COMBINATORIAL REDUCTIONS BETWEEN GRAPH PARTITIONING BY VERTEX SEPARATOR AND HYPERGRAPH PARTITIONING PROBLEMS FOR PARALLEL AND SCIENTIFIC COMPUTING APPLICATIONS

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Enver Kayaaslan

August, 2009

ABSTRACT

COMBINATORIAL REDUCTIONS BETWEEN GRAPH PARTITIONING BY VERTEX SEPARATOR AND HYPERGRAPH PARTITIONING PROBLEMS FOR PARALLEL AND SCIENTIFIC COMPUTING APPLICATIONS

Enver Kayaaslan
M.S. in Computer Engineering
Supervisor: Prof. Dr. Cevdet Aykanat
August, 2009

Hypergraph Partitioning (HP) and Graph Partitioning by Vertex Separator (GPVS) problems are very well known problems which are used in scientific and parallel computing effectively. A typical problem in parallel computing is to partition the data/tasks into several processors such that the overall performance of the computation gets more qualified in terms of time and/or memory. Besides, GPVS is generally used for fill-reducing ordering of sparse matrices for solving sparse linear systems efficiently which lies in the area of scientific computing. In this thesis, the relation between these two problems, HP and GPVS problems, are investigated. Two combinatorial reductions, from HP Problem to GPVS Problem and from GPVS Problem to HP Problem are disclosed along with their theoretical bases. In practice, the nontrivial part of HP Problem to GPVS Problem reduction is the input transformation, that is, converting a graph to a hypergraph such that the reduction holds. The nontrivial part of the reduction from GPVS Problem to HP Problem is the output transformation, that is, decoding a vertex separator of the corresponding graph to a partition for the hypergraph. In this part, some useful impossibility results are derived. These nontrivial parts are investigated deeply and effective and efficient algorithms and methods are proposed.

Furthermore, "oPaToH", an HP-based fill-reducing ordering tool based on PaToH, is enhanced along with implementation of input transformations. Besides, based on fill-reducing ordering tool onmetis, a GPVS-based HP tool "hpmetis" is derived and a Dantzig-Wolfe decomposition tool for efficient parallelizm of linear programming problem solutions is constructed, which is called as "dwmetis". The fill-reducing ordering results obtained with enhanced oPaToH produced more qualified ordering results such as up to %20 improvements for operation count compared to state-of-the

art ordering tools such as onmetis. Note that decreasing operation count relates to performing sparse linear equation solutions faster. The Dantzig-Wolfe decomposition results with dwmetis produced results around 5 times faster than the state-of-the art hypergraph partitioning tool PaToH with comparable quality for net balancing. This is also valuable because the preprocessing overhead is also considered inside the total execution time, generally. As a result, in this work it is showed that parallel and scientific computing applications can be performed faster by exploiting the combinatorial reductions between HP problem and GPVS problem.

Keywords: hypergraph partitioning, graph partitioning by vertex separator, combinatorial scientific computing, parallel computing, combinatorial reduction, fill reducing ordering, Dantzig-Wolfe decomposition, singly-bordered block diagonal form.

ÖZET

PARALEL VE BİLİMSEL HESAPLAMA UYGULAMALARI İÇİN HİPERÇİZGE BÖLÜMLEME VE DÜĞÜM AYIRACI İLE ÇİZGE BÖLÜMLEME PROBLEMLERİNİN BİRBİRİNE KOMBİNATORİYAL DÖNÜŞTÜRÜMÜ

Enver Kayaaslan
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Prof. Dr. Cevdet Aykanat
Ağustos, 2009

Hiperçizge Bölümleme (HB) ve Düğüm Ayıracı ile Çizge Bölümleme (DACB) problemleri literatürde gayet bilinen, koşut ve bilimsel hesaplamada etkin biçimde kullanılan problemlerdir. Koşut hesaplamalarda tipik bir problem, verilerin ve görevlerin değişik sayıda işlemciye öyle şekilde dağıtılmasıdır ki hesaplamanın çalışma performansı zaman ve yer gereksinimleri açısından hızlı ve verimli olsun. Bunun yanında, DACB problemi seyrek doğrusal sistemlerin verimli çözülebilmesi için doluluk azaltan sıralama yapmak için etkin biçimde kullanılmaktadır. Bu da bilimsel hesaplamanın konu sahası içine girmektedir. Bu tez calışmasında, HB ve DACB problemleri arasındaki ilişki incelenmektedir. Bu bağlamda, iki kombinatoriyal dönüştürüm açığa çıkarılmıştır. Birinci dönüştürme, HB probleminden DACB problemine, ikincisi ise DACB probleminden HB probleminedir. HB probleminden DACB problemine dönüştürmede girdi dönüşümü kolay olmamaktadır. Girdi dönüşümünden kasıt, verilen bir çizgeyi, problem dönüştürümü uygun olacak şekilde bir hiperçizgeye dönüştürmektir. DACB probleminden HB problemine dönüştürümde ise çıktı dönüsümü kolay olmamaktadır. Burada da kasıt, verilen bir çizge bölümlemeyi asıl problemimiz olan hiperçizge bölümlemeye dönüştürmektir. Bu kısımda önemli ve faydalı imkansızlık sonuçları türetilmiştir. Bu kolay olmayan kısımlar derinliğince incelenmiş, etkili ve verimli çözümler ve yöntemler önerilmiştir.

Bu çalışma çerçevesinde, bir HB tabanlı doluluk azaltan sıralama aracı olan oPaToH, gelişmiş girdi dönüşümleri ile genişletilmiştir. Bunun yanında, başka bir doluluk azaltan sıralama aracı olan onmetis kullanılarak DACB tabanlı bir HB

aracı olan “hpmetis” üretilmiştir. Ayrıca, yine onmetis kullanılarak bir Dantzig-Wolfe ayrıştırma aracı olan “dwmetis” üretilmiştir. Dantzig-Wolfe ayrıştırma ise doğrusal problemlerin etkin koşutlanmasinda kullanılmaktadır. Deneysel sonuçlarımız gösterdi ki, genişletilen oPaToH, seyrek doğrusal sistem çözümünde hali hazırdaki iyi yöntemlere göre %20'lere varan iyileşme göstermiştir. Üretilen Dantzig-Wolfe ayrıştırma aracı dwmetis ise hali hazırda kullanılan HB tabanlı yöntemlere göre makul kalite farklılıklarında 5 kata kadar hızlı sonuç üretebilmiştir. Bu sonuç da gayet değerlidir, çünkü toplam performans ölçülürken öncalışma zamanı da değer taşımaktadır. Sonuç olarak, bu çalışmada gösterildi ki koşut ve bilimsel hesaplama işlemlerinin, HB ve DACB problemlerini birbirlerine dönüştürümü kullanılarak daha hızlı çalışmaları sağlanabilmektedir.

Anahtar sözcükler: hiperçizge bölümleme, düğüm ayıracı ile çizge bölümleme, kombinatoriyal bilimsel hesaplama, koşut hesaplama, kombinatoriyal dönüştürüm, doluluk azaltan sıralama, Dantzig-Wolfe ayrıştırma.

Acknowledgement

I would like to express my highest gratitude to my supervisor Prof. Dr. Cevdet Aykanat for his guidance, suggestions, and encouragement to my research as being at the beginning steps of my academic life.

I would also like to thank Ümit Çatalyürek for his valuable contributions and tools at the development of the thesis. I also want to thank B. Barla Cambazoğlu for his hospitality in Barcelona and academic discussions on some other research. Finally, I also want to thank Bora Uçar for his valuable contributions on some other research.

I am thankful to valuable professors, Asst. Prof. Dr. Ali Aydın Selçuk, who is also in my thesis committee, and Prof. Dr. Yavuz Oruç for their lovely encourgement to me. I am also grateful to Assoc. Prof. Dr. Emre Alper Yıldırım for reading and commenting on the thesis and giving a very enjoyable lecture on Approximation Algorithms.

I am grateful to my friends and my relatives for their infinite moral support, especially to my research group members: Kadir Akbudak, A.Aylin Tokuç, Ata Türk, Tayfun Küçükyılmaz, Volkan Yazıcı and Reha Oğuz Selvitopi.

I want also thank to the koaches Fatih Deniz, Erkay Uzun and Yunus Emre Aslan for their extraordinary encourgements along with the other koaches. I owe special thanks to my friends Abdullah Bülbül and Erkan Okuyan for sharing their rooms and infinite hospitalities with me in different times.

Finally, I thank TÜBİTAK for its support troughout my master program.

Anneme, Babama ve Abime...

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction and Related Work

Two problems constitute the main ground of the thesis: Hypergraph Partitioning (HP) Problem and Graph Partitioning by Vertex Separator (GPVS) Problem. These problems are well known and commonly used for modeling realistic problems encountered in parallel and scientific computing. Among the graph partitioning problems the Graph Partitioning by Edge Separator (GPES) Problem is more studied. In GPES, the problem is to partition the vertices into a specified number of parts minimizing the total weight of the edges that are shared by two different parts such that the total weights of the vertices in each part are balanced. On the other hand, in the GPVS problem, the separator is formed by vertices instead of edges and the objective is to find a set of vertices, which forms the separator, along with a partitioning of the non-separator vertices to a specified number of parts while minimizing the total weight of the vertices in the separator such that no two vertices in different parts have adjacency in the graph and again the total weights of the vertices in each part are balanced. Recall that in graphs the edges connect exactly two vertices. Hypergraphs can be considered as generalized versions of graphs, where edges can connect arbitrary number of vertices. Here, the vertices of hypergraphs will be referred as nodes, and the edges of hypergraphs will be referred as nets. The HP problem is to partition the nodes while minimizing the total weight of the nets that are shared by at least two different parts such that the total weights of either nodes or nets in each part are balanced. In this work, we generally refer to the HP problem in which the balance on the total weights of the nets in each

part is considered.

There are several applications that are effectively solved by using HP and GPVS problems. In this work, we present combinatorial reductions between these two problems. That is we present a solution for the GPVS problem that utilizes solutions for HP problem by converting the GPVS problem into the HP problem. We also present a solution for the HP problem that utilizes solutions for GPVS problem by converting the HP problem into the GPVS problem. This enables us to use solutions proposed for one problem in solving applications that are being solved by the solutions of the other problem. For example, we can use HP solutions for an application which is used to be solved by GPVS, or vice versa.

The work in this thesis can be considered as the generalization of two previous works: Hypergraph Partitioning-Based Sparse Matrix Ordering [10] by Catalyurek as a chapter in his PhD thesis in 1999 and Decomposing Linear Programs For Parallel Solutions [42] which also has been his master thesis in 1998. However, these works were application-oriented works. Catalyurek used combinatorial reduction from GPVS problem to HP problem to solve fill-reducing problem for efficient sparse linear system solutions. Lieserson and Lewis also studied GPVS solution by using HP problem [35], however in their formulation the HP solution and the GPVS solution was quite loosely related. On the other hand, Pinar used combinatorial reduction from HP problem to GPVS problem to solve the problem of finding Dantzig-Wolfe decomposition of sparse matrix for parallel solutions of LP Problems. Actually, the idea of GPVS-based HP was earlier given by Cong et al. [13, 14] under the circuit partitioning problem. However these works tried to solve HP problem by using GPES. Catalyurek, in his master thesis [9], also studied solving HP problem by using minimum degree ordering which is used to solve GPVS problem. However, Catalyurek also didn't mention about GPVS. Yet to my knowledge, Pinar's work is the first work, which pronounces "GPVS" concretely, to solve HP problem.

The contribution of this thesis can be considered as three-fold:

1) *Generalizing the combinatorial reductions between HP and GPVS problems for*

generic use: The GPVS-based HP idea was specialized under circuit partitioning and Dantzig-Wolfe decomposition, whereas the HP-based GPVS idea was specialized under fill-reducing ordering. In this work, these ideas are generalized for generic use. Some theoretical possibility and impossibility results are presented for the reductions.

2) *Proposing effective and/or efficient algorithms for input transformation of GPVS to HP reduction and output transformation of HP to GPVS reduction:* The nontrivial part of the GPVS to HP reduction is its input transformation. For a given graph, we need to find a hypergraph such that the reduction holds. New algorithms are proposed to handle this along with sound theoretical grounds. Similarly, the nontrivial part of the HP to GPVS reduction is its output transformation. For a given graph partition by vertex separator and a hypergraph, we need to find a hypergraph partition which induces given vertex separator. New algorithms and methods are proposed to handle this along with some theoretical impossibility results.

3) *Producing and enhancing useful tools:*

- *hpmetis*, which is a GPVS-based hypergraph partitioning with net balancing tool, is *derived* from the state-of-the-art fill-reducing ordering tool onmetis [32]
- *dwmetis*, which is a Dantzig-Wolfe decomposition tool, is *derived* from the state-of-the-art fill-reducing ordering tool onmetis [32]
- *oPaToH* [11, 10], which is a fill-reducing ordering tool produced by Catalyurek based on the state-of-the-art HP tool PaToH [11], is *enhanced* via importing extended input transformations

Chapter 2

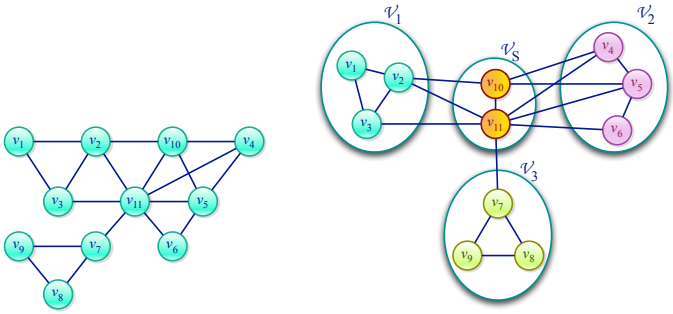
Preliminaries

2.1 Graph Partitioning by Vertex Separator (GPVS)

2.1.1 GPVS Problem

An undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined as a set $\mathcal{V}$ of vertices and set $\mathcal{E}$ of edges. Every edge $e_{ij} \in \mathcal{E}$ connects a pair of distinct vertices v_i and v_j. $Adj_{\mathcal{G}}(v_i)$ is used to denote the set of vertices that are adjacent to vertex v_i in graph $\mathcal{G}$. This operator is also extended to include the adjacency set of a vertex subset $\mathcal{V}' \subseteq \mathcal{V}$, i.e., $Adj_{\mathcal{G}}(\mathcal{V}') = \{v_j \in \mathcal{V} - \mathcal{V}' : v_j \in Adj_{\mathcal{G}}(v_i) \text{ for some } v_i \in \mathcal{V}'\}$. The degree d_i of a vertex v_i is equal to the number of edges incident to v_i, i.e., $d_i = |Adj_{\mathcal{G}}(v_i)|$. A vertex subset $\mathcal{V}_S$ is a K-way *vertex separator* if the subgraph induced by the vertices in $\mathcal{V} - \mathcal{V}_S$ has at least K connected components. Let w_i denote the weight of the vertex $v_i \in \mathcal{V}$.

$\Pi_{VS} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K; \mathcal{V}_S\}$ is a K-way vertex partition of $\mathcal{G}$ by vertex separator. $\mathcal{V}_S \subseteq \mathcal{V}$ if the following conditions hold: $\mathcal{V}_k \subseteq \mathcal{V}$ and $\mathcal{V}_k \neq \emptyset$ for $1 \leq k \leq K$; $\mathcal{V}_k \cap \mathcal{V}_\ell = \emptyset$ for $1 \leq k \leq \ell \leq K$ and $\mathcal{V}_k \cap \mathcal{V}_S = \emptyset$ for $1 \leq k \leq K$; $\bigcup_{k=1}^{K} \mathcal{V}_k \cup \mathcal{V}_S = \mathcal{V}$; removal of $\mathcal{V}_S$ gives K disconnected parts $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K$ (i.e., $Adj_{\mathcal{G}}(\mathcal{V}_k) \subseteq \mathcal{V}_S$ for $1 \leq k \leq K$). A vertex $v_i \in \mathcal{V}_k$ is said to be a boundary vertex of part $\mathcal{V}_k$ if it is adjacent to a vertex in $\mathcal{V}_S$. In a partition Π_{VS}, a vertex that has at least one neighbor in a part is said to *connect*



(a) (b)

Figure 2.1: (a) An example graph $\mathcal{G}$ (b) An example 3-way vertex separator Π_{VS} of $\mathcal{G}$

that part. *Connectivity set* Λ_i of a vertex v_i is defined as the set of parts connected by v_i. *Connectivity* $\lambda_i = |\Lambda_i|$ of a vertex v_i denotes the number of parts connected by v_i. A vertex separator is said to be *narrow* if no subset of it forms a separator, and *wide* otherwise.

Figure 2.1(a) shows an example graph. Figure 2.1(b) presents an example vertex separator on the graph given in Figure 2.1(a).

In the GPVS problem, the partitioning constraint is to maintain a ε-$balance$ on the weights of the K parts of the K-way vertex partition $\Pi_{VS} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K; \mathcal{V}_S\}$. The weight W_k of a part $\mathcal{V}_k$ is defined by the sum of the individual weights of vertices in $\mathcal{V}_k$, i.e., $W_k = \sum_{v_i \in \mathcal{V}_k} w_i$, for $1 \leq k \leq K$. A partition Π_{VS} is said to satisfy ε-$balance$ if the following equation holds:

$$W_k \leq W_{avg}(1+\varepsilon), \quad for\ 1 \leq k \leq K \tag{2.1}$$

where $W_{avg} = \sum_{k=1}^{K} W_k/K$. The basic imbalance ε_0 of a vertex separator Π_{VS} is defined as the minimum value of ε that Π_{VS} has ε-balance. The partitioning objective is to minimize the separator size, which can be defined as the total weight of vertices in the separator, i.e.,

$$Separatorsize(\Pi_{VS}) = \sum_{v_i \in \mathcal{V}_S} w_i. \tag{2.2}$$

Problem 1 *GPVS Problem: Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, an imbalance parameter ε and an integer K, find a K-way partition Π_{VS} of $\mathcal{G}$ by vertex separator,*

minimizing the $Separatorsize(\Pi_{VS})$ *such that* Π_{VS} *has* ε*-balance.*

The version of the GPVS problem in which the separator is restricted to be a *narrow*.separator is referred to here as the *narrow* GPVS (GPVSn) problem.

Problem 2 *GPVSn Problem: Given an undirected graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$*, an imbalance parameter* ε *and an integer* K*, find a* K*-way partition* Π_{VS} *of* $\mathcal{G}$ *by narrow vertex separator, minimizing the* $Separatorsize(\Pi_{VS})$ *such that* Π_{VS} *has* ε*-balance.*

The GPVS problem is known to be NP-hard [6].

2.1.2 Matrix-Theoretic View for GPVS

Given a $p \times p$ symmetric and square matrix M, let $\mathcal{G}(M) = (\mathcal{V}, \mathcal{E})$ denote the standard graph representation of matrix M. Therefore, the M matrix is the adjacency matrix of a given graph $\mathcal{G} = \mathcal{G}(M)$.

$$M_{DB} = PAP^T = \begin{bmatrix} M_1 & & & & M_{B_1}^T \\ & M_2 & & & M_{B_2}^T \\ & & \ddots & & \vdots \\ & & & M_K & M_{B_K}^T \\ M_{B_1} & M_{B_2} & \dots & M_{B_K} & M_B \end{bmatrix}. \tag{2.3}$$

A K-way GPVS $\Pi_{VS} = \{\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_K; \mathcal{V}_S\}$ of $\mathrm{G}(M)$ can be decoded as permuting matrix M into a doubly-bordered block diagonal (DB) form $M_{DB} = PAP^T$ as follows: Π_{VS} is used to define the partial row/column permutation matrix P by permuting the rows/columns corresponding to the vertices of $\mathcal{V}_k$ after those corresponding to the vertices of $\mathcal{V}_{k-1}$ for $2 \leq k \leq K$, and permuting the rows/columns corresponding to the separator vertices to the end. The partitioning objective of minimizing the separator size of Π_{VS} corresponds to minimizing the number of coupling rows/columns in M_{DB}, whereas the partitioning constraint of maintaining balance on the part weights

of Π_{VS} infers balance among the row/columns counts of the square diagonal submatrices in M_{DB}. Figure 2.1.2 shows the matrix view of a 3-way vertex separator Π_{VS} given in Figure 2.1(b) of the graph $\mathcal{G}$ presented in Figure 2.1(a).

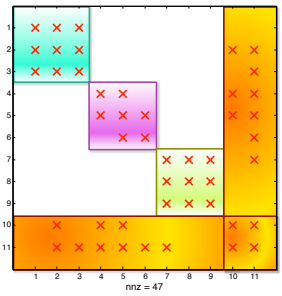
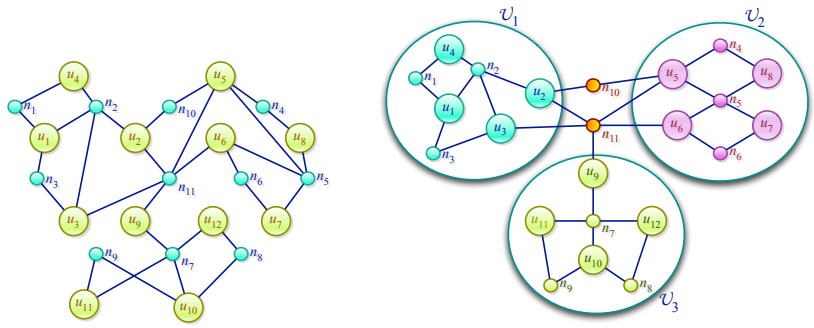


Figure 2.2: The matrix representation M_{DB} of the graph $\mathcal{G}$ given in Fig. 2.1(a) induced by 3-way vertex separator Π_{VS} given in Fig. 2.1(b)

2.2 Hypergraph Partitioning (HP)

2.2.1 HP Problem

A hypergraph $\mathcal{H} = (\mathcal{U}, \mathcal{N})$ is defined as a set $\mathcal{U}$ of nodes (vertices) and a set $\mathcal{N}$ of nets (hyperedges). We refer to the vertices of $\mathcal{H}$ as nodes, to avoid the confusion between graphs and hypergraphs. Every net $n_i \in \mathcal{N}$ connects a subset of nodes of $\mathcal{U}$, which are called as the pins of n_i and denoted as $Pins(n_i)$. The set of nets that connect node u_h is denoted as $Nets(u_h)$. Two distinct nets n_i and n_j are said to be adjacent, if they connect at least one common node. We use the notation $Adj_{\mathcal{H}}(n_i)$ to denote the set of nets that are adjacent to n_i in $\mathcal{H}$, i.e., $Adj_{\mathcal{H}}(n_i) = \{n_j \in \mathcal{N} - \{n_i\} : Pins(n_i) \cap Pins(n_j) \neq \emptyset\}$. We extend this operator to include the adjacency set of a net subset $\mathcal{N}' \subseteq \mathcal{N}$, i.e., $Adj_{\mathcal{H}}(\mathcal{N}') = \{n_i \in \mathcal{N} - \mathcal{N}' : n_i \in Adj_{\mathcal{H}}(n_j)$ for some $n_j \in \mathcal{N}'\}$. The degree d_h of a node u_h is equal to the number of nets that connect u_h, i.e., $d_h = |Nets(u_h)|$. The size s_i of a net n_i is equal to the number of its pins, i.e., $s_i = |Pins(n_i)|$. Let w_i and c_j denote the weight of vertex $u_i \in \mathcal{U}$ and cost of net $n_j \in \mathcal{N}$, respectively.



(a) (b)

Figure 2.3: (a) An example hypergraph $\mathcal{H}$ (b) An example 3-way hypergraph partition Π_{HP} of $\mathcal{H}$

$\Pi_{HP} = \{\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_K\}$ is a K-way node partition of $\mathcal{H}$ if the following conditions hold: $\mathcal{U}_k \subseteq \mathcal{U}$ and $\mathcal{U}_k \neq \emptyset$ for $1 \leq k \leq K$; $\mathcal{U}_k \cap \mathcal{U}_\ell = \emptyset$ for $1 \leq k < \ell \leq K$; $\bigcup_{k=1}^{K} \mathcal{U}_k = \mathcal{U}$. In a partition Π_{HP} of $\mathcal{H}$, a net that connects at least one node in a part is said to *connect* that part. *Connectivity set* Λ_j of a net n_j is defined as the set of parts connected by n_j. *Connectivity* $\lambda_j = |\Lambda_j|$ of a net n_j denotes the number of parts connected by n_j. A net n_i is said to be an internal net of a node-part $\mathcal{U}_k$, if it connects only part $\mathcal{U}_k$, i.e., $Pins(n_i) \subseteq \mathcal{U}_k$. We use $\mathcal{N}_k$ to denote the set of internal nets of node-part $\mathcal{U}_k$, for $1 \leq k \leq K$. A net n_j is said to be cut (external), if it connects more than one node part. We use $\mathcal{N}_S$ to denote the set of external nets, to show that it actually forms a net separator, that is, removal of $\mathcal{N}_S$ gives at least K disconnected parts. As a corollary, a K-way hypergraph partition can be considered as K-way node-partition which induces a $(K+1)$-way net-partition $\Pi_{HP} = \mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K; \mathcal{N}_S$.

Figure 2.3(a) shows an example hypergraph. Figure 2.3(b) presents an example hypergraph partition on the graph given in Figure 2.3(a).

A node u_i is said to be a *free* node of a partition Π_{HP} of $\mathcal{H}$ if all nets of u_i lies in the cut, i.e., $Nets(u_i) \subseteq \mathcal{N}_S$. The set of *free* nodes $\mathcal{U}_F$ of a Π_{HP} of $\mathcal{H}$ is defined as the set of all free nodes of Π_{HP} in $\mathcal{H}$, i.e., $\mathcal{U}_F = \mathcal{U} - \cup_{n_j \notin \mathcal{N}_S} Pins(n_j)$.

In the HP problem, the partitioning constraint is to maintain a ε-*balance* on the weights of the K parts of the K-way node partition $\Pi_{HP} = \{\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_K\}$. The weight W_k of a part $\mathcal{U}_k$ can be either defined by the sum of the individual weights of

nodes in $\mathcal{U}_k$, which is called *Node Balancing*, i.e., $W_k = \sum_{u_i \in \mathcal{U}_k} w_i$, for $1 \leq k \leq K$, or defined by the sum of the individual weights of internal nets in $\mathcal{U}_k$, which is called *Net Balancing*, i.e., $W_k = \sum_{n_j \in \mathcal{N}_k} c_j$, for $1 \leq k \leq K$. A partition Π_{HP} is said to satisfy ε-$balance$ if the following equation holds:

$$W_k \leq W_{avg}(1+\varepsilon), \quad for\ 1 \leq k \leq K \tag{2.4}$$

where $W_{avg} = \sum_{k=1}^{K} W_k/K = \sum_{u_i \in \mathcal{U}} w_i/K$. The basic imbalance ε_0 of a hypergraph partition Π_{HP} is defined as the minimum value of ε that Π_{HP} has ε-balance. The partitioning objective is to minimize the cut size, which can be either defined as the total weight of external (cut) net, which is called *Cutnet* metric, i.e.,

$$Cutsize(\Pi_{HP}) = \sum_{n_j \in \mathcal{N}_S} c_j \tag{2.5}$$

or defined as the total weight of external (cut) nets multiplied along with their connectivity minus one, which is called *Connectivity* metric, i.e.,

$$Cutsize(\Pi_{HP}) = \sum_{n_j \in \mathcal{N}_S} c_j(\lambda - 1) \tag{2.6}$$

Problem 3 *HP Problem: Given a hypergraph $\mathcal{H} = (\mathcal{U}, \mathcal{N})$, an imbalance parameter ε and an integer K, find a K-way node partition Π_{HP} of $\mathcal{H}$, minimizing the $Cutsize(\Pi_{HP})$ such that Π_{HP} has ε-balance.*

The HP problem is known to be NP-hard [36].

