

Applications of the Dulmage-Mendelsohn Decomposition and Network Flow to Graph Bisection Improvement

Cleve Ashcraft * Joseph W.H. Liu †

August 16, 1996

Abstract

In this paper, we consider the use of the Dulmage-Mendelsohn decomposition and network flow on bipartite graphs to improve a graph bisection partition. Given a graph partition $[S, B, W]$ with a vertex separator S and two disconnected components B and W , different strategies are considered based on the Dulmage-Mendelsohn decomposition to reduce the separator size $|S|$ and/or the imbalance between B and W . For the case when the vertices are weighted, we relate this with the bipartite network flow problem. A further enhancement is made on partition improvement by generalizing the bipartite network to solving a general network flow problem. We demonstrate the utility of these improvement techniques on a set of sparse test matrices, where we find top level separators and nested dissection and multisection orderings.

Key words. Dulmage-Mendelsohn decomposition, network flow, graph bisection, ordering algorithms, nested dissection, multisection.

AMS(MOS) subject classifications. 65F05, 65F50, 68R10.

1 Introduction

The ability to find a good separator for a graph is necessary in many application areas [16], [28]. Our motivation to consider this problem is to determine good sparse matrix orderings for direct factorization methods [4], [5], [19], [24].

In a recent paper [2], the authors have applied the notion of blocking to obtain an efficient graph partitioning scheme to find a good vertex separator. The approach has three basic steps. In the first step, we construct a *domain decomposition* of the graph, consisting of a subset of vertices (called a multiselector) whose removal decomposes the graph into a number of *domains*. Each domain is a connected subset of vertices. The second step uses a variant of the Kernighan-Lin scheme [21] on the set of domains to determine an approximation to a good separator. The last step refines the separator using some techniques from bipartite graph matching. One purpose of this paper is to give a full explanation of the machinery used in the separator improvement step.

The fundamental tool used in this final step is the *Dulmage-Mendelsohn decomposition* [8], which is a canonical decomposition of a bipartite graph based on the notion of matching. This decomposition has been used extensively to extract a *vertex separator* from an *edge separator* [15], [20], [23], [26]. The vertices that are incident to an edge in the edge separator

*Boeing Information and Support Services, P. O. Box 24346, Mail Stop 7L-22, Seattle, Washington USA 98124. This research was supported in part by the ARPA Contract DABT63-95-C-0122.

†Department of Computer Science, York University, North York, Ontario, Canada M3J 1P3. This research was supported in part by the Natural Sciences and Engineering Research Council of Canada under grant A5509 and in part by the ARPA Contract DABT63-95-C-0122.

form a *wide* vertex separator. A vertex separator is a *cover* for the edge separator if all edges are incident to a vertex of the separator. Using the Dulmage-Mendelsohn decomposition, one can find one or more vertex covering separators of minimum size that are subsets of this wide vertex separator.

This decomposition has been used to improve a vertex separator in earlier papers [24], [25]. Let the vertices in the graph be partitioned as a vertex separator S and two components B and W . We consider the edge separator that contains edges linking vertices in S to B (or S to W). This defines a wide vertex set containing vertices in S and all vertices in B (or W) adjacent to S . We use the Dulmage-Mendelsohn decomposition to find a covering separator of minimum size from this set of vertices¹.

The same technique can be applied to the new separator and its new adjacent sets, so the overall improvement process is iterative in nature. At each step, a wide vertex separator is taken from the current separator and one of the two components, and a covering vertex separator subset of minimum size is obtained. It is accepted as the new vertex separator only if the quality of its induced partition is better.

In this paper, we consider related approaches, initially developed in [2], to improve a vertex separator. Although the Dulmage-Mendelsohn decomposition is defined only for unit-weight graphs, we are able to extend it to a special class of weighted graphs, thus greatly reducing the execution time in many cases. In the extension, we reformulate it into a much-studied combinatorial problem involving the flow of commodities through an interconnected network: a maximum network flow problem [9], [11], [12], [18], [27]. The solution to our separator improvement step is thus transformed to solving a maximum flow problem on a bipartite network. We also relate the improved separator in the new partition with the min-cut set in the well-known max-flow min-cut theorem on network flows.

We have explored an additional advantage in the transformation of the bipartite graph matching problem to bipartite network flow problem. By adding and deleting edges from the bipartite network we are able to construct a new network that may yield a smaller separator. The new network is not bipartite and the new separator need not be a covering separator. By adding vertices and edges we generate larger networks that can yield still smaller separators.

An outline of this paper is as follows. In Section 2, we give a formal description of the partition improvement problem and introduce the various notations used throughout the paper. Section 3 starts with a discussion on reducing the size of a separator using bipartite graph matching. This provides the motivation to the Dulmage-Mendelsohn decomposition for bipartite graphs. This section is mainly of exposition in nature; the results can be found in [24] and [25]. Section 4 considers the use of the Dulmage-Mendelsohn decomposition to improve the balance of a partition.

In Section 5, we introduce the notion of a *compressed* graph induced by a grouping of vertices that share the same adjacent sets. Compressed graphs can be considered as a special kind of weighted graphs. The Dulmage-Mendelsohn decomposition is then generalized to handle compressed bipartite graphs.

In Section 6, we relate this decomposition on a compressed bipartite graph to a max-flow solution to a bipartite network problem. We point out the equivalence between the generalized matching and a max-flow, and between the improved separator and a min-cut. A new enhancement is also described by transforming the bipartite network to a

¹It is a little-appreciated fact that a covering separator of minimum size may *not* be a separator of minimal size — i.e., a separator of minimal size may not be incident on all the edges of the edge separator.

general network based on the underlying graph structure. We show that a max-flow min-cut solution to this new general network is at least as good as and is often better than that of the bipartite network. We also generalize the network flow approach to even wider separators formed from the separator and many “layers” of adjacent sets from one or both components in the partition.

Section 7 contains experimental results on separator/partition improvements. We compare the improvement in partitions based on solving a max-flow problem on a bipartite network, the induced two-layer network and a centered three-layer wide network. Sparse matrix ordering statistics are also given when these techniques are used in a nested dissection and a multisection ordering code [3]. The multisection statistics are at least as good and often are better than those from the multiple minimum degree ordering approach. Section 8 contains our concluding remarks.

2 Definitions and Notations

Let $G = (V, E)$ be a given undirected graph. The adjacent set of a vertex v is given by:

$$Adj(v) = \{u \neq v \mid (u, v) \in E\}.$$

Without loss of generality, we assume the graph is connected. A *walk* is a sequence of vertices v_0, v_1, \dots, v_m such that $(v_i, v_{i+1}) \in E$. A *path* is a walk without any repeated vertices.

A vertex subset S is a *vertex separator* if the subgraph induced by the vertices in V but not in S has more than one connected component. An *edge separator* is a set of edges whose removal disconnects the graph. A separator is *minimal* if no subset of it forms a separator.

A *bisector* is a separator whose removal gives at least two connected components. We shall use the notation $[S, B, W]$ to represent a 2-set partition, where the removal of the bisector S will give two disconnected portions B and W ; that is, $Adj(B) \subseteq S$ and $Adj(W) \subseteq S$. We measure the *imbalance* of a partition as the dimensionless ratio $\max\{|B|, |W|\} / \min\{|B|, |W|\}$. We shall often assume that B is the bigger portion so that $|B| \geq |W|$ and the imbalance is $|B|/|W|$. Our objective is to determine a well-balanced partition with a small separator size $|S|$.

In this paper, we consider methods to improve a given partition. Therefore, we need to compare the quality of the original and the modified partitions. Following [2], we use this evaluation function

$$\gamma[S, B, W] = |S| \left(1 + \alpha \frac{\max\{|B|, |W|\}}{\min\{|B|, |W|\}} \right),$$

where α is some constant greater than 0. The separator size $|S|$ is the primary metric while the imbalance is used as a “penalty” multiplicative factor. A large value of the constant α places a large emphasis on the balance. We have used the penalty cost function $\gamma[S, B, W]$ with $\alpha = 1$ in all the experiments in Section 7.

Throughout the paper we will be concerned with a subset of vertices, those vertices just “outside” the subset, and those “inside” the subset. To make these concepts clear we introduce the following notation. Let Y be a vertex subset of V . The *interior* of Y is defined to be

$$Int(Y) = \{y \in Y \mid Adj(y) \subseteq Y\},$$

and contains all nodes in Y that are adjacent to no nodes outside of Y . The *boundary* of

Y or its *adjacent set* is the set of nodes not in Y that are adjacent to Y ,

$$Adj(Y) = \{v \in V \setminus Y \mid (y, v) \in E \text{ for some } y \in Y\} = \left(\bigcup_{y \in Y} Adj(y) \right) \setminus Y.$$

The *border* of Y is a subset of Y , namely the boundary of the interior of Y ,

$$Border(Y) = Adj(Int(Y)) \cap Y = Y \setminus Int(Y)$$

or those nodes in Y that are not in the interior of Y .

3 Partition Improvement and the Dulmage-Mendelsohn Decomposition

3.1 A Partition Improvement Algorithm by Moves

Let $[S, B, W]$ be a 2-set partition of a given graph G . Consider a subset Z of S . Let $Z \mapsto W$ be the move of Z to W that moves the subset Z from S to W , thereby creating the following new partition:

$$B_{Z \mapsto W} = B \setminus Adj(Z), \quad W_{Z \mapsto W} = W \cup Z, \quad \text{and} \quad S_{Z \mapsto W} = (S \setminus Z) \cup (Adj(Z) \cap B).$$

We use the notation $[S, B, W]_{Z \mapsto W}$ to refer to the new partition.

We consider a partition improvement scheme that uses moves by finding subsets Z that will help in reducing the evaluation function $\gamma[S, B, W]$. A high-level description of the improvement algorithm is described in Figure 1. The scheme makes a first attempt to reduce the evaluation function of the partition by moving a subset from S to the smaller portion W . If no such move can be found, it tries to improve the partition by moving a separator subset to the larger portion B . It continues until no reduction can be obtained.

```

PARTITION-IMPROVE  $[S, B, W]$ 
  Improved = true
  while Improved do
    if  $|B| < |W|$  then interchange  $B$  and  $W$  // make  $B$  the larger portion
    if a subset  $Z$  of  $S$  is found with  $\gamma([S, B, W]_{Z \mapsto W}) < \gamma[S, B, W]$  then
       $[S, B, W] = [S, B, W]_{Z \mapsto W}$ 
    else
      if a subset  $Z$  of  $S$  is found with  $\gamma([S, B, W]_{Z \mapsto B}) < \gamma[S, B, W]$  then
         $[S, B, W] = [S, B, W]_{Z \mapsto B}$ 
      else
        Improved = false
      end if
    end if
  end while

```

FIG. 1. *Partition Improvement Scheme.*

3.2 Improving the Separator Size by Graph Matching

Recall that the evaluation function $\gamma[S, B, W]$ on a partition is given by the penalty function based on the separator size and the imbalance ratio. In practice, the weight α is chosen to

be close to one so that the separator size has a strong influence on the partition evaluation. Therefore, one way to look for an improvement to the partition is to reduce the separator size. Consider the move $Z \mapsto W$. The new separator size is given by:

$$|S_{Z \mapsto W}| = |S| - |Z| + |Adj(Z) \cap B|.$$

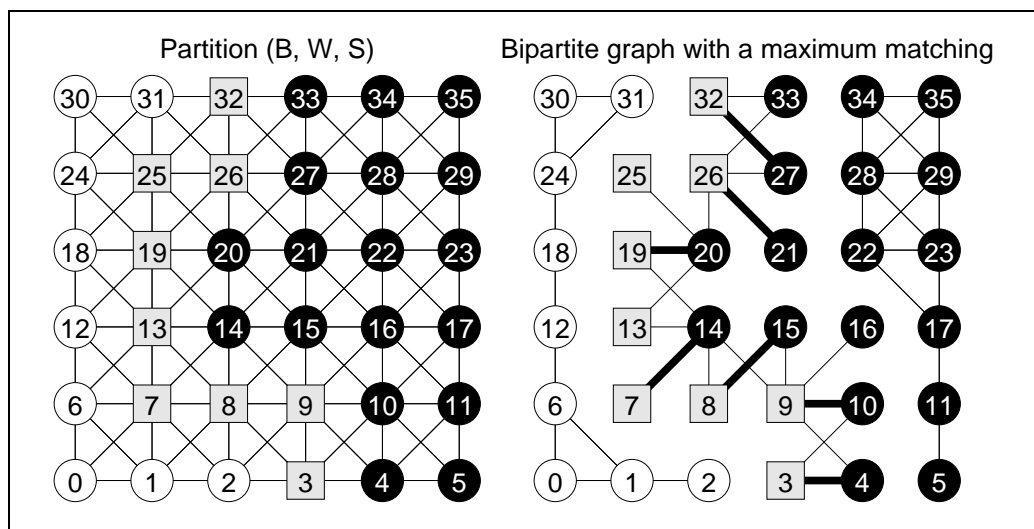
Therefore, if we can find a subset Z of S such that $|Z| > |Adj(Z) \cap B|$, the move of Z to W will result in a reduction of the separator size by an amount of $|Z| - |Adj(Z) \cap B|$. (Note that this does not always guarantee a reduction in the evaluation function value.)

In [24], the technique of bipartite graph matching is used to find such a subset Z of S with $|Z| > |Adj(Z) \cap B|$. We shall first describe the necessary terminologies in graph matching and state the results relevant to this approach.

A *bipartite graph* is an undirected graph whose node set can be divided into two disjoint sets X and Y such that every edge has one endpoint in X and the other in Y . A *matching* of a bipartite graph H is a subset M of edges such that no two edges in this subset have a node in common. A node that is incident to some edge in M is said to be *covered*; otherwise, it is *exposed*. If (x, y) belongs to the matching M , then $x = \text{mate}(y)$ and $y = \text{mate}(x)$. The number of edges in M is called the *size* of the matching. A *maximum matching* is one with the largest possible size. A *complete matching* is a matching of size $\min\{|X|, |Y|\}$.

