BALANCE PRESERVING MIN-CUT REPLICATION SET FOR A K-WAY HYPERGRAPH PARTITIONING

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

Volkan Yazıcı

September, 2010

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Cevdet Aykanat(Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Özcan Öztürk

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Oya-Ekin Karaşan

Approved for the Institute of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Institute

ABSTRACT

BALANCE PRESERVING MIN-CUT REPLICATION SET FOR A K-WAY HYPERGRAPH PARTITIONING

Volkan Yazıcı
M.S. in Computer Engineering
Supervisor: Prof. Dr. Cevdet Aykanat
September, 2010

Replication is a widely used technique in information retrieval and database systems for providing fault-tolerance and reducing parallelization and processing costs. Combinatorial models based on hypergraph partitioning are proposed for various problems arising in information retrieval and database systems. We consider the possibility of using vertex replication to improve the quality of hypergraph partitioning. In this study, we focus on the *Balance Preserving Min-Cut Replication Set (BPMCRS)* problem, where we are initially given a maximum replication capacity and a K-way hypergraph partition with an initial imbalance ratio. The objective in the BPMCRS problem is finding optimal vertex replication sets for each part of the given partition such that the initial cutsize of the partition is improved as much as possible and the initial imbalance is either preserved or reduced under the given replication capacity constraint. In order to address the BPMCRS problem, we propose a model based on a unique blend of coarsening and integer linear programming (ILP) schemes. This coarsening algorithm is based on the Dulmage-Mendelsohn decomposition. Experiments show that the ILP formulation coupled with the Dulmage-Mendelsohn decomposition-based coarsening provides high quality results in feasible execution times for reducing the cost of a given K-way hypergraph partition.

Keywords: partitioning, hypergraph partitioning, replication.

ÖZET

K PARÇALI BIR HİPERÇİZGE BÖLÜMLEMESİ İÇİN DENGE KORUMALI MİN-KESİT ÇOKLAMA KÜMESİ

Volkan Yazıcı
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Prof. Dr. Cevdet Aykanat
Eylül, 2010

Çoklama, veri erişimi ve veritabanı sistemlerinde aksaklığa dayanıklılık ve paralelizasyon ve işleme yüklerinin azaltılması için sıkça kullanılan bir tekniktir. Veri erişimi ve veritabanı sistemlerinde hiperçizge bölümlemesine dayanan bir çok kombinasyonel model önerilmiştir. Bu çalışmada, düğüm çoklamaları kullanılarak hiperçizge bölümlemelerindeki kesit boyutunun azaltılması üzerinde durmaktayız. Bu amaçla, verilen bir maksimum çoklama kapasitesi ve K parçalı hiperçizge bölümlemesi ile *Denge Korumalı Min-Kesit Çoklama Kümesi'nin (DKMKÇK)* bulunması problemi üzerine yoğunlaşmaktayız. DKMKÇK probleminde amaç, her parça için bulunacak bir çoklama kümesi ile baştaki bölümlemenin dengesini koruyarak kesit boyutunu azaltmaktır. Bu amaçla, küçültme (*coarsening*) ve tamsayı doğrusal programlama (*integer linear programming (ILP)*) yöntemlerinin seçkin bileşiminden oluşan bir model öneriyoruz. Modelde kullanılan küçültme algoritması Dulmage-Mendelsohn ayrışımına dayanmaktadır. Yapılan deneylerde, Dulmage-Mendelsohn ayrışımına dayalı küçültme yöntemi ile birlikte kullanılan ILP formülasyonunun mantıklı çalışma zamanları içinde, verilen bir K parçalı hiperçizge bölümlemesinin kesit boyutunu düğüm çoklamaları ile oldukça yüksek seviyelerde azalttığı gözlemlenmiştir.

Anahtar sözcükler: bölümleme, hiperçizge bölümleme, çoklama.

Acknowledgement

Foremost, I would like to express my sincere gratitude to my advisor Prof. Dr. Cevdet Aykanat for the continuous support of my M.S. study and research, for his patience, motivation, enthusiasm, and immense knowledge.

Besides my advisor, I would like to thank the rest of my thesis committee: Asst. Prof. Dr. Özcan Özturk and Assoc. Prof. Dr. Oya-Ekin Karaşan, for their encouragement, insightful comments, and hard questions.

I also would like to thank Ata Türk, Enver Kayaarslan, Tayfun Küçükyılmaz, Reha Oğuz Selvitopi, Önder Bulut, and Ahmet Camcı, for their valuable contributions, fruitful hints and inspiring discussions.

Anneme, Babama ve Kardeşime...

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the literature, combinatorial models based on hypergraph partitioning are proposed for various complex and irregular problems arising in parallel scientific computing [4, 16, 17, 27, 65, 66], VLSI design [2, 41, 46], information retrieval [15], software engineering [8], and database design [25, 26, 43, 47, 62]. These models formulate an original problem as a hypergraph partitioning problem, trying to optimize a certain objective function (e.g., minimizing the total volume of communication in parallel volume rendering, optimizing the placement of circuitry on a die area, minimizing the access to disk pages in processing GIS queries) while maintaining a constraint (e.g., balancing the computational load in a parallel system, using disk page capacities as an upper bound in data allocation) imposed by the problem. In general, the solution quality of the hypergraph partitioning problem directly relates to the formulated problem. Hence, efficient and effective hypergraph partitioning algorithms are important for many applications.

Combinatorial models based on hypergraph partitioning can broadly be categorized into two groups. In the former group, which we call as undirectional hypergraph partitioning models, hypergraphs are used to model a shared relation among the tasks or data represented by the vertices. For instance, hypergraph partitioning models used in database design, information retrieval [15], and GIS queries [25, 26] can be categorized in this group. In the latter group, which we call as directional hypergraph partitioning models, hypergraphs are used to model a

directional (source-destination) relation among the tasks or data represented by the vertices. For example, hypergraph partitioning models used in matrix vector multiplication [18, 19, 70] and VLSI design [2, 41, 46] can be categorized in this group. In this study, we focus on the undirectional hypergraph partitioning models. Directional hypergraph partitioning models are out of the scope of this work.

Replication is a widely used technique in information retrieval and database systems. This technique is generally used for providing fault-tolerance (e.g., maximizing the availability of data in case of a disk failure) and reducing parallelization (e.g., minimizing communication costs in information retrieval systems) and processing (e.g., minimizing disk access costs of a database system) costs. We consider the possibility of using vertex replication to improve the quality of partitioning objective in undirectional hypergraph models. We refer to this problem as *hypergraph partitioning with vertex replication* and there are two viable approaches to this problem. In the first approach, which we call as *one-phase*, replication is performed concurrently with the partitioning. A concurrent work proposes a heuristic for this problem in [60]. In the second approach, which we call as *two-phase*, replication is performed in two separate phases: In the first phase, hypergraph is partitioned and in the second phase, replication is applied to the partition produced in the previous phase. In this study, we propose an efficient and effective replication phase based on a unique blend of an integer linear programming (ILP) formulation and a coarsening algorithm. This coarsening algorithm is based on the Dulmage-Mendelsohn decomposition. In this approach, we iterate over available parts and try to find replication sets corresponding to the vertices that are to be replicated into iterated parts. Replication set of each part is constrained by a maximum replication capacity. Replication sets should be determined in such a way that the partition imbalance is preserved after the replication.

In the literature, there are various studies for replication in different domains. Below we discuss related studies from VLSI design, relational and spatial databases, and information retrieval domains.

In VLSI design, the first in-depth discussion about logic replication is given by [56], where they propose a heuristic approach. Later, [44] and [51] extend the Fidduccia and Mattheyses (FM) iterative improvement algorithm to allow vertices to be duplicated during partitioning. [37] proposes a network flow model to the optimal replication for min-cut partitioning, and an FM based heuristic to the size constrained min-cut replication problem. [45] introduces the concept of functional replication. [69] provides an optimal solution to the min-area min-cut replication problem. [3] presents a survey about circuit partitioning and provide a brief list of existing logic replication schemes. [31] provides enhancements for available gate replication heuristics.

Replication is a well-studied topic in database literature as well, and it is generally coupled with reliability, fault recovery, and parallelization. There are various publications about distributed databases and replication dating back to mid-70s [22, 40, 61]. A majority of these studies are concerned about fault recovery and thus apply full replication of the whole database. [53] presents a survey of current state of the art technologies in distributed database systems. As noted by [33], database systems encapsulate major implications within itself (e.g. transaction management [7], recoverability [9, 23], and serializability [7]) and considering the dynamic nature of the databases, studied methodologies in distributed database replication mainly focus on the consistency issues. With the impact of geographical information systems in the past decade, there has been a growing interest in the storage modelling of large-scale spatial network databases [25, 26] and multidimensional access methods [32, 58]. [63] provides models for declustering and load-balancing in parallel geographic information systems. [59] gives a survey of data partitioning and replication management in distributed geographical information systems. [57] provides a survey of replicated declustering schemes for spatial data. [35] presents a selective data replication scheme for distributed geographical data sets.

Another application area where replication is dubbed as indispensable is search and information retrieval systems. There are many surveys [5, 34, 50, 67, 68] investigating the fundamental concepts of the field. With the growing need for performance and wide acceptance of distributed computing, traditional

information retrieval concepts are augmented for scalability, parallelization and fault-tolerance purposes. Caching, clustering and replication concepts are utilized to enhance these architectures. [36] proposes a text retrieval system which utilizes clustering and full replication of the data structures for scalability purposes. [48] gives a comparison of replication and caching approaches for information retrieval systems. [6] presents an overview of the clustering architecture deployed at Google, where they exploit replication via sharding through clusters. [12, 13, 14] present distributed information retrieval architectures utilizing different clustering and replication models. They present their findings on the effects of networking and query distribution over the performance of replication and clustering. [49] proposes a pipelined distributed information retrieval model, where a naive partial replication scheme duplicating the most frequent terms on all disks.

Chapter 2

Preliminaries

In this chapter, notations that will be used throughout the thesis and the Dulmage-Mendelsohn decomposition is given. In Section 2.1, K-way hypergraph partitioning is presented. Next, in Section 2.2, partitioning with vertex replication is given. Finally, in Section 2.3, a brief explanation of the Dulmage-Mendelsohn decomposition will be shown.

2.1 K-Way Hypergraph Partitioning

A hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ is defined as a two-tuple, where $\mathcal{V}$ denotes the set of vertices and $\mathcal{N}$ denotes the set of nets (hyperedges) among those vertices. Every net $n \in \mathcal{N}$ connects a subset of vertices. The vertices connected by a net n are called its *pins* and denoted as $Pins(n) \subseteq \mathcal{V}$. Two vertices are said to be *adjacent* if they are connected by at least one common net. That is, $v \in Adj(u)$ if there exists a net n such that $u, v \in Pins(n)$. A weight $w(v)$ and a cost $c(n)$ are assigned for each vertex v and net n, respectively. Adjacency $Adj(\cdot)$ and weight $w(\cdot)$ operators easily extend to a set U of vertices, that is, $Adj(U) = \left(\bigcup_{u \in U} Adj(u)\right) - U$ and $w(U) = \sum_{v \in U} w(v)$.

A K-way vertex partition of $\mathcal{H}$ is denoted as $\Pi(\mathcal{V}) = \{V_1, V_2, \ldots, V_K\}$. Here,

parts $V_k \subseteq \mathcal{V}$, for $k = 1, 2, \ldots, K$, are pairwise disjoint and mutually exhaustive. In a partition Π of $\mathcal{H}$, a net that connects at least one vertex in a part is said to *connect* that part. The *connectivity set* $\Lambda(n)$ of a net n is defined as the set of parts connected by n. The *connectivity* $\lambda(n) = |\Lambda(n)|$ of a net n denotes the number of parts connected by n. A net n is said to be *cut* if it connects more than one part (i.e., $\lambda(n) > 1$), and *uncut* otherwise (i.e., $\lambda(n) = 1$). The cut and uncut nets are also referred to as *external* and *internal* nets, respectively. $N_{ext}(V_k)$ denotes the set of external nets of part V_k. A vertex is said to be a *boundary* vertex if it is connected by at least one cut net.

