INFORMS Journal on Computing
Articles in Advance, pp. 1–18
ISSN 1091-9856 (print) | ISSN 1526-5528 (online)

http://dx.doi.org/10.1287/ijoc.2013.0567
© 2013 INFORMS

Constrained Min-Cut Replication for *K*-Way Hypergraph Partitioning

Volkan Yazici
Department of Computer Science, Özyegin University, Istanbul, Turkey, volkan.yazici@ozyegin.edu.tr

Cevdet Aykanat
Department of Computer Engineering, Bilkent University, Ankara, Turkey, aykanat@cs.bilkent.edu.tr

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 *constrained min-cut replication* (*CMCR*) 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 CMCR problem is finding the optimal vertex replication sets for each part of the given partition such that the initial cut size of the partition is minimized, where the initial imbalance is either preserved or reduced under the given replication capacity constraint. In this study, we present a complexity analysis of the CMCR problem and propose a model based on a unique blend of coarsening and integer linear programming (ILP) schemes. This coarsening algorithm is derived from a novel utilization of the Dulmage-Mendelsohn decomposition. Experiments show that the ILP formulation coupled with the Dulmage-Mendelsohn decomposition-based coarsening provides high quality results in practical execution times for reducing the cut size of a given *K*-way hypergraph partition.

Key words: combinatorial optimization; graphs; heuristics; optimization; programming: integer
History: Accepted by Karen Aardal, Area Editor for Design and Analysis of Algorithms; received October 2011; revised August 2012, May 2013; accepted June 2013. Published online in *Articles in Advance*.

1. Introduction

In the literature, hypergraph models have found numerous applications in a broad range of fields, including parallel scientific computing, VLSI design, software engineering, wireless communication networks, information retrieval, and database systems. In proposed models, the combinatorial optimization problem at hand is generally expressed as a hypergraph partitioning problem, trying to optimize a particular objective function (e.g., maximizing the spectrum utilization in a wireless network, minimizing the total disk access cost in a database system) subject to certain constraints (e.g., number of channels allowed in a certain wireless spectrum, block size of a disk page). As implied by the established model, the quality of the produced solution given for the initial combinatorial optimization problem directly relates to the intermediate hypergraph partition. Hence, efficient and effective hypergraph partitioning algorithms play a significant role in hypergraph-based combinatorial optimization models.

Combinatorial models based on hypergraph partitioning can broadly be categorized into two groups. In the former group, *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 and geographic information systems, wireless communication networks, information retrieval, and software engineering can be categorized in this group. In the latter group, *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 VLSI design can be categorized in this group. In this study, we focus on the undirectional hypergraph partitioning models. Directional hypergraph 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 enhancing 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. That is, many existing real-world problems addressed by undirectional hypergraph models allow the coexistence of multiple copies of a vertex. We believe this availability provides room for improvement, which can be exposed by vertex replications on the modeled hypergraph. For instance, in a hypergraph model (Demir et al. 2008), optimizing the disk access costs, where vertices represent junction records and hyperedges represent the access patterns of the aggregate network operations, multiple copies of a record can coexist on a file system to decrease the number of page misses. Likewise, in an information retrieval system modeled by a hypergraph (Cambazoglu and Aykanat 2006), optimizing the overall query throughput, where vertices represent terms and hyperedges represent documents/pages, instances of a term might be placed on multiple servers to decrease the processing times of queries.

We refer to using vertex replication to improve the quality of partitioning objective in undirectional hypergraphs as *hypergraph partitioning with vertex replication* and there are two viable approaches to this problem. In the first approach, called *one-phase*, replication is performed concurrently with the partitioning. In the second approach, called *two-phase*, replication is performed in two separate phases. In the first phase, the hypergraph is partitioned and in the second phase, replication is applied to the partition produced in the previous phase. The one-phase approach has the potential of producing high quality solutions since it considers partitioning and replication simultaneously. However, the two-phase approach is more general and flexible since it enables the use of any one of the state-of-the-art hypergraph partitioning tools in the first phase. The two-phase approach also has the additional flexibility of working on a given partition that already contains replicated vertices.

Our main contribution consists of providing a detailed complexity analysis of the two-phase hypergraph partitioning with vertex replication problem and proposing an efficient and effective replication phase based on a unique blend of coarsening and integer linear programming (ILP) schemes. This coarsening scheme is based on a novel utilization of 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. The replication set of each part is constrained by a maximum replication capacity, and these sets are constrained to be determined in such a way that the partition imbalance is either improved or preserved after the replication. To the best of our knowledge, this is the first study considering the vertex replication in undirectional hypergraph models. To present a baseline, we discuss related studies from directional hypergraph models next.

In VLSI design, the first in-depth discussion about logic replication is given by Russo et al. (1971), where they propose a heuristic approach. Later, Kring and Newton (1991) and Murgai et al. (1991) extend the Fidduccia and Mattheyses (FM) iterative improvement algorithm to allow vertices to be duplicated during partitioning. Hwang and El Gamal (1992) propose 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. Kužnar et al. (1994) introduce the concept of functional replication. Yang and Wong (1995) provide an optimal solution to the min-area min-cut replication problem. Alpert and Kahng (1995) present a survey about circuit partitioning and provide a brief list of existing logic replication schemes. Enos et al. (1997) provide enhancements for available gate replication heuristics. All of these works on VLSI applications fall into the category of vertex replication on directional hypergraphs. Very recently, Selvitopi et al. (2012) propose and discuss a method for replicated partitioning of undirected hypergraphs relying on one-phase approach.

This paper is organized as follows: In §2, preliminary definitions are presented. In §3, a detailed complexity analysis of the problem at hand is given. Then, in §4, the proposed model to the CMCR problem is presented. Later, in §5, experimental setup and results of the conducted experiments are given. Finally, in §6, we conclude the paper.

2. Preliminaries

In this section, notations and definitions that are used throughout the paper are given. In §2.1, we start by defining the K-way hypergraph partitioning problem. Later in §2.2, the partitioning with vertex replication problem is presented. Finally in §2.3, the Dulmage-Mendelsohn decomposition is presented.

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 in $\mathcal{V}$. The vertices connected by a net n are called its *pins* and denoted as $Pins(n) \subseteq \mathcal{V}$. The set of nets connecting a vertex v is denoted as $Nets(v) = \{n \in \mathcal{N} \mid v \in Pins(n)\}$. 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, $Adj(U) = \bigcup_{u \in U} Adj(u) - U$ and $w(U) = \sum_{u \in U} w(u)$.

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., $\lambda(n) = 1$). A vertex is said to be a *boundary* vertex if it is connected by at least one cut net. The cut and uncut nets are also referred to as *external* and *internal* nets, respectively. $N_{\text{ext}}(V_k)$ denotes the set of external nets of part V_k, that is, $N_{\text{ext}}(V_k) = \{n \in \mathcal{N} \mid \lambda(n) > 1, Pins(n) \cap V_k \neq \varnothing\}$. $\mathcal{N}_{\text{ext}}$ is used to refer to all external nets in a partition, i.e., $\mathcal{N}_{\text{ext}} = \{n \in \mathcal{N} \mid \lambda(n) > 1\}$.

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

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

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

There are various *cut-size* metrics for representing the cost $\chi(\Pi)$ of a partition Π. Two most widely used cut-size metrics are given as follows.

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

$$\chi_{\text{cut}}(\Pi) = \sum_{n \in \mathcal{N}_{\text{ext}}} c(n) \tag{1}$$

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

$$\chi_{\text{con}}(\Pi) = \sum_{n \in \mathcal{N}_{\text{ext}}} (\lambda(n) - 1)c(n) \tag{2}$$

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

In Lengauer (1990), it is shown that K-way hypergraph partitioning is $\mathcal{NP}$-hard.

Figure 1 shows a three-way partition of a sample hypergraph $\mathcal{H}$ with 24 boundary vertices and 19 cut nets. Note that 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. In Figure 1, $Pins(n_{19}) = \{v_{20}, v_{21}, v_{23}\}$, $\Lambda(n_{19}) = \{V_2, V_3\}$, $\lambda(n_{19}) = 2$, and so on.

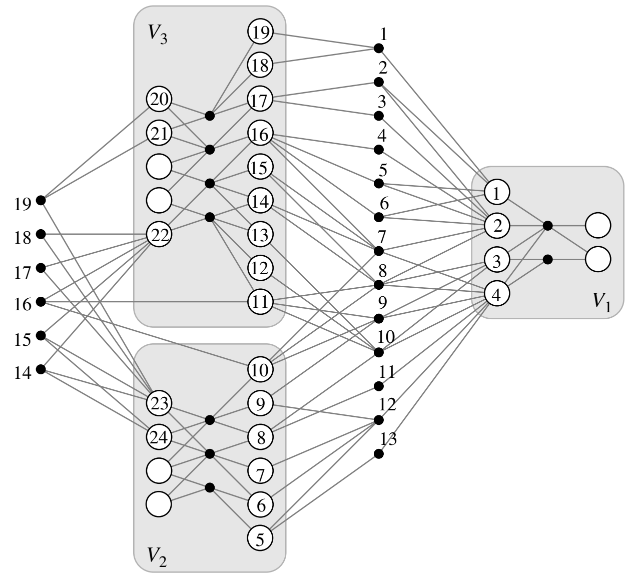


Figure 1 **A Three-Way Partition of a Sample Hypergraph $\mathcal{H}$**

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 $R_k \cap V_k = \varnothing$, 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, since R_k's of $\mathcal{R}(\Pi)$ are overlapping, V_k^r's of Π^r are overlapping as well. 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 $W_{\text{tot}}^r = W_{\text{tot}} + \sum_{k=1}^{K} w(R_k)$.

Given these definitions, the main problem addressed in this paper is defined as follows.

Problem 1 (Constrained Min-Cut Replication (CMCR) for a Given 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 cut size $\chi(\Pi^r)$ of the induced replicated partition Π^r subject to the*

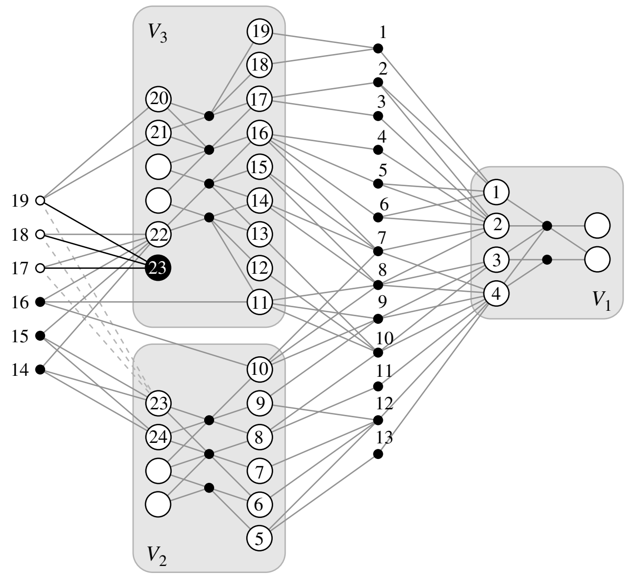


Figure 2 **Replication of v_{23} to V_3 in the Sample Hypergraph $\mathcal{H}$ Given in Figure 1**

replication capacity constraint of $W^r_{\mathrm{tot}} \le (1+\rho)W_{\mathrm{tot}}$ *and the balancing constraint of* $ibr(\Pi^r) \le ibr(\Pi)$.