We now consider the results in graph matching relevant to our context of improving a 2-set partition $[S, B, W]$. Assume that B is the larger portion. Consider the bipartite graph $H = (S, \text{Border}(B), E_H)$ where E_H contains the set of edges between vertices in S and those in $\text{Border}(B)$ of the original graph G . Recall that $\text{Border}(B) = B \cap \text{Adj}(S)$. For simplicity, we often refer to this bipartite graph by $H(S, B)$, and the two defining sets as S and B . However, it is implicit that only the subset $\text{Border}(B)$ of B is used in H . For a node x in this bipartite graph H , we shall use $\text{Adj}_H(x)$ to represent the set of adjacent nodes of x in the bipartite graph H . We extend the notation to $\text{Adj}_H(U)$ for the adjacent set of a subset U of nodes. Note that we use $\text{Adj}(x)$ and $\text{Adj}(U)$ to represent the adjacent sets in the original graph G . It should be clear from the definition of H that for any subset Z of S , $\text{Adj}_H(Z) = \text{Adj}(Z) \cap B$.

FIG. 2. *Bipartite Graph Example from a Separator Partition.*



In Figure 2, we illustrate the induced bipartite graph H for a 6-by-6 grid problem with

the 9-point operator; that is, each interior node is connected to its eight neighbors. For the given separator of size 9, we obtain its associated bipartite graph H . In the figure, a matching between S and B is also given; and the edges in the matching are indicated by thick lines. This matching is of size 7 and it is maximum.

In the separator improvement scheme, we want to find a subset Z of S satisfying $|Z| > |Adj(Z) \cap B|$. The next theorem by Hall [14] relates the *non-existence* of such a subset with bipartite graph matching.

THEOREM 3.1 (HALL [14]). *The bipartite graph H has a complete matching of S into B if and only if for every subset Z of S , $|Z| \leq |Adj_H(Z)| = |Adj(Z) \cap B|$.*

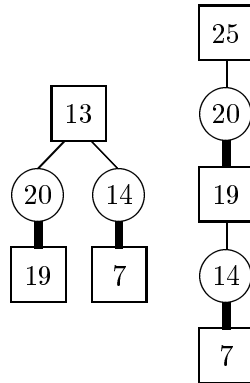
The result in Theorem 3.1 can be used to provide a necessary and sufficient condition for the existence of a size-improving subset Z of S . The condition is that the bipartite graph H does not have a complete matching from S into B . This implies that for a maximum matching, there will be some exposed nodes in S , that is nodes without a mate in the matching. In the example of Figure 2, there are two exposed nodes 13 and 25 in S so that the maximum matching is not complete. We know, by the result of Theorem 3.1 that we can find some size-improving subset Z of S .

To discuss the way to find such subsets, we need the notion of an alternating path. For a given matching M , consider a path $\langle x_0, x_1, \dots, x_k \rangle$ where no vertex is repeated. It is called *alternating* with respect to M if the alternate edges belong to the matching M . For example, in Figure 2, the path $\langle 25, 20, 19, 14, 7 \rangle$ is alternating; the edges $(20, 19)$ and $(14, 7)$ belong to the matching. In [24], alternating paths are used in the following result to find a subset Z satisfying $|Z| > |Adj(Z) \cap B|$.

THEOREM 3.2 (LIU [24]). *Let $x \in S$ be an exposed node in a maximum matching of H . Define $S_x = \{s \in S \mid s \text{ is reachable from } x \text{ via alternating paths}\}$. Then $|S_x| - |Adj_H(S_x)| = 1$.*

The set S_x can be determined by performing a special kind of breadth-first search starting from the exposed node x . The search is restricted to nodes reachable via alternating paths. Then S_x is given by the nodes of S appearing in this *breadth-first search tree* rooted at x . Since we only consider alternating paths in the traversal, we shall refer to this tree as an *alternating breadth-first search tree*. This set S_x can be used as Z to reduce the separator size by one.

For the example in Figure 2, there are two exposed separator nodes: 13 and 25. Immediately below we find the two alternating breadth-first search trees that start from 13 and 25 respectively. It is clear that $S_{13} = \{7, 13, 19\}$ and $Adj_H(S_{13}) = \{14, 20\}$. On the other hand, $S_{25} = \{7, 19, 25\}$ and $Adj_H(S_{25}) = \{14, 20\}$. Figure 3 shows the improvement of the separator by making the move $Z = S_{13}$ (see the top two grids) and the move $Z = S_{25}$ (see the middle two grids).

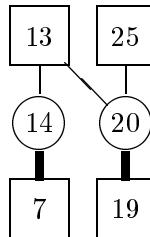


An alternating breadth-first search tree is a special case of an *alternating breadth-first level structure*. Let X_0 be some initial set of exposed nodes in S ; X_0 forms the first level. Define the next level $X_1 = Adj_H(X_0)$, namely those vertices in $Border(B)$ adjacent to vertices in X_0 . The next level X_2 contains all nodes in S that are mates with nodes in X_1 . In general, the level sets have the following form.

$$X_{2i} = \bigcup_{x \in X_{2i-1}} \text{mate}(x) \subseteq S$$

$$X_{2i+1} = Adj_H \left(\bigcup_{j=0}^{2i} X_j \right) \subseteq Border(B)$$

The move set is $Z = X_0 \cup X_2 \cup \dots$ while its boundary set is $Adj_H(Z) = X_1 \cup X_3 \cup \dots$. For example, the alternating breadth-first level structure for $X_0 = \{13, 25\}$ is found below.



The move set is $S_{\{13,25\}} = \{7, 13, 19, 25\}$ while its boundary is $Adj_H(S_{\{13,25\}}) = \{14, 20\}$. Figure 3 shows the improvement of the separator by making the move $Z = S_{\{13,25\}}$ (see the bottom two grids). Note that the resulting separator is smaller than the separator induced by the two move sets S_{13} and S_{25} .

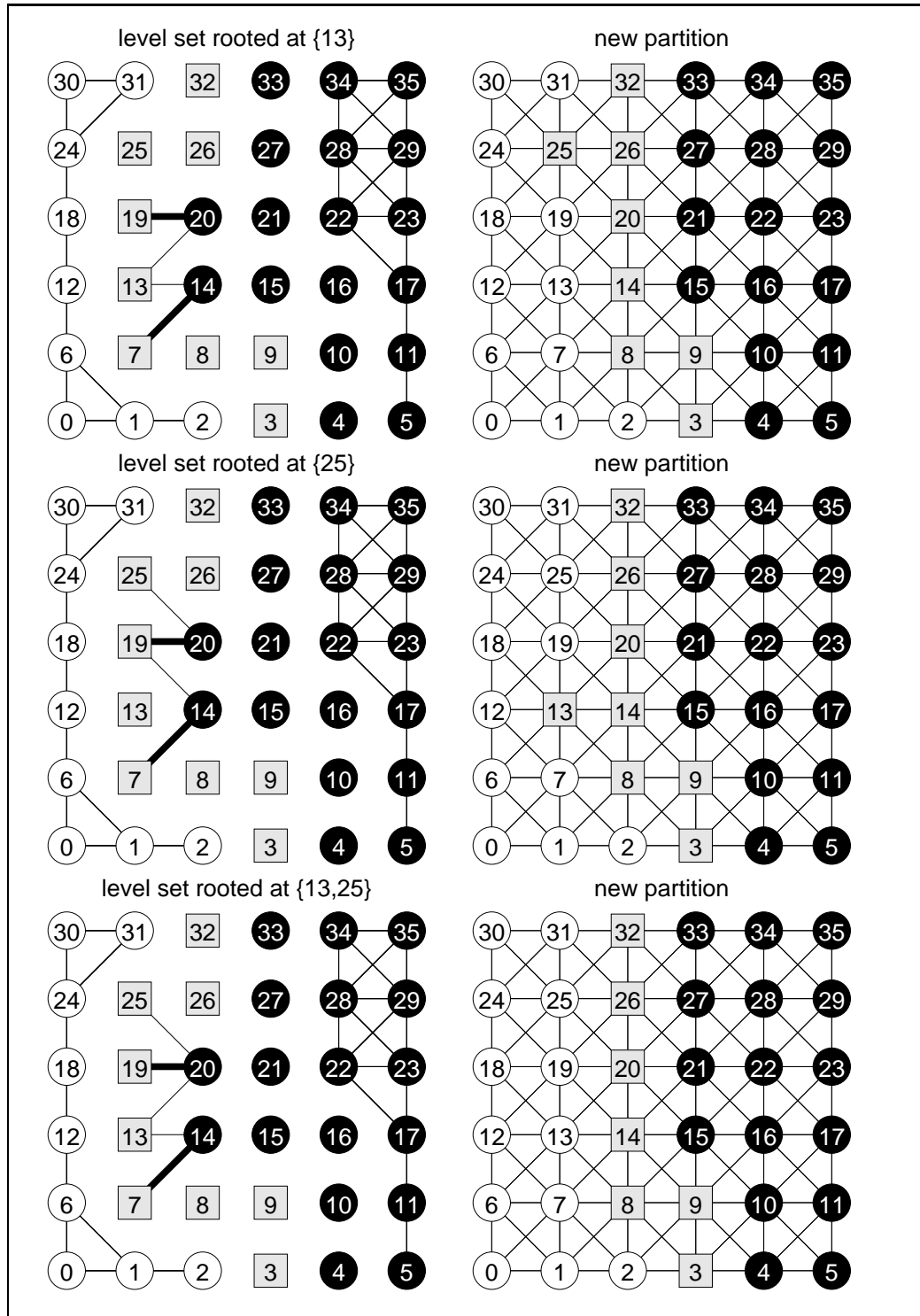
The first improvement to [24] is to use *all* exposed nodes in S to find a subset $Z \subseteq S$ that maximizes the decrease in separator size. It is based on the following extension [25] of the result in Theorem 3.2 for separator-size reduction of greater than one.

THEOREM 3.3 (POTHEN AND FAN [25]). *Define*

$$S_I = \{s \in S \mid s \text{ is reachable from some exposed node in } S \text{ via alternating paths}\}.$$

Then

- $|S_I| - |Adj_H(S_I)| > 0$,
- $|S_I| - |Adj_H(S_I)| = \max_{Z \subseteq S} \{|Z| - |Adj_H(Z)|\}$,
- S_I is the smallest subset of S with this maximum value $|S_I| - |Adj_H(S_I)|$.

FIG. 3. *Alternating Breadth-first Level Structures and their Improved Partitions.*

The subset S_I can be constructed by performing an alternating breadth-first search starting with X_0 , which contains *all* exposed nodes of S .

Theorem 3.2 and Theorem 3.3 provide the end points of a range of separator subsets with the size-improving property. Indeed, consider any subset X_0 of exposed nodes in S . It is easy to verify that the corresponding subset

$$Z = \bigcup \{S_x \mid x \in X_0\}$$

satisfies the condition $|Z| - |Adj_H(Z)| > 0$. This gives a number of choices in selecting a separator-improving subset. Although the subset S_I provides the maximum reduction in separator size, one might opt for a smaller reduction for a better balance in the two components.

3.3 The Dulmage-Mendelsohn Decomposition

In [25], Pothén and Fan relate the subset S_I used in separator size reduction with the Dulmage-Mendelsohn decomposition of bipartite graphs [8]. The decomposition is also useful in our context in finding a balance-improving separator subset. Let $H(S, B)$ be the induced bipartite graph from a given partition $[S, B, W]$. Assume that a maximum matching M is given on H .

The *Dulmage-Mendelsohn decomposition* of S is the decomposition of S into three disjoint subsets: $S = S_I \cup S_R \cup S_X$ where

$$\begin{aligned} S_I &= \{s \in S \mid s \text{ is reachable from some exposed node in } S \text{ via alternating paths}\}, \\ S_X &= \{s \in S \mid s \text{ is reachable from some exposed node in } B \text{ via alternating paths}\}, \\ S_R &= S \setminus (S_I \cup S_X). \end{aligned}$$

Note we use the notation S_I to represent nodes reachable from internal exposed nodes, and S_X from external exposed nodes of S . S_R stands for the remaining nodes. We shall also use the notation $\langle S_I, S_X, S_R \rangle$ to represent the Dulmage-Mendelsohn decomposition of S . We now quote some results on this decomposition relevant to the partition improvement scheme.

THEOREM 3.4 (DULMAGE AND MENDELSON [8]). *The Dulmage-Mendelsohn decomposition $\langle S_I, S_X, S_R \rangle$ of S is independent of the maximum matching used to define the alternating paths.*

THEOREM 3.5 (POTHÉN AND FAN [25]). *The set $S_I \cup S_R$ satisfies the following:*

- $|S_I \cup S_R| - |Adj_H(S_I \cup S_R)| = |S_I| - |Adj_H(S_I)|$,
- $S_I \cup S_R$ is the largest subset of S with the maximum value $\max_{Z \subseteq S} \{|Z| - |Adj_H(Z)|\}$,

Theorem 3.3 states that S_I , if used, is the smallest subset of S with the maximum reduction $|S_I| - |Adj_H(S_I)|$ in separator size. On the other hand, Theorem 3.5 identifies $S_I \cup S_R$ as the largest subset with such maximum reduction in separator size. Moving S_I or $S_I \cup S_R$ will achieve the same amount of size reduction, but the balance for the resulting partition will be better for one or the other of the two moves.

By symmetry, there is a similar Dulmage-Mendelsohn decomposition $\langle B_I, B_X, B_R \rangle$ of B , the other part of the bipartite graph, where

$$\begin{aligned} B_I &= \{b \in B \mid b \text{ is reachable from some exposed node in } B \text{ via alternating paths}\}, \\ B_X &= \{b \in B \mid b \text{ is reachable from some exposed node in } S \text{ via alternating paths}\}, \\ B_R &= B \setminus (B_I \cup B_X). \end{aligned}$$

THEOREM 3.6 (DULMAGE AND MENDELSON [8]). $S_X = Adj_H(B_I)$ and $B_X = Adj_H(S_I)$.

