2\}, \ldots, \{v_{k-1}\}, W\}$, whose weight is $k(k-1)$. On the other hand, since any 3-partition separating some members in V has weight at least $4k-6$, the minimum 3-partition is $\{\{w_i\}, \{w_j\}, (V \cup W) - \{w_i, w_j\}\}$ for some $i \neq j$, whose weight is $4k - 6 - 2\epsilon$. It is then easy to see that, GSA2 outputs k-partition $\mathcal{P}' = \{\{w_1\}, \{w_2\}, \ldots, \{w_{k-1}\}, V\}$, whose weight is $(k-1)(2k-3-\epsilon)$. Thus the performance guarantee is at least as bad as

$$\frac{(k-1)(2k-3-\epsilon)}{k(k-1)} = 2 - \frac{3}{k} - \frac{\epsilon}{k} > 2 - \frac{3}{k} - \epsilon.$$

Let $k \geq 4$ be an even number. Consider a graph $(V \cup W, E_e)$, where $E_e = \{\{v_i, v_j\} \mid i \neq j\} \cup \{\{v_0, w_1\}, \{v_0, w_2\}, \ldots, \{v_0, w_{k-1}\}\} \cup \{\{w_2, w_3\}, \{w_4, w_5\}, \ldots, \{w_{k-2}, w_{k-1}\}\}$. Each edge $\{v_i, v_j\}$ has weight 1, whereas edge $\{v_0, w_1\}$ has weight $k - 1 - \epsilon$. Other edges have weight $\frac{2k-3-\epsilon}{3}$. See also Figure 3.6.

The optimal k-partition is $\mathcal{P} = \{\{v_1\}, \{v_2\}, \ldots, \{v_{k-1}\}, W\}$, whose weight is $k(k-1)$. However, it is easy to see that GSA2 outputs k-partition $\mathcal{P}' = \{\{w_1\}, \{w_2\}, \ldots, \{w_{k-1}\}, V\}$, whose weight is $2(k-1-\epsilon)+\frac{k-2}{2}(4k-6-2\epsilon)$. Thus the performance guarantee is at least as bad as

$$\frac{2(k-1-\epsilon)+(k-2)(2k-3-\epsilon)}{k(k-1)} = 2-\frac{3}{k}+\frac{1}{k^2-k}-\frac{\epsilon}{k-1} > 2-\frac{3}{k}+\frac{1}{k^2-k}-\epsilon.$$

□

3.5 Modified GSA2 (M-GSA2) for general MPP and remarks

3.5.1 Modified GSA2 (M-GSA2) for general MPP

Since problem 3-MCP (multiterminal cut problem) is NP-hard, we cannot expect a polynomial time algorithm to compute a minimum 3-target-split in general (unless P=NP holds). Nevertheless, we note that Lemmas 3.1, 3.2 can be extended to MPP in a straightforward manner, hence the performance guarantees obtained for MPP-NTs hold for their target split versions.

3.5.2 Remarks

Remark 1. Possible improvements of performance guarantee.

It is possible to obtain better performance guarantees in some cases. Let us take GSA for k-PPSS as an example. Given a nonnegative submodular system (V,f), by Lemma 2.1, the weight of the solution $\mathcal{P}_k$ obtained by GSA is at most

$$\sum_{i=1}^{k-1}(f(V_i)+f(V-V_i))-(k-2)f(V)$$

for any k-partition $\{V_1,V_2,\ldots,V_k\}$ of V. Thus the performance guarantee is bounded by

$$(1+\alpha)(1-\frac{1}{k})-\frac{(k-2)f(V)}{opt}, \tag{3.34}$$

where opt is the optimum (the trivial case of $opt=0$ is simply omitted), and α is an arbitrary number satisfying $\sum_{i=1}^{k} f(V-V_i)\le\alpha\sum_{i=1}^{k} f(V_i)$ for all k-partitions $\{V_1,\ldots,V_k\}$ of V. This is meaningful if $f(V)>0$ and opt is not too large; e.g., it gives the correct guarantee 1, instead of $2-\frac{2}{k}$, for a constant function f. We note that it may be hard to find the value of opt in general. A good approximate value is then valuable.

In particular, since

$$\sum_{i=1}^{k} f(V-V_i) \le (k-1)\sum_{i=1}^{k} f(V_i)-k(k-2)f(\emptyset)$$

holds for any k-partition by Lemma 2.2, we see that α can be chosen to be

$$k-1-\frac{k(k-2)f(\emptyset)}{opt}.$$

Hence the performance guarantee is at least as good as

$$\begin{aligned}&\left(k-\frac{k(k-2)f(\emptyset)}{opt}\right)\left(1-\frac{1}{k}\right)-\frac{(k-2)f(V)}{opt}\\ =\ &(k-1)\left(1-\frac{(k-2)f(\emptyset)}{opt}\right)-\frac{(k-2)f(V)}{opt}.\end{aligned}$$

Remark 2. On the timing of using the minimum 2-partition computation.

Recall that we start with a minimum 2-partition in GSA2 for an *even* k (Figure 3.3). This ensures that a k-partition is eventually obtained, since later we increase the size of partition by two in each phase. From the viewpoint of the size of partition, 2-partition can be used in any (single) phase, not necessarily the first. However, it *is critical* in our proof to start with a minimum 2-partition. Otherwise inequality (3.15) in Lemma 3.2 may not hold. Let us illustrate by an instance of 4-PPG in the next graph.

Consider a graph $G = (V, E)$, where $V = \{a, b, c, d, e\}$, and $E = \{\{a, b\}, \{b, c\}, \{c, d\}, \{d, e\}, \{e, c\}\}$. Edge $\{b, c\}$ has weight 1.5 and others have weight 1. Let w denote the edge weight function. This graph is shown in Figure. 3.7.

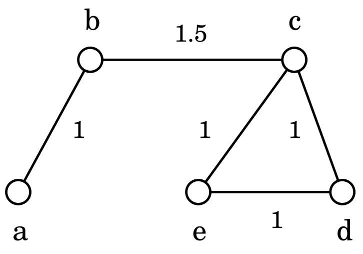


Figure 3.7: Lemma 3.2 fails if we compute a minimum 3-partition at first.

If we compute a minimum 3-partition at first, we may get a solution of 4-partition $\mathcal{P} = \{\{a\}, \{b\}, \{c\}, \{d, e\}\}$, whose weight is 9. However, considering a 4-partition $\{\{a\}, \{d\}, \{e\}, \{b, c\}\}$ (the optimal solution), we have

$$(w_{\text{ex}}(\{a\})+w_{\text{ex}}(V-\{a\}))+(w_{\text{ex}}(\{d\})+w_{\text{ex}}(\{e\})+w_{\text{ex}}(V-\{d\}-\{e\}))=2+6=8,$$

which is *smaller* than the weight of $\mathcal{P}$! This violates the claim in Lemma 3.2.

Actually, the algorithm in [Kap 96] constructs a k-partition in this way. Thus Lemma 4.3 in [Kap 96], which in this case is equivalent to (3.15), is not valid. We further discuss this in the next remark.

Remark 3. Extensions of GSA and GSA2.

A natural question asks whether GSA and GSA2 can be further extended. That is, how does it perform when the size of partition increases three per phase, ... , j per phase $(2 < j < k)$. One may wonder that, naturally, a larger j would take longer running time but should give better performance guarantee. Here we note that, unfortunately, the approach taken in this thesis does not help to answer this question.

In fact, such greedy algorithm was first considered by Kapoor for k-PPG. He claimed (Lemma 4.3 [Kap 96]) that, given a graph with a nonnegative edge weight function w, for any $j \geq 2$ and any k-partition $\mathcal{P} = \{V_1, V_2, \ldots, V_k\}$, by increasing the size of partition j per phase (except for the last phase), the weight of the obtained k-partition is bounded by

$$\sum_{i=1}^{\lfloor\frac{k-1}{j}\rfloor} \left(w_{\text{ex}}(V_{j(i-1)+1}) + \cdots + w_{\text{ex}}(V_{(j(i-1)+j}) + w_{\text{ex}}\left(\bigcup_{\ell=1}^{j} V_{j(i-1)+\ell}\right)\right)$$
$$+ \left(\sum_{i=j\lfloor\frac{k-1}{j}\rfloor+1}^{k-1} w_{\text{ex}}(V_i) + w_{\text{ex}}\left(\bigcup_{i=j\lfloor\frac{k-1}{j}\rfloor+1}^{k-1} V_i\right)\right). \tag{3.35}$$

Notice that this extends Lemmas 2.1, 3.1 and 3.2. However, we here provide a counter example which shows that such an extension is not valid even for $j = 3$ (for $j = 2$, see the previous remark).

Let $k = 7$. Consider a graph (V, E), where $V = \{a, b, c, d, e, f, g, h\}$ and $E = \{\{a, b\}, \{b, c\}, \{c, d\}, \{d, b\}, \{d, e\}, \{e, f\}, \{f, g\}, \{g, e\}, \{g, h\}\}$. Edge $\{d, e\}$ has weight 3 and others have 2. Let w denote the edge weight function. This graph is shown in Figure. 3.8.

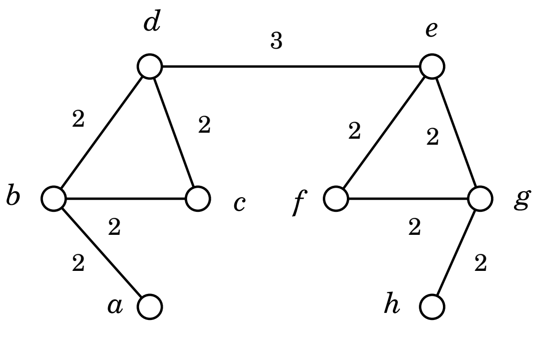


Figure 3.8: An example for which main lemmas cannot be extended.

Similarly to GSA/GSA2, we first get the minimum 4-partition $\{\{a\}, \{b, c, d\}, \{e, f, g\}, \{h\}\}$, and then get a 7-partition $\mathcal{P} = \{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f, g\}, \{h\}\}$ of weight 34. According to (3.35), one may expect that

$$\begin{aligned} w_{\text{ex}}(\mathcal{P}) \leq & \ (w(V_1) + w(V_2) + w(V_3) + w(V_1 \cup V_2 \cup V_3)) \\ & + (w(V_4) + w(V_5) + w(V_6) + w(V_4 \cup V_5 \cup V_6)) \end{aligned} \tag{3.36}$$

holds for any 7-way partition $\mathcal{P}' = \{V_1, V_2, \ldots, V_7\}$. However, taking $\mathcal{P}' = \{\{a\}, \{b\}, \{c\}, \{f\}, \{g\}, \{h\}, \{d, e\}\}$, inequality (3.36) yields $34 \leq 16 + 16 = 32$, a contradiction.

Remark 4. Performance guarantees for k-PPG and k-PPSSS.

In the previous remarks, we have seen that extensions of the main lemmas (Lemmas 2.1, 3.1 and 3.2) may be difficult to obtain even for k-PPG. Notice that, however, bad examples are still unknown for showing that improved guarantees *cannot* be obtained by greedily increasing the size of partition $j \geq 3$ per phase. Here we note that, however, for k-PPSSS (MPP in nonnegative symmetric submodular system), bad examples exist for showing that no guarantees better than $2 - \frac{2}{k}$ can be obtained by greedy splitting.

Let GSA^j denote such greedy algorithm that increases the size of partition j per phase (GSA^1 is just GSA, and GSA^2 is GSA2). We claim that, GSA^j has the same performance guarantee as GSA when they are applied to k-PPSSS. An example is shown in Figure 3.5, by choosing $k = 2j + 1$ (or larger) for each $j \geq 3$.

Chapter 4

A Primal-Dual Approximation Algorithm for SNDP

4.1 Introduction

4.1.1 Network design problem with requirement function (NDRF)

In the following two chapters, we consider the survivable network design problem (SNDP). Recall that the general (bipartite-graph based) formulation of SNDP is the following.

Problem 4.1 (SNDP). *Given a bipartite graph $G = (V, U, E)$ with a set V of terminals, a set U of nonterminals, a set E of edges between V and U, a weight function $w : U \to \mathbf{R}^+$ and connectivity requirements r_{st} for each pair of distinct terminals $s, t \in V$, find a minimum weight subset $U^* \subseteq U$ such that there are at least r_{st} nonterminal disjoint paths between each pair of distinct terminals s and t in the subgraph $G^* = G[V \cup U^*]$ of G.*

(There is a hypergraph-based formulation as stated in Chapter 1. But we only use the above formulation in this chapter.)

Let $d_{\max}$ and $d^+_{\max}$ denote the maximum degree of nonterminals and of nonterminals with positive weights, respectively. Without loss of generality we suppose $d^+_{\max} \geq 2$. We call SNDP with $r_{st} \in \{0, 1\}$ *0-1 SNDP*. Problem *SNDP-G* (SNDP in graph) is SNDP with $d_{\max} = 2$ (this is graph version of the hypergraph-based formulation of SNDP). SNDP-G with $r_{st} \in \{0, 1\}$ is referred as *0-1 SNDP-G*. A problem class between SNDP and SNDP-G is SNDP with $d^+_{\max} = 2$, which we refer as SNDP-G$^+$.

We first give a general formulation of network design problems by employing a requirement function. Let $\mathbf{Z}^+$ denote the set of nonnegative integers. We consider the next *network design problem with requirement function* (NDRF).

Problem 4.2 (NDRF). *Given a bipartite graph $G = (V, U, E)$ with a weight function $w : U \to \mathbf{R}^+$ and a requirement function $r : 2^V \to \mathbf{Z}^+$, NDRF is defined as follows.*

$$
\begin{aligned}
\text{(IP)} \quad & \text{minimize} \quad && \sum_{u \in U} w(u) x_u && && (4.1)\\
& \text{subject to} \quad && x(\Delta(S)) \geq r(S) && S \subseteq V, && (4.2)\\
& && x_u \in \{0, 1\} && u \in U, && (4.3)
\end{aligned}
$$

where $\Delta(S) \triangleq \Gamma(S) \cap \Gamma(V - S) \subseteq U$ and $x(U') \triangleq \sum_{u \in U'} x_u$, $U' \subseteq U$.

Without loss of generality we assume $r(\emptyset) = r(V) = 0$ and $r_{\max} \triangleq \max_S r(S) \leq |U|$, since otherwise there is no feasible solution. We note that NDRF can be described in the words of hypergraph, which will be considered in the next chapter.

4.1.2 A survey on SNDP

We first give a brief survey on problems SNDP and NDRF. We do not review special cases such as SPP (shortest path), MST(minimum spanning tree), STP(Steiner tree) and minimum k edge-connected spanning subgraph. There are numerous literatures on them; see [AMO 93, DSR 00, GP 88, HRW 92, Khu 97, KR 93].

The study on SNDP was started by Agrawal, Klein and Ravi [AKR 95]. Considering 0-1 SNDP-G, they gave a sophisticated approximation algorithm based on primal-dual method. It is shown that 0-1 SNDP-G can be approximated within a factor of 2 by their algorithm. An independent work was also done by Goemans and Williamson [GW 92].