In Figure 2, replication of vertex v_{23} from part V_2 to part V_3 is depicted. After replication, nets n_{17}, n_{18}, n_{19} connecting v_{23} in V_2, are replaced with new connections to replicated v_{23} in V_3. Consequently, nets n_{17}, n_{18}, n_{19} are removed from the cut.

2.3. The Dulmage-Mendelsohn Decomposition

The Dulmage-Mendelsohn decomposition is a canonical decomposition on bipartite graphs and described in a series of papers (Dulmage and Mendelsohn 1963, 1958, 1959; Johnson et al. 1962). Later, Pothen and Fan (1990) formalize this decomposition by a series of lemmas and present further 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 $\mathcal{C}$ with the following definitions based on $\mathcal{M}$:

$$R_V = \{v_i \in R \mid v_i \textit{ is reachable by an alternating path from some unmatched vertex } v_j \in R\},$$

$$R_H = \{v_i \in R \mid v_i \textit{ is reachable by an alternating path from some unmatched vertex } v_j \in C\},$$

$$R_S = R - (R_V \cup R_H),$$

$$C_V = \{v_i \in C \mid v_i \textit{ is reachable by an alternating path from some unmatched vertex } v_j \in R\},$$

$$C_H = \{v_i \in C \mid v_i \textit{ is reachable by an alternating path from some unmatched vertex } v_j \in C\},$$

$$C_S = C - (C_V \cup C_H).$$

The following properties given in Pothen (1984), Pothen and Fan (1990) regarding the R_H, R_S, R_V and C_H, C_S, R_S subsets provide certain features related to 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 the 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. Sets 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 $\mathcal{M}$; 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 (1990) further decompose 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 (1990) will be referred to as *fine-grained decomposition*.

G_X denotes the subgraph of $\mathcal{G}$ induced by the vertex subset X, where X stands for either H, S, or V. 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 $\mathcal{E}$ that connects vertices in parts R_X and C_X. The fine-grained decomposition is formalized as follows.

Definition 3 (Fine-Grained Dulmage-Mendelsohn Decomposition). Let $\mathcal{M}$ be a maximum matching for a bipartite graph $\mathcal{G} = (\mathcal{V} = \mathcal{R} \cup \mathcal{C}, \mathcal{E})$ 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 performed as follows.

- Find connected components in subgraphs G_H and G_V.
- Using G_S, construct a new directed bipartite graph G'_S, where matched edges 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, the resulting fine-grained decomposition is expected to provide many more parts 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 the 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 (Tarjan 1972). Hence, the decomposition phase takes $O(|\mathcal{V}|(|\mathcal{V}| + |\mathcal{E}|))$ time in total.

In Figure 3, 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.

Figure 3(b) 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}, $R_H = \{v_3, v_4\}$ and $C_H = \{v_{11}, v_{12}, v_{13}, v_{14}, v_{15}\}$.

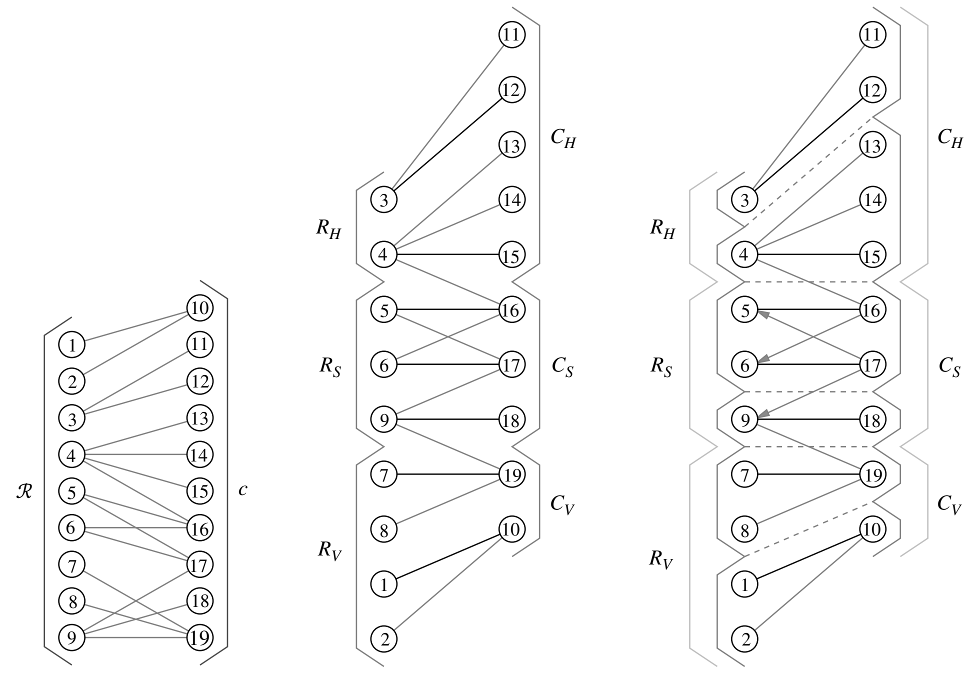
Figure 3(c) depicts a fine-grained decomposition of the sample bipartite graph $\mathcal{G}$ in Figure 3(a). Here, components are separated with dashed lines. For instance, vertices v_3, v_{11}, v_{12} and edges between them constitute a connected component in G_H. As seen in Figure 3(c), unmatched edges (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 appear two strongly-connected components in G'_S composed of vertices v_5, v_6, v_{16}, v_{17}, and v_9, v_{18}.

3. Complexity Analysis

In this section, we investigate the complexity of the CMCR problem and subproblems implicitly induced by the vertex replication. In §3.1, we first prove that CMCR is $\mathcal{NP}$-hard, regardless of vertex weights and net costs. Then in §3.2 we show that finding the cut size of a K-way hypergraph partition with vertex replication might turn out to be an $\mathcal{NP}$-hard problem depending on the used cut-size metric.

3.1. Complexity of Constrained Min-Cut Replication

The first in-depth analysis of CMCR for directional hypergraph models is given in Hwang (1994), where a polynomial-time reduction from the PARTITION



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

Figure 3 The Dulmage-Mendelsohn Decomposition

problem (Garey and Johnson 1990) to the CMCR problem is presented. Moreover, they conjectured that the CMCR remains $\mathcal{NP}$-hard when all vertices and nets have unit weights and costs, respectively. Alternative to the given proof in Hwang (1994), we present a simpler proof based on polynomial-time Turing reduction from the set-union knapsack problem to the CMCR problem. Reduction is performed regardless of the vertex weights and net costs, hence, it provides a solution to the conjecture presented in Hwang (1994) as well.

Before going into the details of the proof, we first present the definition of the set-union knapsack (SUK) problem (Goldschmidt et al. 1994) as follows.

Definition 4 (Set-Union Knapsack (SUK) Problem). Given a set of n items $T = \{1, 2, \ldots, n\}$ and a set of m so-called elements $L = \{1, 2, \ldots, m\}$, each item $j \in T$ corresponds to a subset L_j of the element set L. The items $j \in T$ have nonnegative profits p_j, $j = 1, 2, \ldots, n$, and the elements $i \in L$ 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.

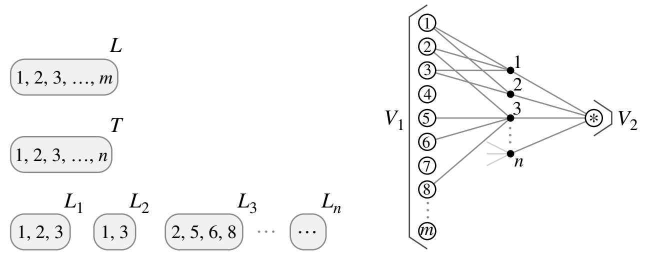
The polynomial-time Turing reduction from SUK to CMCR is presented in the following.

Theorem 1. *Every SUK instance S is polynomial-time Turing reducible to a CMCR instance C, that is, $S \leq_T C$.*

Proof. A SUK can be reduced to a CMCR with a two-way hypergraph partition $\Pi = \{V_1, V_2\}$, where elements of the SUK instance correspond to the vertices of V_1 (i.e., $V_1 = \{v_1, v_2, \ldots, v_m\}$) and items correspond to the cut nets (i.e., $N_{\text{ext}}(V_2) = \{n_1, n_2, \ldots, n_n\}$). Since Π is a two-way hypergraph partition, $N_{\text{ext}}(V_1) = N_{\text{ext}}(V_2)$. We create set $V_2 = \{v_*\}$ such that the weight of v_* exceeds the given replication capacity and it is connected by the external nets of part V_1, i.e., $v_* \in Pins(n)$ for $\forall\, n \in N_{\text{ext}}(V_1)$. Since $w(v_*) > \rho W_{\text{tot}}$, only the replication of vertices in V_1 to V_2 is allowed. A solution to this CMCR instance selects a subset R of vertices in V_1 maximizing the cost of the nets in $N \subseteq N_{\text{ext}}(V_1)$ such that $Pins(n) = R \cup \{v_*\}$ for $\forall\, n \in N$ and subject to $w(R) \leq \rho W_{\text{tot}}$. Hence, a solution to this particular CMCR problem provides a solution to the given SUK problem and transformation from SUK to CMCR is performed in polynomial time. □

One should note that, because of the presented two-way partition in the given proof, cut net and connectivity cut-size metrics yield identical results. In other words, for $\lambda(n) = 2$,

$$\chi_{\text{con}}(\Pi) = \sum_{n \in \mathcal{N}_{\text{ext}}} (\lambda(n) - 1)c(n) = \sum_{n \in \mathcal{N}_{\text{ext}}} c(n) = \chi_{\text{cut}}(\Pi).$$



(a) Sample SUK instance

(b) SUK to CMCR transformation

Figure 4 **Sample SUK to CMCR Transformation**

Goldschmidt et al. (1994) prove that the SUK problem is $\mathcal{NP}$-hard. Given that SUK is polynomial-time Turing reducible to CMCR, the complexity of CMCR can be stated as follows.

