A MULTILEVEL BILINEAR PROGRAMMING ALGORITHM FOR THE VERTEX SEPARATOR PROBLEM *

WILLIAM W. HAGER†, JAMES T. HUNGERFORD‡, AND ILYA SAFRO§

Abstract. The Vertex Separator Problem for a graph is to find the smallest collection of vertices whose removal breaks the graph into two disconnected subsets that satisfy specified size constraints. The Vertex Separator Problem was formulated in the paper 10.1016/j.ejor.2014.05.042 as a continuous (non-concave/non-convex) bilinear quadratic program. In this paper, we develop a more general continuous bilinear program which incorporates vertex weights, and which applies to the coarse graphs that are generated in a multilevel compression of the original Vertex Separator Problem. A Mountain Climbing Algorithm is used to find a stationary point of the bilinear program, while perturbation techniques are used to either dislodge an iterate from a saddle point or escape from a local optimum. We determine the smallest possible perturbation that will force the current iterate to a different location, with a possibly better separator. The algorithms for solving the bilinear program are employed during the solution and refinement phases in a multilevel scheme. Computational results and comparisons demonstrate the advantage of the proposed algorithm.

Key words. vertex separator, continuous formulation, graph partitioning, combinatorial optimization, multilevel computations, graphs, weighted edge contractions, coarsening, relaxation, multilevel algorithm, algebraic distance

AMS subject classifications. 90C35, 90C27, 90C20, 90C06

1. Introduction. Let $G = (\mathcal{V}, \mathcal{E})$ be a graph on vertex set $\mathcal{V} = \{1, 2, \ldots, n\}$ and edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. We assume G is simple and undirected; that is for any vertices i and j we have $(i, i) \notin \mathcal{E}$ and $(i, j) \in \mathcal{E}$ if and only if $(j, i) \in \mathcal{E}$ (note that this implies that $|\mathcal{E}|$, the number of elements in $\mathcal{E}$, is twice the total number of edges in G). For each $i \in \mathcal{V}$, let $c_i \in \mathbb{R}$ denote the cost and $w_i > 0$ denote the weight of vertex i. If $\mathcal{Z} \subseteq \mathcal{V}$, then

$$\mathcal{C}(\mathcal{Z}) = \sum_{i \in \mathcal{Z}} c_i \quad \text{and} \quad \mathcal{W}(\mathcal{Z}) = \sum_{i \in \mathcal{Z}} w_i$$

denote the total cost and weight of the vertices in $\mathcal{Z}$, respectively.

If the vertices $\mathcal{V}$ are partitioned into three disjoint sets $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{S}$, then $\mathcal{S}$ *separates* $\mathcal{A}$ and $\mathcal{B}$ if there is no edge $(i, j) \in \mathcal{E}$ with $i \in \mathcal{A}$ and $j \in \mathcal{B}$. The Vertex Separator Problem (VSP) is to minimize the cost of $\mathcal{S}$ while requiring that $\mathcal{A}$ and $\mathcal{B}$ have approximately the same weight. We formally state the VSP as follows:

$$\begin{aligned} &\min_{\mathcal{A},\mathcal{S},\mathcal{B} \subseteq \mathcal{V}} \quad \mathcal{C}(\mathcal{S}) \\ \text{subject to} \quad &\mathcal{S} = \mathcal{V} \setminus (\mathcal{A} \cup \mathcal{B}), \quad \mathcal{A} \cap \mathcal{B} = \emptyset, \quad (\mathcal{A} \times \mathcal{B}) \cap \mathcal{E} = \emptyset, \\ &\ell_a \le \mathcal{W}(\mathcal{A}) \le u_a, \ \text{ and } \ \ell_b \le \mathcal{W}(\mathcal{B}) \le u_b, \end{aligned} \tag{1.1}$$

* May 25, 2016. The research was supported by the Office of Naval Research under Grants N00014-11-1-0068 and N00014-15-1-2048 and by the National Science Foundation under grants 1522629 and 1522751. Part of the research was performed while the second author was a Givens Associate at Argonne National Laboratory.

† hager@ufl.edu, http://people.clas.ufl.edu/hager/ PO Box 118105, Department of Mathematics, University of Florida, Gainesville, FL 32611-8105. Phone (352) 294-2308. Fax (352) 392-8357.

‡ jamesthungerford@gmail.com, M.A.I.O.R. – Management, Artificial Intelligence, and Operations Research, Srl., Lucca, ITALY

§ isafro@clemson.edu, http://www.cs.clemson.edu/∼isafro, 228 McAdams Hall, School of Computing, Clemson University, Clemson, SC 29634. Phone (864) 656-0637.

where ℓ_a, ℓ_b, u_a, and u_b are given nonnegative real numbers less than or equal to $\mathcal{W}(\mathcal{V})$. The constraints $\mathcal{S} = \mathcal{V} \setminus (\mathcal{A} \cup \mathcal{B})$ and $\mathcal{A} \cap \mathcal{B} = \emptyset$ ensure that $\mathcal{V}$ is partitioned into disjoint sets $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{S}$, while the constraint $(\mathcal{A} \times \mathcal{B}) \cap \mathcal{E} = \emptyset$ ensures that there are no edges between the sets $\mathcal{A}$ and $\mathcal{B}$. Throughout the paper, we assume (1.1) is feasible. In particular, if $\ell_a, \ell_b \geq 1$, then there exist at least two distinct vertices i and j such that $(i, j) \notin \mathcal{E}$; that is, G is not a complete graph.

Vertex separators have applications in VLSI design [24, 28, 39], finite element methods [32], parallel processing [11], sparse matrix factorizations ([10, Sect. 7.6], [15, Chapter 8], and [34]), hypergraph partitioning [23], and network security [7, 25, 31]. The VSP is NP-hard [5, 14]. However, due to its practical significance, many heuristics have been developed for obtaining approximate solutions, including node-swapping heuristics [29], spectral methods [34], semidefinite programming methods [12], and recently a breakout local search algorithm [3].

It has been demonstrated repeatedly that for problems on large-scale graphs, such as finding minimum k-partitionings [6, 19, 22] or minimum linear arrangements [36, 37], optimization algorithms can be much more effective when carried out in a multilevel framework. In a multilevel framework, a hierarchy of increasingly smaller graphs is generated which approximate the original graph, but with fewer degrees of freedom. The problem is solved for the coarsest graph in the hierarchy, and the solution is gradually uncoarsened and refined to obtain a solution for the original graph. During the uncoarsening phase, optimization algorithms are commonly employed locally to make fast improvements to the solution at each level in the algorithm. Although multilevel algorithms are inexact for most NP-hard problems on graphs, they typically produce very high quality solutions and are very fast (often linear in the number of vertices plus the number of edges with no hidden coefficients).

Early methods [16, 34], for computing vertex separators were based on computing edge separators (bipartitions of $\mathcal{V}$ with low cost edge-cuts). In these algorithms, vertex separators are obtained from edge separators by selecting vertices incident to the edges in the cut. More recently, [1] gave a method for computing vertex separators in a graph by finding low cost net-cuts in an associated hypergraph. Some of the most widely used heuristics for computing edge separators are the node swapping heuristics of Fiduccia-Mattheyses [13] and Kernighan-Lin [24], in which vertices are exchanged between sets until the current partition is considered to be locally optimal. Many multilevel edge separator algorithms have been developed and incorporated into graph partitioning packages (see survey in [6]). In [2], a Fiduccia-Mattheyses type heuristic is used to find vertex separators directly. Variants of this algorithm have been incorporated into the multilevel graph partitioners METIS [21, 22] and BEND [20].

In [18], the authors make a departure from traditional discrete-based heuristics for solving the VSP, and present the first formulation of the problem as a continuous optimization problem. In particular, when the vertex weights are identically one, conditions are given under which the VSP is equivalent to solving a continuous bilinear quadratic program.

The preliminary numerical results of [18] indicate that the bilinear programming formulation can serve as an effective tool for making local improvements to a solution in a multilevel context. The current work makes the following contributions:

1. The bilinear programming model of [18] is extended to the case where vertex weights are possibly greater than one. This generalization is important since each vertex in a multilevel compression corresponds to a collection of vertices in the original graph. The bilinear formulation of the compressed graph is not exactly equivalent to the VSP for the compressed graph, but it very closely approximates the VSP as we show.
2. Optimization algorithms applied to the bilinear VSP model converge to stationary points

which may not be global optima. Two techniques are developed to escape from a stationary point and explore a new part of the solution space. One technique uses the first-order optimality conditions to construct an infinitesimal perturbation of the objective function with the property that a gradient descent step moves the iterate to a new location where the separator could be smaller. This technique is particularly effective when the current iterate lies at a saddle point rather than a local optimum. The second technique involves relaxing the constraint that there are no edges between the sets in the partition. Since this constraint is enforced by a penalty in the objective, we determine the smallest possible relaxation of the penalty for which a gradient descent step moves the iterate to a new location where the separator could be smaller.

3. A multilevel algorithm is developed which incorporates the weighted bilinear program in the refinement phase along with the techniques to escape from a stationary point. Computational results are given to compare the quality of the solutions obtained with the bilinear programming approach to a multilevel vertex separator routine in the METIS package. The algorithm is shown to be especially effective on p2p networks and graphs having heavy-tailed degree distributions.

The outline of the paper is as follows. Section 3 reviews the bilinear programming formulation of the VSP in [18] and develops the weighted formulation which is suitable for the coarser levels in our algorithm. Section 4 presents an algorithm for finding approximate solutions to the bilinear program and develops two techniques for escaping from a stationary point. Section 5 summarizes the multilevel framework, while Section 6 gives numerical results comparing our algorithm to METIS. Conclusions are drawn in Section 7.

2. Notation. Vectors or matrices whose entries are all 0 or all 1 are denoted by $\mathbf{0}$ or $\mathbf{1}$ respectively, where the dimension will be clear from the context. If $\mathbf{x} \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$, then $\nabla f(\mathbf{x})$ denotes the gradient of f at $\mathbf{x}$, a row vector, and $\nabla^2 f(\mathbf{x})$ is the Hessian. If $f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, then $\nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y})$ is the row vector corresponding to the first n entries of $\nabla f(\mathbf{x}, \mathbf{y})$, while $\nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})$ is the row vector corresponding to the last n entries. If $\mathbf{A}$ is a matrix, then $\mathbf{A}_i$ denotes the i-th row of $\mathbf{A}$. If $\mathbf{x} \in \mathbb{R}^n$, then $\mathbf{x} \geq \mathbf{0}$ means $x_i \geq 0$ for all i, and $\mathbf{x}^\mathsf{T}$ denotes the transpose, a row vector. Let $\mathbf{I} \in \mathbb{R}^{n \times n}$ denote the $n \times n$ identity matrix, let $\mathbf{e}_i$ denote the i-th column of $\mathbf{I}$, and let $|\mathcal{A}|$ denote the number of elements in the set $\mathcal{A}$.

3. Bilinear programming formulation. Since minimizing $\mathcal{C}(\mathcal{S})$ in (1.1) is equivalent to maximizing $\mathcal{C}(\mathcal{A} \cup \mathcal{B})$, we may view the VSP as the following maximization problem:

$$\begin{aligned} \max_{\mathcal{A},\mathcal{B} \subseteq \mathcal{V}} \quad & \mathcal{C}(\mathcal{A} \cup \mathcal{B}) \\ \text{subject to} \quad & \mathcal{A} \cap \mathcal{B} = \emptyset, \quad (\mathcal{A} \times \mathcal{B}) \cap \mathcal{E} = \emptyset, \\ & \ell_a \leq \mathcal{W}(\mathcal{A}) \leq u_a, \ \text{ and } \ \ell_b \leq \mathcal{W}(\mathcal{B}) \leq u_b. \end{aligned} \tag{3.1}$$

Next, observe that any pair of subsets $\mathcal{A}, \mathcal{B} \subseteq \mathcal{V}$ is associated with a pair of incidence vectors $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$ defined by

$$x_i = \begin{cases} 1, & \text{if } i \in \mathcal{A} \\ 0, & \text{if } i \notin \mathcal{A} \end{cases} \quad \text{and} \quad y_i = \begin{cases} 1, & \text{if } i \in \mathcal{B} \\ 0, & \text{if } i \notin \mathcal{B} \end{cases} . \tag{3.2}$$

Let $\mathbf{H} := (\mathbf{A} + \mathbf{I})$, where $\mathbf{A}$ is the $n \times n$ adjacency matrix for G defined by $a_{ij} = 1$ if $(i,j) \in \mathcal{E}$ and $a_{ij} = 0$ otherwise, and $\mathbf{I}$ is the $n \times n$ identity matrix. Note that since G is undirected, $\mathbf{H}$ is

symmetric. Then we have

$$\begin{aligned}
\mathbf{x}^{\mathsf{T}}\mathbf{H}\mathbf{y} &= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i a_{ij} y_j + \sum_{i=1}^{n} x_i y_i \\
&= \sum_{x_i=1}\sum_{y_j=1} a_{ij} + \sum_{x_i=y_i=1} 1 \\
&= \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{B}} a_{ij} + \sum_{i\in\mathcal{A}\cap\mathcal{B}} 1 \\
&= |(\mathcal{A}\times\mathcal{B})\cap\mathcal{E}| + |\mathcal{A}\cap\mathcal{B}|. \qquad (3.3)
\end{aligned}$$

So, the constraints $\mathcal{A}\cap\mathcal{B}=\emptyset$ and $(\mathcal{A}\times\mathcal{B})\cap\mathcal{E}=\emptyset$ in (3.1) hold if and only if

$$\mathbf{x}^{\mathsf{T}}\mathbf{H}\mathbf{y} = 0. \qquad (3.4)$$

Hence, a binary formulation of (3.1) is

$$\begin{aligned}
&\max_{\mathbf{x},\mathbf{y}\in\{0,1\}^n} \quad \mathbf{c}^{\mathsf{T}}(\mathbf{x}+\mathbf{y}) \\
&\text{subject to} \qquad \mathbf{x}^{\mathsf{T}}\mathbf{H}\mathbf{y} = 0, \qquad (3.5)\\
&\qquad \ell_a \le \mathbf{w}^{\mathsf{T}}\mathbf{x} \le u_a, \ \text{ and } \ \ell_b \le \mathbf{w}^{\mathsf{T}}\mathbf{y} \le u_b,
\end{aligned}$$

where $\mathbf{c}$ and $\mathbf{w}$ are the $n-$dimensional vectors which store the costs c_i and weights w_i of vertices, respectively.

