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

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

denote the total cost and weight of the vertices in \( Z \), respectively.

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

\[
\min_{A, S, B \subseteq V} C(S)
\]

subject to \( S = V \setminus (A \cup B) \), \( A \cap B = \emptyset \), \((A \times B) \cap E = \emptyset \), \( \ell_a \leq W(A) \leq u_a \), and \( \ell_b \leq W(B) \leq u_b \),

\(\ell_a, u_a, \ell_b, u_b\) are the lower and upper bounds on the weights of the sets \( A \) and \( B \).
where $\ell_a$, $\ell_b$, $u_a$, and $u_b$ are given nonnegative real numbers less than or equal to $W(V)$. The constraints $S = V \setminus (A \cup B)$ and $A \cap B = \emptyset$ ensure that $V$ is partitioned into disjoint sets $A$, $B$, and $S$, while the constraint $(A \times B) \cap E = \emptyset$ ensures that there are no edges between the sets $A$ and $B$. 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 $(i, j) \notin E$; 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 $V$ 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, [11] 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 0 or 1 respectively, where the dimension will be clear from the context. If \( x \in \mathbb{R}^n \) and \( f : \mathbb{R}^n \to \mathbb{R} \), then \( \nabla f(x) \) denotes the gradient of \( f \) at \( x \), a row vector, and \( \nabla^2 f(x) \) is the Hessian. If \( f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R} \), then \( \nabla_x f(x, y) \) is the row vector corresponding to the first \( n \) entries of \( \nabla f(x, y) \), while \( \nabla_y f(x, y) \) is the row vector corresponding to the last \( n \) entries. If \( A \) is a matrix, then \( A_i \) denotes the \( i \)-th row of \( A \). If \( x \in \mathbb{R}^n \), then \( x \geq 0 \) means \( x_i \geq 0 \) for all \( i \), and \( x^T \) denotes the transpose, a row vector. Let \( I \in \mathbb{R}^{n \times n} \) denote the \( n \times n \) identity matrix, let \( e_i \) denote the \( i \)-th column of \( I \), and let \( |A| \) denote the number of elements in the set \( A \).

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

\[
\max_{A, B \subseteq V} C(A \cup B) \\
\text{subject to } A \cap B = \emptyset, \quad (A \times B) \cap E = \emptyset, \\
\ell_a \leq W(A) \leq u_a, \quad \text{and} \quad \ell_b \leq W(B) \leq u_b.
\]

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

\[
x_i = \begin{cases} 1, & \text{if } i \in A \\ 0, & \text{if } i \notin A \end{cases} \quad \text{and} \quad y_i = \begin{cases} 1, & \text{if } i \in B \\ 0, & \text{if } i \notin B \end{cases}.
\]

Let \( H := (A + I) \), where \( A \) is the \( n \times n \) adjacency matrix for \( G \) defined by \( a_{ij} = 1 \) if \((i, j) \in E\) and \( a_{ij} = 0 \) otherwise, and \( I \) is the \( n \times n \) identity matrix. Note that since \( G \) is undirected, \( H \) is
symmetric. Then we have
\[
x^T H y = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i a_{ij} y_j + \sum_{i=1}^{n} x_i y_i = \sum_{i \in A} \sum_{j \in B} a_{ij} + \sum_{i \in A \cap B} 1
\]
(3.3)
So, the constraints \(A \cap B = \emptyset\) and \((A \times B) \cap E = \emptyset\) in (3.1) hold if and only if \(x^T H y = 0\).

Hence, a binary formulation of (3.1) is
\[
\max_{x, y \in \{0, 1\}^n} c^T (x + y)
\]
subject to \(x^T H y = 0\), \(\ell_a \leq w^T x \leq u_a\), and \(\ell_b \leq w^T y \leq u_b\).

where \(c\) and \(w\) 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:
\[
\max_{x, y \in \{0, 1\}^n} f(x, y) := c^T (x + y) - \gamma x^T H y
\]
subject to \(\ell_a \leq w^T x \leq u_a\) and \(\ell_b \leq w^T y \leq u_b\).