Corollary 1. *The constrained min-cut replication problem is $\mathcal{NP}$-hard.*

In Figure 4, an example for this Turing reduction is shown. In Figure 4(a), item set T and element set L are composed of n items and m elements, respectively. Each item j in T is associated with an element set L_j, which is a subset of L. The objective is to maximize the profit of the covered items subject to the upper bound on the total weight of the union of the used elements. This SUK instance is mapped to a CMCR instance in Figure 4(b). To enforce the replication direction from V_1 to V_2, v_* is added to V_2 such that v_* weighs much more than the given replication capacity. The relation between items and element subsets are respectively represented by vertices and nets in Figure 4(b). That is, $L_1 = \{1, 2, 3\}$ in Figure 4(a) is represented by net n_1 connecting vertices v_1, v_2, and v_3 in Figure 4(b).

3.2. Cut Size of a Partition with Vertex Replication

Previous studies involving K-way hypergraph partition with vertex replication doesn't investigate the effect of the cut-size metric on the complexity of the problem. However, computation of the minimum cut size for a given partition with vertex replication can stand as a major problem depending on the cut-size metric used. For instance, whereas the list of cut nets is sufficient to compute the cut size for the cut-net metric (Equation (1)), pin mapping of the nets (i.e., which part should be used for a particular pin of a cut net) are also needed for the computation of the cut size for the connectivity metric (Equation (2)). Hence, depending on the used cut-size metric, finding the minimum cut size for a given partition with vertex replication can be an intractable problem. Without vertex replication, 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 (Cut Size of a Partition with Vertex Replication). *Given a partition with vertex replication Π^r and a cut-size metric $\chi(\cdot)$, find the minimum $\chi(\Pi^r)$.*

In terms of connectivity cut-size metric, even for a single net, finding the pin mapping with the least possible number of parts is a set-cover problem, where pins correspond to the element universe and parts correspond to the element sets. This problem is known (Garey and Johnson 1990) to be $\mathcal{NP}$-hard. This facet of the hypergraph partition with vertex replication is stated in the following corollary.

Corollary 2. *Finding the minimum connectivity cut size of a K-way hypergraph partition with vertex replication is $\mathcal{NP}$-hard.*

It should also be noted that a majority of the pins of a cut net tends to be fixed, that is, a majority of the vertices are anticipated to not be replicated and have a single copy in some particular part. After connecting such fixed pins to the relevant parts, it is expected that there remains a negligible number of pins that needs to be considered for a suitable part association. Hence, the problem might turn out to be relatively tractable in practice. On the other hand, in case of an excessive amount of cut nets, this association decision can still stand as an intractable problem.

For the cut-net metric, since Equation (1) just depends on the determination of cut nets, cuts ize can be computed in linear time proportional to the size of the total number of pins.

4. Constrained Min-Cut Replication

In this section, we propose an efficient and effective approach for solving the CMCR 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 cut size via replication. Thus, only the boundary vertices are considered for finding a good replication set $\mathcal{R}$. To handle the balancing constraints on the weights of the parts of the replicated partition, we propose a part-oriented approach by investigating the replication 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 cut size without violating the maximum weight constraint of part V_k. It is also clear that replication of vertices R_k into part V_k can only decrease the cut size because of 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 $N_{\text{ext}}(V_k)$ and the vertex set $Adj(V_k)$ for finding an R_k.

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:     $\kappa_k = (1+\rho)W_{\text{avg}} - w(V_k)$
4:     $\mathcal{H}_k \leftarrow$ construct$(\mathcal{H}, k, \Pi_{k-1}^r)$
5:     $\mathcal{H}_k^{\text{coarse}} \leftarrow$ coarsen$(\mathcal{H}_k)$
6:     $R_k \leftarrow$ select$(\mathcal{H}_k^{\text{coarse}}, \kappa_k)$
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: $\Pi_r \leftarrow \Pi_K^r$.
```

Algorithm 1 displays the general framework of our approach. For each part V_k, we first compute the replication capacity κ_k such that the initial imbalance will be preserved or improved after the replication. Details of this replication capacity computation are deferred to §4.4. Then, we construct the hypergraph $\mathcal{H}_k$ that is used to determine the set of vertices R_k to be replicated into part V_k and 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 $N_{\text{ext}}(V_k)$. This hypergraph construction process is described in §4.1. 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 §4.2. To reduce the high computation cost of ILP for large $\mathcal{H}_k$, a coarsening scheme for $\mathcal{H}_k$ is described in §4.3.

4.1. Boundary Adjacency Hypergraph Construction

Without loss of generality, we describe here the boundary adjacency hypergraph construction 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 replication 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 $N_{\text{ext}}(V_k)$ 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_ℓ in the original partition Π_0^r. During an earlier iteration $k < \ell$, if all pins of net n_j that lie in part V_ℓ are replicated into part V_k, then net n_j disappears in $N_{\text{ext}}(V_\ell)$. In such a case, those pins of net n_j that lie in part V_k and are only connected by net n_j to part V_ℓ disappear from $Adj(V_\ell)$. This update procedure is given in Algorithm 2.

Algorithm 2 (Update(k))

```
1: for $l \leftarrow (k+1)$ to $K$ do
2:  for each net $n_j \in N_{\text{ext}}(V_k)$ do
3:     if $Pins(n_j) \cap R_k \cap V_\ell \neq \varnothing$ then
4:         for each vertex $v \in (Pins(n_j) \cap V_k)$ do
5:             if $Nets(v) \cap N_{\text{ext}}(V_\ell) = \{n_j\}$ then
```