Now, consider the following problem in which the quadratic constraint of (3.5) has been relaxed:

$$\begin{aligned}
&\max_{\mathbf{x},\mathbf{y}\in\{0,1\}^n} \quad f(\mathbf{x},\mathbf{y}) := \ \mathbf{c}^{\mathsf{T}}(\mathbf{x}+\mathbf{y}) - \gamma\mathbf{x}^{\mathsf{T}}\mathbf{H}\mathbf{y} \qquad (3.6)\\
&\text{subject to} \quad \ell_a \le \mathbf{w}^{\mathsf{T}}\mathbf{x} \le u_a \quad \text{and} \quad \ell_b \le \mathbf{w}^{\mathsf{T}}\mathbf{y} \le u_b.
\end{aligned}$$

Here, $\gamma\in\mathbb{R}$. Notice that $\gamma\mathbf{x}^{\mathsf{T}}\mathbf{H}\mathbf{y}$ acts as a penalty term in (3.6) when $\gamma\ge 0$, since $\mathbf{x}^{\mathsf{T}}\mathbf{H}\mathbf{y}\ge 0$ for every $\mathbf{x},\mathbf{y}\in\{0,1\}^n$. Moreover, (3.6) gives a relaxation of (3.5), since the constraint (3.4) is not enforced. Problem (3.6) is feasible since the VSP (3.1) is feasible by assumption. The following proposition gives conditions under which (3.6) is essentially equivalent to (3.5) and (3.1).

PROPOSITION 3.1. *If $\mathbf{w}\ge\mathbf{1}$ and $\gamma>0$ with $\gamma\ge\max\{c_i : i\in\mathcal{V}\}$, then for any feasible point $(\mathbf{x},\mathbf{y})$ in (3.6) satisfying*

$$f(\mathbf{x},\mathbf{y}) \ge \gamma(\ell_a+\ell_b), \qquad (3.7)$$

there is a feasible point $(\bar{\mathbf{x}},\bar{\mathbf{y}})$ in (3.6) such that

$$f(\bar{\mathbf{x}},\bar{\mathbf{y}}) \ge f(\mathbf{x},\mathbf{y}) \quad \textit{and} \quad \bar{\mathbf{x}}^{\mathsf{T}}\mathbf{H}\bar{\mathbf{y}} = 0. \qquad (3.8)$$

Hence, if the optimal objective value in (3.6) is at least $\gamma(\ell_a+\ell_b)$, then there exists an optimal solution $(\mathbf{x}^*,\mathbf{y}^*)$ to (3.6) such that an optimal solution to (3.1) is given by

$$\mathcal{A}=\{i : x_i^*=1\}, \ \ \mathcal{B}=\{i : y_i^*=1\}, \ \ \textit{and} \ \ \mathcal{S}=\{i : x_i^*=y_i^*=0\}. \qquad (3.9)$$

Proof. Let $(\mathbf{x}, \mathbf{y})$ be a feasible point in (3.6) satisfying (3.7). Since $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{H}$ are nonnegative, we have $\mathbf{x}^\mathsf{T}\mathbf{H}\mathbf{y} \geq 0$. If $\mathbf{x}^\mathsf{T}\mathbf{H}\mathbf{y} = 0$, then we simply take $\bar{\mathbf{x}} = \mathbf{x}$ and $\bar{\mathbf{y}} = \mathbf{y}$, and (3.8) is satisfied. Now suppose instead that

$$\mathbf{x}^\mathsf{T}\mathbf{H}\mathbf{y} > 0. \tag{3.10}$$

Then,

$$\gamma(\ell_a + \ell_b) \leq f(\mathbf{x}, \mathbf{y}) = \mathbf{c}^\mathsf{T}(\mathbf{x} + \mathbf{y}) - \gamma\mathbf{x}^\mathsf{T}\mathbf{H}\mathbf{y} \tag{3.11}$$

$$< \mathbf{c}^\mathsf{T}(\mathbf{x} + \mathbf{y}) \tag{3.12}$$

$$\leq \gamma\mathbf{1}^\mathsf{T}(\mathbf{x} + \mathbf{y}). \tag{3.13}$$

Here, (3.11) is due to (3.7), (3.12) is due to (3.10) and the assumption that $\gamma > 0$, and (3.13) holds by the assumption that $\gamma \geq \max\{c_i : i \in \mathcal{V}\}$. It follows that either $\mathbf{1}^\mathsf{T}\mathbf{x} > \ell_a$ or $\mathbf{1}^\mathsf{T}\mathbf{y} > \ell_b$.

Assume without loss of generality that $\mathbf{1}^\mathsf{T}\mathbf{x} > \ell_a$. Since $\mathbf{x}$ is binary and ℓ_a is an integer, we have

$$\mathbf{1}^\mathsf{T}\mathbf{x} \geq \ell_a + 1.$$

Since the entries in $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{H}$ are all non-negative integers, (3.10) implies that there exists an index i such that $\mathbf{H}_i\mathbf{y} \geq 1$ and $x_i = 1$ (recall that subscripts on a matrix correspond to the rows). If $\hat{\mathbf{x}} = \mathbf{x} - \mathbf{e}_i$, then $(\hat{\mathbf{x}}, \mathbf{y})$ is feasible in problem (3.6) since $u_a \geq \mathbf{w}^\mathsf{T}\mathbf{x} > \mathbf{w}^\mathsf{T}\hat{\mathbf{x}}$ and

$$\mathbf{w}^\mathsf{T}\hat{\mathbf{x}} \geq \mathbf{1}^\mathsf{T}\hat{\mathbf{x}} = \mathbf{1}^\mathsf{T}\mathbf{x} - 1 \geq \ell_a.$$

Here the first inequality is due to the assumption that $\mathbf{w} \geq \mathbf{1}$. Furthermore,

$$f(\hat{\mathbf{x}}, \mathbf{y}) = f(\mathbf{x}, \mathbf{y}) - c_i + \gamma\mathbf{H}_i\mathbf{y} \geq f(\mathbf{x}, \mathbf{y}) - c_i + \gamma \geq f(\mathbf{x}, \mathbf{y}), \tag{3.14}$$

since $\mathbf{H}_i\mathbf{y} \geq 1$, $\gamma \geq 0$, and $\gamma \geq c_i$. We can continue to set components of $\mathbf{x}$ and $\mathbf{y}$ to 0 until reaching a binary feasible point $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ for which $\bar{\mathbf{x}}^\mathsf{T}\mathbf{H}\bar{\mathbf{y}} = 0$ and $f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \geq f(\mathbf{x}, \mathbf{y})$. This completes the proof of the first claim in the proposition.

Now, if the optimal objective value in (3.6) is at least $\gamma(\ell_a + \ell_b)$, then by the first part of the proposition, we may find an optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ satisfying (3.4); hence, $(\mathbf{x}^*, \mathbf{y}^*)$ is feasible in (3.5). Since (3.6) is a relaxation of (3.5), $(\mathbf{x}^*, \mathbf{y}^*)$ is optimal in (3.5). Hence, the partition $(\mathcal{A}, \mathcal{S}, \mathcal{B})$ defined by (3.9) is optimal in (3.1). This completes the proof. ∎

Algorithm 3.1 represents the procedure used in the proof of Proposition 3.1 to move from a feasible point in (3.6) to a feasible point $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ satisfying (3.8).

REMARK 3.1. *There is typically an abundance of feasible points in* (3.6) *satisfying* (3.7)*. For example, in the common case where* $\gamma = c_i = w_i = 1$ *for each* i*,* (3.7) *is satisfied whenever* $\mathbf{x}$ *and* $\mathbf{y}$ *are incidence vectors for a pair of feasible sets* $\mathcal{A}$ *and* $\mathcal{B}$ *in* (3.1)*, since in this case*

$$f(\mathbf{x}, \mathbf{y}) = \mathbf{c}^\mathsf{T}(\mathbf{x} + \mathbf{y}) = \mathbf{w}^\mathsf{T}\mathbf{x} + \mathbf{w}^\mathsf{T}\mathbf{y} \geq \ell_a + \ell_b = \gamma(\ell_a + \ell_b).$$

Now consider the following continuous bilinear program, which is obtained from (3.6) by relaxing the binary constraint $\mathbf{x}, \mathbf{y} \in \{0, 1\}^n$:

$$\max_{\mathbf{x},\mathbf{y} \in \mathbb{R}^n} \quad f(\mathbf{x}, \mathbf{y}) := \mathbf{c}^\mathsf{T}(\mathbf{x} + \mathbf{y}) - \gamma\mathbf{x}^\mathsf{T}\mathbf{H}\mathbf{y} \tag{3.15}$$

$$\text{subject to} \quad \mathbf{0} \leq \mathbf{x} \leq \mathbf{1}, \quad \mathbf{0} \leq \mathbf{y} \leq \mathbf{1}, \quad \ell_a \leq \mathbf{w}^\mathsf{T}\mathbf{x} \leq u_a, \quad \text{and} \quad \ell_b \leq \mathbf{w}^\mathsf{T}\mathbf{y} \leq u_b.$$

Input: A binary feasible point $(\mathbf{x}, \mathbf{y})$ for (3.6) satisfying (3.7)
while ($\mathbf{x}^\mathsf{T}\mathbf{H}\mathbf{y} > 0$)
  if ($\mathbf{1}^\mathsf{T}\mathbf{x} > \ell_a$)
    Choose i such that $x_i = 1$ and $\mathbf{H}_i\mathbf{y} \geq 1$.
    Set $x_i = 0$.
  else if ($\mathbf{1}^\mathsf{T}\mathbf{y} > \ell_b$)
    Choose i such that $y_i = 1$ and $\mathbf{H}_i\mathbf{x} \geq 1$.
    Set $y_i = 0$.
  end if
end while

ALGORITHM 3.1. *Convert a binary feasible point for* (3.6) *into a vertex separator without decreasing the objective function value.*

In [18], the authors study (3.15) in the common case where $\mathbf{c} \geq \mathbf{0}$ and $\mathbf{w} = \mathbf{1}$. In particular, the following theorem is proved:

THEOREM 3.2. [18, Theorem 2.1, Part 1] *If* (3.1) *is feasible,* $\mathbf{w} = \mathbf{1}$, $\mathbf{c} \geq \mathbf{0}$, $\gamma \geq \max\{c_i : i \in \mathcal{V}\} > 0$, *and the optimal objective value in* (3.1) *is at least* $\gamma(\ell_a + \ell_b)$, *then* (3.15) *has a binary optimal solution* $(\mathbf{x}, \mathbf{y}) \in \{0,1\}^{2n}$ *satisfying* (3.4).

In the proof of Theorem 3.2, a step-by-step procedure is given for moving from any feasible point $(\mathbf{x}, \mathbf{y})$ in (3.15) to a binary point $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ satisfying $f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \geq f(\mathbf{x}, \mathbf{y})$. Thus, when the vertices all have unit weights, the VSP may be solved with a 4-step procedure:

1. Obtain an optimal solution to the continuous bilinear program (3.15).
2. Move to a binary optimal solution using the algorithm of [18, Theorem 2.1, Part 1].
3. Convert the binary solution of (3.15) to a separator using Algorithm 3.1.
4. Construct an optimal partition via (3.9).

When G has a small number of vertices, the dimension of the bilinear program (3.15) is small, and the above approach may be very effective. However, since the objective function in (3.15) is non-concave, the number of local maximizers in (3.15) grows quickly as $|\mathcal{V}|$ becomes large and solving the bilinear program becomes increasingly difficult.

In order to find good approximate solutions to (3.15) when G is large, we will incorporate the 4-step procedure (with some modifications) into a multilevel framework (see Section 5). The basic idea is to coarsen the graph into a smaller graph having a similar structure to the original graph; the VSP is then solved for the coarse graph via a procedure similar to the one above, and the solution is uncoarsened to give a solution for the original graph.

At the coarser levels of our algorithm, each vertex represents an aggregate of vertices from the original graph. Hence, in order to keep track of the sizes of the aggregates, weights must be assigned to the vertices in the coarse graphs, which means the assumption of Theorem 3.2 that $\mathbf{w} = \mathbf{1}$ does not hold at the coarser levels. Indeed, when $\mathbf{c} \geq \mathbf{0}$ and $\mathbf{w} \neq \mathbf{1}$, (3.15) may not have a binary optimal solution, as we will show. However, in the general case where $\mathbf{w} > \mathbf{0}$ and $\mathbf{c} \in \mathbb{R}^n$, the following weaker result is obtained:

DEFINITION 3.3. *A point* $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{2n}$ *is called* mostly binary *if* $\mathbf{x}$ *and* $\mathbf{y}$ *each have at most one non-binary component.*

PROPOSITION 3.4. *If the VSP* (3.1) *is feasible and* $\gamma \in \mathbb{R}$, *then* (3.15) *has a mostly binary optimal solution.*

Proof. We show that the following stronger property holds:

(P) For any $(\mathbf{x}, \mathbf{y})$ feasible in (3.15), there exists a piecewise linear path to a feasible point $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in \mathbb{R}^{2n}$ which is mostly binary and satisfies $f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \geq f(\mathbf{x}, \mathbf{y})$.