Here, \(\gamma \in \mathbb{R}\). Notice that \(\gamma x^T H y\) acts as a penalty term in (3.6) when \(\gamma \geq 0\), since \(x^T H y \geq 0\) for every \(x, y \in \{0, 1\}^n\). Moreover, (3.6) gives a relaxation of (3.5), since the constraint (3.3) 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 \(w \geq 1\) and \(\gamma > 0\) with \(\gamma \geq \max\{c_i : i \in V\}\), then for any feasible point \((x, y)\) in (3.6) satisfying
\[
f(x, y) \geq \gamma (\ell_a + \ell_b),
\]
there is a feasible point \((\bar{x}, \bar{y})\) in (3.6) such that
\[
f(\bar{x}, \bar{y}) \geq f(x, y) \quad \text{and} \quad x^T H y = 0.
\]
Hence, if the optimal objective value in (3.6) is at least \(\gamma (\ell_a + \ell_b)\), then there exists an optimal solution \((x^*, y^*)\) to (3.6) such that an optimal solution to (3.1) is given by
\[
A = \{i : x_i^* = 1\}, \; B = \{i : y_i^* = 1\}, \; \text{and} \; S = \{i : x_i^* = y_i^* = 0\}.
\]

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

\[
x^THy > 0.
\]

Then,

\[
\gamma(\ell_a + \ell_b) \leq f(x, y) = c^T(x + y) - \gamma x^THy
\]

(3.12)

\[
< c^T(x + y)
\]

(3.13)

\[
\leq \gamma 1^T(x + y).
\]

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 \(1^T x > \ell_a\) or \(1^T y > \ell_b\).

Assume without loss of generality that \(1^T x > \ell_a\). Since \(x\) is binary and \(\ell_a\) is an integer, we have

\[
1^T x \geq \ell_a + 1.
\]

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

\[
w^T \bar{x} \geq 1^T x = 1^T x - 1 \geq \ell_a.
\]

Here the first inequality is due to the assumption that \(w \geq 1\). Furthermore,

\[
f(\bar{x}, y) = f(x, y) - c_i + \gamma H_i y \geq f(x, y) - c_i + \gamma \geq f(x, y),
\]

since \(H_{i,j} \geq 1\), \(\gamma \geq 0\), and \(\gamma \geq c_i\). We can continue to set components of \(x\) and \(y\) to 0 until reaching a binary feasible point \((\bar{x}, y)\) for which \(\bar{x}^THy = 0\) and \(f(\bar{x}, y) \geq f(x, y)\). 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 \((x^*, y^*)\) satisfying (3.4); hence, \((x^*, y^*)\) is feasible in (3.5). Since (3.6) is a relaxation of (3.5), \((x^*, y^*)\) is optimal in (3.5). Hence, the partition \((A, S, B)\) defined by (3.6) is optimal in (3.1). This completes the proof. \(\blacksquare\)

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 \((\bar{x}, y)\) 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 \(x\) and \(y\) are incidence vectors for a pair of feasible sets \(A\) and \(B\) in (3.1), since in this case

\[
f(x, y) = c^T(x + y) = w^T x + w^T y \geq \ell_a + \ell_b = \gamma(\ell_a + \ell_b).
\]

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

\[
\max_{x, y \in \mathbb{R}^n} f(x, y) := c^T(x + y) - \gamma x^THy
\]

subject to \(0 \leq x \leq 1\), \(0 \leq y \leq 1\), \(\ell_a \leq w^T x \leq u_a\), and \(\ell_b \leq w^T y \leq u_b\).
**Input:** A binary feasible point \((x, y)\) for (3.6) satisfying (3.7) while \((x^T H y > 0)\)

- \(\text{if } (1^T x > \ell_a)\)
  - Choose \(i\) such that \(x_i = 1\) and \(H_i y \geq 1\).
  - Set \(x_i = 0\).
- \(\text{else if } (1^T y > \ell_b)\)
  - Choose \(i\) such that \(y_i = 1\) and \(H_i x \geq 1\).
  - 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 \(c \geq 0\) and \(w = 1\). In particular, the following theorem is proved:

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

In the proof of Theorem 3.2, a step-by-step procedure is given for moving from any feasible point \((x, y)\) in (3.15) to a binary point \((\bar{x}, \bar{y})\) satisfying \(f(\bar{x}, \bar{y}) \geq f(x, y)\). 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 \(|V|\) 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 \(w = 1\) does not hold at the coarser levels. Indeed, when \(c \geq 0\) and \(w \neq 1\), (3.15) may not have a binary optimal solution, as we will show. However, in the general case where \(w > 0\) and \(c \in \mathbb{R}^n\), the following weaker result is obtained:

**Definition 3.3.** A point \((x, y) \in \mathbb{R}^{2n}\) is called mostly binary if \(x\) and \(y\) 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 \((x, y)\) feasible in (3.15), there exists a piecewise linear path to a feasible point \((\bar{x}, \bar{y}) \in \mathbb{R}^{2n}\) which is mostly binary and satisfies \(f(\bar{x}, \bar{y}) \geq f(x, y)\).