Later Williamson, Goemans, Mihail and Vazirani [WGMV 93] extended the primal-dual approximation algorithm to general SNDP-G. They formalize a basic mechanism of using primal-dual method. It picks edge sets in $r_{\max}$ phases. In each phase it tries to augment the size of cuts with deficiency at the least cost by solving an integer program. The integer program is in general NP-hard, but can be approximated within factor 2 by a primal-dual method based approach (which extends the algorithms in [AKR 95, GW 92]). They showed that their algorithm has a performance guarantee of $2r_{\max}$, which is the first nontrivial result for the general SNDP-G. An efficient implementation can be found in [GGW 93].

After that, Goemans, Goldberg, Plotkin, Shmoys, Tardos and Williamson [GGPSTW 94] showed that, by augmenting the size of only those cuts with the maximum deficiency in the algorithm of [WGMV 93], a $2\mathcal{H}(r_{\max})$-approximation algorithm can be obtained, where $\mathcal{H}(k) = 1 + \frac{1}{2} + \cdots + \frac{1}{k}$ is the kth harmonic number. It is reported that the primal-dual approximation algorithm works well in practice, see [Wil 95, MSDM 96]. For a detail overview and other applications of the primal-dual algorithms, we refer the readers to the well-written surveys [GW 97, Wil 01].

Jain [Jain 98] showed that there is an edge e with $x_e^* \geq \frac{1}{2}$ in any *basic* solution x^* of the LP relaxation of SNDP-G, where the constraint $x_e \in \{0,1\}$ is relaxed to $0 \leq x_e \leq 1$ for every edge e. He then showed that an *iterative rounding* process yields a 2-approximation algorithm. We note that, although Jain's algorithm has a much stronger theoretical guarantee than the above primal-dual algorithm, it is of less practical value since it requires to solve the LP relaxation of SNDP-G, which is a nontrivial computation task.

Jain, Măndoiu, Vazirani and Williamson [JMVW 99] considered the *element connectivity problem* (ECP). In that problem, there are two types of vertices: terminals and nonterminals, where nonterminals and edges are called the *elements*. Only edges have (nonnegative) weights, and there are connectivity requirements r_{st} for each pair of terminals s and t, which specifies the least number of *element-disjoint* paths to be realized. The objective is to find a minimum weight (spanning) subgraph satisfying the connectivity requirements. Obviously, SNDP-G is a special case of ECP with empty nonterminal set. Following the basic algorithmic schema established in [GGPSTW 94, WGMV 93], Jain *et al.* proposed a primal-dual approximation algorithm for ECP. Verifying that their algorithm satisfies three conditions proposed in [RW 95], they claimed that it is a $2\mathcal{H}(r_{\max})$-approximation algorithm.

We note that, Ravi and Williamson [RW 95] developed a primal-dual method based approximation algorithm for the minimum k-vertex connected spanning subgraph problem. In considering the performance, they claimed that, if three conditions are satisfied, then a guarantee of $2\mathcal{H}(r_{\max})$ can be obtained. This is what Jain *et al.* algorithm based. Unfortunately, due to a bug in [RW 95], even if the three conditions are satisfied, it is still unclear whether the desired guarantee can be obtained or not. See [RW 01] for details. In this section, we will give a complete proof for Jain *et al.* result. Recently, Fleischer, Jain and Williamson [FJW 01] have extended the iterative rounding algorithm in [Jain 98] to ECP and have obtained a 2-approximation algorithm.

Let us consider the general SNDP problem. We first show that ECP is a special case of SNDP, i.e., SNDP-G$^+$. This can be easily seen be replacing each edge by a degree 2 new nonterminal of the same weight, then inserting a new terminal between each pair of two adjacent nonterminals. The obtained graph is thus bipartite. Moreover, positive weights are only given to nonterminals of degree 2. Let $r_{st} = 0$ if at least one of s and t is a new terminal. We thus have reduced ECP to SNDP-G$^+$ in linear time in this way.

Proposition 4.1. *ECP = SNDP-G$^+$.* □

Clearly, SNDP is also NP-hard even if $r_{st} = 1$ holds for any distinct $s, t \in T$ for a given $T \subseteq V$. This problem is called the *node weighted Steiner tree problem.* Using a greedy heuristic, Klein and Ravi [KR 93] have given a $(2\ln|T|)$-approximation algorithm. The guarantee is further improved to $1.35\ln|T|$ (for arbitrary weights) and $\ln|T|$ (for unit weights — also NP-hard) by Guha and Khuller [GK 99]. Takeshita, Fujito and Watanabe [TFW 99] extended the primal-dual approximation algorithm in [GW 92] to 0-1 SNDP (which includes the node weighted Steiner tree problem). They showed a $d_{\max}$-approximation algorithm. It is of further interest to study on the practical performance of algorithms in [KR 93, GK 99] and [TFW 99].

4.1.3 Preliminary

As noted before, we first consider NDRF (Problem 4.2) as formulated by (IP) with a requirement function r. Actually we do not know how to solve it efficiently for arbitrary requirement functions r. Nevertheless, we will see that, under two conditions, it can be approximated in polynomial time with reasonable performance guarantee. The first condition is as follows, whereas the second condition will be stated in Section 4.2.

Condition 1. *Function r is weakly supermodular; i.e., for any $X, Y \subseteq V$, it holds that*

$$r(X) + r(Y) \ \leq \ \max\{r(X \cap Y) + r(X \cup Y),\ r(X - Y) + r(Y - X)\}. \tag{4.4}$$

Let $G = (V, U, E)$ be a bipartite graph. Let $\Delta_A(S) \triangleq \Delta(S) \cap A$ for $S \subseteq V$ and $A \subseteq U$. Notice that we have shown a correspondence between Δ in bipartite graph and δ in the corresponding hypergraph H (see Section 1.3 in Chapter 1). Hence $|\Delta_A| : 2^V \to \mathbf{Z}^+$ is also symmetric and submodular by Proposition 1.4. Therefore, for any $A \subseteq W$ and $X, Y \subseteq V$, we have

$$|\Delta_A(X)| + |\Delta_A(Y)| \ \geq \ |\Delta_A(X \cap Y)| + |\Delta_A(X \cup Y)|, \tag{4.5}$$

$$|\Delta_A(X)| + |\Delta_A(Y)| \ \geq \ |\Delta_A(X - Y)| + |\Delta_A(Y - X)|. \tag{4.6}$$

4.2 Primal-dual approximation algorithm for NDRF

In this section, we present an approximation algorithm for NDRF (IP). It is based on the primal-dual algorithm established in [GGPSTW 94, WGMV 93] for SNDP-G. The proof of the performance guarantee will be given in the next section.

Definition 4.1 (deficiency). *The deficiency of a terminal subset $S \subseteq V$ with respect to a nonterminal subset $A \subseteq U$ is $r(S) - |\Delta_A(S)| \ = \ r(S) - |\Delta(S) \cap A|$.*

Hence a nonterminal subset $A \subseteq U$ is feasible to (IP) if and only if the maximum deficiency (with respect to A) over all $S \subseteq V$ is non-positive, i.e., $|\Delta_A(S)| = |\Delta(S) \cap A| \geq r(S)$.

Analogously to [GGPSTW 94, WGMV 93], our algorithm consists of $r_{\max}$ phases. It starts with $U_0 = \emptyset$. Let $U_i \subseteq U$ be the set of nonterminals picked so far before phase i. At the beginning of the i^{th} phase, the maximum deficiency (with respect to U_{i-1}) is $r_{\max} - i + 1$. We decrease it by one by adding some $A_i \subseteq U - U_{i-1}$. Setting $U_i = U_{i-1} \cup A_i$, we proceed to the next phase until $i = r_{\max}$ holds. Finally, we get the output $U_{r_{\max}}$. It must be feasible to (IP), since the algorithm ensures that the maximum deficiency with respect to $U_{r_{\max}}$ is zero.

In order to get a solution with small weight, we consider the next integer program $(\mathrm{IP})_i$ to find such a set A_i in each phase i.

$$\begin{array}{lllr} (\mathrm{IP})_i & \text{minimize} & \displaystyle\sum_{u\in U-U_{i-1}} w(u)x_u & \\ & \text{subject to} & x(\Delta_{U-U_{i-1}}(S)) \geq h_i(S) & S \subseteq V, \\ & & x_u \in \{0,1\} & u \in U - U_{i-1}, \end{array}$$

where function $h_i : 2^V \to \{0,1\}$ is defined by

$$h_i(S) = \begin{cases} 1 & \text{if } r(S) - |\Delta_{U_{i-1}}(S)| = r_{\max} - i + 1, \\ 0 & \text{otherwise (i.e., } r(S) - |\Delta_{U_{i-1}}(S)| \leq r_{\max} - i). \end{cases} \tag{4.7}$$

Notice that (IP) has feasible solution if and only if $(\mathrm{IP})_i$ has feasible solution for all $i = 1,\ldots,r_{\max}$. In general, $(\mathrm{IP})_i$ is still NP-hard. Nevertheless, later we will see that it can be approximated by a primal-dual approximation algorithm given in Figure 4.2. We first give the resulting algorithm for (IP), called ALGO-IP, in Figure 4.1.

```
1  $U_0 \leftarrow \emptyset$
2  for $i = 1$ to $r_{\max}$
3      $A_i \leftarrow$ the output of the algorithm in Figure 4.2 for $(\mathrm{IP})_i$
4      $U_i \leftarrow U_{i-1} \cup A_i$
5  end /* for */
6  Output $U_{r_{\max}}$.
```

Figure 4.1: ALGO-IP approximation algorithm for (IP).

Given an i, we now consider to approximate $(\mathrm{IP})_i$. We need more notations.

Definition 4.2 (violated set). *A terminal subset $S \subseteq V$ is violated with respect to a non-terminal subset $A \subseteq U-U_{i-1}$ if $h_i(S) = 1$ and $\Delta_A(S) = \emptyset$ hold.*

Obviously, a set $A \subseteq U-U_{i-1}$ is feasible to $(\mathrm{IP})_i$ if and only if there is no violated set. Under the assumption of Condition 1, violated sets enjoy the following property.

Lemma 4.1. *Let $X, Y \subseteq V$ be two violated sets with respect to $A \subseteq U-U_{i-1}$. Then either sets $X \cap Y$ and $X \cup Y$, or sets $X - Y$ and $Y - X$, are violated with respect to A.*

Proof. By definitions of violated set and function h_i, we see that $r(X) - |\Delta_{U_{i-1}}(X)| = r(Y) - |\Delta_{U_{i-1}}(Y)| = r_{\max} - i + 1$ and $\Delta_A(X) = \Delta_A(Y) = \emptyset$ hold.

By (4.5) and (4.6), we have

$$\begin{aligned}|\Delta_A(X\cap Y)|+|\Delta_A(X\cup Y)| &\le |\Delta_A(X)|+|\Delta_A(Y)|=0,\\ |\Delta_A(X-Y)|+|\Delta_A(Y-X)| &\le |\Delta_A(X)|+|\Delta_A(Y)|=0.\end{aligned}$$

Hence,

$$\Delta_A(X\cap Y) \;=\; \Delta_A(X\cup Y) \;=\; \Delta_A(X-Y) \;=\; \Delta_A(Y-X) \;=\; \emptyset.$$

On the other hand, since function r is weakly supermodular (Condition 1), we see that function $r-|\Delta_{U_{i-1}}|$ is also weakly supermodular by Proposition 1.1 (recall that function $|\Delta_{U_{i-1}}|$ is submodular). Therefore, we have

$$\begin{aligned}2(r_{\max}-i+1) &= r(X)-|\Delta_{U_{i-1}}(X)|+r(Y)-|\Delta_{U_{i-1}}(Y)|\\ &\le \max\Big\{r(X\cap Y)-|\Delta_{U_{i-1}}(X\cap Y)|+r(X\cup Y)-|\Delta_{U_{i-1}}(X\cup Y)|,\\ &\qquad r(X-Y)-|\Delta_{U_{i-1}}(X-Y)|+r(Y-X)-|\Delta_{U_{i-1}}(Y-X)|\Big\}.\end{aligned}$$

However, $r(S)-|\Delta_{U_{i-1}}(S)|\le r_{\max}-i+1$ holds for all $S\subseteq V$, since $r_{\max}-i+1$ is the maximum deficiency. Thus the above inequality implies at least one of the next two equalities.

$$\begin{aligned}r(X\cap Y)-|\Delta_{U_{i-1}}(X\cap Y)| &= r(X\cup Y)-|\Delta_{U_{i-1}}(X\cup Y)| \;=\; r_{\max}-i+1,\\ r(X-Y)-|\Delta_{U_{i-1}}(X-Y)| &= r(Y-X)-|\Delta_{U_{i-1}}(Y-X)| \;=\; r_{\max}-i+1.\end{aligned}$$

Hence either $X\cap Y$ and $X\cup Y$, or $X-Y$ and $Y-X$, are violated with respect to A. □

Two sets X and Y are said to *intersect* if $X\cap Y\neq\emptyset$, $X-Y\neq\emptyset$ and $Y-X\neq\emptyset$ hold. An immediate conclusion from Lemma 4.1 is the next corollary, where we omit the words "with respect to A" for simplicity.

Corollary 4.1. *Let X be a minimal violated set. Any violated set Y does not intersect X; i.e., either $X\subseteq Y$ or $X\cap Y=\emptyset$ holds. In particular, if Y is also a minimal violated set, then $X\cap Y=\emptyset$ holds.* □

Let $\mathcal{V}(A)$ denote the family of minimal violated sets with respect to $A\subseteq U-U_{i-1}$. Obviously, A is feasible to $(\mathrm{IP})_i$ if and only if $\mathcal{V}(A)=\emptyset$. We introduce another condition that needs to be satisfied for our algorithm to be polynomial time.