Let $(\mathbf{x}, \mathbf{y})$ be any feasible point of (3.15). If $\mathbf{x}$ and $\mathbf{y}$ each have at most one non-binary component, then we are done. Otherwise, assume without loss of generality there exist indices $k \neq l$ such that

$$0 < x_k \leq x_l < 1.$$

Since $\mathbf{w} > \mathbf{0}$, we can define

$$\mathbf{x}(t) := \mathbf{x} + t\left(\frac{1}{w_k}\mathbf{e}_k - \frac{1}{w_l}\mathbf{e}_l\right)$$

for $t \in \mathbb{R}$. Substituting $\mathbf{x} = \mathbf{x}(t)$ in the objective function yields

$$f(\mathbf{x}(t), \mathbf{y}) = f(\mathbf{x}, \mathbf{y}) + td, \quad \text{where } d = \nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y})\left(\frac{1}{w_k}\mathbf{e}_k - \frac{1}{w_l}\mathbf{e}_l\right).$$

If $d \geq 0$, then we may increase t from zero until either $x_k(t) = 1$ or $x_l(t) = 0$. In the case where $d < 0$, we may decrease t until either $x_k(t) = 0$ or $x_l(t) = 1$. In either case, the number of non-binary components in $\mathbf{x}$ is reduced by at least one, while the objective value does not decrease by the choice of the sign of t. Feasibility is maintained since $\mathbf{w}^{\mathsf{T}}\mathbf{x}(t) = \mathbf{w}^{\mathsf{T}}\mathbf{x}$. We may continue moving components to bounds in this manner until $\mathbf{x}$ has at most one non-binary component. The same procedure may be applied to $\mathbf{y}$. In this way, we will arrive at a feasible point $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ such that $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ each have at most one non-binary component and $f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \geq f(\mathbf{x}, \mathbf{y})$. This proves (P), which completes the proof. ∎

The proof of Proposition 3.4 was constructive. A nonconstructive proof goes as follows: Since the quadratic program (3.15) is bilinear, there exists an optimal solution lying at an extreme point [27]. At an extreme point of the feasible set of (3.15), exactly $2n$ linearly independent constraints are active. Since there can be at most n linearly independent constraints which are active at $\mathbf{x}$, and similarly for $\mathbf{y}$, there must exist exactly n linearly independent constraints which are active at $\mathbf{x}$; in particular, at least $n - 1$ components of $\mathbf{x}$ must lie at a bound, and similarly for $\mathbf{y}$. Therefore, $(\mathbf{x}, \mathbf{y})$ is mostly binary. In the case where $\mathbf{w} \neq \mathbf{1}$, there may exist extreme points of the feasible set which are not binary; for example, consider $n = 3$, $\ell_a = \ell_b = 1$, $u_a = u_b = 2$, $\mathbf{w} = (1, 1, 2)$, $\mathbf{x} = (1, 0, 0.5)$, and $\mathbf{y} = (0, 1, 0.5)$.

Often, the conclusion of Proposition 3.4 can be further strengthened to assert the existence of a solution $(\mathbf{x}, \mathbf{y})$ of (3.15) for which either $\mathbf{x}$ or $\mathbf{y}$ is completely binary, while the other variable has at most one nonbinary component. The rationale is the following: Suppose that $(\mathbf{x}, \mathbf{y})$ is a mostly binary optimal solution and without loss of generality x_i is a nonbinary component of $\mathbf{x}$. Substituting $\mathbf{x}(t) = \mathbf{x} + t\mathbf{e}_i$ in the objective function we obtain

$$f(\mathbf{x}(t), \mathbf{y}) = f(\mathbf{x}, \mathbf{y}) + td, \quad d = \nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y})\mathbf{e}_i.$$

If $d \geq 0$, we increase t, while if $d < 0$, we decrease t; in either case, the objective function $f(\mathbf{x}(t), \mathbf{y})$ cannot decrease. If

$$\ell_a + w_i \leq \mathbf{w}^{\mathsf{T}}\mathbf{x} \leq u_a - w_i, \tag{3.16}$$

Input: A feasible point $(\mathbf{x}, \mathbf{y})$ for the continuous bilinear program (3.15).
while ($\mathbf{x}$ has at least 2 nonbinary components)
  Choose $i, j \in \mathcal{V}$ such that $x_i, x_j \in (0,1)$.
  Update $\mathbf{x} \leftarrow \mathbf{x} + t(\frac{1}{w_i}\mathbf{e}_i - \frac{1}{w_j}\mathbf{e}_j)$, choosing t to ensure that:
    (a) $f(\mathbf{x}, \mathbf{y})$ does not decrease,
    (b) either $x_i \in \{0,1\}$ or $x_j \in \{0,1\}$,
    (c) $\mathbf{x}$ feasible in (3.15).
end while
while ($\mathbf{y}$ has at least 2 nonbinary components)
  Choose $i, j \in \mathcal{V}$ such that $y_i, y_j \in (0,1)$.
  Update $\mathbf{y} \leftarrow \mathbf{y} + t(\frac{1}{w_i}\mathbf{e}_i - \frac{1}{w_j}\mathbf{e}_j)$, choosing t to ensure that:
    (a) $f(\mathbf{x}, \mathbf{y})$ does not decrease,
    (b) either $y_i \in \{0,1\}$ or $y_j \in \{0,1\}$,
    (c) $\mathbf{y}$ feasible in (3.15).
end while

ALGORITHM 3.2. *Convert a feasible point for* (3.15) *into a mostly binary feasible point without decreasing the objective value.*

then we can let t grow in magnitude until either $x_i(t) = 0$ or $x_i(t) = 1$, while complying with the bounds $\ell_a \le \mathbf{w}^\mathsf{T}\mathbf{x}(t) \le u_a$. In applications, either the inequality (3.16) holds, or an analogous inequality $\ell_b + w_j \le \mathbf{w}^\mathsf{T}\mathbf{y} \le u_a - w_j$ holds for $\mathbf{y}$, where y_j is a nonbinary component of $\mathbf{y}$. The reason that one of these inequalities holds is that we typically have $u_a = u_b > \mathcal{W}(\mathcal{V})/2$, which implies that the upper bounds $\mathbf{w}^\mathsf{T}\mathbf{x} \le u_a$ and $\mathbf{w}^\mathsf{T}\mathbf{y} \le u_b$ cannot be simultaneously active. On the other hand, the lower bounds $\mathbf{w}^\mathsf{T}\mathbf{x} \ge \ell_a$ and $\mathbf{w}^\mathsf{T}\mathbf{y} \ge \ell_b$ are often trivially satisfied when ℓ_a and ℓ_b are small numbers like one.

Algorithm 3.2 represents the procedure used in the proof of Proposition 3.4 to convert a given feasible point for (3.15) into a mostly binary feasible point without decreasing the objective function value. In the case where $\mathbf{w} = \mathbf{1}$, the final point returned by Algorithm 3.2 is binary. Although the continuous bilinear problem (3.15) is not necessarily equivalent to the discrete VSP (3.1) when $\mathbf{w} \ne \mathbf{1}$, it closely approximates (3.1) in the sense it has a mostly binary optimal solution. Since (3.15) is a relaxation of (3.5), the objective value at an optimal solution to (3.15) gives an upper bound on the optimal objective value in (3.5), and therefore on the optimal objective value in (3.1). On the other hand, given a mostly binary solution to (3.15), we can typically push the remaining fractional components to bounds without violating the constraints on $\mathbf{w}^\mathsf{T}\mathbf{x}$ and $\mathbf{w}^\mathsf{T}\mathbf{y}$. Then we apply Algorithm 3.1 to this binary point to obtain a feasible point in (3.5), giving a *lower* bound on the optimal objective value in (3.5) and (3.1). In the case where $\mathbf{w} = \mathbf{1}$, the upper and lower bounds are equal.

4. Solving the bilinear program. In our implementation of the multilevel algorithm, we use an iterative optimization algorithm to compute a local maximizer of (3.15), then employ two different techniques to escape from a local optimum. Our optimization algorithm is a modified version of a Mountain Climbing Algorithm originally proposed by Konno [27] for solving bilinear programs.

4.1. Mountain Climbing. Given an initial guess, the Mountain Climbing Algorithm of [27] solves (3.15) by alternately holding $\mathbf{x}$ or $\mathbf{y}$ fixed while optimizing over the other variable. This

Input: A feasible point $(\mathbf{x}, \mathbf{y})$ for (3.15) and $\eta > 0$.
while ($(\mathbf{x}, \mathbf{y})$ not stationary point for (3.15))
  $\hat{\mathbf{x}} \leftarrow \text{argmax}\,\{f(\mathbf{x}, \mathbf{y}) : \mathbf{x} \in \mathcal{P}_a\}$
  $\hat{\mathbf{y}} \leftarrow \text{argmax}\,\{f(\mathbf{x}, \mathbf{y}) : \mathbf{y} \in \mathcal{P}_b\}$
  if ($f(\hat{\mathbf{x}}, \hat{\mathbf{y}}) > \max\{f(\hat{\mathbf{x}}, \mathbf{y}), f(\mathbf{x}, \hat{\mathbf{y}})\} + \eta$)
    $(\mathbf{x}, \mathbf{y}) \leftarrow (\hat{\mathbf{x}}, \hat{\mathbf{y}})$
  else if ($f(\hat{\mathbf{x}}, \mathbf{y}) > f(\mathbf{x}, \hat{\mathbf{y}})$)
    $\mathbf{x} \leftarrow \hat{\mathbf{x}}$
  else
    $\mathbf{y} \leftarrow \hat{\mathbf{y}}$
  end if
end while
return $(\mathbf{x}, \mathbf{y})$

ALGORITHM 4.1. **MCA:** *A modified version of Konno's Mountain Climbing Algorithm for generating a stationary point for* (3.15).

optimization problem for a single variable can be done efficiently since the program is linear in $\mathbf{x}$ and in $\mathbf{y}$. In our modified version of the mountain climbing algorithm, which we call MCA (see Algorithm 4.1), we maximize over $\mathbf{x}$ with $\mathbf{y}$ held fixed to obtain $\hat{\mathbf{x}}$, we maximize over $\mathbf{y}$ with $\mathbf{x}$ held fixed to obtain $\hat{\mathbf{y}}$, and then we maximize over the subspace spanned by the two maximizers. Due to the bilinear structure of the objective function, the subspace maximum is either $f(\hat{\mathbf{x}}, \hat{\mathbf{y}})$, $f(\hat{\mathbf{x}}, \mathbf{y})$, or $f(\mathbf{x}, \hat{\mathbf{y}})$. The step $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ is only taken if it provides an improvement of at least η more than either an $\hat{\mathbf{x}}$ or $\hat{\mathbf{y}}$ step, where η is a small constant (10^{-5} in our experiments). After an $\mathbf{x}$ or $\mathbf{y}$ step is taken, the subspace maximizer alternates between $(\hat{\mathbf{x}}, \mathbf{y})$ and $(\mathbf{x}, \hat{\mathbf{y}})$, and hence only one linear program is solved at each iteration. In our statement of MCA, $\mathcal{P}_a$ and $\mathcal{P}_b$ denote the polyhedral feasible sets for (3.15) defined by

$$\mathcal{P}_i = \{\mathbf{z} \in \mathbb{R}^n : \mathbf{0} \le \mathbf{z} \le \mathbf{1} \text{ and } \ell_i \le \mathbf{w}^\mathsf{T}\mathbf{z} \le u_i\}, \quad i = a, b.$$

The linear programs arising in MCA are solved using a greedy algorithm. In particular, $\max\{f(\mathbf{x}, \mathbf{y}) : \mathbf{x} \in \mathcal{P}_a\}$ is solved by setting all components x_i equal to zero and then sorting the components in decreasing order of their ratio $r_i := \frac{\partial f}{\partial x_i}(\mathbf{x}, \mathbf{y})/w_i$; the components are then visited in order and are pushed up from 0 to 1 until either $\mathbf{w}^\mathsf{T}\mathbf{x} = u_a$ or a component is reached such that $r_i < 0$. It is easy to show that this procedure gives an optimal solution to the LP.

Since (3.15) is non-concave, it may have many stationary points; thus, it is crucial to incorporate techniques to escape from a stationary point and explore a new part of the solution space. The techniques we develop are based on perturbations in either the cost vector $\mathbf{c}$ or in the penalty parameter γ. We make the smallest possible perturbations which guarantee that the current iterate is no longer a stationary point of the perturbed problem. After computing an approximate solution of the perturbed problem, we use it as a starting guess in the original problem and reapply MCA. If we reach a better solution for the original problem, then we save this best solution, and make another perturbation. If we do not reach a better solution, then we continue the perturbation process, starting from the new point.