Let \((x, y)\) be any feasible point of (3.15). If \(x\) and \(y\) 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 \(w > 0\), we can define

\[x(t) := x + t \left( \frac{1}{w_k} e_k - \frac{1}{w_l} e_l \right)\]

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

\[f(x(t), y) = f(x, y) + td, \quad \text{where } d = \nabla_x f(x, y) \left( \frac{1}{w_k} e_k - \frac{1}{w_l} e_l \right).\]

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 \(x\) 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 \(w^T x(t) = w^T x\). We may continue moving components to bounds in this manner until \(x\) has at most one non-binary component. The same procedure may be applied to \(y\). In this way, we will arrive at a feasible point \((\bar{x}, \bar{y})\) such that \(x\) and \(y\) each have at most one non-binary component and \(f(\bar{x}, \bar{y}) \geq f(x, y)\). This proves (P), which completes the proof. \(\square\)

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 \(x\), and similarly for \(y\), there must exist exactly \(n\) linearly independent constraints which are active at \(x\); in particular, at least \(n - 1\) components of \(x\) must lie at a bound, and similarly for \(y\). Therefore, \((x, y)\) is mostly binary. In the case where \(w \neq 1\), there may exist extreme points of the feasible set which are not binary; for example, consider \(n = 3\), \(\ell_a = \ell_b = 1\), \(u_a = u_b = 2\), \(w = (1, 1, 2)\), \(x = (1, 0, 0.5)\), and \(y = (0, 1, 0.5)\).

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

\[f(x(t), y) = f(x, y) + td, \quad d = \nabla_x f(x, y)e_i.\]

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

\[\ell_a + w_i \leq w^T x \leq u_a - w_i,\] (3.16)
Input: A feasible point \((x, y)\) for the continuous bilinear program (3.15).

while ( \(x\) has at least 2 nonbinary components )

Choose \(i, j \in V\) such that \(x_i, x_j \in (0, 1)\).

Update \(x \leftarrow x + t\left(\frac{1}{w_i}e_i - \frac{1}{w_j}e_j\right)\), choosing \(t\) to ensure that:

(a) \(f(x, y)\) does not decrease,
(b) either \(x_i \in \{0, 1\}\) or \(x_j \in \{0, 1\}\),
(c) \(x\) feasible in (3.15).

end while

while ( \(y\) has at least 2 nonbinary components )

Choose \(i, j \in V\) such that \(y_i, y_j \in (0, 1)\).

Update \(y \leftarrow y + t\left(\frac{1}{w_i}e_i - \frac{1}{w_j}e_j\right)\), choosing \(t\) to ensure that:

(a) \(f(x, y)\) does not decrease,
(b) either \(y_i \in \{0, 1\}\) or \(y_j \in \{0, 1\}\),
(c) \(y\) 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 \(\ell_a \leq w^T x(t) \leq u_a\). In applications, either the inequality (3.10) holds, or an analogous inequality \(\ell_b + w_j \leq w^T y \leq u_a - w_j\) holds for \(y\), where \(y_j\) is a nonbinary component of \(y\). The reason that one of these inequalities holds is that we typically have \(u_a = u_b > \frac{W(V)}{2}\), which implies that the upper bounds \(w^T x \leq u_a\) and \(w^T y \leq u_b\) cannot be simultaneously active. On the other hand, the lower bounds \(w^T x \geq \ell_a\) and \(w^T y \geq \ell_b\) are often trivially satisfied when \(\ell_a\) and \(\ell_b\) 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 \(w = 1\), 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 \(w \neq 1\), 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 \(w^T x\) and \(w^T y\). 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 \(w = 1\), 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 \(x\) or \(y\) fixed while optimizing over the other variable. This
A MULTILEVEL ALGORITHM FOR THE VERTEX SEPARATOR PROBLEM

Input: A feasible point \((x, y)\) for (3.15) and \(\eta > 0\).

while \((x, y)\) not stationary point for (3.15):

\[\hat{x} \leftarrow \text{argmax} \{ f(x, y) : x \in \mathcal{P}_a \} \]
\[\hat{y} \leftarrow \text{argmax} \{ f(x, y) : y \in \mathcal{P}_b \} \]

if \(f(\hat{x}, \hat{y}) > \max\{ f(\hat{x}, y), f(x, \hat{y}) \} + \eta \)

\[(x, y) \leftarrow (\hat{x}, \hat{y})\]

else if \(f(\hat{x}, y) > f(x, \hat{y})\)

\[x \leftarrow \hat{x}\]

else

\[y \leftarrow \hat{y}\]

end if

end while

return \((x, y)\)

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 \(x\) and in \(y\). In our modified version of the mountain climbing algorithm, which we call MCA (see Algorithm 4.1), we maximize over \(x\) with \(y\) held fixed to obtain \(\hat{x}\), we maximize over \(y\) with \(x\) held fixed to obtain \(\hat{y}\), 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 \(f(\hat{x}, \hat{y})\), \(f(\hat{x}, y)\), or \(f(x, \hat{y})\). The step \((\hat{x}, \hat{y})\) is only taken if it provides an improvement of at least \(\eta\) more than either an \(\hat{x}\) or \(\hat{y}\) step, where \(\eta\) is a small constant (10\(^{-5}\) in our experiments). After an \(x\) or \(y\) step is taken, the subspace maximizer alternates between \((\hat{x}, \hat{y})\) and \((x, \hat{y})\), 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