2.2.2 Matrix-Theoretic View for HP

Given a $p \times q$ rectangular matrix A, let $\mathcal{H}_{RN}(A) = (\mathcal{U}, \mathcal{N})$ denote the row-net hypergraph representation of matrix A. In this representation, the columns and rows of matrix A respectively constitute the nodes and nets of hypergraph $\mathcal{H}_{RN}(A)$. Matrix A can also be considered as row-net matrix representation of a given hypergraph

$\mathcal{H} = \mathcal{H}_{RN}(A)$.

$$A_{SB} = PAQ = \begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_K \\ A_{B_1} & A_{B_2} & \dots & A_{B_K} \end{bmatrix}. \tag{2.7}$$

A K-way node-partition $\Pi_{HP} = \{\mathcal{U}_1, \mathcal{U}_2, \dots, \mathcal{U}_K\}$, which induces a $(K+1)$-way net partition $\{\mathcal{N}_1, \mathcal{N}_2, \dots, \mathcal{N}_K; \mathcal{N}_S\}$, of $\mathrm{H_{RN}}(A)$ can be decoded as permuting matrix A into a K-way rowwise singly-bordered block diagonal (SB) form as follows: Π_{HP} is used to define the partial column permutation matrix Q by permuting the columns corresponding to the nodes of part $\mathcal{U}_k$ after those corresponding to the nodes of part $\mathcal{U}_{k-1}$ for $2 \leq k \leq K$. The $(K+1)$-way partition on the nets of $\mathrm{H_{RN}}(A)$ is used to define the partial row permutation matrix P by permuting the rows corresponding to the nets of $\mathcal{N}_k$ after those corresponding to the nets of $\mathcal{N}_{k-1}$ for $2 \leq k \leq K$, and permuting the rows corresponding to the external nets to the end. Here, the partitioning objective of minimizing the cutsize of Π_{HP} corresponds to minimizing the number of coupling rows in A_{SB}. The partitioning constraint of balancing on the internal net counts of node parts of Π_{HP} infers balance among the row counts of the rectangular diagonal submatrices in A_{SB}. It is clear that the transpose of A_{SB} will be in a columnwise SB form. Figure 2.2.2 shows the matrix view of a 3-way hypergraph partition Π_{HP} given in Figure 2.3(b) of the hypergraph $\mathcal{H}$ presented in Figure 2.3(a).

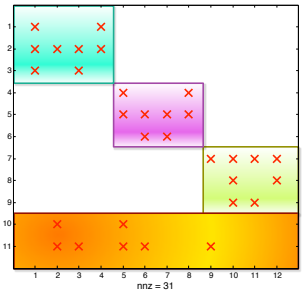
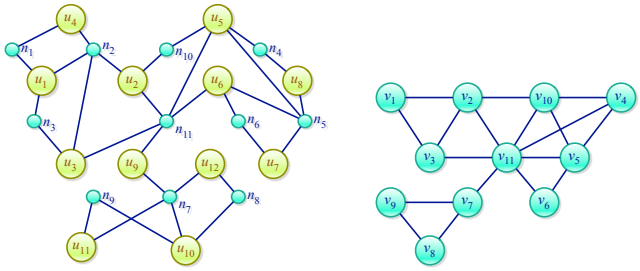


Figure 2.4: The matrix representation A_{SB} of the hypergraph $\mathcal{H}$ given in Fig. 2.3(a) induced by hypergraph partition Π_{HP} given in Fig. 2.3(b)



(a) (b)

Figure 2.5: (a) An example hypergraph $\mathcal{H}$ (b) The net intersection graph $\mathrm{NIG}(\mathcal{H})$ of $\mathcal{H}$

2.3 Net Intersection Graph (NIG) of Hypergraph

2.3.1 NIG Representation

The NIG representation [14], also known as *intersection graph* [1, 5], was proposed and used in the literature as a fast approximation approach for solving the HP problem [29]. In the NIG representation $\mathrm{NIG}(\mathcal{H}) = (\mathcal{V}, \mathcal{E})$ of a given hypergraph $\mathcal{H} = (\mathcal{U}, \mathcal{N})$, each vertex v_i of $\mathrm{NIG}(\mathcal{H})$ corresponds to net n_i of $\mathcal{H}$. There exist an edge between vertices v_i and v_j of $\mathrm{NIG}(\mathcal{H})$ if and only if the respective nets n_i and n_j are adjacent in $\mathcal{H}$, i.e., $e_{i,j} \in \mathcal{E}$ if and only if $n_j \in Adj_{\mathcal{H}}(n_i)$ which also implies that $n_i \in Adj_{\mathcal{H}}(n_j)$. This NIG definition implies that every node u_h of $\mathcal{H}$ induces a clique $\mathcal{C}_h$ in $\mathrm{NIG}(\mathcal{H})$ where $\mathcal{C}_h = Nets(u_h)$.

Figure 2.5(a) shows an example hypergraph and Figure 2.5(b) presents the net intersection graph of the hypergraph given in Figure 2.5(a).

2.3.2 Matrix-Theoretic View for NIG Representation

Given a hypergraph $\mathcal{H}$, the rectangular matrix A with property $\mathcal{H} = \mathcal{H}_{RN}(A)$ is the row-net matrix of $\mathcal{H}$. The matrix $M = AA^T$ forms the adjacency matrix of the graph $\mathcal{G} = \mathrm{NIG}(\mathcal{H})$.

2.4 Graph Partitioning by Edge Separator (GPES)

2.4.1 GPES Problem

An undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined as a set $\mathcal{V}$ of vertices and set $\mathcal{E}$ of edges. Every edge $e_{ij} \in \mathcal{E}$ connects a pair of distinct vertices v_i and v_j. We use the notation $Adj_{\mathcal{G}}(v_i)$ to denote the set of vertices that are adjacent to vertex v_i in graph $\mathcal{G}$. We extend this operator to include the adjacency set of a vertex subset $\mathcal{V}' \subseteq \mathcal{V}$, i.e., $Adj_{\mathcal{G}}(\mathcal{V}') = \{v_j \in \mathcal{V} - \mathcal{V}' : v_j \in Adj_{\mathcal{G}}(v_i) \text{ for some } v_i \in \mathcal{V}'\}$. The degree d_i of a vertex v_i is equal to the number of edges incident to v_i, i.e., $d_i = |Adj_{\mathcal{G}}(v_i)|$. An edge subset $\mathcal{E}_S$ is a K-way *edge separator* if the subgraph induced by the edges in $\mathcal{E} - \mathcal{E}_S$ has at least K connected components. Let w_i and c_{ij} denote the weight of the vertex $v_i \in \mathcal{V}$ and the weight of the edge $e_{ij} \in \mathcal{E}$, respectively.

$\Pi_{ES} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K\}$ is a K-way vertex partition of $\mathcal{G}$ by edge separator if the following conditions hold: $\mathcal{V}_k \subseteq \mathcal{V}$ and $\mathcal{V}_k \neq \emptyset$ for $1 \leq k \leq K$; $\mathcal{V}_k \cap \mathcal{V}_\ell = \emptyset$ for $1 \leq k \leq \ell \leq K$; $\bigcup_{k=1}^{K} \mathcal{V}_k = \mathcal{V}$; removal of $\mathcal{E}_S$ gives K disconnected parts $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K$.

In the GPES problem, the partitioning constraint is to maintain a ε-$balance$ on the weights of the K parts of the K-way vertex partition $\Pi_{ES} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K\}$. The weight W_k of a part $\mathcal{V}_k$ is defined by the sum of the individual weights of vertices in $\mathcal{V}_k$, i.e., $W_k = \sum_{v_i \in \mathcal{V}_k} w_i$, for $1 \leq k \leq K$. A partition Π_{ES} is said to have ε-$balance$ if the following equation holds:

$$W_k \leq W_{avg}(1 + \varepsilon), \quad for\ 1 \leq k \leq K \tag{2.8}$$

where $W_{avg} = \sum_{k=1}^{K} W_k / K = \sum_{v_i \in \mathcal{V}} w_i / K$. The basic imbalance ε_0 of a graph partition Π_{ES} is defined as the minimum value of ε that Π_{ES} has ε-balance. The partitioning objective is to minimize the cut size (separator size), which can be defined as the total weight of edges in the cut (separator), i.e.,

$$Cutsize(\Pi_{ES}) = \sum_{e_{ij} \in \mathcal{E}_S} c_{ij}. \tag{2.9}$$

Problem 4 *GPES Problem: Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, an imbalance*

parameter ε and an integer K, find a K-way partition Π_{ES} of $\mathcal{G}$ by edge separator, minimizing the $Cutsize(\Pi_{ES})$ such that Π_{ES} has ε-balance.

The GPES Problem is known to be NP-hard [6].

2.4.2 Matrix-Theoretic View for GPES

Given a $p \times p$ symmetric and square matrix M, let $\mathcal{G}(M) = (\mathcal{V}, \mathcal{E})$ denote the standard graph representation of matrix M. Therefore, the M matrix is the adjacency matrix of a given graph $\mathcal{G} = \mathcal{G}(M)$.

$$M_D = PAP^T = \begin{bmatrix} \mathbf{M_{11}} & M_{12} & \dots & M_{1K} \\ M_{21} & \mathbf{M_{22}} & \dots & M_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ M_{K1} & M_{K2} & \dots & \mathbf{M_{KK}} \end{bmatrix}. \tag{2.10}$$

A K-way GPVS $\Pi_{VS} = \{\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_K; \mathcal{V}_S\}$ of $\mathrm{G}(M)$ can be decoded as permuting matrix M into a dense block diagonals form $M_D = PAP^T$ as follows: Π_{ES} is used to define the partial row/column permutation matrix P by permuting the rows/columns corresponding to the vertices of $\mathcal{V}_k$ after those corresponding to the vertices of $\mathcal{V}_{k-1}$ for $2 \leq k \leq K$. The partitioning objective of minimizing the separator size of Π_{ES} corresponds to minimizing the number of off-diagonal nonzeros, whereas the partitioning constraint of maintaining balance on the part weights of Π_{ES} infers balance among the row/columns counts of the square diagonal submatrices in M_D.

2.5 Edge Clique Cover (ECC)

2.5.1 ECC Problem

Given a set $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots\}$ of cliques in $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{C}$ is an edge clique cover (ECC) of $\mathcal{G}$ if for each edge $e_{ij} \in \mathcal{E}$ there exists a clique $\mathcal{C}_h \in \mathcal{C}$ that contains both v_i and v_j.

Problem 5 *ECC Problem [34]: Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, find an ECC $\mathcal{C}$ of the graph $\mathcal{G}$ minimizing the number of cliques in $\mathcal{C}$.*

The ECC problem is known to be NP-hard [34].

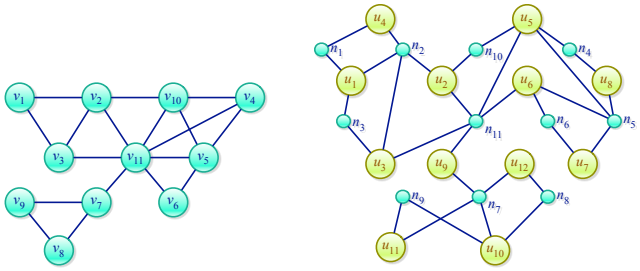
2.5.2 Clique-Node Hypergraph

Given a set $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots\}$ of cliques in graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the clique-node hypergraph $\mathrm{CNH}(\mathcal{G}, \mathcal{C}) = \mathcal{H} = (\mathcal{U}, \mathcal{N})$ of $\mathcal{G}$ for $\mathcal{C}$ is defined as a hypergraph with $|\mathcal{C}|$ nodes and $|\mathcal{V}|$ nets, where $\mathcal{H}$ contains one node u_h for each clique $\mathcal{C}_h$ of $\mathcal{C}$ and one net n_i for each vertex v_i of $\mathcal{V}$, i.e., $\mathcal{U} \equiv \mathcal{C}$ and $\mathcal{N} \equiv \mathcal{V}$. In $\mathcal{H}$, the set of nets that connect node u_h corresponds to the set $\mathcal{C}_h$ of vertices, i.e., $Nets(u_h) \equiv \mathcal{C}_h$ for $1 \leq h \leq |\mathcal{C}|$. In other words, the net n_i connects the nodes corresponding to the cliques that contain vertex v_i of $\mathcal{G}$.

Figure 2.6(a) shows an example graph and Figure 2.6(b) presents the clique-node hypergraph for a given edge clique cover.

2.6 Minimum Set Cover (MSC)

There is a *main set* $\mathcal{T}$ of n elements. Besides, there is a set $\mathcal{F}$ of m subsets of the *main set*, i.e., $\mathcal{F} = \{S_1, S_2, \ldots, S_m\}$ where $\forall S_i \subseteq \mathcal{T}$. A set $\mathcal{F}'$ is said to *cover* a



(a) (b)

Figure 2.6: (a) An example graph $\mathcal{G}$ (b) the clique-node hypergraph $\mathcal{H}$ of $\mathcal{G}$ for ECC $\mathcal{C} = \{\mathcal{C}_1 = \{v_1, v_2, v_3\}, \mathcal{C}_2 = \{v_2, v_{10}, v_{11}\}, \mathcal{C}_3 = \{v_2, v_3, v_{11}\}, \mathcal{C}_4 = \{v_1, v_2\}, \mathcal{C}_5 = \{v_4, v_5, v_{10}, v_{11}\}, \mathcal{C}_6 = \{v_5, v_6, v_{11}\}, \mathcal{C}_7 = \{v_5, v_6\}, \mathcal{C}_8 = \{v_4, v_5\}, \mathcal{C}_9 = \{v_7, v_9\}, \mathcal{C}_{10} = \{v_7, v_8, v_9\}, \mathcal{C}_{11} = \{v_7, v_{11}\}, \mathcal{C}_{12} = \{v_7, v_8\}\}$

set $\mathcal{T}'$, if the union of subsets of $\mathcal{F}'$ is equal to the set $\mathcal{T}'$, i.e., $\cup_{S_i \in \mathcal{F}'} S_i = \mathcal{T}'$. The minimum set cover problem is finding a subset $\mathcal{F}'$ of $\mathcal{F}$. The objective is to minimize the cardinality of the subset $\mathcal{F}'$, whereas the constraint is to make the subset $\mathcal{F}'$ cover the main set $\mathcal{T}$.

Problem 6 *MSC Problem: Given a main set $\mathcal{T}$, and a family $\mathcal{F}$ of subsets of main set, find a subset $\mathcal{F}'$ of $\mathcal{F}$, minimizing the cardinality, $|\mathcal{F}'|$, such that $\mathcal{F}'$ covers the main set $\mathcal{T}$.*

The MSC problem is known to be NP-hard [30].

In the decision version of MSC Problem, an additional input parameter τ presents and Set Cover (SC) Problem asks for the existence of a set $\mathcal{F}' \subseteq \mathcal{F}$ that covers $\mathcal{T}$ with cardinality less than or equal to τ, i.e., $|\mathcal{F}'| \leq \tau$.

2.7 Maximum Set Splitting (MSS)

There is a *main set* $\mathcal{T}$ of n elements. Besides, there is a set $\mathcal{F}$ of m subsets of the *main set*, i.e., $\mathcal{F} = \{S_1, S_2, \ldots, S_m\}$ where $\forall S_i \subseteq \mathcal{T}$. For a given partition a subset S_i is said to be *splitted* if the elements of subset S_i is neither completely in S_1 nor in

S_2. The maximum set splitting problem is finding a 2-way partition of the main set $\mathcal{T}$ minimizing the number of *splitted* subsets in $\mathcal{F}$.

Problem 7 *MSS Problem: Given a main set $\mathcal{T}$, and a family $\mathcal{F}$ of subsets of main set, find a partition $(\mathcal{T}_1, \mathcal{T}_2)$ of $\mathcal{T}$, maximizing the number of* splitted *subsets in $\mathcal{F}$.*

The MSS problem is known to be NP-hard [19].

In the decision version of MSS Problem, Set Splitting (SS) Problem asks for the existence of a partition $(\mathcal{T}_1, \mathcal{T}_2)$ of $\mathcal{T}$ such that all subsets of $\mathcal{F}$ are *splitted*.

Chapter 3

Theoretical Foundations

Theorem 1 *A vertex separator is narrow if and only if there is no vertex in separator with connectivity less than 2.*

Proof If there is a vertex $v_i \in \mathcal{V}_S$ with $\lambda_i = 1$, we can put v_i to the only part $\mathcal{V}_k \in \Lambda_i$, and since $Adj_{\mathcal{G}}(v_i) \subseteq \mathcal{V}_k \cup \mathcal{V}_S$, $\mathcal{V}_S - v_i$ would still be a valid separator. If there is a vertex $v_i \in \mathcal{V}_S$ with $\lambda_i = 0$, we can put v_i to any part and since $Adj_{\mathcal{G}}(v_i) \subseteq \mathcal{V}_S$, $V_S - v_i$ would still be a valid separator. Thus, if a vertex separator is narrow, there is no vertex in the separator with connectivity less than 2. If a vertex separator $\mathcal{V}_S$ is not narrow, there exists a set $\mathcal{V}'_S \subset \mathcal{V}_S$ which forms a separator. Assume that there is a vertex $v_i \in \mathcal{V}_S - \mathcal{V}'_S$ with $\lambda_i \geq 2$. Let the part $\mathcal{V}_k$ be the part that vertex v_i is assigned to. Then $Adj_{\mathcal{G}}(\mathcal{V}_k) \not\subseteq \mathcal{V}_S$, which invalidates the separator. Hence, for all vertices $v_i \in \mathcal{V}_S - \mathcal{V}'S$, $\lambda_i \leq 2$. Therefore, if a vertex separator $\mathcal{V}_S$ is not narrow, there exists a vertex in separator whose connectivity is less than 2. □

For an optimization problem A with input parameters a, if A(a) has no feasible solution than the optimal solution S_A is to be ∞ and $-\infty$, if A is a minimization and maximization problem, respectively.

Proposition 1 *Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, an imbalance parameter ε and an integer K, either GPVSn has no feasible solution or optimal solution S_{GPVS}*

(minimum separator size) of GPVS problem is less than or equal to the optimal solution S_{GPVSn} *(minimum separator size) of GPVSn problem, i.e.,* $S_{GPVS} \leq S_{GPVSn}$.

Proof Any narrow vertex separator is also a valid vertex separator for GPVS problem. Therefore, given the same input, any solution to narrow GPVS problem is also a valid solution to GPVS problem. So, if a feasible solution exists for GPVSn problem, minimum separator size of GPVS problem is less than or equal to the minimum separator size of GPVSn problem given the same input. □

The following theorem is the basis for hypergraph-partitioning-based GPVS approach.

Theorem 2 *Consider a hypergraph* $\mathcal{H} = (\mathcal{U}, \mathcal{N})$ *and its NIG representation* $\mathrm{NIG}(\mathcal{H}) = (\mathcal{V}, \mathcal{E})$. *A K-way node-partition* $\Pi_{HP} = \{\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_K\}$ *of* $\mathcal{H}$ *can be decoded as a K-way vertex separator* $\Pi_{VS} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K; \mathcal{V}_S\}$ *of* $\mathrm{NIG}(\mathcal{H})$*, where*

(a) partitioning objective of minimizing the cutsize of Π_{HP} *according to the cutnet metric corresponds to minimizing the separator size of* Π_{VS}.

(b) partitioning constraint of ε*-balance on internal nets of* Π_{HP} *infers* ε*-balance of* Π_{VS}

Proof As described in [4], the K-way node-partition $\Pi_{HP} = \{\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_K\}$ of $\mathcal{H}$ can be decoded as a $(K+1)$-way net-partition $\{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$. We consider this $(K+1)$-way net-partition $\Pi_{HP} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$ of $\mathcal{H}$ as inducing a K-way GPVS $\Pi_{VS} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K; \mathcal{V}_S\}$ on $\mathrm{NIG}(\mathcal{H})$, where $\mathcal{V}_k \equiv \mathcal{N}_k$, for $1 \leq k \leq K$, and $\mathcal{V}_S \equiv \mathcal{N}_S$. Consider an internal net n_j of node-part $\mathcal{U}_k$ in Π_{HP}, i.e., $n_j \in \mathcal{N}_k$. It is clear that $Adj_{\mathcal{H}}(n_j) \subseteq \mathcal{N}_k \cup \mathcal{N}_S$, which implies $Adj_{\mathcal{H}}(\mathcal{N}_k) \subseteq \mathcal{N}_S$. Since $\mathcal{V}_k \equiv \mathcal{N}_k$ and $\mathcal{V}_S \equiv \mathcal{N}_S$, $Adj_{\mathcal{H}}(\mathcal{N}_k) \subseteq \mathcal{N}_S$ in Π_{HP} implies $Adj_{\mathcal{G}}(\mathcal{V}_k) \subseteq \mathcal{V}_S$ in Π_{VS}. In other words, $Adj_{\mathcal{G}}(\mathcal{V}_k) \cap \mathcal{V}_\ell = \emptyset$, for $1 \leq \ell \leq K$ and $\ell \neq k$. Thus, $\mathcal{V}_S$ of Π_{VS} constitutes a valid separator of size $\sum_{v_i \in \mathcal{V}_S} w_i = \sum_{n_j \in \mathcal{N}_S} c_j$. So, minimizing the cutsize of Π_{HP} with cutnet metric corresponds to minimizing the separator size of

Π_{VS}. Since $\sum_{v_i \in \mathcal{V}_k} w_i = \sum_{n_j \in \mathcal{N}_k} c_j$, for $1 \leq k \leq K$, ε-$balance$ on the internal nets of node parts of Π_{HP} corresponds to ε-$balance$ on vertices of parts of Π_{VS}. □

Corollary 1 *Consider an undirected graph $\mathcal{G}$. A K-way node-partition Π_{HP} of any hypergraph $\mathcal{H}$ for which $\mathrm{NIG}(\mathcal{H}) \equiv \mathcal{G}$ can be decoded as a K-way vertex separator Π_{VS} of $\mathcal{G}$ with same objective value under same constraint parameter.*

The following theorem is the basis for GPVS-based Hypergraph Partitioning approach.

Theorem 3 *Consider a hypergraph $\mathcal{H} = (\mathcal{U}, \mathcal{N})$ and its NIG representation $\mathcal{G} = \mathrm{NIG}(\mathcal{H}) = (\mathcal{V}, \mathcal{E})$. A K-way narrow vertex separator $\Pi_{VS} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K; \mathcal{V}_S\}$ of $\mathcal{G}$ can be decoded as a K-way node-partition $\Pi_{HP} = \{\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_K\}$ of $\mathcal{H}$, where*

(a) partitioning objective of minimizing the separator size of Π_{VS} corresponds to minimizing the cutsize of Π_{HP} according to cutnet metric.

(b) partitioning constraint of ε-$balance$ of Π_{VS} infers ε-$balance$ on internal nets of Π_{HP}.

Proof The K-way narrow vertex separator $\Pi_{VS} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K; \mathcal{V}_S\}$ of $\mathcal{G}$ induces a $(K+1)$-way net-partition $\Pi_{HP} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$ of $\mathcal{H}$, where $\mathcal{N}_k \equiv \mathcal{V}_k$, for $1 \leq k \leq K$, and $\mathcal{N}_S \equiv \mathcal{V}_S$. As to be remembered, the set of *free* nodes $\mathcal{U}_F$ of a Π_{HP} of $\mathcal{H}$ is defined as the set of nodes whose all nets are external nets, i.e., $\mathcal{U}_F = \mathcal{U} - \bigcup_{n_j \notin \mathcal{N}_S} Pins(n_j)$. The net-partition $\Pi_{HP} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$ can be realized by the following node-partition $\Pi_{HP} = \{\mathcal{U}_1 = \mathcal{U}'_1 \cup \mathcal{U}_F, \mathcal{U}_2, \ldots, \mathcal{U}_K\}$ where $\mathcal{U}_k = \bigcup_{n_j \in \mathcal{N}_k} Pins(n_j)$, for $2 \leq k \leq K$ and $\mathcal{U}'_1 = \bigcup_{n_j \in \mathcal{N}_1} Pins(n_j)$. Clearly, this node partition covers all nodes in $\mathcal{H}$. The disjointness of this node partition should be checked. Assume that there exists a node u_i that appears in two different node parts $\mathcal{U}_{k_1}$ and $\mathcal{U}_{k_2}$. This node u_i can not be free, because we assure that free nodes are only appear in node part $\mathcal{U}_1$. Then, since u_i is not a free

node and $u_i \in \mathcal{U}_{k_1} \cap \mathcal{U}_{k_2}$, there exists two nets n_{j_1} and n_{j_2}, from parts $\mathcal{N}_{k_1}$ and $\mathcal{N}_{k_2}$, respectively such that $u_i \in Pins(n_{j_1}) \cap Pins(n_{j_2})$, which yields to the corresponding vertices $v_{j_1} \in \mathcal{V}_{k_1}$ and $v_{j_2} \in \mathcal{V}_{k_2}$ are adjacent vertices in $\mathcal{G}$. This contradicts with the vertex separator definition. Therefore the disjointness property of the node partition $\Pi_{HP} = \{\mathcal{U}_1 = \mathcal{U}'_1 \cup \mathcal{U}_F, \mathcal{U}_2, \ldots, \mathcal{U}_K\}$ holds. Let net partition $\Pi'_{HP} = \{\mathcal{N}'_1, \mathcal{N}'_2, \ldots, \mathcal{N}'_K; \mathcal{N}'_S\}$ of $\mathcal{H}$ be the net partition induced by this node partition. $\mathcal{U}_k = \bigcup_{n_j \in \mathcal{N}_k} Pins(n_j)$, $\mathcal{N}_k \subseteq \mathcal{N}'_k$, for $1 \leq k \leq K$. Since the vertex separator Π_{VS} is narrow, each vertex $v_j \in \mathcal{V}_S$ has connectivity at least 2. This also refers to that $n_j \in \mathcal{N}_S$ has connectivity at least 2 induced by non-free nodes. Therefore, any net $n_j \in \mathcal{N}_S$ is also an element of $\mathcal{N}'_S$, i.e., $\mathcal{N}_S \subseteq \mathcal{N}'_S$. As a result, $\Pi'_{HP} = \Pi_{HP}$ and the node partition $\Pi_{HP} = \{\mathcal{U}_1 = \mathcal{U}'_1 \cup \mathcal{U}_F, \mathcal{U}_2, \ldots, \mathcal{U}_K\}$ induces the net partition $\Pi_{HP} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$ of $\mathcal{H}$. So, minimizing the separator size of Π_{VS} corresponds to minimizing the cutsize of Π_{HP} according to the cutnet metric. Since $\sum_{n_j \in \mathcal{N}_k} c_j = \sum_{v_i \in \mathcal{V}_k} w_i$, for $1 \leq k \leq K$, ε-$balance$ on vertices of parts of Π_{VS} corresponds to ε-$balance$ on the internal nets of node parts of Π_{HP}. □

Corollary 2 *Consider a hypergraph $\mathcal{H}$. A K-way narrow vertex separator Π_{VS} of $\mathcal{G} = \mathrm{NIG}(\mathcal{H})$ can be decoded as a K-way node-partition Π_{HP} of $\mathcal{H}$ with the same objective value under the same constraint parameter.*

Theorem 4 *(Main Theorem) Given a hypergraph $\mathcal{H} = (\mathcal{U}, \mathcal{N})$, an imbalance parameter ε and a part count K, consider the following three problems:*

(1) HP Problem of net balancing with cutnet metric on hypergraph $\mathcal{H}$, imbalance parameter ε and partition amount K.

(2) GPVS Problem on graph $\mathrm{NIG}(\mathcal{H})$, imbalance parameter ε and part count K.

(3) GPVSn Problem on graph $\mathrm{NIG}(\mathcal{H})$, imbalance parameter ε and part count K.

Let S_{HP}, S_{GPVS} and S_{GPVSn} be the optimal solutions of HP problem (1), GPVS problem (2) and GPVSn problem (3), respectively. Then;

$$S_{GPVS} \leq S_{HP} \leq S_{GPVSn}$$

Proof According to Theorem 2, an optimal solution Π_{HP} of HP Problem (1) can be decoded as a feasible solution Π_{VS} of GPVS Problem (2) with same objective value. Therefore $S_{GPVS} \leq S_{HP}$. According to Theorem 3, an optimal solution Π_{VS} of GPVSn Problem (3) can be decoded as a feasible solution Π_{HP} of HP Problem (2) with same objective value. Therefore $S_{HP} \leq S_{GPVSn}$. $\square$

Chapter 4

Hypergraph-Partitioning-based Graph Partitioning by Vertex Separator

Here, we are interested in the hypergraph partitioning (HP) problem in which objective is defined according to the cutnet metric as in Equation 2.5 and the constraint of ε-balance is considered on the internal nets. Theorem 2 forms the basis of hypergraph-partitioning-based graph partitioning by vertex separator (GPVS) approach. The theorem states that a K-way node-partition $\Pi_{HP} = \{\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_K\}$ of $\mathcal{H}$ can be decoded as a K-way vertex separator $\Pi_{VS} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K; \mathcal{V}_S\}$ of $\mathrm{NIG}(\mathcal{H})$, where the objective of minimizing the cutsize of Π_{HP} corresponds to minimizing the separator size of Π_{VS} and constraint of ε-$balance$ of Π_{HP} infers ε-$balance$ of Π_{VS}. Here, we try to solve GPVS problem by using HP problem. So, we are given a graph $\mathcal{G}$ and we are to find a vertex separator Π_{VS} of the graph $\mathcal{G}$. Therefore, the approach is much more related to Corollary 1, which states that a K-way node-partition Π_{HP} of any hypergraph $\mathcal{H}$ for which $\mathrm{NIG}(\mathcal{H}) \equiv \mathcal{G}$ can be decoded as a K-way vertex separator Π_{VS} of $\mathcal{G}$, where the objective of minimizing the cutsize of Π_{HP} corresponds to minimizing the separator size of Π_{VS} and constraint of ε-$balance$ of Π_{HP} infers ε-$balance$ of Π_{VS}.