4.2. c-perturbations. Our perturbations of the cost vector are based on an analysis of the first-order optimality conditions for the maximization problem (3.15). If a feasible point $(\mathbf{x}, \mathbf{y})$ for

(3.15) is a local maximizer, then the objective function can only decrease when we make small moves in the direction of other feasible points, or equivalently,

$$\nabla_{\mathbf{x}} f(\mathbf{x},\mathbf{y})(\tilde{\mathbf{x}}-\mathbf{x}) + \nabla_{\mathbf{y}} f(\mathbf{x},\mathbf{y})(\tilde{\mathbf{y}}-\mathbf{y}) \le 0 \tag{4.1}$$

whenever $(\tilde{\mathbf{x}},\tilde{\mathbf{y}})$ is feasible in (3.15). Another way to state the first-order optimality condition (4.1) employs multipliers for the constraints. In particular, by [33, Theorem 12.1], a feasible point $(\mathbf{x},\mathbf{y})$ for (3.15) is a local maximizer only if there exist multipliers $\boldsymbol{\mu}^a \in \mathcal{M}(\mathbf{x})$, $\boldsymbol{\mu}^b \in \mathcal{M}(\mathbf{y})$, $\lambda^a \in \mathcal{L}(\mathbf{x},\ell_a,u_a)$, $\lambda^b \in \mathcal{L}(\mathbf{y},\ell_b,u_b)$ such that

$$\begin{bmatrix} \nabla_{\mathbf{x}} f(\mathbf{x},\mathbf{y}) \\ \nabla_{\mathbf{y}} f(\mathbf{x},\mathbf{y}) \end{bmatrix} + \begin{bmatrix} \boldsymbol{\mu}^a \\ \boldsymbol{\mu}^b \end{bmatrix} + \begin{bmatrix} \lambda^a \mathbf{w} \\ \lambda^b \mathbf{w} \end{bmatrix} = \mathbf{0}, \tag{4.2}$$

where

$$\begin{aligned} \mathcal{M}(\mathbf{z}) &= \{\boldsymbol{\mu} \in \mathbb{R}^n : \mu_i z_i \le \min\{\mu_i, 0\} \text{ for all } 1 \le i \le n\} \quad \text{and} \\ \mathcal{L}(\mathbf{z},\ell,u) &= \{\lambda \in \mathbb{R} : \lambda \mathbf{w}^{\mathsf{T}}\mathbf{z} \le \min\{\lambda u, \lambda \ell\}\}. \end{aligned}$$

The conditions (4.1) and (4.2) are equivalent. The condition (4.2) is often called the KKT (Karush-Kuhn-Tucker) condition. The usual formulation of the KKT conditions involves introducing a separate multiplier for each upper and lower bound constraint, which leads to eight different multipliers in the case of (3.15). In (4.2) the number of multipliers has been reduced to four through the use of the set $\mathcal{M}$ and $\mathcal{L}$.

In describing our perturbations to the objective function, we attach a subscript to f to indicate the parameter that is being perturbed. Thus $f_{\mathbf{c}}$ denotes the original objective in (3.15), while $f_{\tilde{\mathbf{c}}}$ corresponds to the objective obtained by replacing $\mathbf{c}$ by $\tilde{\mathbf{c}}$.

PROPOSITION 4.1. *If $\gamma \in \mathbb{R}$ and $(\mathbf{x},\mathbf{y})$ satisfies the first-order optimality condition* (4.2) *for* (3.15)*, then in any of the following cases, for any choice of $\epsilon > 0$ and for the indicated choices of $\tilde{\mathbf{c}}$, $(\mathbf{x},\mathbf{y})$ does not satisfy the first-order optimality condition* (4.2) *for $f = f_{\tilde{\mathbf{c}}}$:*

1. *For any $i \ne j$ such that $\mu_i^a = \mu_j^a = 0$, $x_i < 1$, and $x_j > 0$, take*

$$\tilde{c}_k = \begin{cases} c_k + \epsilon & \text{if } k = i, \\ c_k - \epsilon & \text{if } k = j, \\ c_k & \text{otherwise} \end{cases}.$$

2. *If $\lambda^a = 0$ and $\mathbf{w}^{\mathsf{T}}\mathbf{x} < u_a$, then for any i such that $\mu_i^a = 0$ and $x_i < 1$, take $\tilde{c}_i = c_i + \epsilon$ and $\tilde{c}_k = c_k$ for $k \ne i$.*
3. *If $\lambda^a = 0$ and $\mathbf{w}^{\mathsf{T}}\mathbf{x} > \ell_a$, then for any j such that $\mu_j^a = 0$ and $x_j > 0$, take $\tilde{c}_j = c_j - \epsilon$ and $\tilde{c}_k = c_k$ for $k \ne j$.*

Proof. **Part 1.** Let i and j satisfy the stated conditions and define the vector $\mathbf{d} = w_j \mathbf{e}_i - w_i \mathbf{e}_j$. Since $\mathbf{w}^{\mathsf{T}}\mathbf{d} = 0$, $x_i < 1$, and $x_j > 0$, it follows that $\tilde{\mathbf{x}}(t) = \mathbf{x} + t\mathbf{d}$ is feasible in (3.15) for $t > 0$ sufficiently small. Moreover, by (4.2) and the assumption $\mu_i^a = \mu_j^a = 0$, we have $\nabla_{\mathbf{x}} f_{\mathbf{c}}(\mathbf{x},\mathbf{y})\mathbf{d} = 0$. Since

$$\nabla_{\mathbf{x}} f_{\tilde{\mathbf{c}}}(\mathbf{x},\mathbf{y}) = \nabla_{\mathbf{x}} f_{\mathbf{c}}(\mathbf{x},\mathbf{y}) + \epsilon(\mathbf{e}_i^{\mathsf{T}} - \mathbf{e}_j^{\mathsf{T}}),$$

we conclude that $\nabla_{\mathbf{x}} f_{\tilde{\mathbf{c}}}(\mathbf{x},\mathbf{y})\mathbf{d} = \epsilon(w_i + w_j)$, which implies that

$$\nabla_{\mathbf{x}} f_{\tilde{\mathbf{c}}}(\mathbf{x},\mathbf{y})(\tilde{\mathbf{x}}(t) - \mathbf{x}) = \epsilon t(w_i + w_j) > 0. \tag{4.3}$$

Input: A feasible point $(\mathbf{x}, \mathbf{y})$ for (3.15).
$(\mathbf{x}, \mathbf{y}) \leftarrow$ MCA $(\mathbf{x}, \mathbf{y})$
loop
  $\tilde{\mathbf{c}} \leftarrow$ perturb $(\mathbf{c})$
  $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \leftarrow$ MCA $(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{c}})$
  $(\mathbf{x}^*, \mathbf{y}^*) \leftarrow$ MCA $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \mathbf{c})$
  if ($f(\mathbf{x}^*, \mathbf{y}^*) > f(\mathbf{x}, \mathbf{y})$)
    $(\mathbf{x}, \mathbf{y}) \leftarrow (\mathbf{x}^*, \mathbf{y}^*)$
  else
    break
  end if
end loop
return $(\mathbf{x}, \mathbf{y})$

ALGORITHM 4.2. **MCA_CP:** *A modification of MCA which incorporates* $\mathbf{c}$*-perturbations.*

This shows that the first-order optimality condition (4.1) is not satisfied at $(\mathbf{x}, \mathbf{y})$ for $f = f_{\tilde{\mathbf{c}}}$.

Part 2. Define $\mathbf{d} = \mathbf{e}_i$. Since $\mathbf{w}^\mathsf{T}\mathbf{x} < u^a$ and $x_i < 1$, it follows that $\tilde{\mathbf{x}}(t) = \mathbf{x} + t\mathbf{d}$ is feasible in (3.15) for $t > 0$ sufficiently small. Since $\lambda^a = \mu_i^a = 0$, (4.2) implies that $\nabla_{\mathbf{x}} f_{\mathbf{c}}(\mathbf{x}, \mathbf{y})\mathbf{d} = 0$. So,

$$\nabla_{\mathbf{x}} f_{\tilde{\mathbf{c}}}(\mathbf{x}, \mathbf{y})\mathbf{d} = [\nabla_{\mathbf{x}} f_{\mathbf{c}}(\mathbf{x}, \mathbf{y}) + \epsilon \mathbf{e}_i^\mathsf{T}]\mathbf{d} = 0 + \epsilon \mathbf{e}_i^\mathsf{T}\mathbf{d} = \epsilon,$$

which implies $\nabla_{\mathbf{x}} f_{\tilde{\mathbf{c}}}(\mathbf{x}, \mathbf{y})(\tilde{\mathbf{x}}(t) - \mathbf{x}) = \epsilon t > 0$. Hence, the first-order optimality conditions (4.1) are not satisfied at $(\mathbf{x}, \mathbf{y})$ for $f_{\tilde{\mathbf{c}}}$.

Part 3. The analysis parallels the analysis of Part 2. □

Of course, Proposition 4.1 may be applied to either $\mathbf{x}$ or $\mathbf{y}$. Since ϵ was arbitrary, we usually take ϵ to be a tiny positive number (10^{-6} in our experiments). By making a tiny change in the problem, the iterates of the optimization algorithm MCA applied to $f = f_{\tilde{\mathbf{c}}}$ must move away from the current point to a new vertex of the feasible set to improve the objective value in the perturbed problem. Then the solution of the slightly perturbed problem is used as a starting guess for the solution of the original unperturbed problem. Algorithm 4.2, also denoted MCA_CP, incorporates the $\mathbf{c}$-perturbations into MCA. In the figure, the notation MCA $(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{c}})$ indicates that the MCA algorithm is applied to the point $(\mathbf{x}, \mathbf{y})$ using $\tilde{\mathbf{c}}$ in place of $\mathbf{c}$ as the vector of vertex costs. In our experiments, the c-perturbations were performed in the following way: For each i such that $|\mu_i^a| < 10^{-5}$, we set $\tilde{c}_i = c_i + \epsilon$ whenever $x_i < 0.5$ and $\tilde{c}_i = c_i - \epsilon$ otherwise; similar perturbations are made based on the values of μ_i^b and y_i.

4.3. γ-perturbations. Next, we consider perturbations in the parameter γ. According to our theory, we need to take $\gamma \geq \max\{c_i : i \in \mathcal{V}\}$ to ensure an (approximate) equivalence between the discrete (3.1) and the continuous (3.15) VSP. The penalty term $-\gamma \mathbf{x}^\mathsf{T}\mathbf{H}\mathbf{y}$ in the objective function of (3.15) enforces the constraints $\mathcal{A} \cap \mathcal{B} = \emptyset$ and $(\mathcal{A} \times \mathcal{B}) \cap \mathcal{E} = \emptyset$ of (3.1). Thus, by decreasing γ, we relax our enforcement of these constraints and place greater emphasis on the cost of the separator. The next proposition will determine the amount by which we must decrease γ in order to ensure that the current point $(\mathbf{x}, \mathbf{y})$, a local maximizer of f_γ, is no longer a local maximizer of the perturbed problem $f_{\tilde{\gamma}}$. The derivation requires a formulation of the second-order necessary and sufficient optimality conditions given in [17, Cor. 3.3]; applying these conditions to the bilinear program (3.15), we have the following theorem.

THEOREM 4.2. *If $\gamma \in \mathbb{R}$ and $(\mathbf{x}, \mathbf{y})$ is feasible in (3.15), then $(\mathbf{x}, \mathbf{y})$ is a local maximizer if and only if the following hold:*

(C1) $\nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y})\mathbf{d} \le 0$ *for every* $\mathbf{d} \in \mathcal{F}_a(\mathbf{x}) \cap \mathcal{D}$,
(C2) $\nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})\mathbf{d} \le 0$ *for every* $\mathbf{d} \in \mathcal{F}_b(\mathbf{y}) \cap \mathcal{D}$, *and*
(C3) $\mathbf{d}_1^{\mathsf{T}}(\nabla^2 f)\mathbf{d}_2 \le 0$ *for every* $\mathbf{d}_1, \mathbf{d}_2 \in \mathcal{C}(\mathbf{x}, \mathbf{y}) \cap \mathcal{G}$,

where

$$\mathcal{F}_i(\mathbf{z}) = \left\{ \mathbf{d} \in \mathbb{R}^n : \begin{array}{l} d_j \le 0 \text{ for all } j \text{ such that } z_j = 1 \\ d_j \ge 0 \text{ for all } j \text{ such that } z_j = 0 \\ \mathbf{w}^{\mathsf{T}}\mathbf{d} \le 0 \text{ if } \mathbf{w}^{\mathsf{T}}\mathbf{z} = u_i \\ \mathbf{w}^{\mathsf{T}}\mathbf{d} \ge 0 \text{ if } \mathbf{w}^{\mathsf{T}}\mathbf{z} = l_i \end{array} \right\}, \quad i = a, b,\ \mathbf{z} \in \mathbb{R}^n,$$

$$\mathcal{C}(\mathbf{x}, \mathbf{y}) = \{\mathbf{d} \in \mathcal{F}_a(\mathbf{x}) \times \mathcal{F}_b(\mathbf{y}) : \nabla f(\mathbf{x}, \mathbf{y})\mathbf{d} = 0\},$$

$$\mathcal{D} = \bigcup_{i,j=1}^{n} \{\mathbf{e}_i, -\mathbf{e}_i, w_j\mathbf{e}_i - w_i\mathbf{e}_j\}, \quad \text{and} \quad \mathcal{G} = (\mathcal{D} \times \{\mathbf{0}\}) \cup (\{\mathbf{0}\} \times \mathcal{D}). \tag{4.4}$$

The sets $\mathcal{F}_a$ and $\mathcal{F}_b$ are the cones of first-order feasible directions at $\mathbf{x}$ and $\mathbf{y}$. The set $\mathcal{G}$ is a reflective edge description of the feasible set, introduced in [17]; that is, each edge of the constraint polyhedron of (3.15) is parallel to an element of $\mathcal{G}$. Since $\mathcal{D}$ is a finite set, checking the first-order optimality conditions reduces to testing the conditions (C1) and (C2) for the finite collection of elements from $\mathcal{D}$ that are in the cone of first-order feasible directions; testing the second-order optimality conditions reduces to testing the condition (C3) for the elements from $\mathcal{G}$ that are in the critical cone $\mathcal{C}(\mathbf{x}, \mathbf{y})$.