Condition 2. *The family $\mathcal{V}(A)$ of minimum violated sets of any set $A\subseteq U-U_{i-1}$ can be obtained in polynomial time.*

We now describe the algorithm for $(\mathrm{IP})_i$. For this, relax each constraint $x_u\in\{0,1\}$ to $x_u\ge 0$ in $(\mathrm{IP})_i$. The dual of this LP relaxation of $(\mathrm{IP})_i$ is given by

$$\begin{array}{llll}(\mathrm{D})_i & \text{maximize} & \sum_{S\subseteq V} h_i(S)y_S & \\ & \text{subject to} & \displaystyle\sum_{S\subseteq V:u\in\Delta(S)} y_S\le w(u) & u\in U-U_{i-1},\\ & & y_S\ge 0 & S\subseteq V.\end{array}$$

We use y, A, j and $\bar{w}$ to denote the *dual solution*, *primal solution*, *index of iteration* and *reduced weight* (i.e., $\bar{w}(u) \triangleq w(u) - \sum_{S \subseteq V : u \in \Delta(S)} y_S$), respectively. The algorithm maintains a feasible dual solution y and a primal solution A. Note that y is feasible if and only if $\bar{w}(u) \geq 0$ for all $u \in U - U_{i-1}$. We list the algorithm, called PRIMAL-DUAL, in Figure 4.2.

```
1   $y \leftarrow 0, \quad A \leftarrow \emptyset, \quad j \leftarrow 0, \quad \bar{w} \leftarrow w$
2   while $A$ is not feasible
3       $j \leftarrow j + 1$
4       $\mathcal{V}_j \leftarrow$ the family $\mathcal{V}(A)$ of minimal violated sets
5       if exists $S \in \mathcal{V}_j$ such that $\Delta_{U-U_{i-1}-A}(S) = \emptyset$ then
6           halt. $(\mathrm{IP})_i$, hence (IP), has no feasible solution.
7       end /* if */
8       $u_j \leftarrow \mathrm{argmin}\left\{ \frac{\bar{w}(u)}{|\{S \in \mathcal{V}_j \mid u \in \Delta(S)\}|} \;\middle|\; u \in U - U_{i-1} - A \right\}$
9       $\epsilon_j \leftarrow \frac{\bar{w}(u_j)}{|\{S \in \mathcal{V}_j \mid u_j \in \Delta(S)\}|}, \quad y_S \leftarrow y_S + \epsilon_j$ for all $S \in \mathcal{V}_j$
10      $\bar{w}(u) \leftarrow \bar{w}(u) - \epsilon_j |\{S \in \mathcal{V}_j \mid u \in \Delta(S)\}|$ for all $u \in U - U_{i-1} - A$
11      $A \leftarrow A \cup \{u_j\}$
12  end /* while */
13  for $\ell = j$ down to 1
14      if $A - \{u_\ell\}$ is feasible then $A \leftarrow A - \{u_\ell\}$
15  end /* for */
16  Output $A$ (as $A_i$).
```

Figure 4.2: PRIMAL-DUAL: primal-dual approximation algorithm for $(\mathrm{IP})_i$

It starts with a feasible dual solution $y = 0$ and a primal solution $A = \emptyset$. If A is not feasible to $(\mathrm{IP})_i$, then $\mathcal{V}(A)$ is computed. The dual variables for minimal violated sets in $\mathcal{V}(A)$ are then increased *uniformly*, until some dual constraint becomes *tight* (i.e., becomes equality, or equivalently, the corresponding reduced weight is zero). After that, the reduced weights are renewed. One nonterminal of which the dual constraint is tight is included into A. This process is repeated until A becomes feasible, or we can determine the nonexistence of feasible solution. Finally, if a feasible A is obtained, a *reverse delete* step tries to remove "unnecessary" nonterminals from A, in the reverse order as they were included. Note that y is always feasible to D_i, and $\epsilon_j \geq 0$ holds for all j.

Let us consider the running time of algorithm PRIMAL-DUAL. We store only those positive dual variables y_S. Thus Line 1 takes $O(|U|)$ time. Since $|A|$ increases by one after each **while** iteration, there are at most $|U - U_{i-1}| \leq |U|$ **while** iterations. Let θ be the time complexity to compute $\mathcal{V}(A)$. Then Lines 2, 4 and 14 can be done in θ time, since

A is feasible if and only if $\mathcal{V}(A) = \emptyset$. It is not hard to see that Line 6 can be done in $O(|V||U|)$ time, since $|\mathcal{V}(A)| \leq |V|$ by Corollary 4.1, and this dominates other lines. Hence algorithm PRIMAL-DUAL takes $O(|U|(\theta + |V||U|))$ time to compute A_i. Therefore, the time complexity of algorithm ALGO-IP for (IP) is $O(r_{\max}|U|(\theta + |V||U|))$. This is polynomial.

4.3 Proof of performance guarantee

We follow the proof of [GGPSTW 94, WGMV 93] for SNDP-G. Suppose that (IP) has feasible solution. Then A_i and y obtained by PRIMAL-DUAL are feasible to (IP)$_i$ and (D)$_i$, respectively. Given next is the main lemma.

Lemma 4.2. *Let A_i and y be the output and the corresponding dual solution obtained at the end of algorithm PRIMAL-DUAL, respectively. It holds that*

$$\sum_{u \in A_i} w(u) \leq d^+_{\max} \sum_{S \subseteq V} h_i(S) y_S. \qquad \square$$

Before proving the lemma, let us see that it implies the claimed guarantee $d^+_{\max}\mathcal{H}(r_{\max})$.

Theorem 4.1. *Let opt_{IP} be the optimal value of (IP). Let $U_{r_{\max}} = \bigcup_{i=1}^{r_{\max}} A_i$ be the output of algorithm ALGO-IP. It holds that*

$$\sum_{u \in U_{r_{\max}}} w(u) \leq d^+_{\max}\mathcal{H}(r_{\max}) opt_{\mathrm{IP}}. \tag{4.8}$$

Proof. This can be shown analogously to [GGPSTW 94]. We give for completeness.

Relax the constraints $x_u \in \{0,1\}$ in (IP) to $0 \leq x_u \leq 1$ for all $u \in U$. The dual of this LP relaxation of (IP) is given by

$$\begin{array}{lllr} \text{(D)} & \text{maximize} & \sum_{S \subseteq V} r(S) y_S - \sum_{u \in U} z_u & \\ & \text{subject to} & \displaystyle\sum_{S \subseteq V : u \in \Delta(S)} y_S \leq w(u) + z_u & u \in U, \\ & & y_S \geq 0, \quad z_u \geq 0 & S \subseteq V, u \in U. \end{array}$$

Let opt_{D} be the optimal value of (D). By the weak duality theorem of linear programming, $opt_{\mathrm{D}} \leq opt_{\mathrm{IP}}$ holds. Consider the i^{th} phase. Let y be the dual solution of (D)$_i$ as used in Lemma 4.2. Let

$$z_u = \begin{cases} \displaystyle\sum_{S \subseteq V : u \in \Delta(S)} y_S & \text{if } u \in U_{i-1}, \\ 0 & \text{otherwise } (u \in U - U_{i-1}). \end{cases}$$

It is easy to verify that (y, z) is feasible to (D). Thus

$$\begin{aligned} opt_{\mathrm{IP}} &\geq opt_{\mathrm{D}} \geq \sum_{S\subseteq V} r(S)y_S - \sum_{u\in U} z_u \\ &= \sum_{S\subseteq V} r(S)y_S - \sum_{u\in U_{i-1}} \sum_{S:u\in\Delta(S)} y_S \\ &= \sum_{S\subseteq V} r(S)y_S - \sum_{S\subseteq V} |\Delta_{U_{i-1}}(S)|y_S \\ &= \sum_{S\subseteq V} (r(S) - |\Delta_{U_{i-1}}(S)|)y_S = (r_{\max} - i + 1)\sum_{S\subseteq V} h_i(S)y_S. \end{aligned}$$

The last equality follows because $y_S = 0$ holds for all S with $h_i(S) = 0$, whereas $h_i(S) = 1$ holds if and only if $r(S) - |\Delta_{U_{i-1}}(S)| = r_{\max} - i + 1$. Therefore, by Lemma 4.2, we have

$$\begin{aligned} \sum_{u\in A_i} w(u) &\leq d^+_{\max} \sum_{S\subseteq V} h_i(S)y_S \leq \frac{d^+_{\max} opt_{\mathrm{IP}}}{r_{\max} - i + 1}, \\ \Longrightarrow \quad \sum_{u\in U_{r_{\max}}} w(u) &= \sum_{i=1}^{r_{\max}} \sum_{u\in A_i} w(u) \\ &\leq \sum_{i=1}^{r_{\max}} \left(\frac{d^+_{\max} opt_{\mathrm{IP}}}{r_{\max} - i + 1} \right) = d^+_{\max}\mathcal{H}(r_{\max}) opt_{\mathrm{IP}}. \end{aligned}$$

This completes the proof for the theorem. □

We give the proof for Lemma 4.2 in the following, which ends at the end of this section.

Proof. First suppose $w(u) > 0$ for all $u \in U$, i.e., $d^+_{\max}$ is the maximum degree of nonterminals. The case in which $w(u) = 0$ holds for some u will be treated at the end of this proof.

Let L be the number of **while** iterations. Notice that, for any $\ell = 1, 2, \ldots, L$, we have

$$w(u_\ell) = \sum_{S\subseteq V: u_\ell\in\Delta(S)} y_S = \sum_{j=1}^{L} \big|\{S \in \mathcal{V}_j \mid u_\ell \in \Delta(S)\}\big|\epsilon_j.$$

Therefore, it holds that

$$\sum_{u\in A_i} w(u) = \sum_{u\in A_i} \sum_{1\leq j\leq L} \big|\{S \in \mathcal{V}_j \mid u \in \Delta(S)\}\big|\epsilon_j = \sum_{1\leq j\leq L} \sum_{S\in\mathcal{V}_j} |\Delta_{A_i}(S)|\epsilon_j.$$

On the other hand, since $y_S = \sum_{j:S\in\mathcal{V}_j} \epsilon_j$, we have

$$\sum_{S\subseteq V} h_i(S)y_S = \sum_{S\subseteq V} y_S = \sum_{S\subseteq V} \sum_{j:S\in\mathcal{V}_j} \epsilon_j = \sum_{1\leq j\leq L} \sum_{S\in\mathcal{V}_j} \epsilon_j = \sum_{1\leq j\leq L} |\mathcal{V}_j|\epsilon_j.$$

Thus to prove Lemma 4.2, it suffices to show

$$\sum_{S\in\mathcal{V}_j} |\Delta_{A_i}(S)| \leq d^+_{\max}|\mathcal{V}_j|, \quad j = 1, \ldots, L. \tag{4.9}$$

Actually, we can prove a stronger fact that is stated as follows. A set $B \subseteq U - U_{i-1}$ is *minimal augmentation* of set $A \subseteq U - U_{i-1}$, where A is infeasible to $(\mathrm{IP})_i$, if $A \subseteq B$ and B is feasible to $(\mathrm{IP})_i$ but the removal of any $w \in B - A$ violates the feasibility. We here claim that, for any $A \subseteq U - U_{i-1}$ and any minimal augmentation B of A, it holds that

$$\sum_{S \in \mathcal{V}(A)} |\Delta_B(S)| \ \le\ d^+_{\max}|\mathcal{V}(A)|. \tag{4.10}$$

Notice that (4.9) can be obtained from (4.10), by letting $A = \{w_1, \ldots, w_{j-1}\}$ and $B = A \cup A_i$ for $j = 1, \ldots, L$ (note $|\Delta_B(S)| \ge |\Delta_{A_i}(S)|$, and B is a minimal augmentation of A due to the *reverse delete step* in Lines 13 – 15 of algorithm PRIMAL-DUAL).

Thus we conclude with the proof of (4.10). In the following, we fix A and B, and use "violated set" instead of "violated set with respect to A" for simplicity. We first introduce the notation of *witness set*. Let $B_{\mathrm{V}} \triangleq \bigcup_{S \in \mathcal{V}(A)} \Delta_B(S) = \bigcup_{S \in \mathcal{V}(A)} \Delta(S) \cap B \subseteq B - A$.

Definition 4.3 (witness set). *A terminal subset $C \subseteq V$ is witness set of a nonterminal $u \in B_{\mathrm{V}}$ if* (i) $h_i(C) = 1$, *and* (ii) $\Delta_B(C) = \Delta(C) \cap B = \{u\}$ *hold.*

Notice that a witness set is a violated set (note $u \notin A$). Moreover, there must exist a witness set for any $u \in B_{\mathrm{V}}$, since B is feasible and the removal of u violates the feasibility. Let us call $\{C_u \mid u \in B_{\mathrm{V}}\}$ *witness set family*, where for each $u \in B_{\mathrm{V}}$, exactly one witness set C_u of u is included. Hence in a witness set family, we have

$$C_a \ne C_b \implies a \notin \Delta_B(C_b),\ b \notin \Delta_B(C_a) \implies a \notin \Delta(C_b),\ b \notin \Delta(C_a).$$

Lemma 4.3. *There exists a laminar (i.e., intersect-free) witness set family.*

Proof. Given a witness set family, we construct a laminar witness set family.

Let C_a and C_b be two intersecting witness sets with respect to $a, b \in B_{\mathrm{V}}$, respectively. Since they are violated, we see that either sets $C_a \cap C_b$ and $C_a \cup C_b$, or sets $C_a - C_b$ and $C_b - C_a$ are also violated (by Lemma 4.1). Suppose that $C_a \cap C_b$ and $C_a \cup C_b$ are violated. We show that C_a and C_b can be replaced by $C_a \cap C_b$ and $C_a \cup C_b$ in the witness set family.

First, by the definition of violated set, $C_a \cap C_b$ and $C_a \cup C_b$ must satisfy condition (i) in the definition of witness set. We show that condition (ii) is also satisfied. By the feasibility of B, $|\Delta_B(C_a \cap C_b)| \ge 1$ and $|\Delta_B(C_a \cup C_b)| \ge 1$ hold. However, by submodularity, we have

$$|\Delta_B(C_a \cap C_b)| + |\Delta_B(C_a \cup C_b)| \ \le\ |\Delta_B(C_a)| + |\Delta_B(C_b)| = 2.$$

Hence it must be the case of

$$|\Delta_B(C_a \cap C_b)| \ =\ |\Delta_B(C_a \cup C_b)| \ =\ 1. \tag{4.11}$$

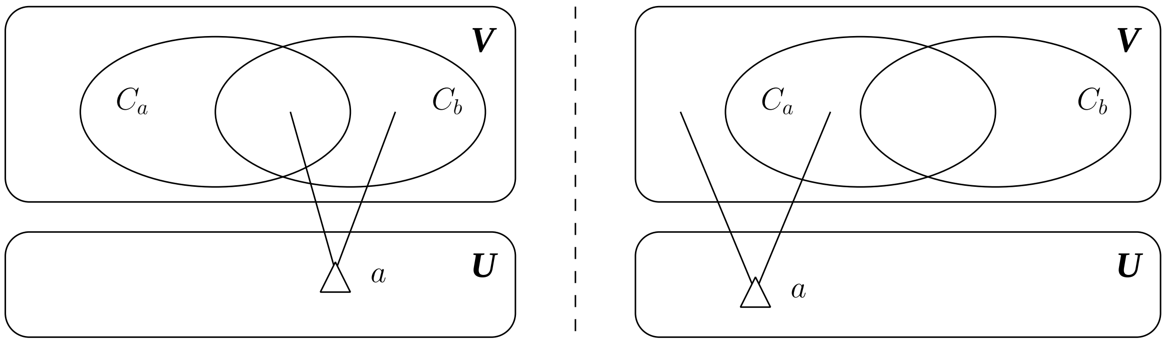


Figure 4.3: It holds that $a \in \Delta_B(C_a \cap C_b) \cup \Delta_B(C_a \cup C_b)$.

On the other hand, $C_a \neq C_b$ implies $a \notin \Delta(C_b)$. Hence either $\Gamma(a) \subseteq C_b$ or $\Gamma(a) \cap C_b = \emptyset$ holds. By $a \in \Delta(C_a)$ we see that, if $\Gamma(a) \subseteq C_b$ then $a \in \Delta_B(C_a \cap C_b)$; otherwise if $\Gamma(a) \cap C_b = \emptyset$ then $a \in \Delta_B(C_a \cup C_b)$. This is illustrated in Figure 4.3.

For the same reason, $b \in \Delta_B(C_a \cap C_b) \cup \Delta_B(C_a \cup C_b)$ holds. Combining with (4.11), we see that $\Delta_B(C_a \cap C_b) = \{a'\}$ and $\Delta_B(C_a \cup C_b) = \{b'\}$ hold for $\{a', b'\} = \{a, b\}$. Thus we can replace C_a and C_b by $C_a \cap C_b$ and $C_a \cup C_b$ in the witness set family.