The definition of GPVS problem in Section 2.1.1 is reshowed below:

Problem: *GPVS Problem: Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, an imbalance parameter ε and an integer K, find a K-way partition Π_{VS} of $\mathcal{G}$ by vertex separator, minimizing the $Separatorsize(\Pi_{VS})$ such that Π_{VS} has ε-balance.*

Combinatorial Reduction 1 *GPVS Problem $\Rightarrow$ HP Problem*

Algorithm 1 *GPVS Problem $\Rightarrow$ HP Problem Reduction*

Require: $\mathcal{G} = (\mathcal{V}, \mathcal{E}), \varepsilon, K$
1: $\mathcal{G} \rightarrow \mathcal{H}$
2: $\Pi_{HP} \leftarrow HP(\mathcal{H}, \varepsilon, K)$
3: $\Pi_{HP} \rightarrow \Pi_{VS}$
4: **return** Π_{VS}

1) *Constructing a hypergraph $\mathcal{H}$ for which $\mathrm{NIG}(\mathcal{H}) \equiv \mathcal{G}$:* This is not trivial because there is no unique construction of a hypergraph $\mathcal{H}$ for which $\mathrm{NIG}(\mathcal{H}) \equiv \mathcal{G}$. The approaches for constructing such hypergraph $\mathcal{H}$ is explained in this chapter in detail.

2) *Finding K-way Hypergraph Partition Π_{HP} with minimum cutsize under ε-balance*

3) *Decoding Π_{HP} to a vertex separator Π_{VS}:* A node-partition Π_{HP} of $\mathcal{H}$ can be trivially be decoded to the vertex separator Π_{VS} of $\mathcal{G}$ as follows: A node-partition $\Pi_{HP} = \{\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_K\}$ of $\mathcal{H}$ induces a net-partition $\Pi_{HP} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$ of $\mathcal{H}$ and Π_{VS} of $\mathcal{G}$ is constructed from this net partition as $\Pi_{VS} = \{\mathcal{V}_1 \equiv \mathcal{N}_1, \mathcal{V}_2 \equiv \mathcal{N}_2, \ldots, \mathcal{V}_K \equiv \mathcal{N}_K; \mathcal{V}_S \equiv \mathcal{N}_S\}$.

Figure 4 shows the combinatorial reduction from HP problem to GPVS problem as follows. In Figure 4.1(a), firstly we are given a graph $\mathcal{G}$. In the second step, we construct a hypergraph $\mathcal{H}$ based on finding an ECC $\mathcal{C}$ of G as in Figure 4.1(b). Then, in the third step, we find a hypergraph partition Π_{HP} of $\mathcal{H}$ as in Figure 4.1(c). Finally, in the forth step, we decode Π_{HP} as a vertex separator Π_{VS} of the given graph $\mathcal{G}$ as in Figure 4.1(d).

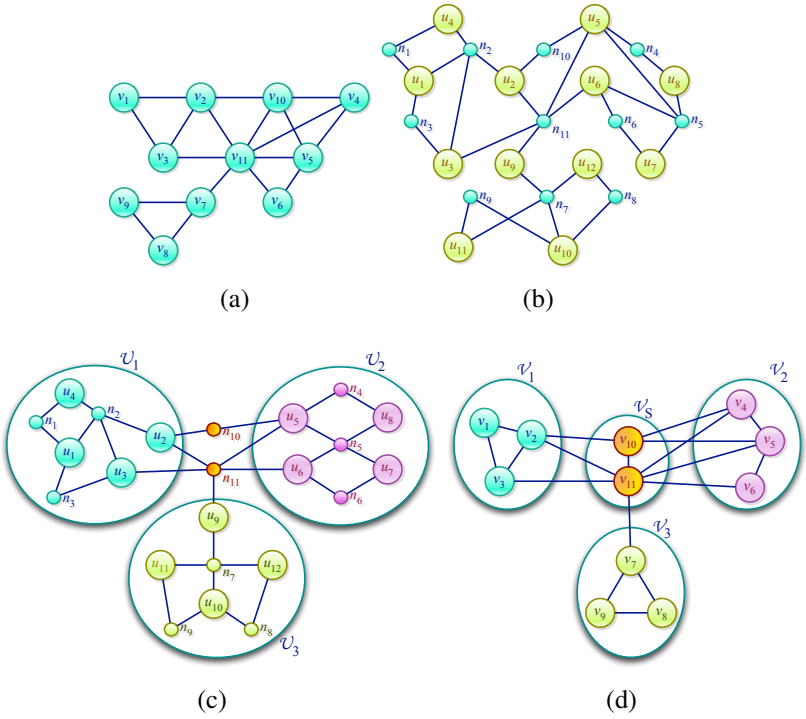


Figure 4.1: (a) An example graph $\mathcal{G}$ (b) The clique-node hypergraph $\mathcal{H} = CNH(\mathcal{G}, \mathcal{C})$ of $\mathcal{G}$ for an ECC $\mathcal{C}$ (c) A hypergraph partition Π_{HP} of $\mathcal{H}$ (d) The vertex separator Π_{VS} of $\mathcal{G}$ induced by Π_{HP} of $\mathcal{H}$

4.1 Hypergraph Construction

4.1.1 Theoretical Base

The following two theorems state that, for a given graph $\mathcal{G}$, the problem of constructing a hypergraph whose NIG representation is same as $\mathcal{G}$ is equivalent to the problem of finding an ECC of $\mathcal{G}$.

Theorem 5 *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a hypergraph $\mathcal{H} = (\mathcal{U}, \mathcal{N})$, if $\mathrm{NIG}(\mathcal{H}) \equiv \mathcal{G}$ then $\mathcal{H} \equiv \mathrm{CNH}(\mathcal{G}, \mathcal{C})$ with $\mathcal{C} = \{\mathcal{C}_h \equiv Nets(u_h) : 1 \leq h \leq |\mathcal{U}|\}$ is an ECC of $\mathcal{G}$.*

Proof Since $\mathrm{NIG}(\mathcal{H}) \equiv \mathcal{G}$, there is an edge $e_{ij} = \{v_i, v_j\}$ in $\mathcal{G}$ if and only if nets n_i and n_j are adjacent in $\mathcal{H}$ which means there exists a node u_h in $\mathcal{H}$ such that both

$n_i \in Nets(u_h)$ and $n_j \in Nets(u_h)$. Since u_h induces the clique $\mathcal{C}_h \in \mathcal{C}$, $\mathcal{C}_h$ contains both vertices v_i and v_j. □

Note that $\mathcal{C} = \{\mathcal{C}_h \equiv Nets(u_h) : 1 \leq h \leq |\mathcal{U}|\}$ is the unique ECC of $\mathcal{G}$ satisfying $\mathcal{H} \equiv \mathrm{CNH}(\mathcal{G}, \mathcal{C})$.

Theorem 6 *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, for any ECC $\mathcal{C}$ of $\mathcal{G}$, the NIG representation of clique-node hypergraph of $\mathcal{C}$ is equivalent to $\mathcal{G}$, i.e.,* $\mathrm{NIG}(\mathrm{CNH}(\mathcal{G}, \mathcal{C})) \equiv \mathcal{G}$.

Proof By construction, two nets n_i and n_j are adjacent in $\mathrm{CNH}(\mathcal{G}, \mathcal{C})$ if and only if there exists a clique $\mathcal{C}_h \in \mathcal{C}$ such that $\mathcal{C}_h$ contains both vertices v_i and v_j in $\mathcal{G}$. Since $\mathcal{C}$ is an ECC of $\mathcal{G}$, there is such a clique $\mathcal{C}_h \in \mathcal{C}$ if and only if there is an edge e_{ij} in $\mathcal{G}$. □

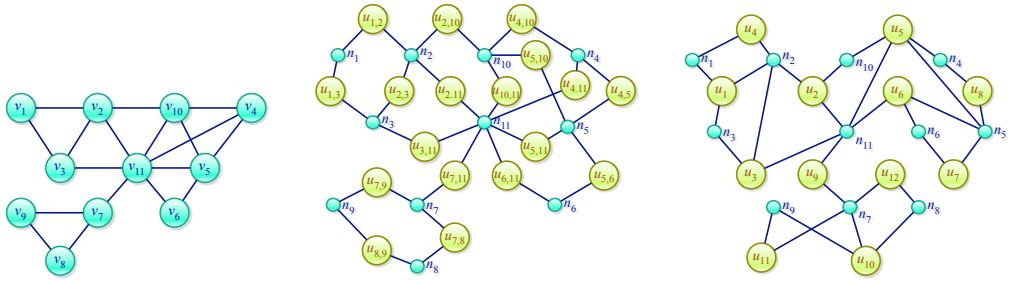
4.1.2 Hypergraph Construction Methodology

The approach for hypergraph construction is first finding an ECC $\mathcal{C}$ of $\mathcal{G}$, and then constructing the hypergraph $\mathcal{H}$ as the clique-node hypergraph of $\mathcal{G}$ for $\mathcal{C}$. This hypergraph construction methodology is presented as follows:

Algorithm 2 Hypergraph Construction Methodology

Require: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
1: $\mathcal{C} \leftarrow ECC(\mathcal{G})$
2: $\mathcal{H} \leftarrow \mathrm{CNH}(\mathcal{G}, \mathcal{C})$
3: **return** $\mathcal{H}$

The choice of the ECC may affect the run-time performance of HP-tool depending on the size of the clique-node hypergraph. Since the number of nets in the clique-node hypergraph is fixed, the number of cliques and the sum of the clique sizes, which respectively correspond to the number of nodes and pins, determine the size of the hypergraph. Hence, an ECC with small number of large cliques is likely to induce a clique-node hypergraph of small size.



(a) (b) (c)

Figure 4.2: (a) An example graph $\mathcal{G}$ (b) The 2-clique-node hypergraph $\mathcal{H}^2$ (c) A 4-clique-node hypergraph $\mathcal{H}^4$

The ECC problem can be considered to be relevant to our problem of finding a "good" ECC. The ECC problem is known to be NP-hard [34] and the literature contains a number of heuristics [23, 33, 34] for solving the ECC problem. However, even the fastest heuristics' [23] running time complexity is $O(|\mathcal{V}||\mathcal{E}|)$, which makes it an impractical approach.

In this work, three different types of ECCs are investigated, namely $\mathcal{C}^2, \mathcal{C}^3$, and $\mathcal{C}^4$, to observe the effects of increasing clique size in the solution quality and run-time performance of the proposed approach. Here, $\mathcal{C}^2$ denotes the ECC of all 2-cliques (edges), i.e., $\mathcal{C}^2 = \mathcal{E}$; $\mathcal{C}^3$ denotes an ECC of 2- and 3-cliques; $\mathcal{C}^4$ denotes an ECC of 2-, 3-, and 4-cliques. In general, $\mathcal{C}^k$ denotes an ECC of cliques in which maximum clique size is bounded above by k. Note that $\mathcal{C}^2$ is unique, whereas $\mathcal{C}^3$ and $\mathcal{C}^4$ are not necessarily unique. The clique-node hypergraph induced by $\mathcal{C}^k$ will be referred as $\mathcal{H}^k = \mathrm{CNH}(\mathcal{G}, \mathcal{C}^k)$.

The clique-node hypergraph $\mathcal{H}^2$ deserves special attention, since it is uniquely defined for a given graph $\mathcal{G}$. In $\mathcal{H}^2$, there exists one node of degree 2 for each edge e_{ij} of $\mathcal{G}$. The net n_i corresponding to vertex v_i of $\mathcal{G}$ connects all nodes corresponding to the edges that are incident to vertex v_i, for $1 \leq i \leq |\mathcal{V}|$. So, $\mathcal{H}^2$ contains $|\mathcal{E}|$ nodes, $|\mathcal{V}|$ nets, and $2|\mathcal{E}|$ pins. The running time of HP-based GPVS using $\mathcal{H}^2$ is expected to be quite high because of the large number of nodes and pins. Figure 4.1.2 displays the 2-clique-node hypergraph $\mathcal{H}^2$ and a 4-clique-node hypergraph $\mathcal{H}^4$ of the sample graph $\mathcal{G}$ given in Figure 4.2(a). As seen in Figure 4.2(b), each node of $\mathcal{H}^2$ is labeled as u_{ij} to show the one-to-one correspondence between nodes of $\mathcal{H}^2$ and edges of $\mathcal{G}$.

Algorithm 3 $\mathcal{C}^3$ Construction Algorithm

Require: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
1: **for each** vertex $v \in \mathcal{V}$ **do**
2: $\pi_1[v] \leftarrow NIL$
3: **for each** edge $e_{ij} \in \mathcal{E}$ **do**
4: cover$[e_{ij}] \leftarrow 0$
5: $\mathcal{C}^3 \leftarrow \emptyset$
6: **for each** vertex $v_i \in \mathcal{V}$ **do**
7: **for each** vertex $v_j \in Adj_{\mathcal{G}}(v_i)$ with $j > i$ **do**
8: $\pi_1[v_j] \leftarrow v_i$
9: **for each** vertex $v_j \in Adj_{\mathcal{G}}(v_i)$ with $j > i$ **do**
10: **for each** vertex $v_k \in Adj_{\mathcal{G}}(v_j)$ with $k > j$ **do**
11: **if** $\pi_1[v_k] = v_i$ **and** $\sum_{e \in \binom{\{v_i, v_j, v_k\}}{2}}$ cover$[e] < 2$ **then**
12: $\mathcal{C}^3 \leftarrow \mathcal{C}^3 \cup \{\{v_i, v_j, v_k\}\}$ ▷ Add the 3-clique to $\mathcal{C}^3$
13: **for each** edge $e \in \binom{\{v_i, v_j, v_k\}}{2}$ **do**
14: cover$[e] \leftarrow 1$
15: **if** cover$[e_{ij}] = 0$ **then**
16: $\mathcal{C}^3 \leftarrow \mathcal{C}^3 \cup \{\{v_i, v_j\}\}$ ▷ Add the 2-clique to $\mathcal{C}^3$
17: cover$[e_{ij}] \leftarrow 1$

That is, node u_{ij} of $\mathcal{H}^2$ corresponds to edge e_{ij} of $\mathcal{G}$, where $Nets(u_{ij}) = \{n_i, n_j\}$.

4.1.3 Hypergraph Construction Algorithms

Algorithm 3 and Algorithm 4 display the algorithms developed for constructing a $\mathcal{C}^3$ and a $\mathcal{C}^4$, respectively. The goal of both algorithms is to minimize the number of pins in the clique-node hypergraphs as much as possible. Both algorithms visit the vertices in random order in order to introduce randomization to the ECC construction process. In both algorithms, each edge is processed along only one direction (i.e., from low to high numbered vertex) to avoid identifying the same clique more than once.

In Algorithm 3, for each visited vertex v_i, the 3-clique(s) that contain v_i are searched for by trying to locate 2-cliques between the vertices in $Adj_{\mathcal{G}}(v_i)$. This search is performed by scanning the adjacency list of each vertex v_j in $Adj_{\mathcal{G}}(v_i)$. For each vertex, a parent field π_1 is maintained for efficient identification of 3-cliques during this search. An identified 3-clique $\mathcal{C}_h$ is selected for inclusion in $\mathcal{C}^3$ if the number of already covered edges of $\mathcal{C}_h$ is at most 1. The rationale behind this selection criteria is as follows: Recall that a 3-clique in $\mathcal{C}^3$ adds 3 pins to $\mathcal{H}^3$, since it incurs a node of

Algorithm 4 $\mathcal{C}^4$ Construction Algorithm

Require: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
1: **for each** vertex $v \in \mathcal{V}$ **do**
2: $\pi_1[v] \leftarrow \pi_2[v] \leftarrow NIL$
3: **for each** edge $e_{ij} \in \mathcal{E}$ **do**
4: cover$[e_{ij}] \leftarrow 0$
5: $\mathcal{C}^4 \leftarrow \emptyset$
6: **for each** vertex $v_i \in \mathcal{V}$ **do**
7: **for each** vertex $v_j \in Adj_{\mathcal{G}}(v_i)$ with $j > i$ **do**
8: $\pi_1[v_j] \leftarrow v_i$
9: **for each** vertex $v_j \in Adj_{\mathcal{G}}(v_i)$ with $j > i$ **do**
10: **for each** vertex $v_k \in Adj_{\mathcal{G}}(v_j)$ with $k > j$ **do**
11: $\pi_2[v_k] \leftarrow v_j$
12: **for each** vertex $v_k \in Adj_{\mathcal{G}}(v_j)$ with $k > j$ **do**
13: **if** $\pi_1[v_k] = v_i$ **then**
14: **if** $\sum_{e \in \binom{\{v_i, v_j, v_k\}}{2}}$ cover$[e] < 2$ **then**
15: **for each** vertex $v_\ell \in Adj_{\mathcal{G}}(v_k)$ with $\ell > k$ **do**
16: **if** $\pi_1[v_\ell] = v_i$ **and** $\pi_2[v_\ell] = v_j$ **then**
17: **if** $\sum_{e \in \binom{\{v_i, v_j, v_k, v_\ell\}}{2}}$ cover$[e] < 4$ **then**
18: $\mathcal{C}^4 \leftarrow \mathcal{C}^4 \cup \{\{v_i, v_j, v_k, v_\ell\}\}$ ▷ Add the 4-clique to $\mathcal{C}^4$
19: **for each** edge $e \in \binom{\{v_i, v_j, v_k, v_\ell\}}{2}$ **do**
20: cover$[e] \leftarrow 1$
21: **if** $\sum_{e \in \binom{\{v_i, v_j, v_k\}}{2}}$ cover$[e] < 2$ **then**
22: $\mathcal{C}^4 \leftarrow \mathcal{C}^4 \cup \{\{v_i, v_j, v_k\}\}$ ▷ Add the 3-clique to $\mathcal{C}^4$
23: **for each** edge $e \in \binom{\{v_i, v_j, v_k\}}{2}$
24: cover$[e] \leftarrow 1$
25: **if** cover$[e_{ij}] = 0$ **then**
26: $\mathcal{C}^4 \leftarrow \mathcal{C}^4 \cup \{\{v_i, v_j\}\}$ ▷ Add the 2-clique to $\mathcal{C}^4$
27: cover$[e_{ij}] \leftarrow 1$

degree 3 in $\mathcal{H}^3$. If only one edge of $\mathcal{C}_h$ is already covered by other 3-clique(s) in $\mathcal{C}^3$, it is still beneficial to cover the remaining two edges of $\mathcal{C}_h$ by selecting $\mathcal{C}_h$ instead of selecting the two 2-cliques covering those uncovered edges, because the former selection incurs 3 pins whereas the latter incurs 4 pins. If, however, any two edges of $\mathcal{C}_h$ are already covered by another 3-clique in $\mathcal{C}^3$, it is clear that the remaining uncovered edge is better to be covered by a 2-clique. After scanning the adjacency list of v_j in $Adj_{\mathcal{G}}(v_i)$, if edge $\{v_i, v_j\}$ is not covered by any 3-clique then it is added to $\mathcal{C}^3$ as a 2-clique.

In Algorithm 4, for each visited vertex v_i, the 4-clique(s) that contain v_i are searched for after finding the 3-clique(s) that contain v_i as in Algorithm 3. For each

3-clique $\{v_i, v_j, v_k\}$ identified in $Adj_{\mathcal{G}}(v_i)$, the 4-clique(s) that contain the 3-clique $\{v_i, v_j, v_k\}$ are searched for by scanning $Adj_{\mathcal{G}}(v_k)$, where v_k is the last vertex added to the 3-clique. For each vertex, a second parent field π_2 is maintained together with π_1 for efficient identification of a vertex v_ℓ in $Adj_{\mathcal{G}}(v_k)$ that is adjacent to both v_j and v_i. A 4-clique $\mathcal{C}_h = \{v_i, v_j, v_k, v_\ell\}$ is selected to be added to $\mathcal{C}^4$, if at most 3 out of 6 edges of $\mathcal{C}_h$ are already covered by other 3- and/or 4-clique(s) in $\mathcal{C}^4$. After scanning $Adj_{\mathcal{G}}(v_k)$, if at most one edge of the 3-clique $\{v_i, v_j, v_k\}$ is covered by other 3- or 4-cliques then it is added to $\mathcal{C}^4$ as a 3-clique. After scanning $Adj_{\mathcal{G}}(v_j)$, if edge $\{v_i, v_j\}$ is not covered by any 3- or 4-clique then it is added to $\mathcal{C}^4$ as a 2-clique.

Note that the ideas in Algorithms 3 and 4 can be extended to a general approach for constructing $\mathcal{C}_k$. However, this general approach requires maintaining $k-2$ parent fields for each vertex. Secondly, in some cases, such as Linear Programming Problems, the ECC $\mathcal{C}$ of the graph $\mathcal{G}$ might be given naturally. In such situations, the hypergraph $\mathcal{H}$ can be directly constructed skipping the ECC construction procedure.

4.2 Hypergraph Sparsening

Assume a hypergraph $\mathcal{H}$ of a given graph $\mathcal{G}$, where $\mathrm{NIG}(\mathcal{H}) \equiv \mathcal{G}$, exists either by constructing from an ECC which is naturally given or constructed using the techniques explained in Section 4.1. This hypergraph $\mathcal{H}$ can further be sparsened. Still, some nodes or pins might be redundant in terms of determining the graph $\mathcal{G}$. Here, two problems will be proposed and discussed. For the purpose of comprising sparsened hypergraph $\mathcal{H}'$, one problem aims at deleting as many nodes from $\mathcal{H}$, whereas the other aims at deleting as many pins from $\mathcal{H}$ so as not to disturb NIG property, i.e., $\mathrm{NIG}(\mathcal{H}') \equiv \mathcal{G}$.

4.2.1 Hypergraph Sparsening by Node Deletion (HS-n)

Problem 8 *Hypergraph Sparsening by Node Deletion Problem (HS-n Problem): Given a hypergraph $\mathcal{H} = (\mathcal{U}, \mathcal{N})$, find a hypergraph $\mathcal{H}'$ minimizing the number of*

nodes of $\mathcal{H}'$ *such that* $\mathrm{NIG}(\mathcal{H}') \equiv \mathrm{NIG}(\mathcal{H})$.

Theorem 7 *The HS-n Problem is NP-Hard.*

Proof Consider the decision version of the HS-n Problem. For the NP-completeness proof, a reduction from Set Cover Problem, which is decision version of Minimum Set Cover Problem, is used. For a given Set Cover Instance $(\mathcal{T}, \mathcal{F})$, where $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ of m elements and $\mathcal{F} = \{S_1, S_2, \ldots, S_n\}$ of n subsets, the HS-n Instance $\mathcal{H} = (\mathcal{U}, \mathcal{N})$ is constructed as $\mathcal{U} = \{u_1, u_2, \ldots, u_{n+m+1}\}$ and $\mathcal{N} = \{n_1, n_2, \ldots, n_{2m+2}\}$. The pin construction is done as follows:

- $Pins(n_{2i-1}) = \{u_j : t_i \in S_j\} \cup \{u_{n+1}, u_{n+i+1}\}$, for $1 \leq i \leq m$.
- $Pins(n_{2i}) = \{u_j : t_i \in S_j\} \cup \{u_{n+j+1} : 1 \leq j \neq i \leq m\}$, for $1 \leq i \leq m$.
- $Pins(n_{2m+1}) = \{u_{n+1}\}$.
- $Pins(n_{2m+2}) = \{u_{n+j+1} : 1 \leq j \leq m\}$.

Take a HS-n solution $\mathcal{H}'$ with $\tau+m+1$ nodes. The nodes $u_{n+1}, u_{n+2}, \ldots, u_{n+m+1}$ should exists in $\mathcal{H}'$, because each vertex u_i has at least one unique edge $e \in \mathrm{NIG}(\mathcal{H})$ which can be constructed by only the node u_i, for $n{+}1 \leq i \leq n{+}m{+}1$. The remaining uncovered m edges in graph $\mathrm{NIG}(\mathcal{H})$ corresponds to m elements of the main set of SC Instance. By the construction of hypergraph $\mathcal{H}$, for the remaining uncovered m edges, a node u_i can cover edge e_j if and only if subset S_i includes element t_j, for $1 \leq i \leq n$ and $1 \leq j \leq m$. In this construction, there exists a hypergraph $\mathcal{H}'$ such that $NIG(\mathcal{H}') \equiv NIG(\mathcal{H})$ with size smaller or equal to $\tau + m + 1$ if and only if there exists a set cover $\mathcal{F}'$ with size smaller or equal to τ. Therefore, the decision version of HS-n problem is NP-complete which implies that HS-n problem is NP-hard. □

Combinatorial Reduction 2 *HS-n Problem ⇒ MSC Problem*

1) *Constructing the MSC instance* $(\mathcal{T}, \mathcal{F})$*:* The edges of the graph $\mathcal{G} = \mathrm{NIG}(\mathcal{H})$ constitutes the main set $\mathcal{T} = \{t_{ij} : e_{ij} \in \mathcal{G}\}$, whereas the nodes of hypergraph

Algorithm 5 *HS-n Problem ⇒ MSC Problem Reduction*

Require: $\mathcal{H} = (\mathcal{U}, \mathcal{N})$
1: $\mathcal{H} \rightarrow (\mathcal{T}, \mathcal{F})$
2: $\mathcal{F}' \leftarrow MSC(\mathcal{T}, \mathcal{F})$
3: $\mathcal{F} \rightarrow \mathcal{H}'$
4: **return** $\mathcal{H}'$

$\mathcal{H}$ constitutes family $\mathcal{F} = \{S_1, S_2, \ldots, S_{|\mathcal{U}|}\}$, where $S_h = \{t_{ij} : \{n_i, n_j\} \subseteq Nets(u_h)\}$, for $1 \leq h \leq |\mathcal{U}|$.

2) *Finding minimum set cover $\mathcal{F}'$ on $(\mathcal{T}, \mathcal{F})$.*

3) *Decoding $\mathcal{F}'$ to HS-n Solution $\mathcal{H}'$:* The subsets of $\mathcal{F}'$ comprises the node set $\mathcal{U}'$ of $\mathcal{H}'$ as $\mathcal{U}' = \{u_h \in \mathcal{U} : S_h \in \mathcal{F}'\}$.

The objective of finding a minimum number of subsets from $\mathcal{F}$ covering the main set corresponds to finding a node set set $\mathcal{U}' \subseteq \mathcal{U}$ of $\mathcal{H}$ minimizing the number of nodes $\mathcal{H}'$. The constraint of covering the main set ensures that $\mathrm{NIG}(\mathcal{H}') \equiv \mathrm{NIG}(\mathcal{H})$. Note that this combinatorial reduction preserves the optimality, i.e., an optimal solution to the constructed MSC Problem instance yields an optimal solution to the given HS-n Problem instance.

The minimum set cover problem is known to be NP-hard [30]. However, there is a well known $(\ln n)$-approximation algorithm [15], which works as follows: It grows a cover set using a sequence of greedy decisions. The greedy decision at each step is to select a subset that covers as many uncovered elements as possible. The algorithm terminates when all elements are covered.

For the sake of efficiency, Algorithm 6 is proposed which interleaves node selection operations with pin deletion operations.