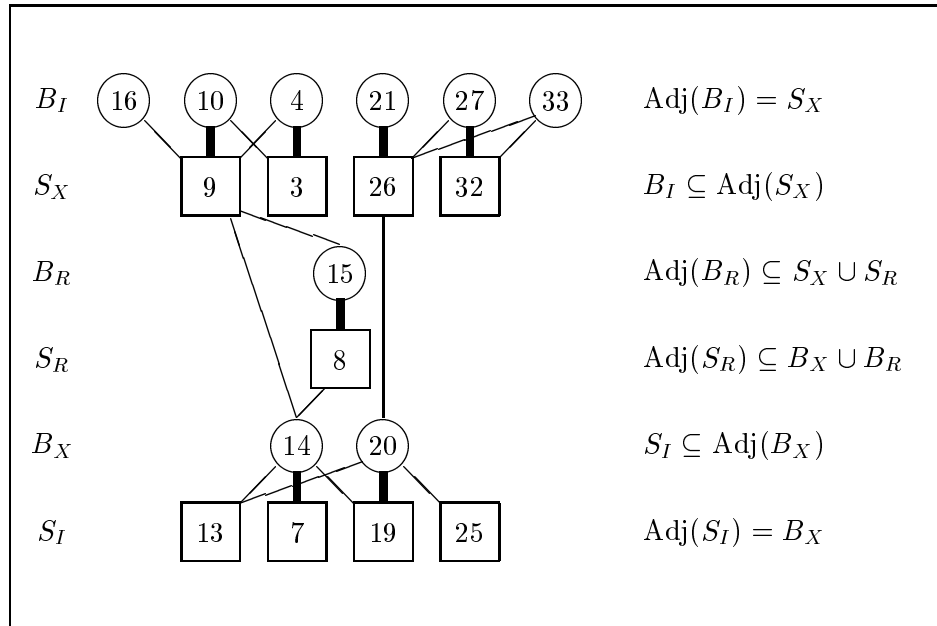
The set S_X is given by the adjacent set of B_I , the set of reachable nodes in B from internal exposed nodes via alternating paths. B_I can be determined in the same way as S_I , by forming the alternating breadth-first search forest from the set of exposed nodes in B .

For the example in Figure 2, the sets of exposed nodes in S and B are $\{13, 25\}$ and $\{16, 33\}$ respectively. This gives:

$$\begin{array}{ll} S_I = \{7, 13, 19, 25\} & B_I = \{4, 10, 16, 21, 27, 33\} \\ S_X = \{3, 9, 26, 32\} & B_X = \{14, 20\} \\ S_R = \{8\} & B_R = \{15\} \end{array}$$

In Figure 4, we illustrate the Dulmage-Mendelsohn decomposition of the bipartite graph H of Figure 2. The six sets are arranged in such a fashion to illustrate their adjacency relationships.

FIG. 4. *Dulmage-Mendelsohn Decomposition*



It is instructive to interpret the decompositions $\langle S_I, S_X, S_R \rangle$ and $\langle B_I, B_X, B_R \rangle$ in connection with our partition improvement objective. For the given separator S , we can extend it to include its adjacent set in the B portion to obtain a *wide* separator $S \cup Border(B)$. The Dulmage-Mendelsohn decomposition provides a machinery whereby a separator can be obtained from this wide separator, such that it is of minimum cover among all separator subsets of $S \cup Border(B)$. Indeed, it is clear that the following are two such separator subsets:

$$S_X \cup S_R \cup B_X, \quad S_X \cup B_R \cup B_X.$$

Either one of them can be used to achieve a maximum reduction in separator size in the new partition.

4 Using the Dulmage-Mendelsohn Decomposition to Improve Balance

4.1 Using the Set S_R

In the discussion in the last section, we are looking for a separator-improving subset Z of S satisfying $|Z| > |Adj(Z) \cap B|$. If no such subset can be found, no reduction in separator size by graph matching is possible. In terms of the Dulmage-Mendelsohn decomposition, this is equivalent to the condition that the current separator S is already of minimum size among covering separator subsets of $S \cup Border(B)$. The algorithm as presented in [24] will terminate if there is no reduction in separator size via graph matching.

However, based on our evaluation function $\gamma(B, W, S)$, it may still be possible to improve the partition by reducing the imbalance ratio $\max\{|B|, |W|\} / \min\{|B|, |W|\}$. We can search for a subset Z of S with $|Adj_H(Z)| = |Z|$. A move of such a subset to the smaller portion W will replace Z by $Adj_H(Z)$ in S so that there will be no change in separator size. However, there may be a reduction in the imbalance.

When S_I is empty (implying that size reduction is not possible by this approach), the subset S_R can be used to reduce the imbalance. We now establish an interesting property of this subset in the next theorem. We first need the following lemma.

LEMMA 4.1. *Let $S_I = \emptyset$. Consider a subset Z of S . If $Z \cap S_X \neq \emptyset$, then $|Z| < |Adj_H(Z)|$.*

Proof. $S_I = \emptyset$ implies that there is a complete matching from S into B . By Theorem 3.1, $|Z| \leq |Adj_H(Z)|$ for every subset Z of S .

Let Z be a subset of S with $Z \cap S_X \neq \emptyset$. Assume for contradiction that $|Z| = |Adj_H(Z)|$. This means $Adj_H(Z)$ is exactly the set of matched vertices of Z for a given maximum matching. Let s be a vertex in $Z \cap S_X$. Then there exists an exposed vertex $b_e \in B$ and an alternating path from b_e to s :

$$(b_e, s_1, b_1, \dots, s_t, b_t, s_{t+1} = s)$$

where each pair $\{s_i, b_i\}$ belongs to the maximum matching. Let m be the smallest index such that $s_m \in Z$. If $m = 1$, this is a contradiction since $b_e \in Adj_H(s_1) \subseteq Adj_H(Z)$ and b_e does not have a mate in Z . For the case $m > 1$, this is again a contradiction since $b_{m-1} \in Adj_H(s_m) \subseteq Adj_H(Z)$ and the mate s_{m-1} of b_{m-1} is not in Z by the choice of m . Therefore, we have $|Z| < |Adj_H(Z)|$. ■

THEOREM 4.1. *Let $S_I = \emptyset$. The separator subset S_R is the largest subset of S such that its size is the same as the size of its adjacent set.*

Proof. By Theorem 3.5, we have

$$|S_I \cup S_R| - |Adj_H(S_I \cup S_R)| = |S_I| - |Adj_H(S_I)|,$$

so that if $S_I = \emptyset$, $|S_R| - |Adj_H(S_R)| = 0$.

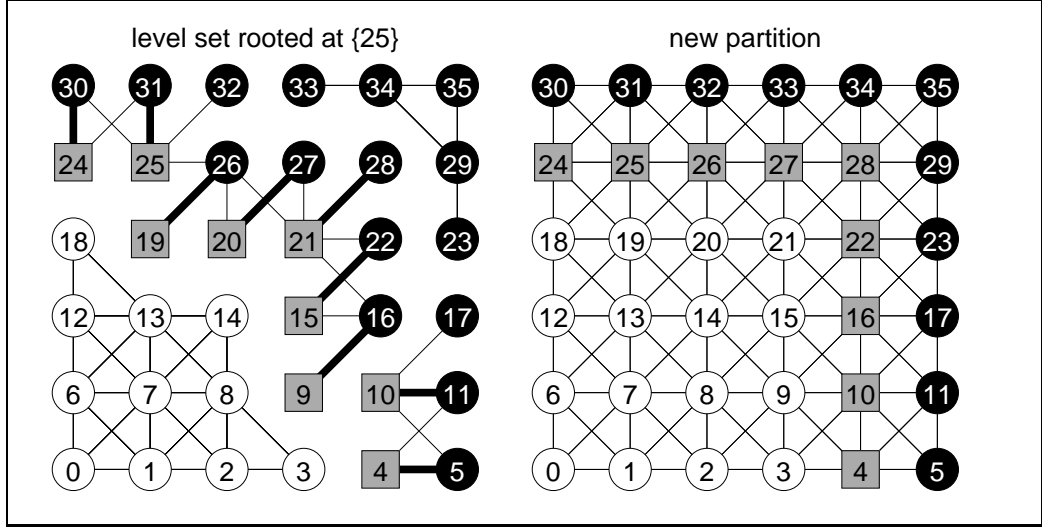
Consider any subset Z of S with the property $|Z| = |Adj_H(Z)|$. By the result of Theorem 4.1, $Z \cap S_X = \emptyset$, which implies $Z \subseteq S_R$. ■

By this result, the subset S_R is the key to find a balance-improving separator subset. We first note from [25] that in general we have $|S_R| = |B_R|$. Furthermore, we have

$$Adj_H(S_I \cup S_R) = B_X \cup B_R,$$

so that when $S_I = \emptyset$, we have $B_X = \emptyset$ and $Adj_H(S_R) = B_R$. Therefore, when the separator subset S_I is empty, the move of S_R to W will give a new separator

$$S_{S_R \rightarrow W} = (S \cup B_R) \setminus S_R,$$

FIG. 5. *Improving the balance.*

so that $|S_{S_R \rightarrow W}| = |(S \cup B_R) \setminus S_R| = |S|$,

Consider the example in Figure 5. There is a complete matching from the set S to B so that in the induced bipartite graph, $S_I = \emptyset$. This implies the separator-improving technique in the last section is not applicable. Note that the Dulmage-Mendelsohn decomposition is given by:

$$\begin{aligned} S_I &= \emptyset & B_I &= \{5, 11, 17, 30, 31, 32\} \\ S_X &= \{4, 10, 24, 25\} & B_X &= \emptyset \\ S_R &= \{9, 15, 19, 20, 21\} & B_R &= \{16, 22, 26, 27, 28\} \end{aligned}$$

For this example, moving the subset S_R from S to W will have the net effect of replacing it by B_R in S . In this way, the new separator will be $\{4, 10, 16, 22, 24, 25, 26, 27, 28\}$, which is the same size as before.

Now consider a separator subset Z with the property $|Z| = |Adj_H(Z)|$. Moving it to the portion W will preserve the separator size. The next result gives a simple necessary and sufficient condition for the move to improve the evaluation function $\gamma[S, B, W]$.

THEOREM 4.2. *Let $[S, B, W]$ be a given partition with $|B| \geq |W|$ and $S_I = \emptyset$. Consider a subset Z with $|Z| = |Adj_H(Z)|$. The move of the subset Z to W will reduce the evaluation function if and only if $|Z| < |B| - |W|$.*

Proof. Let

$$[S, B, W]_{Z \rightarrow W} = [S_{Z \rightarrow W}, B_{Z \rightarrow W}, W_{Z \rightarrow W}]$$

be the new partition after the move of the subset Z from S to W . It is clear that $|S_{Z \rightarrow W}| = |S|$, $|B_{Z \rightarrow W}| = |B| - |Z|$, and $|W_{Z \rightarrow W}| = |W| + |Z|$.

“Case 1”: $|B_{Z \rightarrow W}| \geq |W_{Z \rightarrow W}|$.

$$\begin{aligned} \gamma[S, B, W] - \gamma[S, B, W]_{Z \rightarrow W} &= |S| \left(1 + \alpha \frac{|B|}{|W|} \right) - |S| \left(1 + \alpha \frac{|B| - |Z|}{|W| + |Z|} \right) \\ &= \frac{\alpha |S| |Z| (|B| + |W|)}{|W| (|W| + |Z|)} > 0 \end{aligned}$$

“Case 2”: $|B_{Z \rightarrow W}| < |W_{Z \rightarrow W}|$.

$$\begin{aligned}
\gamma[S, B, W] - \gamma[S, B, W]_{Z \mapsto W} &= |S| \left(1 + \alpha \frac{|B|}{|W|}\right) - |S| \left(1 + \alpha \frac{|W| + |Z|}{|B| - |Z|}\right) \\
&= \frac{\alpha |S| |Z| (|B| + |W|)(|B| - |W| - |Z|)}{|W|(|B| - |Z|)}
\end{aligned}$$

Assume $|Z| < |B| - |W|$. The evaluation function will be reduced in case 1. Moreover, in case 2, we have $|B| - |W| - |Z| > 0$ so that $\gamma[S, B, W] - \gamma[S, B, W]_{Z \mapsto W} > 0$.

On the other hand, assume that $\gamma[S, B, W] - \gamma[S, B, W]_{Z \mapsto W} > 0$. In case 1, we have $|B| - |Z| > |W| + |Z|$, which implies that

$$|B| - |W| > 2|Z| > |Z|.$$

Furthermore, in case 2, a reduction in the evaluation function implies that $|B| - |W| - |Z| > 0$ or $|Z| < |B| - |W|$. ■

By Theorem 4.1 and Theorem 4.2, to improve the balance of a given partition, we should be looking for a subset Z of S_R such that $|Z| = |Adj_H(Z)| < |B| - |W|$. Of course, if $|S_R| < |B| - |W|$, this set S_R is a good choice. Otherwise, we need to find proper subsets of S_R .

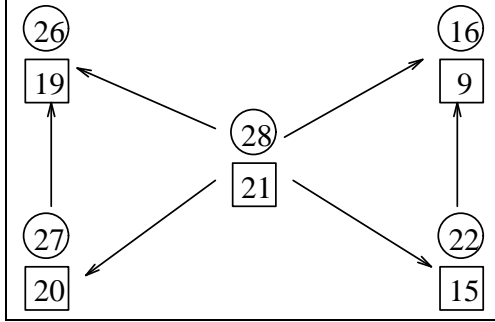
4.2 Finding Balance-Improving Subsets of S_R

Finding a subset Z with $|Z| = |Adj_H(Z)|$ is related to the problem of reordering a sparse square matrix to block lower triangular form. In [25], Pothen and Fan provide an algorithm to compute the block triangular form of a sparse matrix. In their “fine decomposition” step, the square submatrix associated with the vertices in S_R and B_R are further reordered into block lower triangular form. (Pothen and Fan actually compute a block upper triangular form; but the algorithm can be adapted for block lower triangular form.) Their approach involves the following substeps:

- Form a directed graph based on the bipartite subgraph of S_R and B_R . The directed graph consists of nodes from S_R . For two nodes x and y in S_R , there is a directed edge from x to y in this new directed graph if and only if there is an edge from x to the mate of y in B_R .
- Determine the strongly connected components of this directed graph. (The quotient graph using the strongly connected components forms a directed acyclic graph or in short, a dag).
- Order the strongly connected components of this directed graph by a *reverse topological ordering* (i.e. an ordering of the nodes in the directed graph so that *all* the directed edges are pointing backwards to the left).

The reverse topological ordering of the strongly connected components of this directed graph will induce an ordering of the vertices in S_R and B_R so that the bipartite graph with this new reordering has a block lower triangular form. It should be clear from the block lower triangular structure that any subset Z of nodes of S_R corresponding to the leading blocks in the triangular form has this desirable property $|Z| = |Adj_H(Z)|$.