Similarly, if $C_a - C_b$ and $C_b - C_a$ are violated sets, we can use them to replace C_a and C_b. In both cases, this process decreases the total number of pairs of intersecting sets. Thus it halts after a finite number of steps, and we then obtain a laminar witness set family. □

Let $\mathcal{F} = \{V\} \cup \{C_u \mid u \in B_{\mathrm{V}}\}$ be the family obtained by adding $\{V\}$ to a laminar witness set family. Let us construct a rooted tree $\mathcal{T}$ from $\mathcal{F}$ by the set inclusion relationship. To avoid confusion, we use "node" in tree $\mathcal{T}$, and use "vertex" in graph G.

Tree $\mathcal{T}$ consists of $|\mathcal{F}|$ nodes: u_C for $C \in \mathcal{F}$. The root is u_V. The parent of each non-root node u_C is the node $u_{C'}$ for the minimum $C' \in \mathcal{F}$ satisfying $C \subset C'$. See Figure 4.4. We *associate* each minimum violated set $S \in \mathcal{V}(A)$ with the node u_C for the minimum

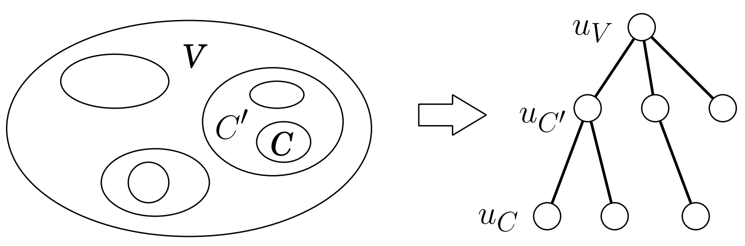


Figure 4.4: Construct a tree from a laminar set family by the set inclusion relationship.

$C \in \mathcal{F}$ satisfying $S \subseteq C$. Let $u(S)$ denote the node with which S is associated. Let $n_C = \big|\{S \in \mathcal{V}(A) \mid u(S) = u_C\}\big|$ denote the number of minimal violated sets that are associated with node u_C. Let $Q = \{u_C \mid n_C \geq 1\}$ be the subset of nodes with which at least one minimal violated set is associated. Clearly we have

$$|\mathcal{V}(A)| = \sum_{u_C \in Q} n_C. \tag{4.12}$$

For a non-root node u_C, C is a witness set, hence violated. This implies that, if node u_C has degree one, then $C = u(S)$ holds for some $S \in \mathcal{V}(A)$. In other words, *all non-root nodes of degree one must belong to Q*. Let $d(u_C)$ denote the degree of node u_C. This observation shows that

$$\sum_{u_C \notin Q} d(u_C) \geq 2(|\mathcal{F}| - |Q|) - 1.$$

On the other hand, $\sum_{u_C} d(u_C) = 2(|\mathcal{F}| - 1)$ holds since $\mathcal{T}$ is a tree. Thus we have

$$\sum_{u_C \in Q} d(u_C) = \sum_{u_C} d(u_C) - \sum_{u_C \notin Q} d(u_C) \leq 2|Q| - 1. \tag{4.13}$$

Let us next show that

$$\sum_{u_C \in Q} \min\{d^+_{\max} - 1, n_C\} d(u_C) \leq d^+_{\max} \sum_{u_C \in Q} n_C. \tag{4.14}$$

For this, let $X = \{u_C \in Q | n_C \geq d^+_{\max} - 1\}$, $Y = \{u_C \in Q | n_C = 1\} - X$ and $Z = Q - X - Y$. The left-hand side of (4.14) is then at most

$$\begin{aligned}
&(d^+_{\max} - 1) \sum_{u_C \in X} d(u_C) + \sum_{u_C \in Y} d(u_C) + (d^+_{\max} - 2) \sum_{u_C \in Z} d(u_C) \\
\leq\ &(d^+_{\max} - 1)\Big(2(|X| + |Y| + |Z|) - 1 - \sum_{u_C \in Y} d(u_C) - \sum_{u_C \in Z} d(u_C)\Big) \\
&\qquad + \sum_{u_C \in Y} d(u_C) + (d^+_{\max} - 2) \sum_{u_C \in Z} d(u_C)
\end{aligned}$$

by (4.13) and $|Q| = |X| + |Y| + |Z|$. On the other hand, the right-hand side of (4.14) is at least $d^+_{\max}((d^+_{\max} - 1)|X| + |Y| + 2|Z|)$. By a simple calculation, we see that (4.14) holds.

Finally, let us show that, for each $u_C \in Q$, it holds that

$$\sum_{S \in \mathcal{V}(A): u(S) = u_C} |\Delta_B(S)| \leq \min\{d^+_{\max} - 1, n_C\} d(u_C). \tag{4.15}$$

This implies (4.10), hence the lemma by combining with (4.12) and (4.14). For this, consider an arbitrary $S \in \mathcal{V}(A)$ and an arbitrary $v \in \Delta_B(S)$. Let C_v be the witness set of v in family $\mathcal{F}$. By the definitions of witness set and witness set family, we have the next observation.

Observation 4.1. *For any set $C' \in \mathcal{F} - \{C_v\}$, either $C' \subset C_v$ or $C_v \subset C'$ holds. Moreover, $C' \subset C_v$ implies $\Gamma(v) \cap C' = \emptyset$, whereas $C_v \subset C'$ implies $\Gamma(v) \subseteq C'$.* □

Since witness set is also violated set, we see that either $S \subseteq C_v$ or $S \cap C_v = \emptyset$ must hold.

Case 1: $S \subseteq C_v$. Notice that there is no $C' \in \mathcal{F}$ satisfying $S \subseteq C' \subset C_v$ (otherwise $\Gamma(v) \cap S \subseteq \Gamma(v) \cap C' = \emptyset$ holds by Observation 4.1, a contradiction). Hence S is associated with C_v, i.e., $u(S) = u_{C_v}$. Let u_C be the parent of u_{C_v} in tree $\mathcal{T}$ (such u_C must exist since $C_v \neq V$). Again by Observation 4.1, we have $\Gamma(v) \subseteq C$. We use an *upward directed edge* (u_{C_v}, u_C) (*tail* u_{C_v} and *head* u_C) to represent this case for an $S \in \mathcal{V}(A)$ and a $v \in \Delta_B(S)$ satisfying $u(S) = u_{C_v}$, where u_C is the parent node of u_{C_v}.

Notice that, edge (u_{C_v}, u_C) may not be unique, since there may exists other $S' \in \mathcal{V}(A)$ satisfying $v \in \Delta_B(S')$ and $u(S') = u_{C_v}$. Multiple directed edges (u_{C_v}, u_C) are allowed, but only one edge is assigned for a single set S'. By Corollary 4.1, such sets (S and all S') must be pairwise-disjoint subsets of C_v. Hence the total number of directed edges (u_{C_v}, u_C) is bounded by $\min\{|\Gamma(v)|-1, n_{C_v}\} \leq \min\{d^+_{\max}-1, n_{C_v}\}$ (notice that $|\Gamma(v) - C_v| \geq 1$ holds).

Case 2: $S \cap C_v = \emptyset$. Similarly, we see that $u(S) = u_C$ holds for the parent u_C of u_{C_v}. We use a *downward directed edge* (u_C, u_{C_v}) to represent this case for an $S \in \mathcal{V}(A)$ and a $v \in \Delta_B(S)$ satisfying $u(S) = u_C$, where u_C is the parent node of u_{C_v}. The total number of these edges (u_C, u_{C_v}) is at most $\min\{d^+_{\max} - 1, n_C\}$.

For a fixed $u_C \in Q$, these two cases may be possible simultaneously. But we have seen that, for each undirected edge $\{u_C, u_{C'}\}$ in tree $\mathcal{T}$, there are at most $\min\{d^+_{\max} - 1, n_C\}$ directed edges $(u_C, u_{C'})$. Thus there are at most $\min\{d^+_{\max} - 1, n_C\}d(u_C)$ directed edges with tail u_C. On the other hand, the way that directed edges are produced implies that the total number of directed edges with tail u_C (over all $S \in \mathcal{V}(A)$ and all $v \in \Delta_B(S)$) is nothing but $\sum_{S \in \mathcal{V}(A): u(S) = u_C} |\Delta_B(S)|$. Hence (4.15) is shown. Therefore we have proved Lemma 4.2 under the assumption $w(u) > 0$ for all $u \in U$.

It is easy to see that Lemma 4.2 also holds when there is a nonterminal $u \in U$ satisfying $w(u) = 0$. To see this, notice that, we only need to show (4.9) for indices j satisfying $\epsilon_j > 0$. However, this implies that $w(u) \geq \bar{w}(u) > 0$ for all $u \in \bigcup_{S \in \mathcal{V}_j} \Delta_{A_i - A}(S)$. Thus $|\Gamma(u)| \leq d^+_{\max}$ holds for all $u \in \bigcup_{S \in \mathcal{V}_j} \Delta_{A_i - A}(S)$, and the proof can be done in a straightforward manner. □

4.4 Implementation for SNDP

It is SNDP that motivated the study on NDRF. In this section, we consider to implement algorithm ALGO-IP for SNDP.

Let $G = (V, U, E)$ be a bipartite graph with a terminal set V and a nonterminal set U. We first show that SNDP can be formulated as NDRF (IP) with a requirement function r defined by

$$r(S) \;=\; \max\{r_{st} \mid s \in S,\ t \in V - S\}, \quad S \subseteq V. \tag{4.16}$$

A useful idea when considering nonterminal disjoint paths in bipartite graph is the following transformation $\mathcal{D}$, which is illustrated in Figure 4.5. (A directed graph can be viewed as a graph with edge orientation. An edge in a directed graph is denoted by (u, v) from tail u to head v. A path in directed graph follows the edge orientation.)

Definition 4.4 (Transformation $\mathcal{D} : G \to \overrightarrow{G}$). *Make a copy u^c and add a directed edge (u, u^c) of capacity one for each nonterminal u. Replace each undirected edge $\{v, u\}$ by directed edges (v, u) and (u^c, v) of capacity $+\infty$.*

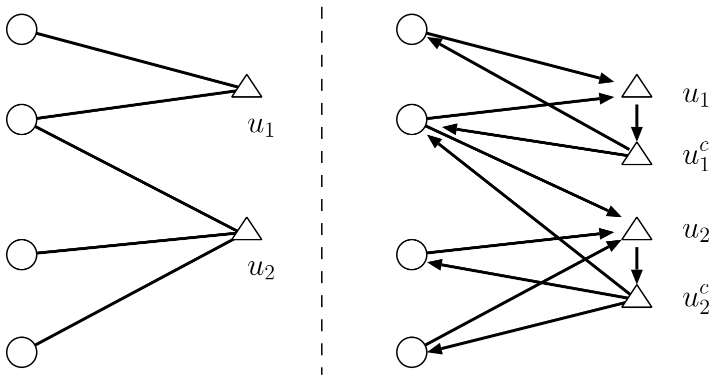


Figure 4.5: $\mathcal{D}$ transforms a bipartite graph into a directed graph, where edges (u, u^c) have capacity one, and other edges have capacity ∞.

Let $X^c \triangleq \{u^c | u \in X\}$ for $X \subseteq U$. In the following, a vertex in G is also treated as a vertex in $\overrightarrow{G}$, whereas notations Γ and Δ are used only with respect to G. Given an $X \subseteq U$, let us consider the digraph $\overrightarrow{G[V \cup X]}$. Let C be a set satisfying $\emptyset \neq C \subset T \cup X \cup X^c$. We call the set of directed edges from a vertex in C to a vertex not in C (directed) *cut*, and denote it by $\delta^+(C)$. The capacity of cut $\delta^+(C)$ is the sum of capacities of edges in $\delta^+(C)$.

Observation 4.2. *If the capacity of a cut $\delta^+(C)$ is finite, then $\Gamma_X(S) \triangleq \Gamma(S) \cap X \subseteq C$ and $(\Delta_X(S))^c \cap C = \emptyset$ must hold for $S = C \cap V$. Hence the capacity of (any) cut $\delta^+(C)$ is at least $|\Delta_X(C \cap V)|$.*

Observation 4.3. *For any set $S \subseteq V$, there is a cut $\delta^+(C)$ that has capacity $|\Delta_X(S)|$ and satisfies $S = C \cap V$. More precisely, we can choose $C = S \cup \Gamma_X(S) \cup (\Gamma_X(S) - \Delta_X(S))^c$.*

Now we are ready to show that SNDP can be formulated as NDRF with function r defined by (4.16). For two terminals $s, t \in V$, any k nonterminal disjoint s, t-paths in $G[V \cup X]$ can be viewed as an integer s, t-flow of value k in $\overrightarrow{G[V \cup X]}$, and vice versa. Thus X is feasible to SNDP (in G) if and only if the maximum s, t-flow in $\overrightarrow{G[V \cup X]}$ has value at least r_{st} for each pair of distinct $s, t \in V$. By maxflow-mincut theorem, this equals to that, any s, t-cut $\delta^+(C)$

in $\overrightarrow{G[V \cup X]}$ (i.e., a cut satisfying $s \in C$ and $t \notin C$) has capacity at least r_{st}. By the above observations, this is equivalent to that $|\Delta_X(S)| \geq \max\{r_{st} | s \in S,\ t \in V - S\}$ for all $S \subseteq V$. Thus SNDP is equivalent to NDRF with function r defined by (4.16).

We next show that Conditions 1 and 2 are satisfied. Condition 1 can be verified in a straightforward manner. Let us show Condition 2; i.e., the minimum violated sets with respect to any set $A \subseteq U - U_{i-1}$ (in the i^{th} phase) can be found in polynomial time. Again, we use "violated set" instead of "violated set with respect to A" for simplicity.

Lemma 4.4. *Let $\tilde{A} = U_{i-1} \cup A$. Let S be a minimal violated set, where $r(S) = r_{st}$ holds for $s \in S$ and $t \in V - S$. Then $S = C_{st} \cap V$ holds for a minimal minimum s,t-cut $\delta^+(C_{st})$ in directed graph $\overrightarrow{G[V \cup \tilde{A}]}$.*

Proof. Let $C = S \cup \Gamma_{\tilde{A}}(S) \cup (\Gamma_{\tilde{A}}(S) - \Delta_{\tilde{A}}(S))^c$. We show $C = C_{st}$, which implies the lemma. Clearly $\delta^+(C)$ is an s,t-cut. Let us next show that it is a minimum s,t-cut. The capacity of $\delta^+(C)$ is $|\Delta_{\tilde{A}}(S)|$, whereas the capacity of $\delta^+(C_{st})$ is at least $|\Delta_{\tilde{A}}(S')|$ for $S' = C_{st} \cap V$. Let us show $|\Delta_{\tilde{A}}(S)| \leq |\Delta_{\tilde{A}}(S')|$ (hence $|\Delta_{\tilde{A}}(S)| = |\Delta_{\tilde{A}}(S')|$).