In Algorithm 6, two successive for loops at lines 2 – 10 construct the list of nodes of $\mathcal{H}$ that cover each edge of $\mathcal{G}$ and compute the number of edges covered by each node. A priority queue $\mathcal{Q}$ which contains all nodes is built in line 11, where nodes are keyed by the number of uncovered edges that they cover. The while loop in lines 12 – 26 repeatedly identifies and selects a node that covers the maximum number of uncovered edges and then updates the relevant data structures accordingly. Meanwhile, the

Algorithm 6 Node-Deletion-Oriented Sparsening Algorithm

Require: $\mathcal{H} = (\mathcal{U}, \mathcal{N})$, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
1: totalUncovered $\leftarrow 0$
2: **for each** edge $e_{ij} \in \mathcal{E}$ with $i \leq j$ **do**
3: NodeCoverList[e_{ij}] $\leftarrow \emptyset$
4: covered[e_{ij}] $\leftarrow$ FALSE
5: totalUncovered $\leftarrow$ totalUncovered $+1$
6: **for each** node $u_h \in \mathcal{U}$ **do**
7: key[u_h] $\leftarrow 0$
8: **for each** net pair $(n_i, n_j) \in Nets(u_h)$ with $i \leq j$ **do**
9: NodeCoverList[e_{ij}] $\leftarrow$ NodeCoverList[e_{ij}] $\cup \{u_h\}$
10: key[u_h] $\leftarrow$ key[u_h] $+1$
11: $\mathcal{Q} \leftarrow$ PriorityQueue($\{u_h : u_h$ is node of $\mathcal{H}\}$, key$)$
12: **while** totalUncovered > 0 **do**
13: $u_h \leftarrow$ EXTRACT-MAX($\mathcal{Q}$)
14: **for each** net $n_i \in Nets(u_h)$ **do**
15: DeleteFlag[pin_{ih}] $\leftarrow$ TRUE
16: **for each** net pair $(n_i, n_j) \in Nets(u_h)$ with $i \leq j$ **do**
17: NodeCoverList[e_{ij}] $\leftarrow$ NodeCoverList[e_{ij}] $-\{u_h\}$
18: **if** covered[e_{ij}] $=$ FALSE **then**
19: covered[e_{ij}] $\leftarrow$ TRUE
20: totalUncovered $\leftarrow$ totalUncovered -1
21: **for each** node $u_{h'} \in$ NodeCoverList[e_{ij}] **do**
22: key[$u_{h'}$] $\leftarrow$ key[$u_{h'}$] -1 ▷ DECREASE-KEY operation
23: DeleteFlag[pin_{ih}] $\leftarrow$ DeleteFlag[pin_{jh}] $\leftarrow$ FALSE
24: **for each** net $n_i \in Nets(u_h)$ **do**
25: **if** DeleteFlag[pin_{ih}] $=$ TRUE **then**
26: delete pin pin_{ih}
27: **for each** node $u_h \in \mathcal{Q}$ **do**
28: delete node u_h

pins of the selected node are processed for deletion with respect to set of previously selected and processed nodes. After detecting a full coverage of the edges of graph $\mathcal{G}$, the set of nodes remaining in the priority queue are deleted in lines 27 – 28. This algorithm can also be made to run in $O(\sum_{u_h \in \mathcal{H}} |Nets(u_h)|^2)$ time through a priority queue implementation with $O(1)$–time extract-max and decrease-key operations. A sorted linear array implementation can achieve the desired bounds for those operations by exploiting the bounded integer key values and decrement-type decrease-key operations.

4.2.2 Hypergraph Sparsening by Pin Deletion (HS-p)

This approach for hypergraph sparsening is also mentioned by Catalyurek **??**. Here, the NP-hardness of the problem is presented.

Problem 9 *Hypergraph Sparsening by Pin Deletion Problem (HS-p Problem): Given a hypergraph $\mathcal{H} = (\mathcal{U}, \mathcal{N})$, find a hypergraph $\mathcal{H}'$ minimizing the number of pins of $\mathcal{H}'$ such that* $\mathrm{NIG}(\mathcal{H}') \equiv \mathrm{NIG}(\mathcal{H})$.

Theorem 8 *The HS-p Problem is NP-hard.*

Proof Consider the decision version of the HS-p Problem. For the NP-completeness proof, a reduction from Set Cover Problem, which is decision version of Minimum Set Cover Problem, is used. For a given Set Cover Instance $(\mathcal{T}, \mathcal{F})$, where $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ of m elements and $\mathcal{F} = \{S_1, S_2, \ldots, S_n\}$ of m subsets, the HS-p Instance $\mathcal{H} = (\mathcal{U}, \mathcal{N})$ is constructed as $\mathcal{U} = \{u_1, u_2, \ldots, u_{n+1}\}$ and $\mathcal{N} = \{n_1, n_2, \ldots, n_{m+n+2}\}$. The pin construction is done as follows:

- $Pins(n_1) = \{u_1, u_2, \ldots, u_{n+1}\}$.
- $Pins(n_{i+1}) = \{u_j : t_i \in S_j\}$, for $1 \leq i \leq m$.
- $Pins(n_{m+i+1}) = \{u_i, u_{n+1}\}$, for $1 \leq i \leq n$.
- $Pins(n_{m+n+2}) = \{u_{n+1}\}$.

Let p denote the total number of elements of all subsets of $\mathcal{F}$. Take a HS-p solution $\mathcal{H}'$ with $\tau+2n+p+2$ nodes. All pins except pins $u_1, u_2, \ldots, u_n$ of net n_1 should exist in $\mathcal{H}'$, because each of them has at least one unique edge $e \in \mathrm{NIG}(\mathcal{H})$ that can not be constructed without that node. The remaining uncovered m edges in graph $\mathrm{NIG}(\mathcal{H})$ corresponds to m elements of the main set of SC Instance. By the construction of hypergraph $\mathcal{H}$, for the remaining uncovered m edges, a pin u_i of net n_1 can cover edge e_j if and only if subset S_i includes element t_j, for $1 \leq i \leq n$ and $1 \leq j \leq m$. In this construction, there exists a hypergraph $\mathcal{H}'$ such that $NIG(\mathcal{H}') \equiv NIG(\mathcal{H})$

with size smaller or equal to $\tau + 2n + p + 2$ if and only if there exists a set cover $\mathcal{F}'$ with size smaller or equal to τ. Therefore, the decision version of HS-p problem is NP-complete which implies that HS-p problem is NP-hard. □

4.3 Hypergraph Construction for Bipartite Graphs

$\mathcal{H}^2$ can be considered as dual hypergraph of 2-pin hypergraph representation, in which all edges are considered as nets with two pins, of the given graph $\mathcal{G}$.As mentioned in Section 4.1.2, $\mathcal{H}^2$ is uniquely defined for a given graph $\mathcal{G}$. Furthermore, if $\mathcal{G}$ is a bipartite graph, $\mathcal{H}^2$ is the unique such hypergraph, which is stated as following theorem.

Theorem 9 *Given a bipartite graph $\mathcal{G}$, $\mathcal{H}^2$ is the only hypergraph $\mathcal{H}$, whose NIG representation is equal to $\mathcal{G}$, i.e.,* $\mathrm{NIG}(\mathcal{H}) \equiv \mathcal{G}$.

Proof Since $\mathcal{G}$ is a bipartite graph, it contains no 3-cliques. Therefore C^2 is the only ECC C of $\mathcal{G}$. From the Theorem 5, $\mathcal{H}^2$ is the only hypergraph $\mathcal{H}$ such that $\mathrm{NIG}(\mathcal{H}) \equiv \mathcal{G}$. □

4.4 Matrix-Theoretical View

Here, the association between the graph-theoretic and matrix-theoretic views of the HP-based GPVS formulation is revealed. Given a $p \times p$ symmetric and square matrix M, let $\mathcal{G}(M) = (\mathcal{V}, \mathcal{E})$ denote the standard graph representation of matrix M.

In graph-theoretic discussion given in Section 4.1, we are looking for a hypergraph $\mathcal{H}$ whose NIG representation is equivalent to $\mathrm{G}(M)$. In matrix-theoretic view, this corresponds to looking for a structural factorization $M = AA^T$ of matrix M, where A is a $p \times q$ rectangular matrix. Here, structural factorization refers to the fact that $A = \{a_{ij}\}$ is a {0,1}-matrix, where AA^T determines the sparsity patterns of M. In this factorization, the rows of matrix A correspond to the vertices of $\mathrm{G}(M)$ and

the set of columns of matrix A determines an ECC $\mathcal{C}$ of $\mathrm{G}(M)$. So, matrix A can be considered as a clique incidence matrix of $\mathrm{G}(M)$. That is, column c_h of matrix A corresponds to a clique $\mathcal{C}_h$ of $\mathcal{C}$, where $a_{ih} \neq 0$ implies that vertex $v_i \in \mathcal{C}_h$. The row-net hypergraph model $\mathrm{H_{RN}}(A)$ of matrix A is equivalent to the clique-node hypergraph of graph $\mathrm{G}(M)$ for the ECC $\mathcal{C}$ determined by the columns of A, i.e., $\mathrm{H_{RN}}(A) \equiv \mathrm{CNH}(\mathrm{G}(M), \mathcal{C})$. In other words, the NIG representation of row-net hypergraph model $\mathrm{H_{RN}}(A)$ of matrix A is equivalent to $\mathrm{G}(M)$, i.e., $\mathrm{NIG}(\mathrm{H_{RN}}(A)) \equiv \mathrm{G}(M)$.

An SB form A_{SB} of A induces a DB form M_{DB} of M, since multiplying A_{SB} with its transpose produces a DB form of M. That is,

$$
\begin{aligned}
A_{SB}A_{SB}^T &= \begin{bmatrix} A_1 & & \\ & \ddots & \\ & & A_K \\ A_{B_1} & \dots & A_{B_K} \end{bmatrix} \begin{bmatrix} A_1^T & & A_{B_1}^T \\ & \ddots & \vdots \\ & A_K^T & A_{B_K}^T \end{bmatrix} \\
&= \begin{bmatrix} A_1A_1^T & & & A_1A_{B_1}^T \\ & \ddots & & \vdots \\ & & A_KA_K^T & A_KA_{B_1}^T \\ A_{B_1}A_1^T & \dots & A_{B_K}A_K^T & \sum_k A_{B_k}A_{B_k}^T \end{bmatrix} \qquad (4.1) \\
&= \begin{bmatrix} M_1 & & & M_{B_1}^T \\ & \ddots & & \vdots \\ & & M_K & M_{B_K}^T \\ M_{B_1} & \dots & M_{B_K} & M_B \end{bmatrix} = M_{DB} \qquad (4.2)
\end{aligned}
$$

As seen in (4.2), the number of rows/columns in the square diagonal block $A_kA_k^T$ of M_{DB} is equal to the number of rows of the rectangular diagonal block A_k of A_{SB}. Furthermore, the number of coupling rows/columns in M_{DB} is equal to the number of coupling rows in A_{SB}. So, minimizing the number of coupling rows in A_{SB} corresponds to minimizing the number of coupling rows/columns in M_{DB}, whereas balancing on row counts of the rectangular diagonal submatrices in A_{SB} infers balance among the row/column counts of the square diagonal submatrices in M_{DB}. Thus, given a structural factorization $M = AA^T$ of matrix M, the proposed HP-based GPVS

formulation corresponds to formulating the problem of permuting M into a DB block diagonal form as an instance of the problem of permuting A into an SB block diagonal form.

Chapter 5

Graph Partitioning by Vertex Separator-based Hypergraph Partitioning

Here, we are interested in the hypergraph partitioning (HP) problem in which objective is defined according to the cutnet metric as in Equation 2.5 and the constraint of ε-balance is considered on the internal nets. The objective and constraint of Graph Partitioning by Vertex Separator (GPVS) problem is stated in Section 2.1.1. Theorem 3 forms the basis of GPVS-based hypergraph partitioning approach. The theorem states that a K-way narrow vertex separator $\Pi_{VS} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K; \mathcal{V}_S\}$ of $\mathrm{NIG}(\mathcal{H})$ can be decoded as a K-way node-partition $\Pi_{HP} = \{\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_K\}$ of $\mathcal{H}$, where the objective of minimizing the separator size of Π_{VS} corresponds to minimizing the cutsize of Π_{HP} and the constraint of ε-$balance$ of Π_{VS} infers ε-$balance$ of Π_{HP}. Here, we try to solve HP problem by using GPVS problem. So, we are given a hypergraph $\mathcal{H}$ and we are to find a node-partition Π_{HP} of the hypergraph $\mathcal{H}$. Therefore, the approach is much more related to Corollary 2, which states that a K-way narrow vertex separator Π_{VS} of graph $\mathcal{G} = \mathrm{NIG}(\mathcal{H})$ can be decoded as a K-way node separator Π_{HP} of $\mathcal{H}$, where the objective of minimizing the separator size of Π_{VS} corresponds to minimizing the cutsize of Π_{HP} and the constraint of ε-$balance$ of Π_{VS} infers ε-$balance$ of Π_{HP}.

The definition of HP problem in Section 2.2.1 is reshowed below:

Problem: *HP Problem: Given a hypergraph $\mathcal{H} = (\mathcal{U}, \mathcal{N})$, an imbalance parameter ε and an integer K, find a K-way node partition Π_{HP} of $\mathcal{H}$, minimizing the $Cutsize(\Pi_{HP})$ such that Π_{HP} has ε-balance.* To solve the HP Problem, we use a combinatorial reduction to GPVS Problem instead of GPVSn Problem.

Combinatorial Reduction 3 *HP Problem $\Rightarrow$ GPVS Problem*

Algorithm 7 *HP Problem $\Rightarrow$ GPVS Problem Reduction*

Require: $\mathcal{H} = (\mathcal{U}, \mathcal{N}), \varepsilon, K$
1: $\mathcal{H} \rightarrow \mathcal{G}$
2: $\Pi_{VS} \leftarrow GPVS(\mathcal{G}, \varepsilon, K)$
3: $\Pi_{VS} \rightarrow \Pi_{HP}$
4: **return** Π_{HP}

1) *Constructing the Net Intersection Graph $\mathcal{G}$:* This is straightforward by just taking Net Intersection Graph of $\mathcal{H}$.

2) *Finding K-way Vertex Separator Π_{VS} with minimum separator size under ε-balance*

3) *Decoding Π_{VS} to node-partition Π_{HP}:* This is not trivial because given the vertex separator Π_{VS} of $\mathcal{G}$, there is no unique and easy construction of Π_{HP} of $\mathcal{H}$ if the separator is not narrow. The approaches for decoding to a node-partition Π_{HP} of $\mathcal{H}$ is explained in this chapter in detail.

Figure 5 shows the combinatorial reduction from GPVS problem to HP problem as follows. In Figure 5.1(a), firstly we are given a hypergraph $\mathcal{H}$. In the second step, we construct the net intersection graph $\mathcal{G} = \mathrm{NIG}(\mathcal{H})$ as in Figure 5.1(b). Then, in the third step, we find a vertex separator Π_{VS} of $\mathcal{G}$ as in Figure 5.1(c) Finally, in the forth step, we decode Π_{VS} as a hypergraph partition Π_{HP} of the given hypergraph $\mathcal{H}$ as in Figure 5.1(d).

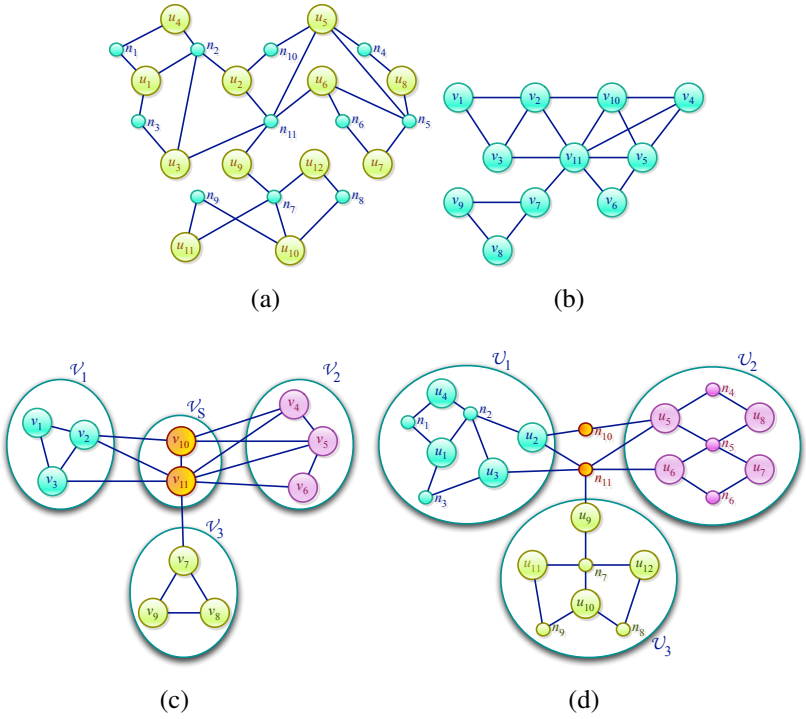


Figure 5.1: (a) An example graph $\mathcal{H}$ (b) The net intersection graph $\mathcal{G} = \mathrm{NIG}(\mathcal{H})$ (c) A vertex separator Π_{VS} of $\mathcal{G}$ (d) The hypergraph partition Π_{HP} of $\mathcal{H}$ induced by Π_{VS} of $\mathcal{G}$

5.1 Hypergraph Partition Construction

For a given hypergraph $\mathcal{H}$, we obtain a vertex separator $\Pi_{VS} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K; \mathcal{V}_S\}$ of graph $\mathcal{G} = \mathrm{NIG}(\mathcal{H})$. Π_{VS} refers to a $(K+1)$-way net-partition $\Pi_{HP_N} = \{\mathcal{N}_1 \equiv \mathcal{V}_1, \mathcal{N}_2 \equiv \mathcal{V}_2, \ldots, \mathcal{N}_K \equiv \mathcal{V}_K; \mathcal{N}_S \equiv \mathcal{V}_S\}$ of $\mathcal{H}$. However, in hypergraph partitioning the set $\mathcal{U}$ is partitioned and a K-way node-partition $\Pi'_{HP_U} = \{\mathcal{U}'_1, \mathcal{U}'_2, \ldots, \mathcal{U}'_K\}$ induces a $(K+1)$-way net-partition $\Pi'_{HP_N} = \{\mathcal{N}'_1, \mathcal{N}'_2, \ldots, \mathcal{N}'_K; \mathcal{N}'_S\}$ of $\mathcal{H}$, where $\mathcal{N}'_k = \{n_j \in \mathcal{N} : Pins(n_j) \subseteq \mathcal{U}'_k\}$ for $1 \leq k \leq K$, $\mathcal{N}'_S = \mathcal{N} - \bigcup_{1 \leq k \leq K} \mathcal{N}'_k$. Unfortunately, for a given net-partition Π_{HP_N} it is not guaranteed that there exists a node-partition Π_{HP_U} such as to induce Π_{HP_N}. Even worse, to find whether there exists such a node-partition is an NP-complete problem.

5.1.1 Theoretical Base

Problem 10 *Hypergraph Net Decoding (HND) Problem: Given a hypergraph* $\mathcal{H} = (\mathcal{U}, \mathcal{N})$ *and a* $(K+1)$*-way net-partition* Π_{HP_N} *of* $\mathcal{H}$*, does there exists a* K*-way node-partition* Π_{HP_U} *of* H *such that* Π_{HP_U} *induces* Π_{HP_N}*?*

Theorem 10 *The HND Problem is NP-Complete.*

Proof For the NP-completeness proof, a reduction from Set Splitting Problem, which is a decision version of Maximum Set Splitting Problem, is used. For a given Set Splitting Instance $(\mathcal{T}, \mathcal{F})$, where $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ of m elements and $\mathcal{F} = \{S_1, S_2, \ldots, S_n\}$ of n subsets, the HND Instance $(\mathcal{H} = (\mathcal{U}, \mathcal{N}), K, \Pi_{HP_N})$ is constructed as $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$, $\mathcal{N} = \{n_1, n_2, \ldots, n_n\}$ where $Pins(n_j) = \{u_i \in \mathcal{U} : t_i \in S_j\}$, $K = 2$ and $\Pi_{HP_N} = \{\mathcal{N}_1 = \emptyset, \mathcal{N}_2 = \emptyset; \mathcal{N}_S = \mathcal{N}\}$. If there exists a node-partition Π_{HP_N} of $\mathcal{H}$, then a partition (T_1, T_2) of the main set $\mathcal{T}$ can be constructed as $T_1 = \{t_i : u_i \in \mathcal{U}_1\}$ and $T_2 = \{t_i : u_i \in \mathcal{U}_2\}$. Note that a net n_j is in cut, i.e., $n_j \in \mathcal{N}_S$ if and only if the subset S_j of $\mathcal{F}$ is splitted. Since in Π_{HP_N} is given as all nets are in cut, it implies that all subsets of $\mathcal{F}$ are splitted. If there exists a partition of $\mathcal{T}$ into (T_1, T_2), then than node-partition $\Pi_{HP_U} = \{\mathcal{U}_1, \mathcal{U}_2\}$ of $\mathcal{H}$ is constructed as $\mathcal{U}_1 = \{u_i : t_i \in T_1\}$ and $\mathcal{U}_2 = \{u_i : t_i \in T_2\}$ which will induce the net-partition Π_{HP_N}. □

Definition 1 (Partial Node Partition) *Given a hypergraph* $\mathcal{H} = (\mathcal{U}, \mathcal{N})$ *and a* $(K+1)$*-way net-partition* $\Pi_{HP_N} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$ *of* $\mathcal{H}$*, the partial node-partition* $\Pi^p_{HP_U} = \{\mathcal{U}^p_1, \mathcal{U}^p_2, \ldots, \mathcal{U}^p_K\}$ *of* $\mathcal{H}$ *for* Π_{HP_N} *is defined as the node-partition constructed as* $\mathcal{U}^p_k = Pins(\mathcal{N}_k)$*, for* $1 \leq k \leq K$*. Furthermore* $\mathcal{U}^p$ *is defined as the nodes in partial node-partition* $\Pi^p_{HP_U}$*.*

Definition 2 (Free Node) *Given a hypergraph* $\mathcal{H} = (\mathcal{U}, \mathcal{N})$*, a node* $u_i \in \mathcal{U}$ *is said to be* free *for a* $(K+1)$*-way net-partition* $\Pi_{HP_N} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$ *of* $\mathcal{H}$*, if all nets of node* u_i *is an element of* $\mathcal{N}_S$*.*

Let the free set $\mathcal{U}_F$ of the hypergraph $\mathcal{H}$ for a given net-partition Π_{HP_N} define the set of all free nodes of $\mathcal{H}$ for Π_{HP_N}. Note that, $\mathcal{U} = \mathcal{U}^p \cup \mathcal{U}_F$ and $\mathcal{U}^p \cap \mathcal{U}_F = \emptyset$. Thus, for a hypergraph $\mathcal{H}$, a K-way net-partition $\Pi_{HP_N} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$ can be thought as to induce a $(K+1)$-way node-partition $\Pi_{HP} = \{\mathcal{U}_1^p, \mathcal{U}_2^p, \ldots, \mathcal{U}_K^p; \mathcal{U}_F\}$.

Definition 3 (Free Node Set of Net) *Given a hypergraph $\mathcal{H} = (\mathcal{U}, \mathcal{N})$ and a $(K+1)$-way net-partition $\Pi_{HP_N} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$ of $\mathcal{H}$, the free node set $\mathcal{F}_j$ of a net $n_j \in \mathcal{N}$ for Π_{HP_N} is defined as the set of free nodes of net n_j for Π_{HP_N}, i.e., $\forall n_j \in \mathcal{N}, \mathcal{F}_j = \mathcal{U}_F \cap Pins(n_j)$.*

Definition 4 (Free Node Assignment) *Given a hypergraph $\mathcal{H} = (\mathcal{U}, \mathcal{N})$ and a $(K+1)$-way net-partition $\Pi_{HP_N} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$ of $\mathcal{H}$, a free node assignment $f : \mathcal{U}_F \rightarrow [1:K]$ is a function from the free node set $\mathcal{U}_F$ of $\mathcal{H}$ for Π_{HP_N} to part indices $[1:K]$.*

Definition 5 (Complete Node/Net Partition) *Given a hypergraph $\mathcal{H} = (\mathcal{U}, \mathcal{N})$, a $(K+1)$-way net-partition $\Pi_{HP_N} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$ of $\mathcal{H}$ and a free node assignment f, the complete node-partition $\Pi_{HP_U}^f = \{\mathcal{U}_1^f, \mathcal{U}_2^f, \ldots, \mathcal{U}_K^f\}$ of $\mathcal{H}$ for Π_{HP_N} induced by f is defined as the node-partition constructed as $\mathcal{U}_k^f = \mathcal{U}_k^p \cup \{u_i \in \mathcal{U}_F : f(u_i) = k\}$, for $1 \leq k \leq K$. The complete net-partition $\Pi_{HP_N}^f = \{\mathcal{N}_1^f, \mathcal{N}_2^f, \ldots, \mathcal{N}_K^f; \mathcal{N}_S^f\}$ of $\mathcal{H}$ for Π_{HP_N} induced by f is defined as the net-partition induced by the complete node-partition $\Pi_{HP_U}^f$ induced by f.*

Proposition 2 *Given a $(K+1)$-way net-partition $\Pi_{HP_N} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$ of a given hypergraph $\mathcal{H}$, for the complete net-partition $\Pi_{HP_N}^f$ produced by any free-node assignment f, we have $\mathcal{N}_S^f \subseteq \mathcal{N}_S$.*

Proof Assume the contrary. Take a free-node assignment f for the induced complete net-partition $\Pi_{HP_N}^f$, there exists a net $n_j \in \mathcal{N}_S^f$ but $n_j \notin \mathcal{N}_S$. Then there exists a net-part $\mathcal{N}_k$ of Π_{HP_N} such that $n_j \in \mathcal{N}_k$, which implies that $Pins(n_j) \subseteq \mathcal{U}_k^p$, where $\mathcal{U}_k^p$ is a node part of partial node partition $\Pi_{HP_U}^p$. From the definition of the complete node-partition $\Pi_{HP_U}^f$ induced by the free node assignment f, $\mathcal{U}_k^p \subseteq \mathcal{U}_k^f$ which yields

$Pins(n_j) \subseteq \mathcal{U}_k^f$. It implies that $n_j \in \mathcal{N}_k^f$ which contradicts with the assumption $n_j \in \mathcal{N}_S^f$. □

Corollary 3 *Given a hypergraph $\mathcal{H}$ and an optimal vertex separator $\Pi_{VS} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K; \mathcal{V}_S\}$ of $\mathcal{G} = \mathrm{NIG}(\mathcal{H})$ with ε-balance, let $\Pi_{HP_N} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$ be the $(K+1)$-way net-partition of $\mathcal{H}$ induced by Π_{VS}. If there exists no node-partition Π_{HP_U} which induces the net-partition Π_{HP_N} then for any free node assignment f, the basic imbalance ε_0^f of the complete net-partition $\Pi_{HP_N}^f$ induced by f, is larger than ε, i.e., $\varepsilon_0^f > \varepsilon$.*

The Proposition 2 and Corollary 3 disclose the rationale behind the objective of Free Node Assignment (FNA) Problem, which is defined below. That is, they explain why the objective of FNA is minimizing the basic imbalance instead of minimizing the cut with the given ε-balance.

Problem 11 *Free Node Assignment Problem (FNA Problem): Given a hypergraph $\mathcal{H} = (\mathcal{U}, \mathcal{N})$ and a $(K+1)$-way net-partition $\Pi_{HP_N} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$ of $\mathcal{H}$, find a free node assignment $f : \mathcal{U}_F \rightarrow [1:K]$, minimizing the basic imbalance ε_0^f of the complete net-partition $\Pi_{HP_N}^f$ induced by f.*

Theorem 11 *FNA Problem is NP-Hard.*

Proof Consider the decision version of FNA Problem in which the instance is $(\mathcal{H}, K, \Pi_{HP_N}, \varepsilon)$ and the problem asks for the existence of a free node assignment f where induced complete net-partition $\Pi_{HP_N}^f$ has ε-balance. For the NP-completeness proof, a reduction from Hypergraph Net Decoding Problem is used. For a given HND instance $(\mathcal{H}, K, \Pi_{HP_N})$, the FNA instance is $(\mathcal{H}, K, \Pi_{HP_N}, \varepsilon_0)$ with unit net costs and ε_0 is the basic imbalance of the given net-partition $\Pi_{HP_N}^f$. If there exists a node partition Π_{HP_U} which induces the net-partition Π_{HP_N} then there exists a free node assignment f where $\Pi_{HP_U}^f = \Pi_{HP_U}$ and so $\Pi_{HP_N}^f = \Pi_{HP_N}$ implying that there exists a free node assignment f, $\Pi_{HP_N}^f$ has ε_0-balance. From the Corollary 3, if there exists no node-partition Π_{HP_U} inducing the net-partition Π_{HP_N} then there exists no free node assignment f such that $\Pi_{HP_N}^f$ has ε_0-balance. □

Table 5.1: Partition of $\mathcal{N}_S$ of a net-partition Π_{HP_N}

	$\mathcal{F}_j = 0$	$\mathcal{F}_j = 1$	$\mathcal{F}_j \geq 2$
$\lambda_j^p = 0$	$\mathcal{N}_\emptyset$	$\mathcal{N}_{I_2}$	$\mathcal{N}_F$
$\lambda_j^p = 1$	$\mathcal{N}_{I_1}$	$\mathcal{N}_P$	$\mathcal{N}_P$
$\lambda_j^p \geq 2$	$\mathcal{N}_E$	$\mathcal{N}_E$	$\mathcal{N}_E$

Consider the hypergraph $\mathcal{H} = (\mathcal{U}, \mathcal{N})$ and a net-partition Π_{HP_N} of $\mathcal{H}$. Let partial node connectivity λ_j^p and partial node connectivity set Λ_j^p respectively denote the connectivity and connectivity set of a net $n_j \in \mathcal{N}$ according to the partial node-partition $\Pi_{HP_U}^p$ of $\mathcal{H}$ for Π_{HP_N}.