```
6:          $Adj(V_\ell) = Adj(V_\ell) - \{v\}$
7:      $N_{\text{ext}}(V_\ell) = N_{\text{ext}}(V_\ell) - \{n_j\}$
8:      $\Lambda(n_j) = \Lambda(n_j) - V_\ell$
            {optional for cut-net metric}.
```

Two distinct boundary adjacency hypergraphs are required to encapsulate the cut-net (Equation (1)) and connectivity (Equation (2)) cut-size metrics, which will be referred to as $\mathcal{H}_k^{\text{cut}} = (\mathcal{V}_k^{\text{cut}}, \mathcal{N}_k^{\text{cut}})$ and $\mathcal{H}_k^{\text{con}} = (\mathcal{V}_k^{\text{con}}, \mathcal{N}_k^{\text{con}})$, respectively. The construction procedures of $\mathcal{H}_k^{\text{cut}}$ and $\mathcal{H}_k^{\text{con}}$ are depicted in Algorithms 3 and 4, respectively. In these hypergraphs, the vertex set is composed of $Adj(V_k)$, and 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 in $\mathcal{H}_k$ connected by vertices in R_k is maximized without violating the balance constraint imposed on V_k. The net set definition for $\mathcal{H}_k^{\text{cut}}$ and $\mathcal{H}_k^{\text{con}}$ should be done according to this maximization objective.

Algorithm 3 (Construct $(\mathcal{H}, k, \Pi_{k-1}^r)$ for Cut-Net Metric)

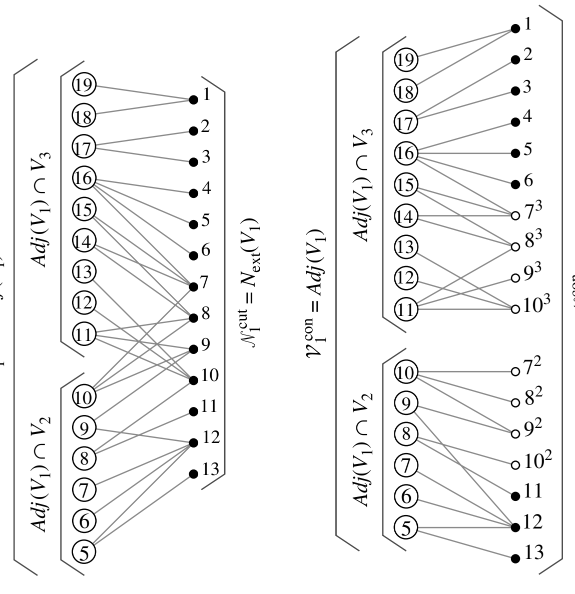
```
1: $\mathcal{V}_k^{\text{cut}} \leftarrow Adj(V_k)$
2: $\mathcal{N}_k^{\text{cut}} \leftarrow N_{\text{ext}}(V_k)$
3: for each net $n_j \in N_{\text{ext}}(V_k)$ do
4:     $Pins(n_j) \leftarrow Pins(n_j) - V_k$
5: return $\mathcal{H}_k^{\text{cut}} \leftarrow (\mathcal{V}_k^{\text{cut}}, \mathcal{N}_k^{\text{cut}})$.
```

For the cut-net metric, to reduce the cut-size contribution of a net n_j in $N_{\text{ext}}(V_k)$, the net n_j should be made internal to part V_k, which is possible only when all pins of net n_j in $Adj(V_k)$ are replicated into V_k. (Consequently, the contribution of n_j to $\chi_{\text{cut}}(\Pi^r)$ will vanish.) Thus, the net set of $\mathcal{H}_k^{\text{cut}}$ is selected as the external nets of part V_k (line 2 of Algorithm 3). Because $\mathcal{H}_k^{\text{cut}}$ is used to find the set of vertices to be replicated into part V_k, the boundary vertices of part V_k are built from the pin list of the nets in $\mathcal{H}_k^{\text{cut}}$, excluding the vertices that are already available in V_k (lines 3–4 of Algorithm 3).

Algorithm 4 (Construct $(\mathcal{H}, k, \Pi_{k-1}^r, \kappa_k)$ for Connectivity Metric)

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

For the connectivity metric, to reduce the cut-size contribution of a net n_j in $N_{\text{ext}}(V_k)$, it is sufficient to replicate a subset of the pins of net n_j such that $\lambda(n_j)$ will decrease after the replication. Consequently, the contribution of n_j to $\chi_{\text{con}}(\Pi^r)$ will accordingly diminish as well. To encapsulate this connectivity cut-size contribution of vertices in a part for an external net,


$\mathcal{V}_1^{\text{cut}} = Adj(V_1)$
$Adj(V_1) \cap V_3$
$Adj(V_1) \cap V_2$
$\mathcal{N}_1^{\text{cut}} = N_{\text{ext}}(V_1)$
$\mathcal{V}_1^{\text{con}} = Adj(V_1)$
$\mathcal{N}_1^{\text{con}}$
19 18 17 16 15 14 13 12 11 10 9 8 7 6 5
1 2 3 4 5 6 7 8 9 10 11 12 13
7^3 8^3 9^3 10^3 7^2 8^2 9^2 10^2

(a) $\mathcal{H}_1^{\text{cut}}$ of part V_1 (b) $\mathcal{H}_1^{\text{con}}$ of part V_1

Figure 5 Sample Boundary Adjacency Hypergraph Construction

we enhance $N_{\text{ext}}(V_k)$ through a *net splitting* operation such that each external net $n_j \in N_{\text{ext}}(V_k)$ is split into $\lambda(n_j) - 1$ new nets. Splitting is performed as follows: For each net n_j in $N_{\text{ext}}(V_k)$, 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). As a result of this splitting operation, replicating the vertices connected by n_j^ℓ now corresponds to removing V_ℓ from $\Lambda(n_j)$, in other words, decreasing $\lambda(n_j)$ by 1.

Figure 5 shows the boundary adjacency hypergraphs $\mathcal{H}_1^{\text{cut}}$ (Figure 5(a)) and $\mathcal{H}_1^{\text{con}}$ (Figure 5(b)) of part V_1 in Figure 1 for cut-net and connectivity metrics, respectively. Comparing Figure 1 with Figures 5(a) and 5(b) show 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 $\mathcal{H}_1^{\text{cut}}$ and $\mathcal{H}_1^{\text{con}}$.

Comparing Figure 1 with Figures 5(a) and 5(b) reveals that each of the external nets $n_1, n_2, \ldots, n_{13}$ of V_1 incurs a single net in $\mathcal{H}_1^{\text{cut}}$. Similarly, each of the external nets $n_1, n_2, \ldots, n_6$ and n_{11}, n_{12}, n_{13} of V_1 that have a connectivity of two incurs a single net in $\mathcal{H}_1^{\text{con}}$. On the other hand, each of the external nets n_7, n_8, n_9, n_{10} of V_1 that have a connectivity of $3 = \lambda(n_j)$, as a result of the net splitting operation, incurs $2 = \lambda(n_j) - 1$ nets in $\mathcal{H}_1^{\text{con}}$. For example, n_7 with $Pins(n_7) = \{v_{10}, v_{14}, v_{15}, v_{16}\}$ connects both V_2 and V_3, and it incurs two nets n_7^2 and n_7^3 in $\mathcal{H}_1^{\text{con}}$, where $Pins(n_7^2) = \{v_{10}\}$ and $Pins(n_7^3) = \{v_{14}, v_{15}, v_{16}\}$. Note that n_7^2 and n_7^3 are simply shown as 7^2 and 7^3 in Figure 5(b).

To further understand the effect of the net splitting operation, consider net n_9 in $N_{\text{ext}}(V_1)$. As seen in Figure 5(a), net n_9 of $\mathcal{H}_1^{\text{cut}}$ connects the vertices $\{v_9, v_{10}, v_{11}\}$. So, the cut-net cut size imposed by net n_9 can only be reduced if the vertices $\{v_9, v_{10}, v_{11}\}$ are replicated into part V_1. On the other hand, as seen in Figure 1, the connectivity cut size imposed by n_9 can only be reduced if the vertices $\{v_9, v_{10}\}$ from V_2 and/or $\{v_{11}\}$ from V_3 are replicated into part V_1. That is, the replication of vertices $\{v_9, v_{10}\}$ or $\{v_{11}\}$ into part V_1 reduces $\lambda(n_9)$ by 1, and the replication of three vertices together into part V_1 reduces $\lambda(n_9)$ by 2. To encapsulate this connectivity cut-size contribution of vertices in parts V_2 and V_3, n_9 is split into two nets as n_9^2 and n_9^3 in $\mathcal{H}_1^{\text{con}}$ (see Figure 5(b)). Similarly, net splitting is performed for nets n_7, n_8, n_{10} as well.

In the first iteration of Algorithm 1, since there are no replicated vertices, each net splitting is unique in $\mathcal{H}_1^{\text{con}}$. However, in the following iterations (i.e., $k > 1$), net splittings are not necessarily unique for the further $\mathcal{H}_k^{\text{con}}$ constructions because of the replicated vertices. That is, multiple copies of a vertex induce multiple *pin selection* options for a net. And each different pin selection induces a different net splitting in the boundary adjacency hypergraph. Figure 6 shows this pin selection problem that occurs in the construction of $\mathcal{H}_k^{\text{con}}$, where vertex v_4 is replicated into part V_m in the mth iteration for $m < k$. Figures 6(b) and 6(c) show two possible pin selections for net n_1 that connects v_4. For the first mapping in Figure 6(b), replication of v_4 and v_5 appear to be necessary to remove V_ℓ from $\Lambda(n_1)$, whereas, for the second mapping in Figure 6(c), replication of v_5 is sufficient for the same purpose. As depicted in Figure 6, pin selections of nets directly affect the number of vertices to be replicated for removing a part from the connectivity list of a particular net. A closer look at the problem would lead to the fact that, as a direct implication of Corollary 2, pin selection is a set cover problem, which is $\mathcal{NP}$-hard. In our model, for a net n_j and a vertex $v_i \in Pins(n_j)$, if there exists a copy of v_i 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 selects v_i from part V_k^r; otherwise, it selects v_i from part V_ℓ that is provided by the initial partition. For instance, in Figure 6, if v_4 is replicated to part V_m in a previous iteration to decrease $\lambda(n_1)$, then n_1 selects v_4 from V_m^r; otherwise, it selects v_4 from V_ℓ.

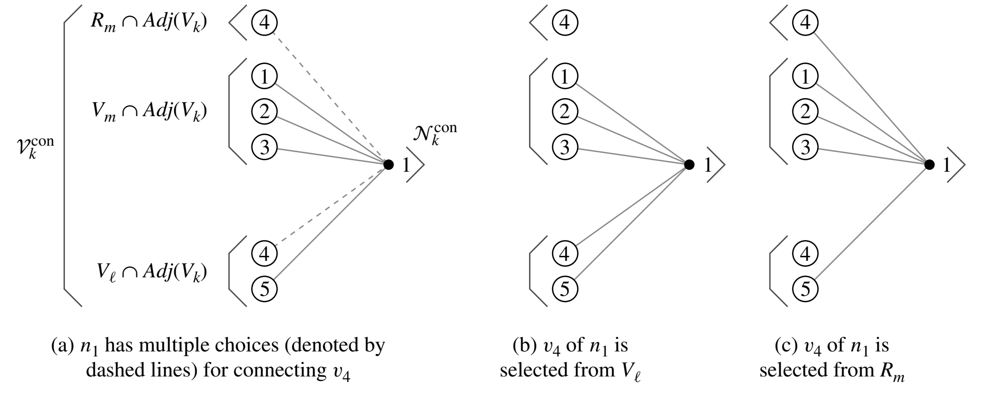
4.2. Vertex Selection in Boundary Adjacency Hypergraph

In our approach, the 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. Note that $\mathcal{H}_k$ is built in such a way that replicating the pins of a net in $\mathcal{N}_k$ has a direct effect on the partition cut size imposed by part V_k. Hence, it is clear that only the replication of vertices in $\mathcal{V}_k$ have the potential of decreasing the cut size imposed by part V_k. In this section, our objective is the selection of an optimal 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 a boundary adjacency hypergraph $\mathcal{H}_k$ and a 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 nets covered under a given capacity constraint of $w(R_k) \leq \kappa_k$. During the vertex selection procedure, a net n_j in $\mathcal{N}_k$ is said to be *covered* when all of its pins are selected for R_k, that is, $Pins(n_j) \subseteq R_k$. In §3.1, we proved that this maximization objective is an $\mathcal{NP}$-hard problem. In this section, we provide an ILP formulation for this vertex selection objective as follows:

$$\text{maximize} \sum_{n_j \in \mathcal{N}_k} c(n_j)x(n_j) \tag{3}$$

$$\text{subject to } |Pins(n_j)|x(n_j) \leq \sum_{v_i \in Pins(n_j)} y(v_i), \quad \forall n_j \in \mathcal{N}_k, \tag{4}$$

$$\sum_{v_i \in \mathcal{V}_k} w(v_i)y(v_i) \leq \kappa_k, \tag{5}$$



(a) n_1 has multiple choices (denoted by dashed lines) for connecting v_4 (b) v_4 of n_1 is selected from V_ℓ (c) v_4 of n_1 is selected from R_m

Figure 6 Sample Net Splitting Problem

where

$$x(n_j)=\begin{cases}1 & \text{if } \forall v_i \in Pins(n_j) \text{ is selected},\\ 0 & \text{otherwise},\end{cases}$$

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