Notice that S is a violated set, implying $h_i(S) = 1$ and $\Delta_A(S) = \emptyset$. Thus $r(S) - |\Delta_{U_{i-1}}(S)| = r_{\max} - i + 1$ and $\Delta_A(S) = \emptyset$ hold. Hence we have

$$|\Delta_{\tilde{A}}(S)| \;=\; |\Delta_{U_{i-1}}(S)| + |\Delta_A(S)| \;=\; r_{st} - r_{\max} + i - 1. \tag{4.17}$$

Similarly, if S' is also a violated set, then we have

$$|\Delta_{\tilde{A}}(S')| \;=\; r(S') - r_{\max} + i - 1 \;\geq\; r_{st} - r_{\max} + i - 1. \tag{4.18}$$

(Note that $s \in S'$ and $t \in V - S'$ hold.) Otherwise S' is not violated, we then have $h_i(S') = 1$ and $\Delta_A(S') \neq \emptyset$, or $h_i(S') = 0$ (hence $r(S') - |\Delta_{U_{i-1}}(S)| \leq r_{\max} - i$). In both cases, it holds that

$$|\Delta_{\tilde{A}}(S')| \;\geq\; r(S') - r_{\max} + i \;\geq\; r_{st} - r_{\max} + i. \tag{4.19}$$

Thus we have $|\Delta_{\tilde{A}}(S)| \leq |\Delta_{\tilde{A}}(S')|$, which implies that $\delta^+(C)$ is a minimum s,t-cut. The above proof shows that S' must be a violated set. Hence $S \subseteq S'$ by Corollary 4.1, which implies $C \subseteq C_{st}$. Since C_{st} is minimal, we have $C = C_{st}$. □

Lemma 4.4 shows that, we can identify the minimal violated sets by computing a minimal minimum s,t-cut in $\overrightarrow{G[V \cup \tilde{A}]}$ for each pair of $s, t \in V$, and checking if they are violated and minimal among these $O(|V|^2)$ cuts. It is well known that the (unique) minimal minimum s,t-cut can be found by one maxflow computation in $O(p^3)$ time for a p-vertices directed graph ([GT 88]). Thus the time for finding minimal violated sets is dominated by $O(|V|^2)$ maxflow computations. Hence algorithm ALGO-IP can have $O(r_{\max}|U||V|^2(|V|+|U|)^3)$ running time. We summary the arguments so far as the next theorem.

Theorem 4.2. *The SNDP problem can be approximated within a factor of $d^+_{\max}\mathcal{H}(r_{\max})$ in $O(r_{\max}mn^2(n+m)^3)$ time, where $d^+_{\max}$ is the maximum degree of nonterminals with positive weight, $r_{\max}$ is the maximum requirement, m and n are the numbers of nonterminals and terminals, respectively.* □

4.5 Remarks

We note that the guarantee $d^+_{\max}$ in Lemma 4.2 of algorithm PRIMAL-DUAL is essentially tight (a tight example will be given later). Nevertheless, it can deliver optimal solutions for some problem classes. For instance, it is easy to see that algorithm PRIMAL-DUAL simulates Kruskal's algorithm [Kru 56] for MST (minimum spanning tree problem) in graphs. It also simulates Dijkstra's algorithm [Dij 59] for SPP (shortest path problem) in graphs. Thus at least for MST and SPP in graphs, algorithm PRIMAL-DUAL finds optimal solutions. See [GW 97] for more details.

Let us consider the *shortest path problem in hypergraphs*, i.e., SNDP with $r_{st} = 1$ for two given terminals s and t (0 for all other pairs of vertices). It can be solved by Dijkstra's algorithm in the next manner.

> Transform the input bipartite graph into a directed graph by $\mathcal{D}$ as defined in Definition 4.4. Assign each directed edge (u, u^c) for a nonterminal u with the same weight of u, and assign other edges with weight 0. Then apply Dijkstra's algorithm to the resulting directed graph. Finally, transform the output (directed) path to a path in the input bipartite graph in a straightforward manner.

It is easy to see that the above procedure finds an optimal solution. However, it is nothing but the procedure of algorithm PRIMAL-DUAL. Hence PRIMAL-DUAL can find an optimal solution for the shortest path problem in hypergraphs. We note that this problem is equivalent to finding a shortest path in given node-weighted graph.

We now show a tight example for Lemma 4.2. See Figure 4.6.

There are $d+p$ terminals $v_1, v_2, \ldots, v_{d+p}$. Terminal v_1 is connected to other terminals via cost $1+\epsilon$ nonterminals (hollow triangles) for a small $\epsilon > 0$. Terminals $v_2, \ldots, v_d$ are connected to $v_{d+1}, \ldots, v_{d+p}$ via p cost-d degree-d nonterminals (solid triangles), in such a way that every v_i, $i = 2, \ldots, d$, is a neighbor of all the p nonterminals (thus for any one of these p nonterminals, there is a unique neighbor v_j of it with $d+1 \le j \le d+p$). The objective is to find a minimum cost set of nonterminals to connect terminals $v_2, \ldots, v_{d+p}$.

Notice that, at the beginning of the algorithm PRIMAL-DUAL, the minimal violated set family is $\mathcal{V}(\emptyset) = \{\{v_2\}, \ldots, \{v_{d+p}\}\}$. Thus the algorithm will assign the dual variables of the singleton sets $\{v_i\}$ to one, $2 \le i \le d+p$. As a result, all the nonterminals of cost d are picked.

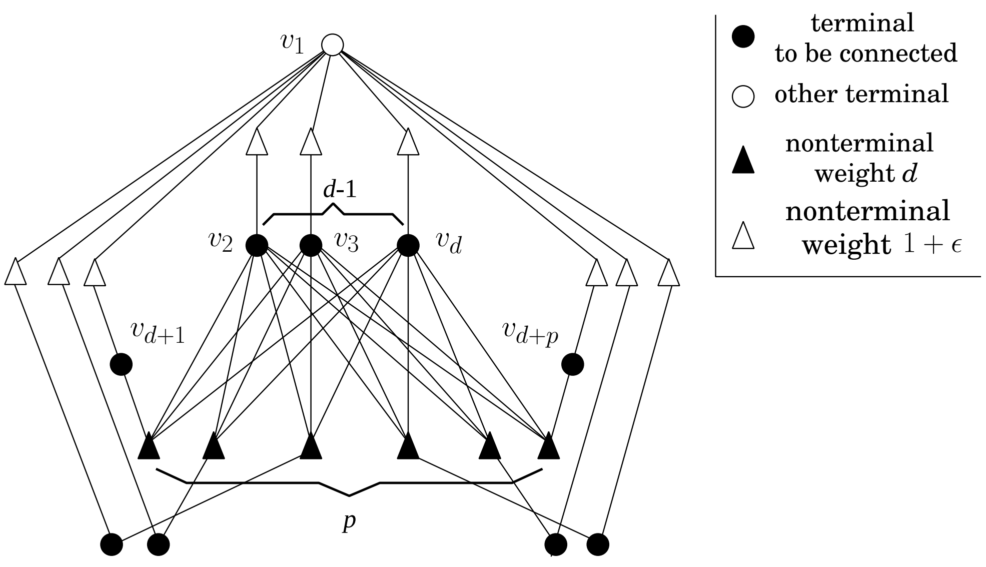


Figure 4.6: A tight example for Lemma 4.2.

Hence the output has cost dp. However, the optimal solution is the set of nonterminals with cost $1+\epsilon$. Thus the guarantee is as bad as $\frac{dp}{(d+p-1)(1+\epsilon)}$, which tends to d when $p \to \infty$ and $\epsilon \to 0$.

We note that, Goemans et al. [GGPSTW 94] have shown that the performance guarantee $2\mathcal{H}(r_{\max})$ for SNDP-G is tight up to a factor of 2. It is thus interesting to know whether an algorithm with improved performance guarantee can be developed, e.g., via an iterative rounding process for SNDP-G and SNDP-G$^+$ as used by in [Jain 98, FJW 01]. This question is still open. But we will see in the next chapter that improved guarantee can be obtained by a very simple transformation.

Chapter 5

Another Approach for SNDP

In this chapter, we consider to design approximation algorithms for SNDP (survivable network design problem) on the basis of algorithm designed for simpler versions, i.e., SNDP-G or SNDP-G$^+$. The approach is very simple. Given an instance of SNDP, we define an instance of SNDP (or SNDP-G$^+$) in a graph with only edge weights. After solving it by a known (approximation) algorithm, we convert the output to a feasible solution of the original problem of SNDP. Finally, we show that a similar approach is also applicable for NDRF (network design with requirement function).

5.1 Introduction

Recall that SNDP, SNDP-G and SNDP-G$^+$ are formulated as follows respectively.

Problem 5.1 (SNDP). *Given a bipartite graph $G = (V, U, E)$ with a nonterminal weight function $w : U \to \mathbf{R}^+$ and connectivity requirements r_{st} for each pair of distinct terminals $s, t \in V$, find a minimum weight subset $U^* \subseteq U$ such that there are at least r_{st} nonterminal disjoint paths between each pair of $s, t \in V$ in the subgraph $G^* = G[V \cup U^*]$ of G.*

Problem 5.2 (SNDP-G). *Given a graph $G = (V, E)$ with an edge weight function $w : E \to \mathbf{R}^+$ and connectivity requirements r_{st} for each pair of distinct vertices $s, t \in V$, find a minimum weight subset $E^* \subseteq E$ such that there are at least r_{st} edge disjoint paths between each pair of $s, t \in V$ in the subgraph $G^* = (V, E^*)$ of G.*

Problem 5.3 (SNDP-G$^+$). *Given a bipartite graph $G = (V, U, E)$ with a terminal set V, a nonterminal set U, an edge weight function $w : E \to \mathbf{R}^+$ and connectivity requirements r_{st} for each pair of distinct terminals $s, t \in V$, find a minimum weight edge $E^* \subseteq E$ such that there are at least r_{st} nonterminal disjoint paths between each pair of $s, t \in V$ in the subgraph $G^* = (V \cup U, E^*)$ of G.*

Notice that the original definition of SNDP-G$^+$ is slightly different (see Section 4.1).

5.2 Algorithms for SNDP

Given an instance $I_{HG} = (G = (V, U, E), w, r)$ of SNDP, in order to employ an approximation algorithms for SNDP-G (or for SNDP-G$^+$), we first define an edge weight function $\bar{w}$ according to the nonterminal weight function w, which is illustrated in Figure 5.1.

Definition 5.1 (weight function $\bar{w}$). *Given an instance $I_{HG} = (G = (V, U, E), w, r)$ of SNDP, assign each edge $e \in \delta(u)$ with weight $\bar{w}(e) = \frac{w(u)}{d_u}$ for each $u \in U$ with degree d_u.*

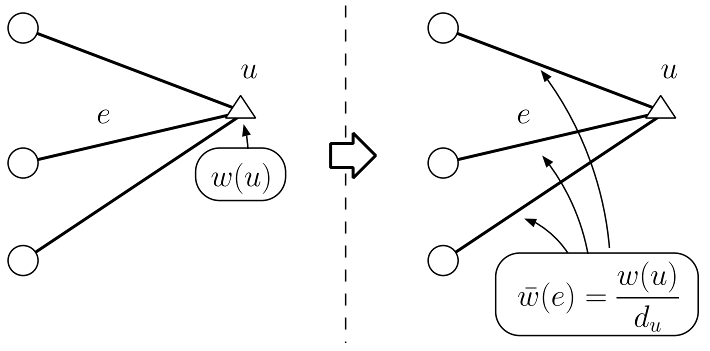


Figure 5.1: Construction of the edge weight function $\bar{w}$ from nonterminal weight function w.

5.2.1 Approximating SNDP by employing algorithm for SNDP-G

Given an instance $I_{HG} = (G = (V, U, E), w, r)$ of SNDP, we define an instance $I_G = (G' = (V \cup U, E), \bar{w}, r')$ of SNDP-G, where graph G' is actually the same as G, and the edge weight function $\bar{w}$ is defined in Definition 5.1. The connectivity requirement r' is given by

$$r'_{st} = \begin{cases} r_{st} & \text{if } s, t \in V, \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

Suppose that I_G is solved (or approximated) by a known algorithm. We construct from the output to a feasible solution of I_{HG}. The resulting algorithm, called SNDP-G-TO-SNDP, is given in Figure 5.2.

Clearly, algorithm SNDP-G-TO-SNDP runs in polynomial time if so does the algorithm used in Step 2. Let us now prove the correctness and derive its performance guarantee. Again, given a bipartite graph, we use $d_{\max}$ and $d^+_{\max}$ to denote the maximum degree of nonterminals and of nonterminals with positive weights, respectively. We have the next theorem.

Theorem 5.1. *Algorithm SNDP-G-TO-SNDP finds a feasible solution U' of the given instance I_{HG} of SNDP if $d_{\max} \le 3$. If an α-approximation algorithm is used in the second step, then the weight of U' is at most $\frac{d^+_{\max}\alpha}{2}$ times of the optimum.*

Input: An instance $I_{HG} = (G = (V, U, E), w, r)$ of SNDP.
Output: A feasible solution $U' \subseteq U$.

```
1   Construct the instance $I_G = (G' = (V \cup U, E), \bar{w}, r')$ of SNDP-G.
2   Apply an approximation algorithm (for SNDP-G) to $I_G$.
        Let $E' \subseteq E$ be the output subset of edges.
3   **Output** $U' = \{u \in U \mid |E' \cap \delta(u)| \geq 2\}$.
```

Figure 5.2: Algorithm SNDP-G-TO-SNDP.

Proof. Consider two distinct terminals $s, t \in V$. Since E' is feasible to I_G, there are at least r_{st} edge disjoint paths in graph $G' = (V \cup U, E')$ (i.e., graph G) between s and t. Notice that, for any nonterminal $u \in U$ in an s, t-path, there are exactly two edges in the path that are incident to u (i.e., in $\delta(u)$). Since each nonterminal has degree at most three, this implies that edge disjoint s, t-paths are also nonterminal disjoint. Therefore, by the definition of U', there exist at least r_{st} nonterminal disjoint paths in graph $G[V \cup U']$ between s and t, showing the first part of theorem.

Now let $U^* \subseteq U$ be an optimal solution of the original instance I_{HG} of SNDP. Let

$$E^* = \{e \in E \mid e \cap W^* \neq \emptyset\}.$$

Notice that $\bar{w}(E^*) = w(U^*)$ holds by the definition of $\bar{w}$. Since graph is bipartite, we see that nonterminal disjoint paths are also edge disjoint. Thus edge subset E^* is feasible to the instance I_G of SNDP-G. Since we use an α-approximation algorithm in the second step, the weight of E' is thus at most α times the weight of E^*; i.e.,

$$\bar{w}(E') \leq \alpha \bar{w}(E^*) = \alpha w(U^*).$$

By the definition of U', a nonterminal u (of weight $w(u)$) is included in U' if and only if there are at least two edges (with weight $\frac{w(u)}{d_u}$) in $E' \cap \delta(u)$. Thus we have

$$w(U') \leq \frac{d^+_{\max}}{2} \bar{w}(E') \leq \frac{d^+_{\max}}{2} w(U^*).$$

(Note that nonterminals with zero weights can be omitted in the weight of U'). $\square$

In particular, since there is a 2-approximation algorithm for SNDP-G ([Jain 98]), Theorem 5.1 shows that SNDP with $d_{\max} \leq 3$ can be approximated within a factor of $d^+_{\max}$ in polynomial time. In the next subsection, we will show that, this result can be extended to the general SNDP without restriction on $d_{\max}$.