Table 5.1.1 presents which set any net n_j in $\mathcal{N}_S$ lies into as discussed in Proposition 3.

Proposition 3 *Given a hypergraph $\mathcal{H} = (\mathcal{U}, \mathcal{N})$ and a net-partition $\Pi_{HP_N} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$ of $\mathcal{H}$, consider the partial node/net partition $\Pi_{HP_U}^p$ and $\Pi_{HP_N}^p$ of $\mathcal{H}$ for Π_{HP_N}. For any free node assignment f, the net set $\mathcal{N}_S$ of the given net-partition Π_{HP_N} is the composite of the following disjoint net sets, i.e., $\mathcal{N}_S = \mathcal{N}_\emptyset \cup \mathcal{N}_{I_1} \cup \mathcal{N}_{I_2} \cup \mathcal{N}_E \cup \mathcal{N}_F \cup \mathcal{N}_P$, where;*

- $\mathcal{N}_\emptyset = \{n_j \in \mathcal{N}_S : F_j = \emptyset$ *and* $\lambda_j^p = 0\}$
- $\mathcal{N}_{I_1} = \{n_j \in \mathcal{N}_S : F_j = \emptyset$ *and* $\lambda_j^p = 1\}$
- $\mathcal{N}_{I_2} = \{n_j \in \mathcal{N}_S : |F_j| = 1$ *and* $\lambda_j^p = 0\}$
- $\mathcal{N}_E = \{n_j \in \mathcal{N}_S : \lambda_j^p \geq 2\}$
- $\mathcal{N}_F = \{n_j \in \mathcal{N}_S : |F_j| \geq 2$ *and* $\lambda_j^p = 0\}$
- $\mathcal{N}_P = \{n_j \in \mathcal{N}_S : |F_j| \geq 1$ *and* $\lambda_j^p = 1\}$

Proof Since a net $n_j \in \mathcal{N}_S$ is exactly one of the sets of $\mathcal{N}_\emptyset$, $\mathcal{N}_{I_1}$, $\mathcal{N}_{I_2}$, $\mathcal{N}_E$, $\mathcal{N}_F$ and $\mathcal{N}_P$, the sets are disjoint and covers $\mathcal{N}_S$. □

Theorem 12 *Given a hypergraph* $\mathcal{H} = (\mathcal{U}, \mathcal{N})$ *and a net-partition* $\Pi_{HP_N} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$ *of* $\mathcal{H}$*, consider the partial node partition* $\Pi^p_{HP_U}$ *of* $\mathcal{H}$ *for* Π_{HP_N}*. For any free node assignment* f*;*

(i) $\mathcal{N}^f = \mathcal{N}$

(ii) $\mathcal{N}_k \subseteq \mathcal{N}^f_k$*, for* $1 \leq k \leq K$

(iii) $\mathcal{N}^f_S \subseteq \mathcal{N}_S$

(iv) $\mathcal{N}_\emptyset = \emptyset$

(v) $\mathcal{N}_{I_1} \cap \mathcal{N}^f_S = \emptyset$

(vi) $\mathcal{N}_{I_2} \cap \mathcal{N}^f_S = \emptyset$

(vii) $\mathcal{N}_E \subseteq \mathcal{N}^f_S$

Proof of the Theorem 12 is given at the Appendix. As the result of the Theorem 12, a net of $\mathcal{N}_{I_1} \cup \mathcal{N}_{I_2}$ and $\mathcal{N}_E$ is inevitably internal and external in complete net partition, respectively. The remaining nets of $\mathcal{N}_S$ constitute $\mathcal{N}_F \cup \mathcal{N}_P$ and we call a net in this set as *preservable net*.

Lemma 1 *Given a hypergraph* $\mathcal{H} = (\mathcal{U}, \mathcal{N})$ *and a vertex separator* Π_{VS} *of* $\mathcal{G} = \mathrm{NIG}(\mathcal{H}) = (\mathcal{V}, \mathcal{E})$*, for a net* $n_j \in \mathcal{N}$*, the partial node connectivity* λ^p_j *of the net* n_j *of* $\mathcal{H}$ *for the net-partition* Π_{HP_N} *which is induced by* Π_{VS} *and the connectivity of the corresponding vertex* v_j *of* $\mathcal{G}$ *are equal, i.e.,* $\lambda^p_j = \lambda_j$*.*

Proof Consider a net $n_{j_1} \in \mathcal{N}$ and the corresponding vertex $v_{j_1} \in \mathcal{V}$. Part $\mathcal{U}^p_k$ of $\Pi^p_{HP_U}$ is in the partial connectivity set of a net $n_{j_1} \in \mathcal{N}$ for Π_{HP_N} if and only if there exists a node $u_i \in Pins(n_{j_1})$ that lies in part $\mathcal{U}^p_k$, i.e. $\mathcal{U}^p_k \cap Pins(n_{j_1}) \neq \emptyset$. Part $\mathcal{V}_k$ of Π_{VS} is in the connectivity set of a vertex $v_{j_1} \in \mathcal{V}$ if and only if there exists a vertex $v_{j_2} \in Adj(v_{j_1}) \cup \{v_{j_1}\}$ that lies in part $\mathcal{V}_k$, i.e. $\mathcal{V}_k \cap (Adj(v_{j_1}) \cup \{v_{j_1}\}) \neq \emptyset$. If there exists a node $u_i \in Pins(n_{j_1})$ that lies in part $\mathcal{U}^p_k$ then there exists a net $n_{j_2} \in \mathcal{N}_k$ (Theorem 12(ii)) such that $u_i \in Pins(n_{j_2})$ which implies that there exists a vertex

$v_{j_2} \in \mathcal{V}_k$ such that v_{j_1} is adjacent or equal to v_{j_2}, i.e., $v_{j_2} \in Adj(v_{j_1}) \cup \{v_{j_1}\}$. If there exists a vertex $v_{j_2} \in Adj(v_{j_1}) \cup \{v_{j_1}\}$ that lies in part $\mathcal{V}_k$ then there exists a node $u_i \in \mathcal{U}$ which is both connected to nets n_{j_1} and n_{j_2}. Since $n_{j_2} \in \mathcal{N}_k$ and $u_i \in Pins(n_{j_2})$, $u_i \in \mathcal{U}_k^p$. □

The following theorem presents that if we obtain a narrow vertex separator, then hypergraph partitioning decoding exists and trivial.

Theorem 13 *Given a hypergraph $\mathcal{H} = (\mathcal{U}, \mathcal{N})$ and a vertex separator Π_{VS} of $\mathcal{G} = \mathrm{NIG}(\mathcal{H}) = (\mathcal{V}, \mathcal{E})$, if Π_{VS} is narrow then for any free node assignment f, consider the net-partition Π_{HP_N} induced by vertex separator Π_{VS}. The complete node-partition $\Pi^f_{HP_U}$ of $\mathcal{H}$ for Π_{HP_N} induced by f, induces the net-partition Π_{HP_N}, i.e., $\Pi^f_{HP_N} = \Pi_{HP_N}$.*

Proof If Π_{VS} is narrow, then for any vertex $v_j \in \mathcal{V}_S$, $\lambda_j \geq 2$ and correspondingly for any net $n_j \in \mathcal{N}_S$, $\lambda^p_j \geq 2$ which implies that $\mathcal{N}_E = \mathcal{N}_S$. Since $\mathcal{N}_S = \mathcal{N}_E \subseteq \mathcal{N}^f_S$ (Theorem 12(vii)) and $\mathcal{N}^f_S \subseteq \mathcal{N}_S$ (Theorem 12(iii)), $\mathcal{N}^f_S = \mathcal{N}_S$. Since $\mathcal{N}_k \subseteq \mathcal{N}^f_k$ (Theorem 12(ii)), $\mathcal{N}^f = \mathcal{N}$ (Theorem 12(i)) and $\mathcal{N}^f_S = \mathcal{N}_S$, $\mathcal{N}_k = \mathcal{N}^f_k$. Therefore $\Pi^f_{HP_N} = \Pi_{HP_N}$. □

The following sets the basis for hypergraph partition construction algorithms.

Theorem 14 *Given a hypergraph $\mathcal{H} = (\mathcal{U}, \mathcal{N})$, a net-partition Π_{HP_N} of $\mathcal{H}$ and a free node assignment f;*

(i) For a net $n_j \in N_F$, $n_j \in \mathcal{N}^f_S$ if and only if there exists two different free nodes with different assignments of free node set $\mathcal{F}_j$ of net n_j, i.e., $n_j \in \mathcal{N}^f_S \iff \exists_{i \neq j} u_i, u_j \in \mathcal{F}_j$ s.t. $f(u_i) \neq f(u_j)$.

(ii) For a net $n_j \in N_E$, $n_j \in \mathcal{N}^f_S$ if and only if there exists a free node with different assignment than the part that n_j is connected to, i.e., $n_j \in \mathcal{N}^f_S \iff \exists u_i \in \mathcal{F}_j$ s.t. $f(u_i) \neq k$, where $\Lambda^p_j = \{\mathcal{U}^p_k\}$.

Proof For a net $n_j \in N_F$, since the only nodes connected to net n_j are the nodes of $\mathcal{F}_j$, $n_j \in \mathcal{N}_S^f$ if and only if there exists two different free nodes with different assignments of free node set $\mathcal{F}_j$ of net n_j. For a net $n_j \in N_F$, since the all nodes connected to net n_j excluding the nodes of $\mathcal{F}_j$ are in part $\mathcal{U}_k$, the net n_j has a connectivity to part $\mathcal{U}_k^f$ of $\Pi_{HP_U}^f$, inevitably. Therefore, $n_j \in \mathcal{N}_S^f$ if and only if there exists a free node with different assignment than the part that n_j is connected to. □

5.1.2 Hypergraph Partition Construction Methodology

The approach for hypergraph partition construction comprises firstly constructing the partial node partition, secondly finding a free node assignment and lastly constructing the hypergraph partition as the complete node partition induced by the free node assignment. This hypergraph partition construction Methodology is presented as follows:

Algorithm 8 Hypergraph Partition Construction Methodology

Require: $\mathcal{H} = (\mathcal{U}, \mathcal{N})$, Π_{VS} of $\mathrm{NIG}(\mathcal{H})$
1: $\Pi_{VS} \rightarrow \Pi_{HP_N}$
2: $\Pi_{HP_N} \rightarrow \Pi_{HP_U}^p$
3: $f \leftarrow FNA(\mathcal{H}, \Pi_{HP_N})$
4: $\Pi_{HP} \leftarrow (\Pi_{HP_U}^p, f)$
5: **return** Π_{HP}

By this construction, the problem of constructing hypergraph partitioning Π_{HP} of $\mathcal{H}$ given a vertex separator Π_{VS} of $\mathcal{G} = \mathrm{NIG}(\mathcal{H})$ induces to the Free Node Assignment (FNA) Problem.

The quality of free node assignment directly affects the overall GPVS-based Hypergraph Partitioning quality. Considering the objective and constraint of HP Problem it seems that the goal of the FNA Problem should be finding a free node assignment that minimizes the cutsize of induced complete node partition such that the balance constraint is esteemed. However, we have some impossibility results on such partitions. Theorem 4 tells us that there doesn't exist a hypergraph partition with smaller cutsize holding the same balance constraint. Furthermore Corollary 3 implies that the minimum imbalance ratio of hypergraph partition can not fall under the imbalance ratio of

given optimal vertex separator. It also implies that the lower bound for the imbalance ratio can be realized, i.e., the desirable balance constraint can be realized only when the separator is preserved in cut as a whole. If it is not possible, then we will inevitably yield to a solution without conforming the desirable balance constraint. Nonetheless, we can try to minimize the imbalance disturbance, i.e., minimize the basic imbalance. The objective of preserving cut also conforms with the objective of minimizing the basic imbalance, so we can just use minimizing the basic imbalance as the objective of the Free Node Assignment Problem.

5.1.3 Hypergraph Partition Construction Algorithms

In this section, simple yet effective algorithms are proposed to solve Free Node Assignment Problem.

5.1.3.1 Randomized Approximation Algorithm

The proposed algorithm depends on the relation between the cutsize and imbalance (basic imbalance) of complete node partition for a free node assignment. When the external nets are preserved in the complete node partition, i.e., $\mathcal{N}_S = \mathcal{N}_S^f$, it is guaranteed that the imbalance is minimal. However, when the cutsize is maximized at complete node partition, it is not necessary that the imbalance is minimized.

Problem 12 *Maximum Cut Free Node Assignment (MC-FNA) Problem: Given a hypergraph $\mathcal{H} = (\mathcal{U}, \mathcal{N})$, a $(K+1)$-way net-partition $\Pi_{HP_N} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$ of $\mathcal{H}$ and a weight function $w : \mathcal{N} \rightarrow Z^+$, find a free node assignment $f : \mathcal{U}_F \rightarrow [1 : K]$, minimizing the cutsize $\sum_{n_j \in \mathcal{N}_S^f} c_j$ of the complete node partition $\Pi_{HP_U}^f$ of $\mathcal{H}$ for Π_{HP_N} induced by f.*

Although the reduction from FNA Problem to MC-FNA Problem is not perfect, the results will relate to each other, i.e., maximizing cutsize relates to minimizing the imbalance (basic imbalance).

Theorem 15 *MC-FNA Problem is NP-Hard.*

Proof Consider the decision version of MC-FNA Problem in which there is an additional parameter C and the problem asks for the existence of a free node assignment f where the cutsize of the induced complete node partition $\Pi^f_{HP_U}$ is smaller than or equal to C. For the NP-completeness proof, a reduction from HND Problem is used. Consider the HND Problem with instance $(\mathcal{H}, K, \Pi_{HP_N})$. Recall that the decision version of FNA Problem has an instance $(\mathcal{H}, K, \Pi_{HP_N}, \varepsilon)$ with unit net costs ε_0 is the basic imbalance of the given net-partition $\Pi^f_{HP_N}$. The decision version of FNA Problem with instance $(\mathcal{H}, K, \Pi_{HP_N}, \varepsilon)$ returns YES if and only if HND Problem with instance $(\mathcal{H}, K, \Pi_{HP_N})$ returns YES. As Corollary 3 implies that FNA Problem with instance $(\mathcal{H}, K, \Pi_{HP_N}, \varepsilon)$ returns YES if and only if MC-FNA Problem with instance $(\mathcal{H}, K, \Pi_{HP_N}, C_{max})$ returns YES, where $C_{max} = \sum_{n_j \in \mathcal{N}_S} c_j$. Thus, HND Problem with instance $(\mathcal{H}, K, \Pi_{HP_N})$ returns YES if and only if MC-FNA Problem with instance $(\mathcal{H}, K, \Pi_{HP_N}, C_{max})$ returns YES. □

The following algorithm is proposed to solve the FNA Problem.

Algorithm 9 Algorithm RANDOMASSIGN

Require: $\mathcal{H} = (\mathcal{U}, \mathcal{N})$, K, Π_{HP_N}

1: $\forall u_i \in \mathcal{U}_F$, $f(u_i) \leftarrow \begin{cases} 1 & wp\ 1/K \\ 2 & wp\ 1/K \\ \vdots & \vdots \\ K & wp\ 1/K \end{cases}$

2: **return** f

Theorem 16 *Algorithm RANDOMASSIGN is a Randomized* $\left(\frac{K-1}{K}\right)$*-approximation algorithm for the Maximum Cut Free Node Assignment (MC-FNA) Problem, where* K *is the number of parts of partitions.*

Proof As Proposition 3 and Theorem 12 state, for any free node assignment f, a net $n_j \in \mathcal{N}_{I_1} \cup \mathcal{N}_{I_2}$ is inevitably internal and a net $n_j \in \mathcal{N}_E$ is inevitably external. Let W_{opt} define the solution value for the MC-FNA Problem with the given parameters. Then,

$W_{max} = \sum_{n_j \in \mathcal{N}_F \cup \mathcal{N}_P \cup \mathcal{N}_E} c_j$ constitutes an upper-bound for W_{opt}, i.e., $W_{opt} \leq W_{max}$. We define a random variable $\hat{x}_j$ for each net $n_j \in \mathcal{N}_F \cup \mathcal{N}_P$ such that

$$\hat{x}_j = \begin{cases} 1, & \text{if } n_j \in \mathcal{N}_S^f \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

Then, the value of the cutsize of the complete node-partition induced by the free node assignment f produced by Algorithm RANDOMASSIGN is as follows:

$$W^f = \sum_{n_j \in \mathcal{N}_F \cup \mathcal{N}_P} c_j \hat{x}_j + \sum_{n_j \in \mathcal{N}_E} c_j \tag{5.2}$$

Then, the expected value $\mathrm{E}\{W\}$ can be computed as follows:

$$\begin{aligned} \mathrm{E}\{W^f\} &= \mathrm{E}\{ \sum_{n_j \in \mathcal{N}_F \cup \mathcal{N}_P} c_j \hat{x}_j + \sum_{n_j \in \mathcal{N}_E} c_j \} & (5.3) \\ &= \sum_{n_j \in \mathcal{N}_F \cup \mathcal{N}_P} c_j \mathrm{E}\{\hat{x}_j\} + \sum_{n_j \in \mathcal{N}_E} c_j & (5.4) \end{aligned}$$

For the expected value $\mathrm{E}\{\hat{x}_j\}$ of a random value $\hat{x}_j$ can be computed as follows:

$$\begin{aligned} \mathrm{E}\{\hat{x}_j\} &= 1.P\{\hat{x}_j = 1\} + 0.P\{\hat{x}_j = 0\} & (5.5) \\ &= P\{\hat{x}_j = 1\} & (5.6) \\ &= 1 - P\{\hat{x}_j = 0\} & (5.7) \end{aligned}$$

Regarding to Proposition 3, for a net $n_j \in \mathcal{N}_F$, $|\mathcal{F}_j| \geq 2$ and for a net $n_j \in \mathcal{N}_P$, $|\mathcal{F}_j| \geq 1$. Therefore, for a net $n_j \in \mathcal{N}_F$,

$$P\{\hat{x}_j = 0\} = K \frac{1}{K^{|\mathcal{F}_j|}} = \frac{1}{K^{|\mathcal{F}_j|-1}} \leq \frac{1}{K} \tag{5.8}$$

For a net $n_j \in \mathcal{N}_P$:

$$P\{\hat{x}_j = 0\} = K \frac{1}{K^{|\mathcal{F}_j|}} = \frac{1}{K^{|\mathcal{F}_j|}} \leq \frac{1}{K} \tag{5.9}$$

This implies the following:

$$\mathrm{E}\{\hat{x}_j\} \geq 1 - \frac{1}{K} \geq \frac{K-1}{K} \tag{5.10}$$

As a result:

$$\begin{aligned}
\mathrm{E}\{W^f\} &= \sum_{n_j \in \mathcal{N}_F \cup \mathcal{N}_P} c_j \mathrm{E}\{\hat{x}_j\} + \sum_{n_j \in \mathcal{N}_E} c_j && (5.11)\\
&\geq \sum_{n_j \in \mathcal{N}_F \cup \mathcal{N}_P} c_j \left(\frac{K-1}{K}\right) + \sum_{n_j \in \mathcal{N}_E} c_j && (5.12)\\
&\geq \left(\frac{K-1}{K}\right) \left(\sum_{n_j \in \mathcal{N}_F \cup \mathcal{N}_P} c_j + \sum_{n_j \in \mathcal{N}_E} c_j\right) && (5.13)\\
&= \left(\frac{K-1}{K}\right) W_{max} && (5.14)\\
&\geq \left(\frac{K-1}{K}\right) W_{opt} && (5.15)
\end{aligned}$$

For all free node assignments f, since $W^f \leq W^{opt}$, $\mathrm{E}\{W^f\} \leq W_{opt}$. Since $\left(\frac{K-1}{K}\right) W_{opt} \leq \mathrm{E}\{W^f\} \leq W_{opt}$, Algorithm RANDOMASSIGN is a Randomized $\left(\frac{K-1}{K}\right)$-approximation algorithm for the maximization problem Maximum Cut Free Node Assignment (MC-FNA) Problem. □

Note that, when K gets larger, the gap between solution of the algorithm and the optimal solution gets smaller. In the worst case of $K = 2$, the approximation ratio is $1/2$. Note that the complexity of the Algorithm RANDOMASSIGN is $\theta(N)$, where N denotes the number of free nodes.

5.1.3.2 Greedy Heuristic Algorithm

The proposed algorithm is a greedy heuristic that assigns a selected free node to the yet least loaded partition. The selection order is computed before the selection procedure starts and while selecting the free node, this order is strictly complied. This algorithm tries to solve the FNA Problem directly, i.e, tries to minimize the basic imbalance of the complete net-partition induced by the computed free node assignment. Let degree of a free node u_i be the number of nets connected to u_i, that still lies in the separator. The selection order is the non-increasing order of free nodes on their degrees.

Note that the complexity of the Algorithm GREEDYASSIGN is $\theta(K + P + Nlog(K))$, where N denotes the number of free nodes and P denotes the number

Algorithm 10 Algorithm GREEDYASSIGN

Require: $\mathcal{H} = (\mathcal{U}, \mathcal{N})$, K, Π_{HP_N}
1: $\mathcal{U}_F = \{u_i \in \mathcal{U} : Nets(u_i) \subseteq \mathcal{N}_S$
2: **for each** part $\mathcal{N}_k^f \in \Pi_{HP_N}^f$ **do**
3: partwgts[$\mathcal{N}_k^f$] $\leftarrow 0$
4: **for each** part $\mathcal{N}_k \in \Pi_{HP_N}$ **do**
5: **for each** net $n_j \in \mathcal{N}_k$ **do**
6: partwgts[$\mathcal{N}_k^f$] $\leftarrow$ partwgts[$\mathcal{N}_k^f$] $+c_j$
7: $\Lambda(n_j) \leftarrow \{\mathcal{N}_k^f\}$
8: **for each** net $n_j \in \mathcal{N}_S$ **do**
9: nfree[n_j] $\leftarrow |Pins(n_j) \cap \mathcal{U}_F|$
10: $\Lambda(n_j) \leftarrow \emptyset$
11: **for each** node $u_i \in Pins(n_j) - \mathcal{U}_F$ **do**
12: $\Lambda(n_j) \leftarrow \Lambda(n_j) \cup \{N_k^f : k = f(u_i)\}$
13: **for each** node $u_i \in \mathcal{U}_F$ **do**
14: degree[u_i] $\leftarrow |Nets(u_i) \cap \mathcal{N}_S|$
15: $\mathcal{Q} \leftarrow$ PriorityQueue($\{\mathcal{N}_k^f \in \Pi_{HP_N^f}\}$, partwgts)
16: **for each** node $u_i \in \mathcal{U}_F$ in non-decreasing order of degree values **do**
17: $\mathcal{N}_k^f \leftarrow$ EXTRACT-MAX($\mathcal{Q}$)
18: $f(u_i) \leftarrow k$
19: **for each** net $n_j \in Nets(u_i)$ **do**
20: **if** nfree[n_j] > 0 **then**
21: nfree[n_j] $\leftarrow$ degree[n_j] -1
22: $\Lambda(n_j) \leftarrow \Lambda(n_j) \cup \mathcal{N}_k^f$
23: **if** nfree[n_j] $= 0$ and $\Lambda(n_j) = \{\mathcal{N}_k^f\}$ **then**
24: partwgts[$\mathcal{N}_k^f$] $\leftarrow$ partwgts[$\mathcal{N}_k^f$] $+c_j$
25: DOWN-HEAPIFY($\mathcal{Q}, \mathcal{N}_k^f$)

of pins of the hypergraph $\mathcal{H}$.

5.2 Matrix-Theoretical View

Here, the association between the graph-theoretic and matrix-theoretic views of the GPVS-based HP formulation is revealed. Given a $p \times q$ rectangular matrix A, let $\mathcal{H} = \mathcal{H}_{RN}(A) = (\mathcal{U}, \mathcal{N})$ denote the row-net hypergraph representation of matrix A.

In graph-theoretic discussion given in Section 5.1, we are taking the NIG representation of the hypergraph $\mathcal{H}$. In matrix-theoretic view, this corresponds to the structural multiplication of matrix A with A^T, i.e., $M = AA^T$, where M is the $p \times p$

square and symmetric matrix. Here, structural multiplication refers to the fact that $A = \{a_{ij}\}$ is a {0,1}-matrix, where AA^T determines the sparsity patterns of M. In this scheme, the rows/columns of matrix M correspond to the nets of $\mathcal{H} = \mathcal{H}_{RN}(A)$. The standard graph model $\mathcal{G}(M)$ of matrix M is equivalent to the net intersection graph $\mathcal{G} = \mathrm{NIG}(\mathcal{H})$ of the hypergraph $\mathcal{H}$ which is the row-net hypergraph model $\mathcal{H}_{RN}(A)$ of matrix A, i.e., $\mathrm{NIG}(\mathrm{H_{RN}}(A)) \equiv \mathrm{G}(M)$.

A DB form M_{DB} of M induces an SB form A_{SB} of A.

$$M_{DB} = \begin{bmatrix} M_1 & & & M_{B_1}^T \\ & \ddots & & \vdots \\ & & M_K & M_{B_K}^T \\ M_{B_1} & \dots & M_{B_K} & M_B \end{bmatrix} \tag{5.16}$$

$$= \begin{bmatrix} A_1A_1^T & & & A_1A_{B_1}^T \\ & \ddots & & \vdots \\ & & A_KA_K^T & A_KA_{B_1}^T \\ A_{B_1}A_1^T & \dots & A_{B_K}A_K^T & (\sum_k A_{B_k}A_{B_k}^T) + A_BA_B^T \end{bmatrix}. \tag{5.17}$$

$$A_{SB} = \begin{bmatrix} A_1 & & & \\ & \ddots & & \\ & & A_K & \\ A_{B_1} & \dots & A_{B_K} & A_B \end{bmatrix} \tag{5.18}$$

As seen in (5.18), the number of rows/columns in the square diagonal block $A_kA_k^T$ of M_{DB} is equal to the number of rows of the rectangular diagonal block A_k of A_{SB}. Furthermore, the number of coupling rows/columns in M_{DB} is equal to the number of coupling rows in A_{SB}. So, minimizing the number of coupling rows/columns in M_{DB} corresponds to minimizing the number of coupling rows in A_{SB}, whereas balancing on row/column counts of the square diagonal submatrices in M_{DB} infers balance among the row counts of the rectangular diagonal submatrices in A_{SB}. Thus, the proposed GPVS-based HP formulation corresponds to formulating the problem of permuting A into a SB block diagonal form as an instance of the problem of permuting M into an DB block diagonal form.

It should be noted that decoding M_{DB} to A_{SB} is quite straightforward compared to decoding Π_{VS} of $\mathcal{G}(M)$ to a Π_{HP} of $\mathcal{H}_{RN}(A)$. This is because the net-partition depends on the node-partition of the hypergraph, a net cannot be enforced to be in cut explicitly, whereas a non-coupling row can be enforced to be in border. Actually, by the definition, the row and column disjointness of blocks is necessary and sufficient for a matrix A to be said in singly-border block diagonal form. This property has one-to-one correspondence with the definition of vertex separator of graph representation of the matrix $M = AA^T$. Therefore, GPVS formulation models the problem of transforming a matrix A into SB form, more powerfully (Theorem 4).

Chapter 6

Experimental Results with Applications

6.1 Hypergraph Partitioning-based GPVS Formulation

The experimental evaluation is performed using 50 matrices obtained from the University of Florida sparse matrix collection [17]. The first 25 matrices are general symmetric and square M matrices arising in different application domains, whereas the remaining 25 M matrices are derived from Linear Programming (LP) constraint matrices using $M = AA^T$. Table 6.1 illustrates the properties of these matrices. In this table, p and $nnz(M)$ denote, respectively, the number of rows/columns and nonzeros of matrix M. For an M-matrix derived from an LP problems, the number of columns q and nonzeros $nnz(A)$ are also listed for the respective A-matrix. Note that the number of rows of A is equal to the number of rows/columns of M. The general matrices are further divided into three groups (first 5, second 5 and remaining 15) according to the size of the maximum cliques that can be obtained from their graph representations. The reason for this division will become clear in Section 6.1.1. The matrices in each category/group are listed in increasing order of number of nonzeros.