It is instructive to apply this scheme to the example of Figure 5. The new directed graph formed will consist of nodes from $S_R = \{9, 15, 19, 20, 21\}$. Figure 6 shows the directed graph; in each vertex of this directed graph, we label it with both the node in S_R and its

FIG. 6. *Induced Directed Graph.*

mate in B . There is no cycle in this directed graph, so that each node forms a strongly connected component. Furthermore, the following is a reverse topological ordering:

$$9, 15, 19, 20, 21$$

and the corresponding matrix is lower triangular:

$$\begin{array}{c}
 9 \\
 15 \\
 19 \\
 20 \\
 21
 \end{array}
 \begin{array}{ccccc}
 & 16 & 22 & 26 & 27 & 28 \\
 \left(\begin{array}{cccccc}
 \bullet & & & & & \\
 \bullet & \bullet & & & & \\
 & & \bullet & & & \\
 & & \bullet & \bullet & & \\
 \bullet & \bullet & \bullet & \bullet & \bullet &
 \end{array} \right)
 \end{array}$$

We can then deduce from this reverse topological ordering that all of the following subsets have the property $|Z| = |Adj_H(Z)|$:

$$\{9\}, \{9, 15\}, \{9, 15, 19\}, \{9, 15, 19, 20\}, \{9, 15, 19, 20, 21\}.$$

It is interesting to note that there are different reverse topological orderings of this directed graph. They will provide additional such subsets. For example, $\{19, 20, 9, 15, 21\}$ is a different reverse topological ordering, and the subsets $\{19\}, \{19, 20\}, \{19, 20, 9\}$ also have the size-preserving property.

5 Partition Improvement on Compressed Graphs

5.1 Compressed Graphs

The Dulmage-Mendelsohn decomposition is the basic tool used in the last two sections to improve a given 2-set partition. In this section, we explore efficient ways of computing this decomposition for some practical classes of matrix problems. It is common for graphs from applications to have sets of vertices with identical adjacency structures, e.g., in a finite element graph, a given geometric location may have multiple displacements and rotations. Such vertex pairs are sometimes referred to as *indistinguishable* in the sparse matrix research community. More formally, two vertices x and y are said to be *indistinguishable* if

$$Adj(x) \cup \{x\} = Adj(y) \cup \{y\}.$$

The notion of *compressed graph* is introduced in [1], [6], where each vertex of the compressed graph corresponds to (possibly) several indistinguishable vertices in the original

graph. A compressed graph can be viewed as a *quotient graph* of the original unit-weight graph consisting of weighted compressed vertices. The motivation in [1] is for efficient implementations of some sparse matrix ordering algorithms. The number of vertices in a compressed graph can be several times smaller than that of the original unit-weight graph. Since most graph algorithms have a strong $O(|V|)$ or $O(|E|)$ component to their complexity, it would be quite beneficial to work with a compressed graph instead of the original graph.

Following [1], we use boldface $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ to represent a compressed graph. A boldface \mathbf{v} is used to denote a compressed vertex in \mathbf{V} with an indistinguishable set of original vertices. For a given compressed graph, it is helpful to define its associated *compression* to be a mapping $\kappa : V \rightarrow \mathbf{V}$, where $\kappa(v)$ is the compressed vertex in \mathbf{V} containing the vertex v . This means $\kappa(v)$ is a subset of vertices in V (containing v), that are indistinguishable from v in G . However we do not require $\kappa(v)$ to include *all* possible indistinguishable vertices of v . An edge (\mathbf{u}, \mathbf{v}) is in the compressed edge set \mathbf{E} if $(\mathbf{u} \times \mathbf{v}) \cap E \neq \emptyset$. The theory and algorithm to be developed apply to any level of compression (partial or complete). Note that this is a *loss-less* representation; that is, given \mathbf{E} and κ , we can always recover the original edge set E .

We also extend the usage of κ to subsets: for a subset $Y \subseteq V$, $\kappa(Y) = \{\kappa(y) \mid y \in Y\} \subseteq \mathbf{V}$. For a compressed vertex $\mathbf{v} \in \mathbf{V}$, define its *weight* $wt(\mathbf{v})$ to be the number of indistinguishable vertices contained in \mathbf{v} . This notion can be extended to the weight of a subset of compressed vertices: for a subset \mathbf{Y} of \mathbf{V} ,

$$wt(\mathbf{Y}) = \sum \{wt(\mathbf{y}) \mid \mathbf{y} \in \mathbf{Y}\}.$$

We now consider the partition improvement techniques of the last two sections in the context of compressed graphs.

5.2 Definitions for Bipartite Compressed Graphs

Let \mathbf{G} be a given compressed graph with compression κ and a two-set partition $[\mathbf{S}, \mathbf{B}, \mathbf{W}]$; that is, $Adj(\mathbf{B}) \subseteq \mathbf{S}$ and $Adj(\mathbf{W}) \subseteq \mathbf{S}$. We first make a connection of this compressed partition with a partition on the original graph G .

THEOREM 5.1. *There is a unique two-set partition $[\overline{\mathbf{S}}, \overline{\mathbf{B}}, \overline{\mathbf{W}}]$ on G such that $\kappa(\overline{\mathbf{S}}) = \mathbf{S}$, $\kappa(\overline{\mathbf{B}}) = \mathbf{B}$, and $\kappa(\overline{\mathbf{W}}) = \mathbf{W}$.*

Proof. It is clear that the subsets defined by:

$$\overline{\mathbf{S}} = \{v \in V \mid \kappa(v) \in \mathbf{S}\}, \quad \overline{\mathbf{B}} = \{v \in V \mid \kappa(v) \in \mathbf{B}\}, \quad \overline{\mathbf{W}} = \{v \in V \mid \kappa(v) \in \mathbf{W}\}$$

satisfy the conditions $\kappa(\overline{\mathbf{S}}) = \mathbf{S}$, $\kappa(\overline{\mathbf{B}}) = \mathbf{B}$, and $\kappa(\overline{\mathbf{W}}) = \mathbf{W}$. To prove that they form a 2-set partition on G , it suffices to show that $Adj(\overline{\mathbf{B}}) \subseteq \overline{\mathbf{S}}$ (by symmetry, we have $Adj(\overline{\mathbf{W}}) \subseteq \overline{\mathbf{S}}$). This is the case since otherwise, it would have implied that \mathbf{W} and \mathbf{B} are adjacent contradicting the fact that \mathbf{S} separates them. The uniqueness follows from the fact that $[\mathbf{S}, \mathbf{B}, \mathbf{W}]$ is a partition on \mathbf{G} . ■

The simple connection in Theorem 5.1 allows the partition improvement techniques described in Sections 3 and 4 using the Dulmage-Mendelsohn decomposition to be applied to the induced partition $[\overline{\mathbf{S}}, \overline{\mathbf{B}}, \overline{\mathbf{W}}]$ of the unit-weight graph. We now show that the decomposition of the original graph can be readily obtained from a similar decomposition of the compressed graph. We first extend the various notions used in the formulation of the Dulmage-Mendelsohn decomposition from unit-weight graphs to compressed graphs.

5.2.1 Compressed matching and maximum matching As before, let $[\mathbf{S}, \mathbf{B}, \mathbf{W}]$ be a given partition on the compressed graph \mathbf{G} . This will in turn define a bipartite compressed graph $\mathbf{H}(\mathbf{S}, \text{Border}(\mathbf{B}))$. We extend the notion of matching to bipartite compressed graphs. In the unit-weight case, a matching can be viewed as an assignment of an integer value $f(s, b)$ to each edge (s, b) in the bipartite graph such that

- for every edge (s, b) , $f(s, b) \geq 0$;
- for every vertex $\tilde{s} \in S$, $1 \geq \sum\{f(\tilde{s}, b) \mid b \in B\}$;
- for every vertex $\tilde{b} \in B$, $1 \geq \sum\{f(s, \tilde{b}) \mid s \in S\}$.

It follows then the assigned values can either be 0 (unmatched) or 1 (matched). Furthermore, a maximum matching is one that will maximize the sum $\sum\{f(s, b) \mid s \in S, b \in B\}$, which is the number of edges in the matching.

With this interpretation, we generalize a compressed matching of a bipartite compressed graph to be an assignment of integer values $f(\mathbf{s}, \mathbf{b})$ to the edges such that they satisfy the following three conditions.

- for every edge (\mathbf{s}, \mathbf{b}) , $f(\mathbf{s}, \mathbf{b}) \geq 0$;
- for every compressed vertex $\tilde{\mathbf{s}} \in \mathbf{S}$, $wt(\tilde{\mathbf{s}}) \geq \sum\{f(\tilde{\mathbf{s}}, \mathbf{b}) \mid \mathbf{b} \in \mathbf{B}\}$;
- for every compressed vertex $\tilde{\mathbf{b}} \in \mathbf{B}$, $wt(\tilde{\mathbf{b}}) \geq \sum\{f(\mathbf{s}, \tilde{\mathbf{b}}) \mid \mathbf{s} \in \mathbf{S}\}$;

Note that we have used the weight of each vertex instead of unit weight in the above bounds on the edge value sums. Similarly, a maximum compressed matching is one that maximizes the total edge value:

$$\sum\{f(\mathbf{s}, \mathbf{b}) \mid \mathbf{s} \in \mathbf{S}, \mathbf{b} \in \mathbf{B}\}.$$

Compressed exposed nodes and alternating paths

In the unit-weight bipartite graph, an exposed node s_e is one such that none of its incident edges belong to the matching. In terms of the value $f(s, b)$, this is equivalent to the condition that for the node s_e ,

$$1 > \sum\{f(s_e, b) \mid b \in B\} = 0.$$

In a compressed bipartite graph, we define an exposed node $\mathbf{s}_e \in \mathbf{S}$ to be a node such that

$$wt(\mathbf{s}_e) > \sum\{f(\mathbf{s}_e, \mathbf{b}) \mid \mathbf{b} \in \mathbf{B}\}.$$

The *exposure* of \mathbf{s} is defined to be $wt(\mathbf{s}) - \sum\{f(\mathbf{s}, \mathbf{b}) \mid \mathbf{b} \in \mathbf{B}\}$. Therefore, the exposure of an exposed node is positive. Exposed nodes and exposure are similarly defined for compressed vertices in \mathbf{B} .

For a given compressed matching \mathbf{M} of the compressed bipartite graph, consider a path:

$$\mathbf{s}_0 \longrightarrow \mathbf{b}_1 \longrightarrow \mathbf{s}_1 \longrightarrow \mathbf{b}_2 \longrightarrow \mathbf{s}_2 \longrightarrow \dots \mathbf{b}_m \longrightarrow \mathbf{s}_m \longrightarrow \dots$$

It is an compressed alternating path with respect to \mathbf{M} if the alternate edges

$$(\mathbf{b}_1, \mathbf{s}_1), (\mathbf{b}_2, \mathbf{s}_2), \dots (\mathbf{b}_m, \mathbf{s}_m), \dots$$

all have positive edge values from the matching \mathbf{M} . In such a case, we use the following to represent an alternating path:

$$\mathbf{s}_0 \longrightarrow \mathbf{b}_1 \Longrightarrow \mathbf{s}_1 \longrightarrow \mathbf{b}_2 \Longrightarrow \mathbf{s}_2 \longrightarrow \dots \mathbf{b}_m \Longrightarrow \mathbf{s}_m \longrightarrow \dots$$

where a double-lined arrow is used to indicated an edge with positive edge value. An alternating path that starts with a compressed node from \mathbf{B} is similarly defined.

5.3 Decomposition in Bipartite Compressed Graphs

Recall that our objective is to improve a partition using the Dulmage-Mendelsohn decomposition; and we want to take advantage of compression in finding such decomposition. As before, let $[\mathbf{S}, \mathbf{B}, \mathbf{W}]$ be a given partition on the compressed graph \mathbf{G} and $\mathbf{H}(\mathbf{S}, \text{Border}(\mathbf{B}))$ be the corresponding bipartite compressed graph. Furthermore, let \mathbf{M} be a maximum compressed matching on \mathbf{H} . Consider the decomposition $\mathbf{S}_I \cup \mathbf{S}_R \cup \mathbf{S}_X$ of \mathbf{S} where

$$\begin{aligned} \mathbf{S}_I &= \{\mathbf{s} \in \mathbf{S} \mid \mathbf{s} \text{ is reachable from some } \mathbf{exposed} \text{ node in } \mathbf{S} \text{ via } \mathbf{alternating} \text{ paths}\}, \\ \mathbf{S}_X &= \{\mathbf{s} \in \mathbf{S} \mid \mathbf{s} \text{ is reachable from some } \mathbf{exposed} \text{ node in } \mathbf{B} \text{ via } \mathbf{alternating} \text{ paths}\}, \\ \mathbf{S}_R &= \mathbf{S} \setminus (\mathbf{S}_I \cup \mathbf{S}_X). \end{aligned}$$

Note that we have used the boldface **exposed** and **alternating** in the above decomposition to emphasize the use of the extended definitions of compressed exposed nodes and compressed alternating paths for compressed bipartite graphs. The decomposition $\mathbf{B}_I \cup \mathbf{B}_R \cup \mathbf{B}_X$ of \mathbf{B} can be similarly defined.

We now make the connection of this decomposition with the Dulmage-Mendelsohn decomposition in the unit-weight graph. Consider the unique partition $[\overline{\mathbf{S}}, \overline{\mathbf{B}}, \overline{\mathbf{W}}]$ of the unit-weight graph G satisfying $\kappa(\overline{\mathbf{S}}) = \mathbf{S}$, $\kappa(\overline{\mathbf{B}}) = \mathbf{B}$, and $\kappa(\overline{\mathbf{W}}) = \mathbf{W}$ in Theorem 5.1. This partition will in turn determine a unit-weight bipartite graph $\overline{H}(\overline{\mathbf{S}}, \overline{\mathbf{B}})$. We now relate the decomposition $\mathbf{S}_I \cup \mathbf{S}_R \cup \mathbf{S}_X$ with the Dulmage-Mendelsohn decomposition of $\overline{\mathbf{S}}$ in this $\overline{H}(\overline{\mathbf{S}}, \overline{\mathbf{B}})$. Note that the Dulmage-Mendelsohn decomposition $\langle \overline{\mathbf{S}}_I, \overline{\mathbf{S}}_X, \overline{\mathbf{S}}_R \rangle$ of $\overline{\mathbf{S}}$ is independent of any maximum matching used in \overline{H} . But in order to make the connection between the two decompositions, we need to define an induced matching of \overline{H} from \mathbf{H} .