5.2.2 Approximating SNDP by employing algorithm for SNDP-G$^+$

Given an instance $I_{HG} = (G = (V, U, E), w, r)$ of SNDP, we consider to approximate it by employing an approximation algorithm for SNDP-G$^+$. The resulting algorithm, called SNDP-G$^+$-TO-SNDP, is given in Figure 5.3.

Input: An instance $I_{HG} = (G=(V, U, E), w, r)$ of SNDP.
Output: A feasible solution $U'' \subseteq U$.

```
1   Construct instance $I_{G^+} = (G=(V,U,E), \bar{w}, r')$ of SNDP-G$^+$,
        where $V$ is the terminal set and $U$ is the nonterminal set.
        Functions $\bar{w}$ $r'$ are defined by Definition 5.1 and (5.1), respectively.
2   Apply an approximation algorithm (for SNDP-G$^+$) to $I_{G^+}$.
        Let $E''$ be the output subset of edges.
3   **Output** $U'' = \Big\{u \in U \mid |E'' \cap \delta(u)| \geq 2\Big\}$.
```

Figure 5.3: Algorithm SNDP-G$^+$-TO-SNDP

Theorem 5.2. *Algorithm SNDP-G$^+$-TO-SNDP finds a feasible solution U'' of the input instance I_{HG} of SNDP. If a β-approximation algorithm is used in the second step, then the weight of U'' is at most $\frac{d^+_{\max}\beta}{2}$ times of the optimum.*

Proof. Since E'' is feasible to the instance I_{G^+} of SNDP-G$^+$, we see that, there are at least r_{st} nonterminal disjoint paths in graph (V, U, E'') between each pair of distinct terminals $s, t \in V$. For any nonterminal $u \in U$ in an s,t-path, there are exactly two edges in the path that are incident to u (i.e., in $\delta(u)$). Thus set U'' is feasible to I_{HG}.

Now let $U^* \subseteq U$ be an optimal solution of I_{HG}. Define $E^* = \{e \in E \mid e \cap W^* \neq \emptyset\}$. The feasibility of U^* to I_{HG} implies the feasibility of E^* to the instance I_{G^+} of SNDP-G$^+$. It is then easy to see the rest of the theorem. □

In particular, we can use the 2-approximation algorithm for SNDP-G$^+$ ([FJW 01]), and hence obtain the next corollary.

Corollary 5.1. *SNDP can be approximated within a factor of $d^+_{\max}$ in polynomial time.* □

5.3 Extensions to NDRF

Let us consider to generalize the simple approach in the previous section to NDRF. Let NDRF-G be the graph restriction of hypergraph-based formulation of NDRF. The formulation of NDRF and NDRF-G are given in the following.

Problem 5.4 (NDRF). *Given a bipartite graph $G = (V, U, E)$ with a terminal set V, a nonterminal set U, a set E of edges between V and U, a nonterminal weight function $w : U \to \mathbf{R}^+$ and a requirement function $r : 2^V \to \mathbf{Z}^+$, NDRF is defined as follows.*

$$\begin{aligned}
&\text{minimize} && \sum_{u \in U} w(u)x_u && && (5.2)\\
&\text{subject to} && x(\Delta(S)) \geq r(S) && \forall S \subseteq V, && (5.3)\\
& && x_u \in \{0,1\} && \forall u \in U. && (5.4)
\end{aligned}$$

Recall that $\Delta(S) = \Gamma(S) \cap \Gamma(V - S)$, where $\Gamma(S) = \{v \notin S \mid \exists\, e, v \in e, e \cap S \neq \emptyset\}$ is the set of neighbors of S.

Problem 5.5 (NDRF-G). *Given a graph $G = (V, E)$ with a vertex set V, an edge set E, an edge weight function $w : E \to \mathbf{R}^+$ and a requirement function $r : 2^V \to \mathbf{Z}^+$, NDRF-G is defined as follows.*

$$\begin{aligned}
&\text{min} && \sum_{e \in E} w(e)x_e && && (5.5)\\
&\text{s.t.} && x(\delta(S)) \geq r(S) && \forall S \subseteq V, && (5.6)\\
& && x_e \in \{0,1\} && \forall e \in E. && (5.7)
\end{aligned}$$

Recall that $\delta(S) = \{e \mid \emptyset \neq e \cap S \neq e\}$ is the set of edges incident to S.

5.3.1 Approximating NDRF by employing algorithm for NDRF-G

Define a requirement function $\bar{r} : 2^{V \cup U} \to \mathbf{Z}^+$ by

$$\bar{r}(S) \triangleq r(S \cap V) \quad S \subseteq V \cup U. \tag{5.8}$$

Given an instance $I_{HG} = (G = (V, U, E), w, r)$ of NDRF satisfying $d_{\max} \leq 3$, we consider the next algorithm, called NDRF-G-TO-NDRF, in Figure 5.4. Notice that since there is no conception of path, the third step is different from SNDP-G-TO-SNDP.

We note that, in order to apply one of the algorithms in [GGW 93, GGPSTW 94, Jain 98, WGMV 93] in the second step, certain conditions must be satisfied. As one of such conditions, it easy to see that if r is weakly supermodular, then so is $\bar{r}$.

Again, algorithm NDRF-G-TO-NDRF runs in polynomial time if and only if so does the algorithm used in the second step. We have the next theorem.

Theorem 5.3. *Algorithm NDRF-G-TO-NDRF finds a feasible solution U' of the input instance I_{HG} of NDRF if $d_{\max} \leq 3$. If an α-approximation algorithm is used in the second step, then the weight of U' is at most $d^+_{\max}\alpha$ times of the optimum.*

Input: An instance $I_{HG} = (G = (V, U, E), w, r)$ of NDRF.
Output: A feasible solution $U' \subseteq U$.

```
1   Construct instance $I_G = (G' = (V \cup U, E), \bar{w}, \bar{r})$ of NDRF-G,
        where functions $\bar{w}$ and $\bar{r}$ are given by Definition 5.1 and (5.8) respectively.
2   Apply an approximation algorithm (for NDRF-G) to $I_G$.
        Let $E' \subseteq E$ be the output subset of edges.
3   **Output** $U' = \Big\{u \in U \mid E' \cap \delta(u) \neq \emptyset\Big\}$.
```

Figure 5.4: Algorithm NDRF-G-TO-NDRF

Proof. We first show that U' is feasible to I_{HG}; i.e., $|\Delta(S) \cap U'| \geq r(S)$ holds for all $S \subseteq V$. For this, let

$$\bar{S} \;=\; S \cup \{u \in \Delta(S) \mid \exists\, t_1, t_2 \in S, \text{ s.t. } t_1 \neq t_2 \text{ and } \{t_1, u\}, \{t_2, u\} \in E'\Big\}.$$

It is easy to see that $|\delta(\bar{S}) \cap E'| = |\Delta(S) \cap U'|$ holds (note that the degree of any nonterminal is at most 3). Therefore, we obtain $|\Delta(S) \cap U'| = |\delta(\bar{S}) \cap E'| \geq \bar{r}(\bar{S}) = r(S)$.

Similarly to the proof of Theorem 5.1, the rest of the theorem can be shown easily. □

5.3.2 Approximating NDRF by employing algorithm for NDRF-G$^+$

It is not hard to see that, NDRF with $d^+_{\max} = 2$ is a generalization of SNDP-G$^+$. Let us call it *NDRF-G*$^+$. Given an instance $I_{HG} = (G = (V, U, E), w, r)$ of NDRF, we consider to approximate I_{HG} by employing an algorithm for NDRF-G$^+$.

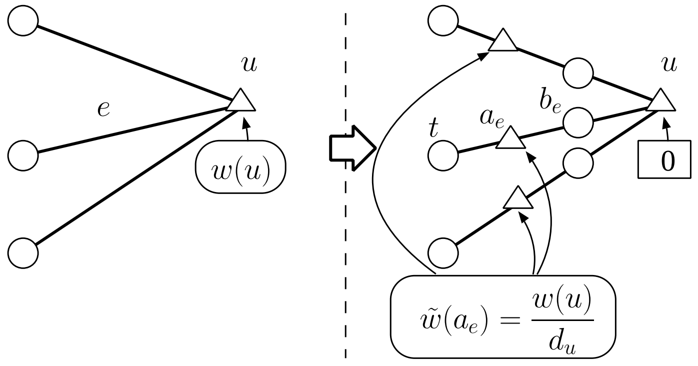


Figure 5.5: Construction of the weight function $\tilde{w}$ from the weight function w.

For this, we first define a nonterminal weight function $\tilde{w}$ as follows (see Figure 5.5). For all nonterminals $u \in U$ with weights $w(u) > 0$ of degree $d_u \geq 3$, we replace each edge

$e = \{t, u\} \in E$ with three edges $e_1 = \{t, a_e\}$, $e_2 = \{a_e, b_e\}$ and $e_3 = \{b_e, u\}$, introducing a new nonterminal a_e and a new terminal b_e. Let the weight of each nonterminal a_e be $\frac{w(u)}{d_u}$, and the weight of each nonterminal u be zero. Notice that $d^+_{\max} = 2$ holds in the resulting bipartite graph. Let $\tilde{w}$ denote the resulting nonterminal weight function, and $\tilde{V}$, $\tilde{U}$ and $\tilde{E}$ denote the resulting terminal set, nonterminal set and edge set, respectively.

Define a connectivity requirement function $\tilde{r}$ by $\tilde{r}(S) \triangleq r(S \cap V)$ for all $S \subseteq \tilde{V}$. Function $\tilde{r}$ is weakly supermodular if so is r. In this way, we can convert the instance I_{HG} of NDRF to an instance $I_{G^+} = (\tilde{G} = (\tilde{V}, \tilde{U}, \tilde{E}), \tilde{w}, \tilde{r})$ of NDRF-G$^+$. We consider the next algorithm, called NDRF-G$^+$-TO-NDRF, in Figure 5.6.

Input: An instance $I_{HG} = (G = (V, U, E), w, r)$ of NDRF.
Output: A feasible solution $U'' \subseteq U$.

```
1 | Construct the instance $I_{G^+} = (\tilde{G} = (\tilde{V}, \tilde{U}, \tilde{E}), \tilde{w}, \tilde{r})$ of NDRF-G$^+$.
2 | Apply an approximation algorithm on $I_{G^+}$.
  |     Let $\tilde{U}' \subseteq \tilde{U}$ be the output subset of nonterminals.
3 | **Output** $U'' = \Big\{u \in U \mid a_e \in \tilde{U}'$ holds for some $e = \{t, u\} \in E\Big\}$.
```

Figure 5.6: Algorithm NDRF-G$^+$-TO-NDRF

The correctness and performance guarantee of algorithm NDRF-G$^+$-TO-NDRF are shown in the next theorem. The proof can be done similarly to the proof of Theorems 5.2 and 5.3, thus is omitted. We note that there is an approximation algorithm [JMVW 99] developed for NDRF-G$^+$ (under certain conditions). The approximation algorithm ALGO-IP in Chapter 4 also treats NDRF-G$^+$ as a special case.

Theorem 5.4. *Algorithm NDRF-G$^+$-TO-NDRF finds a feasible solution U'' to the input instance I_{HG} of NDRF. If an β-approximation algorithm is used in the second step, then the weight of U'' is at most $d^+_{\max}\beta$ times of the optimum.* □

Chapter 6

Conclusion

6.1 Approximation algorithms for MPP

In this thesis, we present a simple and unified framework for developing and analyzing approximation algorithms for various multiway partition problems in submodular systems. We formulate them as MPP (multiway partition problem), or MPP-NT (multiway partition problem with no target), and calculate a k-partition by greedily increasing the size of partition one per phase (algorithms GSA and M-GSA), or two per phase (algorithm GSA2 and M-GSA2). The approach used in this thesis for proving performance guarantees is unified, efficient and flexible. We summarize the obtained results in the next two tables.

Table 6.1: Approximation results obtained in this thesis for MPP-NT.

Problems	Guarantee	Running time	Remark
k-PPG	$2-\frac{2}{k}$	$O(kn(m+n\log n))$ [*]	simpler proof
k-PPG	$\approx 2-\frac{3}{k}$ [†]	$O(kmn^3\log(n^2/m)))$	improved guarantee
k-PPH-T1	$(1-\frac{1}{k})d^+_{\max}$ [‡]	$O(kn(n\log n+D))$ [§]	fast, simpler and improved
k-PPH-T2	$2-\frac{2}{k}$	$O(kn(n\log n+D))$	simpler and faster
k-PPH-T3	$2-\frac{2}{k}$	$O(kn(n\log n+D))$	fast, simpler and improved
k-PPSSS	$2-\frac{2}{k}$	$O(k\lvert V\rvert^3\theta)$ [¶]	simpler proof
k-PPMSS[‖]	$2-\frac{2}{k}$	$O(k\lvert V\rvert^3\theta)$	new
k-PPSS	$k-1$	$O(k\lvert V\rvert^3\theta)$	new

[*] m: number of edges; n: number of vertices.

[†] $2-\frac{3}{k}$ for odd $k\geq 3$, and $2-\frac{3k-4}{k^2-k}=2-\frac{3}{k}+\frac{1}{k^2-k}$ for even $k\geq 2$.

[‡] $d^+_{\max}$: maximum degree of hyperedges with positive weights.

[§] D: sum of degrees of hyperedges.

[¶] θ: time bound of the oracle for function f.

[‖] The same result holds for submodular system (V,f) satisfying $f(\emptyset)\geq 0$ and $f(V)\geq f(S)$, $\emptyset\neq S\subset V$.

We note that, while algorithm GSA can be done in polynomial time, implementing algorithm GSA2 in polynomial time is still open for most MPP-NTs. Nevertheless, assuming that polynomial time implementation are available, we have derived the performance guarantees of GSA2 for some MPP-NTs. We summarize them in Table 6.2.

Table 6.2: Performance guarantees of algorithm GSA2 for MPP-NT (assuming the existence of polynomial time implementations).

Problems	Guarantee
k-PPH-T1	$\approx (1-\frac{3}{2k})\min\{k, d^+_{\max}\}$ [*]
k-PPH-T2	$\approx 2-\frac{3}{k}$ [†]
k-PPH-T3	$\approx 2-\frac{3}{k}$ if $d^+_{\max}=2$ [‡]
k-PPH-T3	$2-\frac{2}{k}$ if $d^+_{\max}\geq 3$
k-PPSSS	$2-\frac{2}{k}$
k-PPMSS[§]	$2-\frac{3}{k}$
k-PPSS	$\lfloor\frac{k}{2}\rfloor$

[*] $(1-\frac{3}{2k})\min\{k, d^+_{\max}\}$ for an odd $k \geq 3$, and $(1-\frac{3k-4}{2k(k-1)})\min\{k, d^+_{\max}\}$ for an even $k \geq 2$.

[†] $2-\frac{3}{k}$ for an odd $k \geq 3$, and $2-\frac{3k-4}{k^2-k} = 2-\frac{3}{k}+\frac{1}{k^2-k}$ for an even $k \geq 2$.

[‡] $2-\frac{3}{k}$ for an odd $k \geq 3$, and $2-\frac{3k-4}{k^2-k} = 2-\frac{3}{k}+\frac{1}{k^2-k}$ for an even $k \geq 2$.