Table 6.1: Properties of test matrices for HP-based GPVS formulation

General Matrices			LP Problems				
	$p \times p$ M matrix			$p \times p$ M matrix		$p \times q$ A matrix	
name	p	$nnz(M)$	name	p	$nnz(M)$	q	$nnz(A)$
ncvxqp9	16554	54040	ls_pdf_02	2953	23281	7716	16571
aug3dcqp	35543	128115	delf_A_36	3170	33508	6654	15397
c-53	30235	355139	lp_dfl001	6071	82267	12230	35632
c-59	41282	480536	model9	2879	103961	10939	55956
c-67	57975	530229	nl	7039	105089	15325	47035
lshp3025	3025	20833	ge	10099	112129	16369	44825
lshp3466	3466	23896	lp_ken_13	28632	161804	42659	97246
bodyy4	17546	121550	lpi_gosh	3792	206010	13455	99953
rail_20209	20209	139233	cq9	9278	221590	21534	96653
cvxbqp1	50000	349968	lp_osa_14	2337	230023	54797	317097
shuttle_eddy	10429	103599	co9	10789	249205	22924	109651
nasa4704	4704	104756	pltexpa	26894	269736	70364	143059
bcsstk24	3562	159910	model10	4400	293260	16819	150372
skirt	12598	196520	fome12	24284	329068	48920	142528
bcsstk28	4410	219024	lp_cre_d	8926	372266	73948	246614
s1rmq4m1	5489	262411	r05	5190	406158	9690	104145
vibrobox	12328	301700	world	34506	582064	67147	198883
crystk01	4875	315891	mod2	34774	604910	66409	199810
bcsstm36	23052	320606	lp_maros_r7	3136	664080	9408	144848
gridgena	48962	512084	ex3sta1	17443	679857	17516	68779
k1_san	67759	559774	psse2	28634	736338	11028	115262
finan512	74752	596992	fxm3_16	41340	765526	85575	392252
msc23052	23052	1142686	Kemelmacher	28452	781148	9693	100875
bcsstk35	30237	1450163	graphics	29493	1577187	11822	117954
oilpan	73752	2148558	stat96v5	2307	1790467	75779	233921

In Section 6.1.1, hypergraph construction performances of graphs both induced by general matrices and LP constraint matrices are discussed. Moreover, hypergraph sparsening performance of graphs induced by LP constraint matrices are presented. In Section 6.1.2, HP-based fill-reducing ordering performances of matrices are investigated.

6.1.1 Hypergraph Construction

Table 6.2 displays the hypergraph construction performance in $\mathcal{G}(M)$ of general matrices for HP-based GPVS formulation, whereas Table 6.3 displays the hypergraph construction performance in $\mathcal{G}(AA^T)$ of LP problems and hypergraph sparsening performance in $\mathcal{G}(AA^T)$ of LP problems using row-net hypergraph of matrix A. The performance results are given in terms of number of nodes and pins. In the tables, $\mathcal{H}^2$, $\mathcal{H}^3$ and $\mathcal{H}^4$ denote the clique-node hypergraphs induced by ECCs $\mathcal{C}^2$, $\mathcal{C}^3$ and $\mathcal{C}^4$, respectively. Recall that ECCs $\mathcal{C}^3$ and $\mathcal{C}^4$ are constructed from $\mathrm{G}(M)$ using Algorithm 3 and Algorithm 4, respectively. In Table 6.3, $\mathcal{H}^A$ denotes the row-net hypergraph of A matrix, whereas $\tilde{\mathcal{H}}^A$ denotes the hypergraph obtained by sparsening the hypergraph $\mathcal{H}^A$. In the tables, $|\mathcal{U}|$ and p, respectively, denote the number of nodes and the number of pins in the corresponding constructed hypergraph. Note that the number of nodes in $\mathcal{H}^2$ is equal to the number of edges in the graph $\mathcal{G}(M)$. Also note that the number of nets for all constructed hypergraphs is equal and it is equal to the number of vertices in the graph $\mathcal{G}(M)$. In the tables, the $\mathcal{H}^2$ model is considered as the base model, so the number of nodes and pins of $\mathcal{H}^3$, $\mathcal{H}^4$, $\mathcal{H}^A$ and $\tilde{\mathcal{H}}^A$ are displayed as normalized with respect to those of $\mathcal{H}^2$.

As seen in Table 6.2, the size of clique-node hypergraph for a given M matrix decreases in terms of both number of nodes and pins when larger cliques of $\mathrm{G}(M)$ are considered while constructing the hypergraph. That is, $\mathcal{H}^4$ has smaller size than $\mathcal{H}^3$, which in turn has smaller size than $\mathcal{H}^2$. However, the first five and the first ten out of 25 general matrices do not lead to 3-cliques and 4-cliques, respectively. So, the $\mathcal{H}^2$, $\mathcal{H}^3$ and $\mathcal{H}^4$ hypergraphs are the same for the first five general M matrices, whereas the $\mathcal{H}^3$ and $\mathcal{H}^4$ hypergraphs are the same for the first ten general M matrices. The second `geomean` row shows the values when those five and ten matrices are excluded

Table 6.2: Hypergraph construction performance in $\mathcal{G}(M)$ of general matrices for HP-based GPVS formulation

General Matrices							
		$\mathcal{H}^2$		$\mathcal{H}^3$		$\mathcal{H}^4$	
name	$\lvert\mathcal{V}\rvert = \lvert\mathcal{N}\rvert$	$\lvert\mathcal{U}\rvert = \lvert\mathcal{E}\rvert$	p	$\lvert\mathcal{U}\rvert$	p	$\lvert\mathcal{U}\rvert$	p
ncvxqp9	16554	23047	45540	1.00	1.00	1.00	1.00
aug3dcqp	35543	50286	100572	1.00	1.00	1.00	1.00
c-53	30235	170989	341978	1.00	1.00	1.00	1.00
c-59	41282	219627	439254	1.00	1.00	1.00	1.00
c-67	57975	236980	473960	1.00	1.00	1.00	1.00
lshp3025	3025	8904	17808	0.35	0.53	0.35	0.53
lshp3466	3466	10215	20430	0.35	0.53	0.35	0.53
bodyy4	17546	52196	104392	0.36	0.53	0.36	0.53
rail_20209	20209	59512	119024	0.48	0.71	0.48	0.71
cvxbqp1	50000	149984	299968	0.45	0.67	0.45	0.67
shuttle_eddy	10429	46585	93170	0.51	0.75	0.43	0.68
nasa4704	4704	50026	100052	0.48	0.72	0.29	0.57
bcsstk24	3562	78174	156348	0.49	0.73	0.30	0.60
skirt	12598	91964	183925	0.48	0.72	0.30	0.57
bcsstk28	4410	107307	214614	0.49	0.73	0.31	0.62
s1rmq4m1	5489	137811	275622	0.49	0.74	0.32	0.64
vibrobox	12328	165250	330500	0.50	0.74	0.33	0.61
crystk01	4875	155508	311016	0.50	0.74	0.31	0.62
bcsstm36	23052	165097	319314	0.52	0.74	0.34	0.59
gridgena	48962	231561	463122	0.54	0.74	0.39	0.60
k1_san	67759	256411	512821	0.45	0.65	0.27	0.49
finan512	74752	261120	522240	0.49	0.68	0.27	0.47
msc23052	23052	565881	1131762	0.49	0.74	0.32	0.63
bcsstk35	30237	709963	1419926	0.49	0.73	0.30	0.60
oilpan	73752	1761718	3523436	0.49	0.74	0.30	0.59
geomean				**0.54**	**0.74**	**0.42**	**0.65**
6–,11–				**0.47**	**0.69**	**0.32**	**0.59**

from the geometric averaging. That is, 6– and 11– refer to the geometric averages when the first five and the first ten matrices are excluded, respectively.

As seen in Table 6.3, the size of clique-node hypergraph of graph $\mathcal{G}(AA^T)$ for a given A matrix decreases in terms of both number of nodes and pins when larger cliques of $\mathrm{G}(AA^T)$ are considered while constructing the hypergraph. However, using row-net hypergraph $\mathcal{H}^A$ of A matrix drastically decreases size as to be smaller than even $\mathcal{H}^4$ hypergraphs in general. Furthermore, As also seen in the table, sparsening the hypergraph $\mathcal{H}^A$ also yields considerably smaller size hypergraph $\tilde{\mathcal{H}}^A$.

Table 6.3: Hypergraph construction and sparsening performances in $\mathcal{G}(AA^T)$ of LP problems for HP-based GPVS formulation

LP Problems											
		$\mathcal{H}^2$		$\mathcal{H}^3$		$\mathcal{H}^4$		$\mathcal{H}^A$		$\tilde{\mathcal{H}}^A$	
name	$\lvert\mathcal{V}\rvert = \lvert\mathcal{N}\rvert$	$\lvert\mathcal{U}\rvert = \lvert\mathcal{E}\rvert$	p	$\lvert\mathcal{U}\rvert$	p	$\lvert\mathcal{U}\rvert$	p	$\lvert\mathcal{U}\rvert$	p	$\lvert\mathcal{U}\rvert$	p
lp_pds_02	2953	10164	20328	0.75	0.83	0.74	0.81	0.76	0.82	0.74	0.81
delf_A_36	3170	15169	30338	0.48	0.70	0.34	0.60	0.44	0.51	0.18	0.31
lp_dfl001	6071	38098	76196	0.51	0.70	0.37	0.56	0.32	0.47	0.28	0.44
model9	2879	50730	101271	0.51	0.75	0.33	0.62	0.22	0.56	0.13	0.48
nl	7039	49034	98059	0.52	0.74	0.36	0.61	0.30	0.48	0.16	0.37
ge	10099	51015	102030	0.50	0.72	0.35	0.60	0.32	0.44	0.18	0.31
lp_ken_13	28632	66586	133172	0.62	0.72	0.62	0.72	0.64	0.73	0.64	0.73
lpi_gosh	3792	101213	202322	0.52	0.75	0.35	0.66	0.14	0.49	0.10	0.47
cq9	9278	106187	212343	0.50	0.74	0.34	0.60	0.20	0.45	0.11	0.33
lp_osa_14	2337	113843	227686	0.51	0.76	0.48	0.74	0.48	1.40	0.46	0.73
co9	10789	119330	238538	0.50	0.74	0.35	0.63	0.19	0.46	0.10	0.33
pltexpa	26894	121421	242842	0.51	0.69	0.43	0.63	0.58	0.58	0.33	0.46
model10	4400	144431	288861	0.51	0.75	0.33	0.62	0.12	0.52	0.10	0.49
fome12	24284	152392	304784	0.51	0.70	0.37	0.56	0.32	0.47	0.28	0.44
lp_cre_d	8926	184120	365790	0.57	0.78	0.47	0.68	0.40	0.67	0.38	0.58
r05	5190	200503	400987	0.50	0.74	0.34	0.66	0.04	0.26	0.04	0.26
world	34506	274179	547958	0.49	0.72	0.34	0.60	0.24	0.37	0.11	0.29
mod2	34774	285487	570555	0.49	0.72	0.34	0.60	0.22	0.35	0.10	0.28
lp_maros_r7	3136	330472	660944	0.50	0.75	0.35	0.70	0.03	0.22	0.01	0.11
ex3sta1	17443	331207	662414	0.49	0.73	0.31	0.61	0.05	0.10	0.02	0.08
psse2	28634	353852	707704	0.48	0.72	0.30	0.60	0.02	0.16	0.01	0.08
fxm3_16	41340	362093	724186	0.51	0.74	0.37	0.65	0.23	0.55	0.13	0.29
Kemelmacher	28452	376348	752696	0.48	0.72	0.31	0.62	0.03	0.13	0.03	0.13
graphics	29493	773847	1547694	0.49	0.74	0.32	0.64	0.03	0.08	0.01	0.06
stat96v5	2307	894082	1788162	0.50	0.75	0.34	0.68	0.00	0.25	0.00	0.01
geomean				**0.52**	**0.74**	**0.37**	**0.64**	**0.14**	**0.38**	**0.09**	**0.26**

6.1.2 Hypergraph Partitioning-based Fill-Reducing Ordering

This idea backs to PhD work of Catalyurek [10] and also there is a recent work by Catalyurek, Aykanat and Kayaaslan [8].

6.1.2.1 Fill-Reducing Ordering

In most scientific computing applications, the core of the computation is solving a symmetric system of linear equations in the form $Mx = b$. Direct methods, such as LU and Cholesky factorizations, are commonly preferred for solving such systems for their numerical robustness. A typical first step of a direct method is a heuristic reordering of the rows and columns of M to reduce *fill* in the triangular factor matrices. The fill is the set of zero entries in M that become nonzero in the triangular factor matrices. Another goal in reordering is to reduce the number of floating-point operations required to perform the triangular factorization, also known as *operation count*. It is equal to the sum of the squares of the nonzeros of each eliminated row/column, hence it is directly related with the number of fills.

For a symmetric matrix, the evolution of the nonzero structure during the factorization can easily be described in terms of its graph representation [43]. In graph terms, the elimination of a vertex (which corresponds to a row/column of the matrix) creates edges for every pair of its adjacent vertices. In other words, elimination of a vertex makes its adjacent vertices clique of size its degree minus one. In this process, the added edges directly correspond to the *fill* in the matrix. Obviously, the amount of fill and operation count depends on the row/column elimination order. The aim of ordering is to reduce these quantities which leads to both faster and less memory intensive solution of the linear system. Unfortunately this problem is known to be NP-hard [46]

Heuristic methods for fill reducing ordering can be divided into mainly two categories: *bottom-up* (also called *local*) and *top-down* (also called *global*) approaches [38]. In the bottom-up category, one of the most popular ordering methods is *Minimum Degree* (MD) heuristic [45] in which at every elimination step vertex with

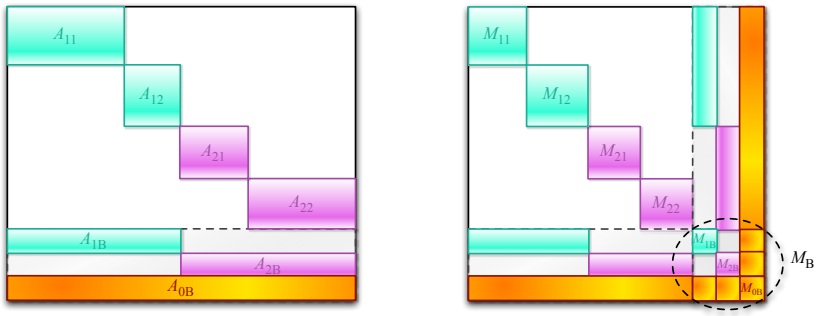
minimum degree, hence the name, is chosen for elimination. Success of the MD heuristic is followed by many variants of it, such as Quotient Minimum Degree (QMD) [21], Multiple Minimum Degree (MMD) [37], Approximate Minimum Degree (AMD) [2], and Approximate Minimum Fill (AMF) [44]. Among the top-down approaches, one of the most famous and influential one is surely *nested dissection* (ND) [20]. The main idea of ND is as follows: Consider a partitioning of vertices into three sets: $\mathcal{V}_1$, $\mathcal{V}_2$ and $\mathcal{V}_S$, such that the removal of $\mathcal{V}_S$, called *separator*, decouples $\mathcal{V}_1$ and $\mathcal{V}_2$. If we order the vertices of $\mathcal{V}_S$ after the vertices of $\mathcal{V}_1$ and $\mathcal{V}_2$, certainly no fill can occur between the vertices of $\mathcal{V}_1$ and $\mathcal{V}_2$. Furthermore, the elimination process in $\mathcal{V}_1$ and $\mathcal{V}_2$ are independent tasks and their elimination only incurs fill to themselves and $\mathcal{V}_S$. Hence, the ordering of the vertices of $\mathcal{V}_1$ and $\mathcal{V}_2$ can be computed by applying the algorithm recursively. In ND, since the quality of the ordering depends on the size of $\mathcal{V}_S$, finding a small separator is desirable.

Although the ND scheme has some nice theoretical results [20], it has not been widely used until the development of recent multilevel graph partitioning tools. State-of-the-art ordering tools [12, 25, 27, 32] are mostly a hybrid of top-down and bottom-up approaches and built using incomplete ND approach that utilizes a multilevel graph partitioning framework [7, 24, 26, 31] for recursively identifying separators until a part becomes sufficiently small. After this point, a variant of MD, like *Constraint Minimum Degree* (CMD) [38] is used for the ordering of parts.

6.1.2.2 Hypergraph Partitioning-based Solution

Given a $p \times p$ symmetric and square matrix $M = \{m_{ij}\}$ for fill reducing ordering, let $\mathcal{G}(M) = (\mathcal{V}, \mathcal{E})$ denote the standard graph representation of matrix M.

As described in [3], the fill-reducing matrix reordering schemes based on incomplete nested dissection can be classified as: *nested dissection* (ND) and *multisection* (MS). Both schemes apply 2-way GPVS (bisection) recursively on $\mathrm{G}(M)$ until the parts (domains) become fairly small. After each bisection step, the vertices in the



(a) (b)

Figure 6.1: (a) A sample 4-way SB form of a matrix A obtained through a 2-level recursive hypergraph bipartitioning process; (b) the corresponding 4-way DB form of matrix $M = AA^T$

2-way separator (bisector) are removed and the further bisection operations are recursively performed on the subgraphs induced by the parts of bisection. In the proposed recursive-HP-based ordering approach, the constructed hypergraph $\mathcal{H}$ (where $\mathrm{NIG}(\mathcal{H}) \equiv \mathrm{G}(M)$) is bipartitioned recursively until the number of internal nets of the parts become fairly small. After each bipartitioning step, the cut nets are removed and the further bipartitioning operations are recursively performed on the sub-hypergraphs induced by the node parts of the bipartition. Note that this cut-net removal scheme in recursive 2-way HP corresponds to the above-mentioned separator-vertex removal scheme in recursive 2-way GPVS. In this work, we only consider MS scheme. The parts of the multiway separator are ordered using an MD-based algorithm before the separator. It is clear that the parts can be ordered independently. These two schemes differ in the order that they number the vertices of the multiway separator. In the MS scheme, the multiway separator is ordered using an MD-based algorithm as a whole in a single step.

Figure 6.1 displays a sample 4-way SB form of matrix A and the corresponding 4-way DB form of matrix M induced by a 2-level recursive bipartitioning/bisection process. Here, the bipartitioning/bisection operation at the root level is numbered as 0, whereas the bipartitioning/bisection operations at the second level are numbered as 1 and 2. The parts of a bipartition/bisection are always numbered as 1 and 2, whereas the border is numbered as B. For example, A_{11}/M_{11} and A_{12}/M_{12} denote the diagonal domain submatrices corresponding to the two parts of the bipartitioning/bisection

operation 1, whereas A_{21}/M_{21} and A_{22}/M_{22} denote the diagonal domain submatrices corresponding to the two parts of the bipartitioning/bisection operation 2. As seen in the figure, $M_{0B} = A_{0B}A_{0B}^T$ denotes the diagonal border submatrix corresponding to the 2-way separator obtained at the root level, whereas $M_{1B} = A_{1B}A_{1B}^T$ and $M_{2B} = A_{2B}A_{2B}^T$ denote the diagonal border submatrices corresponding to the 2-way separators obtained at the second level. Note that M_B denotes the diagonal border submatrix corresponding to the overall 4-way separator. Diagonal domain submatrices are ordered before the diagonal border submatrix M_B. The Schur complement of the overall diagonal border submatrix M_B is ordered as a whole.

6.1.2.3 Experimental Results

Catalyurek embedded the HP-based GPVS formulation into the state-of-the-art HP tool PaToH [11] in [10] and throughout this work the mentioned tool is enhanced as to enable use of 3-clique- and 4-clique-node hypergraphs as described in Algorithms 3 and 4 along with the new sparsening method described in Section 4.2 and Algorithm 6. This tool along with the enhancements is named as oPaToH. In oPaToH, the recursive hypergraph bipartitioning process is continued until the number of internal nets of a part of a bipartition drops below 200 or the number of nodes of a part of a bipartition drops below 100. All experiments are performed on a PC equipped with a 1.6 Ghz Pentium 4, and 1 GB memory.

Table 6.4: The geometric mean results of HP-based fill-reducing ordering

	General Matrices						LP Problems					
				oPaToH-MS						oPaToH-MS		
criteria	MMD	oe	SM	$\mathcal{H}^2$	$\mathcal{H}^3$	$\mathcal{H}^4$	MMD	oe	SM	$\mathcal{H}^4$	$\mathcal{H}^A$	$\tilde{\mathcal{H}}^A$
opc	1.40	1.08	1.01	0.92	**0.88**	**0.84**	1.27	2.22	1.42	0.84	0.79	**0.79**
$nnz(L)$	1.13	1.02	1.00	0.97	**0.94**	**0.92**	1.10	1.40	1.19	0.94	0.91	**0.91**
time (s)	0.81	1.34	3.86	3.75	**3.63**	**4.08**	1.46	0.98	5.19	7.62	1.95	**1.79**

The ordering performances of oPaToH is summarized in Table 6.4. The metrics used to evaluate the ordering are operation count (opc), nonzero fill ($nnz(L)$) and preprocessing time. The geometric mean of the normalized results according to $onmetis$

are showed in the table. The table compares the ordering results also with other state-of-the-art ordering tools MMD, $oemetis$ and $SMOOTH$. As seen in the table, oPaToH produces more qualified ordering results in terms of opc and $nnz(L)$. For general matrices, in the second `geomean` row, 1–, 6– and 11– refer to the geometric averages when none, the first five and the first ten of the matrices are excluded, respectively. Generally, oPaToH even using 2-clique-node hypergraphs $\mathcal{H}^2$ improves the ordering quality at a cost of spending much more time to find ordering. Moreover, oPaToH produces better orderings than all presented ordering tools, on the average. In terms of preprocessing time, oPaToH performs faster only than SMOOTH when $\mathcal{H}^2$ or $\mathcal{H}^3$ is used for general matrices and $\mathcal{H}^A$ or $\tilde{\mathcal{H}}^A$ is used for LP problems. In the ordering of general matrices, if we concentrate on using 3-clique- and 4-clique-node hypergraphs $\mathcal{H}^3$ and $\mathcal{H}^4$, we see that the ordering quality increases considerably with increasing size of cliques. On average, using $\mathcal{H}^3$ amortizes its time cost of constructing the 3-clique-node hypergraph, whereas using $\mathcal{H}^4$ doesn't amortize. But, using $\mathcal{H}^4$ leads to better orderings than using $\mathcal{H}^3$. In the ordering of matrices induced by LP problems, we see that using $\mathcal{H}^A$ yields considerably better orderings than using even $\mathcal{H}^4$. Although, sparsening $\mathcal{H}^A$ doesn't improve the ordering quality, it improves the preprocessing time of finding the ordering.

Tables 6.5, 6.6 and 6.7 gives detailed results of ordering performances of oPaToH. Table 6.5 compares the ordering quality performance in terms of operation count opc in linear solution of ordered matrix. Table 6.6 compares the ordering quality performance in terms of nonzero fill count in linear solution of ordered matrix which is equal to $nnz(L)$. Finally, Table 6.7 investigates the time performances of the ordering tools. In the tables, the results of ordering tools (except $onmetis$) are given normalized to results of $onmetis$.

Table 6.5: Operation counts of HP-based fill-reducing ordering

General Matrices					LP Problems						
		oPaToH-MS					oPaToH-MS				
name	onmetis	$\mathcal{H}^2$	$\mathcal{H}^3$	$\mathcal{H}^4$	name	onmetis	$\mathcal{H}^2$	$\mathcal{H}^3$	$\mathcal{H}^4$	$\mathcal{H}^A$	$\tilde{\mathcal{H}}^A$
ncvxqp9	5.94E+06	0.74	0.74	0.74	lp_pds_02	7.34E+08	0.85	0.84	0.82	0.78	0.77
aug3dcqp	2.78E+08	0.92	0.92	0.92	delf_A_36	1.76E+06	0.98	0.86	0.87	0.73	0.80
c-53	3.03E+07	0.85	0.85	0.85	lp_dfl001	7.34E+08	0.75	0.72	0.70	0.73	0.72
c-59	2.48E+09	1.17	1.17	1.17	model9	4.86E+06	0.68	0.67	0.67	0.70	0.70
c-67	1.32E+07	0.88	0.88	0.88	nl	4.20E+07	0.87	0.88	0.90	0.90	0.87
lshp3025	3.01E+06	0.97	1.02	1.02	ge	2.28E+07	1.15	1.19	0.99	1.07	0.93
lshp3466	3.65E+06	0.99	0.91	0.91	lp_ken_13	1.72E+07	0.98	0.97	0.97	0.97	0.97
bodyy4	3.29E+07	1.04	1.03	1.03	lpi_gosh	3.66E+07	0.96	0.88	0.84	0.82	0.81
rail_20209	1.31E+07	1.01	0.98	0.98	cq9	4.10E+07	1.00	0.99	1.00	0.89	0.91
cvxbqp1	4.56E+08	0.99	0.98	0.98	lp_osa_14	6.21E+06	1.06	1.06	1.06	1.06	1.06
shuttle_ed.	2.07E+07	1.09	1.06	1.04	co9	5.03E+07	0.96	0.97	0.97	0.96	0.96
nasa4704	3.88E+07	0.63	0.70	0.67	pltexpa	1.57E+08	0.78	0.68	0.67	0.61	0.68
bcsstk24	4.14E+07	0.91	0.83	0.82	model10	5.69E+07	1.06	0.99	0.97	1.02	0.99
skirt	2.92E+07	1.09	1.00	0.97	fome12	2.58E+09	0.93	0.93	0.91	0.89	0.91
bcsstk28	5.40E+07	0.76	0.78	0.76	lp_cre_d	2.05E+08	0.87	0.90	0.89	0.86	0.84
s1rmq4m1	1.07E+08	0.93	0.92	0.91	r05	1.21E+08	0.44	0.44	0.44	0.44	0.44
vibrobox	1.31E+09	0.79	0.56	0.53	world	3.52E+08	0.71	0.71	0.71	0.64	0.64
crystk01	2.73E+08	1.20	1.12	1.17	mod2	4.24E+08	0.57	0.55	0.55	0.51	0.52
bcsstm36	1.14E+08	0.76	0.75	0.77	lp_maros_r7	7.19E+08	1.10	1.09	1.02	0.92	0.92
gridgena	3.61E+08	0.88	0.84	0.84	ex3sta1	7.89E+09	1.35	1.22	1.59	1.02	1.03
k1_san	4.00E+08	1.30	0.86	0.93	psse2	7.67E+07	0.68	0.70	0.65	0.59	0.62
finan512	1.56E+08	0.95	0.91	0.80	fxm3_16	2.76E+07	0.72	0.74	0.72	0.74	0.73
msc23052	6.40E+08	0.84	0.85	0.85	Kemelm.	1.11E+09	1.65	1.28	1.14	0.89	0.89
bcsstk35	4.81E+08	0.82	0.80	0.81	graphics	1.33E+09	0.83	0.82	0.84	0.73	0.73
oilpan	2.78E+09	0.95	0.97	1.00	stat96v5	2.55E+09	0.94	0.98	0.88	0.85	0.79
geomean		**0.92**	**0.89**	**0.88**			**0.88**	**0.86**	**0.84**	**0.79**	**0.79**
1-,6-,11-		**0.92**	**0.88**	**0.84**							