PROPOSITION 4.3. *Let $\gamma \in \mathbb{R}$, let $(\mathbf{x}, \mathbf{y})$ be a feasible point in (3.15) which satisfies the first-order optimality condition* (C1)*, and let $\tilde{\gamma} \le \gamma$.*

1. *Suppose $\ell_a < \mathbf{w}^{\mathsf{T}}\mathbf{x} < u_a$. Then* (C1) *holds at $(\mathbf{x}, \mathbf{y})$ for $f = f_{\tilde{\gamma}}$ if and only if $\tilde{\gamma} \ge \alpha_1$, where*

$$\alpha_1 := \max \left\{ \frac{c_j}{\mathbf{H}_j\mathbf{y}} : j \in \mathcal{J} \right\} \text{ and } \mathcal{J} := \{j : x_j < 1 \text{ and } \mathbf{H}_j\mathbf{y} > 0\},$$

   with $\alpha_1 := -\infty$ if $\mathcal{J} = \emptyset$.
2. *Suppose $\mathbf{w}^{\mathsf{T}}\mathbf{x} = u_a$. Then* (C1) *holds at $(\mathbf{x}, \mathbf{y})$ for $f = f_{\tilde{\gamma}}$ if and only if $\tilde{\gamma} \ge \alpha_2$ where*

$$\alpha_2 := \inf\ \Gamma \text{ and } \Gamma := \left\{ \alpha \in \mathbb{R} : \frac{1}{w_i}\frac{\partial f_\alpha}{\partial x_i}(\mathbf{x}, \mathbf{y}) \le \frac{1}{w_j}\frac{\partial f_\alpha}{\partial x_j}(\mathbf{x}, \mathbf{y}) \text{ for all } x_i < 1 \text{ and } x_j > 0 \right\}.$$

Proof. **Part 1.** Since $\ell_a < \mathbf{w}^{\mathsf{T}}\mathbf{x} < u_a$, the cone of first-order feasible directions for $\mathbf{x}$ is given by

$$\mathcal{F}_a(\mathbf{x}) = \{\mathbf{d} \in \mathbb{R}^n : d_i \ge 0 \text{ when } x_i = 0 \text{ and } d_i \le 0 \text{ when } x_i = 1,\ i = 1, \ldots, n\}.$$

It follows that for each $i = 1, \ldots, n$,

$$\begin{array}{rcl} \mathbf{e}_i & \in & \mathcal{F}_a(\mathbf{x}) \text{ if and only if } x_i < 1, \\ -\mathbf{e}_i & \in & \mathcal{F}_a(\mathbf{x}) \text{ if and only if } x_i > 0, \\ (w_j\mathbf{e}_i - w_i\mathbf{e}_j) & \in & \mathcal{F}_a(\mathbf{x}) \text{ if and only if } x_i < 1 \text{ and } x_j > 0. \end{array}$$

Hence, the first-order optimality condition (C1) for $f_{\tilde{\gamma}}$ can be expressed as follows:

$$\nabla_{\mathbf{x}} f_{\tilde{\gamma}}(\mathbf{x},\mathbf{y})\mathbf{e}_i \le 0 \ \text{ when } \ x_i < 1, \tag{4.5}$$

$$\nabla_{\mathbf{x}} f_{\tilde{\gamma}}(\mathbf{x},\mathbf{y})\mathbf{e}_i \ge 0 \ \text{ when } \ x_i > 0, \tag{4.6}$$

$$\nabla_{\mathbf{x}} f_{\tilde{\gamma}}(\mathbf{x},\mathbf{y})(w_j\mathbf{e}_i - w_i\mathbf{e}_j) \le 0 \ \text{ when } \ x_i < 1 \text{ and } x_j > 0. \tag{4.7}$$

Since (4.7) is implied by (4.5) and (4.6), it follows that (C1) holds if and only if (4.5) and (4.6) hold. Since (C1) holds for f_γ, we know that

$$\nabla_{\mathbf{x}} f_{\gamma}(\mathbf{x},\mathbf{y})\mathbf{e}_i = c_i - \gamma\mathbf{H}_i\mathbf{y} \ge 0 \ \text{ when } \ x_i > 0.$$

Hence, since $\tilde{\gamma} \le \gamma$ and $\mathbf{H}_i\mathbf{y} \ge 0$,

$$\nabla_{\mathbf{x}} f_{\tilde{\gamma}}(\mathbf{x},\mathbf{y})\mathbf{e}_i = c_i - \tilde{\gamma}\mathbf{H}_i\mathbf{y} \ge 0 \quad \text{when } x_i > 0.$$

Hence, (C1) holds with respect to $\tilde{\gamma}$ if and only if (4.5) holds. Since (C1) holds for $f = f_\gamma$, we have

$$c_j - \gamma\mathbf{H}_j\mathbf{y} \le 0 \text{ when } x_j < 1. \tag{4.8}$$

Hence, for every j such that $x_j < 1$ and $\mathbf{H}_j\mathbf{y} = 0$, we have

$$c_j - \tilde{\gamma}\mathbf{H}_j\mathbf{y} = c_j = c_j - \gamma\mathbf{H}_j\mathbf{y} \le 0.$$

So, (4.5) holds if and only if $c_j - \tilde{\gamma}\mathbf{H}_j\mathbf{y} \le 0$ for every $j \in \mathcal{J}$; that is, if and only if $\tilde{\gamma} \ge \alpha_1$. This completes the proof of Part 1.

Part 2. Since $\mathbf{w}^{\mathsf{T}}\mathbf{x} = u_a$, the cone of first-order feasible directions at $\mathbf{x}$ has the constraint $\mathbf{w}^{\mathsf{T}}\mathbf{d} \le 0$. Consequently, $\mathbf{e}_i \notin \mathcal{F}_a(\mathbf{x}) \cap \mathcal{D}$ for any i, and the first-order optimality condition (C1) for $f_{\tilde{\gamma}}$ reduces to (4.6)–(4.7). As in Part 1, (4.6) holds, since $\tilde{\gamma} \le \gamma$. Condition (4.7) is equivalent to $\tilde{\gamma} \in \Gamma$. Since (4.7) holds for $f = f_\gamma$, we have $\gamma \in \Gamma$. Since $\nabla_{\mathbf{x}} f_{\tilde{\gamma}}(\mathbf{x},\mathbf{y})$ is a affine function of $\tilde{\gamma}$, the set of $\tilde{\gamma}$ satisfying (4.7) for some i and j such that $x_j > 0$ and $x_i < 1$ is a closed interval, and the intersection of the intervals over all i and j for which $x_j > 0$ and $x_i < 1$ is also a closed interval. Hence, since $\tilde{\gamma} \le \gamma \in \Gamma$, we have $\tilde{\gamma} \in \Gamma$ if and only if $\tilde{\gamma} \ge \alpha_2$. This completes the proof of Part 2. ∎

Remark 4.1: Of course, Proposition 4.3 also holds when the variables $\mathbf{x}$ and $\mathbf{y}$ and the bounds (ℓ_a, u_a) and (ℓ_b, u_b) are interchanged. In most applications, u_a and $u_b > \mathcal{W}(\mathcal{V})/2$, $\ell_a = \ell_b = 1$, and as the iterates converge to a solution of (3.15), either the constraint $\mathbf{w}^{\mathsf{T}}\mathbf{x} \le u_a$ or the constraint $\mathbf{w}^{\mathsf{T}}\mathbf{y} \le u_b$ is active. Thus, for a given iterate $(\mathbf{x},\mathbf{y})$, the assumptions of Part 1 typically apply to either $\mathbf{x}$ or $\mathbf{y}$, while the assumptions of Part 2 apply to the other variable. Although the Part 2 condition seems complex, it often provides no useful information in the following sense: In a multilevel implementation, the vertex costs (and weights) are often 1 at the finest level, and at coarser levels, the vertex costs may not differ greatly. When the vertex costs are equal, (4.7) holds when $\tilde{\gamma}$ has the same sign as γ; that is, as long as $\tilde{\gamma} \ge 0$. Thus, $\alpha_2 = 0$. Since α_1 is typically positive, the tighter bound on $\tilde{\gamma}$ is the interval $[\alpha_1, \gamma]$, which means that when $\tilde{\gamma} < \alpha_1$, $(\mathbf{x},\mathbf{y})$ is no longer a stationary point for $f = f_{\tilde{\gamma}}$.

Algorithm 4.3, also denoted MCA_GR, approximately solves (3.15), while incorporating both **c**-perturbations and γ-refinements. Here, the notation MCA$(\mathbf{x},\mathbf{y},\tilde{\gamma})$ indicates that the MCA algorithm is applied to the point $(\mathbf{x},\mathbf{y})$ using $\tilde{\gamma}$ in place of γ as the penalty parameter. In our implementation, the γ-refinements are performed in the following way: γ is reduced from the initial value α_1 in 10 uniform decrements until it reaches 0 or the optimal objective value improves. We note that in 4.3, the **c**-perturbations are embedded in the γ-refinement procedure in order to obtain a local optimizer of high quality for each $\tilde{\gamma}$.

Input: A feasible point $(\mathbf{x}, \mathbf{y})$ for (3.15).
$(\mathbf{x}, \mathbf{y}) \leftarrow$ MCA_CP $(\mathbf{x}, \mathbf{y})$
$\tilde{\gamma} \leftarrow \alpha_1$
while ($\tilde{\gamma} > 0$)
    $\tilde{\gamma} \leftarrow$ reduce $(\tilde{\gamma})$
    $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \leftarrow$ MCA_CP $(\mathbf{x}, \mathbf{y}, \tilde{\gamma})$
    $(\mathbf{x}^*, \mathbf{y}^*) \leftarrow$ MCA_CP $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \gamma)$
    if ($f(\mathbf{x}^*, \mathbf{y}^*) > f(\mathbf{x}, \mathbf{y})$)
        $(\mathbf{x}, \mathbf{y}) \leftarrow (\mathbf{x}^*, \mathbf{y}^*)$
        $\tilde{\gamma} \leftarrow \alpha_1$
end while
return $(\mathbf{x}, \mathbf{y})$

ALGORITHM 4.3. **MCA_GR:** *A refinement algorithm for* (3.15) *which incorporates* **c***-perturbations and* γ*-refinements.*

5. Multilevel algorithm. We now give an overview of a multilevel algorithm, which we call BLP, for solving the vertex separator problem. The algorithm consists of three phases: coarsening, solving, and uncoarsening.

Coarsening. Vertices are visited one by one and each vertex is matched with an unmatched adjacent vertex, whenever one exists. Matched vertices are merged together to form a single vertex having a cost and weight equal to the sum of the costs and weights of the constituent vertices. Multiple edges which arise between two vertices are combined and assigned an edge weight equal to the sum of the weights of combined edges (in the original graph, all edges are assumed to have weight 1). This coarsening process repeats until the graph has fewer than 75 vertices or fewer than 10 edges.

The goal of the coarsening phase is to reduce the number of degrees of freedom in the problem, while preserving its structure so that the solutions obtained for the coarse problems give a good approximation to the solution for the original problem. We consider two matching rules: random and heavy-edge. In heavy-edge based matching, each vertex is matched with an unmatched neighbor such that the edge between them has the greatest weight over all unmatched neighbors. Heavy edge matching rules have been used in multilevel algorithms such as [19, 22], and were originally developed for edge-cut problems. In our initial experiments, we also considered a third rule based on an *algebraic distance* [8] between vertices. However, the results were not significantly different from heavy-edge matching, which is not surprising, since (like heavy-edge rules) the algebraic distance was originally developed for minimizing edge-cuts.

Solving. For each of the graphs in the multilevel hierarchy, we approximately solve (3.15) using MCA_GR. For the coarsest graph, the starting guess is $x_i = u_a/\mathcal{W}(\mathcal{V})$ and $y_i = u_b/\mathcal{W}(\mathcal{V})$, $i = 1, 2, \ldots, n$. For the finer graphs, a starting guess is obtained from the next coarser level using the uncoarsening process described below. After MCA_GR terminates, Algorithm 3.2 along with the modification discussed after Proposition 3.4 are used to obtain a binary approximation to a solution of (3.15), and then Algorithm 3.1 is used to convert the binary solution into a vertex separator.

Uncoarsening. We use the solution for the vertex separator problem computed at any level in the multilevel hierarchy as a starting guess for the solution at the next finer level. Sophisticated cycling techniques like the W- or F-cycle [4] were not implemented. A starting guess for the

next finer graph is obtained by unmatching vertices in the coarser graph. Suppose that we are uncoarsening from level l to $l-1$ and $(\mathbf{x}^l, \mathbf{y}^l)$ denotes the solution computed at level l. If vertex i at level l is obtained by matching vertices j and k at level $l-1$, then our starting guess for $(\mathbf{x}^{l-1}, \mathbf{y}^{l-1})$ is simply $(x_j^{l-1}, y_j^{l-1}) = (x_i^l, y_i^l)$ and $(x_k^{l-1}, y_k^{l-1}) = (x_i^l, y_i^l)$.