For a K-way partition Π of a given hypergraph $\mathcal{H}$, imbalance ratio $ibr(\Pi)$ is defined as follows:

$$ibr(\Pi) = \frac{W_{max}}{W_{avg}} - 1.$$

Here, $W_{max} = \max_{V_k \in \Pi}\{w(V_k)\}$ and $W_{avg} = W_{tot}/K$, where $W_{tot} = w(\mathcal{V})$.

There are various *cutsize* metrics for representing the cost $\chi(\Pi)$ of a partition Π. Two most widely used cutsize metrics are given below.

- *Cut-net* metric: The cutsize is equal to the sum of the costs of the cut nets.

$$\chi(\Pi) = \sum_{n \in \mathcal{N}_{ext}} c(n) \tag{2.1}$$

- *Connectivity* metric: Each cut net n contributes $(\lambda(n) - 1)c(n)$ to the cutsize.

$$\chi(\Pi) = \sum_{n \in \mathcal{N}_{ext}} (\lambda(n) - 1)\, c(n) \tag{2.2}$$

Given these definitions, the K-way hypergraph partitioning problem is defined as follows.

Definition 1 K-Way Hypergraph Partition. *Given a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N})$, number of parts K, a maximum imbalance ratio ε, and a cutsize metric $\chi(\cdot)$; find*

a K-way partition Π of $\mathcal{H}$ that minimizes $\chi(\Pi)$ subject to the balancing constraint $ibr(\Pi) \leq \varepsilon$.

This problem is known [46] to be an $\mathcal{NP}$-hard problem.

2.2 K-Way Hypergraph Partitioning With Vertex Replication

For a given K-way partition Π of $\mathcal{H}$, $\mathcal{R}(\Pi) = \{R_1, R_2, \ldots, R_K\}$ denotes the *replication set*, where $R_k \subseteq \mathcal{V}$ and $R_k \cap V_k \neq \emptyset$, for $k = 1, 2, \ldots, K$. That is, R_k denotes the subset of vertices added to part V_k of Π as *replicated* vertices. Note that replication subsets are possibly pairwise overlapping since a vertex might be replicated in more than one part. The replication set $\mathcal{R}(\Pi)$ for a given partition Π of $\mathcal{H}$ induces the following *K-way hypergraph partition with vertex replication*:

$$\Pi^r(\Pi, \mathcal{R}) = \{V_1^r = V_1 \cup R_1,\ V_2^r = V_2 \cup R_2,\ \ldots,\ V_K^r = V_K \cup R_K\}.$$

Note that although V_k's of Π are pairwise disjoint, V_k^r's of Π^r are overlapping. Previously defined $\chi(\cdot)$ and $ibr(\cdot)$ functions are directly applicable to Π^r without any changes. The total weight after replication is defined as $W_{tot}^r = W_{tot} + \sum_{R_k \in \mathcal{R}} w(R_k)$. The main problem addressed in this paper is the following.

Problem 1 Balance Preserving Min-Cut Replication Set (BPMCRS) for a K-Way Hypergraph Partition. *Given a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N})$, a K-way partition Π of $\mathcal{H}$, and a replication capacity ratio ρ; find a K-way replication set $\mathcal{R}(\Pi)$ that minimizes the cutsize $\chi(\Pi^r)$ of the induced replicated partition Π^r subject to the replication capacity constraint of $W_{tot}^r \leq (1+\rho)W_{tot}$ and the balancing constraint of $ibr(\Pi^r) \leq ibr(\Pi)$.*

Even without the balancing constraint, the min-cut replication set (MCRS) problem is known [38] to be $\mathcal{NP}$-hard. Alternative to the proof of Hwang [38],

a simple transformation of the set-union knapsack (SUK) problem – which is known [42] to be $\mathcal{NP}$-hard – to the MCRS problem is presented in Appendix A.

2.3 The Dulmage-Mendelsohn Decomposition

The Dulmage-Mendelsohn (DM) decomposition is a canonical decomposition on bipartite graphs and described in a series of papers [28, 29, 30, 39] by Dulmage, Johnson, and Mendelsohn. Pothen and Fan [55] formalized this decomposition by a series of lemmas and explained their enhancements.

A *bipartite graph* $\mathcal{G} = (\mathcal{V} = \mathcal{R} \cup \mathcal{C}, \mathcal{E})$ is a graph whose vertex set $\mathcal{V}$ is partitioned into two parts $\mathcal{R}$ and $\mathcal{C}$ such that the edges in $\mathcal{E}$ connect vertices in two different parts. A *matching* on a bipartite graph is a subset of its edges without any common vertices. A *maximum matching* is a matching that contains the largest possible number of edges.

Definition 2 The Dulmage-Mendelsohn Decomposition. *Let $\mathcal{M}$ be a maximum matching for a bipartite graph $\mathcal{G} = (\mathcal{V} = \mathcal{R} \cup \mathcal{C}, \mathcal{E})$. The Dulmage-Mendelsohn decomposition canonically decomposes $\mathcal{G}$ into three parts*

$$\Pi = \{V_H = R_H \cup C_H,\, V_S = R_S \cup C_S,\, V_V = R_V \cup C_V\},$$

where R_H, R_S, R_V and C_H, C_S, C_V respectively are subsets of $\mathcal{R}$ and $\mathcal{C}$ sets with the following definitions based on $\mathcal{M}$:

$$\begin{aligned}
R_V &= \{v_i \in R \mid v_i \textit{ is reachable by an alternating path from some unmatched vertex } v_j \in R\} \\
R_H &= \{v_i \in R \mid v_i \textit{ is reachable by an alternating path from some unmatched vertex } v_j \in C\} \\
R_S &= R - (R_V \cup R_H) \\
C_V &= \{v_i \in C \mid v_i \textit{ is reachable by an alternating path from some unmatched vertex } v_j \in R\} \\
C_H &= \{v_i \in C \mid v_i \textit{ is reachable by an alternating path from some unmatched vertex } v_j \in C\} \\
C_S &= C - (C_V \cup C_H)
\end{aligned}$$

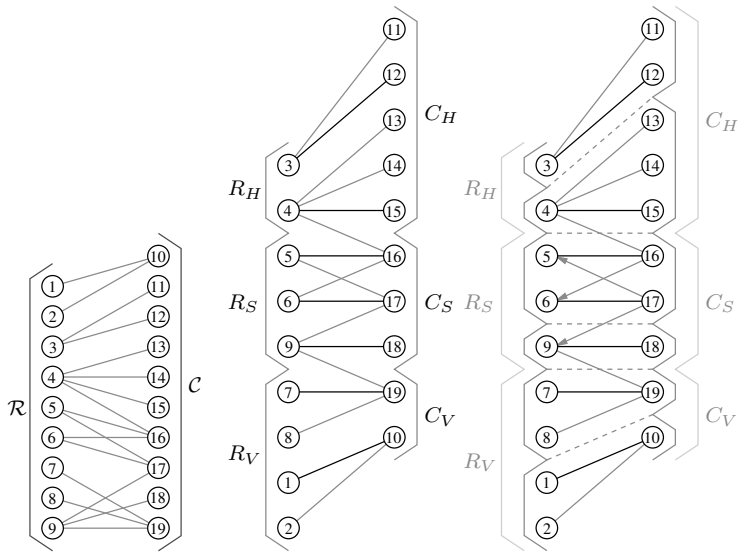
Following properties given in [54, 55] regarding the R_H, R_S, R_V and C_H, C_S, R_S subsets provide certain features related with the structure of the Dulmage-Mendelsohn decomposition. The sets R_V, R_S, and R_H are pairwise disjoint; similarly, the sets C_V, C_S, and C_H are pairwise disjoint. A matching edge of $\mathcal{M}$ connects: a vertex in R_V only to a vertex in C_V; a vertex in R_S only to a vertex in C_S; and a vertex in R_H only to a vertex in C_H. Vertices in R_S are perfectly matched to vertices in C_S. No edge connects: a vertex in C_H to vertices in R_S or R_V; a vertex in C_S to vertices in R_V. C_H and R_V are the unique smallest sets that maximize the $|C_H| - |R_H|$ and $|R_V| - |C_V|$ differences, respectively. The subsets R_H, R_S, R_V and C_H, C_S, C_V are independent of the choice of the maximum matching M; hence the Dulmage-Mendelsohn decomposition is a *canonical decomposition* of the bipartite graph.

For larger bipartite graphs, one might opt for a more *fine-grained decomposition*. For this purpose, Pothen and Fan [55] further decomposes R_H, R_S, R_V and C_H, C_S, C_V sets into smaller subsets. For the simplicity of the forthcoming discussions, the Dulmage-Mendelsohn decomposition will be referred to as *coarse-grained decomposition* and enhancements of Pothen and Fan will be referred to as *fine-grained decomposition*.

G_X denotes a bipartite subgraph of $\mathcal{G}$, where X is one of H, S, or V. That is, for a given bipartite subgraph $G_X = (V_X = R_X \cup C_X, E_X)$, E_X corresponds to the subset of edges in $\mathcal{E}$, which connects either a vertex from R_X to a vertex in C_X, or a vertex from C_X to a vertex in R_X. The fine-grained decomposition is formalized as follows.

Definition 3 Fine-Grained Dulmage-Mendelsohn Decomposition. *Let M be a maximum matching for a bipartite graph $\mathcal{G} = (\mathcal{V} = \mathcal{R} \cup \mathcal{C}, \mathcal{E})$ and G_H, G_S, G_V be bipartite subgraphs induced by the coarse-grained decomposition of $\mathcal{R}$ and $\mathcal{C}$ sets into R_H, R_S, R_V and C_H, C_S, C_V subsets. Fine-grained decomposition of bipartite subgraphs G_H, G_S, and G_V is constructed as follows.*

- *Find connected components in G_H and G_V subgraphs.*
- *Using G_S, construct a new directed bipartite graph G'_S, where matched edges*



(a) Sample bipartite graph. (b) Coarse-grained Dulmage-Mendelsohn decomposition. (c) Fine-grained Dulmage-Mendelsohn decomposition.

Figure 2.1: The Dulmage-Mendelsohn decomposition.

are left undirected, and other unmatched edges are directed from C_S to R_S. Find strongly connected components in G'_S.

Depending on the structure of the given bipartite graph and maximum matching, resultant fine-grained decomposition is expected to provide much more number of partitions than its coarse-grained equivalent.

For a given bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a maximum matching can be found in $O(|\mathcal{E}|\sqrt{|\mathcal{V}|})$ time due to Hopcroft-Karp algorithm. In the coarse-grained decomposition phase, a depth-first search is performed for every unmatched vertex for finding alternating paths. Thus, coarse-grained decomposition runs in $O(|\mathcal{E}|\sqrt{|\mathcal{V}|}) + O(|\mathcal{V}|(|\mathcal{V}| + |\mathcal{E}|))$ time, that is, in $O(|\mathcal{V}|(|\mathcal{V}| + |\mathcal{E}|))$ time. In the fine-grained decomposition phase, connected components for G_H and G_V can be found in $O(|\mathcal{V}|+|\mathcal{E}|)$ time via breadth-first search and strongly-connected components in G'_S can be found in $O(|\mathcal{V}|+|\mathcal{E}|)$ time via Tarjan's algorithm [64]. Hence, decomposition phase takes $O(|\mathcal{V}|(|\mathcal{V}| + |\mathcal{E}|))$ time in total.