Let \mathbf{M} be a compressed matching on $\mathbf{H}(\mathbf{S}, \text{Border}(\mathbf{B}))$. An induced matching $\overline{\mathbf{M}}$ on the unit-weight bipartite graph $\overline{H}(\overline{\mathbf{S}}, \overline{\mathbf{B}})$ can be defined as follows. For each compressed vertex \mathbf{s} , we have the size condition:

$$wt(\mathbf{s}) \geq \sum \{f(\mathbf{s}, \mathbf{b}) \mid \mathbf{b} \in \mathbf{B}\}.$$

Therefore for each incident edge (\mathbf{s}, \mathbf{b}) , we can always assign $f(\mathbf{s}, \mathbf{b})$ number of distinct $\overline{\mathbf{S}}$ -vertices from \mathbf{s} to this compressed edge. Similar allotments of $\overline{\mathbf{B}}$ -vertices from compressed vertices of \mathbf{B} to compressed incident edges can be assigned.

Now for each compressed edge (\mathbf{s}, \mathbf{b}) with value $f(\mathbf{s}, \mathbf{b})$ in the matching \mathbf{M} , there will be $f(\mathbf{s}, \mathbf{b})$ number of $\overline{\mathbf{S}}$ -vertices from \mathbf{s} and the same number of $\overline{\mathbf{B}}$ -vertices from \mathbf{b} assigned to this edge. Since each $\overline{\mathbf{S}}$ -vertex in \mathbf{s} is adjacent to each $\overline{\mathbf{B}}$ -vertex in \mathbf{b} , we can get $f(\mathbf{s}, \mathbf{b})$ different edges consisting of pairs of adjacent assigned vertices from \mathbf{s} and \mathbf{b} . We place them in the set $\overline{\mathbf{M}}$.

After doing this for every compressed edge, we see that the set of edges in $\overline{\mathbf{M}}$, by construction, does not have common vertices. This means that the set $\overline{\mathbf{M}}$ forms a matching. Furthermore, this set $\overline{\mathbf{M}}$ satisfies the following property.

LEMMA 5.1. *Given \mathbf{M} is a maximum matching on the compressed bipartite graph \mathbf{H} , $\overline{\mathbf{M}}$ is a maximum matching on \overline{H} .*

Proof. Assume for contradiction that $\overline{\mathbf{M}}$ is not maximum. We can therefore find an alternating path connecting two exposed vertices, say $\overline{\mathbf{s}} \in \overline{\mathbf{S}}$ and $\overline{\mathbf{b}} \in \overline{\mathbf{B}}$ (such paths are usually referred to as an augmenting path):

$$\overline{\mathbf{s}} \longrightarrow \overline{\mathbf{b}}_1 \implies \overline{\mathbf{s}}_1 \longrightarrow \dots \overline{\mathbf{b}}_t \implies \overline{\mathbf{s}}_t \longrightarrow \overline{\mathbf{b}}.$$

Through compression, this corresponds to a path or walk in the compressed graph:

$$\kappa(\bar{s}) \longrightarrow \kappa(\bar{b}_1) \implies \kappa(\bar{s}_1) \longrightarrow \dots \longrightarrow \kappa(\bar{b}_t) \implies \kappa(\bar{s}_t) \longrightarrow \kappa(\bar{b}).$$

Since $\bar{s} \in \kappa(\bar{s})$ and $\bar{b} \in \kappa(\bar{b})$ are both exposed in \bar{M} , we can increase the total matching in \mathbf{M} by at least one by alternately increasing and decreasing the f values along this path. This contradicts the fact that \mathbf{M} is a maximum matching on \mathbf{H} . ■

By this lemma, \bar{M} is a maximum matching on \bar{H} so that the Dulmage-Mendelsohn decomposition $\langle \bar{S}_I, \bar{S}_X, \bar{S}_R \rangle$ of \bar{S} can be determined using \bar{M} .

THEOREM 5.2. $\kappa(\bar{S}_I) = \mathbf{S}_I$, $\kappa(\bar{S}_R) = \mathbf{S}_R$, $\kappa(\bar{S}_X) = \mathbf{S}_X$.

Proof. We only prove $\kappa(\bar{S}_I) = \mathbf{S}_I$ and leave the remaining two for the readers. We first show $\kappa(\bar{S}_I) \subseteq \mathbf{S}_I$. Consider a compressed vertex $\kappa(\bar{s}) \in \kappa(\bar{S}_I)$, with $\bar{s} \in \bar{S}_I$. This means that there exists an alternating path:

$$s_0 \longrightarrow b_1 \implies s_1 \longrightarrow \dots \longrightarrow b_t \implies s_t = \bar{s}$$

from some exposed node $s_0 \in \bar{S}$. This induces a compressed alternating path or walk in \mathbf{H} :

$$\kappa(s_0) \longrightarrow \kappa(b_1) \implies \kappa(s_1) \longrightarrow \dots \longrightarrow \kappa(b_t) \implies \kappa(s_t) = \kappa(\bar{s})$$

and $\kappa(s_0)$ is exposed in \mathbf{S} . Therefore $\kappa(\bar{s}) \in \mathbf{S}_I$.

We now show $\mathbf{S}_I \subseteq \kappa(\bar{S}_I)$. Consider a compressed node $\mathbf{s} \in \mathbf{S}_I$. There exists an alternating path in \mathbf{H} :

$$\mathbf{s}_0 \longrightarrow \mathbf{b}_1 \implies \mathbf{s}_1 \longrightarrow \dots \longrightarrow \mathbf{b}_t \implies \mathbf{s}_t = \mathbf{s},$$

from an exposed \mathbf{s}_0 . Choose a $\bar{s}_0 \in \mathbf{s}_0$, that is exposed in \bar{M} . For $0 < i < t$, choose a matched edge (\bar{b}_i, \bar{s}_i) in \bar{M} where $\kappa(\bar{b}_i) = \mathbf{b}_i$ and $\kappa(\bar{s}_i) = \mathbf{s}_i$; one is guaranteed since $f(\mathbf{b}_i, \mathbf{s}_i) > 0$. This forms an alternating path in \bar{M} from an exposed node \bar{s}_0 to \bar{s}_t ; therefore $\bar{s}_t \in \bar{S}_I$. The result follows since $\kappa(\bar{s}_t) = \mathbf{s}$. ■

The next two theorems follow directly from the results in Theorem 5.2. The first theorem states that the weight of \mathbf{S}_I is less than that of \mathbf{B}_X so that the partition $[\mathbf{S}, \mathbf{B}, \mathbf{W}]$ can be improved by replacing the subset \mathbf{S}_I with its adjacent set \mathbf{B}_X . The second theorem relates the weights of the two subsets \mathbf{S}_R and \mathbf{B}_R .

THEOREM 5.3. *If \mathbf{S}_I is non-empty, then $wt(\mathbf{S}_I) < wt(\mathbf{B}_X)$.*

THEOREM 5.4. $wt(\mathbf{S}_R) = wt(\mathbf{B}_R)$.

6 Partition Improvement by Maximum Network Flow

6.1 Bipartite Compressed Graph Matching by Maximum Network Flow

Finding an assignment of edge values to a bipartite compressed graph that corresponds to a maximum matching can be reformulated into a much-studied combinatorial problem involving flow through a network [9], [11], [12], [18], [27]. A *network* is a weighted directed graph with two special nodes: one with no incoming edges (the *source*), one with no outgoing edges (the *sink*). There are a set of capacity constraints given to the edges and vertices. Most discussions of the network flow problem in the literature assume the use of edge capacities. The generalization to include both edge and vertex capacity is well known, (for example, [22, pages 120-121]). For our purposes, we only need to consider networks with

finite vertex capacities, i.e., each vertex y is given a non-negative integer value $capacity(y)$, called the *capacity* of the vertex. All edges have infinite capacity.

A *flow* is a function that assigns a non-negative integer value $flow(y, z)$ to each directed edge (y, z) . The flow satisfies two conditions:

- the amount of *in-flow* equals the amount of *out-flow* at each vertex except the source and sink;
- the in-flow must be within capacity of each vertex.

Let $inflow(y)$ denote the amount of in-flow into the vertex y , that is,

$$inflow(y) = \sum \{flow(v, y) \mid v \in V\}$$

and let $outflow(y)$ be the amount of out-flow from the vertex y ,

$$outflow(y) = \sum \{flow(y, v) \mid v \in V\}.$$

For every vertex aside from the source and sink, the flow function satisfies

$$inflow(y) = outflow(y) \leq capacity(y).$$

We shall also refer to $inflow(y)$ (or equivalently $outflow(y)$) as the flow across the vertex y . A vertex y is said to be *saturated* or at capacity if $inflow(y) = capacity(y)$; otherwise, it is said to be below capacity or to have excess capacity. By convention the source and sink have infinite capacity.

The value of the flow is the amount of out-flow from the source node, $outflow(\mathbf{source})$ ², or equivalently, the amount of in-flow into the sink node, $inflow(\mathbf{sink})$. The *network flow problem* is to find a flow with the maximum value for a given network. It should be emphasized that we consider only integer capacity and flow values.

We now describe a network flow problem that when solved will give a solution to the maximum matching problem for bipartite compressed graphs. As before, let $\mathbf{H}(\mathbf{S}, \mathit{Border}(\mathbf{B}))$ be our bipartite compressed graph with weight function $wt(*)$. A bipartite network is constructed as follows.

- In addition to the **source** and **sink**, the nodes in the network are the vertices in \mathbf{S} and $\mathit{Border}(\mathbf{B})$.
- For each vertex $\mathbf{s} \in \mathbf{S}$, add the directed edge $(\mathbf{source}, \mathbf{s})$ to the network.
- For each vertex $\mathbf{b} \in \mathit{Border}(\mathbf{B})$, add the directed edge $(\mathbf{b}, \mathbf{sink})$ to the network.
- For each edge (\mathbf{s}, \mathbf{b}) , $\mathbf{s} \in \mathbf{S}$, $\mathbf{b} \in \mathit{Border}(\mathbf{B})$, in the graph $\mathbf{H}(\mathbf{S}, \mathit{Border}(\mathbf{B}))$, add a directed edge (\mathbf{s}, \mathbf{b}) to the network, where flow is assumed to go from \mathbf{s} to \mathbf{b} along this edge.
- All edges have infinite capacity. For each vertex \mathbf{y} in $\mathbf{H}(\mathbf{S}, \mathit{Border}(\mathbf{B}))$, we set $capacity(\mathbf{y}) = wt(\mathbf{y})$.

²We use boldface for **source** and **sink** to emphasize that we are working on the weighted compressed graph.

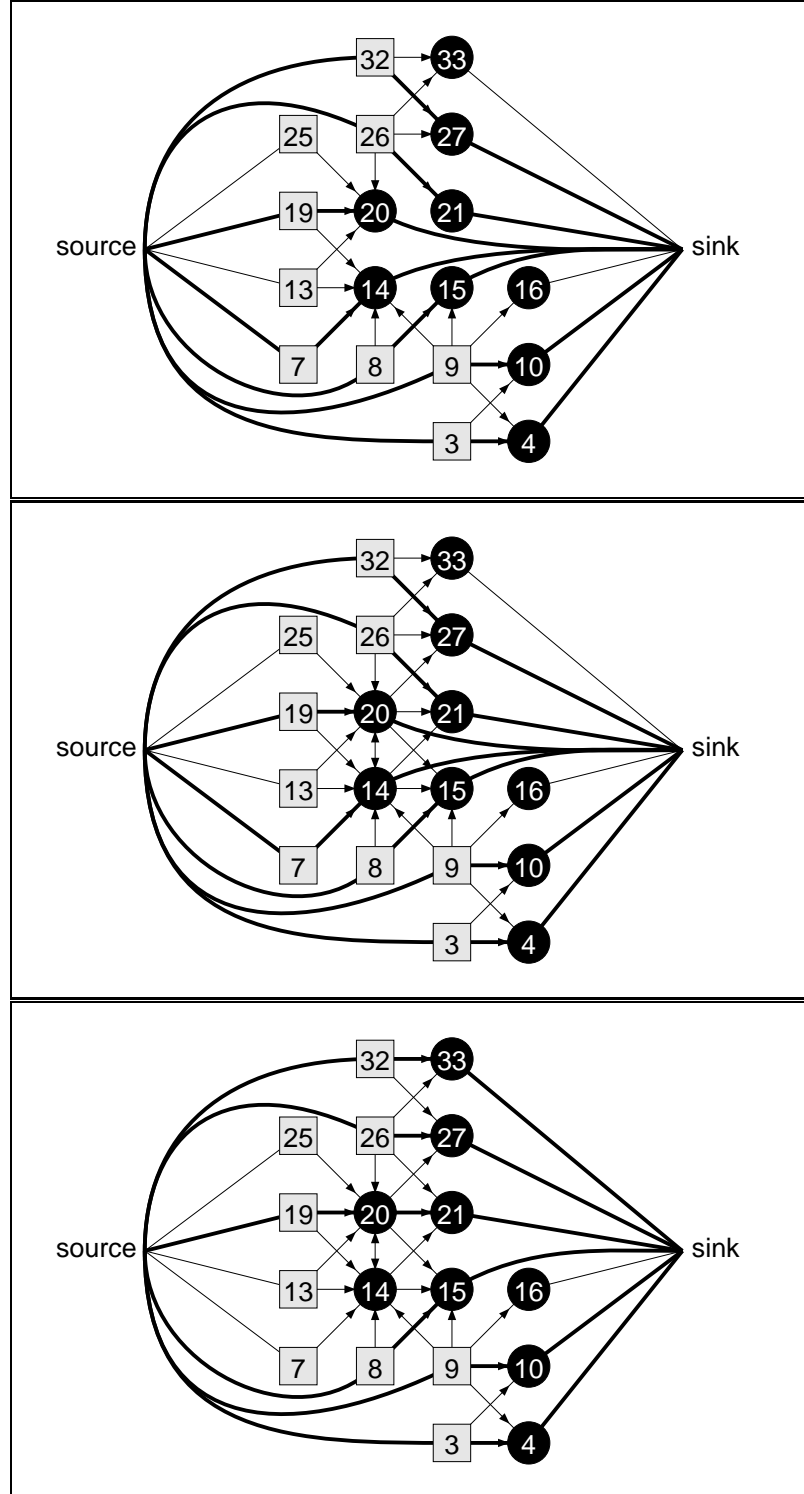


FIG. 7. Top: \mathcal{N}_b , the original bipartite network used to find the Dulmage-Mendelsohn decomposition; Middle: \mathcal{N}_m , the intermediate network found by adding edges that do not increase the \max flow; Bottom: \mathcal{N}_w , the final three-layer network found by deleting edges from the middle layer vertices to the sink.