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

The linear programs arising in MCA are solved using a greedy algorithm. In particular, \(\max \{ f(x, y) : x \in \mathcal{P}_a \}\) is solved by setting all components \(x_i\) equal to zero and then sorting the components in decreasing order of their ratio \(r_i := \frac{\partial f}{\partial x_i}(x, y)/w_i\); the components are then visited in order and are pushed up from 0 to 1 until either \(w^T z = u_a\) 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 \(c\) or in the penalty parameter \(\gamma\). 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 \((x, y)\) for
we conclude that
\[ \nabla_x f(x, y)(\tilde{x} - x) + \nabla_y f(x, y)(\tilde{y} - y) \leq 0 \]

whenever \((\tilde{x}, \tilde{y})\) 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 \((x, y)\) for \((3.15)\) is a local maximizer only if there exist multipliers \(\mu^a \in M(x), \mu^b \in M(y), \lambda^a \in L(x, \ell_a, u_a), \lambda^b \in L(y, \ell_b, u_b)\) such that

\[ \left[ \begin{array}{c} \nabla_x f(x, y) \\ \nabla_y f(x, y) \end{array} \right] + \left[ \begin{array}{c} \mu^a \\ \mu^b \end{array} \right] + \left[ \begin{array}{c} \lambda^a \w \\ \lambda^b \w \end{array} \right] = 0, \]

where

\[ M(z) = \{ \mu \in \mathbb{R}^n : \mu_i z_i \leq \min \{ \mu_i, 0 \} \text{ for all } 1 \leq i \leq n \} \text{ and } \]
\[ L(x, \ell, u) = \{ \lambda \in \mathbb{R} : \lambda w^T z \leq \min \{ \lambda u, \lambda \ell \} \}. \]

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 \(M\) and \(L\).

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

**Proposition 4.1.** If \(\gamma \in \mathbb{R}\) and \((x, y)\) 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 \(c, (x, y)\) does not satisfy the first-order optimality condition \((4.2)\) for \(f = \tilde{f}_c\):

1. For any \(i \neq j\) such that \(\mu^a_i = \mu^b_j = 0\), \(x_i < 1\), 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 \(w^T x < u_a\), then for any \(i\) such that \(\mu^a_i = 0\) and \(x_i < 1\), take \(\tilde{c}_i = c_i + \epsilon\) and \(\tilde{c}_k = c_k\) for \(k \neq i\).
3. If \(\lambda^a = 0\) and \(w^T x > \ell_a\), then for any \(j\) such that \(\mu^b_j = 0\) and \(x_j > 0\), take \(\tilde{c}_j = c_j - \epsilon\) and \(\tilde{c}_k = c_k\) for \(k \neq j\).

**Proof.** Part 1. Let \(i\) and \(j\) satisfy the stated conditions and define the vector \(d = w_i e_i - w_j e_j\). Since \(w^T d = 0\), \(x_i < 1\), and \(x_j > 0\), it follows that \(\tilde{x}(t) = x + td\) is feasible in \((3.15)\) for \(t > 0\) sufficiently small. Moreover, by \((4.2)\) and the assumption \(\mu^a_i = \mu^b_j = 0\), we have \(\nabla_x f_c(x, y) d = 0\).

Since
\[ \nabla_x f_c(x, y) = \nabla_x f_c(x, y) + \epsilon (e^T_i - e^T_j), \]

we conclude that \(\nabla_x f_c(x, y) d = \epsilon (w_i + w_j)\), which implies that
\[ \nabla_x f_c(x, y)(\tilde{x}(t) - x) = \epsilon (w_i + w_j) > 0. \]
Input: A feasible point \((x, y)\) for (3.16).

\[
(x, y) \leftarrow \text{MCA} (x, y)
\]

loop
\[
\hat{c} \leftarrow \text{perturb} (c)
\]
\[
(\hat{x}, \hat{y}) \leftarrow \text{MCA} (x, y, \hat{c})
\]
\[
(x^*, y^*) \leftarrow \text{MCA} (\hat{x}, \hat{y}, c)
\]
if \((f(x^*, y^*) > f(x, y))\)
\[
(x, y) \leftarrow (x^*, y^*)
\]
else
break
end if
end loop
return \((x, y)\)

**Algorithm 4.2. MCA\_CP:** A modification of MCA which incorporates \(c\)-perturbations.

This shows that the first-order optimality condition (4.1) is not satisfied at \((x, y)\) for \(f = f_c\).