In Fig. 2.1, application of coarse-grained and fine-grained Dulmage-Mendelsohn decompositions are demonstrated on a sample bipartite graph $\mathcal{G} = (\mathcal{V} = \mathcal{R} \cup \mathcal{C}, \mathcal{E})$. This sample hypergraph is composed of 19 vertices and 17 undirected edges. Fig. 2.1b demonstrates a coarse-grained Dulmage-Mendelsohn decomposition of $\mathcal{G}$ for a given maximum matching $\mathcal{M}$. Here, matched edges are drawn in black and V_H, V_S, and V_V parts produced by the coarse-grained decomposition are separated via borders. For instance, v_3 is matched with v_{12} and $R_H = \{v_3, v_4\}$ and $C_H = \{v_{11}, v_{12}, v_{13}, v_{14}, v_{15}\}$.

Fig. 2.1c demonstrates a fine-grained decomposition of the sample bipartite graph $\mathcal{G}$ in Fig. 2.1a. Here, components are separated via dashed lines. That is, vertices v_3, v_{11}, v_{12} and edges between them constitute a connected component in G_H. As seen in Fig. 2.1c, unmatched edges (v_5, v_{17}), (v_6, v_{16}), and (v_9, v_{17}) in G_S are directed from C_S to R_S to construct G'_S. There appears two strongly-connected components in G'_S: v_5, v_6, v_{16}, v_{17} and v_9, v_{18}.

Chapter 3

Balance Preserving Min-Cut Replication Set

In this chapter, we propose an efficient and effective approach for solving the BPMCRS problem. It is clear that, given a K-way partition Π of $\mathcal{H}$, only the boundary vertices in Π have the potential of decreasing the cutsize via replication. Thus, only the boundary vertices are considered for finding a good replication set $\mathcal{R}$. In order to be able to handle the balancing constraints on the weights of the parts of the replicated partition, we propose a part-oriented approach by investigating the replications to be performed on each part (in some particular order).

Consider a replication set R_k for a part V_k of Π. Note that R_k has to maximize the reduction in the cutsize without violating the maximum weight constraint of part V_k. It is also clear that, replication of vertices of R_k into part V_k can only decrease the cutsize due to the external nets of part V_k. So, while searching for a good R_k, we consider only the external nets of part V_k and the boundary vertices of other parts that are connected by the external nets of part V_k. That is, we only consider the net set $N_{ext}(V_k)$ and the vertex set $Adj(V_k)$ for finding an R_k.

Algorithm 1 displays a general framework for our approach. As seen in the algorithm, for each part V_k, we first compute the replication capacity κ_k so that

Algorithm 1 FIND_REPLICATION_SET($\mathcal{H}, \Pi, W, \rho$)

1: $\Pi_0^r \leftarrow \Pi$
2: **for** $k \leftarrow 1$ to K **do**
3: $\kappa_k = (1+\rho)W_{avg} - w(V_k)$
4: $\mathcal{H}_k \leftarrow$ CONSTRUCT($\mathcal{H}, k, \Pi_{k-1}^r$)
5: $\mathcal{H}_k^{coarse} \leftarrow$ COARSEN($\mathcal{H}_k$)
6: $R_k \leftarrow$ SELECT($\mathcal{H}_k^{coarse}, \kappa_k$)
7: $\Pi_k^r \leftarrow \{V_1 \cup R_1, \ldots, V_k \cup R_k, V_{k+1}, \ldots, V_K\}$
8: UPDATE(k)
9: **end for**
10: $\Pi_r \leftarrow \Pi_K^r$

the initial imbalance will be preserved or improved after the replication. Then, we construct the hypergraph $\mathcal{H}_k$, which is referred to here as the *boundary adjacency hypergraph*. Vertices of $\mathcal{H}_k$ correspond to $Adj(V_k)$ and nets of $\mathcal{H}_k$ are derived from $N_{ext}(V_k)$. This hypergraph construction process is described in Section 3.1. After constructing $\mathcal{H}_k$, a good R_k is selected from the vertices of $Adj(V_k)$ via using an ILP approach described in Section 3.2. In order to reduce the high computation cost of ILP for large $\mathcal{H}_k$, a novel Dulmage-Mendelsohn decomposition-based coarsening scheme for $\mathcal{H}_k$ is described in Section 3.3.

3.1 Boundary Adjacency Hypergraph Construction

Without loss of generality, here we describe the boundary adjacency hypergraph construction operation to be performed in the kth iteration of our algorithm for the purpose of deciding on the vertices to be replicated into part V_k. Note that prior to this construction process, the effects of the replications performed in the previous iterations are reflected on Π_{k-1}^r (line 7 of Algorithm 1) and the boundary vertices and cut nets are updated accordingly (line 8 of Algorithm 1). For the simplicity of the forthcoming discussions, we use $Adj(V_k)$ and $N_{ext}(V_k)$ to refer to the updated adjacency vertex and external net sets of part V_k, respectively. For example, consider an external net n_j of part V_k in the original partition Π_0^r.

During an earlier iteration $\ell < k$, if all pins of net n_j that lie in part V_k are replicated into part V_ℓ, then net n_j disappears in $N_{ext}(V_k)$. In such a case, those pins of net n_j that lie in part V_ℓ and that are only connected by net n_j to part V_k disappear from $Adj(V_k)$.

Algorithm 2 UPDATE(k)

```
1: for l ← (k + 1) to K do
2:    for each net n_j ∈ N_ext(V_k) do
3:       if (Pins(n_j) ∩ V_ℓ) ∩ R_k ≠ ∅ then
4:          for each vertex v ∈ (Pins(n_j) ∩ V_k) do
5:             if Nets(v) ∩ N_ext(V_ℓ) = {n_j} then
6:                Adj(V_ℓ) = Adj(V_ℓ) − {v}
7:             end if
8:          end for
9:          N_ext(V_ℓ) = N_ext(V_ℓ) − {n_j}
10:         Λ(n_j) = Λ(n_j) − V_ℓ {optional for cut-net metric}
11:      end if
12:   end for
13: end for
```

Two distinct boundary adjacency hypergraphs are required to encapsulate the cut-net (Eq. 2.1) and connectivity (Eq. 2.2) cutsize metrics, which will be referred to as $\mathcal{H}_k^{cut}$ and $\mathcal{H}_k^{con}$, respectively. The construction process for the former and latter are depicted in Algorithms 3 and 4, respectively. In both hypergraphs, the vertex set is composed of $Adj(V_k)$. In both of these hypergraphs, the objective is to find a set of vertices $R_k \subseteq Adj(V_k)$ to be replicated into part V_k, such that the total cost of nets covered by R_k is maximized without violating the balance constraint imposed on V_k. The net set definition for $\mathcal{H}_k^{cut}$ and $\mathcal{H}_k^{con}$ should be done in according to this coverage objective. Note that a net n_j of $\mathcal{H}_k^{cut}/\mathcal{H}_k^{con}$ is said to be covered by R_k if all pins of n_j in $Adj(V_k)$ lie within R_k.

Algorithm 3 CONSTRUCT($\mathcal{H}, k, \Pi_{k-1}^r$) for cut-net metric

```
1: V_k^cut ← Adj(V_k)
2: N_k^cut ← N_ext(V_k)
3: for each net n_j ∈ N_ext(V_k) do
4:    Pins(n_j) ← Pins(n_j) − V_k
5: end for
6: return H_k^cut ← (V_k^cut, N_k^cut)
```

For the cut-net metric, in order to reduce the cutsize related with a net n_j in $N_{ext}(V_K)$, the net n_j should be made internal to part V_k, which is feasible only when all pins of net n_j in $Adj(V_K)$ are replicated into V_k. Thus, the net set of $\mathcal{H}_k^{cut}$ is selected as the external net set of part V_k (line 2 of Algorithm 3). Since $\mathcal{H}_k^{cut}$ is used to find the set of vertices to be replicated into part V_k, the boundary vertices of part V_k should be extracted from the pin list of the nets of $\mathcal{H}_k^{cut}$ (lines 3–4 of Algorithm 3).

Algorithm 4 CONSTRUCT$(\mathcal{H}, k, \Pi_{k-1}^r, \kappa_k)$ for connectivity metric

1: $\mathcal{V}_k^{con} \leftarrow Adj(V_k)$
2: $\mathcal{N}_k^{con} \leftarrow \emptyset$
3: **for** each net $n_j \in N_{ext}(V_k)$ **do**
4: **for** each part $V_\ell \in \Lambda(n_j)$ and $V_\ell \neq V_k$ **do**
5: $\mathcal{N}_k^{con} \leftarrow \mathcal{N}_k^{con} \cup \{n_j^\ell\}$
6: $Pins(n_j^\ell) \leftarrow Pins(n_j) \cap V_\ell$
7: **end for**
8: **end for**
9: **return** $\mathcal{H}_k^{con} \leftarrow (\mathcal{V}_k^{con}, \mathcal{V}_k^{con})$

For the connectivity metric, in order to reduce the cutsize related with a net n_j in $N_{ext}(V_K)$, it is sufficient to replicate a subset of the pins of net n_j so that $\lambda(n_j)$ in Π^r will decrease. That is, number of parts connected by net n_j will decrease after the replication. For this reason, the nets of $\mathcal{H}_k^{con}$ is derived from $N_{ext}(V_k)$ by applying a net splitting operation to each external net in such a way that each external net n_j is split into $\lambda(n_j) - 1$ new nets. This splitting operation is performed as follows: For each net n_j in $N_{ext}(V_k)$, we traverse over the connectivity set $\Lambda(n_j)$ of n_j and introduce a new net n_j^ℓ for each part $V_\ell \neq V_k$ in $\Lambda(n_j)$. The newly introduced net n_j^ℓ is set to connect only those pins of n_j that lie in part V_ℓ (lines 4–6 of Algorithm 4).

Fig. 3.1 shows a 3-way partition of a sample hypergraph $\mathcal{H}$ with 24 boundary vertices and 19 cut nets. In figures, circles denote vertices and dots denote nets, where a number i in a circle denotes a vertex v_i and a number j besides a dot denotes a net n_j. Note that only boundary vertices and cut nets are numbered for the sake of simplicity. Fig. 3.2 shows the boundary adjacency hypergraphs $\mathcal{H}_1^{cut}$ (Fig. 3.2a) and $\mathcal{H}_1^{con}$ (Fig. 3.2b) for part V_1 for cut-net and connectivity metrics,

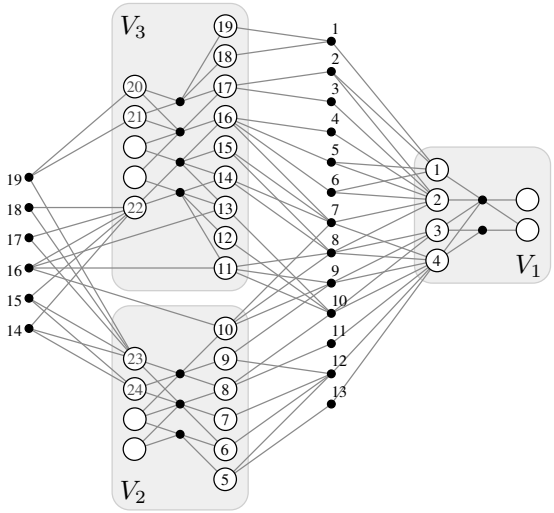
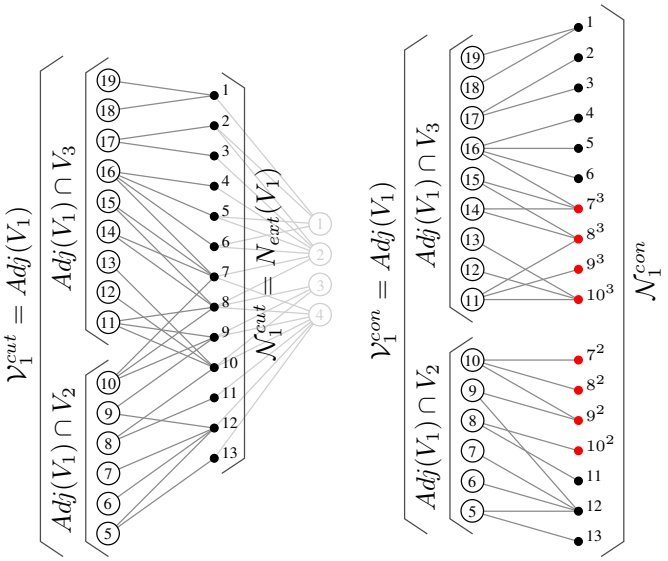


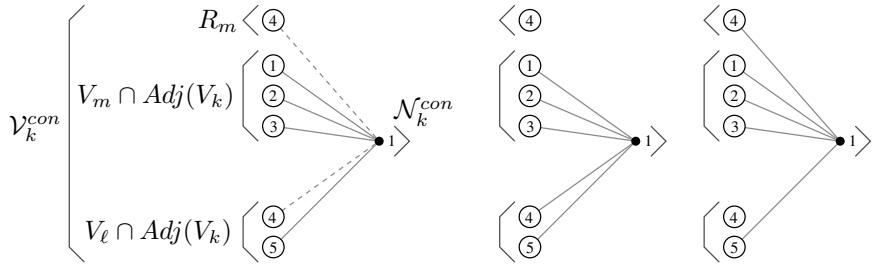
Figure 3.1: A 3-way partition of a sample hypergraph $\mathcal{H}$.