Binary variable $x(n_j)$ is set to 1, if all the pins of net n_j are selected to be replicated into part V_k, that is, if n_j is covered. Likewise, if vertex v_i is selected for replication, binary variable $y(v_i)$ is set to 1. Objective (3) tries to maximize the sum of the cost of the nets covered. Inequality (4) constrains a net n_j to be set covered when all of its pins are selected, that is, $x(n_j)=1$ if $y(v_i)=1$ for $\forall v_i \in Pins(n_j)$. In expression (5), the sum of the weights of the selected vertices are constrained by κ_k. Since there are no restrictions on vertex replications, but inequality (5), this 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 doesn't have any effect on the covered nets. But once the set of $x(n_j)$'s is computed, necessary $y(v_i)$ values can be extracted from $Pins(n_j)$ without allowing any redundant vertex replications.

In the given ILP formulation, for each boundary adjacency hypergraph $\mathcal{H}_k$, there are $|\mathcal{V}_k|+|\mathcal{N}_k|$ variables for $x(n_j)$'s and $y(v_i)$'s, and $|\mathcal{N}_k|+1$ constraints (inequalities (4) and (5)), and a single maximization objective.

The ILP model formalized in expressions (3), (4), and (5) provides the optimal net selection scheme for a given boundary adjacency hypergraph $\mathcal{H}_k$ and a maximum replication capacity κ_k. In §3.1, we saw that this optimization objective corresponds to the set-union knapsack problem, which is known to be $\mathcal{NP}$-hard. Because of the intractability of the problem, from a practical point of view, this formulation is anticipated to consume a significant amount of time as the sum of the input variables ($|\mathcal{V}_k|$ and $|\mathcal{N}_k|$) grows excessively in size. To reduce this high computation cost of the ILP phase, preprocessing procedures are introduced next and applied to each constructed $\mathcal{H}_k$ before vertex selection.

1. Remove infeasible nets (that κ_k isn't sufficient to replicate all of its pins) and the vertices that are only connected by such nets.
2. Coarsen boundary adjacency hypergraphs.
3. Restrict ILP solver running time to a certain duration.

4.3. Coarsening of Boundary Adjacency Hypergraph

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

The Dulmage-Mendelsohn decomposition operates on bipartite graphs, 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 $\mathcal{E}_k$. Note that this $\mathcal{H}_k$ to $\mathcal{G}_k$ projection is performed in linear time in the order of $O(|\mathcal{V}_k|+|\mathcal{N}_k|+|Pins(\mathcal{N}_k)|)$ and is easily reversible.

Vertex selection in boundary adjacency hypergraphs is constrained by the total weight of the selected vertices for replication and its objective is to maximize the cost of the covered nets. 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 §2.3, the 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 showed that every boundary adjacency hypergraph $\mathcal{H}_k$ can be represented as a bipartite graph $\mathcal{G}_k$. Hence, we can use 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.

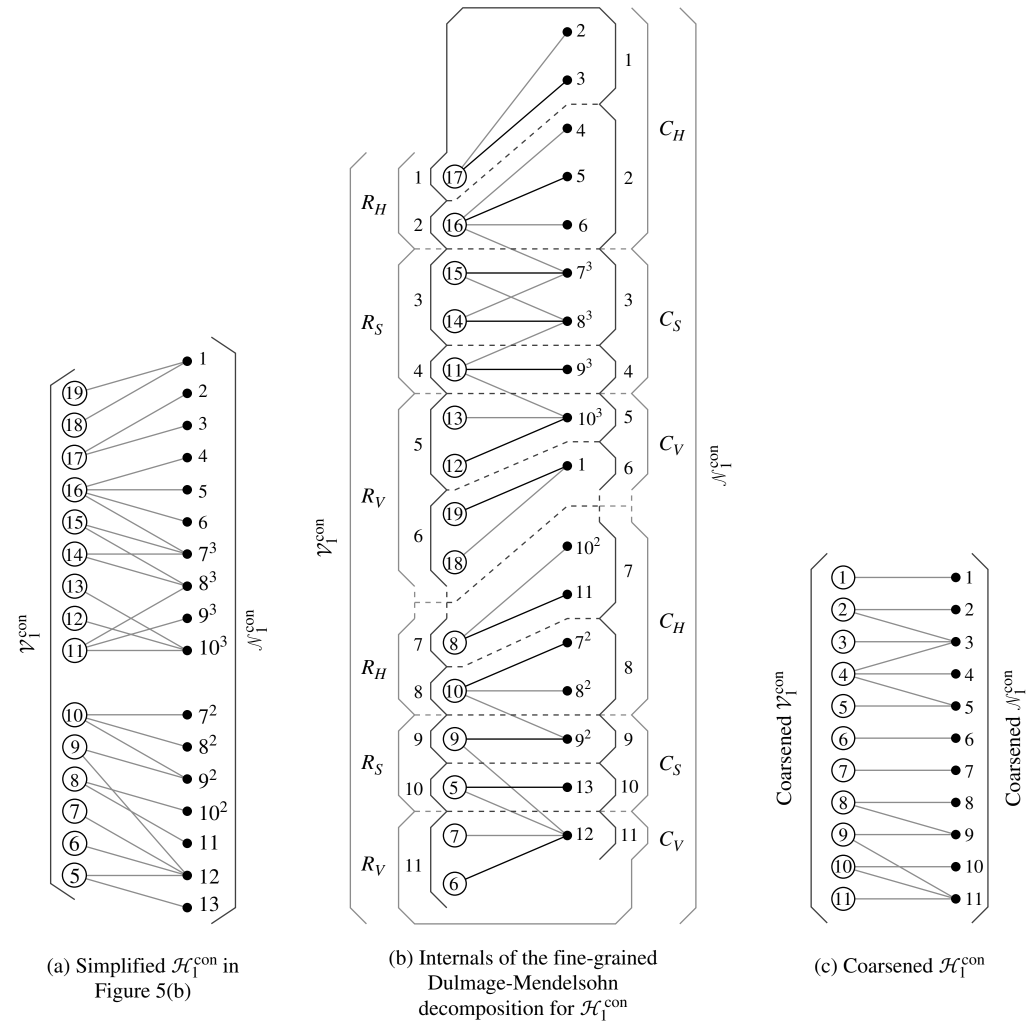
Note that Dulmage-Mendelsohn decomposition does not take vertex weights and net costs into account, hence, one might argue that it might be possible to produce better coarsening results by utilizing other clustering algorithms in the literature. This issue is investigated in the conducted experiments and results are detailed in §5.5.

In §2.3, it is shown that the coarse- and fine-grained Dulmage-Mendelsohn decompositions run in $O(|\mathcal{V}|(|\mathcal{V}|+|\mathcal{E}|))$ time in total. In the 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 bound translates 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 $O((|\mathcal{V}_k|+|\mathcal{N}_k|)(|\mathcal{V}_k|+|\mathcal{N}_k|+\sum_{n_j\in\mathcal{N}_k}|Pins(n_j)|))$. Note that it is further possible to slightly lower this bound by running the decomposition for each connected component of the input bipartite graph separately.

Figure 7(a) demonstrates a simplified drawing of the boundary adjacency hypergraph $\mathcal{H}_1^{\text{con}}$ given in Figure 5(b). Figure 7(b) demonstrates the coarse- and fine-grained Dulmage-Mendelsohn decomposition of $\mathcal{H}_1^{\text{con}}$. Note that for simplicity in Figure 7(b), because components of $\mathcal{H}_1^{\text{con}}$ related with parts V_2 and V_3 are disjoint, Dulmage-Mendelsohn decomposition is performed separately. That is, due to each net and vertex related with part V_1 and V_2, there are two R_H sets, two C_H sets, etc. Components in Figure 7(b) constitute the new vertices and nets in Figure 7(c). For instance, the 3rd component composed of vertices v_{14}, v_{15} and nets n_7^3, n_8^3 in Figure 7(b) constitutes the vertex v_3 and net n_3 in the coarsened hypergraph in Figure 7(c).

4.4. Replication Capacity

The 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, the 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 cut size as much as possible. For this purpose, we set κ_k to $(1+\rho)W_{\text{avg}}-w(V_k)$ for each part V_k. That is, we aim to increase 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


$\mathcal{V}_1^{\text{con}}$
$\mathcal{N}_1^{\text{con}}$
R_H
R_S
R_V
C_H
C_S
C_V
Coarsened $\mathcal{V}_1^{\text{con}}$
Coarsened $\mathcal{N}_1^{\text{con}}$

(a) Simplified $\mathcal{H}_1^{\text{con}}$ in Figure 5(b)

(b) Internals of the fine-grained Dulmage-Mendelsohn decomposition for $\mathcal{H}_1^{\text{con}}$

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

Figure 7 **Fine-Grained Dulmage-Mendelsohn Decomposition of $\mathcal{H}_1^{\text{con}}$**

(i.e., $(1+\rho)W_{\text{avg}}$). Because replication introduces new vertices to the parts, this scheme will just increase the weight of the parts that are smaller than $(1+\rho)W_{\text{avg}}$. Here we prove that either the partition imbalance is preserved after the replication or extra measures can be taken to satisfy the imbalance constraint.

• If $(1+\rho)W_{\text{avg}} < W_{\max}$, after the replication, W_{avg} is expected to increase, and $W_{\max}$ stays the same. Hence, balance will stay the same even in the worst case, that is, no replication; otherwise, balance will be improved.

• Otherwise, if $(1+\rho)W_{\text{avg}} \geq W_{\max}$, we can follow two approaches:

1. We have enough room to raise the total weight of each part to $(1+\rho)W_{\text{avg}}$. That is, even if the replication does not consume all available capacity and increases the imbalance, we can reduce the final imbalance to its initial value by making arbitrary vertex replications without considering any optimization objectives.

2. If arbitrary vertex replications incur extra costs (e.g., cloning gates in an integrated circuit), at each kth iteration, we can calibrate κ_k so that the imbalance $ibr(\Pi^r_k)$ of the produced partition Π^r_k will stay below the initial imbalance $ibr(\Pi)$. In addition to $(1+\rho)W_{\text{avg}} - w(V_k)$ upper bound, a lower bound can be computed as follows, which provides a range for κ_k to be picked within throughout the replication to preserve the final imbalance.

$$ibr(\Pi) \geq ibr(\Pi^r_k),$$

$$\frac{W_{\max}}{W_{\text{avg}}} - 1 \geq (1+\rho)W_{\text{avg}} \cdot \left(\frac{1}{K}\left[\sum_{\ell=1}^{k-1} w(V^r_\ell) + w(V_k) + \kappa_k + \sum_{\ell=k+1}^{K} w(V_\ell)\right]\right)^{-1} - 1, \tag{6}$$

$$\frac{W_{\max}}{W_{\text{avg}}} \geq \frac{K(1+\rho)W_{\text{avg}}}{\sum_{\ell=1}^{k-1} w(V^r_\ell) + \kappa_k + \sum_{\ell=k}^{K} w(V_\ell)},$$

$$\kappa_k \geq \frac{K(1+\rho)W^2_{\text{avg}}}{W_{\max}} - \sum_{\ell=1}^{k-1} w(V^r_\ell) - \sum_{\ell=k}^{K} w(V_\ell).$$

In Equation (6), we know that the maximum part weight will be less than or equal to $(1+\rho)W_{\text{avg}}$ given in the numerator. Note that the smaller values will have a positive impact and decrease the imbalance. In the denominator, we compute the average part weight by summing over the existing part weights including the replications performed so far.

In a majority of the undirectional hypergraph models (particularly, in information retrieval systems and geospatial databases), vertex replications do not incur any extra costs, and furthermore, are likely to improve the overall performance. For instance, vertex replications increase the probability of vertices' availability in a database system and as a result of this, multiple copies of vertices provide a potential to decrease the cache misses. Hence, considering the nature of the problem at hand, we use selective vertex replications to preserve the imbalance. That being said, calibration of κ_k might be used as well. Note that in both cases, the initial imbalance will be preserved.

In our model, at the kth iteration of the algorithm, we try to raise $w(V_k)$ to $(1+\rho)W_{\text{avg}}$, 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.

5. Experimental Results

In this section, experimental results evaluated for various data set collections with different model configurations are presented. First, in §5.1, experimental data set collections are detailed. Next, implementation details are given in §5.2. In §5.3, we present the results regarding the initial partitions of the data sets. Then, in §5.4, the replication results for cut size and imbalance reductions are given. Next, in §§5.5 and 5.6, we discuss the effect of coarsening and part-ordering schemes. Finally, we discuss the running time constraints in §5.7.

Note that for a fully detailed specification of all available data sets, partitioning and replication results for various parameters and their graphical comparisons, please consult the provided online supplement (available as supplemental material at http://dx.doi.org/10.1287/ijoc.2013.0567).

5.1. Data Set Collection