6. Numerical results. The multilevel algorithm was programmed in C++ and compiled using g++ with optimization O3 on a Dell Precision T7610 Workstation with a Dual Intel Xeon Processor E5-2687W v2 (16 physical cores, 3.4GHz, 25.6MB cache, 192GB memory). The sorting phase in the solution of the linear programs in MCA was carried out by calling *std* :: *sort*, the $(O(n \log n))$ sorting routine implemented in the C++ standard library. Comparisons were made with the routine

METIS_ComputeVertexSeparator,

available from METIS 5.1.0 [22]. The following options were employed:

METIS_IPTYPE_NODE (coarsest problem solved with node growth scheme),
METIS_RTYPE_SEP2SIDED (Fiduccia-Mattheyses-like refinement scheme).

In a preliminary experiment, we also considered the refinement option METIS_RTYPE_SEP1SIDED, but the results obtained were not significantly different. On the average, the option METIS_RTYPE_SEP2SIDED provides slightly better results. Additionally, we considered both heavy-edge matching (METIS_CTYPE_SHEM) and random matching (METIS_CTYPE_RM).

For our experiments, we considered 59 sparse graphs with dimensions ranging from $n = 1,000$ to $n = 1,965,206$ and sparsities ranging from 1.43×10^{-6} to 1.32×10^{-2}, where sparsity is defined as the ratio $\frac{|\mathcal{E}|}{n(n-1)}$ (recall that $|\mathcal{E}|$ is equal to twice the number of edges). Twenty of these graphs correspond to the adjacency matrix for symmetric matrices from the University of Florida Sparse Matrix Collection [9]. Column 2 of Table 6.1 gives the number of vertices for each of these graphs, followed by the number of edges ($|\mathcal{E}|/2$), the sparsity, and the minimum, maximum, and average vertex degrees, respectively.

Eight large graphs with heavy-tailed degree distribution (HTDD, i.e., there is a large gap between minimum and maximum vertex degree) were selected from the Stanford SNAP database [30] (see Table 6.2).

Fifteen graphs (see Table 6.3) were taken from [38], and were designed to be especially challenging for multilevel graph partitioners. The challenge in these graphs derives from the fact that the optimal vertex separator is sparse, yet densely connected to the two shores ($\mathcal{A}$ and $\mathcal{B}$). Also, these graphs represent mixtures of different structures (similar to multi-mode networks) which makes the coarsening uneven.

The Vertex Separator Problem is of particular importance in cyber security. For example, it can be used to disconnect a largest connected component in a network to prevent a possible spread of an attack or to find non-robust structures. Therefore, we also experimented with a set of 9 peer-to-peer networks from SNAP that were collected in [35] (see Table 6.4).

The UF, HTDD, Hard, and p2p graphs have at most $139,752$ nodes. In order to assess the performance of BLP on very large scale graphs, we also considered 7 graphs from the Konect database [26] having between $317,080$ and $1,965,206$ nodes and between $925,872$ and $11,095,298$ edges (see Table 6.5).

Vertex costs c_i and weights w_i were assumed to be 1 at the finest level for all graphs. For the bounds on the shores of the separator, we took $\ell_a = \ell_b = 1$ and $u_a = u_b = \lfloor 0.6n \rfloor$, which are the default bounds used by METIS. Here $\lfloor r \rfloor$ denotes the largest integer not greater than r. In

				Degree		
Graph	$\|\mathcal{V}\|$	$\|\mathcal{E}\|/2$	Sparsity	Min	Max	Ave
bcspwr09	1723	2394	1.61E-03	1	14	2.78
bcsstk17	10974	208838	3.47E-03	0	149	38.06
c-38	8127	34781	1.05E-03	1	888	8.56
c-43	11125	56275	9.09E-04	1	2619	10.12
crystm01	4875	50232	4.23E-03	7	26	20.61
delaunay_n13	8192	24547	7.32E-04	3	12	5.99
Erdos992	6100	7515	4.04E-04	0	61	2.46
fxm3_6	5026	44500	3.52E-03	3	128	17.71
G42	2000	11779	5.89E-03	4	249	11.78
jagmesh7	1138	3156	4.88E-03	3	6	5.55
lshp3466	3466	10215	1.70E-03	3	6	5.89
minnesota	2642	3303	9.47E-04	1	5	2.5
nasa4704	4704	50026	4.52E-03	5	41	21.27
net25	9520	195840	4.32E-03	2	138	41.14
netscience	1589	2742	2.17E-03	0	34	3.45
netz4504	1961	2578	1.34E-03	2	8	2.63
sherman1	1000	1375	2.75E-03	0	6	2.75
sstmodel	3345	9702	1.73E-03	0	17	5.8
USpowerGrid	4941	6594	5.40E-04	1	19	2.67
yeast	2361	6646	2.39E-03	0	64	5.63

TABLE 6.1
UF Graphs.

				Degree		
Graph	$\|\mathcal{V}\|$	$\|\mathcal{E}\|/2$	Sparsity	Min	Max	Ave
ca-HepPh	7241	202194	7.71E-03	2	982	55.85
email-Enron	9660	224896	4.82E-03	2	2532	46.56
email-EuAll	16805	76156	5.39E-04	1	3360	9.06
oregon2_010505	5441	19505	1.32E-03	1	1888	7.17
soc-Epinions1	22908	389439	1.48E-03	1	3026	34
web-NotreDame	56429	235285	1.48E-04	1	6852	8.34
web-Stanford	122749	1409561	1.87E-04	1	35053	22.97
wiki-Vote	3809	95996	1.32E-02	1	1167	50.4

TABLE 6.2
Heavy-tailed degree distribution graphs from the SNAP database.

order to enable future comparisons with our solver, we have made this benchmark set available at http://people.cs.clemson.edu/∼isafro/data.html.

6.1. Refinement comparison. In this subsection, we compare an FM-style refinement to a refinement based upon solving (3.15) using the following two experiments:

1. Obtain an approximate solution $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ to the VSP by calling the multilevel algorithm METIS_ComputeVertexSeparator. Next, refine $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ by calling MCA_GR.
2. Obtain an approximate solution $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ to the VSP by invoking the multilevel algorithm BLP. Next, refine $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ by calling METIS_NodeRefine.

				Degree		
Graph	$\|\mathcal{V}\|$	$\|\mathcal{E}\|/2$	Sparsity	Min	Max	Ave
barth5_1Ksep_50in_5Kout	32212	101805	1.96E-04	1	22	6.32
bcsstk30_500sep_10in_1Kout	58348	2016578	1.18E-03	0	219	69.12
befref_fxm_2_4_air02	14109	98224	9.87E-04	1	1531	13.92
bump2_e18_aa01_model1_crew1	56438	300801	1.89E-04	1	604	10.66
c-30_data_data	11023	62184	1.02E-03	1	2109	11.28
c-60_data_cti_cs4	85830	241080	6.55E-05	1	2207	5.62
data_and_seymourl	9167	55866	1.33E-03	1	229	12.19
finan512_scagr7-2c_rlfddd	139752	552020	5.65E-05	1	669	7.9
mod2_pgp2_slptsk	101364	389368	7.58E-05	1	1901	7.68
model1_crew1_cr42_south31	45101	189976	1.87E-04	1	17663	8.42
msc10848_300sep_100in_1Kout	21996	1221028	5.05E-03	1	722	111.02
p0291_seymourl_iiasa	10498	53868	9.78E-04	1	229	10.26
sctap1-2b_and_seymourl	40174	140831	1.75E-04	1	1714	7.01
south31_slptsk	39668	189914	2.41E-04	1	17663	9.58
vibrobox_scagr7-2c_rlfddd	77328	435586	1.46E-04	1	669	11.27

TABLE 6.3
Hard graphs from [38].

				Degree		
Graph	$\|\mathcal{V}\|$	$\|\mathcal{E}\|/2$	Sparsity	Min	Max	Ave
p2p-Gnutella04	10879	39994	6.76E-04	0	103	7.35
p2p-Gnutella05	8846	31839	8.14E-04	1	88	7.2
p2p-Gnutella06	8717	31525	8.30E-04	1	115	7.23
p2p-Gnutella08	6301	20777	1.05E-03	1	97	6.59
p2p-Gnutella09	8114	26013	7.90E-04	1	102	6.41
p2p-Gnutella24	26518	65369	1.86E-04	1	355	4.93
p2p-Gnutella25	22687	54705	2.13E-04	1	66	4.82
p2p-Gnutella30	36682	88328	1.31E-04	1	55	4.82
p2p-Gnutella31	62586	147892	7.55E-05	1	95	4.73

TABLE 6.4
Peer-to-peer networks from [35].

				Degree		
Graph	$\|\mathcal{V}\|$	$\|\mathcal{E}\|/2$	Sparsity	Min	Max	Ave
out.as-skitter	1696415	11095298	7.71E-06	1	35455	13.08
out.com-amazon	334863	925872	1.65E-05	1	549	5.53
out.com-dblp	317080	1049866	2.09E-05	1	343	6.62
out.com-youtube	1134890	2987624	4.64E-06	1	28754	5.27
out.roadNet-CA	1965206	2766607	1.43E-06	1	12	2.82
out.roadNet-PA	1088092	1541898	2.60E-06	1	9	2.83
out.roadNet-TX	1379917	1921660	2.02E-06	1	12	2.79

TABLE 6.5
Konect graphs from [26].

Graph Type	MCA_GR			METIS_NodeRefine		
	avg	min	max	avg	min	max
UF	0.02	0.00	0.22	0.03	0.00	0.38
HTDD	0.36	0.00	2.06	0.06	0.00	0.55
Hard	0.07	0.00	0.54	0.16	0.00	1.64
p2p	0.49	0.03	1.20	0.27	0.04	0.58
Total	0.16	0.00	2.06	0.12	0.00	1.64

TABLE 6.6
Percent improvement in separator sizes using MCA_GR or METIS_NodeRefine.

METIS_NodeRefine is a refinement routine which improves upon an initial solution using a Fiduccia-Mattheyses-style refinement: Vertices in $\mathcal{S}$ are moved into either $\mathcal{A}$ or $\mathcal{B}$ and their neighbors in the opposite shore are moved into $\mathcal{S}$. Vertices having the largest gains are moved first.

The results of the two experiments are given in Table 6.6. Columns labeled MCA_GR give the average, minimum, and maximum improvement in the size of $\mathcal{S}$ from calling MCA_GR in Experiment 1, and the last three columns give the results for Experiment 2. Here, the improvement is expressed as a percentage using the formula $100(\mathcal{C}(\mathcal{S}_{\text{initial}}) - \mathcal{C}(\mathcal{S}_{\text{final}}))/\mathcal{C}(\mathcal{V})$. In Experiment 1, we observed that in every initial solution obtained by METIS_ComputeVertexSeparator, the upper bounds on both of the sets $\mathcal{A}$ and $\mathcal{B}$ were inactive, which we can show implies that the METIS solution is a local minimizer in the continuous quadratic program (3.15). Hence, the algorithm MCA was unable to improve upon the METIS solutions. However, MCA_GR gave improvements of 0.16% on average, compared to only 0.12% in Experiment 2, when METIS_NodeRefine was used. The greatest improvements achieved by MCA_GR are seen in the HTDD and p2p graphs. METIS_NodeRefine was the most effective on the UF and Hard graphs.

6.2. Multilevel solution comparison. Tables 6.7–6.11 compare the costs $\mathcal{C}(\mathcal{S})$ of the vertex separators found by the multilevel implementations BLP and METIS. The coarsening phases of both algorithms involve matching vertices, which depends on a random seed. Therefore 100 trials (with different random seeds) were run for each graph. The tables report the average, minimum, and maximum costs obtained by each algorithm. Columns labeled METIS_RM and BLP_RM give the results of using METIS or BLP with random matching, and columns labeled METIS_HE and BLP_HE indicate heavy-edge based matching. The columns labeled BLP_RMFM and BLP_HEFM correspond to a hybrid approach, in which solutions are refined by first performing Fiduccia-Mattheyses-like (FM) swaps, followed by MCA_GR. FM swaps were performed by calling METIS_NodeRefine. Due to large running times of BLP on the Konect test set, only BLP_RM and METIS_RM were compared for these graphs.

The data in Tables 6.7–6.11 is summarized in Tables 6.12–6.15. For instance, Table 6.12 gives the percentage of graphs of each type for which the average size of the separator obtained by BLP_RM was strictly better than METIS_RM (% Wins) and the average, minimum, and maximum percentage improvement in the average separator size compared to METIS, as measured by the expression $100(\mathcal{C}(\mathcal{S}_{\text{METIS}}) - \mathcal{C}(\mathcal{S}_{\text{BLP}}))/\mathcal{C}(\mathcal{V})$. Note that here, neither solution is used as an initial guess for the other algorithm, unlike the experiments of Section 6.1. Table 6.13 compares BLP_RMFM with METIS_RM, and Tables 6.14 and 6.15 compare BLP_HE and BLP_HEFM with METIS_HE.

First, we note that the average improvement was positive for all versions of BLP on all graph

types, except for BLP_HE on the UF graphs and BLP_RM on the Konect graphs. However, even when METIS's separators are smaller, the difference in size is often not substantial (less than 0.1%).

The BLP algorithms seem to be the most effective on the p2p, HTDD, and Hard graphs. Based on our observations from Section 6.1, the high performance of BLP on the p2p and HTDD graphs is probably due to the refinement algorithm, while the high performance on the Hard graphs may be attributed (at least partially) to minor differences in the coarsening schemes used by METIS and BLP. The p2p graphs performed exceptionally well, giving an improvement over the METIS solution in 100% of the trials. As expected, the hybrid algorithms performed the best on average.