(a) Boundary adjacency hypergraph $\mathcal{H}_1^{cut}$ of part V_1.

(b) Boundary adjacency hypergraph $\mathcal{H}_1^{con}$ of part V_1.

Figure 3.2: Sample boundary adjacency hypergraph construction.



(a) n_1 has multiple choices for v_4 connection.

(b) v_4 of n_1 is selected from V_ℓ.

(c) v_4 of n_1 is selected from R_m.

Figure 3.3: Sample net splitting problem.

respectively. Comparing Fig. 3.1, with Figs. 3.2a and 3.2b shows that V_2's and V_3's boundary vertices $v_5, v_6, \ldots, v_{19}$ that are connected by at least one external net of V_1 constitute the vertices of both $\mathcal{H}_1^{cut}$ and $\mathcal{H}_1^{con}$.

Comparing Fig. 3.1 with Figs. 3.2a and 3.2b shows that each of the external nets $n_1, n_2, \ldots, n_{13}$ of V_1 incurs a net in $\mathcal{H}_1^{cut}$. Similarly, each of the external nets $n_1, n_2, \ldots, n_6$ and n_{11}, n_{12}, n_{13} of V_1, which have a connectivity of 2, incurs a single net in $\mathcal{H}_1^{con}$. On the other hand, each of the external nets n_7, n_8, n_9, n_{10} of V_1, which have a connectivity of 3, incurs 2 nets in $\mathcal{H}_1^{con}$. For example, n_7 with $Pins(n_7) = \{v_{10}, v_{14}, v_{16}\}$ connects both parts V_2 and V_3, and it incurs two nets n_7^2 and n_7^3 in $\mathcal{H}_1^{con}$, where $Pins(n_7^2) = \{v_{10}\}$ and $Pins(n_7^3) = \{v_{14}, v_{16}\}$. Note that n_7^2 and n_7^3 are respectively shown as 7^2 and 7^3 in Fig. 3.2b.

As seen in Fig. 3.2a, net n_9 of $\mathcal{H}_1^{cut}$ is covered by the vertex set $\{v_9, v_{10}, v_{11}\}$. So, the cut-net cutsize related with net n_9 can be reduced only if all of the vertices v_9, v_{10}, v_{11} are replicated into part V_1. On the other hand, as seen in Fig. 3.2b, net n_9^2 of $\mathcal{H}_1^{con}$ is covered by the vertex set $\{v_9, v_{10}\}$ and n_9^3 of $\mathcal{H}_1^{con}$ is covered by the vertex set $\{v_{11}\}$. So, the connectivity cutsize related with net n_9 can be reduced by 1 (assuming unit net costs) either by replicating vertices v_9 and v_{10} into part V_1 or by replicating vertex v_{11} into part V_1. Note that, although the vertex set $\{v_9, v_{10}, v_{11}\}$ covers only net n_9 in $\mathcal{H}_1^{cut}$, it covers nets n_9^2, n_7^3, n_8^3, and n_9^3 in $\mathcal{H}_1^{con}$. So, replicating the vertex set $\{v_9, v_{10}, v_{11}\}$ into part V_1 reduces the cut-net cutsize by 1, whereas, it reduces the connectivity cutsize by 4.

In the first iteration of Algorithm 1, each net splitting is unique in $\mathcal{H}_1^{con}$, since there are no replicated vertices. However, in the following iterations of Algorithm 1, net splittings may not be unique for the further $\mathcal{H}_k^{con}$ constructions because of the replicated vertices. That is, multiple copies of a vertex induces multiple *pin selection* options for a net. And each different pin selection induces a different net splitting in the boundary adjacency hypergraph. Fig. 3.3 shows this pin selection problem that occurs in the construction of $\mathcal{H}_k^{con}$, where vertex v_4 was replicated into part V_m in the mth iteration for $m < k$. Figs. 3.3b and 3.3c show two possible selections for net n_1, which connects to the replicated vertex v_4. In Fig. 3.3b, replication of v_4 and v_5 appear to be necessary to cover n_1, whereas, in Fig. 3.3c, replication of v_5 is sufficient to cover n_1. As depicted in Fig. 3.3, pin selections of nets directly affect the minimization of the number of replicated vertices for covering a particular net. In our proposed model, for a net n_j and vertex $v_j \in Pins(n_j)$, if there exists a copy of v_j in part V_k that was previously replicated into V_k for the purpose of decreasing the connectivity set of $\lambda(n_j)$, then n_j is connected to v_j in part V_k; otherwise, it is connected to the v_j in part V_ℓ that is provided by the initial partitioning. For instance, in Fig. 3.3, if v_4 is replicated to part V_m in a previous iteration for n_1, then n_1 is connected to v_4 in R_m; otherwise, it is connected to v_4 in V_ℓ.

3.2 Vertex Selection in Boundary Adjacency Hypergraph

In our approach, boundary adjacency hypergraph $\mathcal{H}_k = (\mathcal{V}_k, \mathcal{N}_k)$ is derived from the cut nets of part V_k and the adjacent vertices to part V_k. Since nets in $\mathcal{H}_k^{cut}$ and $\mathcal{H}_k^{con}$ correspond to the cut nets for the cut-net and connectivity cutsize metrics, covering these nets has a direct effect on the cutsize related with part V_k. Hence, it is clear that only the vertices in $\mathcal{V}_k$ have the potential of decreasing the cutsize related with part V_k via replication. In this section, our objective is the optimal selection of a subset R_k of vertices in $\mathcal{V}_k$ that are to be replicated into part V_k. Optimality in this context is defined as, given boundary adjacency hypergraph

$\mathcal{H}_k$ and maximum replication capacity κ_k, selecting a subset R_k of vertices in $\mathcal{V}_k$ that maximize the sum of the costs of the covered nets under a given capacity constraint of $w(R_k) \leq \kappa_k$. This *net coverage* objective corresponds to the set-union knapsack problem [42] (SUKP). (See Appendix A for details.) We provide an ILP formulation for this problem as follows.

$$\text{maximize} \quad \sum_{n_j \in \mathcal{N}_k} c(n_j)x(n_j) \tag{3.1}$$

$$\text{subject to} \quad |Pins(n_j)|x(n_j) \leq \sum_{v_i \in Pins(n_j)} y(v_i) \text{ for } \forall n_j \in \mathcal{N}_k \tag{3.2}$$

$$\sum_{v_i \in \mathcal{V}_k} w(v_i)y(v_i) \leq \kappa_k \tag{3.3}$$

$$\text{where} \quad x(n_j) = \begin{cases} 1, & \text{if net } n_j \text{ is covered} \\ 0, & \text{otherwise} \end{cases}$$

$$y(v_i) = \begin{cases} 1, & \text{if vertex } v_i \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

Binary variable $x(n_j)$ is set to 1, if a net n_j is covered by the selected vertices. Likewise, if a vertex v_i is selected for replication, binary variable $y(v_i)$ is set to 1. Objective (3.1) tries to maximize the sum of the cost $c(n_j)$ of every covered net for which $x(n_j) = 1$. Inequality (3.2) constrains a net n_j to be covered if all of its pins are selected, i.e., net n_j is covered if $y(v_i) = 1$ for every $v_i \in Pins(n_j)$. In expression (3.3), the sum of the weights of the selected vertices are constrained by κ_k. Since there are no restrictions on vertex replications, but inequality (3.3), formulation might produce redundant vertex replications as much as κ_k allows. That is, for certain vertices v_i, $y(v_i)$ can be set to 1, where v_i is not contained by the adjacencies of the covered nets. But once the set of $x(n_j)$ is computed, necessary $y(v_i)$ values can be extracted from $Pins(n_j)$ without allowing any redundant vertex replications.

In given ILP formulation, for each boundary adjacency hypergraph $\mathcal{H}_k$, there are $|\mathcal{V}_k| + |\mathcal{N}_k|$ variables for $x(n_j)$ and $y(n_j)$, and $|\mathcal{N}_k| + 1$ constraints (inequalities (3.2) and (3.3)), and a single maximization objective.

ILP model formalized in expressions (3.1)–(3.3) provides the optimal net coverage for a given boundary adjacency hypergraph $\mathcal{H}_k$ and maximum replication capacity κ_k. In Appendix A, the relation between set-union knapsack problem and net coverage in boundary adjacency hypergraph is detailed and it is proved that the net coverage problem is an $\mathcal{NP}$-hard problem. Hence, from a practical point of view, this formulation is expected to consume a significant amount of time as the sum of input variables – $|\mathcal{V}_k|$ and $|\mathcal{N}_k|$ – increase in size. To reduce this high computation cost of the ILP phase, below preprocessing procedures are introduced and applied to $\mathcal{H}_k$ at each iteration before the vertex selection process for replication.

1. Remove infeasible nets (that κ_k isn't sufficient to cover via vertex replication) and the vertices that are only connected by such nets.
2. Use heuristics to coarsen boundary adjacency hypergraph into a smaller hypergraph.
3. Restrict ILP solver running time to a certain duration.