In the top network of Figure 7 we illustrate the bipartite network obtained for the separator example of Figure 2. Arrows are used on edges to indicate the direction of flow (except for those involving the **source** and the **sink**). Edges with positive (zero) flow are thick (thin) lines. Note that there is a directed edge from the source to every vertex in \mathbf{S} (the set of “square” vertices) and one from every vertex in $\text{Border}(\mathbf{B})$ (the set of “circle” vertices) to the sink.

We shall use the notation \mathcal{N}_b (b for bipartite) to represent this bipartite network. To establish the equivalence between a max-flow solution on this bipartite network with a maximum matching on the bipartite compressed graph, we use the equivalence of *flow augmenting paths* in the former with *augmenting paths* in the latter. An augmenting path in the bipartite compressed graph is an alternating path whose first and last vertices are exposed in \mathbf{S} .

It is simple to generalize such augmenting paths for bipartite network flows. Indeed, a flow augmenting path for a bipartite network is a sequence of edges from the source to the sink with alternate *forward* and *backward* edges:

$$\mathbf{source} \longrightarrow \mathbf{v}_1 \longrightarrow \mathbf{v}_2 \longleftarrow \mathbf{v}_3 \longrightarrow \mathbf{v}_4 \longleftarrow \dots \mathbf{v}_k \longrightarrow \mathbf{sink}.$$

Furthermore, each backward edge $(\mathbf{v}_{2j+1}, \mathbf{v}_{2j})$ has positive flow and the vertices \mathbf{v}_1 and \mathbf{v}_k are below capacity. It is easy to relate this with an augmenting path in the original bipartite compressed graph. Since \mathbf{v}_1 and \mathbf{v}_k are below capacity, they are exposed in the graph matching. Any backward edge with positive flow means the two incident vertices are matched.

Since we will be considering flows on a general network, we must further generalize the notion of a flow augmenting path. When edges have finite capacity, a flow augmenting path is a path from the source to the sink such that forward edges are below capacity and backward edges have positive flow. In our networks the edges have infinite capacity and the vertices have finite capacity, so a flow augmenting path is a sequence of vertices $(\mathbf{source} = \mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_k, \mathbf{v}_{k+1} = \mathbf{sink})$ with these four properties.

- Two consecutive vertices \mathbf{v}_i and \mathbf{v}_{i+1} are connected by an edge in the network — a forward edge is of the form $(\mathbf{v}_i, \mathbf{v}_{i+1})$, a backward edge is of the form $(\mathbf{v}_{i+1}, \mathbf{v}_i)$.
- Any two consecutive forward edges $(\mathbf{v}_{i-1}, \mathbf{v}_i)$ and $(\mathbf{v}_i, \mathbf{v}_{i+1})$ implies vertex \mathbf{v}_i is below capacity.
- Any backward edge $(\mathbf{v}_{i+1}, \mathbf{v}_i)$ has nonzero flow, i.e., $\text{flow}(\mathbf{v}_{i+1}, \mathbf{v}_i) > 0$.
- A vertex may appear in the path once or twice, via a forward edge, a backward edge, or both³.

The overall flow value can be increased by increasing flow along the forward edges and decreasing flow along the backward edges.

6.2 Min-cut in Network Flow

The dual to the network max-flow is a *min-cut*. In our context of networks with finite vertex capacities and infinite edge capacities, a *cut* is a set of vertices whose removal separates the

³Technically speaking, if a vertex is visited twice we have a flow augmenting *walk*. Had we taken the more conventional route of handling vertex capacities by expanding a vertex \mathbf{v} into a pair of vertices connected by an edge $(\mathbf{v}^-, \mathbf{v}^+)$ whose capacity is the weight of the vertex, then \mathbf{v}^- would be visited by a forward edge, \mathbf{v}^+ would be visited by a forward or a backward edge, and there would be no repeated vertices along the path.

source from the **sink**, i.e., a separator of the graph from which the network was derived. A *min-cut* is a cut such that its size

$$\sum \{capacity(v) \mid v \text{ belongs to the cut}\}$$

is minimum among all cuts. The well-known max-flow min-cut theorem states that the size of a min-cut is the same as the value of a max-flow.

It is interesting to relate min-cuts with the Dulmage-Mendelsohn decomposition. For a bipartite compressed graph, once we find a maximum matching we can determine the Dulmage-Mendelsohn decomposition and thus construct one or more minimum cover separators, such as $\mathbf{S}_X \cup \mathbf{S}_R \cup \mathbf{B}_X$ and $\mathbf{S}_X \cup \mathbf{B}_R \cup \mathbf{B}_X$. A covering separator of minimum size is equivalent to a *min-cut* of a bipartite network constructed from \mathbf{S} and $Border(\mathbf{B})$.

There are two specific min-cuts of the network that are of interest. The tool we use is a *flow alternating path*. A flow alternating path differs from a flow augmenting path in that it need not start from the **source** nor end at the **sink**. Therefore, any contiguous sequence of edges from a flow augmenting path is a flow alternating path. We can now define the following subset:

$$\mathbf{R}_{\mathbf{source}} = \{\mathbf{v} \in \mathbf{V} \mid \mathbf{v} \text{ is reachable from } \mathbf{source} \text{ via a flow alternating path}\}.$$

Intuitively, the subset $\mathbf{R}_{\mathbf{source}}$ provides the “bottle-neck” that limits the total flow to its present value. Indeed, the border of $\mathbf{R}_{\mathbf{source}}$ is a min-cut of the network. A similar subset can be defined with respect to the sink:

$$\mathbf{R}_{\mathbf{sink}} = \{\mathbf{v} \in \mathbf{V} \mid \text{the } \mathbf{sink} \text{ is reachable from } \mathbf{v} \text{ via a flow alternating path}\}.$$

The border of $\mathbf{R}_{\mathbf{sink}}$ is a min-cut of the network. For the network at the top of Figure 7, the two reach sets and their borders are given below.

$$\begin{aligned} \mathbf{R}_{\mathbf{source}} &= \{3, 7, 8, 9, 13, 14, 19, 20, 25, 26, 32\} \\ Border(\mathbf{R}_{\mathbf{source}}) &= \{3, 8, 9, 14, 20, 26, 32\} \\ \mathbf{R}_{\mathbf{sink}} &= \{3, 4, 9, 10, 14, 15, 16, 20, 21, 26, 27, 32, 33\} \\ Border(\mathbf{R}_{\mathbf{sink}}) &= \{3, 9, 14, 15, 20, 26, 32\} \end{aligned}$$

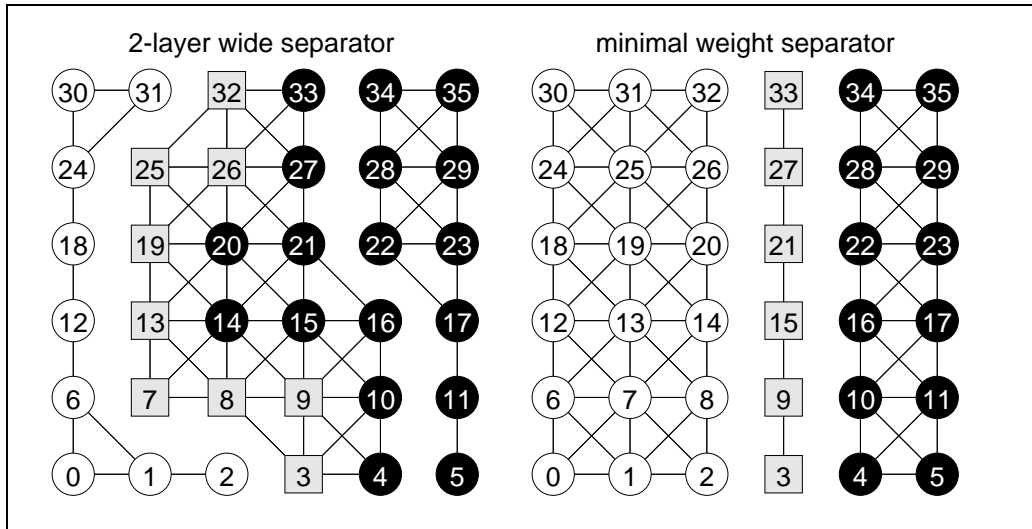
In the context of the Dulmage-Mendelsohn decomposition, $\mathbf{R}_{\mathbf{source}} = \mathbf{S}_I \cup \mathbf{B}_X \cup \mathbf{S}_R \cup \mathbf{S}_X$, $Border(\mathbf{R}_{\mathbf{source}}) = \mathbf{B}_X \cup \mathbf{S}_R \cup \mathbf{S}_X$, $\mathbf{R}_{\mathbf{sink}} = \mathbf{B}_I \cup \mathbf{S}_X \cup \mathbf{B}_R \cup \mathbf{B}_X$, and $Border(\mathbf{R}_{\mathbf{sink}}) = \mathbf{S}_X \cup \mathbf{B}_R \cup \mathbf{B}_X$.

6.3 Enhancement Techniques by Network Flow

In this subsection, we consider new partition improvement techniques based on network flows. We first consider a motivating example. Consider again the grid at the bottom of Figure 3. Using the Dulmage-Mendelsohn decomposition, we can determine the move set $\mathbf{S}_I = \{7, 13, 19, 25\}$ that decreases the separator size the most. The size of the new separator $\{3, 9, 14, 15, 20, 26, 32\}$ is seven. On the other hand, consider the two grids in Figure 8. The left hand grid shows a wide separator $\mathbf{S} \cup Border(\mathbf{B})$ that contains eighteen nodes. The right hand grid shows a separator subset of size six, smaller than the “best” separator that was found using the Dulmage-Mendelsohn decomposition.

There is no contradiction here, yet there is a subtle point that needs to be understood. Theorem 3.3 states that \mathbf{S}_I is the smallest subset of \mathbf{S} that if absorbed by \mathbf{W} will result in

FIG. 8. A 2-layer wide separator and its minimal weight separator subset



the largest decrease of separator size. The “move” that generated the partition in the right hand grid of Figure 8 had \mathbf{W} absorb the separator vertices $\{7, 8, 13, 19, 25, 26, 32\}$, but \mathbf{W} also absorbed the black vertices $\{14, 20\}$, so it is a more general move than that covered by Theorem 3.3. Indeed, $\{7, 8, 13, 14, 19, 20, 25, 26, 32\}$ is the smallest subset of $\mathbf{S} \cup Border(\mathbf{B})$, which when moved to \mathbf{W} will result in the largest decrease in separator size.

We first offer an intuitive explanation to the enhancement. Our goal is to improve an initial partition $[\mathbf{S}, \mathbf{B}, \mathbf{W}]$ of a given compressed graph. The separator \mathbf{S} is first used to construct a compressed bipartite graph based on \mathbf{S} and its adjacent set $Border(\mathbf{B})$ in \mathbf{B} . In Section 6.1, we construct a bipartite network \mathcal{N}_b based on this compressed bipartite graph. A max-flow min-cut solution to this bipartite network \mathcal{N}_b can then be used to obtain an improved new partition for the original compressed graph.

We shall modify our bipartite network so that the max-flow value (and hence min-cut size) of the new network is no larger and possibly smaller. More importantly, the min-cut of this new network also corresponds to a separator of the underlying compressed graph. There is potential to obtain a smaller separator than the one from the original bipartite network.

We now describe how to construct the new network. Let $\mathbf{S} \cup Border(\mathbf{B})$ be the wide separator induced from \mathbf{S} . We have a new partition $[\mathbf{S} \cup Border(\mathbf{B}), Int(\mathbf{B}), \mathbf{W}]$. The wide separator has two portions \mathbf{S} and $Border(\mathbf{B})$. Consider a further subdivision of the subset $Border(\mathbf{B})$ into

$$\mathbf{Y} = \{\mathbf{b} \in Border(\mathbf{B}) \mid Adj(\mathbf{b}) \cap Int(\mathbf{B}) = \emptyset\} \quad \text{and} \quad \mathbf{Z} = Border(\mathbf{B}) \setminus \mathbf{Y}.$$

\mathbf{Y} contains those vertices in $Border(\mathbf{B})$ that are not adjacent to $Int(\mathbf{B})$, while \mathbf{Z} has those vertices that are adjacent to $Int(\mathbf{B})$.

By using these subsets we can form the new network.

- In addition to the source and sink, the nodes in the network are the vertices in \mathbf{S} and $Border(\mathbf{B}) = \mathbf{Y} \cup \mathbf{Z}$.
- For each vertex $\mathbf{s} \in \mathbf{S}$, add the directed edge $(\mathbf{source}, \mathbf{s})$ to the network.

- For each vertex $\mathbf{z} \in \mathbf{Z}$, add the directed edge $(\mathbf{z}, \mathbf{sink})$ to the network.
- For $\mathbf{s} \in \mathbf{S}$ and $\mathbf{b} \in \mathbf{Y} \cup \mathbf{Z} = \text{Border}(\mathbf{B})$ where (\mathbf{s}, \mathbf{b}) is an edge in the original compressed graph, add the directed edge (\mathbf{s}, \mathbf{b}) to the network.
- For $\mathbf{y} \in \mathbf{Y}$ and $\mathbf{b} \in \mathbf{Y} \cup \mathbf{Z} = \text{Border}(\mathbf{B})$, if (\mathbf{y}, \mathbf{b}) is an edge in the original compressed graph, add the directed edge (\mathbf{y}, \mathbf{b}) to the network.
- All edges have infinite capacity. For each vertex \mathbf{s} in $\mathbf{S} \cup \text{Border}(\mathbf{B})$ we set $\text{capacity}(\mathbf{s}) = \text{wt}(\mathbf{s})$.

We shall refer to this new network by \mathcal{N}_w (w for wide). Let us first apply the construction on the partition example of Figure 2. We note that the wide separator is subdivided into these three sets:

$$\mathbf{S} = \{3, 7, 8, 9, 13, 19, 25, 26, 32\}, \quad \mathbf{Y} = \{14, 20\} \quad \text{and} \quad \mathbf{Z} = \{4, 10, 15, 16, 21, 27, 33\}.$$

\mathcal{N}_w is the bottom network of Figure 7. The readers should compare this network with \mathcal{N}_b , the original bipartite network \mathcal{N}_b , at the top of Figure 7.

We are now ready to establish the important result that this new network \mathcal{N}_w has a max-flow (or min-cut) solution no larger than the one from the bipartite network \mathcal{N}_b using the same wide separator $\mathbf{S} \cup \text{Border}(\mathbf{B})$. To prove this result we will construct an intermediate network \mathcal{N}_m by adding the following directed edges into the bipartite network \mathcal{N}_b .