**Part 2.** Define \(d = e_i\). Since \(w^T x < u^a\) and \(x_i < 1\), it follows that \(\hat{x}(t) = x + td\) is feasible in (3.15) for \(t > 0\) sufficiently small. Since \(\lambda^a = \mu_i^a = 0\), (4.2) implies that \(\nabla_{x} f_c(x, y)d = 0.\) So,

\[
\nabla_{x} f_c(x, y)d = [\nabla_{x} f_c(x, y) + \epsilon e^T_i]d = 0 + \epsilon e^T_i d = \epsilon,
\]

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

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

Of course, Proposition 4.1 may be applied to either \(x\) or \(y\). Since \(\epsilon\) was arbitrary, we usually take \(\epsilon\) to be a tiny positive number \((10^{-6} \text{ in our experiments})\). By making a tiny change in the problem, the iterates of the optimization algorithm MCA applied to \(f = f_c\) 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 \(c\)-perturbations into MCA. In the figure, the notation MCA \((x, y, \hat{c})\) indicates that the MCA algorithm is applied to the point \((x, y)\) using \(\hat{c}\) in place of \(c\) as the vector of vertex costs. In our experiments, the c-perturbations were performed in the following way: For each \(i\) such that \(|\mu_i^b| < 10^{-5}\), we set \(\hat{c}_i = c_i + \epsilon\) whenever \(x_i < 0.5\) and \(\hat{c}_i = c_i - \epsilon\) otherwise; similar perturbations are made based on the values of \(\mu_i^b\) and \(y_i\).

#### 4.3. \(\gamma\)-perturbations

Next, we consider perturbations in the parameter \(\gamma\). According to our theory, we need to take \(\gamma \geq \max\{c_i : i \in V\}\) to ensure an (approximate) equivalence between the discrete (3.14) and the continuous (3.15) VSP. The penalty term \(-\gamma x^T H y\) in the objective function of (3.15) enforces the constraints \(A \cap B = \emptyset\) and \((A \times B) \cap E = \emptyset\) of (3.1). Thus, by decreasing \(\gamma\), 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 \(\gamma\) in order to ensure that the current point \((x, y)\), a local maximizer of \(f_{\gamma}\), is no longer a local maximizer of the perturbed problem \(f_{\gamma}\). 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 $(x, y)$ is feasible in (3.15), then $(x, y)$ is a local maximizer if and only if the following hold:

- (C1) $\nabla_x f(x, y) d \leq 0$ for every $d \in \mathcal{F}_a(x) \cap D$,
- (C2) $\nabla_y f(x, y) d \leq 0$ for every $d \in \mathcal{F}_b(y) \cap D$, and
- (C3) $d_1^T \nabla^2 f d_2 \leq 0$ for every $d_1, d_2 \in \mathcal{C}(x, y) \cap \mathcal{G}$,

where

$$\mathcal{F}_i(z) = \left\{ d \in \mathbb{R}^n : \begin{array}{l}
  d_j \leq 0 \text{ for all } j \text{ such that } z_j = 1 \\
  d_j \geq 0 \text{ for all } j \text{ such that } z_j = 0
\end{array}, \quad i = a, b, z \in \mathbb{R}^n,$$

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

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

The sets $\mathcal{F}_a$ and $\mathcal{F}_b$ are the cones of first-order feasible directions at $x$ and $y$. 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 $D$ 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 $D$ 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 $\mathcal{C}(x, y)$.

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

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

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

with $\alpha_1 := -\infty$ if $\mathcal{J} = \emptyset$.

2. Suppose $w^T x = u_a$. Then (C1) holds at $(x, y)$ for $f = f_\tilde{\gamma}$ if and only if $\tilde{\gamma} \geq \alpha_2$ where

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

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

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

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

- $e_i \in \mathcal{F}_a(x)$ if and only if $x_i < 1$,
- $-e_i \in \mathcal{F}_a(x)$ if and only if $x_i > 0$,
- $(w_j e_i - w_i e_j) \in \mathcal{F}_a(x)$ if and only if $x_i < 1$ and $x_j > 0$. 