3.3 Coarsening of Boundary Adjacency Hypergraph

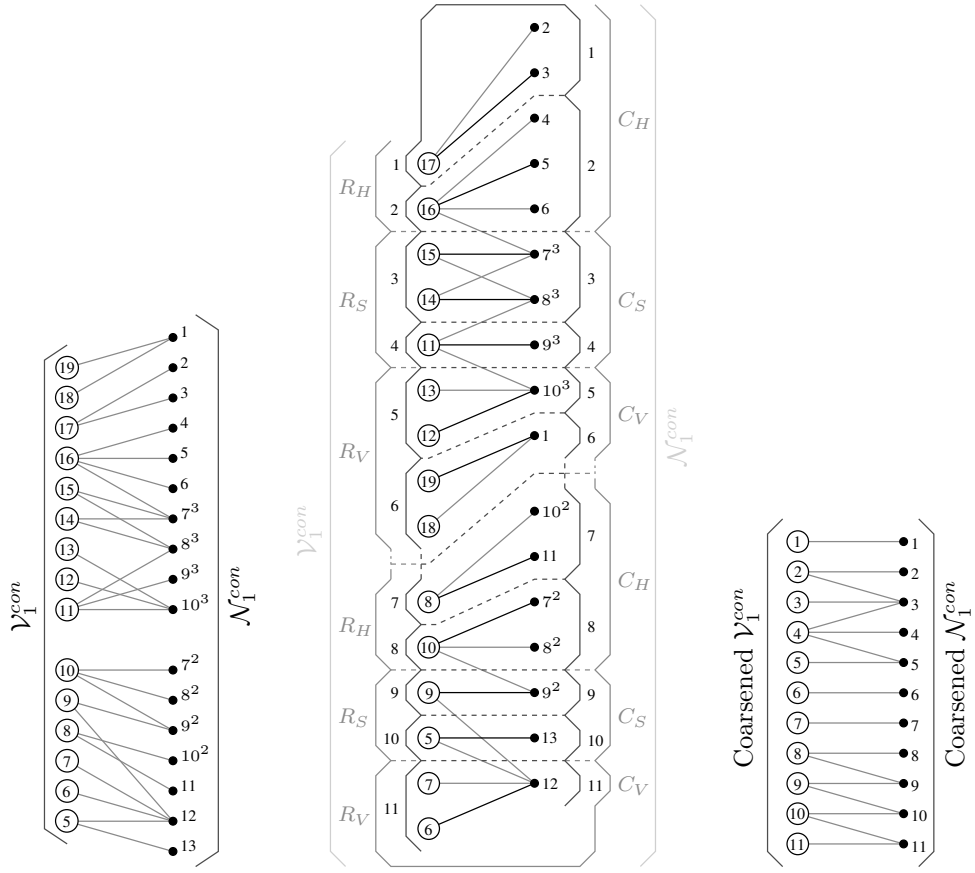
In order to reduce the high computation cost of the ILP phase, we propose an effective coarsening approach based on the Dulmage-Mendelsohn decomposition. At kth iteration of the algorithm, we coarsen the boundary adjacency hypergraph $\mathcal{H}_k$ to $\mathcal{H}_k^{coarse}$. Then, instead of $\mathcal{H}_k$, we pass this $\mathcal{H}_k^{coarse}$ to the ILP solver.

The Dulmage-Mendelsohn decomposition operates on bipartite graphs $\mathcal{G} = (\mathcal{V} = \mathcal{R} \cup \mathcal{C}, \mathcal{E})$, hence, each boundary adjacency hypergraph $\mathcal{H}_k = (\mathcal{V}_k, \mathcal{N}_k)$ is represented in terms of its bipartite graph equivalent $\mathcal{G}_k = (\mathcal{V}_k = \mathcal{R}_k \cup \mathcal{C}_k, \mathcal{E}_k)$ for coarsening. Vertices $\mathcal{V}_k$ and nets $\mathcal{N}_k$ in $\mathcal{H}_k$ constitute the $\mathcal{R}_k$ and $\mathcal{C}_k$ sets in $\mathcal{G}_k$, respectively. That is, for a vertex $v_i \in \mathcal{V}_k$ there is a corresponding vertex $v_{v_i} \in \mathcal{R}_k$ and for a net $n_j \in \mathcal{N}_k$ there is a corresponding vertex $v_{n_j} \in \mathcal{C}_k$. Pins

between nets and vertices constitute the edge set $\mathcal{E}_k$ of $\mathcal{G}_k$. That is, for a net $n_j \in \mathcal{N}_k$ and $v_i \in Pins(n_j)$ there is an undirected edge (v_{v_i}, v_{n_j}) in $\mathcal{E}_k$. After the decomposition, clusters in $\mathcal{G}_k$ are easily projected back to $\mathcal{H}_k$ by reversing back the transformation.

Vertex selection in boundary adjacency hypergraph is constrained by the total weight of the selected vertices for replication and its objective is to maximize the cost of the nets covered. Thus, our objective in the coarsening phase is to cluster vertices and nets in such a way that the vertex groups with similar net coverage characteristics get clustered together. Characterization in this context is intuitively estimated as a ratio between the number of vertices in the cluster and the nets covered by these vertices. That is, clusters with small number of vertices covering a large number of nets correspond to the high-quality replications; clusters with average number of vertices covering an average number of nets correspond to the mid-quality replications; and, clusters with large number of vertices covering a small number of nets correspond to the low-quality replications. As described in Section 2.3, the coarse-grained Dulmage-Mendelsohn decomposition states that C_H and R_V are the unique smallest sets that maximize the $|C_H| - |R_H|$ and $|R_V| - |C_V|$ differences and $|R_S| = |C_S|$. We know that every boundary adjacency hypergraph $\mathcal{H}_k$ can be represented as a bipartite graph $\mathcal{G}_k$. Hence, we can use fine-grained Dulmage-Mendelsohn decomposition to encapsulate the replication characteristics of the original hypergraph into its coarsened representation, where components in R_H correspond to high-quality replications, components in R_S correspond to mid-quality replications, and components in R_V correspond to low-quality replications.

In Section 2.3, it is shown that the coarse-grained and fine-grained Dulmage-Mendelsohn decomposition runs in $O(|\mathcal{V}|(|\mathcal{V}| + |\mathcal{E}|))$ time in total. In case of $\mathcal{G}_k = (\mathcal{V}_k = \mathcal{R}_k \cup \mathcal{C}_k, \mathcal{E}_k)$ bipartite graph representation of the boundary adjacency hypergraph, this value is equal to $O(|\mathcal{V}_k|(|\mathcal{V}_k| + |\mathcal{E}_k|)$. And from the relation between $\mathcal{R}_k$, $\mathcal{C}_k$ and $\mathcal{V}_k$, $\mathcal{N}_k$, it becomes $O((|\mathcal{V}_k| + |\mathcal{N}_k|)((|\mathcal{V}_k| + |\mathcal{N}_k|) + \sum_{n_j \in \mathcal{N}_k} |Pins(n_j)|))$.



(a) Boundary adjacency hypergraph $\mathcal{H}_1^{con}$ in Fig. 3.2b.

(b) The fine-grained Dulmage-Mendelsohn decomposition of $\mathcal{H}_1^{con}$.

(c) Coarsened $\mathcal{H}_1^{con}$.

Figure 3.4: The fine-grained Dulmage-Mendelsohn decomposition of sample boundary adjacency hypergraph $\mathcal{H}_1^{con}$.

Fig. 3.4a demonstrates a simplified drawing of the boundary adjacency hypergraph $\mathcal{H}_1^{con}$ given in Fig. 3.2b. Fig. 3.4b demonstrates the coarse-grained and fine-grained Dulmage-Mendelsohn decomposition of $\mathcal{H}_1^{con}$. In Fig. 3.4b, since parts V_2 and V_3 are disjoint, it is possible to apply the Dulmage-Mendelsohn decomposition separately to parts V_2 and V_3. Components in Fig. 3.4b constitute the new vertices and nets in Fig. 3.4c. For instance, the 3^{rd} component composed of vertices v_{14}, v_{15} and nets n_7^3, n_8^3 in Fig. 3.4b constitute the vertex v_3 and net n_3 in the coarsened hypergraph in Fig. 3.4c.

3.4 Balance Preserving Replication Capacity Computation

Maximum replication capacity κ_k represents the amount of replication allowed into part V_k. Note that the maximum replication capacity κ_k of each part V_k directly affects the contribution of R_k to the partition imbalance. That is, even a single miscalculated κ_k might result in a significant change in the imbalance of the whole partition. Hence, maximum replication capacity of each part must be chosen in such a way that, after the replication, imbalance of the partition is preserved and the replication capacity is consumed to reduce the cutsize as much as possible. For this purpose, we set κ_k to $(1+\rho)W_{avg} - w(V_k)$ for each part V_k. That is, we aim to raise the weight of part V_k, i.e., $w(V_k)$, to the average weight of a part after all available replication capacity is consumed, i.e., $(1+\rho)W_{avg}$. Since replication introduces new vertices to the parts, this scheme will just increase the weight of the parts that are smaller than $(1+\rho)W_{avg}$. Hence, partition imbalance changes as follows.

- If $(1+\rho)W_{avg} < W_{max}$, after the replication, W_{avg} is expected to increase, while W_{max} stays the same. Hence, balance will stay the same even in the worst case, that is, no replication; otherwise, balance will be improved.
- Otherwise, we have enough room to raise the total weight of each part to $(1+\rho)W_{avg}$. That is, even if the replication doesn't consume all available

capacity and increase the imbalance, we can reduce the final imbalance to its initial value by making dummy vertex replications without considering any net coverages.

At kth iteration of the algorithm, we try to raise $w(V_k)$ to $(1+\rho)W_{avg}$, which corresponds to the part weight of an optimally balanced partition. Hence, after the replication, a significant reduction in the partition imbalance ratio is highly expected. This observation unsurprisingly holds with the experimental results as well.

3.5 Part Visit Order for Replication

In our approach, parts are visited in some particular order and replication set R_k of a part V_k directly affects the boundary adjacency graph $\mathcal{H}_\ell$ for $\ell > k$. Hence, ordering of the parts plays an important role considering the global quality of the proposed scheme. This effect can be observed both in the cutsize and imbalance reduction. For instance, processing parts in increasing weight order might result in poor imbalance reductions. That is, most of the replication capacity might be consumed by larger parts and $W_{max} - W_{avg}$ difference could not be reduced as expected. Moreover, one would intuitively select parts whose average boundary adjacency hypergraph net degree is smaller compared to others. That is, consider a net n_j connected to m vertices in part V_k and n vertices in part V_ℓ. If $m < n$, part ordering should be done in such a way that V_ℓ should be processed first to cover net n_j with the least possible number of replications, i.e., m vertices. In the experimentation results, comparison of evaluated ordering schemes are given.

Chapter 4

Experimental Results

In this chapter, experimental results evaluated for various data set collections are given. First, in Section 4.1, experimented data set collections are detailed. Next, implementation details are given in Section 4.2. In Section 4.3, we present the results regarding the initial partitions of the data sets. Then, in Section 4.4, the replication results for cutsize and imbalance reductions are given. Next, in Sections 4.5 and 4.6, we discuss the effect of coarsening and part ordering schemes over replication results. Finally, we present system resource usage statistics of an implementation of the proposed model in Section 4.7.

4.1 Data Set Collection

There are various hypergraph models successfully incorporated into spatial database [26, 25] and information retrieval [15] systems. For experimentation purposes, we used sample hypergraphs from these domains and investigated the effect of replication in these hypergraph models.

To investigate the effect of replication in spatial databases, a wide range of real-life road network (RN) data sets are collected from US Tiger/Line [11] (Minnesota7 including 7 counties Anoka, Carver, Dakota, Hennepin, Ramsey, Scott,

Washington; San Francisco; Oregon; New Mexico; Washington), US Department of Transportation [52] (California Highway Planning Network), and Brinkhoff's network data generator [10] (Oldenburg; San Joaquin). Hypergraph models of RN data sets are constructed according to the clustering hypergraph model presented in [26].

To examine the effect of replication in information retrieval systems, text crawls are downloaded from the Stanford WebBase project[1, 21] (CalGovernor, Facebook, Wikipedia) and University of Florida Sparse Matrix Collection [24] (Stanford). Stanford data set represents links in a set of crawled web pages. In hypergraph models of Stanford WebBase project data sets, terms correspond to vertices and documents correspond to nets. This construction scheme is detailed in [15].

Properties of the hypergraphs extracted from the collected data sets are presented in Table 4.1. Column explanations of the hypergraph properties table are as follows.

Column	Explanation
$\|Pins\|$	Total number of pins in $\mathcal{N}$, i.e., $\|Pins\| = \sum_{n_j \in \mathcal{N}} \|Pins(n_j)\|$.
$d_{avg}^{\mathcal{N}}$	Average net degree.
c_{avg}	Average net cost.
$d_{avg}^{\mathcal{V}}$	Average vertex degree.
w_{avg}	Average vertex weight.