[§] The same result holds for submodular system (V, f) satisfying $f(\emptyset) \geq 0$ and $f(V) \geq f(S)$, $\emptyset \neq S \subset V$.

It seems difficult to show the performance guarantee for higher order greedy algorithms that increase the size of partition by three or more per phase. This is because analogous properties of those in Lemmas 2.1, 3.1 and 3.2 no longer hold even for k-PPG.

On the other hand, we have seen that, the performance guarantee $2-\frac{2}{k}$ of algorithm GSA for k-PPSSSis the best possible for the greedy splitting approach. That is, for any fixed $j \geq 1$, there is an instance of k-PPSSS for which one cannot get a performance guarantee better than $2-\frac{2}{k}$, even by greedily increasing the size of partition j per phase. This seems quite different from k-PPG.

We also note that it is still open to solve k-PPSS in polynomial time for $k \geq 3$. It is of further interest to consider k-PPSS and the extensions of greedy splitting approach.

Algorithms GSA and GSA2 can be modified to apply to general MPP. We have shown that the modified GSA (M-GSA) can be implemented in polynomial time, whereas the modified GSA2 (M-GSA2) do not have polynomial running time in general unless P=NP. For the problem of k-TPG, using algorithm M-GSA, we give a simple proof for the result of [MNI 93]. For other MPP problems except for k-MCP and k-TPH-T1 with $|T| = k$, we have shown new results by algorithm M-GSA. We summarize these results in Table 6.1.

Table 6.3: Approximation results obtained in this thesis for general MPP.

Problems	Guarantee	Remark
k-TPG	$2-2/k$	simpler proof
k-TPH-T1	$(1-1/k)\min\{k, d^+_{\max}\}$	new
k-TPH-T2, k-TPH-T3, k-TPMSS[a], k-TPSSS	$2-2/k$	new
k-TPSS	$k-1$	new

[a]The same result holds for submodular system (V, f) satisfying $f(\emptyset) \geq 0$ and $f(V) \geq f(S)$, $\emptyset \neq S \subset V$.

6.2 Approximation algorithms for SNDP

We have designed an approximation algorithm for SNDP (survivable network design problem). It is based on the primal-dual algorithm for SNDP-G (SNDP in graphs) established in [GGPSTW 94, WGMV 93]. As a result, we have shown that a performance guarantee of $d^+_{\max}\mathcal{H}(r_{\max})$ can be obtained, where $d^+_{\max}$ is the maximum degree of hyperedges with positive weights, $r_{\max}$ is the maximum connectivity requirement, and $\mathcal{H}(k) = 1 + \frac{1}{2} + \cdots + \frac{1}{k}$. Like the algorithms in [GGPSTW 94, WGMV 93], our algorithm is applicable to a more general problem NDRF (network design problem with requirement function), provided that certain conditions are satisfied.

We have also shown how to utilize an approximation algorithm for SNDP-G or SNDP-G$^+$ (SNDP with $d^+_{\max} = 2$) to approximate SNDP. In particular, we have shown that SNDP can be approximated within a factor of $d^+_{\max}$ in polynomial time. This is a much stronger performance guarantee than the primal-dual method based approximation algorithm, but may be of less practical value since it requires to solve the LP relaxation of SNDP.

Bibliography

[AKR 95] A. Agrawal, P. Klein, and R. Ravi, "When Trees Collide: An Approximation Algorithm for the Generalized Steiner Tree Problem in Networks," *SIAM J. Comput.*, vol. 24, no. 3, pp. 440–456, 1995.

[AMO 93] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows.* Prentice-Hall, 1993.

[Bar 00] F. Barahona, "On the k-cut Problem," *Oper. Res. Lett.*, vol. 26, pp. 99–105, 2000.

[Bel 57] R. E. Bellman, *Dynamic Programming.* Princeton Univ. Press, 1957.

[Ben 95] A. A. Benczúr, "Counterexamples for Directed and Node Capacitated Cut-Trees," *SIAM J. Comput.*, vol. 24, no. 3, pp. 505–510, 1995.

[BTV 99] D. Bertsimas, C. P. Teo, and R. Vohra, "Analysis of LP Relaxations for Multiway and Multicut Problems," *Networks*, vol. 34, pp. 102–114, 1999.

[BG 97] M. Burlet and O. Goldschmidt, "A New and Improved Algorithm for the 3-cut Problem," *Oper. Res. Lett.*, vol. 21, pp. 225–227, 1997.

[CCPS 98] W. J. Cook, W. H.Cunningham, W. R. Pulleyblank, and A. Schrijver, *Combinatorial Optimization.* New York, Wiley, 1998.

[CKR 98] G. Călinescu, H. Karloff, and Y. Rabani, "An Improved Approximation Algorithm for Multiway Cut," in *Proc. STOC* 1998. Full paper in *J. Comput. System Sci.* 60, no. 3, pp. 504–574, 2000.

[CGKLS 97] C. S. Chekuri, A. V. Goldberg, D. R. Karger, M. S. Levine, and C. Stein, "Experimental Study of Minimum Cut Algorithms," in *Proc. SODA* 1997, pp. 324–333.

[CO 96] S. Chopra and J. H. Owen, "Extended Formulations for the A-cut Problem," *Math. Program.*, vol. 73, no. 1, Ser. A, pp. 7–30, 1996.

[CO 99] S. Chopra and J. H. Owen, "A note on Formulations for the *A*-Partition Problem on Hypergraphs," *Discrete Appl. Math.*, vol. 90, pp. 115–133, 1999.

[CR 91] S. Chopra and M. R. Rao, "On the Multiway Cut Polyhedron," *Networks*, vol. 21, pp. 51–89, 1991.

[Col 87] C. J. Colbourn, *The Combinatorics of Network Reliability.* Oxford Univ. Press, 1987.

[Cun 85] W. H. Cunningham, "Optimal Attack and Reinforcement of a Network," *J. Assoc. Comput. Mach.*, vol. 32, no. 3, pp. 549–561, 1985.

[Cun 91] W. H. Cunningham, The Optimal Multiterminal Cut Problem, in *DIMACS Series in Disc. Math. and Theor. Comput. Sci.* vol. 5, pp. 105–120, 1991.

[CT 99] W. H. Cunningham and L. Tang, "Optimal 3-Terminal Cuts and Linear Programming," in *Proc. IPCO* 1999, *LNCS* 1610, pp. 114–125.

[DJPSY 83] E. Dalhaus, D. S. Johnson, C. H. Papadimitriou, P. Seymour, and M. Yannakakis, "The Complexity of Multiway Cuts," extended abstract, 1983. "The Complexity of the Multiterminal Cuts," *SIAM J. Comput.*, vol. 23, no. 4, pp. 864–894, 1994.

[DMM 97] M. Dell'Amico, F. Maffioli, and S. Martello (eds.), *Annotated Bibliographies in Combinatorial Optimization.* Wiley, 1997.

[Dij 59] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[Don 88] W. E. Donath, "Logic Partitioning," in B. T. Preas and M. J. Lorenzetti (eds.), *Physical Design Automation of VLSI Systems*, Benjamin Cummings, Menlo Park, CA, pp. 65–86, 1988.

[DSR 00] D. Z. Du, J. M. Smith, and J. H. Rubinstein (eds.), *Advances in Steiner Trees.* Kluwer Academic Pub. 2000.

[EFS 99] P. Erdos, A. Frank, and L. A. Székely, "Minimum Multiway Cuts in Trees," *Discrete Appl. Math.*, vol. 87, pp. 67–75, 1998.

[ES 94] P. Erdos and L. A. Székely, "On Weighted Multiway Cuts in Trees," *Math. Program.*, 65, pp. 93–105, 1994.

[ET 75] S. Even and R. E. Tarjan, "Network Flow and Testing Graph Connectivity," *SIAM J. Comput.*, 4, pp. 507–518, 1975.

[Fer 98] C. G. Fernandes, "A Better Approximation Ratio for the Minimum Size k-Edge-Connected Spanning Subgraph Problem," *J. Algorithms*, vol. 28, no. 1, pp. 105–124, 1998.

[Fra 97] A. Frank, Matroids and Submodular Functions, in [DMM 97], pp. 65–80.

[Fuj 91] S. Fujishige, *Submodular Functions and Optimization* (Annals of discrete mathematics 47). Elsevier, North-Holland, 1991.

[FF 56] L. R. Ford and D. R. Fulkerson, "Maximal Flow through a Network," *Canad. J. Math.*, no. 8, pp. 399–404, 1956.

[FJW 01] L. Fleischer, K. Jain, and D. P. Williamson, "An Iterative Rounding 2-Approximation Algorithms for the Element Connectivity Problem," in *Proc. FOCS*, 2001 (to appear).

[FK 00] A. Freund and H. Karlof, "A Lower Bound of $8/(7+\frac{1}{k})$ on the Integrality Ratio of the Căalinescu-Karloff-Rabani Relaxation for Multiway Cut," *Inform. Process. Lett.*, 75, pp. 43–50, 2000.

[FT 87] M. L. Fredman and R. E. Tarjan, "Fibonacci Heaps and Their Uses in Improved Network Optimization Problems," *J. Assoc. Comput. Mach.*, vol. 34, no. 3, pp. 596–615, 1987.

[Gab 95] H. N. Gabow, "A Matroid Approach to Finding Edge Connectivity and Packing Arborescences," *J. Compu. System Sci.*, vol. 50, no. 2, pp. 259–273, 1995.

[GGPSTW 94] M. X. Goemans, A. V. Goldberg, S. Plotkin, D. Shmoys, E. Tardos, and D. P. Williamson, "Improved Approximation Algorithms for Network Design Problems," in *Proc. SODA* 1994, pp. 223–232.

[GGU 72] M. R. Garey, R. L. Graham, and J. D. Ullman, "Worst Case Analysis of Memory Allocation Algorithms," in *Proc. STOC* 1972, pp. 143–150.

[GGW 93] H. N. Gabow, M. X. Goemans, and D. P. Williamson, "An Efficient Approximation Algorithm for the Survivable Network Design Problem," in *Proc. 3rd MPS Conf. on IPCO* 1993. Full paper in *Math. Program.*, vol. 82, no. 1-2, Ser. B, pp. 12–40, 1998.

[GH 61] R. E. Gomory and T. C. Hu, "Multi-Terminal Network Flows," *J. Soc. Indust. Appl. Math.*, vol. 9, no. 4, pp. 551–570, 1961.

[GH 88] O. Goldschmidt and D. S. Hochbaum, "A Polynomial Algorithm for the k-cut Problem for fixed k," in *Proc. FOCS* 1988. Full paper in *Math. Oper. Res.*, vol. 19, no. 1, pp. 24–37, 1994.

[GH 90] O. Goldschmidt, and D. S. Hochbaum, "Asymptotically Optimal Linear Algorithm for the Minimum k-cut in a Random Graph," *SIAM J. Discrete Math.*, vol. 3, no. 1, pp. 58–73, 1990.

[GJ 79] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness.* W. H. Freeman, San Francisco, 1979.

[GJS 76] M. R. Garey, D. S. Johnson, and L. Stockmeyer, "Some Simplified NP-complete Graph Problems," *Theoret. Comput. Sci.*, vol. 1, no. 3, pp. 237–267, 1976.

[GK 99] S. Guha and S. Khuller, "Improved Methods for Approximating Node Weighted Steiner Trees and Connected Dominating Sets," *Inform. and Compu.*, 150, pp. 57–74, 1999.

[GLS 88] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization.* Springer, Berlin, 1988.

[GP 88] G. Gallo and S. Pallottino, "Shortest Paths Algorithms," in *Annals of Operations Research* 13, pp. 3–79, 1988.

[GR 95] M. X. Goemans and V. S. Ramakrishnan, "Minimizing Submodular Functions over Families of Sets," *Combinatorica* 15, pp. 499–513, 1995.

[GT 88] A. V. Goldberg and R. E. Tarjan, "A New Approach to the Maximum Flow Problem," *J. Assoc. Comput. Mach.*, vol. 35, no. 4, pp. 921–940, 1988.

[GVY 94] N. Garg, V. V. Vazirani, and M. Yannakakis, "Multiway Cuts in Directed and Node Weighted Graphs (Extended Abstract)," in *Proc. ICALP* 1994, *LNCS* 820, pp. 487–498.

[GW 92] M. X. Goemans and D. P. Williamson, "A General Approximation Technique for Constrained Forest Problems," in *Proc. SODA* 1992. Full paper in *SIAM J. Comput.*, vol. 24, no. 2, pp. 296–317, 1995.

[GW 97] M. X. Goemans and D. P. Williamson, The Primal-Dual Method for Approximation Algorithms and its Application to Network Design Problems, in [Hoc 97], pp. 144–191.

[Gus 83] D. Gusfield, "Connectivity and Edge-Disjoint Spanning Trees," *Inform. Process. Lett.*, vol. 16, no. 2, pp. 87–89, 1983.

[Had 95] S. W. Hadley, "Approximation Techniques for Hypergraph Partitioning Problems," *Discrete Appl. Math.*, vol. 59, pp. 115–127, 1995.

[Har 93] D. Hartvigsen, “Minimum Path Bases,” *J. Algorithms*, vol. 15, no. 1, pp. 125–142, 1993.

[Har 98] D. Hartvigsen, “The Planar Multiterminal Cut Problem,” *Discrete Appl. Math.*, vol. 85, pp. 203–222, 1998.

[He 91] X. He, “An Improved Algorithm for the Planar 3-Cut Problem,” *J. Algorithms*, vol. 12, no. 1, pp. 23-37, 1991.

[Hoc 97] D. S. Hochbaum (ed.), *Approximation Algorithms for NP-hard Problems.* PWS, 1997.

[HO 94] J. Hao and J. B. Orlin, “A Faster Algorithm for Finding the Minimum Cut in a Directed Graph,” *J. Algorithms*, vol. 17, no. 3, pp. 424–446, 1994.

[HRW 92] F. K. Hwang, D. S. Richards, and P. Winter, *The Steiner Tree Problem* (Annals of Discrete Mathematics, Vol 53), North Holland, 1992.

[HS 85] D. S. Hochbaum and D. B. Shmoys, “An $O(|V|^2)$ Algorithm for the Planar 3-Cut Problem,” *SIAM J. Algebraic Discrete Methods*, vol. 6, no. 4, pp. 707–712, 1985.

[Hu 69] T. C. Hu, *Integer Programming and Network Flows.* Addison-Welsey Publishing Co., Reading, MA, 1969.

[Iba 87] T. Ibaraki, *Enumerative Approaches to Combinatorial Optimization.* Annals of Operations Research, Vol. 10, 11, Baltzer, Basel, 1987.

[IFF 00] S. Iwata, L. L. Fleischer, and S. Fujishige, “A Combinatorial, Strongly Polynomial Time Algorithm for Minimizing Submodular Functions,” in *Proc. STOC* 2000, pp. 97–106.

[Jain 98] K. Jain, “A Factor 2 Approximation Algorithm for the Generalized Steiner Network Problem,” in *Proc. FOCS* 1998. Full paper in *Combinatorica*, vol. 21, no. 1, pp. 39–60, 2001.