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

\[
\nabla_x f_\gamma(x, y)e_i \leq 0 \quad \text{when } x_i < 1,
\]

\[
\nabla_x f_\gamma(x, y)e_i \geq 0 \quad \text{when } x_i > 0,
\]

\[
\nabla_x f_\gamma(x, y)(w_j e_i - w_i e_j) \leq 0 \quad \text{when } x_i < 1 \text{ and } x_j > 0.
\]

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 \( f_\gamma \), we know that

\[
\nabla_x f_\gamma(x, y)e_i = c_i - \gamma H_i y \geq 0 \quad \text{when } x_i > 0.
\]

Hence, since \( \tilde{\gamma} \leq \gamma \) and \( H_i y \geq 0 \),

\[
\nabla_x f_\gamma(x, y)e_i = c_i - \tilde{\gamma} H_i y \geq 0 \quad \text{when } x_i > 0.
\]

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

\[
\nabla_x f_\gamma(x, y)e_i = c_i - \gamma H_i y = 0 \quad \text{when } x_i < 1.
\]

Hence, for every \( j \) such that \( x_j < 1 \) and \( H_j y = 0 \), we have

\[
\nabla_x f_\gamma(x, y)e_i = c_j - \gamma H_j y = 0 \quad \text{when } x_j < 1.
\]

So, (4.6) holds if and only if \( c_j - \tilde{\gamma} H_j y \leq 0 \) for every \( j \in J \); that is, if and only if \( \tilde{\gamma} \geq \alpha_1 \). This completes the proof of Part 1.

**Part 2.** Since \( w^T x = u_a \), the cone of first-order feasible directions at \( x \) has the constraint \( w^T d \leq 0 \). Consequently, \( e_i \notin F_a(x) \cap D \) for any \( i \), and the first-order optimality condition (C1) for \( f_\gamma \) reduces to (4.6)–(4.7). As in Part 1, (4.6) holds, since \( \tilde{\gamma} \leq \gamma \). Condition (4.7) is equivalent to \( \tilde{\gamma} \in \Gamma \). Since (4.7) holds for \( f = f_\gamma \), we have \( \gamma \in \Gamma \). Since \( \nabla_x f_\gamma(x, y) \) 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 \( \tilde{\gamma} \leq \gamma \in \Gamma \), we have \( \tilde{\gamma} \in \Gamma \) if and only if \( \tilde{\gamma} \geq \alpha_2 \). This completes the proof of Part 2.

**Remark 4.1:** Of course, Proposition 4.3 also holds when the variables \( x \) and \( y \) and the bounds \( (\ell_a, u_a) \) and \( (\ell_b, u_b) \) are interchanged. In most applications, \( u_a \leq b(V) / 2 \), \( \ell_a = \ell_b = 1 \), and as the iterates converge to a solution of (4.15), either the constraint \( w^T x \leq u_a \) or the constraint \( w^T y \leq u_b \) is active. Thus, for a given iterate \( (x, y) \), the assumptions of Part 1 typically apply to either \( x \) or \( y \), 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 \( \gamma \); that is, as long as \( \tilde{\gamma} \geq 0 \). Thus, \( \alpha_2 = 0 \). Since \( \alpha_1 \) is typically positive, the tighter bound on \( \tilde{\gamma} \) is the interval \( [\alpha_1, \gamma] \), which means that when \( \tilde{\gamma} < \alpha_1 \), \( (x, y) \) is no longer a stationary point for \( f = f_\gamma \).

Algorithm 4.3 also denoted MCA_{GR}, approximately solves (4.15), while incorporating both c-perturbations and \( \gamma \)-refinements. Here, the notation MCA\((x, y, \tilde{\gamma})\) indicates that the MCA algorithm is applied to the point \( (x, y) \) using \( \tilde{\gamma} \) in place of \( \gamma \) as the penalty parameter. In our implementation, the \( \gamma \)-refinements are performed in the following way: \( \gamma \) is reduced from the initial value \( \alpha_1 \) 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 \( \gamma \)-refinement procedure in order to obtain a local optimizer of high quality for each \( \tilde{\gamma} \).
Input: A feasible point \((x, y)\) for (3.15).

\((x, y) \leftarrow \text{MCA\_CP} (x, y)\)

\(\tilde{\gamma} \leftarrow \alpha_1\)

while \((\tilde{\gamma} > 0)\)

\(\tilde{\gamma} \leftarrow \text{reduce} (\tilde{\gamma})\)

\((\tilde{x}, \tilde{y}) \leftarrow \text{MCA\_CP} (x, y, \tilde{\gamma})\)

\((x^*, y^*) \leftarrow \text{MCA\_CP} (\tilde{x}, \tilde{y}, \gamma)\)

if \((f(x^*, y^*) > f(x, y))\)

\((x, y) \leftarrow (x^*, y^*)\)

end while

return \((x, y)\)

Algorithm 4.3. MCA\_GR: A refinement algorithm for (3.15) which incorporates \(c\)-perturbations and \(\gamma\)-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 \(x_i = u_a/W(V)\) and \(y_i = u_b/W(V)\), \(i = 1, 2, \ldots, n\). 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
A MULTILEVEL ALGORITHM FOR THE VERTEX SEPARATOR PROBLEM

next finer graph is obtained by unmatching vertices in the coarser graph. Suppose that we are uncoarsening from level \( l \) to \( l - 1 \) and \((x_l^l, y_l^l)\) 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 \((x_{l-1}^l, y_{l-1}^l)\) is simply \((x_{l-1}^l j, y_{l-1}^l j) = (x_{l}^l i, y_{l}^l i)\) and \((x_{l-1}^l k, y_{l-1}^l k) = (x_{l}^l i, y_{l}^l i)\).