In Table 4.1, hypergraphs are grouped by their domains (RN and IR) and sorted in increasing $|Pins|$ order. As seen in the table, RN hypergraphs have relatively small average net degrees. This may give us the intuition that in RN data sets, covering a net is likely to be easier compared to IR data sets. In IR hypergraphs, large $|Pins|$ and $d_{avg}^{\mathcal{N}}$ values show that the boundary adjacency hypergraphs are expected to be quite large in size, hence, it is expected that coarsening will play an important role in these hypergraphs.

Type	$\mathcal{H}$	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{N}\rvert$	$\lvert Pins\rvert$	$d_{avg}^{\mathcal{N}}$	c_{avg}	$d_{avg}^{\mathcal{V}}$	w_{avg}
RN	Oldenburg	5389	13003	32945	2.5	8.4	6.1	46.9
	California	14185	33414	94857	2.8	6.7	6.7	53.3
	SanJoaquin	22987	44944	131603	2.9	8.3	5.7	52.5
	Minnesota	46103	78371	239422	3.1	13.1	5.2	53.5
	SanFrancisco	213371	319305	967917	3.0	9.1	4.5	51.5
	Wyoming	317100	512754	1443433	2.8	8.9	4.6	49.0
	NewMexico	556115	781219	2270120	2.9	8.9	4.1	49.5
	Oregon	601672	811166	2332870	2.9	9.5	3.9	48.3
	Washington	652063	824650	2427615	2.9	11.7	3.7	49.0
IR	Stanford	281903	281903	2312497	8.2	1.0	8.2	8.2
	CalGovernor	92279	30805	3004908	97.5	1.0	32.6	1.0
	Facebook	4618974	66568	14277456	214.5	1.0	3.1	1.0
	Wikipedia	1350762	70115	43285851	617.4	1.0	32.0	1.0

Table 4.1: Data set properties.

Replication capacity is calculated by $\rho|V|w_{avg}$ and a high capacity will intuitively result in more replications covering more nets. Hence, low values of $|V|w_{avg}$ is expected to produce relatively poor results in replication. For instance, Oldenburg and CalGovernor is highly expected to fall in this area. However, this case is not likely to be applicable for others.

4.2 Implementation Details

Conducted replication experiments are evaluated on a Debian GNU/Linux 5.0.4 (x86_64) system running on an AMD Opteron (2.1 GHz) processor. During tests, ANSI C sources are compiled using `gcc` bundled with 4.3 release of GNU Compiler Collections where `-O3 -pipe -fomit-frame-pointer` flags are turned on. IBM ILOG CPLEX 12.1 is used in single-threaded mode to solve ILP problems. ILP pass for each boundary adjacency hypergraph is time limited to 200 milliseconds. PaToH [20] v3.1 is used with default parameters for initial partitioning of the data sets. Coarsening is disabled for boundary adjacency hypergraphs where total number of pins are smaller than or equal to 30.

4.3 Initial K-Way Hypergraph Partitioning

In the BPMCRS problem, it is assumed that an initial partition of the supplied hypergraph is provided. For this purpose, we partitioned the hypergraphs for two different K values – 128 and 256 – via PaToH. In Table 4.2, partition properties of the test hypergraphs are given. In this table, columns correspond to the particular properties of partitions as follows.

Column	Explanation
$\chi(\Pi)$	Connectivity cutsize.
$ibr(\Pi)$	Imbalance ratio.
$\lvert\mathcal{N}^*\rvert$	# of cut nets.
$d_{avg}^{\mathcal{N}^*}$	Average cut net degree.
c_{avg}^{*}	Average cut net cost.
λ_{avg}^{*}	Average cut net connectivity.
$\lvert\mathcal{V}^*\rvert$	# of boundary vertices.
$d_{avg}^{\mathcal{V}^*}$	Average boundary vertex degree.
w_{avg}^{*}	Average boundary vertex weight.

For RN data sets, where $d_{avg}^{\mathcal{N}^*}$ is approximately the same, the correlation between $|Pins|$ in Table 4.1 and $\chi(\Pi)$ in Table 4.2 points out that the connectivity cutsize increases proportional to the total number of pins. However, for IR data sets, varying $d_{avg}^{\mathcal{N}^*}$ values also affect the $\chi(\Pi)$ and we observe that high $d_{avg}^{\mathcal{N}}$ values generally imply high $\chi(\Pi)$ values.

4.4 Replication Results

In Table 4.3, replication results are listed for hypergraph partitions given in Table 4.2. Column explanations of the Table 4.3 are as follows.

Type	K	$\mathcal{H}$	$\chi(\Pi)$	$ibr(\Pi)$	$\lvert\mathcal{N}^*\rvert$	$d_{avg}^{\mathcal{N}^*}$	c_{avg}^*	$\lvert\mathcal{V}^*\rvert$	$d_{avg}^{\mathcal{V}^*}$	w_{avg}^*
RN	128	Oldenburg	15377	4.3	1993	2.8	7.5	1498	7.1	50.3
		California	21404	5.3	3661	3.2	5.7	3016	7.3	56.1
		SanJoaquin	27939	25.4	3709	3.2	7.4	3136	7.2	57.7
		Minnesota	40108	97.4	3719	3.3	10.7	3408	6.8	59.1
		SanFrancisco	44986	5.6	6255	3.3	7.2	6194	6.2	56.3
		Wyoming	46421	5.0	6527	3.0	7.1	6599	5.6	51.0
		NewMexico	44386	4.2	6505	3.1	6.8	6896	5.4	51.9
		Oregon	51154	5.0	7079	3.0	7.2	7463	5.3	51.3
		Washington	58721	12.0	6621	3.1	8.8	7059	5.4	53.1
	256	Oldenburg	24148	5.3	3068	2.8	7.6	2224	7.1	50.2
		California	34254	5.5	5535	3.3	5.9	4554	7.2	56.1
		SanJoaquin	44961	13.4	5777	3.2	7.6	4871	7.1	57.8
		Minnesota	66581	177.1	6088	3.3	10.8	5595	6.8	59.0
		SanFrancisco	72541	3.9	9972	3.3	7.2	10021	6.2	56.8
		Wyoming	69708	32.3	9829	3.0	7.1	9935	5.6	51.5
		NewMexico	73516	4.3	10489	3.1	7.0	11181	5.4	52.6
		Oregon	77227	4.2	10678	3.1	7.2	11386	5.4	52.1
		Washington	91527	5.4	10132	3.2	9.0	10994	5.4	53.8
IR	128	Stanford	15904	228.2	9181	116.2	1.0	167826	11.0	11.0
		CalGovernor	201391	5.7	24476	119.6	1.0	92275	32.6	1.0
		Facebook	324393	1.4	58467	234.7	1.0	4611479	3.1	1.0
		Wikipedia	1040098	4.2	69117	623.5	1.0	1350568	32.0	1.0
	256	Stanford	24408	777.3	12523	93.2	1.0	173321	10.9	10.9
		CalGovernor	298223	5.1	27724	107.4	1.0	92278	32.6	1.0
		Facebook	415405	1.2	61934	225.7	1.0	4617453	3.1	1.0
		Wikipedia	1470241	4.9	69608	620.5	1.0	1350736	32.0	1.0

Table 4.2: Properties of hypergraph partitions.

Column	Explanation
$\chi(\%)$	Connectivity cutsize reduction, i.e., $\chi(\%) = (1 - \chi(\Pi^r)/\chi(\Pi)) \times 100$.
$ibr(\%)$	Imbalance ratio reduction, i.e., $ibr(\%) = (1 - ibr(\Pi^r)/ibr(\Pi)) \times 100$.
$\lvert Pins(\mathcal{H}_k)\rvert$	Average of total # of pins of each $\mathcal{H}_k$, i.e., $\lvert Pins(\mathcal{H}_k)\rvert = (\sum_{k=1}^{K} \sum_{n_j \in \mathcal{N}_k} \lvert Pins(n_j)\rvert)/K$.
$\lvert Pins(\%)\rvert$	Reduction in pin count after coarsening, i.e., $\lvert Pins(\%)\rvert = (1 - \lvert Pins(\mathcal{H}_k^{coarse})\rvert/\lvert Pins(\mathcal{H}_k)\rvert) \times 100$.

As seen in Table 4.3, since $d_{avg}^{\mathcal{N}^*}$ values are approximately the same for RN data sets, in a majority of the tests $|\mathcal{V}|$ variable dominates the effect on the quality of the replication. That is, compared to other RN hypergraphs, low $|\mathcal{V}|$ values of Oldenburg hypergraph resulted in low quality replications due to the low replication capacity of $\rho|V|w_avg$. On the other hand, for RN hypergraphs with high $|\mathcal{V}|$ values – i.e., Wyoming, NewMexico, Oregon, and Washington – replication removed almost every net from the cut. For 128-way RN hypergraph partitions, a replication amount of 1%, provides 51.8% reduction in the connectivity cutsize and 16.1% reduction in the imbalance ratio on the average. Same amount of replication provides 56.7% reduction in the connectivity cutsize and 16.2% reduction in the imbalance ratio for 256-way hypergraph partitions. By looking at these improvements, it can be concluded that RN hypergraphs are quite suitable for replication.

For IR data sets, since $d_{avg}^{\mathcal{N}^*}$ values of the IR hypergraph partitions are much larger than those of the RN hypergraphs and high replication percentages are common practice in IR systems, replication is evaluated with higher values – 10% and 20% – of ρ. Compared to RN hypergraph partitions, both $|\mathcal{V}|$ and $d_{avg}^{\mathcal{N}^*}$ values are quite varying among IR hypergraph partitions and both have a more prominent effect on the quality of the replication. For instance, the effect of high $|\mathcal{V}|$ and low $d_{avg}^{\mathcal{N}^*}$ values of Facebook is distinctive in the replication results. To conclude, replication can yield promising results depending on the structure of the hypergraph, which can be estimated by simple observations in $|\mathcal{V}|$ and $d_{avg}^{\mathcal{N}^*}$

Type	ρ	K	$\mathcal{H}$	$\chi(\%)$	$ibr(\%)$	$\lvert Pins(\mathcal{H}_k)\rvert$	$\lvert Pins(\%)\rvert$
RN	0.01	128	Oldenburg	9.7	24.0	16.1	13.0
			California	15.4	19.8	54.8	70.6
			SanJoaquin	18.3	4.9	50.8	70.5
			Minnesota	37.3	2.0	73.1	72.9
			SanFrancisco	73.8	18.6	94.2	48.2
			Wyoming	99.0	20.8	84.3	30.5
			NewMexico	99.7	24.7	80.7	26.5
			Oregon	100.0	20.7	89.7	24.0
			Washington	100.0	9.3	86.5	14.3
		256	Oldenburg	6.1	19.6	12.1	0.0
			California	9.6	18.9	22.7	31.3
			SanJoaquin	13.7	8.4	36.4	62.4
			Minnesota	22.4	1.5	51.5	71.7
			SanFrancisco	45.4	26.2	88.3	72.7
			Wyoming	71.5	4.1	67.2	49.1
			NewMexico	98.5	23.9	71.7	41.4
			Oregon	99.9	24.5	70.7	39.5
			Washington	99.2	19.2	70.0	34.6
IR	0.10	128	Stanford	49.6	13.1	2101.1	368.0
			CalGovernor	3.8	100.0	7989.2	93.5
			Facebook	44.6	100.0	649449.1	98.7
			Wikipedia	11.4	100.0	3159094.2	98.6
		256	Stanford	69.0	24.0	3348.7	422.7
			CalGovernor	1.7	100.0	2082.7	90.9
			Facebook	45.6	100.0	415554.9	98.3
			Wikipedia	7.6	100.0	557740.3	97.8
	0.20	128	Stanford	39.0	10.3	1111.0	182.5
			CalGovernor	9.7	100.0	36296.2	96.2
			Facebook	57.8	100.0	612529.1	98.7
			Wikipedia	18.2	100.0	5732360.4	99.0
		256	Stanford	52.9	18.8	1638.6	267.6
			CalGovernor	5.1	100.0	6596.4	92.8
			Facebook	59.2	100.0	383360.9	98.4
			Wikipedia	15.7	100.0	2217957.5	98.0

Table 4.3: Replication results.

values.