Table 6.6: Factor nonzero counts of HP-based fill-reducing ordering

General Matrices					LP Problems						
		oPaToH-MS					oPaToH-MS				
name	onmetis	$\mathcal{H}^2$	$\mathcal{H}^3$	$\mathcal{H}^4$	name	onmetis	$\mathcal{H}^2$	$\mathcal{H}^3$	$\mathcal{H}^4$	$\mathcal{H}^A$	$\tilde{\mathcal{H}}^A$
ncvxqp9	123,462	0.95	0.95	0.95	lp_pds_02	1,277,948	0.97	0.96	0.95	0.93	0.93
aug3dcqp	1,011,206	0.90	0.90	0.90	delf_A_36	50,185	0.98	0.95	0.95	0.91	0.93
c-53	391,804	1.07	1.07	1.07	lp_dfl001	1,277,948	0.85	0.83	0.82	0.84	0.83
c-59	3,221,503	0.97	0.97	0.97	model9	97,041	0.86	0.86	0.86	0.87	0.87
c-67	439,898	1.02	1.02	1.02	nl	298,181	0.94	0.94	0.95	0.95	0.94
lshp3025	72,058	0.99	1.00	1.00	ge	267,891	1.01	1.01	0.97	0.99	0.96
lshp3466	84,338	0.99	0.97	0.97	lp_ken_13	349,677	1.02	1.02	1.02	1.02	1.02
bodyy4	501,494	1.01	1.00	1.00	lpi_gosh	249,770	0.97	0.94	0.94	0.93	0.92
rail_20209	319,401	1.02	1.01	1.01	cq9	406,070	0.99	0.99	0.99	0.95	0.96
cvxbqp1	1,978,965	0.97	0.97	0.97	lp_osa_14	116,160	1.02	1.02	1.02	1.02	1.02
shuttle_ed.	352,363	1.03	1.01	1.01	co9	471,887	0.97	0.97	0.97	0.96	0.96
nasa4704	303,720	0.81	0.83	0.82	pltexpa	1,229,360	0.85	0.81	0.81	0.79	0.81
bcsstk24	314,104	0.94	0.91	0.91	model10	392,002	1.03	1.00	0.99	1.01	0.99
skirt	465,553	1.02	0.99	0.98	fome12	4,732,776	0.95	0.95	0.94	0.92	0.93
bcsstk28	403,052	0.89	0.90	0.90	lp_cre_d	752,413	0.93	0.95	0.94	0.93	0.92
s1rmq4m1	646,878	0.96	0.96	0.95	r05	528,707	0.82	0.82	0.82	0.82	0.82
vibrobox	2,482,629	0.85	0.74	0.72	world	2,001,206	0.86	0.86	0.86	0.83	0.83
crystk01	1,003,272	1.06	1.03	1.05	mod2	2,214,268	0.79	0.78	0.78	0.76	0.76
bcsstm36	879,713	0.90	0.90	0.90	lp_maros_r7	1,401,207	1.03	1.03	1.00	0.95	0.95
gridgena	2,671,255	0.96	0.95	0.94	ex3sta1	8,052,424	1.12	1.09	1.26	0.99	0.99
k1_san	2,605,291	1.12	0.94	0.97	psse2	963,141	0.87	0.87	0.86	0.83	0.84
finan512	1,747,840	1.02	0.99	0.96	fxm3_16	686,497	0.93	0.94	0.93	0.95	0.94
msc23052	2,934,092	0.90	0.90	0.91	Kemelm.	4,152,721	1.23	1.10	1.04	0.94	0.94
bcsstk35	3,049,203	0.90	0.90	0.90	graphics	4,966,956	0.89	0.89	0.89	0.85	0.85
oilpan	9,137,443	0.96	0.97	0.98	stat96v5	2,169,949	0.97	0.99	0.94	0.93	0.90
geomean		**0.97**	**0.95**	**0.95**			**0.95**	**0.94**	**0.94**	**0.91**	**0.91**
1-,6-,11-		**0.97**	**0.94**	**0.92**							

Table 6.7: Total execution times of HP-based fill-reducing ordering

General Matrices					LP Problems						
		oPaToH-MS					oPaToH-MS				
name	onmetis	$\mathcal{H}^2$	$\mathcal{H}^3$	$\mathcal{H}^4$	name	onmetis	$\mathcal{H}^2$	$\mathcal{H}^3$	$\mathcal{H}^4$	$\mathcal{H}^A$	$\tilde{\mathcal{H}}^A$
ncvxqp9	0.160 s	2.06	2.06	2.06	lp_pds_02	0.120 s	5.80	4.95	5.05	5.54	5.10
aug3dcqp	0.540 s	1.30	1.30	1.30	delf_A_36	0.030 s	5.17	3.80	3.57	2.05	1.70
c-53	0.570 s	3.41	3.41	3.41	lp_dfl001	0.120 s	5.28	3.97	3.76	2.22	2.44
c-59	0.960 s	3.77	3.77	3.77	model9	0.060 s	9.35	8.05	7.22	2.40	2.62
c-67	1.060 s	9.40	9.40	9.40	nl	0.090 s	11.19	7.06	6.50	2.05	2.30
lshp3025	0.010 s	9.30	6.00	6.00	ge	0.140 s	4.78	3.41	2.82	1.61	1.48
lshp3466	0.020 s	5.70	3.50	3.50	lp_ken_13	0.300 s	6.10	3.49	4.06	2.97	3.42
bodyy4	0.180 s	4.45	2.62	2.62	lpi_gosh	0.110 s	17.22	13.65	11.83	3.08	3.33
rail_20209	0.240 s	4.18	2.44	2.44	cq9	0.170 s	20.19	13.43	12.53	3.13	2.75
cvxbqp1	0.650 s	5.52	3.47	3.43	lp_osa_14	0.140 s	0.55	11.86	34.02	5.41	0.92
shuttle_ed.	0.110 s	6.43	4.05	3.53	co9	0.200 s	19.63	13.75	11.43	3.14	2.51
nasa4704	0.030 s	4.10	4.57	4.70	pltexpa	0.500 s	4.26	2.80	2.53	1.51	1.80
bcsstk24	0.020 s	3.85	5.05	8.30	model10	0.150 s	15.15	13.97	12.62	3.63	4.03
skirt	0.170 s	8.40	5.26	4.23	fome12	0.670 s	7.98	4.89	4.07	2.57	2.78
bcsstk28	0.030 s	1.93	3.23	5.03	lp_cre_d	0.350 s	47.90	26.63	19.69	15.71	13.35
s1rmq4m1	0.030 s	2.30	5.03	7.43	r05	0.170 s	6.64	4.59	5.24	0.94	1.53
vibrobox	0.340 s	10.14	6.61	5.45	world	0.830 s	11.95	7.00	5.54	1.82	1.77
crystk01	0.600 s	4.03	5.97	8.05	mod2	0.870 s	11.18	7.04	5.34	1.63	1.60
bcsstm36	0.380 s	0.85	0.98	1.03	lp_maros_r7	0.320 s	51.00	31.61	30.35	1.02	1.23
gridgena	0.720 s	6.63	4.46	3.73	ex3sta1	0.450 s	8.18	7.81	6.34	0.68	0.82
k1_san	1.200 s	4.61	2.47	1.95	psse2	0.220 s	2.50	3.34	4.38	0.73	0.85
finan512	1.290 s	5.08	2.98	1.72	fxm3_16	0.900 s	7.72	5.78	5.42	2.15	1.42
msc23052	0.220 s	1.82	3.52	4.52	Kemelm.	0.940 s	11.74	6.36	5.46	0.81	0.91
bcsstk35	0.300 s	2.43	3.56	5.57	graphics	0.390 s	3.04	4.86	6.59	0.42	0.71
oilpan	0.640 s	1.45	2.60	4.69	stat96v5	0.670 s	175.4	104.9	82.9	0.70	0.16
geomean		**3.75**	**3.52**	**3.74**			9.38	7.89	7.62	**1.94**	**1.79**
1-,6-,11-		**3.75**	**3.63**	**4.08**							

6.2 GPVS-based Hypergraph Partitioning Formulation

The experimental evaluation is performed using 36 $p \times q$ sparse rectangular matrices obtained from the University of Florida sparse matrix collection [17]. Table 6.8 illustrates the properties of these matrices. In this table, p, q and $nnz(A)$ denote, respectively, the number of rows, columns and nonzeros of matrix A. In the second part, p and $nnz(M)$ denote, respectively, the number of rows/columns and nonzeros of matrix $M = AA^T$. Note that the number of rows of a matrix A is equal to the number of rows/columns of corresponding matrix $M = AA^T$.

In Section 6.2.1, hypergraph partition construction performances of vertex separators of net intersection graphs of hypergraphs induced by the matrices are discussed. These sparse rectangular matrices are obtained from LP constraint matrices and in Section 6.2.2, GPVS-based Dantzig-Wolfe decomposition performances of these matrices are investigated.

6.2.1 Hypergraph Partition Construction

Table 6.9 displays the hypergraph partition construction performance of partitions with different number of part counts: for 2,4,8,16,32,64,128-way partitions, whereas Table 6.10 shows the hypergraph partition construction performance of 128-way partitions Π_{VS} of $NIG(\mathcal{H}^A)$, where $\mathcal{H}^A$ represents the row-net hypergraph of A matrix. In the tables ε_0 represents the basic imbalance of corresponding partition. The tables are composed of three parts. In the first part, vertex separator results on graph $NIG(\mathcal{H}^A)$ is displayed in terms of basic imbalance ε_0 and percentage of separator size $(\mathcal{V}_S)$ to all vertices of the resultant vertex separator Π_{VS}. In the second part, $(\mathcal{U}_F)$ denotes the percentage of free nodes to all nodes and $(\mathcal{N}_{F \cup P})$ denotes the percentage of preservable nets to all external nets of the net-partition Π_{HP_N} induced by the resultant vertex separator Π_{VS}. Third part also has two subparts. At each subpart the complete node partition Π_{HP_U} results are displayed induced by the free node assignment resulted by each of the Algorithms RANDOMASSIGN and GREEDYASSIGN Here,

Table 6.8: Properties of test matrices for GPVS-based HP formulation

	A			$M = AA^T$	
name	p	q	$nnz(A)$	p	$nnz(M)$
lp_ken_07	2426	3602	8404	2426	14382
delf_A_36	3170	5598	15397	3170	33508
lp_pds_02	2953	7716	16571	2953	23281
fxm2-16	3900	7335	32972	3900	74806
lp_dfl001	6071	12230	35632	6071	82267
ge	10099	16369	44825	10099	112129
nl	7039	15325	47035	7039	105089
lp_ken_11	14694	21349	49058	14694	82454
model7	3358	9560	51027	3358	97438
cq5	5048	11748	51571	5048	117920
model9	2787	10939	55956	2787	103869
co5	5774	12325	57993	5774	131692
p05	5090	9590	59045	5090	224528
lp_pds_06	9881	29351	63220	9881	88003
ex3sta1	17443	17516	68779	17443	679857
cq9	9278	21534	96653	9278	221590
lp_ken_13	28632	42659	97246	28632	161804
lpi_gosh	3790	13455	99953	3790	206008
r05	5190	9690	104145	5190	406158
lp_pds_10	16558	49932	107605	16558	149658
co9	10789	22924	109651	10789	249205
scfxm1-2b	19036	33047	111052	19036	538278
p010	10090	19090	118000	10090	448318
fome12	24284	48920	142528	24284	329068
pltexpa	26894	70364	143059	26894	269736
model10	4400	16819	150372	4400	293260
nemswrld	6647	28550	192283	6647	361421
world	34506	67147	198883	34506	582064
mod2	34774	66409	199810	34774	604910
route	20894	43019	206782	20894	1294804
lp_pds_20	33798	108175	232647	33798	320120
lp_cre_d	6476	73948	246614	6476	369816
lp_cre_b	7240	77137	260785	7240	396398
lpl1	32460	32460	328036	32460	6882066
fxm3_16	41340	85575	392252	41340	765526
stormg2-125	65935	172431	433256	65935	1953519

Table 6.9: Hypergraph partition construction performance for GPVS-based HP formulation with geometric means

	$\Pi_{VS}(\text{NIG}(\mathcal{H}^A))$		$\Pi_{HP_N}(\mathcal{H}^A)$		$\Pi_{HP_U}(\mathcal{H}^A)$					
					RANDOMASSIGN			GREEDYASSIGN		
K-way	ε_0	$(\mathcal{V}_S)$	$(\mathcal{U}_F)$	$(\mathcal{N}_{F\cup P})$	ε_0	$(\mathcal{N}_S)$	fp	ε_0	$(\mathcal{N}_S)$	fp
2-way	0.01	%1.0	%0.1	%0.0	0.01	%0.9	%18.3	0.01	%0.9	%19.0
4-way	0.04	%4.8	%0.6	%0.2	0.04	%4.6	%93.2	0.05	%4.7	%70.9
8-way	0.08	%6.8	%1.0	%0.6	0.08	%6.7	%92.0	0.08	%6.6	%71.4
16-way	0.10	%8.9	%1.6	%1.1	0.10	%8.8	%97.0	0.10	%8.8	%92.2
32-way	0.15	%13.3	%2.7	%2.1	0.15	%13.3	%99.0	0.17	%13.2	%94.1
64-way	0.21	%18.7	%4.9	%3.4	0.21	%18.6	%99.5	0.27	%18.4	%92.4
128-way	0.30	%23.8	%7.2	%4.8	0.30	%23.7	%99.8	0.48	%23.3	%90.9

$(\mathcal{N}_S)$ denotes percentage of cut size to all nets of the constructed hypergraph partition Π_{HP_U}. fp shows the free node assignment performance of the algorithms in terms of percentage of preserved nets at Π_{HP_U} to all preservable nets of the net-partition Π_{HP_N} induced by Π_{VS}.

As seen in Table 6.9, when part counts of the partition increases, basic imbalance and separator size of vertex separator increases. Furthermore, percentage of free nodes and percentage of preservable nets also increase with increasing number of parts. For the free node assignment performance, it can be seen that Algorithm RANDOMASSIGN performs almost perfectly, whereas Algorithm GREEDYASSIGN shows a poorer performance. As seen in the table, Algorithm RANDOMASSIGN could preserve much more preservable nets than Algorithm GREEDYASSIGN, on the average. Also note that when part count of partitions increases, the preservability also increases and Algorithm RANDOMASSIGN could preserve almost all preservable nets. This also confirms the $\frac{(K-1)}{K}$-approximability of Algorithm RANDOMASSIGN, because larger K is expected to yield better performance. As seen in the table, when preservable nets are preserved more, the basic imbalance of hypergraph partition Π_{HP} is nearer to basic imbalance of vertex separator Π_{VS}. Algorithm RANDOMASSIGN achieved to preserve the imbalance and separator size values of the vertex separator, whereas Algorithm GREEDYASSIGN deviated from the original imbalance and separator size which yielded just worse imbalance ratio. As also seen in the table, the basic imbalance and percentage of cutsize results of complete node partition Π_{HP_U} is also

Table 6.10: Hypergraph partition construction performance of 128-way partition for GPVS-based HP formulation

	$\Pi_{VS}(\text{NIG}(\mathcal{H}^A))$		$\Pi_{HP_N}(\mathcal{H}^A)$		$\Pi_{HP_U}(\mathcal{H}^A)$					
					RANDOMASSIGN			GREEDYASSIGN		
K-way	ε_0	$(\mathcal{V}_S)$	$(\mathcal{U}_F)$	$(\mathcal{N}_{F\cup P})$	ε_0	$(\mathcal{N}_S)$	fp	ε_0	$(\mathcal{N}_S)$	fp
lp_ken_07	%13.3	0.28	%3.5	%3.0	%13.3	0.28	%100.0	%13.3	0.28	%100.0
delf_A_36	%26.2	0.22	%12.2	%2.2	%26.1	0.22	%100.0	%26.1	0.22	%100.0
lp_pds_02	%14.8	0.22	%1.9	%1.2	%14.7	0.22	%100.0	%14.7	0.22	%100.0
fxm2-16	%32.9	0.31	%7.0	%9.4	%32.5	0.37	%99.4	%32.3	0.36	%95.4
lp_dfl001	%47.8	0.28	%19.3	%6.6	%47.8	0.28	%100.0	%47.2	0.69	%87.6
ge	%21.7	0.31	%6.5	%10.6	%21.7	0.31	%99.8	%21.5	0.31	%93.4
nl	%17.6	0.24	%5.0	%2.7	%17.6	0.24	%99.3	%17.6	0.24	%98.5
lp_ken_11	%2.9	0.27	%0.5	%0.0	%2.9	0.27	%100.0	%2.9	0.27	%100.0
model7	%32.6	0.33	%12.2	%17.8	%32.6	0.33	%99.6	%32.0	0.44	%91.0
cq5	%22.6	0.32	%8.6	%7.7	%22.6	0.32	%99.9	%22.5	0.32	%95.8
model9	%46.5	0.52	%27.2	%21.2	%46.4	0.51	%100.0	%41.8	9.28	%54.3
co5	%25.3	0.22	%9.8	%12.9	%25.2	0.22	%100.0	%24.4	0.72	%86.1
p05	%29.0	0.30	%18.9	%4.1	%29.0	0.30	%100.0	%29.0	0.30	%99.1
lp_pds_06	%19.5	0.23	%5.9	%3.9	%19.5	0.23	%100.0	%19.3	0.25	%82.2
ex3sta1	%50.6	0.41	%10.1	%8.5	%50.5	0.41	%99.7	%50.5	0.41	%98.8
cq9	%20.3	0.30	%6.3	%9.3	%20.3	0.30	%99.9	%20.2	0.30	%94.2
lp_ken_13	%3.5	0.17	%0.9	%0.1	%3.5	0.17	%100.0	%3.5	0.17	%100.0
lpi_gosh	%29.3	0.37	%7.9	%11.7	%28.8	0.38	%99.5	%28.8	0.37	%98.0
r05	%34.7	0.42	%25.2	%4.2	%34.6	0.42	%99.8	%34.6	0.42	%99.0
lp_pds_10	%22.1	0.23	%6.2	%3.5	%22.1	0.23	%100.0	%22.0	0.23	%89.3
co9	%28.1	0.27	%9.3	%18.0	%27.9	0.29	%99.6	%26.2	1.43	%83.1
scfxm1-2b	%7.3	0.14	%0.6	%2.4	%7.2	0.15	%99.9	%7.2	0.15	%100.0
p010	%17.7	0.16	%10.1	%2.8	%17.7	0.16	%99.5	%17.7	0.16	%98.7
fome12	%46.1	0.28	%17.9	%11.2	%46.1	0.27	%99.9	%45.2	1.43	%85.8
pltexpa	%24.2	0.24	%7.1	%5.2	%24.2	0.24	%99.9	%24.1	0.24	%97.9
model10	%53.6	0.57	%30.6	%18.1	%53.5	0.61	%99.6	%52.1	2.67	%86.6
nemswrld	%52.6	0.45	%38.6	%29.7	%52.3	0.46	%99.8	%47.1	11.83	%67.2
world	%18.7	0.21	%5.7	%3.4	%18.7	0.21	%99.8	%18.7	0.21	%96.1
mod2	%19.5	0.23	%5.5	%2.5	%19.5	0.23	%99.8	%19.5	0.23	%97.8
route	%48.9	0.63	%19.8	%25.8	%48.8	0.63	%99.6	%48.8	0.62	%99.3
lp_pds_20	%21.5	0.21	%5.3	%2.5	%21.5	0.21	%99.9	%21.5	0.21	%85.8
lp_cre_d	%42.6	0.49	%26.0	%16.8	%42.5	0.49	%99.9	%40.4	3.44	%72.7
lp_cre_b	%38.4	0.40	%24.4	%16.7	%38.4	0.40	%100.0	%37.2	1.46	%81.9
lpl1	%74.7	1.43	%29.7	%21.6	%73.4	1.56	%99.8	%73.1	1.60	%98.0
fxm3_16	%5.5	0.10	%0.1	%0.3	%5.5	0.10	%99.7	%5.5	0.10	%100.0
stormg2-125	%17.7	0.29	%1.2	%3.7	%17.6	0.29	%99.5	%17.5	0.32	%92.3
geomean	**%23.8**	**0.30**	**%7.2**	**%4.8**	**%23.7**	**0.30**	**%99.8**	**%23.3**	**0.48**	**%90.9**

related to performance of free node assignment. The results also implicitly imply that the number of inevitably internal nets of $\mathcal{N}_S$ of Π_{HP_N} is negligibly small. The results, as a whole, conform to the theoretical results discussed in Section 5.1.1 and confirm the validity of decision of trying to preserve the preservable nets in cut.

6.2.2 GPVS-based Dantzig-Wolfe Decomposition of Linear Programming Problems

6.2.2.1 Dantzig-Wolfe Decomposition of LP Problems

Dantzig and Wolfe [16] disclosed the exploitation of the block-angular structure or equivalently singly-border block diagonal (SB) form for solving linear programs (LPs) and then the problem of transforming a matrix into the block-angular structure is named as Dantzig-Wolfe Decomposition. The motivation was solving large LPs with limited memory. After that, several studies are done to investigate the parallelization techniques [22, 28, 39]. The techniques proposed in [16, 36, 40] highlighted the iterative algorithms, where each iteration involves solving K independent LP subproblems. These subproblems correspond to the block constraints. This process follows by a coordination phase for coordinating the solutions of the subproblems due to the coupling constraints. The two nice properties are, firstly, as the solution times of most LPs increase in practice as a quadratic or cubic function with the size of the problem, it is more efficient to solve a set of small problems. Secondly, they bring forth to a natural, coarse-grain parallelism by processing the subproblems concurrently. In the implementation of coarse-grain parallelism, at each iteration, slave processors solve the LP subproblems concurrently, whereas a master processor solves the serial master problem in the coordination phase. The successful decomposition-based techniques can be used for coarse-grain parallel solution of general LP problems by transformation to block-angular forms [18]. This transformation can be presented as permuting the rectangular constraint matrix of the LP problem into a singly-bordered block diagonal (SB) form, as shown in Equation 2.7. The transformation problem refers to permuting LP constraint matrix minimizing the border size, while maintaining a given balance criterion on the diagonal blocks. Here, permuting rows refers to permuting constraints and

permuting columns refers to permuting variables. Thus, each processor concurrently solves a sub-LP Problem with disjoint constraints and disjoint variables. The objective of minimizing the border size corresponds to minimizing the size of the serial master problem, whereas maintaining a given balance criterion on the diagonal blocks corresponds to maintaining the given balance criterion on the load of the solution of the subproblems. An HP-based solution to the decomposition problem is presented by Pinar et al. [42] Furthermore, Aykanat et al. [4] proposed a two-phase bipartite-graph model to transform sparse rectangular matrices into SB form, where in the first phase they transform matrix into doubly-bordered block diagonal (DB) form by finding a GPVS on the bipartite graph representation of the rectangular matrix, and at the second phase, they transform into SB form by using $column\text{-}splitting$ technique.

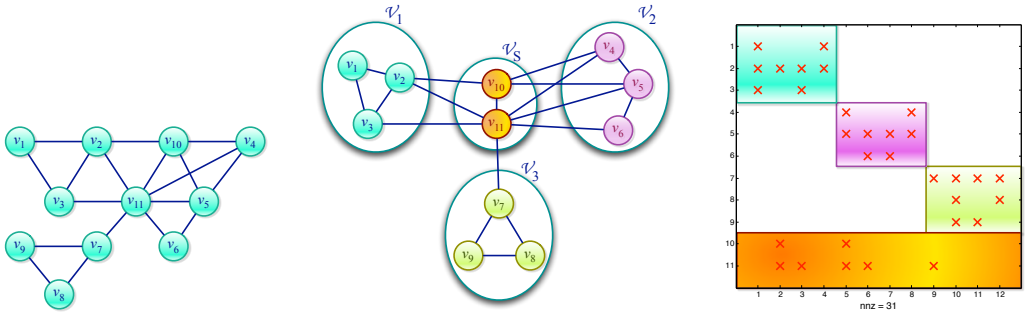
6.2.2.2 GPVS-based Solution

An GPVS-based solution is mentioned firstly by Pinar and Aykanat in 1997 [41]. However, this work lacked to use a vertex separator tool directly.

We are given a $p \times q$ sparse rectangular matrix A, which corresponds to the constraint matrix of an LP Problem. Consider A matrix as a $\{0, 1\}$-matrix. An hypergraph partition on the row-net hypergraph of A matrix forms a SB form A_{SB} of A matrix. Aykanat et al. **??** proposed a HP-based approach to solve this problem. Referring to that work, by exploiting the idea of GPVS-based HP formulation, we can form the A_{SB} form, which corresponds to Dantzig-Wolfe Decomposition, of A matrix by using GPVS as described in Algorithm 11.

Algorithm 11 GPVS-based Dantzig-Wolfe Decomposition Algorithm

Require: A, ε, K
1: $M \rightarrow AA^T$
2: $\Pi_{VS} \leftarrow GPVS(\mathcal{G}(M), \varepsilon, K)$
3: $\Pi_{VS} \rightarrow M_{DB}$
4: $M_{DB} \rightarrow A_{SB}$
5: **return** A_{SB}



(a) (b) (c)

Figure 6.2: (a) The graph $\mathcal{G}(AA^T)$ (b) A vertex separator Π_{VS} of $\mathcal{G}(AA^T)$ (c) The Dantzig-Wolfe Decomposition of A induced by Π_{VS}.

The algorithm works as follows:

1) *Performing* $M = AA^T$*:* This is done by multiplying A matrix with its transpose A^T.

2) *Finding* K*-way Vertex Separator* Π_{VS} *of* $\mathcal{G}(M)$ *with minimum separatorsize under* ε*-balance*

3) *Converting* Π_{VS} *to DB form of* M *matrix:* Π_{VS} induces a DB form M_{DB} of M matrix as described in Section 2.1.2

4) *Decoding* M_{DB} *to* A_{SB}*:* This is straightforward by just using the same row/column permutation of M matrix as the row permutation of A matrix. Column permutation is performed as permuting the columns including nonzero with k'th row-part of A_{SB}/M_{DB} after the columns including nonzero with $(k-1)$'th part of A_{SB}/M_{DB}. Permute the remaining columns at last. Note that here remaining columns will refer to free nodes of row-net hypergraph of A for net-permutation induced by the row-permutation of A_{SB}.

Figure 6.2 displays the methodology as follows. As displayed in Figure 6.2(a), we first construct the graph $\mathcal{G}(M = AA^T)$ for a given matrix A. Then, we find a vertex separator Π_{VS} of $\mathcal{G}$ under the constraints as shown in Figure 6.2(b). Finally, we convert this vertex separator Π_{VS} to a Dantzig-Wolfe Decomposition of A matrix

as in Figure 6.2(c). Note that we didn't need to convert A matrix to hypergraph and decode the vertex separator Π_{VS} to a hypergraph partition.

It should be noted that, the produced Dantzig-Wolfe decomposition A_{SB} of matrix A might not induce a hypergraph partition of row-net hypergraph of A. This is because a row in the border share a column with at least two subblocks. But for Dantzig-Wolfe decomposition, there is no such a constraint. For the graph partitioning by vertex separator problem, the separator does not need to be *narrow* . This property brings a flexibility in the search space. Thus, GPVS formulation is more powerful than HP formulation for Dantzig-Wolfe decomposition of sparse rectangular matrices. The question is that GPVS formulation is exact for this decomposition. Indeed, it is exact. Because, only consideration for the definition of a Dantzig-Wolfe decomposition is column-disjointness of the subblocks of A_{SB}. The only consideration for the definition of a vertex separator is non-adjacency of parts of Π_{VS} on $\mathcal{G}(AA^T)$. These two consideration are reflective to each other. Therefore, we conclude that HP formulation is not perfect for Dantzig-Wolfe decomposition, whereas GPVS formulation is perfect, theoretically. Since the smaller sized border would generally yield a better decomposition, this detail is not very important in practice. So that, using hypergraph partitioning for Dantzig-Wolfe decomposition is still reasonable and valuable.

6.2.2.3 Experimental Results

The experiments are performed on 36 sparse rectangular matrices mentioned in Section 6.2. These matrices arise from Linear Programming Problems. The validity of GPVS-based Dantzig-Wolfe decomposition on these matrices are checked by comparing with a HP-based Dantzig-Wolfe decomposition. For HP-based decomposition, state-of-the-art hypergraph partitioning tool PaToH [11]. However, throughout this work, a GPVS tool is derived from the state-of-the-art fill-reducing ordering tool onmetis [32]. In onmetis, a given graph is recursively bipartitioned and separator removal is applied after each bipartitioning. However, in onmetis, the balance of part is concerned not directly on parts but on parts with separator. Secondly, recursive bipartitioning continues until parts get fairly small. The tool is modified as to make K-way

Table 6.11: The geometric mean results of GPVS-based Dantzig-Wolfe decomposition of LP Problems

	PaToH		dwmetis		dwmetis / PaToH		
						$time$	
K-way	imb	$border$	imb	$border$	$border$	with AA^T	w/o AA^T
2-way	0.01	%1.9	0.01	%2.6	1.32	0.25	0.21
4-way	0.04	%3.5	0.04	%4.8	1.34	0.24	0.21
8-way	0.08	%5.2	0.08	%6.8	1.29	0.22	0.21
16-way	0.09	%6.9	0.10	%8.9	1.29	0.21	0.20
32-way	0.14	%10.5	0.15	%13.3	1.26	0.21	0.20
64-way	0.21	%14.9	0.21	%18.7	1.25	0.21	0.20
128-way	0.28	%20.3	0.29	%23.8	1.17	0.21	0.19

partition and concern the balance on parts, directly and named as $vsmetis$. For GPVS-based decomposition, vsmetis is used in experiments. To define the problem clearly, for a given sparse rectangular matrix A, we are trying to find a Dantzig-Wolfe decomposition (or SB form) of the matrix A, minimizing the border size such that the number of rows of each block is balanced.

The experiments are performed under different K values. For a given matrix A, the row-net hypergraph of A with unit net weights is used as input to PaToH, whereas, the graph of AA^T with unit vertex weights is used as input to vsmetis. Besides vsmetis, a Dantzig-Wolfe decomposition tool, called as $dwmetis$, is trivially generated that performs Algorithm 11. The execution time of $dwmetis$ consists of constructing $\mathcal{G}(M = AA^T)$, finding a vertex separator by vsmetis and decoding the resultant vertex separator to Dantzig-Wolfe decomposition. The execution time of HP-based Dantzig-Wolfe decomposition tool, which we will still refer as PaToH here, is composed of constructing row-net hypergraph $\mathcal{H}^A$ of A matrix and finding a hypergraph partitioning by PaToH and decoding the resultant hypergraph partitioning to Dantzig-Wolfe decomposition. The experiments are performed as to make the imbalance values on block sizes of the decompositions nearly equal. Thus, the border size determines the quality of the decomposition.