There are various hypergraph models successfully incorporated into spatial database (Shekhar et al. 2002, Demir et al. 2008) and information retrieval (Boley et al. 1999, Hotho et al. 2006) systems. For experimentation purposes, we use sample hypergraphs from these domains and investigate 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 U.S. Tiger/Line (Bureau 2002) [Minnesota including seven counties Anoka, Carver, Dakota, Hennepin, Ramsey, Scott, Washington; San Francisco; Oregon; New Mexico; Washington], U.S. Department of Transportation (2004) [California Highway Planning Network], and Brinkhoff's network data generator (Brinkhoff 2002) [Oldenburg; San Joaquin]. Hypergraphs for RN data

Table 1 Data Set Properties

Type	$\mathcal{H}$	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{N}\rvert$	$\lvert Pins\rvert$	$d^{\mathcal{N}}_{\text{avg}}$	c_{avg}	$d^{\mathcal{V}}_{\text{avg}}$	w_{avg}
RN	California	14,185	33,414	94,857	2.8	6.7	6.7	53.4
	Minnesota	46,103	78,371	239,422	3.1	13.1	5.2	53.5
	NewMexico	556,115	781,219	2,270,120	2.9	8.9	4.1	49.5
	Oldenburg	5,389	13,003	32,945	2.5	8.4	6.1	46.9
	Oregon	601,672	811,166	2,332,870	2.9	9.5	3.9	48.3
	SanFrancisco	213,371	319,305	967,917	3.0	9.1	4.5	51.5
	SanJoaquin	22,987	44,944	131,603	2.9	8.3	5.7	52.5
	Washington	652,063	824,650	2,427,615	2.9	11.8	3.7	49.0
	Wyoming	317,100	512,754	1,443,433	2.8	9.0	4.6	49.0
IR	CalGovernor	92,279	30,805	3,004,908	97.5	1.0	32.6	1.0
	Facebook	4,618,974	66,568	14,277,456	214.5	1.0	3.1	1.0
	Wikipedia	1,350,762	70,115	43,285,851	617.4	1.0	32.0	1.0
	Stanford	281,903	281,903	2,312,497	8.2	1.0	8.2	8.2

sets are constructed according to the clustering model presented by Demir et al. (2008).

To examine the effect of replication in information retrieval (IR) systems, text crawls are downloaded from the Stanford WebBase project (2006, 2010) [CalGovernor, Facebook, Wikipedia] and University of Florida Sparse Matrix Collection (Davis and Hu 2011) [Stanford]. In these information retrieval data sets, hypergraphs are constructed in such a way that terms correspond to vertices and documents correspond to nets. This construction scheme directly reflects the utilization of hypergraph models in information retrieval literature and a detailed explanation is given by Cambazoglu and Aykanat (2006).

In Table 1, properties of the hypergraphs extracted from the collected data sets are presented. Here, $|Pins|$ denotes the total number of pins in $\mathcal{N}$ of the related hypergraph, that is, $|Pins| = \sum_{n_j\in\mathcal{N}} |Pins(n_j)|$. The $d^{\mathcal{N}}_{\text{avg}}$ and $d^{\mathcal{V}}_{\text{avg}}$ columns represent average net and vertex degrees, respectively. Likewise, c_{avg} and w_{avg} denote average net costs and vertex weights, respectively. In Table 1, hypergraphs are grouped according to their domains (RN and IR) and sorted in increasing $|Pins|$ order.

Compared to IR hypergraphs, RN hypergraphs have relatively small average net degrees. This gives the intuition that in RN data sets, covering a net requires less vertex replications compared to IR data sets. Moreover, a high amount of replication capacity ρW_{avg} (i.e., $\rho|\mathcal{V}|w_{\text{avg}}$) is anticipated to result in more net coverage, and consequently, more decrease in the cut size. Hence, low values of $|\mathcal{V}|w_{\text{avg}}$ are presumed to produce relatively poorer replication results. For instance, this observation is highly expected to hold for Oldenburg and CalGovernor data sets, where $|\mathcal{V}|w_{\text{avg}}$ values are relatively low.

5.2. Implementation Details

Experiments are carried out on a Debian GNU/Linux 6.0.5 (x86_64) system running on an Intel Xeon (2.4 GHz) Processor. During tests, ANSI C sources are compiled using `gcc` bundled with release 4.3 of GNU Compiler Collections, where `CFLAGS` is set to `-O3-pipe-fomit-frame-pointer`. IBM ILOG CPLEX 12.1 is used in single-threaded mode to solve ILP problems. PaToH (Catalyurek and Aykanat 1999) v3.1 is used with default parameters for initial partitioning of the data sets. Coarsening is disabled for boundary adjacency hypergraphs where the total number of pins are less than or equal to 30.

5.3. Initial K-Way Hypergraph Partitioning

In a two-phase approach, it is assumed that an initial partition of the hypergraphs is a priori provided. For this purpose, we partition the hypergraphs according to the connectivity cut-size metric for two different K values (128 and 256). In Table 2, partition properties of the hypergraphs are given. Here, $\chi(\Pi)$ denotes the connectivity cut size of the partition; $ibr(\Pi)$ stands for imbalance ratio multiplied by 100; $|\mathcal{N}^*|$ and $|\mathcal{V}^*|$ columns denote the total number of cut nets and boundary vertices, respectively. Likewise, $d^{\mathcal{N}*}_{\text{avg}}$ and $d^{\mathcal{V}*}_{\text{avg}}$ represent average cut net and boundary vertex degrees; c^*_{avg} and w^*_{avg} denote average cut net costs and boundary vertex weights, respectively. A close look at Table 2 will reveal peaks in imbalance ratios for particular hypergraphs, i.e., San Joaquin, Minnesota, Stanford. These peaks come from the fact that such hypergraphs consist of a custom family of vertices with excessively high weights. Hence, the partitioner cannot move the vertices in a proper manner between parts to preserve the imbalance.

5.4. Replication Results

In Table 3, replication results are listed for hypergraph partitions given in Table 2. Here, $\chi(\%)$ denotes the reduction in connectivity cut size in percentages, i.e., $\chi(\%) = (1 - \chi(\Pi^r)/\chi(\Pi)) \times 100$. Here, part visit ordering scheme and ILP phase time limit are set to $\mathcal{O}_1$ and $\alpha = 1$, respectively. For the rest of the

Table 2 Properties of Hypergraph Partitions

Type	K	$\mathcal{H}$	$\chi(\Pi)$	$ibr(\Pi)$	$\lvert\mathcal{N}^*\rvert$	$d_{\text{avg}}^{\mathcal{N}*}$	c_{avg}^*	$\lvert\mathcal{V}^*\rvert$	$d_{\text{avg}}^{\mathcal{V}*}$	w_{avg}^*
RN	128	California	20,877	6.7	3,557	3.2	5.7	2,918	7.2	56.0
		Minnesota	39,633	4.2	3,660	3.3	10.7	3,322	6.9	58.7
		NewMexico	44,304	4.7	6,510	3.0	6.8	6,874	5.4	51.9
		Oldenburg	15,805	4.5	2,034	2.8	7.7	1,537	7.1	50.4
		Oregon	50,172	5.3	6,930	3.0	7.2	7,350	5.3	51.5
		SanFrancisco	45,089	7.0	6,263	3.3	7.2	6,195	6.2	56.5
		SanJoaquin	27,834	24.3	3,674	3.3	7.4	3,124	7.2	57.6
		Washington	59,526	6.1	6,593	3.1	9.0	7,121	5.4	53.2
		Wyoming	46,231	5.0	6,622	3.0	7.0	6,648	5.6	51.1
	256	California	35,246	4.6	5,669	3.2	6.0	4,594	7.2	56.0
		Minnesota	66,437	38.5	5,999	3.3	11.0	5,540	6.7	58.9
		NewMexico	71,904	5.5	10,432	3.1	6.9	10,996	5.5	52.5
		Oldenburg	24,328	8.2	3,112	2.8	7.5	2,294	7.0	50.1
		Oregon	77,832	4.6	10,760	3.1	7.2	11,437	5.4	52.0
		SanFrancisco	72,240	10.6	9,900	3.3	7.2	9,914	6.2	56.6
		SanJoaquin	44,470	92.7	5,705	3.2	7.5	4,821	7.1	57.4
		Washington	92,628	15.6	10,169	3.2	9.1	11,093	5.4	53.7
		Wyoming	71,645	6.5	10,110	3.0	7.1	10,306	5.6	51.4
IR	128	CalGovernor	199,548	5.6	24,825	118.5	1.0	92,268	32.6	1.0
		Facebook	318,957	1.4	58,301	235.2	1.0	4,611,075	3.1	1.0
		Wikipedia	1,043,753	4.2	69,047	623.9	1.0	1,350,588	32.0	1.0
		Stanford	15,993	948.0	9,297	114.7	1.0	169,643	10.9	10.9
	256	CalGovernor	298,092	6.3	26,417	112.1	1.0	92,278	32.6	1.0
		Facebook	422,341	1.2	61,826	225.9	1.0	4,617,288	3.1	1.0
		Wikipedia	1,468,648	4.9	69,491	621.2	1.0	1,350,735	32.0	1.0
		Stanford	24,003	776.6	12,441	93.8	1.0	172,207	10.9	10.9

article, unless otherwise noted, these default values will be assumed. Discussions of these parameters are presented in §§5.6 and 5.7. $ibr(\%)$ represents the reduction in imbalance ratio in percentages, i.e., $ibr(\%) = (1 - ibr(\Pi^r)/ibr(\Pi)) \times 100$. Fields $\chi^N(\%)$ and $\chi^P(\%)$ investigate the effect of coarsening on the overall cut-size reduction; $\chi^N(\%)$ denotes the reduction in connectivity cut size in percentages, where coarsening is totally turned off; and $\chi^P(\%)$ represents the best achieved connectivity cut-size reduction in percentages that is chosen among all PaToH coarsening algorithms. The other two columns provide insight to the characteristics of the generated boundary adjacency hypergraphs and the effect of coarsening. That is, $|Pins(\mathcal{H}_k)|$ denotes the average of the total number of pins of each $\mathcal{H}_k$, i.e., $|Pins(\mathcal{H}_k)| = (\sum_{k=1}^{K} \sum_{n_j \in \mathcal{N}_k} |Pins(n_j)|)/K$, and $|Pins(\%)|$ denotes the reduction in pin count after coarsening, i.e., $|Pins(\%)| = (1 - |Pins(\mathcal{H}_k^{\text{coarse}})|/|Pins(\mathcal{H}_k)|) \times 100$.

In Table 3, since $d_{\text{avg}}^{\mathcal{N}*}$ values are approximately the same for the whole RN collection, 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 results in low quality replications because of the low replication capacity of $\rho|V|w_{\text{avg}}$. On the other hand, for RN hypergraphs with high $|\mathcal{V}|$ values—e.g., Wyoming, NewMexico, Oregon, and Washington—replication removed almost every external net from the cut. By looking at imbalance and cut-size reductions in Table 3, RN hypergraphs appear as a perfect candidate for replication and yield remarkably promising results. That is, with diminutive amounts of replication, it is possible to remove almost all external nets from the cut by replication in RN hypergraph partitions.