In Table 4.3, $ibr(\%)$ column gives the reduction in the imbalance ratio of the partition in percentages. For RN data sets, small $\rho|V|w_{avg}$ doesn't provide enough replication capacity to improve the balance. Average partition imbalance reduction is around 16% for RN hypergraphs. Since IR data sets provide considerable amounts of replication capacity, average partition imbalance reduction is around 100% for IR data sets. To summarize, replication provides significant imbalance reductions in a majority of the conducted experiments.

In Table 4.3, the last two columns provide information about the average size of the constructed boundary adjacency hypergraphs and the effect of the coarsening on these hypergraphs. For RN data sets, coarsening reduced the size of the constructed boundary adjacency hypergraphs by 41.2%-44.7%, on the average, for 128-way and 256-way partitions, respectively. For IR data sets, coarsening reduced the size of the boundary adjacency hypergraphs by 96.3%-97.18% on the average. These results imply that coarsening is quite effective in the contraction of the boundary adjacency hypergraphs and provide significants reductions in the size of input supplied to the ILP solver.

4.5 Comparison of Coarsening Algorithms and The Effect of Coarsening

The Dulmage-Mendelsohn decomposition provides quite promising coarsening results. However, it does not take vertex weights and net costs into account. Hence, it is possible that other coarsening algorithms can prove to be more effective than the Dulmage-Mendelsohn decomposition by taking vertex weights and net costs into account. To investigate this issue, we adopted 17 different state-of-the-art coarsening algorithms (HCM, PHCM, MANDIS, AVEDIS, CANBERRA, ABS, GCM, SHCM, HCC, HPC, ABSHCC, ABSHPC, CONC, GCC, SHCC, NC, MNC) that are implemented in PaToH to obtain coarsened boundary adjacency hypergraphs. We supplied these coarsened boundary adjacency hypergraphs to

the ILP solver and observed their effects on the cutsize reduction.

In our experiments, we evaluated all of the adopted coarsening algorithms over all data sets for different K and ρ settings. We observed that each of the adopted coarsening algorithms show high fluctuations in the quality of the coarsened boundary adjacency hypergraphs. Quality measure in this context is the effectiveness of the ILP phase running on the coarsened hypergraphs. On the other hand, the Dulmage-Mendelsohn decomposition showed a stable performance and in a significant majority (87.6 %) of the experiments performed in the top three.

Coarsening provides a lossy compression of the boundary adjacency hypergraph. To further investigate the effectiveness of the coarsening and determine the information loss due to the coarsening, experiments are evaluated with two different setups. In the first setup S_1, experiments are evaluated with the coarsening and time limitation constraints. In the second setup S_2, ILP phase is performed without coarsening and time limitations. In S_2, since there is no limitation on the execution time of the ILP solver, ILP phase dominated the majority of the total runtime in tests. For IR data sets with dense boundary adjacency hypergraphs – e.g., Facebook, Wikipedia – total replication phase took hours to complete. In case of RN data sets, where boundary adjacency hypergraphs are relatively sparse, ILP phase completed in the same amount of time. In S_2, ILP phase produced slightly better results in terms of the quality of the replication. In S_1, the reduction in the quality due to the loss of information in coarsening varies between 2.3% and 7.8% compared to S_2. To sum up, in a majority of the experiments, S_1 produces on par results with S_2.

4.6 Part Visit Orderings

In the conducted experiments, three different part visit ordering schemes are evaluated for hypergraphs given in Table 4.1. In the first scheme O_1, parts are ordered by increasing average net degree of their boundary adjacency hypergraph values. In O_2, parts are sorted in increasing weight order. In the last scheme O_3,

parts are chosen randomly. On the average, O_1 and O_2 performs around 5.7-10.3% better results compared to O_3 in terms of reduction in the connectivity cutsize. O_1 performs slightly (2.2%) better cutsize reductions compared to O_2. To conclude, since ILP phase coupled with coarsening performs quite effective in terms of consuming replication capacity with the maximum possible number of net coverages, part ordering generally causes relatively minor variations in the replication quality. In conducted experiments, results are given according to O_1 scheme.

4.7 System Resource Consumptions

At kth iteration of the replication algorithm, we construct the boundary adjacency hypergraph $\mathcal{H}_k$, coarsen $\mathcal{H}_k$ to $\mathcal{H}_k^{coarse}$, select vertices that are to be replicated from $\mathcal{H}_k^{coarse}$ via ILP. For RN data sets, where boundary adjacency hypergraphs are generally small in size, ILP phase is generally dominated the total runtime of the replication and replication is finished 3-8 times faster than the partitioning time of PaToH. For IR data sets, where boundary adjacency hypergraphs are large in size, 60.8% of the total runtime is consumed by the coarsening, and ILP and $\mathcal{H}_k$ construction took 28.2% and 11.1% of the total runtime, respectively. In the experimented IR data sets, replication of large boundary adjacency hypergraphs performed at most 3.5 times slower compared to the partitioning time of PaToH.

Chapter 5

Conclusion

Motivated by the problem of finding a replication set for a given K-way hypergraph partition and a maximum replication capacity ratio, we proposed a part-oriented approach based on a unique blend of an ILP formulation and a coarsening algorithm using the Dulmage-Mendelsohn decomposition. Experiments show that proposed model provides promising results both in terms of the quality of the replication set and the runtime performance. The Dulmage-Mendelsohn decomposition-based coarsening scheme is found to be quite successive for encapsulating the replication characteristics of a hypergraph into its coarsened representation. In the light of conducted experiments, the Dulmage-Mendelsohn decomposition-based coarsening coupled with the ILP formulation provide effective results for covering nets in a boundary adjacency hypergraph.

Appendix A

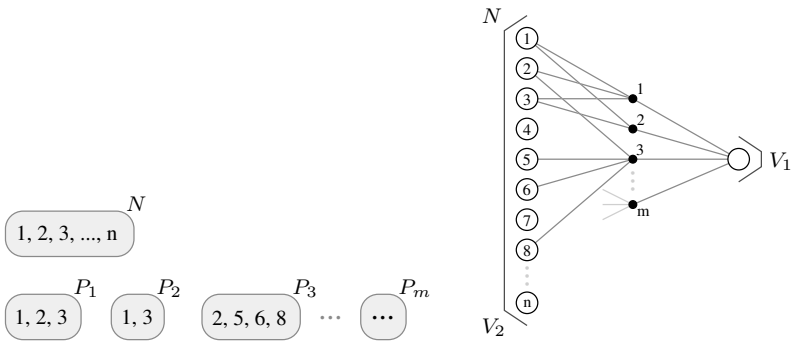
SUK to MCRS Transformation

In this chapter, we present a simple transformation of SUK (Set-Union Knapsack) problem [42] to an MCRS (Min-Cut Replication Set) problem, which is a generalization of Problem 1 without balancing constraints. SUK problem is defined [42] as follows.

Definition 4 Set-Union Knapsack Problem. *Given a set of n items $N = \{1, 2, \ldots, n\}$ and a set of m so-called elements $P = \{1, 2, \ldots, m\}$, each item j corresponds to a subset P_j of the element set P. The items j have nonnegative profits p_j, $j = 1, 2, \ldots, n$, and the elements i have nonnegative weights w_i, $i = 1, 2, \ldots, m$. The total weight of a set of items is given by the total weight of the elements of the union of the corresponding element sets. Find a subset of the items with total weight not exceeding the knapsack capacity while maximizing the profit.*

SUK is known [42] to be an $\mathcal{NP}$-hard problem. A simple transformation of SUK problem to MCRS problem can be given as follows.

Theorem 1 *Every set-union knapsack (SUK) problem can be represented in terms of a min-cut replication set (MCRS) problem.*



(a) Sample SUK instance.

(b) SUK to MCRS transformation.

Figure A.1: Sample SUK problem to MCRS problem transformation.

Proof. One can transform a SUK problem to an MCRS for 2-way hypergraph partition $\Pi = \{V_1, V_2\}$ problem, where elements of SUK problem correspond to the boundary vertices of V_1 and element sets correspond to the cut nets. Consider a special MCRS problem where cut nets are connected to a single vertex in V_2 whose weight is exceeding the given replication capacity. Thus only replication of vertices in V_1 into V_2 is possible. A solution to this particular MCRS problem would provide a solution to the SUK problem. □

In Fig. A.1, a sample SUK to MCRS transformation is shown. In Fig. A.1a, item set N and element set P are composed of n items and m elements, respectively. Each item j in N is associated with an element set P_j, which is a subset of P. Objective is to maximize the profit of the covered items, where there is an upper bound on the total weight of the used elements. This SUK instance is mapped to a MCRS problem in Fig. A.1b, where orientation of the replication is forced towards a single direction. That is, the single vertex in part V_1 weights much more than the given replication capacity, forcing replication direction from V_1 to V_2. In addition, items and elements correspond to nets and vertices in Fig. A.1b, respectively. That is, $P_1 = \{1, 2, 3\}$ in Fig. A.1a is represented by net n_1 connecting vertices v_1, v_2, and v_3 in Fig. A.1b.

Appendix B

Finding the Cutsize of a Partition With Vertex Replications

Previous studies involving K-way hypergraph partitioning with vertex replications doesn't investigate the effect of the cutsize metric on the conducted experiments. However, computation of the minimum cutsize for a given partition with vertex replications can stand as a major problem. For instance, the list of cut nets is sufficient to compute the cutsize for cut-net metric (Eq. 2.1). However, pin mapping of the nets (i.e., which partition should be used for a particular pin of a cut net) are necessary for the computation of the cutsize for connectivity metric (Eq. 2.2). Hence, depending on the used cutsize metric, finding the minimum cutsize for a given partition with vertex replications is a significant problem. (Without vertex replications, since every vertex has a unique copy and, hence, every net has a unique pin mapping, this decision problem does not arise.) This issue is generalized in Problem 2.

Problem 2 Finding the Cutsize of a Partition With Vertex Replication. *Given a partition with vertex replication* Π^r *and a cutsize metric* $\chi(\cdot)$*, find the minimum* $\chi(\Pi^r)$.

Considering the connectivity metric, even for a single net, finding the pin mapping with the least possible number of parts is a set cover optimization problem (i.e., pins correspond to the element universe, and parts correspond to the element sets), which is known to be $\mathcal{NP}$-hard. On the other hand, it should be noted that a majority of the pins of a cut net tends to be fixed (i.e., not replicated and has a unique copy in some particular part) and after connecting the floating pins (i.e., pins that can be connected to different copies in different parts) of a cut net to these fixed parts, there remains an insignificant number of pins that needs to be determined for connection. Hence, problem turns out to be relatively cheaply computable in practice. But for a vast number of cut nets, this still can stand as an intractable problem.

For cut-net metric, since Eq. 2.1 just depends on the determination of cut nets, cutsize can be computed in linear time proportional to the size of the total number of pins.

Bibliography

[1] The stanford webbase project. `http://diglib.stanford.edu:8091/~testbed/doc2/WebBase`, Aug 2010.

[2] C. J. Alpert, J.-H. Huang, and A. B. Kahng. Multilevel circuit partitioning. In *DAC '97: Proceedings of the 34th annual Design Automation Conference*, pages 530–533, New York, NY, USA, 1997. ACM.