Due to the large dimension of the Konect graphs, combined with their abnormal degree distributions, the coarsening scheme of BLP produced a large number of coarse levels for these graphs (over 200 in some cases), since on many levels only a small number of vertices could be matched. Since the computational bottleneck of BLP is the refinement phase, we decided to refrain from refining a given coarse solution until the number of vertices increased by a factor of 2 during the uncoarsening process, in order to improve the running time of BLP. This is one possible explanation for the relatively poor performance of BLP for these problems.

In some applications, such as cybersecurity, the size of a vertex separator is of primary importance, while in other applications, such as sparse matrix reordering, small separators must be found quickly. Table 6.16 compares the average, geometric mean, minimum, and maximum CPU time (in seconds) for BLP_RM and METIS_RM on each of the five test sets. For the first four test sets in this table, the average CPU time for BLP was on average about 260 times slower than METIS. However, we note that the average BLP solution was strictly better than the best METIS solution (over 100 trials) in 25 out the 52 instances in these test sets, and in fact for all 9 p2p instances. For the Konect graphs, the CPU time gap between BLP and METIS was approximately 28 times. We suspect that this relative improvement in CPU time is due to the reduced number of refinement phases used for these problems. Finally, we stress that BLP has not been optimized for speed. Figure 6.1 gives a log-log plot of $n = |\mathcal{V}|$ versus CPU time for all 59 instances. The best fit line through the data in the log-log plot has a slope of approximately 1.65, which indicates that the CPU time of BLP is between a linear and a quadratic function of the number of vertices.

In order to determine the computational bottlenecks of BLP, we examined a flat profile of the code, using the Linux utility GNU gprof. We randomly selected one problem from each of the first four test sets and found that between 61% and 87% of the CPU time was consumed by the routine which implements the greedy algorithm for solving the linear programs in MCA. The remainder of the CPU time was shared by objective value computations, matrix vector product computations (between $\mathbf{A}$ and $\mathbf{x}$ or $\mathbf{y}$), and Karush-Kuhn-Tucker multiplier computations, which were used to determine the perturbations in $\mathbf{c}$ or γ required to escape a local optimum.

Thus, for applications in which speed is more important than solution quality, the following modifications may be investigated:

1. Instead of resolving the linear program in MCA from scratch, we could exploit the structure of the previously computed solution to update it.
2. In each iteration of the Mountain Climbing Algorithm, we need the products $\mathbf{Ax}_k$ and $\mathbf{Ay}_k$ between the matrix and a vector. We could save the previous products $\mathbf{Ax}_{k-1}$ and $\mathbf{Ay}_{k-1}$ and only recompute the parts of the products that change.

7. Conclusion. We have developed a new algorithm (BLP) for solving large-scale instances of the Vertex Separator Problem (1.1). The algorithm incorporates a multilevel framework; that is, the original graph is coarsened several times; the problem is solved for the coarsest graph; and the solution to the coarse graph is gradually uncoarsened and refined to obtain a solution to the

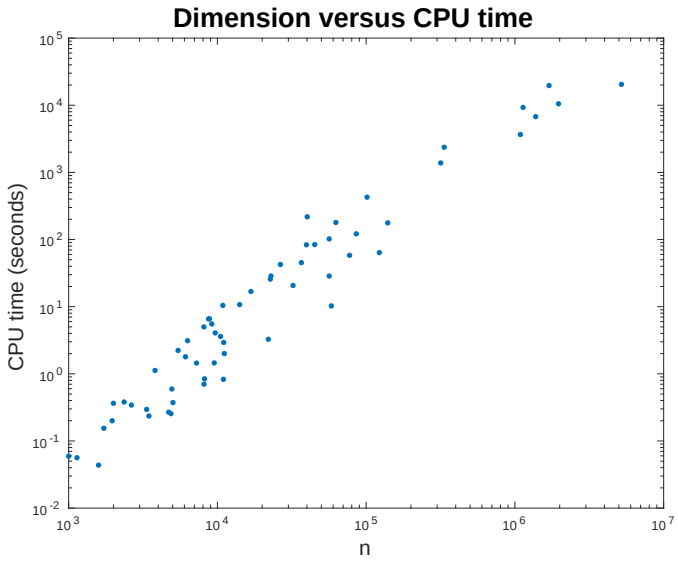

FIG. 6.1. *Number of vertices n versus CPU time for BLP_RM.*

original graph. A key feature of the algorithm is the use of the continuous bilinear program (3.15) in both the solution and refinement phases. In the case where vertex weights are all equal to one (or a constant), (3.15) is an exact formulation of the VSP in the sense that there exists a binary solution satisfying (3.4), from which an optimal solution to the VSP can be recovered using (3.9). When vertex weights are not all equal, we showed that (3.15) still approximates the VSP in the sense that there exists a mostly binary solution.

During the solution and refinement phases of BLP, the bilinear program is solved approximately by applying the algorithm MCA_GR, a mountain climbing algorithm which incorporates perturbation techniques to escape from stationary points and explore a new part of the search space. One technique, referred to as **c**-perturbations, uses the first-order optimality conditions to derive a tiny perturbation that will force an iterate to a new location with a possibly improved separator. The second technique, referred to as γ-perturbations, improves the separator by relaxing the requirement that there are no edges between the sets in the partition. We determined the smallest relaxation that will generate a new partition. To our knowledge, this is the first multilevel algorithm to make use of a continuous optimization based refinement method for the family of graph partitioning problems. The numerical results of Section 6 indicate that BLP is capable of locating vertex separators of high quality (comparing against METIS), and is particularly effective on p2p graphs, HTDD graphs (graphs with heavy-tailed distributions), and graphs having relatively sparse separators.

REFERENCES

[1] S. Acer, E. Kayaaslan, and C. Aykanat, *A recursive bipartitioning algorithm for permuting sparse square matrices into block diagonal form with overlap*, SIAM Journal on Scientific Computing, 35 (2013), pp. C99–C121.

[2] C. C. Ashcraft and J. W. H. Liu, *A partition improvement algorithm for generalized nested dissection*, Tech. Rep. BCSTECH-94-020, Boeing Computer Services, Seattle, WA, 1994.

[3] U. Benlic and J. Hao, *Breakout local search for the vertex separator problem*, in Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, 2013.

[4] A. Brandt and D. Ron, *Chapter 1 : Multigrid solvers and multilevel optimization strategies*, in Multilevel Optimization and VLSICAD, J. Cong and J. R. Shinnerl, eds., Kluwer, 2003.

[5] T. Bui and C. Jones, *Finding good approximate vertex and edge partitions is NP-hard*, Information Processing Letters, 42 (1992), pp. 153–159.

[6] A. Buluç, H. Meyerhenke, I. Safro, P. Sanders, and C. Schulz, *Recent advances in graph partitioning*, Accepted in Algorithm Engineering: Selected Results and Surveys. LNCS 9220, Springer-Verlag, Arxiv, abs/1311.3144 (2014).

[7] M. Burmester, Y. Desmedt, and Y. Wang, *Using approximation hardness to achieve dependable computation*, in Randomization and Approximation Techniques in Computer Science, M. Luby, J. Rolim, and M. Serna, eds., vol. 1518 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1998, pp. 172–186.

[8] J. Chen and I. Safro, *Algebraic distance on graphs*, SIAM J. Sci. Comput., 33 (2011), pp. 3468–3490.

[9] T. A. Davis, *University of Florida sparse matrix collection*, 1994. NA Digest, vol 92, no. 42.

[10] ———, *Direct Methods for Sparse Linear Systems*, SIAM, Philadelphia, PA, 2006.

[11] C. Evrendilek, *Vertex separators for partitioning a graph*, Sensors, 8 (2008), pp. 635–657.

[12] U. Feige, M. Hajiaghayi, and J. Lee, *Improved approximation algorithms for vertex separators*, SIAM J. Comput., 38 (2008), pp. 629–657.

[13] C. M. Fiduccia and R. M. Mattheyses, *A linear-time heuristic for improving network partitions*, in Proc. 19th Design Automation Conf., Las Vegas, NV, 1982, pp. 175–181.

[14] J. Fukuyama, *NP-completeness of the planar separator problems*, Journal of Graph Algorithms and Applications, 10, No. 2 (2006), pp. 317–328.

[15] A. George and J. W. H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[16] J. R. Gilbert and E. Zmijewski, *A parallel graph partitioning algorithm for a message-passing multiprocessor*, Intl. J. Parallel Programming, 16 (1987), pp. 498–513.

[17] W. W. Hager and J. T. Hungerford, *Optimality conditions for maximizing a function over a polyhedron*, Math. Program., 145 (2014), pp. 179–198.

[18] ———, *Continuous quadratic programming formulations of optimization problems on graphs*, European J. Oper. Res., 240 (2015), pp. 328–337.

[19] B. Hendrickson and R. Leland, *A multilevel algorithm for partitioning graphs*, in Proc. Supercomputing '95, IEEE, Nov. 1995.

[20] B. Hendrickson and E. Rothberg, *Effective sparse matrix ordering: just around the bend*, in Proc. of 8th SIAM Conference on Parallel Processing for Scientific Computing, 1997.

[21] G. Karypis and V. Kumar, *METIS: unstructured graph partitioning and sparse matrix ordering system*, tech. rep., Dept. of Computer Science, Univ. of Minnesota, 1995.

[22] ———, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392.

[23] E. Kayaaslan, A. Pinar, U. Catalyürek, and C. Aykanat, *Partitioning hypergraphs in scientific computing applications through vertex separators*, SIAM J. Sci. Comput., 34(2) (2012), pp. A970–A992.

[24] B. W. Kernighan and S. Lin, *An efficient heuristic procedure for partitioning graphs*, Bell System Tech. J., 49 (1970), pp. 291–307.

[25] D. Kim, G. R. Frye, S.-S. Kwon, H. J. Chang, and A. O. Tokuta, *On combinatoric approach to circumvent internet censorship using decoy routers*, in Military Communications Conference, 2013. MILCOM 2013., IEEE, 2013, pp. 1–6.

[26] *The Koblenz Network collection*. Online reference at `\protect\vrulewidth0pt\protect\href{http://konect.uni-koblenz.de/}{http://konect.uni-koblenz.de/}`.

[27] H. Konno, *A cutting plane algorithm for solving bilinear programs*, Mathematical Programming, 11 (1976), pp. 14–27.

[28] C. Leiserson, *Area-efficient graph layout (for VLSI)*, in Proc. 21st Annual Symposium on the Foundations of

Computer Science, IEEE, 1980, pp. 270–281.
[29] C. Leiserson and J. Lewis, *Orderings for parallel sparse symmetric factorization*, in Third SIAM Conference on Parallel Processing for Scientific Computing, SIAM Publications, 1987, pp. 27–31.
[30] J. Leskovec and A. Krevl, *SNAP Datasets: Stanford large network dataset collection*. `http://snap.stanford.edu/data`, June 2014.
[31] C. Litsas, A. Pagourtzis, G. Panagiotakos, and D. Sakavalas, *On the resilience and uniqueness of CPA for secure broadcast*, IACR Cryptology ePrint Archive, (2013).
[32] G. Miller, S. H. Teng, W. Thurston, and S. Vavasis, *Geometric separators for finite element meshes*, SIAM J. Sci. Comput., 19 (1998), pp. 364–386.
[33] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, 2nd ed., 2006.
[34] A. Pothen, H. D. Simon, and K. Liou, *Partitioning sparse matrices with eigenvectors of graphs*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 430–452.
[35] M. Ripeanu, I. Foster, and A. Iamnitchi, *Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design*, IEEE Internet Computing Journal, 6 (2002), pp. 50–57.
[36] D. Ron, I. Safro, and A. Brandt, *Relaxation-based coarsening and multiscale graph organization*, Multiscale Modeling & Simulation, 9 (2011), pp. 407–423.
[37] I. Safro, D. Ron, and A. Brandt, *Graph minimum linear arrangement by multilevel weighted edge contractions*, J. Algorithms, 60 (2006), pp. 24–41.
[38] I. Safro, P. Sanders, and C. Schulz, *Advanced coarsening schemes in graph partitioning*, in Symposium on Experimental Algorithms, LNCS, vol. 7276, 2012, pp. 369–380.
[39] J. Ullman, *Computational Aspects of VLSI*, Computer Science Press, Rockville, Md., 1984.