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

\[
\text{METIS\_ComputeVertexSeparator},
\]

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

- \text{METIS\_IPTYPE\_NODE} (coarsest problem solved with node growth scheme),
- \text{METIS\_RTYPE\_SEP2SIDED} (Fiduccia-Mattheyses-like refinement scheme).

In a preliminary experiment, we also considered the refinement option \text{METIS\_RTYPE\_SEP1SIDED}, but the results obtained were not significantly different. On the average, the option \text{METIS\_RTYPE\_SEP2SIDED} provides slightly better results. Additionally, we considered both heavy-edge matching (\text{METIS\_CTYPE\_SHEM}) and random matching (\text{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 \( 1.43 \times 10^{-6} \) to \( 1.32 \times 10^{-2} \), where sparsity is defined as the ratio \( |E| / n(n-1) \) (recall that \(|E|\) 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 (\(|E|/2\)), 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 (A and 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 \( l_a = l_b = 1 \) 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 \( r \).
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 \((\tilde{x}, \tilde{y})\) to the VSP by calling the multilevel algorithm \textsc{METIS\_ComputeVertexSeparator}. Next, refine \((\tilde{x}, \tilde{y})\) by calling \textsc{MCA\_GR}.

2. Obtain an approximate solution \((\tilde{x}, \tilde{y})\) to the VSP by invoking the multilevel algorithm \textsc{BLP}. Next, refine \((\tilde{x}, \tilde{y})\) by calling \textsc{METIS\_NodeRefine}.

### Table 6.1

| Graph   | \(|V|\) | \(|E|/2\) | Sparsity | Min  | Max  | Ave  |
|---------|--------|----------|----------|-------|------|------|
| bspwr09 | 1723   | 2394     | 1.61E-03 | 1     | 14   | 2.78 |
| bcsstk17| 10974  | 20883    | 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   | 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 |
| stmodel | 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 6.2

| Graph        | \(|V|\) | \(|E|/2\) | 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 |

Heavy-tailed degree distribution graphs from the SNAP database.
Table 6.3

| Graph                  | |V| | |E|/2 | Sparsity | Min | Max | Ave |
|------------------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| barth5_1Ksep50in50Kout | 32212           | 1011805         | 1.96E-04        | 1               | 22              | 6.32            |
| bcstk30_500sep100in1Kout | 58348           | 2016578         | 1.18E-03        | 0               | 219             | 69.12           |
| befrefem2_1ran1air02  | 14109           | 98224           | 9.87E-04        | 1               | 1531            | 13.92           |
| bump21ran1ma01model1crew1 | 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           |
| fin512_firelddd         | 139752          | 552020          | 5.65E-05        | 1               | 669             | 7.9             |
| mod2_pgp2_lptsk        | 101364          | 389368          | 7.58E-05        | 1               | 1901            | 7.68            |
| model1_crew1_cr42_south31 | 45101          | 189976          | 1.87E-04        | 1               | 17663           | 8.42            |
| msc10848_300sep100in1Kout | 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_lptsk          | 39668           | 189914          | 2.41E-04        | 1               | 17663           | 9.58            |
| vibrobox_firelddd       | 77328           | 435586          | 1.46E-04        | 1               | 669             | 11.27           |

Table 6.4

| Graph                  | |V| | |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 6.5

| Graph                  | |V| | |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            |
METIS\_NodeRefine is a refinement routine which improves upon an initial solution using a Fiduccia-Mattheyses-style refinement: Vertices in $S$ are moved into either $A$ or $B$ and their neighbors in the opposite shore are moved into $S$. 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 $S$ 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 $100(\frac{C(S_{initial}) - C(S_{final})}{C(V)})$. In Experiment 1, we observed that in every initial solution obtained by METIS\_ComputeVertexSeparator, the upper bounds on both of the sets $A$ and $B$ 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 $C(S)$ 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 $100(\frac{C(S_{METIS}) - C(S_{BLP})}{C(V)})$. 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 $n = |V|$ 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 $A$ and $x$ or $y$), and Karush-Kuhn-Tucker multiplier computations, which were used to determine the perturbations in $c$ or $\gamma$ 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 $Ax_k$ and $Ay_k$ between the matrix and a vector. We could save the previous products $Ax_{k-1}$ and $Ay_{k-1}$ 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
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 \(\epsilon\)-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 \(\gamma\)-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. Kayaaaslan, 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. Hajaghayi, and J. Lee, Improved approximation algorithms for vertex separators, SIAM J. Comput., 38 (2008), pp. 629–657.