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

[4] C. Aykanat, A. Pinar, and U. V. Çatalyürek. Permuting sparse rectangular matrices into block-diagonal form. *SIAM J. Sci. Comput.*, 25(6):1860–1879, 2004.

[5] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, May 1999.

[6] L. A. Barroso, J. Dean, and U. Hölzle. Web search for a planet: The google cluster architecture. *IEEE Micro*, 23:22–28, 2003.

[7] P. A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, 1987.

[8] R. H. Bisseling, S. Cerav-erbas, M. Lorenz, R. Pendavingh, C. Reeves, M. Rger, and A. Verhoeven. Partitioning a call graph. In *in: Second International Workshop on Combinatorial Scientific Computing*, 2005.

[9] L. A. Bjork. Recovery scenario for a db/dc system. In *ACM '73: Proceedings of the ACM annual conference*, pages 142–146, New York, NY, USA, 1973. ACM.

[10] T. Brinkhoff. A framework for generating network-based moving objects. *Geoinformatica*, 6(2):153–180, 2002.

[11] U. C. Bureau. Topologically integrated geographic encoding and referencing system (*TIGER*). `http://www.census.gov/geo/www/tiger/`, 2002.

[12] F. Cacheda, V. Carneiro, V. Plachouras, and I. Ounis. Network analysis for distributed information retrieval architectures. *Advances in Information Retrieval*, pages 527–529, 2005.

[13] F. Cacheda, V. Carneiro, V. Plachouras, and I. Ounis. Performance comparison of clustered and replicated information retrieval systems. In *ECIR'07: Proceedings of the 29th European conference on IR research*, pages 124–135, Berlin, Heidelberg, 2007. Springer-Verlag.

[14] F. Cacheda, V. Plachouras, and I. Ounis. A case study of distributed information retrieval architectures to index one terabyte of text. *Inf. Process. Manage.*, 41(5):1141–1161, 2005.

[15] B. B. Cambazoglu and C. Aykanat. A term-based inverted index organization for communication-efficient parallel query processing. Tokio, Japan, October 2006.

[16] B. B. Cambazoglu and C. Aykanat. Hypergraph-partitioning-based remapping models for image-space-parallel direct volume rendering of unstructured grids. *IEEE Trans. Parallel Distrib. Syst.*, 18(1):3–16, 2007.

[17] U. Catalyurek and C. Aykanat. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. *IEEE Trans. Parallel Distrib. Syst.*, 10(7):673–693, 1999.

[18] U. Catalyurek and C. Aykanat. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. *IEEE Trans. Parallel Distrib. Syst.*, 10(7):673–693, 1999.

[19] U. Catalyurek and C. Aykanat. A hypergraph-partitioning approach for coarse-grain decomposition. In *Supercomputing '01: Proceedings of the 2001 ACM/IEEE conference on Supercomputing (CDROM)*, pages 28–28, New York, NY, USA, 2001. ACM.

[20] U. V. Catalyurek and C. Aykanat. *PaToH*: partitioning tool for hypergraphs. *Technical Report*, 1999.

[21] J. Cho, H. Garcia-Molina, T. Haveliwala, W. Lam, A. Paepcke, S. Raghavan, and G. Wesley. Stanford webbase components and applications. *ACM Trans. Internet Technol.*, 6(2):153–186, 2006.

[22] S. B. Davidson, H. Garcia-Molina, and D. Skeen. Consistency in partitioned networks. *ACM Comput. Surv.*, 17(3):341–370, 1985.

[23] C. T. Davies, Jr. Recovery semantics for a db/dc system. In *ACM '73: Proceedings of the ACM annual conference*, pages 136–141, New York, NY, USA, 1973. ACM.

[24] T. A. Davis. University of florida sparse matrix collection. *NA Digest*, 92, 1994.

[25] E. Demir and C. Aykanat. Efficient successor retrieval operations for aggregate query processing on clustered road networks. *Inf. Sci.*, 180(14):2743–2762, 2010.

[26] E. Demir, C. Aykanat, and B. Barla Cambazoglu. Clustering spatial networks for aggregate query processing: A hypergraph approach. *Inf. Syst.*, 33(1):1–17, 2008.

[27] N. J. Dingle, P. G. Harrison, and W. J. Knottenbelt. Uniformization and hypergraph partitioning for the distributed computation of response time densities in very large markov models. *J. Parallel Distrib. Comput.*, 64(8):908–920, 2004.

[28] A. L. Dulmage and i. Mendelsohn. Two algorithms for bipartite graphs. *J. Soc. Ind. Appl. Math.*, 1I:183–194, 1963.

[29] A. L. Dulmage and N. S. Mendelsohn. Coverings of bipartite graphs. *Can. J. Math.*, 10:517–534, 1958.

[30] A. L. Dulmage and N. S. Mendelsohn. A structure theory of bipartite graphs of finite exterior dimension. *Trans. Roy. Soc. Can. Sec.*, III(53):1–13, 1959.

[31] M. Enos, S. Hauck, and M. Sarrafzadeh. Replication for logic bipartitioning. In *ICCAD '97: Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design*, pages 342–349, Washington, DC, USA, 1997. IEEE Computer Society.

[32] V. Gaede and O. Günther. Multidimensional access methods. *ACM Comput. Surv.*, 30(2):170–231, 1998.

[33] J. Gray, P. Helland, P. O'Neil, and D. Shasha. The dangers of replication and a solution. In *SIGMOD '96: Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, pages 173–182, New York, NY, USA, 1996. ACM.

[34] E. Greengrass. *Information Retrieval: A Survey.* Nov 2000.

[35] X. Gu and R. T. Pascoe. Selective data replication for distributed geographical data sets. In *GIS '08: Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, pages 1–4, New York, NY, USA, 2008. ACM.

[36] D. Hawking. Scalable text retrieval for large digital libraries. In *ECDL '97: Proceedings of the First European Conference on Research and Advanced Technology for Digital Libraries*, pages 127–145, London, UK, 1997. Springer-Verlag.

[37] J. Hwang and A. El Gamal. Optimal replication for min-cut partitioning. In *ICCAD '92: Proceedings of the 1992 IEEE/ACM international conference on Computer-aided design*, pages 432–435, Los Alamitos, CA, USA, 1992. IEEE Computer Society Press.

[38] L. J. Hwang. *Replication in partitioned networks.* PhD thesis, 1994.

[39] D. M. Johnson, A. L. Dulmage, and N. Mendelsohn. Connectivity and reducibility of graphs. *Can. J. Math.*, 14:529–539, 1962.

[40] J. B. R. Jr. and N. Goodman. A survey of research and development in distributed database management. In *VLDB '1977: Proceedings of the third international conference on Very large data bases*, pages 48–62. VLDB Endowment, 1977.

[41] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: application in vlsi domain. In *DAC '97: Proceedings of the 34th annual Design Automation Conference*, pages 526–529, New York, NY, USA, 1997. ACM.

[42] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.

[43] M. Koyutürk and C. Aykanat. Iterative-improvement-based declustering heuristics for multi-disk databases. *Inf. Syst.*, 30(1):47–70, 2005.

[44] C. Kring and A. Newton. A cell-replicating approach to minicut-based circuit partitioning. In *Computer-Aided Design, 1991. ICCAD-91. Digest of Technical Papers., 1991 IEEE International Conference on*, pages 2–5, 11-14 1991.

[45] R. Kužnar, F. Brglez, and B. Zajc. Multi-way netlist partitioning into heterogeneous fpgas and minimization of total device cost and interconnect. In *DAC '94: Proceedings of the 31st annual Design Automation Conference*, pages 238–243, New York, NY, USA, 1994. ACM.

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

[47] D.-R. Liu and M.-Y. Wu. A hypergraph based approach to declustering problems. *Distrib. Parallel Databases*, 10(3):269–288, 2001.

[48] Z. Lu and K. S. McKinley. Partial collection replication versus caching for information retrieval systems. In *SIGIR '00: Proceedings of the 23rd*

annual international ACM SIGIR conference on Research and development in information retrieval, pages 248–255, New York, NY, USA, 2000. ACM.

[49] A. Moffat, W. Webber, J. Zobel, and R. Baeza-Yates. A pipelined architecture for distributed text query evaluation. *Inf. Retr.*, 10(3):205–231, 2007.

[50] A. Moffat, J. Zobel, and D. Hawking. Recommended reading for ir research students. *SIGIR Forum*, 39(2):3–14, 2005.

[51] R. Murgai, R. K. Brayton, and A. Sangiovanni-Vincentelli. On clustering for minimum delay/area. pages 6–9, nov 1991.

[52] U. D. of Transportation. Federal highway administration, the national highway planning network. `http://www.fhwa.dot.gov/planning/nhpn/`, 2004.

[53] M. T. Özsu and P. Valduriez. Distributed database systems: Where are we now? *Computer*, 24(8):68–78, 1991.

[54] A. Pothen. *Sparse null bases and marriage theorems*. PhD thesis, Ithaca, NY, USA, 1984.

[55] A. Pothen and C.-J. Fan. Computing the block triangular form of a sparse matrix. *ACM Trans. Math. Softw.*, 16(4):303–324, 1990.

[56] R. L. Russo, P. H. Oden, and P. K. Wolff. A heuristic procedure for the partitioning and mapping of computer logic graphs. *IEEE Trans. Comput.*, 20(12):1455–1462, 1971.

[57] A. Saman Tosun. Analysis and comparison of replicated declustering schemes. *IEEE Trans. Parallel Distrib. Syst.*, 18(11):1578–1591, 2007.

[58] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Aug 1989.

[59] A. Selivonenko. *Data partitioning and replication management in distributed geographic information system (GIS) database*. PhD thesis, Florida International University, 2005.

[60] R. O. Selvitopi. Replicated hypergraph partitioning. Master's thesis, Bilkent University, Ankara, Turkey, 2010.

[61] R. M. Shapiro and R. E. Millstein. Reliability and fault recovery in distributed processing. In *OCEANS'77, Conference Record, October 17–19, 1977, Los Angeles*, volume II, pages 31D.1–31D.5, 1977.

[62] S. Shekhar, C.-T. Lu, S. Chawla, and S. Ravada. Efficient join-index-based spatial-join processing: A clustering approach. *IEEE Trans. on Knowl. and Data Eng.*, 14(6):1400–1421, 2002.

[63] S. Shekhar, S. Ravada, D. Chubb, and G. Turner. Declustering and load-balancing methods for parallelizing geographic information systems. *Knowledge and Data Engineering, IEEE Transactions on*, 10(4):632 –655, jul/aug 1998.

[64] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[65] B. Uçar and C. Aykanat. Revisiting hypergraph models for sparse matrix partitioning. *SIAM Rev.*, 49(4):595–603, 2007.

[66] B. Vastenhouw and R. H. Bisseling. A two-dimensional data distribution method for parallel sparse matrix-vector multiplication. *SIAM Rev.*, 47(1):67–95, 2005.

[67] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes*. Morgan Kaufmann Publishing, San Francisco, May 1999.

[68] P. Yalagandula. *A scalable information management middleware for large distributed systems*. PhD thesis, Austin, TX, USA, 2005. Adviser-Dahlin, Michael D.

[69] H. H. Yang and D. F. Wong. New algorithms for min-cut replication in partitioned circuits. In *ICCAD '95: Proceedings of the 1995 IEEE/ACM international conference on Computer-aided design*, pages 216–222, Washington, DC, USA, 1995. IEEE Computer Society.

[70] A. N. Yzelman and R. H. Bisseling. Cache-oblivious sparse matrix-vector multiplication by using sparse matrix partitioning methods. *SIAM J. Sci. Comput.*, 31(4):3128–3154, 2009.