Graph	Best	METIS_HE			BLP_HE			BLP_HEFM		
		avg	min	max	avg	min	max	avg	min	max
bcspwr09	6	7.52	6	13	12.20	6	25	9.92	6	22
bcsstk17	126	143.88	132	186	158.22	126	234	146.24	126	216
c-38	12	14.20	12	22	41.90	14	103	24.16	12	54
c-43	83	141.67	103	155	135.76	115	166	108.17	83	138
crystm01	65	67.31	65	90	77.64	65	85	73.93	65	85
delaunay_n13	69	74.02	69	83	82.70	72	126	78.61	69	92
Erdos992	58	108.07	95	125	72.57	64	82	69.10	58	84
fxm3_6	42	53.48	42	88	66.73	42	90	52.96	42	87
G42	412	440.97	424	462	452.28	438	469	441.38	412	466
jagmesh7	14	14.03	14	15	19.90	14	42	21.67	14	42
lshp3466	51	55.41	51	61	58.96	51	105	52.33	51	73
minnesota	14	16.80	14	23	20.75	14	40	19.30	14	35
nasa4704	163	176.61	163	206	196.23	168	274	180.25	165	262
net25	510	597.34	510	915	557.27	510	993	516.13	510	578
netscience	0	0.09	0	3	0.00	0	0	0.00	0	0
netz4504	16	18.01	17	20	21.42	16	41	19.93	16	33
sherman1	18	29.98	28	39	21.23	18	28	19.88	18	26
sstmodel	20	24.28	22	35	27.34	21	37	24.77	20	33
USpowerGrid	8	8.98	8	14	20.09	8	44	12.19	8	22
yeast	137	192.62	182	213	165.16	156	178	147.23	137	155
Graph	Best	METIS_RM			BLP_RM			BLP_RMFM		
bcspwr09	6	7.47	6	11	11.88	7	28	9.82	6	18
bcsstk17	126	147.29	138	168	153.45	126	356	156.01	126	346
c-38	12	24.82	12	72	51.37	14	97	26.32	12	57
c-43	83	140.86	117	156	133.44	108	164	106.46	94	118
crystm01	65	66.50	65	90	79.60	65	85	77.57	65	80
delaunay_n13	69	75.49	69	90	86.98	71	125	81.56	69	127
Erdos992	58	121.23	109	141	73.08	64	86	69.29	59	82
fxm3_6	42	60.82	42	90	73.55	57	101	67.95	42	99
G42	412	441.48	427	458	451.01	440	464	443.23	426	463
jagmesh7	14	14.14	14	21	21.23	14	39	19.87	14	49
lshp3466	51	55.45	52	61	63.36	51	98	55.14	51	86
minnesota	14	17.34	14	23	21.03	15	40	18.70	14	30
nasa4704	163	175.45	168	188	174.63	168	208	170.94	166	181
net25	510	676.32	641	714	546.63	510	990	528.05	510	621
netscience	0	0.16	0	5	0.00	0	0	0.00	0	0
netz4504	16	18.29	16	23	21.15	16	36	20.00	16	33
sherman1	18	30.70	29	50	21.52	18	30	20.56	18	27
sstmodel	20	24.85	22	40	25.99	20	37	24.31	20	34
USpowerGrid	8	9.13	8	16	18.84	8	35	12.31	8	23
yeast	137	212.37	177	236	165.14	153	178	147.33	137	158

TABLE 6.7
Vertex Separator Costs $\mathcal{C}(\mathcal{S})$ for UF graphs.

Graph	Best	METIS_HE			BLP_HE			BLP_HEFM		
		avg	min	max	avg	min	max	avg	min	max
ca-HepPh	583	754.23	668	839	657.29	583	706	678.05	591	736
email-Enron	426	687.12	604	804	484.94	426	578	481.25	436	583
email-EuAll	5	8.99	5	57	9.41	6	18	7.33	6	12
oregon2_010505	37	58.57	48	70	53.30	41	64	43.59	37	59
soc-Epinions1	2382	2975.27	2818	3078	2423.42	2382	2465	2529.81	2457	2576
web-NotreDame	132	399.97	270	518	431.95	132	543	415.45	134	505
web-Stanford	29	133.52	29	575	415.13	95	815	261.23	72	584
wiki-Vote	680	704.86	694	731	716.23	698	764	706.03	680	735
Graph	Best	METIS_RM			BLP_RM			BLP_RMFM		
ca-HepPh	583	767.56	683	851	674.28	621	720	683.55	625	745
email-Enron	426	709.29	650	839	496.20	451	547	487.84	440	574
email-EuAll	5	76.04	5	348	11.99	7	35	10.25	6	28
oregon2_010505	37	79.00	66	113	53.60	46	68	43.67	38	51
soc-Epinions1	2382	3072.42	2915	3224	2431.98	2399	2475	2529.72	2467	2572
web-NotreDame	132	437.77	274	611	462.24	190	567	419.79	136	489
web-Stanford	29	143.73	29	484	384.79	134	495	283.44	63	382
wiki-Vote	680	708.97	694	737	716.06	696	768	709.51	680	737

TABLE 6.8
Vertex Separator Costs $\mathcal{C}(\mathcal{S})$ for HTDD graphs.

Graph	Best	METIS_HE			BLP_HE			BLP_HEFM		
		avg	min	max	avg	min	max	avg	min	max
vsp_barth5_1Ksep_50in_5Kout	987	1329.76	1131	1451	1546.57	1309	1767	1353.79	987	1640
vsp_bcsstk30_500sep_10in_1Kout	528	752.65	552	1228	870.56	570	1510	830.00	550	1644
vsp_befref_fxm_2_4_air02	270	1072.63	989	1142	284.98	270	302	284.87	275	300
vsp_bump2_e18_aa01_model1_crew1	3849	4306.55	4264	4343	3978.57	3903	4404	4024.77	3849	4411
vsp_c-30_data_data	453	510.31	453	594	555.38	475	656	513.32	460	605
vsp_c-60_data_cti_cs4	2222	2600.78	2525	2684	2930.37	2354	3590	2826.26	2222	3589
vsp_data_and_seymourl	1030	1253.12	1148	1341	1150.97	1097	1271	1084.40	1045	1258
vsp_finan512_scagr7-2c_rlfddd	4605	7438.78	7216	7697	4871.55	4776	5029	4692.32	4608	4798
vsp_mod2_pgp2_slptsk	5739	5859.63	5809	5905	7434.46	5767	9711	6391.40	5739	9091
vsp_model1_crew1_cr42_south31	1838	2216.08	2086	2631	2507.86	2424	2576	1971.33	1838	2013
vsp_msc10848_300sep_100in_1Kout	279	648.16	279	929	723.60	343	1421	669.93	387	1209
vsp_p0291_seymourl_iiasa	511	536.25	532	542	516.66	511	528	521.60	513	528
vsp_sctap1-2b_and_seymourl	3390	4114.48	3831	4373	3925.59	3732	4390	3715.86	3390	4134
vsp_south31_slptsk	1971	2054.39	1982	2116	2328.52	2242	2567	2031.13	1971	2081
vsp_vibrobox_scagr7-2c_rlfddd	2762	3467.55	3362	3613	2856.32	2801	2992	2867.30	2762	3050
Graph	Best	METIS_RM			BLP_RM			BLP_RMFM		
vsp_barth5_1Ksep_50in_5Kout	987	1346.14	1043	1530	1549.01	1326	1818	1361.58	1123	1613
vsp_bcsstk30_500sep_10in_1Kout	528	636.70	528	844	627.95	565	854	629.66	570	850
vsp_befref_fxm_2_4_air02	270	1464.03	1328	1584	288.35	274	314	285.45	275	303
vsp_bump2_e18_aa01_model1_crew1	3849	4378.39	4280	4793	3986.52	3897	4389	4070.99	3894	4429
vsp_c-30_data_data	453	536.73	471	611	530.51	458	602	500.92	463	568
vsp_c-60_data_cti_cs4	2222	2636.81	2384	2741	2869.78	2345	3545	2666.91	2226	3365
vsp_data_and_seymourl	1030	1243.14	1091	1347	1139.91	1087	1207	1082.04	1030	1248
vsp_finan512_scagr7-2c_rlfddd	4605	7610.25	7400	7883	4975.58	4861	5170	4691.99	4605	4776
vsp_mod2_pgp2_slptsk	5739	5884.52	5838	5925	6123.36	5776	8920	6605.61	5741	9133
vsp_model1_crew1_cr42_south31	1838	2740.18	2558	2894	2508.41	2413	2581	1973.98	1926	2013
vsp_msc10848_300sep_100in_1Kout	279	523.70	279	715	553.37	279	791	541.28	279	785
vsp_p0291_seymourl_iiasa	511	535.98	531	545	516.61	511	533	521.36	513	529
vsp_sctap1-2b_and_seymourl	3390	4269.77	3884	4557	3921.85	3766	4030	3764.07	3418	4159
vsp_south31_slptsk	1971	2416.25	2350	2498	2298.76	2205	2563	2033.78	1977	2082
vsp_vibrobox_scagr7-2c_rlfddd	2762	4182.86	3995	5240	2878.12	2804	2995	2877.13	2789	2998

TABLE 6.9
Vertex Separator Costs $\mathcal{C}(\mathcal{S})$ for hard graphs.

Graph	Best	METIS_HE			BLP_HE			BLP_HEFM		
		avg	min	max	avg	min	max	avg	min	max
p2p-Gnutella04	1656	2055.11	1986	2103	1710.59	1675	1755	1697.08	1662	1738
p2p-Gnutella05	1306	1666.38	1629	1724	1369.70	1340	1404	1348.47	1306	1385
p2p-Gnutella06	1253	1605.98	1568	1653	1305.96	1265	1343	1291.29	1260	1327
p2p-Gnutella08	771	1009.10	976	1043	825.25	794	855	795.99	772	821
p2p-Gnutella09	975	1287.01	1253	1327	1044.83	1021	1081	1003.17	975	1038
p2p-Gnutella24	2463	3284.91	3203	3380	2728.58	2671	2781	2511.26	2467	2553
p2p-Gnutella25	2043	2761.52	2691	2836	2279.78	2227	2345	2089.72	2043	2143
p2p-Gnutella30	3016	4267.82	4094	4398	3320.50	3245	3422	3097.88	3035	3176
p2p-Gnutella31	4905	5985.32	5888	6184	5581.26	5460	5750	5002.65	4905	5081
Graph	Best	METIS_RM			BLP_RM			BLP_RMFM		
p2p-Gnutella04	1656	2140.50	2089	2200	1708.87	1673	1749	1696.39	1656	1742
p2p-Gnutella05	1306	1720.54	1687	1750	1369.40	1341	1407	1350.60	1324	1389
p2p-Gnutella06	1253	1689.01	1641	1727	1306.65	1275	1339	1290.61	1253	1318
p2p-Gnutella08	771	1042.30	1003	1075	825.46	795	857	794.54	771	822
p2p-Gnutella09	975	1328.33	1293	1367	1044.11	1010	1076	1003.04	981	1038
p2p-Gnutella24	2463	3617.52	3538	3685	2727.85	2670	2806	2514.59	2463	2563
p2p-Gnutella25	2043	3008.89	2931	3095	2276.96	2233	2335	2091.69	2046	2144
p2p-Gnutella30	3016	4692.64	4580	4798	3321.06	3239	3433	3093.14	3016	3184
p2p-Gnutella31	4905	6904.14	6619	7468	5571.38	5432	5712	5014.85	4929	5071

TABLE 6.10
Vertex Separator Costs $\mathcal{C}(\mathcal{S})$ for peer-to-peer networks.

Graph	Best	METIS_RM			BLP_RM		
		avg	min	max	avg	min	max
out.as-skitter	15006	20654.00	15006	23149	30834.29	26728	32380
out.com-amazon	4237	4470.61	4237	4644	5622.31	5386	5893
out.com-dblp	10600	10859.04	10600	11136	12762.98	11855	13848
out.com-youtube	16729	28355.39	26911	30371	17480.22	16729	18165
out.roadNet-CA	99	124.13	99	168	336.60	127	1222
out.roadNet-PA	109	128.90	109	155	343.11	131	865
out.roadNet-TX	59	77.41	59	130	248.71	74	533

TABLE 6.11
Vertex Separator Costs $\mathcal{C}(\mathcal{S})$ for Konect networks.

		% Improvement		
Graph Type	% BLP Wins	avg	min	max
UF	35.00	0.10	-0.62	2.00
p2p	100.00	3.52	2.13	4.39
HTDD	62.50	0.84	-0.20	2.80
Hard	73.33	0.96	-0.63	8.33
Konect	14.29	-0.09	-0.60	0.96
Total	55.93	0.92	-0.63	8.33

TABLE 6.12
Comparison of separator costs $\mathcal{C}(\mathcal{S})$ obtained by BLP_RM and METIS_RM.

		% Improvement		
Graph Type	% BLP Wins	avg	min	max
UF	45.00	0.26	-0.50	2.75
p2p	100.00	4.04	3.02	4.57
HTDD	74.22	1.22	-0.11	3.36
Hard	77.81	1.33	-0.08	8.35
Total	68.48	1.37	-0.50	8.35

TABLE 6.13
Comparison of separator costs $\mathcal{C}(\mathcal{S})$ obtained by BLP_RMFM and METIS_RM.

		% Improvement		
Graph Type	% BLP Wins	avg	min	max
UF	30.00	-0.02	-0.57	1.16
p2p	100.00	2.59	0.65	3.44
HTDD	50.00	0.67	-0.30	2.41
Hard	46.67	0.38	-1.55	5.58
Total	50.00	0.65	-1.55	5.58

TABLE 6.14
Comparison in separator costs $\mathcal{C}(\mathcal{S})$ obtained by BLP_HE and METIS_HE.

		% Improvement		
Graph Type	% BLP Wins	avg	min	max
UF	40.00	0.17	-0.67	1.92
p2p	100.00	3.11	1.57	3.61
HTDD	62.50	0.66	-0.10	2.13
Hard	60.00	0.75	-0.52	5.58
Total	59.62	0.92	-0.67	5.58

TABLE 6.15
Comparison of separator costs $\mathcal{C}(\mathcal{S})$ obtained by BLP_HEFM and METIS_HE.

	BLP_RM				METIS_RM			
Graph Type	avg	geomean	min	max	avg	geomean	min	max
UF	0.56	0.34	0.03	3.08	0.01	0.00	0.00	0.12
p2p	36.08	14.99	1.48	276.09	0.16	0.13	0.05	0.49
HTDD	18.34	7.64	0.64	104.82	0.13	0.08	0.01	0.55
Hard	88.59	32.23	1.58	719.48	0.28	0.21	0.05	0.84
Konect	9258.19	5989.74	689.58	27888.17	334.55	4.97	0.63	2702.44
Total	1264.62	10.44	0.03	27888.17	44.72	0.00	0.00	2702.44

TABLE 6.16
CPU times (in seconds) for each algorithm on different graph types.