- For $\mathbf{y} \in \mathbf{Y}$ and $\mathbf{b} \in \mathbf{Y} \cup \mathbf{Z} = \text{Border}(\mathbf{B})$, if (\mathbf{y}, \mathbf{b}) is an edge in the original compressed graph, add the directed edge (\mathbf{y}, \mathbf{b}) to the network.

\mathcal{N}_m is the middle network in Figure 7 and contains the edges $(14, 20)$, $(20, 14)$, $(14, 21)$, $(14, 15)$, $(20, 15)$ and $(20, 27)$ in addition to those found in \mathcal{N}_b .

The following lemma will be used in the next theorem to show that the max-flow values for \mathcal{N}_b and \mathcal{N}_m are identical. It proves that adding an edge connecting two vertices that are both adjacent to the sink does not change the max-flow value.

LEMMA 6.1. *Let x and y be two vertices in a given network \mathcal{N}_0 , such that both x and y are connected to the sink. Consider the new network \mathcal{N}_1 by adding a directed edge (x, y) to \mathcal{N}_0 . The networks \mathcal{N}_0 and \mathcal{N}_1 have the same max-flow values.*

Proof. Since the network \mathcal{N}_1 has one additional edge than \mathcal{N}_0 , its max-flow value is at least as large as that of \mathcal{N}_0 . Consider a flow function f_1 for \mathcal{N}_1 that achieves the max-flow value for \mathcal{N}_1 . If $f_1(x, y) = 0$, there is an equivalent flow function for the network \mathcal{N}_0 . If $f_1(x, y) > 0$, define the following flow function f_0 for \mathcal{N}_0 :

$$f_0(x, \text{sink}) = f_1(x, \text{sink}) + f_1(x, y),$$

$$f_0(y, \text{sink}) = f_1(y, \text{sink}) - f_1(x, y),$$

$f_0(x, y) = 0$ (there is no directed edge from x to y in \mathcal{N}_0), and the f_0 values are the same as the f_1 values for the other vertices. It is easy to see that f_0 is a flow function for \mathcal{N}_0 and its flow value is the same as the max-flow value for \mathcal{N}_1 . ■

THEOREM 6.1. *The max-flow values of the networks \mathcal{N}_b and \mathcal{N}_m are the same.*

Proof. First note that the network \mathcal{N}_m is constructed from \mathcal{N}_b by adding a number of directed edges to vertices that are directly linked to the sink. By applying Lemma 6.1 a number of times, we have the result that the networks \mathcal{N}_b and \mathcal{N}_m have the same max-flow values. ■

After we delete from \mathcal{N}_m all edges $(\mathbf{y}, \mathbf{sink})$ for $\mathbf{y} \in Y$, (in our example these edges are $(14, \mathbf{sink})$ and $(20, \mathbf{sink})$), we are left with the network \mathcal{N}_w . We now show that the max-flow value for \mathcal{N}_w is no larger, and can be smaller, than the max-flow value for \mathcal{N}_b . The reach sets from the source and sink are

$$\begin{aligned} \mathbf{R}_{\text{source}} &= \{3, 7, 8, 9, 13, 14, 15, 19, 20, 21, 25, 26, 27, 32, 33\}, \\ \mathbf{R}_{\text{sink}} &= \{3, 4, 9, 10, 15, 16, 21, 27, 33\}, \end{aligned}$$

and they both have the same border, and thus give rise to the same min-cut, $\{3, 9, 15, 21, 27, 33\}$, which has six vertices compared with seven vertices for a min-cut of \mathcal{N}_b .

THEOREM 6.2. *The max-flow value of the network \mathcal{N}_w is less than or equal to the max-flow value of \mathcal{N}_b .*

Proof. Compare the networks \mathcal{N}_m and \mathcal{N}_w . The network \mathcal{N}_w can be obtained from \mathcal{N}_m by removing those directed edges $(\mathbf{y}, \mathbf{sink})$, for $\mathbf{y} \in \mathbf{Y}$. Since \mathcal{N}_w is a sub-network of \mathcal{N}_m , the max-flow value of \mathcal{N}_w must be smaller than or equal to that of \mathcal{N}_m . ■

6.4 Generalization to Wider Separators

The technique introduced in the last section hinges on the choice of the wide separator $\mathbf{S} \cup \text{Border}(\mathbf{B})$. It is easy to generalize this technique for “wider” separators.

Consider a given partition $[\tilde{\mathbf{S}}, \tilde{\mathbf{B}}, \tilde{\mathbf{W}}]$, where the separator set $\tilde{\mathbf{S}}$ need not be minimal but can be quite large. Subdivide the separator set $\tilde{\mathbf{S}}$ into three subsets:

$$\mathbf{X} = \text{Border}(\tilde{\mathbf{S}} \cup \tilde{\mathbf{B}}), \quad \mathbf{Y} = \text{Int}(\tilde{\mathbf{S}}) \quad \text{and} \quad \mathbf{Z} = \text{Border}(\tilde{\mathbf{S}} \cup \tilde{\mathbf{W}}).$$

A network can be constructed in the same manner as given in the last subsection by adding edges from the source to vertices in \mathbf{X} , from vertices in \mathbf{Z} to the sink, and retaining the underlying edges associated with \mathbf{Y} from the original graph. A max-flow min-cut solution to this network will determine a separator subset of $\tilde{\mathbf{S}}$ with minimum weight among all such separator subsets.

The wide separator $\mathbf{S} \cup \text{Border}(\mathbf{B})$ we have used in our last subsection can be viewed as having two layers: \mathbf{S} and $\text{Border}(\mathbf{B})$. Let us now consider a 3-layer separator, given by:

$$\tilde{\mathbf{S}} = \text{Border}(\mathbf{W}) \cup \mathbf{S} \cup \text{Border}(\mathbf{B}),$$

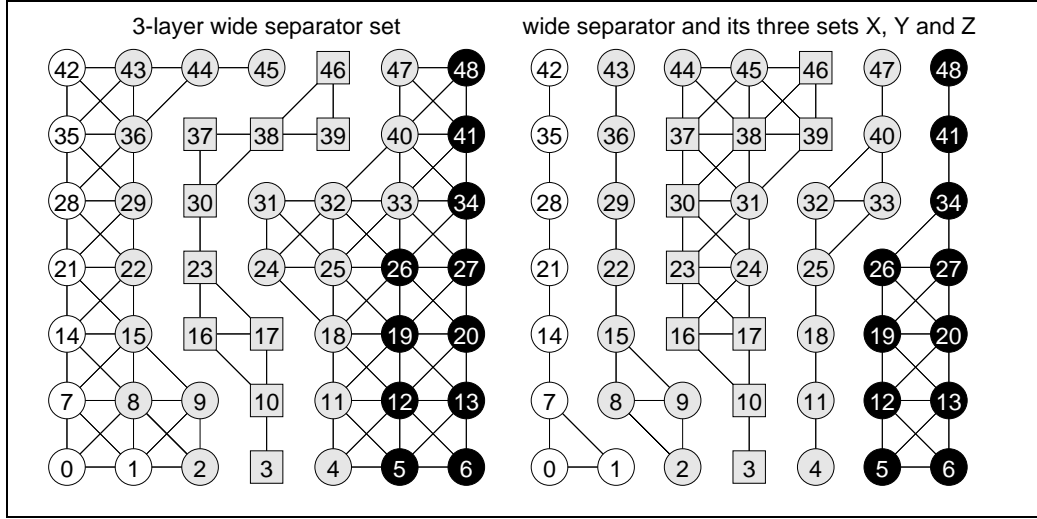
and solve a flow problem on a three-layer network \mathcal{N}_3 .

Figure 9 contains an example to illustrate a 3-layer separator $\tilde{\mathbf{S}}$ given by the union of the following three layers:

$$\begin{aligned} \text{Border}(\mathbf{W}) &= \{2, 8, 9, 15, 22, 29, 36, 43, 44, 45\}, \\ \mathbf{S} &= \{3, 10, 16, 17, 23, 30, 37, 38, 39, 46\}, \\ \text{Border}(\mathbf{B}) &= \{4, 11, 18, 24, 25, 31, 32, 33, 40, 47\}. \end{aligned}$$

The remaining white vertices form the partition subset $\tilde{\mathbf{W}}$, while remaining black vertices form $\tilde{\mathbf{B}}$.

FIG. 9. Finding a minimal separator using a 3-layer network



The right grid in Figure 9 shows the decomposition of the wide separator $\tilde{\mathbf{S}}$ into the three subsets \mathbf{X} , \mathbf{Y} , and \mathbf{Z} . They form the basis on which the network is formed and max-flow min-cut problem is solved. It should be pointed out that often there are more than one min-cut solution. In this example there are three — $\{2, 9, 16, 23, 30, 37, 44\}$, $\{3, 10, 17, 24, 31, 38, 45\}$ and $\{4, 11, 18, 25, 32, 39, 46\}$.

When $\tilde{\mathbf{S}}$ is even wider, say five or seven layers, the space from which we find a minimal weight separator is large. As the number of layers in $\tilde{\mathbf{S}}$ increases, the weight of a minimal separator cannot increase. As in our example in Figure 9, there often will be more than one choice of minimal weight separators; we want to choose one that minimizes our partition evaluation function.

7 Experimental Results

7.1 A closer look at 2-layer smoothing

In this section, we provide some experimental evidence on improving partitions based on the Dulmage-Mendelsohn decomposition. Table 1 contains a typical iteration history for the algorithm in Figure 1. The sparse matrix BCSSTK37, taken from the Harwell-Boeing collection [7], has 25503 degrees of freedom and 1115474 edges. After compression, we work with the weighted compressed graph with 7093 vertices and 88924 edges.

The partitioning algorithm used is from the paper [2]; readers are referred to it for more details. We first constructed a domain decomposition of the graph — there were 141 domains for this test. The initial partition split the domains into two groups of near equal weight. The interface vertices had weight 1166, and the partition has imbalance of $\max\{|\mathbf{B}|, |\mathbf{W}|\} / \min\{|\mathbf{B}|, |\mathbf{W}|\} = 1.013$. We then applied a block Kernighan-Lin algorithm on the domain-segment graph to reduce the separator size to 572, but with an increase in imbalance to 1.118. The separator at this stage tends to be “locally smooth” when it coincides with the boundary of a domain, but the domains do not generally align themselves to form smooth bisectors of the graph.

We then executed the algorithm in Figure 1. Note that the initial imbalance of 1.118 is rather high. At the first step we evaluate two moves that would reduce the separator size and the size of the larger component, namely $\mathbf{Z} = \mathbf{S}_I$ and $\mathbf{Z} = \mathbf{S}_I \cup \mathbf{S}_R$. The $\mathbf{S}_I \cup \mathbf{S}_R$

TABLE 1
Iteration History for BCSSTK37

	$ \mathbf{S} $	imbalance	reduction in $ \mathbf{S} $	partition cost
initial two-set partition	1166	1.013	—	2347.2
after Block Kernighan-Lin	572	1.118	50.9%	1211.5
1. $\mathbf{S}_I \cup \mathbf{S}_R \mapsto \mathbf{W}$	518	1.062	9.4%	1068.1
2. $\mathbf{S}_I \cup \mathbf{S}_R \mapsto \mathbf{W}$	484	1.038	6.6%	986.4
3. $\mathbf{S}_I \cup \mathbf{S}_R \mapsto \mathbf{W}$	480	1.017	0.8%	968.2
4. $\mathbf{S}_I \mapsto \mathbf{W}$	471	1.001	1.9%	942.5
5. $\mathbf{S}_I \mapsto \mathbf{W}$	460	1.012	2.3%	925.5
6. $\mathbf{S}_I \mapsto \mathbf{W}$	446	1.015	3.0%	898.7
7. $\mathbf{S}_R \mapsto \mathbf{W}$	446	1.013	0.0%	897.8
8. $\mathbf{S}_I \mapsto \mathbf{B}$	438	1.030	1.8%	889.1
9. $\mathbf{S}_I \mapsto \mathbf{B}$	434	1.041	0.9%	885.8
10. $\mathbf{S}_I \mapsto \mathbf{B}$	420	1.051	3.2%	861.4
11. $\mathbf{S}_I \mapsto \mathbf{B}$ rejected	419	1.069	0.2%	867.0

move reduces the partition cost function more. This holds for three moves, as we see both the separator weight and the imbalance decrease together. The next three moves are \mathbf{S}_I moves, for the balance is close to unity and the \mathbf{S}_R sets are relatively large.

At step 5, note that the move $\mathbf{Z} = \mathbf{S}_I \mapsto \mathbf{W}$ results in a reduction in separator size but an increase in imbalance. After the move the new set $\mathbf{B}_{\mathbf{Z} \mapsto \mathbf{W}}$ is now smaller than $\mathbf{W}_{\mathbf{Z} \mapsto \mathbf{W}}$ and the difference $|\mathbf{W}_{\mathbf{Z} \mapsto \mathbf{W}}| - |\mathbf{B}_{\mathbf{Z} \mapsto \mathbf{W}}|$ is greater than the previous difference $|\mathbf{B}| - |\mathbf{W}|$. At the next step, we maintain the convention that \mathbf{W} is the smaller portion so that the \mathbf{W} in the $\mathbf{S}_I \mapsto \mathbf{W}$ move at step 6 is the $\mathbf{B}_{\mathbf{Z} \mapsto \mathbf{W}}$ from step 5. Again for step 6, there is a reduction in separator size but increase in imbalance. Step 7 is an instance where the balance is improved with no reduction in separator size. Steps 1-7 were all moves of subsets to the smaller component, so the separator is smoothed in one direction. There is still reduction in the separator to be had by smoothing it against the smaller component, i.e., the larger component absorbs part of the separator, as we see in steps 8-11. The separator weight decreases by 5.9% during steps 8-10 while the imbalance increases from 1.013 to 1.051. At step 11 there is still a possible reduction in separator weight, where $|\mathbf{Z}| = 106$ and $|\text{Adj}_H(\mathbf{Z})| = 105$. Making this move would increase the partition cost function, so the algorithm terminates.

7.2 Comparing 2-layer and 3-layer smoothers

We have tested the various partition improvement techniques described in this paper on a collection of test matrix problems. Table 2 contains the description of ten sparse matrix problems from the Harwell-Boeing collection [7].

Table 3 presents statistics for finding a top level separator for the three algorithms. The cost is $|\mathbf{S}| \left(1 + \alpha \frac{\max(|\mathbf{B}|, |\mathbf{W}|)}{\min(|\mathbf{B}|, |\mathbf{W}|)}\right)$, where the penalty multiplier $\alpha = 1$. The median cost value for twenty-five runs is found in the table — for each run the matrix was randomly permuted. The initial partition is obtained from domain decomposition followed by the block Kernighan-Lin scheme in [2] as discussed in the last subsection.