[JMVW 99] K. Jain, I. Măndoiu, V. V. Vazirani, and D. P. Williamson, “A Primal-dual Schema Based Approximation Algorithm for the Element Connectivity Problem,” in *Proc. SODA* 1999, pp. 484–489.

[Joh 74] D. S. Johnson, “Approximation Algorithms for Combinatorial Problems,” *J. Comput. System Sci.*, 9, pp. 256–278, 1974.

[JRT 95] M. Jünger, G. Reinelt, and S. Thienel, Practical Problem Solving with Cutting Plane Algorithms in Combinatorial Optimization, in W. Cook, L. Lovász and P. Seymour (eds.), *Combinatorial Optimization.* AMS, pp. 111–152, 1995.

[JRT 00] M. Jünger, G. Reinelt, and S. Thienel, "Practical Performance of Efficient Minimum Cut Algorithms," *Algorithmica*, vol. 26, no. 1, pp. 172–195, 2000.

[Kap 96] S. Kapoor, "On Minimum 3-cuts and Approximating k-cuts Using Cut Trees," in *Proc. IPCO* 1996, *LNCS* 1084, pp. 132–146.

[Kar 96] D. R. Karger, "Minimum Cuts in Near-Linear Time," in *Proc. STOC* 1996. Full paper in *J. ACM*, vol. 47, no. 1, pp. 46–76. 2000.

[Karp 72] R. M. Karp, Reducibility among Combinatorial Problems, in R. E. Miller, J. W. Thatcher (eds.), *Complexity of Computer Computations.* Plenum Press, New York, pp. 85–103, 1972.

[Khu 97] S. Khuller, Approximation Algorithms for Finding Highly Connected Subgraphs, in [Hoc 97], pp. 236–265.

[KK 98] David R. Kosiur and Dave Kosiur, *IP Multicasting: the Complete Guide to Interactive Corporate Networks.* John Wiley & Sons, 1998.

[KKSTY 99] D. R. Karger, P. Klein, C. Stein, M. Thorup, and N. E. Young, "Rounding Algorithms for a Geometric Embedding of Minimum Multiway Cut," in *Proc. STOC* 1999, pp. 668–678.

[KR 93] P. Klein and R. Ravi, "A Nearly Best-Possible Approximation Algorithm for Node-Weighted Steiner Trees," in *Proc. IPCO* 1993. Full paper in *J. Algorithms*, vol. 19, no. 1, pp. 104–115, 1995.

[KR 97] S. Khuller and B. Raghavachari, "Improved Approximation Algorithms for Uniform Connectivity Problems," *J. Algorithms*, vol. 21, no. 2, pp. 434–450, 1997.

[Kru 56] J. B. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," in *Proc. AMS* 7, 1956, pp. 48–50.

[KS 96] D. R. Karger and C. Stein, "A New Approach to the Minimum Cut Problems," *J. ACM*, vol. 43, no. 4, pp. 601–640, 1996.

[KV 00] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms.* Springer-Verlag, 2000.

[KW 96] R. Klimmek and F. Wagner, "A Simple Hypergraph Min Cut Algorithm," Technical Report b 96-02, Freie Universität Berlin, 1996.

[KWY 97] Y. Kamidoi, S. Wakabayasi, and N. Yoshida, "A New Approach to the Minimum k-way Partition Problem for Weighted Graphs," *Technical Report of IEICE COMP* 97-25, pp. 25–32, 1997.

[KWY 02] Y. Kamidoi, S. Wakabayasi, and N. Yoshida, "A Divide-and-Conquer Approach to the Minimum k -Way Cut Problem," *Algorithmica*, vol. 32, no. 2, pp. 262–276, 2002.

[Law 62] E. L. Lawler, "Electrical Assemblies with a Minimum Number of Interconnections," *IEEE Trans. Electronic Computers*, EC-11, pp. 781–782, 1962.

[Law 73] E. L. Lawler, "Cutsets and Partitions of Hypergraphs," *Networks*, vol. 3, pp. 275–285, 1973.

[LB 96] A. Lucena and J. E. Beasley, Branch and Cut Algorithms, in J. E. Beasley (ed.), *Advances in Linear and Integer Programming.* Oxford Univ. Press, pp. 187–221, 1996.

[Len 90] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout.* Wiley, New York, 1990.

[Lev 00] M. S. Levine, "Fast Randomized Algorithms for Computing Minimum $\{3,4,5,6\}$-Way Cuts," in *Proc. SODA* 2000, pp. 735–742.

[LKP 90] C. H. Lee, M. Kim, and C. I. Park, "An Efficient k-way Graph Partitioning Algorithm for Task Allocation in Parallel Computing Systems," in *Proc. IEEE Int. Conf. on Computer-Aided Design*, 1990, pp. 748–751.

[Lov 83] L. Lovász, Submodular Functions and Convexity, in A. Bachem, M. Grötschel and B. Korte (eds.), *Mathematical Programming: the State of the Art*, Springer, Berlin, pp. 235–257, 1983.

[Mat 93] D. W. Matula, "A Linear Time $2+\epsilon$ Approximation Algorithm for Edge Connectivity," in *Proc. SODA* 1993, pp. 500–504.

[Men 27] K. Menger, "Zur allgemeinen Kurventheorie," *Fund. Math.*, 10, pp. 96–115, 1927.

[MNI 93] N. Maeda, H. Nagamochi, and T. Ibaraki, "Approximate Algorithms for Multiway Objective Point Split Problems of Graphs" (in Japanese), Computing Devices and Algorithms (Kyoto, 1993). *Surikaisekikenkyusho Kokyuroku* no. 833, pp. 98–109, 1993.

[MSDM 96] M. Mihail, D. Shallcross, N. Dean and M. Mostrel, "A Commercial Application of Survivable Network Design: ITP/INPLANS CCS Network Topology Analyzer," in *Proc. SODA* 1996, pp. 279–287.

[Nar 91] H. Narayanan, "The Principal Lattice of Partitions of a Submodular Function," *Linear Algebra Appl.*, vol. 144, pp. 179–216, 1991.

[Nar 97] H. Narayanan, *Submodular Functions and Electrical Networks* (Annals of discrete mathematics 54). Elsevier, 1997.

[NI 92] H. Nagamochi and T. Ibaraki, "Computing edge connectivity in multigraphs and capacitated graphs," *SIAM J. Discrete Math.*, vol. 5, no. 1, pp. 54–66, 1992.

[NI 98] H. Nagamochi and T. Ibaraki, "A Note on Minimizing Submodular Functions," *Inform. Process. Lett.*, 67, pp. 239–244, 1998.

[NI 00] H. Nagamochi and T. Ibaraki, "A Fast Algorithm for Computing Minimum 3-way and 4-way Cuts," *Math. Program.*, 88, no. 3, pp. 507–520, 2000.

[NKI 00] H. Nagamochi, S. Katayama, and T. Ibaraki, "A Faster Algorithm for Computing Minimum 5-way and 6-way Cuts in Graphs," *J. Combi. Optim.*, vol. 4, pp. 35–78, 2000.

[NR 01] J. Naor and Y. Rabani, "Tree Packing and Approximating k-Cuts," in *Proc. SODA* 2001, pp. 26–27.

[NRP 96] H. Narayanan, S. Roy, and S. Patkar, "Approximation Algorithms for Min-k-Overlap Problems Using the Principal Lattice of Partitions Approach," *J. Algorithms*, vol. 21, no. 2, pp. 306–330, 1996.

[NZ 97] J. Naor and L. Zosin, "A 2-Approximation Algorithm for the Directed Multiway Cut Problem," in *Proc. FOCS* 1997. Full paper in *SIAM J. Comput.*, vol. 31, no. 2, pp. 477–482. 2001.

[NW 61] C. St. J. A. Nash-Williams, "Edge-disjoint Spanning Trees of Finite Graphs," *J. London Math. Soc.*, 36, pp. 445–450, 1961.

[OL 96] I. H. Osman and G. Laporte, Meta-heuristics: a bibliography. *Ann. Oper. Res.* 63, pp. 513–623, 1996.

[Pap 94] C. H. Paradimitriou, *Computational Complexity.* Addison-Wesley, Reading, MA, 1994.

[Pul 82] W. R. Pulleyblank, presentation at SIAM Meeting on Optimization, MIT, Boston, 1982.

[Que 93] M. Queyranne, “A Gomory-Hu Tree for Symmetric Submodular Functions,” unpublished manuscript, November 1993.

[Que 95] M. Queyranne, “Minimizing Symmetric Submodular Functions,” in *Proc. SODA* 1995. Full paper in *Math. Program.*, vol. 82 no. 1–2, Ser. B, pp. 3–12, 1998.

[Que 99] M. Queyranne, “On Optimum Size-Constrained Set Partitions,” AUSSOIS 1999, France, March 8–12, 1999.

[Ree 93] C. R. Reeves (ed.), *Modern Heuristic Techniques for Combinatorial Problems.* Blackwell Scientific Publications, Oxford, 1993.

[RW 95] R. Ravi and D. P. Williamson, “An Approximation Algorithm for Minimum-Cost Vertex-Connectivity Problems,” in *Proc. SODA* 1995. Full paper in *Algorithmica*, vol. 18, no. 1, pp. 21–43, 1997.

[RW 01] R. Ravi and D. P. Williamson, “Erratum: An Approximation Algorithm for Minimum-Cost Vertex-Connectivity Problems,” manuscript, July 2001.

[Sch 98] A. Schrijver, *Theory of Linear and Integer Programming.* John Wiley & Sons, 1998.

[Sch 00] A. Schrijver, “A Combinatorial Algorithm Minimizing Submodular Functions in Strongly Polynomial Time,” *J. Combin. Theory* B 80, no. 2, pp. 346–355, 2000.

[Ston 77] H. S. Stone, “Multiprocessor Scheduling with the Aid of Network Flow Algorithms,” *IEEE Trans. on Software Engg.*, SE-3, pp. 85–93, 1977.

[Stoe 92] M. Stoer, *Design of Survivable Networks.* Lecture Notes Math. 1531, Springer, Heidelberg, 1992.

[SV 91] H. Saran and V. V. Vazirani, “Finding k-cuts within Twice the Optimal,” in *Proc. FOCS* 1991. Full paper in *SIAM J. Comput.*, vol. 24, no. 1, pp. 101–108, 1995.

[SW 94] M. Stoer and F. Wagner, “A Simple Min-cut Algorithm,” in *Proc. 2nd Annual European Symposium on Algorithms*, 1994. Full paper in *J. ACM*, vol. 44, no. 4, pp. 585–591, 1997.

[Tit 99] P. Tittmann, "Partitions and Network Reliability," *Discrete Appl. Math.*, vol. 95, pp. 445–453, 1999.

[TFW 99] K. Takeshita, T. Fujito, and T. Watanabe, "On Primal-Dual Approximation Algorithms for Several Hypergraph Problems" (in Japanese). IPSJ *Mathematical Modeling and Problem Solving* no. 23-3, pp. 13–18, 1999.

[Tut 61] W. T. Tutte, "On the Problem of Decomposing a Graph into n Connected Factors," *J. London Math. Soc.*, 36, pp. 221–230, 1961.

[TZ 94] I. Tomescu and M. Zimand, "Minimum Spanning Hypertrees," *Discrete Appl. Math.*, vol. 54, no. 1, pp. 67–76, 1994.

[War 98] D. M. Warme, *Spanning Trees in Hypergraphs with Applications to Steiner Trees.* PhD Dissertation, Univ. Virginia, Dept. Computer Science, 1998.

[Wil 95] D. P. Williamson, Computational Experience, in a minicourse on approximation algorithms at IBM Almaden, 1995.

[Wil 01] D. P. Williamson, "The Primal-Dual Method for Approximation Algorithms," to appear in *Math. Program.*

[WGMV 93] D. P. Williamson, M. X. Goemans, M. Mihail, and V. V. Vazirani, "A Primal-dual Approximation Algorithm for Generalized Steiner Network Problems," in *Proc. STOC* 1993. Full paper in *Combinatorica* 15, pp. 435–454, 1995.

[Yeh 01] W. C. Yeh, "A Simple Algorithm for the Planar Multiway Cut Problem". *J. Algorithms*, vol. 39, no. 1, pp. 68–77, 2001.

[ZNI 99] L. Zhao, H. Nagamochi, and T. Ibaraki, "Approximating the Minimum k-way Cut in a Graph via Minimum 3-way Cuts," in *Proc. ISAAC* 1999. Full paper in *J. Comb. Optim.*, vol. 5, no. 4, pp. 397–410, 2001.

[ZNI 01a] L. Zhao, H. Nagamochi, and T. Ibaraki, "A Primal-Dual Approximation Algorithm for the Survivable Network Design Problem in Hypergraphs," in *Proc. STACS* 2001, *LNCS* 2010, pp. 478–489. Full paper accepted for publication in *Discrete Appl. Math.*

[ZNI 01b] L. Zhao, H. Nagamochi, and T. Ibaraki, "A Note on Approximating the Survivable Network Design Problem in Hypergraphs," *IEICE Trans. Inf. & Syst.*, vol. E85-D, no. 2, 2002 (to appear).

[ZNI 01c] L. Zhao, H. Nagamochi, and T. Ibaraki, "A Unified Framework for Approximating Multiway Partition Problems (Extended Abstract)," in *Proc. ISAAC* 2001, *LNCS* 2223, pp. 682–694.

Index

Δ ... 74
δ ... 10
Γ ... 10
$\sigma(G)$ ... 6
$G[S]$ ... 2
$H[S]$ ... 10
k-partition ... 4
k-PPH-T1 ... 12, 23, 99, 100
k-PPH-T2 ... 12, 22, 99, 100
k-PPH-T3 ... 12, 23, 99, 100
k-PPMSS ... 22, 99, 100
k-PPSS ... 22, 99, 100
k-PPSSS ... 22, 99, 100
k-TPG ... 101
k-TPH-T1 ... 11, 12, 27, 101
k-TPH-T2 ... 12, 27, 101
k-TPH-T3 ... 12, 27, 101
k-TPMSS ... 27, 101
k-TPSS ... 27, 101
k-TPSSS ... 27, 101
k-way cut ... 5, 11
k-way cut problem ... 6
  k-PPG ... 7, 22, 99
w_{in} ... 18

approximation algorithm ... 3

cut ... 10
cut function ... 7, 12, 19
  w_{ex} ... 7, 18

graph ... 1

hyperedge disjoint paths ... 14
hypergraph ... 10

monotone ... 3
MPP-NT ... 21
multicasting problem ... 17
multiterminal cut problem ... 7
  k-MCP ... 8
multiway partition problem ... 4, 26
  MPP ... 4, 26

network design problem with requirement function (NDRF) ... 73
nonterminal disjoint paths ... 14

path ... 1, 10
performance guarantee ... 3

SNDP-G ... 15, 91, 101
SNDP-G$^+$ ... 73, 91, 101
Steiner tree problem ... 15
strength ... 6, 32
  $\sigma(G, w)$ ... 6
subgraph ... 2, 10
  induced subgraph ... 2, 10
submodular ... 3
supermodular ... 3
survivable network design problem ... 14
  SNDP ... 15, 91

target split problem ... 5
  k-TPG ... 7, 27
target-split ... 4
tree ... 2, 18

weakly supermodular ... 3