The decomposition performances of dwmetis compared to PaToH is summarized in Table 6.11. In the table, results are summarized under three catagories. In the first and second catagory, the decomposition performance of PaToH and dwmetis is given,

respectively, in terms of imbalance on the row numbers of the blocks and percentage of the border size to total number of rows. At the third catagory, the normalized values of the decomposition performance of dwmetis with respect to PaToH is given in terms of bordersize and execution time. The normalized values for the execution time is presented in two groups. In first group, the AA^T structural multiplication is included to the total time, whereas in the second group it isn't included. As seen in the table, imbalance on the number of rows of the blocks and border size increases as the number of blocks of the decomposition increases, as expected, for both PaToH and dwmetis. As also seen in the table, dwmetis produces larger border sizes with comparable imbalance values. Even though, this seems to contradict the theoretical results, this is possible because the problems are NP-complete and the tools are not perfect. Moreover, PaToH applies heuristics which effectively make moves of columns, whereas dwmetis applies heuristics which effectively make moves of rows of matrix A. Thus, move of columns might yield a superior decomposition in terms of border size. Nonetheless, the ratio between dwmetis and PaToH decreases from 1.32 for 2-way decomposition to 1.17 for 128-way decomposition. When we concentrate on the execution times, it is clearly seen that dwmetis performs drastically faster, that is, around 5 times faster than PaToH. The execution time performance gap between the execution time with and without structural multiplication AA^T operation gets smaller with increasing number of blocks. This is reasonable, because the structural multiplication cost is constant for decompositions of all number of blocks, whereas vertex separator execution time increases as the number of parts (blocks) increases. Tables 6.12 and 6.13 present detailed results for PaToH and dwmetis, respectively, where the number of blocks are selected to be 16, 32, 64 and 128.

Table 6.12: The Dantzig-Wolfe decomposition performance of PaToH

	imb				*border*				*total time* (s)			
	16	32	64	128	16	32	64	128	16	32	64	128
lp_ken_07	0.15	0.21	0.36	0.64	%1.2	%4.4	%9.2	%15.3	0.014	0.024	0.036	0.056
delf_A_36	0.07	0.13	0.14	0.23	%8.9	%11.9	%17.1	%27.6	0.028	0.040	0.063	0.097
lp_pds_02	0.07	0.10	0.13	0.16	%8.0	%10.1	%12.8	%15.1	0.032	0.043	0.057	0.076
fxm2-16	0.07	0.10	0.21	0.34	%4.8	%8.6	%20.5	%32.0	0.046	0.068	0.104	0.139
lp_dfl001	0.31	0.30	0.35	0.31	%22.4	%30.9	%36.9	%41.0	0.126	0.157	0.183	0.216
ge	0.07	0.11	0.17	0.23	%5.9	%7.4	%10.4	%17.8	0.078	0.099	0.135	0.189
nl	0.06	0.07	0.15	0.19	%7.3	%8.5	%10.9	%16.2	0.098	0.112	0.140	0.184
lp_ken_11	0.06	0.06	0.07	0.09	%0.4	%0.7	%1.3	%2.4	0.108	0.125	0.148	0.187
model7	0.09	0.21	0.32	0.43	%14.1	%21.4	%28.1	%33.5	0.094	0.118	0.143	0.165
cq5	0.07	0.11	0.17	0.28	%6.1	%8.3	%13.4	%20.2	0.132	0.155	0.181	0.214
model9	0.12	0.22	0.33	0.46	%12.4	%28.4	%35.1	%40.2	0.131	0.165	0.185	0.206
co5	0.08	0.12	0.22	0.41	%5.9	%9.1	%13.4	%19.0	0.151	0.175	0.202	0.238
p05	0.17	0.16	0.18	0.23	%7.7	%11.4	%17.7	%29.3	0.277	0.298	0.330	0.359
lp_pds_06	0.11	0.13	0.20	0.17	%9.0	%11.2	%13.0	%15.9	0.184	0.219	0.254	0.292
ex3sta1	0.18	0.19	0.30	0.32	%28.3	%34.3	%40.4	%47.2	0.310	0.353	0.387	0.433
cq9	0.08	0.11	0.16	0.22	%5.6	%7.4	%11.0	%16.7	0.342	0.368	0.416	0.469
lp_ken_13	0.04	0.12	0.16	0.19	%0.2	%0.5	%1.7	%3.1	0.352	0.394	0.441	0.508
lpi_gosh	0.27	0.54	1.22	2.52	%18.7	%21.1	%24.7	%30.4	0.204	0.237	0.271	0.302
r05	0.17	0.16	0.19	0.30	%9.3	%13.0	%19.1	%30.4	0.546	0.569	0.570	0.631
lp_pds_10	0.12	0.15	0.17	0.18	%9.4	%11.6	%13.2	%15.5	0.400	0.443	0.495	0.559
co9	0.07	0.11	0.20	0.35	%5.2	%7.1	%11.0	%16.2	0.420	0.451	0.494	0.548
scfxm1-2b	0.08	0.11	0.14	0.15	%5.2	%5.6	%5.8	%6.7	0.174	0.210	0.249	0.303
p010	0.12	0.18	0.19	0.18	%4.2	%7.3	%10.8	%17.3	1.400	1.466	1.510	1.521
fome12	0.17	0.24	0.34	0.34	%15.4	%18.2	%22.7	%31.6	1.213	1.330	1.420	1.532
pltexpa	0.08	0.12	0.15	0.21	%5.4	%6.8	%14.2	%17.7	0.428	0.512	0.621	0.692
model10	0.19	0.32	0.43	0.55	%23.4	%40.0	%45.6	%49.9	0.388	0.462	0.503	0.536
nemswrld	0.11	0.22	0.50	0.90	%20.5	%27.3	%31.0	%35.5	0.706	0.760	0.795	0.833
world	0.12	0.11	0.16	0.23	%4.9	%8.1	%11.3	%13.5	0.727	0.839	0.927	1.016
mod2	0.13	0.10	0.16	0.21	%5.0	%8.2	%11.8	%14.4	0.707	0.817	0.914	1.001
route	0.00	0.08	0.17	0.41	%1.4	%8.9	%17.9	%39.5	6.172	6.524	6.664	6.908
lp_pds_20	0.13	0.14	0.17	0.19	%9.8	%11.7	%13.7	%15.7	1.088	1.221	1.311	1.414
lp_cre_d	0.09	0.14	0.27	0.35	%16.6	%21.8	%30.2	%37.0	2.557	2.654	2.738	2.822
lp_cre_b	0.07	0.15	0.21	0.32	%16.2	%20.0	%26.1	%32.7	2.658	2.743	2.818	2.884
lpl1	0.39	0.53	0.77	1.29	%36.5	%47.9	%57.1	%71.5	1.974	2.121	2.249	2.296
fxm3_16	0.03	0.04	0.06	0.10	%0.4	%2.8	%4.4	%5.9	0.722	0.886	1.036	1.182
stormg2-125	0.15	0.15	0.15	0.18	%9.0	%11.2	%12.8	%14.7	1.449	1.678	1.891	2.114
geomean	**0.09**	**0.14**	**0.21**	**0.28**	**%6.9**	**%10.5**	**%14.9**	**%20.3**				

Table 6.13: The Dantzig-Wolfe decomposition performance of dwmetis

	imb				*border*				*total time* (s)			
	16	32	64	128	16	32	64	128	16	32	64	128
lp_ken_07	0.12	0.16	0.23	0.28	%1.2	%4.1	%8.1	%13.3	0.005	0.007	0.011	0.014
delf_A_36	0.09	0.09	0.13	0.22	%12.2	%13.4	%19.5	%26.2	0.010	0.013	0.017	0.022
lp_pds_02	0.08	0.13	0.16	0.22	%7.3	%10.7	%13.7	%14.8	0.010	0.014	0.017	0.019
fxm2-16	0.07	0.11	0.25	0.31	%4.3	%6.7	%23.6	%32.9	0.015	0.020	0.035	0.047
lp_dfl001	0.06	0.11	0.13	0.28	%46.0	%44.9	%45.8	%47.8	0.028	0.032	0.039	0.042
ge	0.13	0.15	0.19	0.31	%6.6	%8.2	%12.0	%21.7	0.036	0.043	0.051	0.064
nl	0.10	0.13	0.15	0.24	%8.6	%10.5	%18.8	%17.6	0.028	0.032	0.033	0.043
lp_ken_11	0.16	0.11	0.17	0.27	%0.5	%1.0	%1.7	%2.9	0.033	0.039	0.044	0.051
model7	0.07	0.16	0.27	0.33	%20.6	%26.5	%31.7	%32.6	0.027	0.033	0.038	0.039
cq5	0.10	0.12	0.22	0.32	%7.8	%11.5	%17.5	%22.6	0.024	0.028	0.034	0.044
model9	0.14	0.35	0.54	0.52	%17.4	%38.0	%41.7	%46.5	0.024	0.034	0.040	0.044
co5	0.09	0.18	0.20	0.22	%8.8	%15.7	%19.0	%25.3	0.029	0.034	0.038	0.045
p05	0.13	0.14	0.18	0.30	%7.9	%14.6	%18.5	%29.0	0.029	0.033	0.038	0.050
lp_pds_06	0.10	0.19	0.26	0.23	%9.1	%12.0	%17.0	%19.5	0.038	0.046	0.054	0.061
ex3sta1	0.14	0.24	0.28	0.41	%32.7	%39.3	%45.3	%50.6	0.184	0.206	0.224	0.239
cq9	0.10	0.12	0.20	0.30	%7.3	%8.5	%15.0	%20.3	0.051	0.057	0.064	0.076
lp_ken_13	0.09	0.12	0.15	0.17	%0.2	%0.5	%1.8	%3.5	0.084	0.095	0.108	0.121
lpi_gosh	0.22	0.21	0.28	0.37	%23.1	%24.8	%26.3	%29.3	0.043	0.045	0.049	0.051
r05	0.12	0.13	0.18	0.42	%16.2	%17.2	%24.3	%34.7	0.055	0.053	0.061	0.073
lp_pds_10	0.12	0.15	0.19	0.23	%12.2	%14.3	%18.2	%22.1	0.078	0.089	0.100	0.115
co9	0.09	0.12	0.21	0.27	%12.3	%14.3	%14.6	%28.1	0.059	0.064	0.081	0.081
scfxm1-2b	0.07	0.11	0.12	0.14	%5.8	%5.9	%6.1	%7.3	0.137	0.149	0.161	0.174
p010	0.11	0.13	0.14	0.16	%4.5	%7.2	%10.9	%17.7	0.067	0.074	0.080	0.091
fome12	0.14	0.15	0.16	0.28	%32.6	%38.3	%42.6	%46.1	0.204	0.222	0.231	0.241
pltexpa	0.14	0.15	0.24	0.24	%10.5	%13.1	%22.4	%24.2	0.143	0.163	0.191	0.205
model10	0.27	0.41	0.55	0.57	%32.3	%48.1	%51.7	%53.6	0.062	0.071	0.079	0.080
nemswrld	0.13	0.40	0.49	0.45	%23.0	%39.6	%49.0	%52.6	0.100	0.111	0.127	0.130
world	0.15	0.15	0.17	0.21	%8.5	%12.2	%15.6	%18.7	0.257	0.290	0.320	0.345
mod2	0.13	0.16	0.18	0.23	%8.2	%12.1	%16.2	%19.5	0.265	0.299	0.332	0.360
route	0.00	0.05	0.12	0.63	%1.8	%9.2	%22.3	%48.9	0.338	0.403	0.457	0.494
lp_pds_20	0.09	0.14	0.17	0.21	%13.3	%16.0	%18.0	%21.5	0.228	0.255	0.273	0.299
lp_cre_d	0.12	0.19	0.39	0.49	%19.8	%23.4	%42.3	%42.6	0.084	0.097	0.101	0.117
lp_cre_b	0.11	0.16	0.31	0.40	%19.6	%24.2	%31.5	%38.4	0.093	0.103	0.115	0.121
lpl1	0.27	0.57	0.88	1.43	%39.8	%54.0	%63.4	%74.7	1.844	1.986	1.982	2.033
fxm3_16	0.04	0.06	0.08	0.10	%0.3	%3.3	%4.7	%5.5	0.286	0.338	0.371	0.405
stormg2-125	0.18	0.21	0.23	0.29	%12.4	%15.2	%16.1	%17.7	0.753	0.827	0.869	0.918
average	**0.10**	**0.15**	**0.21**	**0.29**	**%8.9**	**%13.3**	**%18.7**	**%23.8**				

Chapter 7

Conclusion and Future Work

Hypergraph Partitioning (HP) and Graph Partitioning by Vertex Separator (GPVS) problems are very well known problems which are used in scientific and parallel computing effectively. A typical problem in parallel computing is to partition the data/tasks into several processors such that the overall performance of the computation gets more qualified in terms of time and/or memory. Besides, GPVS is generally used for fill-reducing ordering of sparse matrices for solving sparse linear systems efficiently which lies in the area of scientific computing. In this thesis, the relation between these two problems are investigated. Two combinatorial reductions, from HP Problem to GPVS Problem and from GPVS Problem to HP Problem are disclosed along with their theoretical bases. In practice, the nontrivial part of HP Problem to GPVS Problem reduction is the input transformation, that is, converting a graph to a hypergraph such that the reduction holds. The nontrivial part of the reduction from GPVS Problem to HP Problem is the output transformation, that is, decoding a vertex separator of the corresponding graph to a partition for the hypergraph. These nontrivial parts are investigated deeply and effective and efficient algorithms and methods are proposed. Furthermore, "oPaToH", an HP-based ordering tool based on PaToH, is enhanced along with implementation of input transformations. Besides, based on fill-reducing ordering tool onmetis, a GPVS-based HP tool "hpmetis" is derived and a Dantzig-Wolfe decomposition tool for efficient parallelizm of linear programming problem solutions

is constructed, which is called as "dwmetis". The fill-reducing ordering results obtained with enhanced oPaToH produced more qualified ordering results such as up to %20 improvements for operation count compared to state-of-the art ordering tools such as onmetis. Note that decreasing operation count relates to performing sparse linear equation solutions faster. The Dantzig-Wolfe decomposition results with dwmetis produced results around 5 times faster than the state-of-the art hypergraph partitioning tool PaToH with comparable quality for net balancing. This is also valuable because the preprocessing overhead is also considered inside the total execution time, generally. As a result, in this work it is showed that parallel and scientific computing applications can be performed faster by exploiting the combinatorial reductions between HP problem and GPVS problem.

As the future work, the following can be considered:

i) GPVS can be used for permuting a sparse rectangular matrix to double bordered block diagonal (DB) form as stated by Aykanat et al [4]. We can exploit HP-based GPVS formulation to permute a sparse rectangular matrix to DB form. Since the graph to be partitioned by vertex separator is a bipartite graph, the only hypergraph is $\mathcal{H}^2$ for using HP-based GPVS formulation which in turn relates to hypergraph partitioning of row-col-net hypergraph of the sparse rectangular matrix to be permuted into DB form.

ii) The most common balance criteria for hypergraph partitioning problem is balancing on node weights. Therefore, GPVS-based HP formulation still can be used as to approximate the balance of node weights by using net weights. Simply, using inverse cut heuristic is anticipated to work well to approximate node balance, which we also confirmed by our preliminary works. The main difficulty is that we need separator be minimized according to one weight function, whereas balance on parts should be done according to another weight function.

iii) In the HP-based GPVS approach, only the construction of $\mathcal{H}^2$, $\mathcal{H}^3$ and $\mathcal{H}^4$ is considered in this work. However, a more general hypergraph construction algorithm can be considered for the use of HP-based GPVS formulation. Furthermore, these algorithms can be specialized such to produce better vertex separators.

Bibliography

[1] C. J. Alpert and A. B. Kahng. Recent directions in netlist partitioning: A survey. *VLSI Journal*, 19(1–2):1–81, 1995.

[2] P. Amestoy, T. Davis, and I. Duff. An approximate minimum degree ordering algorithm. Technical Report TR-94-039, University of Florida, Dec 1994.

[3] C. Ashcraft and J. W. H. Liu. Applications of the dulmage-mendelsohn decomposition and network flow to graph bisection improvement. *SIAM Journal on Matrix Analysis and Applications*, 19(2):325–354, 1998.

[4] C. Aykanat, A. Pınar, and U. V. Çatalyürek. Permuting sparse rectangular matrices into block-diagonal form. *SIAM Journal on Scientific Computing*, 26(6):1860–1879, 2004.

[5] A. Brandstadt, V. B. Le, and J. P. Spinrad. *Graph Classes: A Surway*. SIAM, 1999.

[6] T. N. Bui and C. Jones. Finding good approximate vertex and edge partitions is NP-hard. *Infomation Processing Letters*, 42(3):153–159, May 1992.

[7] T. N. Bui and C. Jones. A heuristic for reducing fill in sparse matrix factorization. In *Proc. 6th SIAM Conf. Parallel Processing for Scientific Computing*, pages 445–452, 1993.

[8] U. Catalyurek, C. Aykanat, and E. Kayaaslan. Hypergraph partitioning-based fill-reducing ordering. *submitted to SIAM Journal on Scientific Computing*, 2009.

[9] U. V. Çatalyürek. A constructive multi-way circuit partitioning algorithm based on minimum degree ordering. Master's thesis, Bilkent University, Computer Engineering and Information Science, Sep 1994.

[10] U. V. Çatalyürek. *Hypergraph Models for Sparse Matrix Partitioning and Reordering*. PhD thesis, Bilkent University, Computer Engineering and Information Science, Nov 1999.

[11] U. V. Çatalyürek and C. Aykanat. *PaToH: A Multilevel Hypergraph Partitioning Tool, Version 3.0*. Bilkent University, Department of Computer Engineering, Ankara, 06533 Turkey. PaToH is available at `http://bmi.osu.edu/~umit/software.htm`, 1999.

[12] C. Chevalier and F. Pellegrini. Pt-scotch: A tool for efficient parallel graph ordering. *Parallel Computing*, 34:318–331, 2008.

[13] J. Cong, L. Hagen, and A. B. Kahng. Net partitions yield better module partitions. In *Proceedings of 29th ACM/IEEE Design Automation Conference*, pages 47–52, 1992.

[14] J. Cong, W. Labio, and N. Shivakumar. Multi-way vlsi circuit partitioning based on dual net representation. In *Proceedings of IEEE International Conference on Computer-Aided Design*, pages 56–62, 1994.

[15] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, McGraw-Hill, New York, NY, 1990.

[16] G. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

[17] T. Davis. University of Florida sparse matrix collection: `http://www.cise.ufl.edu/research/sparse/`. *NA Digest*, 92/96/97(42/28/23), 1994/1996/1997.

[18] M. C. Ferris and J. D.Horn. Partitioning mathematical programs for parallel solution. *Mathematical Programming*, 80:35–61, 1998.

[19] M. R. Garey and D. S. Johnson. *Computers and Intractability*. W.H. Freeman and Co., New York, New York, 1979.

[20] J. A. George. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, 10:345–363, 1973.

[21] J. A. George and J. W. H. Liu. *Computer solution of large sparse positive definite systems*. Prentice-Hall, 1981.

[22] S. K. Gnanendran and J. K. Ho. Load balancing in the parallel optimization of block-angular linear programs,. *Mathematical Programming*, 62:41–67, 1993.

[23] J. Gramm, J. Guo, F. Huffner, and R. Niedermeier. Data reduction, exact, and heuristic algorithms for clique cover. In *Proceedings of the Eighth Workshop on Algorithm Engineering*, 2006.

[24] A. Gupta. Fast and effective algorithms for graph partitioning and sparse matrix ordering. Technical Report RC 20453, IBM T. J. Watson Research Center, Yorktown Heights, NY, 1996.

[25] A. Gupta. Watson graph partitioning package. Technical Report RC 20453, IBM T. J. Watson Research Center, Yorktown Heights, NY, 1996.

[26] B. Hendrickson and R. Leland. A multilevel algorithm for partitioning graphs. In *Proc. Supercomputing '95*. ACM, December 1995.

[27] B. Hendrickson and E. Rothberg. Effective sparse matrix ordering: just around the bend. In *Proc. Eighth SIAM Conf. Parallel Processing for Scientific Computing*, 1997.

[28] J. K. Ho, T. C. Lee, and R. P. Sundarraj. Decomposition of linear programs using parallel computation. *Mathematical Programming*, 42:391–405, 1988.

[29] A. B. Kahng. Fast hypergraph partition. In *Proceedings of the 26th ACM/IEEE Design Automation Conference*, pages 762–766, 1989.

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

[31] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Technical Report TR 95-035, Department of Computer Science, University of Minnesota, 1995.

[32] G. Karypis and V. Kumar. *MeTiS A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices Version 4.0*. University of Minnesota, Department of Comp. Sci. and Eng., Army HPC Research Center, Minneapolis, 1998.

[33] E. Kellerman. Determination of keyword conflict. *IBM Technical Disclosure Bulletin*, 1973.

[34] L. Kou, L. Stockmeyer, and C.K.Wong. Covering edges by cliques with regard to keyword conflicts and intersection graphs. *Communications of the ACM*, February 1978.

[35] C. E. Leiserson and J. G. Lewis. Orderings for parallel sparse symmetric matrix factorization. In *Third SIAM Conference on Parallel Processing for Scientific Computing*, pages 27–31, 1987.

[36] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. Willey–Teubner, Chichester, U.K., 1990.

[37] J. W. H. Liu. Modification of the minimum degree algorithm by multiple elimination. *ACM Transactions on Mathematical Software*, 11:141–153, 1985.

[38] J. W. H. Liu. The minimum degree ordering with constraints. *SIAM Journal on Scientific and Statistical Computing*, 10:1136–1145, 1989.

[39] D. Medhi. Parallel bundle-based decomposition for large-scale structured mathematical programming problems. *Annals of Operations Research*, 22(101–127), 1990.

[40] D. Medhi. Bundle-based decomposition for structured large-scale convex optimization: Error estimate and application to block-angular linear programs. *Mathematical Programming*, 66:79–101, 1994.

[41] A. Pınar and C. Aykanat. An effective model to decompose linear programs for parallel solution. In *Proceedings of the Third International Workshop on Applied Parallel Computing, PARA'96*, volume 1184 of *Lecture Notes in Computer Science*, pages 592–601. Springer-Verlag, 1997.

[42] A. Pınar, U. V. Çatalyürek, C. Aykanat, and M. Pınar. Decomposing linear programs for parallel solution. *Lecture Notes in Computer Science*, 1041:473–482, 1996.

[43] D. J. Rose. *Graph Theory and Computing*, chapter A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations, pages 183–217. Academic Press, 1972.

[44] E. Rothberg. Ordering sparse matrices using approximate minimum local fill. *submitted for publication*, 1996.

[45] W. F. Tinney and J. W. Walker. Direct solution of sparse network equations by optimally ordered triangular factorization. In *Proc. IEEE*, volume 55, pages 1801–1809, 1967.

[46] M. Yannakis. Computing the minimum fill-in is np-complete. *SIAM J. Algebraic Discrete Methods*, 2:77–79, 1981.

Appendix A

Main Theorem Strictness Results

The following presents the examples where the inequalities given in Theorem 4 might be strict.

(i) HP solution might not correspond any GPVSn solution: Consider HP problem instance $(\mathcal{H}=(\mathcal{U},\mathcal{N}), K=2, \varepsilon=0)$ with unit net costs, where

- $\mathcal{U} = \{u_1, u_2, u_3, u_4\}$
- $\mathcal{N} = \{n_1, n_2, n_3, n_4\}$
- $Pins(n_1) = \{u_1, u_2\}$
- $Pins(n_2) = \{u_3, u_4\}$
- $Pins(n_3) = \{u_3\}$
- $Pins(n_4) = \{u_4\}$

The node-partition $\Pi_{HP} = \{\mathcal{U}_1=\{u_1, u_3\}, \mathcal{U}_2=\{u_2, u_4\}\}$ forms a feasible solution for the given HP problem instance, whereas there exists no feasible GPVSn solution for the GPVSn problem instance $(\mathcal{G}=NIG(\mathcal{H}), K=2, \varepsilon=0)$ with unit vertex weights. □

(ii) GPVS solution might not correspond any HP solution: Consider HP problem instance $(\mathcal{H}=(\mathcal{U},\mathcal{N}), K=2, \varepsilon=0)$ with unit net costs, where

- $\mathcal{U} = \{u_1, u_2, u_3\}$
- $\mathcal{N} = \{n_1, n_2, n_3\}$
- $Pins(n_1) = \{u_1\}$
- $Pins(n_2) = \{u_2\}$
- $Pins(n_3) = \{u_3\}$

The vertex separator $\Pi_{VS} = \{\, \mathcal{V}_1 = \{v_1\},\ \mathcal{V}_2 = \{v_2\};\ \mathcal{V}_S = \{v_3\}\}$ forms a feasible GPVS solution for the GPVS problem instance $(\mathcal{G} = NIG(\mathcal{H}), K = 2, \varepsilon = 0)$ with unit vertex weights, whereas there exists no feasible HP solution for the given HP problem instance.

Appendix B

Properties of Net-Partition

The following presents the proof of the Theorem 12.

Theorem 12 *Given a hypergraph* $\mathcal{H} = (\mathcal{U}, \mathcal{N})$ *and a net-partition* $\Pi_{HP_N} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$ *of* $\mathcal{H}$*, consider the partial node partition* $\Pi^p_{HP_U}$ *of* $\mathcal{H}$ *for* Π_{HP_N}*. For any free node assignment* f*;*

(i) $\mathcal{N}^f = \mathcal{N}$

Since all nodes has assigned to some part, all nets are either external or also assigned to some part.

(ii) $\mathcal{N}_k \subseteq \mathcal{N}^f_k$, for $1 \leq k \leq K$

Since all pins of an internal net n_j of $\mathcal{N}_k$ is an element of $\mathcal{U}_k$, n_j is also an internal net of $\mathcal{N}^f_k$. If all free nodes of a net $n_j \in \mathcal{N}_S$ is assigned to part $\mathcal{U}_k$, than n_j is also an internal net of $\mathcal{N}^f_k$. Therefore $\mathcal{N}_k \subseteq \mathcal{N}^f_k$, for $1 \leq k \leq K$.

(iii) $\mathcal{N}^f_S \subseteq \mathcal{N}_S$

This property can be considered as a corollary of first two properties.

(iv) $\mathcal{N}_\emptyset = \emptyset$

Assuming that there is no node with empty net set and no net with empty pin set, if a net n_j has no free node than it has a non-free node which leads to a

connectivity of v_j. If a vertex v_j has no connectivity than n_j has no non-free node which implies n_j at least one free node in its pin list.

(v) $\mathcal{N}_{I_1} \cap \mathcal{N}_S^f = \emptyset$

This property implies that the nets of S_{I_1} are inevitably internal nets independent of free node partition Π_F. For a net $n_j \in S_{I_1}$, n_j is an internal net of k_j'th part of Π_{HP} corresponding to the vertex-part $\mathcal{V}_{k_j}$ of Π_{VS}, where $\Lambda_j = \{\mathcal{V}_{k_j}\}$. Because all non-free nodes of net n_j lie in node-part $\mathcal{U}_{k_j}$ of Π_{HP} and n_j has no free nodes, all nodes of net n_j lie in node-part $\mathcal{U}_{k_j}$. This yields to that n_j is an internal net of $\mathcal{U}_{k_j}$ of Π_{HP}.

(vi) $\mathcal{N}_{I_2} \cap \mathcal{N}_S^f = \emptyset$

This property implies that the nets of S_{I_2} are also inevitably internal nets but dependent on the free node partition Π_F. For a net $n_j \in S_{I_1}$, n_j is an internal net of k'th part of Π_{HP} corresponding to the node-part $\mathcal{U}_k$, where the only free node $u_i \in \mathcal{F}_j$ is in the free node part $\mathcal{U}_{F_k}$ of Π_F.

(vii) $\mathcal{N}_E \subseteq \mathcal{N}_S^f$

This property implies that the nets of S_C are inevitably external nets. For a net $n_j \in S_{I_1}$, n_j is an internal net of k'th part of Π_{HP} corresponding to the node-part $\mathcal{U}_k$, where the only free node $u_i \in \mathcal{F}_j$ is in the free node part $\mathcal{U}_{F_k}$ of Π_F.