The three algorithms tested are labeled \mathcal{N}_b , \mathcal{N}_w , and \mathcal{N}_3 respectively in the table.

TABLE 2
Statistics for Harwell-Boeing Matrices

MATRIX	ORIGINAL		COMPRESSED		MMD		
	$ V $	$ E $	$ V $	$ E $	NZF/ 10^3	OPS/ 10^6	CPU
BCSSTK30	28924	2014568	9289	222884	3725	777	1.72
BCSSTK31	35588	1145828	17403	288806	5156	2400	4.70
BCSSTK32	44609	1970092	14821	226974	5147	1048	2.84
BCSSTK33	8738	583166	4344	164284	2654	1301	1.10
BCSSTK35	30237	1419926	6611	65934	2780	406	0.90
BCSSTK36	23052	1120088	4351	37166	1767	626	0.51
BCSSTK37	25503	1115474	7093	88924	2829	548	1.00
BCSSTK39	46772	2042522	10140	81762	7669	2194	1.33
MN12	264002	13115458	51920	569226	40404	24810	12.45
PWT	217918	11634424	41531	483791	63992	49875	7.93

Column \mathcal{N}_b has statistics for the partition improvement algorithm in Figure 1 using the Dulmage-Mendelsohn decomposition, i.e., it solves the max-flow problem defined on the bipartite network \mathcal{N}_b . Column \mathcal{N}_w contains results for the partition improvement algorithm in Figure 1 using the 2-layer wide network \mathcal{N}_w . These two algorithms iterate until no improvement can be made. Inside the loop, they make a first attempt to improve the partition based on a two-layer separator $\mathbf{S} \cup \text{Border}(\mathbf{B})$ using the current separator \mathbf{S} and the larger portion \mathbf{B} . If no improvement on this attempt, it will then try the two-layer separator $\mathbf{S} \cup \text{Border}(\mathbf{W})$ with the smaller portion \mathbf{W} .

TABLE 3
Top Level Separators, median cost of twenty-five runs

matrix	using \mathcal{N}_b			using \mathcal{N}_w			using \mathcal{N}_3		
	cost	$ S $	balance	cost	$ S $	balance	cost	$ S $	balance
BCSSTK30	467	223	1.095	421	209	1.012	421	209	1.012
BCSSTK31	707	353	1.001	679	339	1.003	680	332	1.049
BCSSTK32	791	355	1.228	717	322	1.226	711	271	1.624
BCSSTK33	847	421	1.012	847	421	1.012	847	421	1.012
BCSSTK35	344	162	1.121	306	144	1.128	307	96	2.194
BCSSTK36	715	357	1.002	644	325	1.043	662	331	1.000
BCSSTK37	894	440	1.031	889	437	1.033	889	437	1.033
BCSSTK39	451	225	1.003	451	225	1.003	451	225	1.003
MN12	1736	861	1.017	1662	815	1.039	1609	791	1.034
PWT	1441	720	1.001	1441	720	1.001	1442	720	1.003

The algorithm associated with \mathcal{N}_3 is also iterative in nature. It is simpler since it tries to improve the partition using the 3-layer set $\mathbf{S} \cup \text{Border}(\mathbf{B}) \cup \text{Border}(\mathbf{W})$. It continues until no improvement can be obtained. Our experience shows that the algorithm for \mathcal{N}_3 typically requires half the number of steps or less when compared to the first two algorithms. But of course, it takes more time at each step since it is solving a larger network problem. We see that often using the network \mathcal{N}_w gives sizable partition improvement over the network

\mathcal{N}_b . Using the 3-layer network sometimes gives additional but small improvement.

TABLE 4
Nested dissection with respect to multiple minimum degree

matrix	factor entries			factor ops			ordering cpu		
	\mathcal{N}_b	\mathcal{N}_w	\mathcal{N}_3	\mathcal{N}_b	\mathcal{N}_w	\mathcal{N}_3	\mathcal{N}_b	\mathcal{N}_w	\mathcal{N}_3
BCSSTK30	1.24	1.11	1.13	1.88	1.42	1.46	4.86	5.16	6.07
BCSSTK31	0.89	0.84	0.84	0.58	0.52	0.50	3.21	3.20	3.48
BCSSTK32	1.12	1.09	1.07	1.48	1.38	1.33	4.40	3.96	4.19
BCSSTK33	0.86	0.83	0.80	0.71	0.65	0.57	4.86	4.74	7.21
BCSSTK35	1.15	1.11	1.09	1.55	1.41	1.36	4.20	4.13	4.20
BCSSTK36	1.13	1.07	1.07	1.42	1.25	1.25	4.47	4.47	4.47
BCSSTK37	1.09	1.07	1.06	1.36	1.35	1.30	4.24	4.29	4.42
BCSSTK39	0.94	0.94	0.94	0.95	0.94	0.94	4.11	4.11	4.05
MN12	1.08	1.00	0.97	1.07	0.92	0.82	3.53	3.56	3.57
PWT	0.74	0.74	0.74	0.47	0.47	0.46	4.26	4.22	4.36

We have also used the three partition improvement algorithms to find separators in the context of finding fill-reducing sparse matrix orderings. Tables 4 and 5 contains statistics of nested dissection orderings and multisection orderings [3] using the three partition improvement schemes. The statistics are scaled by results from the multiple minimum degree ordering. Each result in the tables comes from the run that generated the median factor operations in twenty-five runs.

TABLE 5
Multisection with respect to multiple minimum degree

matrix	factor entries			factor ops			ordering cpu		
	\mathcal{N}_b	\mathcal{N}_w	\mathcal{N}_3	\mathcal{N}_b	\mathcal{N}_w	\mathcal{N}_3	\mathcal{N}_b	\mathcal{N}_w	\mathcal{N}_3
BCSSTK30	1.09	1.01	1.04	1.30	1.08	1.15	4.87	5.16	6.07
BCSSTK31	0.90	0.86	0.85	0.61	0.58	0.55	3.19	3.22	3.49
BCSSTK32	0.97	0.95	0.94	0.90	0.85	0.84	4.04	3.96	4.19
BCSSTK33	0.81	0.79	0.79	0.61	0.57	0.57	4.86	4.74	7.20
BCSSTK35	1.03	1.00	0.99	1.06	1.01	0.97	4.20	4.12	4.19
BCSSTK36	0.96	0.94	0.94	0.85	0.82	0.82	4.49	4.48	4.47
BCSSTK37	0.95	0.94	0.93	0.87	0.85	0.84	4.24	4.28	4.41
BCSSTK39	0.89	0.89	0.90	0.77	0.78	0.79	4.12	4.11	4.05
MN12	1.00	0.94	0.93	0.88	0.77	0.75	3.53	3.58	3.56
PWT	0.79	0.79	0.79	0.59	0.60	0.59	4.25	4.21	4.36

We have experimented with using a network with five layers, seven layers and more to improve separators. Any improvement is usually modest while the run times for the orderings increase dramatically as the time to solve the max-flow problems for the larger networks takes a larger portion of the ordering time.

Wide separators have a disadvantage for the min-cuts may be spread across the wide separator. Consider an example where we start with a partition that has good balance. When we use a very wide separator (say seven levels) to form a network, a min-cut may lie far to one side or the other of the “thin” separator. Though the separator induced by

the min-cut might be smaller than the present separator, the partition that would result may have a larger cost, and so the new partition would not be accepted. There is one min-cut closest to the source and one closest to the sink (the two may be identical), and neither might result in a better partition. We are not primarily interested in finding the minimal weight separator — we want a partition whose cost is minimal. To this end we are exploring ways to modify the network such that the min-cut determines a partition with minimal cost.

8 Concluding Remarks

In this paper, we have presented a detailed exposition of the Dulmage-Mendelsohn decomposition of bipartite graphs in the context of improving bisector-based partitions. In the literature, this decomposition has been used to obtain a vertex separator from an edge separator, and in iteratively improving a vertex separator. We have also used the decomposition to improve the balance of a partition.

Another contribution of this paper is the extension of the Dulmage-Mendelsohn decomposition to compressed graphs, a special type of weighted graphs that occur naturally and frequently in practice. For such graphs, we have related the decomposition with the well-known maximum flow network problem. Finding a separator of minimum cover based on the Dulmage-Mendelsohn decomposition is the same as obtaining a min-cut of a bipartite network problem. We have also introduced an enhancement by solving a slightly modified network problem, the solution of which will often yield a smaller separator.

We have provided experimental results to demonstrate the viability of the approaches to improve bisectors and partitions. These results should be viewed as additional evidence to those included in our earlier paper [2]. We recommend this smoothing step using graph matching or network max-flow min-cut as a standard final process on all dissection-based ordering codes. Indeed, such smoother codes are present in the recently developed software, such as the new CHACO code [17] by Hendrickson and Rothberg, and the IBM Watson Graph Partitioning code WGPP [13] by Gupta.

Max-flow techniques have potential application in other contexts, particularly to find separators of coarse graphs used in multilevel algorithms [13], [17], [20] or the domain/segments graphs from a domain-decomposition approach [2]. While we have concentrated on “thin” networks, where the distance from the source to the sink is small, in principal one can attack much wider separators, perhaps containing all of a graph save for a source and sink vertex. While this would be prohibitively expensive for a large graph, it could be profitably used for a coarse graph or domain/segment graph. The drawback is that a min-cut might naturally lie very close to the source or sink and thus induce a poorly balanced partition. By increasing the weight of vertices close to the source or sink one can force the min-cut to split the graph into two more equally-sized pieces [10].

Acknowledgements

We would like to thank Matt Berge of Boeing Information and Support Services for several enlightening conversations on network flow, and Stan Eisenstat of Yale University for many insightful comments on an earlier draft. We owe a great debt to John Gilbert of Xerox PARC for his notes on bipartite graphs and the Dulmage-Mendelsohn decomposition.

References

- [1] C. ASHCRAFT, *Compressed graphs and the minimum degree algorithm*, SIAM J. Sci. Comput., 16 (1995), pp. 1404–1411.

- [2] C. ASHCRAFT AND J. LIU, *Using domain decomposition to find graph bisectors*, Tech. Rep. ISSTECH-95-024, Boeing Computer Services, Seattle, WA, 1995.
- [3] ———, *Robust ordering of sparse matrices using multisection*, Tech. Rep. ISSTECH-96-002, Boeing Computer Services, Seattle, WA, 1996.
- [4] ———, *Generalized nested dissection: some recent progress*, in Fifth SIAM Conference on Applied Linear Algebra, Snowbird, Utah, June 18, 1994.
- [5] T. BUI AND C. JONES, *A heuristic for reducing fill-in in sparse matrix factorization*, in Proceeding of Sixth SIAM Conference on Parallel Processing, 1993, pp. 445–452.
- [6] A. C. DAMHAUG, *Sparse Solution of Finite Element Equations*, PhD thesis, The Norwegian Institute of Technology, 1992.
- [7] I. DUFF, R. GRIMES, AND J. LEWIS, *Sparse matrix test problems*, ACM Trans. on Math Software, 15 (1989), pp. 1–14.
- [8] A. DULMAGE AND N. MENDELSON, *Coverings of bipartite graphs*, Can. J. Math, 10 (1958), pp. 517–534.
- [9] J. EDMONDS AND R. M. KARP, *Theoretical improvements in algorithmic efficiency for network flow problem*, Journal of the ACM, 19 (1972), pp. 248–264.
- [10] S. C. EISENSTAT. personal communication.
- [11] L. R. FORD AND D. R. FULKERSON, *Flows in Networks*, Princeton University Press, Princeton, N.J., 1962.
- [12] D. R. FULKERSON, *Flow networks and combinatorial operations research*, Amer. Math. Monthly, 73 (1966), p. 115.
- [13] A. GUPTA, *WGPP: Watson Graph Partitioning and sparse matrix ordering Package*, Tech. Rep. Users Manual, IBM T.J. Watson Research Center, New York, 1996.
- [14] P. HALL, *On representatives of subsets*, J. London Math. Society, 10 (1935), pp. 26–30.
- [15] B. HENDRICKSON AND R. LELAND, *The Chaco user's guide*, Tech. Rep. SAND93-2339, Sandia National Laboratories, Albuquerque, NM, 1993.
- [16] ———, *An improved spectral graph partitioning algorithm for mapping parallel computations*, SIAM J. Sci. Comput., 16 (1995), pp. 452–469.
- [17] B. HENDRICKSON AND E. ROTHBERG, *Improving the runtime and quality of nested dissection ordering*, tech. rep., Sandia National Laboratories, 1996.
- [18] A. ITAI AND Y. SHILOACH, *Maximum flow in planar networks*, SIAM J. Comp., 8 (1979), pp. 135–150.
- [19] G. KARYPIS AND V. KUMAR, *A fast and high quality multilevel scheme for partitioning irregular graphs*, Tech. Rep. TR 95-035, Department of Computer Science, University of Minnesota, Minnesota, 1995.
- [20] ———, *Metis: Unstructured graph partitioning and sparse matrix ordering system*, tech. rep., Department of Computer Science, University of Minnesota, Minnesota, 1995.
- [21] B. W. KERNIGHAN AND S. LIN, *An efficient heuristic procedure for partitioning graphs*, Bell System Technical Journal, 49 (1970), pp. 291–307.
- [22] E. L. LAWLER, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, 1967.
- [23] C. E. LEISERSON AND J. G. LEWIS, *Ordering for parallel sparse symmetric factorization*, in Parallel Processing for Scientific Computing, 1989, pp. 27–31.
- [24] J. W. H. LIU, *A graph partitioning algorithm by node separators*, ACM Trans. on Math Software, 15 (1989), pp. 198–219.
- [25] A. POTHEN AND C. FAN, *Computing the block triangular form of a sparse matrix*, ACM Trans. on Math Software, 16 (1990), pp. 303–324.
- [26] A. POTHEN, H. SIMON, AND K. LIOU, *Partitioning sparse matrices with eigenvectors of graphs*, SIAM J. Matrix Analysis and Applic., 11 (1990), pp. 430–452.
- [27] R. E. TARJAN, *Data Structures and Network Algorithms*, SIAM, Philadelphia, PA, 1983.
- [28] J. D. ULLMAN, *Computational Aspects of VLSI*, Computer Science Press, Rockville, Md, 1984.