For IR data sets, since $d_{\text{avg}}^{\mathcal{N}*}$ values of the partitions are much larger than those of the RN hypergraphs, covering nets require many more vertex replications. Consequently, high replication percentages are a common practice in IR systems. To address this issue, replication is evaluated with relatively higher ρ values of 10% and 20% for IR data sets. Compared to RN hypergraph partitions, both $|\mathcal{V}|$ and $d_{\text{avg}}^{\mathcal{N}*}$ values are quite varying among IR hypergraph partitions and both have a more prominent effect on the quality of the replication. For instance, high $|\mathcal{V}|$ and low $d_{\text{avg}}^{\mathcal{N}*}$ values are the major drivers behind the remarkably successful replication results for Facebook hypergraph partitions. On the other hand, compared to Facebook, replication outcomes are slightly poorer for Wikipedia hypergraph partitions because of the low $|\mathcal{V}|$ and high $d_{\text{avg}}^{\mathcal{N}*}$ values. As results in Table 3 point out, given enough replication capacity, depending on the hypergraph and partition characteristics ($|\mathcal{V}|$, $d_{\text{avg}}^{\mathcal{N}*}$, etc.), replication in IR data sets is capable of removing a notable majority of the external nets from

Table 3 Replication Results

Type	ρ	K	$\mathcal{H}$	χ(%)	*ibr*(%)	χ^N(%)	χ^P(%)	$\lvert Pins(\mathcal{H}_k)\rvert$	$\lvert Pins(\%)\rvert$
RN	0.01	128	California	16.0	15.7	16.1	10.9	55.0	70.6
			Minnesota	36.5	24.6	36.4	27.3	63.7	76.1
			NewMexico	99.7	22.0	99.7	99.7	82.9	27.3
			Oldenburg	10.2	22.9	10.2	7.2	19.7	25.8
			Oregon	100.0	19.7	100.0	100.0	90.3	22.9
			SanFrancisco	76.8	15.1	77.1	70.6	86.0	51.1
			SanJoaquin	19.6	5.1	19.6	15.0	53.0	70.6
			Washington	100.0	17.2	100.0	100.0	89.0	29.3
			Wyoming	98.1	21.0	98.1	97.1	85.5	33.6
		256	California	9.5	22.6	9.5	6.1	19.7	22.2
			Minnesota	25.1	3.6	24.9	16.1	62.9	75.9
			NewMexico	100.0	18.9	100.0	100.0	68.7	30.4
			Oldenburg	5.6	13.1	5.6	4.6	13.4	3.6
			Oregon	100.0	22.5	100.0	100.0	75.9	39.0
			SanFrancisco	49.5	10.4	49.6	43.5	73.4	63.9
			SanJoaquin	13.4	2.1	13.3	9.1	37.5	63.4
			Washington	100.0	7.3	100.0	100.0	73.4	31.8
			Wyoming	68.8	16.2	69.1	62.9	71.6	52.1
IR	0.10	128	CalGovernor	4.1	100.0	5.0	1.6	7,606.8	93.7
			Facebook	45.8	100.0	28.6	16.8	628,544.1	98.8
			Wikipedia	12.8	100.0	4.6	3.4	3,134,367.4	98.6
			Stanford	51.1	10.0	51.7	22.6	2,167.8	91.2
		256	CalGovernor	1.4	100.0	2.4	0.9	1,985.1	90.7
			Facebook	45.2	100.0	31.1	18.7	423,512.8	98.3
			Wikipedia	7.9	100.0	5.5	4.4	555,788.4	97.8
			Stanford	40.5	10.3	41.0	20.2	1,081.7	86.2
	0.20	128	CalGovernor	9.7	100.0	6.9	4.0	34,652.8	96.2
			Facebook	59.1	100.0	38.2	26.0	576,620.1	98.8
			Wikipedia	22.9	100.0	6.4	3.9	5,516,646.7	99.2
			Stanford	65.8	18.4	70.9	34.9	2,916.6	89.5
		256	CalGovernor	4.5	100.0	5.3	2.5	6,422.8	92.8
			Facebook	58.6	100.0	40.0	27.8	383,796.1	98.4
			Wikipedia	17.3	100.0	9.1	6.8	2,199,807.3	98.1
			Stanford	53.9	18.8	54.7	29.9	1,609.1	83.7

the cut while also providing almost perfect imbalance improvements.

5.5. Coarsening Results

In Table 3, the last two columns provide information about the average size of the constructed boundary adjacency hypergraphs and the contraction percentage due to the coarsening, that is, $|Pins(\mathcal{H}_k)|$ and $|Pins(\%)|$, respectively. Coarsening reduces the size of the constructed boundary adjacency hypergraphs by 43.9% for RN data sets, and 94.5% for IR data sets, on average. This significant difference between the contraction percentages of RN and IR collections are because of the connected component construction mechanism of the Dulmage-Mendelsohn decomposition. That is, boundary adjacency hypergraphs with high $d_{\text{avg}}^{\mathcal{N}^*}$ values as in IR hypergraph partitions, result in more tightly-coupled vertices in the bipartite graphs passed to the coarsening phase. And in the coarsening, bipartite graphs with high vertex degrees result in less number of connected components. At first glance, such bipartite graphs are anticipated to end up with low-quality clusters during coarsening, that is, a significant amount of information loss is expected compared to the case where there would be no clustering at all. But as test results point out, which will be detailed further next, Dulmage-Mendelsohn decomposition successfully encapsulates the replication characteristics of the individual clusters even with excessive contraction ratios.

As experimental results point out, the Dulmage-Mendelsohn decomposition performs quite effectively in terms of the hypergraph coarsening quality. That is, the sizes of the coarsened hypergraphs are quite small, whereas produced hypergraphs retain the characteristics of the actual pins and edges in terms of replication quality. However, that being said, the Dulmage-Mendelsohn decomposition does not take vertex weights and net costs into account. Moreover, bipartite graphs with high vertex degrees might end up with low-quality clusters because of the contraction mechanism based on connected component extraction. Hence, it raises the question: could it be possible to produce a more effective coarsening phase by taking mentioned constraints 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. Moreover, a pseudocoarsening scheme is introduced where no coarsening is performed at all to establish a baseline. Using these clustering algorithms, we then repeated the whole replication procedure and observed their effects on the final cut size. These results are given in the $\chi^N(\%)$ and $\chi^P(\%)$ columns of Table 3.

When the results in $\chi(\%)$ and $\chi^P(\%)$ columns of Table 3 are to be compared, Dulmage-Mendelsohn decomposition on the average achieves a cut-size reduction of 1.81 times better than the average of the best achieved results among PaToH coarsening algorithms. In the case where coarsening is turned off, for relatively small data sets (e.g., RN, CalGovernor, and Stanford) $\chi(\%)$ and $\chi^N(\%)$ values are almost equivalent. This evidences that the Dulmage-Mendelsohn decomposition successfully achieves to preserve the replication characteristics of the initial hypergraph. The advantage of coarsening is more prominently observable for large data sets such as Facebook and Wikipedia. This is because a time-limited ILP phase is anticipated to achieve better cut-size reductions when given smaller hypergraphs. This observation holds with the experimental results: In terms of cut-size reduction for Facebook and Wikipedia data sets, replication with the Dulmage-Mendelsohn coarsening scheme on the average performs 1.97 times better than the case where coarsening is totally turned off. As presented results indicate, Dulmage-Mendelsohn decomposition shows remarkable results for preserving the replication characteristics of the initial hypergraph. Moreover, the lower degree improvements in the cut size for the case where coarsening is turned off confirms the necessity of coarsening for large data sets.

5.6. Part Visit Orderings

In the proposed model, because parts are processed in some particular order, ordering of the parts plays an important role on the final quality of the replication scheme. That is, consider a net n_j such that $\Lambda(n_j)=\{V_\ell, V_k\}$ and $w(Pins(n_j)\cap V_\ell) < w(Pins(n_j)\cap V_k)$. In such a case, one definitely would prefer to cover net n_j by consuming the least possible amount of the given replication capacity, that is, by replicating vertices from V_ℓ to V_k, hence part V_k is expected to be processed first for covering net n_j. On the other hand, considering the massive amount of vertex connections scattered among parts, complexity of this ordering decision increases proportional to the total number of cut nets, boundary vertices, and their degrees.

To investigate the effect of part visit orderings on the final cut size, three different ordering schemes $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3$ are adopted in the conducted experiments. In the first scheme $\mathcal{O}_1$, parts are ordered by increasing average net degrees of their boundary adjacency hypergraphs. This scheme conveys the ordering that aims to reduce the cut size as much as possible in the earlier iterations. In the second scheme $\mathcal{O}_2$, parts are sorted in increasing weight order. This second ordering tries to maximize the balance at the end of the replication. And in the last scheme $\mathcal{O}_3$, parts are chosen randomly to establish a baseline for the comparison of the evaluated part visit orderings.

In the conducted experiments, different part visit orderings resulted in minor variances on the final quality of the replication. That is,

$$\sqrt{\frac{1}{n}\sum_i\big[\chi_i(\%,\mathcal{O}_k)-\chi_i(\%,\mathcal{O}_\ell)\big]^2}\le 4.56,$$

where $\chi_i(\%,\mathcal{O}_k)$ denotes the connectivity cut-size reduction in percentages using ordering scheme $\mathcal{O}_k$ for $k\in\{1,2,3\}$. Subscript i is used to address the sample space formed of different K and ρ settings of available data sets; n denotes the size of this sample space. Being that said, in a majority of the tests, ordering scheme $\mathcal{O}_1$ performed at the top in terms of the cut-size reduction. Further investigation of the empirical results point out that the ILP phase coupled with the coarsening performs quite effectively in terms of consuming the available replication capacity and, as a result of this, different part visit orderings do not have a significant impact on the final cut size and imbalance.

5.7. The Effect of Running Time

In an efficient two-phase model implementation, time spent in the replication phase is anticipated to be upper bounded by a constant factor of the running time spent in the initial partitioning phase empowered by state-of-the-art hypergraph partitioning tools. In our proposed replication phase, selection of the vertices from boundary adjacency hypergraphs is anticipated to dominate the total running time due to the $\mathcal{NP}$-hardness of the problem. As a first step, we introduced a polynomial time coarsening scheme to minimize this duration. Later, we put an upper bound on the maximum running time of the ILP solver such that the total amount of time spent for ILP runs cannot exceed a constant factor of the time spent in the partitioning phase.

Let $\mathcal{T}_{\text{part}}$ denote the time spent for partitioning the hypergraph. In the conducted experiments, $\mathcal{T}_{\text{ilp}} = \alpha\mathcal{T}_{\text{part}}/K$ is used to upper bound the time spent for a single ILP run. (Replication results given in Table 3 are evaluated for $\alpha=1$, that is, the total time spent for ILP is upper bounded by the partitioning runtime.)

To investigate the effect of ILP runtime on the final replication quality, experiments are evaluated with different α settings. In a majority of the tests, increasing α did not provide an impact as expected and resulted in minor improvements in the final cut size. That is,

$$\max\left|\chi_i(\%,\alpha_k)-\chi_i(\%,\alpha_\ell)\right|\le 1.01,$$
$$\max\left|\chi_i^N(\%,\alpha_k)-\chi_i^N(\%,\alpha_\ell)\right|\le 11.51,$$

for $\alpha_k,\alpha_\ell\in\{1,2,4\}$. Here, $\chi_i(\%,\alpha_k)$ denotes the cut-size reduction in percentages using Dulmage-Mendelsohn decomposition, where α is set to α_k. Likewise, $\chi_i^N(\%,\alpha_k)$ denotes the cut-size reduction in percentages without any coarsening and α is set to α_k. (Subscript i is used to address the sample space formed of different K and ρ settings of available data sets.)

This negligible amount of quality difference between varying α configurations of $\chi(\%,\alpha)$ holds with the fact that a majority of the ILP running time is generally known to be consumed to prove the optimality of the so far found solution and iteratively make small improvements in the branches being processed of the whole branch-and-bound procedure. Hence, a good quality solution is expected to be produced in a small amount of time at the beginning. This observation is more prominent with our findings that the ILP phase performed with compact hypergraphs produced by Dulmage-Mendelsohn decomposition as input, achieves to find a good quality solution even when α is set to 1. On the other hand, as boundary adjacency hypergraphs get denser, the complexity of the constraints in the ILP formulation linearly increases. In such a case, increasing the α might turn out to be a viable option. Although boundary adjacency hypergraphs constructed from IR data sets appear to be excessively denser, in the conducted experiments, coarsening does a good job at contracting these hypergraphs before passing them to ILP, and considerably diminishes the effect of the dense input data. Since this optimization does not exist for the case where coarsening is turned off, increasing α yields improvements up to 11.51% in the cut-size reduction for $\chi^N(\%,\alpha)$. To sum up, in our proposed two-phase approach, giving the same amount of time to the first and second phases is sufficient for the replication to perform in an efficient and effective way.

6. Conclusion

Motivated by the problem of hypergraph partitioning with vertex replication, we first presented a detailed complexity analysis of the problem, and then proposed a part-oriented approach based on a unique blend of ILP and coarsening schemes. As detailed in the previous sections, we provided a succinct proof of the problem itself and implied subproblems that are found to be $\mathcal{NP}$-hard. Later, we proposed a part-oriented approach with a novel coarsening scheme utilizing the Dulmage-Mendelsohn decomposition. In the experiments, we compared our model with different configurations and observed the effect of the change in the results. Conducted experiments showed that the proposed model provides high quality results in practical execution times.

Supplemental Material

Supplemental material to this paper is available at http://dx.doi.org/10.1287/ijoc.2013.0567.

Acknowledgments

A10 The authors thank the anonymous reviewers for their helpful and constructive comments that greatly contributed to improving the final version of the paper.

References

Alpert CJ, Kahng AB (1995) Recent directions in netlist partitioning: A survey. *Integr. VLSI J.* 19:1–81.

Boley D, Gini M, Gross R, (Sam) Han E-H, Hastings K, Karypis G, Kumar V, Mobasher B, Moore J (1999) Document categorization and query generation on the world wide web using WebACE. *Artificial Intelligence Rev.* 13:365–391.

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

A11 Bureau U.S. Census (2002) Topologically integrated geographic encoding and referencing system (*TIGER*). *U.S. Census Bureau.* http://www.census.gov/geo/www/tiger/.

A12 Cambazoglu BB, Aykanat C (2006) A term-based inverted index organization for communication-efficient parallel query processing. Tokyo, Japan.

A13 Catalyurek UV, Aykanat C (1999) *PaToH*: Partitioning tool for hypergraphs. Department of Computer Engineering, Bilkent University, Ankara, Turkey.

A14 Cho J, Garcia-Molina H, Haveliwala T, Lam W, Paepcke A, Raghavan S, Wesley G (2006) Stanford WebBase components and applications. *ACM Trans. Internet Tech.* 6:153–186.

Davis TA, Hu Y (2011) The University of Florida sparse matrix collection. *ACM Trans. Math. Software* A15 38:1:1–1:25.

Demir E, Aykanat C, Cambazoglu BB (2008) Clustering spatial networks for aggregate query processing: A hypergraph approach. *Inform. System* 33:1–17.

Dulmage AL, Mendelsohn NS (1958) Coverings of bipartite graphs. *Can. J. Math.* 10:517–534.

Dulmage AL, Mendelsohn NS (1959) A structure theory of bipartite graphs of finite exterior dimension. *Trans. Roy. Soc. Can. Sec.* 3:1–13.

Dulmage AL, Mendelsohn NS (1963) Two algorithms for bipartite graphs. *J. Soc. Ind. Appl. Math.* 2:A16 183–194.

A17 Enos M, Hauck S, Sarrafzadeh M (1997) Replication for logic bipartitioning. *ICCAD '97: Proc. 1997 IEEE/ACM Internat. Conf. Comput.-Aided Design* (IEEE Computer Society, Washington, DC), 342–349.

Garey MR, Johnson DS (1990) *Computers and Intractability; A Guide to the Theory of NP-Completeness* (W. H. Freeman & Co., New York).

Goldschmidt O, Nehme D, Yu G (1994) Note: On the set-union knapsack problem. *Naval Res. Logist.* (*NRL*) 41:833–842.

A18 Hotho A, Jäschke R, Schmitz C, Stumme G (2006) Information retrieval in folksonomies: Search and ranking. *The Semantic Web: Research and Applications*, Vol. 4011 (Springer), 411–426.

Hwang J, El Gamal A (1992) Optimal replication for min-cut partitioning. *ICCAD '92: Proc. 1992 IEEE/ACM Internat. Conf. Comput.-Aided Design* (IEEE Computer Society Press, Los Alamitos, CA), 432–435.

A19 Hwang LJ (1994) Replication in partitioned networks. Ph.D. thesis.

Johnson DM, Dulmage AL, Mendelsohn NS (1962) Connectivity and reducibility of graphs. *Can. J. Math.* 14:529–539.

A20 Kring C, Newton AR (1991) A cell-replicating approach to mincut-based circuit partitioning. *Comput.-Aided Design, 1991. ICCAD-91. Digest of Technical Papers., 1991 IEEE Internat. Conf.*, 2–5.

Kužnar R, Brglez F, Zajc B (1994) Multi-way netlist partitioning into heterogeneous FPGAs and minimization of total device cost and interconnect. *DAC '94: Proc. 31st Annual Design Automation Conf.* (ACM, New York), 238–243.

Lengauer T (1990) *Combinatorial Algorithms for Integrated Circuit Layout* (Willey–Teubner, Chichester, UK).

A21 Murgai R, Brayton RK, Sangiovanni-Vincentelli A (1991) On clustering for minimum delay/ara. *Comput.-Aided Design, 1991. ICCAD-91. Digest of Technical Papers., 1991 IEEE Internat. Conf.*, 6–9.

Pothen A (1984) Sparse null bases and marriage theorems. Ph.D. thesis, Cornell University, Ithaca, NY.

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

Russo RL, Oden PH, Wolff PK (1971) A heuristic procedure for the partitioning and mapping of computer logic graphs. *IEEE Trans. Comput.* 20:1455–1462.

Selvitopi OR, Turk A, Aykanat C (2012) Replicated partitioning for undirected hypergraphs. *J. Parallel Distrib. Comput.* 72:547–563.

Shekhar S, Lu C-T, Chawla S, Ravada S (2002) Efficient join-index-based spatial-join processing: A clustering approach. *IEEE Trans. Knowledge Data Engrg.* 14:1400–1421.

Tarjan R (1972) Depth-first search and linear graph algorithms. *SIAM J. Comput.* 1:146–160.

A22 The Stanford WebBase Project (2010) The Stanford WebBase Project. http://diglib.stanford.edu:8091/~testbed/doc2/WebBase.

U.S. Department of Transportation (2004) Federal Highway Administration, the national highway planning network. A23 http://www.fhwa.dot.gov/planning/nhpn/.

Yang HH, Wong DF (1995) New algorithms for min-cut replication in partitioned circuits. *ICCAD '95: Proc. 1995 IEEE/ACM Internat. Conf. Comput.-Aided Design* (IEEE Computer Society, Washington, DC), 216–222.

Author Queries

A1 Au: Please confirm title, author names, and affiliations. Add postal codes to both affiliations.

A2 Au: Please confirm key words.

A3 Au: Spell out VLSI at first use

A4 Au: Please recast sentence as to not begin with math, per style.

A5 Au: Recast sentence to avoid beginning with math term

A6 Au: Should "PARTITION" be in all caps here?

A7 Au: Check all figures and tables as set.

A8 Au: "WebBase project (2006)" not in reference list. Please add to references or delete citation.

A9 Au: Recast the sentence to avoid beginning with math terms, per style

A10 Au: Please provide all funding information, if applicable, including the names of the institutions as well as any grant numbers associated with the funding.

A11 Au: Please provide access date. More info?

A12 Au: What type of reference is this? Thesis, report? Add more info.

A13 Au: What type of reference is this? Thesis, report?

A14 Au: Cho et al. (2006) not cited in text. Please cite or delete.

A15 Au: Please verify the page numbers for Davis and Hu (2011).

A16 Au: Confirm volume number

A17 Au: Add editors?

A18 Au: Add editors and specific publisher location.

A19 Au: Provide affiliation

A20 Au: Conference location? Publisher and location of pub.?

A21 Au: Publisher and location of pub.?

A22 Au: Add last access date

A23 Au: Add last access date