[13] C. M. Fiduccia and R. M. Matheyses, 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] Continuous quadratic programming formulations of optimization problems on graphs, European J. Oper. Res., 240 (2015), pp. 328–337.

[23] E. Kayaaaslan, 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\url{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
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.

J. Leskovec and A. Krevl, *SNAP Datasets: Stanford large network dataset collection*, June 2014.

C. Litsas, A. Pagourtzis, G. Panagiotakos, and D. Sakavalas, *On the resilience and uniqueness of CPA for secure broadcast*, IACR Cryptology ePrint Archive, (2013).

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.

J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, 2nd ed., 2006.

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.

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.

D. Ron, I. Safro, and A. Brandt, *Relaxation-based coarsening and multiscale graph organization*, Multiscale Modeling & Simulation, 9 (2011), pp. 407–422.

I. Safro, D. Ron, and A. Brandt, *Graph minimum linear arrangement by multilevel weighted edge contractions*, J. Algorithms, 60 (2006), pp. 24–41.

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.

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      |
| 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_r13 | 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       |
| sstmodel20| 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 |
|-----------|------|----------|--------|----------|
|           |      | avg      | min    | max      | avg      | min    | max      |
| 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_r13 | 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       |
| sstmodel20| 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 $\mathcal{C}(S)$ 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_RFM |
|--------------|------|----------|--------|---------|
|              |      | avg      | min    | max     | avg      | min    | max     | avg      | min    | max     |
| 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 $C(S)$ for HTDD graphs.
### Table 6.9

| Graph                  | Best | METIS_HE | BLP_HE | BLP_HEFM |
|------------------------|------|----------|--------|----------|
|                        | avg  | min | max    | avg  | min | max    |
| vsp_barth5_1Ksep_50in_5Kout | 987  | 1346.14 | 1043 | 1530    | 1549.01 | 1326 | 1818   |
| vsp_bcsstk30_500sep_10in_1Kout | 528  | 636.70 | 528 | 844   | 627.95 | 565 | 854    |
| vsp_befref_xm_2_air02 | 270  | 1464.03 | 1328 | 1584    | 288.35 | 274 | 314    |
| vsp_jump2_p18_ao1_model1_crew1 | 3849 | 4378.39 | 4280 | 4793    | 3986.52 | 3897 | 4389   |
| vsp_vodata_citizens4 | 453  | 536.25 | 532 | 542    | 516.66 | 511 | 528    |
| vsp_vodata_and_seymourl | 2222 | 2636.81 | 2384 | 2741    | 2869.78 | 2345 | 3545   |
| vsp_vadata_3049992 | 4605 | 7438.78 | 7216 | 7697    | 7443.46 | 5767 | 9711   |
| vsp_vadata_442_south31 | 1838 | 2216.08 | 2086 | 2631    | 2507.86 | 2424 | 2576   |
| vsp_vmsc10848_100in_1Kout | 279  | 648.16 | 279 | 929    | 723.60 | 343 | 1421   |
| vsp_vadata_and_seymourl | 511  | 536.25 | 532 | 542    | 516.66 | 511 | 528    |
| vsp_vdata_702_sametek | 3390 | 4114.48 | 3831 | 4373    | 3921.85 | 3766 | 4030   |
| vsp_vdata_and_seymourl | 1971 | 2054.39 | 1982 | 2116    | 2328.52 | 2242 | 2567   |
| vsp_vibrobox_scagr7-2_lfddd | 2762 | 3467.55 | 3362 | 3613    | 2856.32 | 2801 | 2992   |

**Vertex Separator Costs C(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 |
|--------------|------|----------|--------|----------|
|              | avg  | min | max  | avg | min | max  | avg | min | max  |
| 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 $C(S)$ for peer-to-peer networks.

**Table 6.11**

Vertex Separator Costs $C(S)$ for Konect networks.
### Table 6.12
Comparison of separator costs $C(S)$ obtained by BLP\_RM and METIS\_RM.

| Graph Type | % BLP Wins | % Improvement |
|------------|-------------|---------------|
|            |             | 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 6.13
Comparison of separator costs $C(S)$ obtained by BLP\_RMFM and METIS\_RM.

| Graph Type | % BLP Wins | % Improvement |
|------------|-------------|---------------|
|            |             | 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 6.14
Comparison in separator costs $C(S)$ obtained by BLP\_HE and METIS\_HE.

| Graph Type | % BLP Wins | % Improvement |
|------------|-------------|---------------|
|            |             | 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 6.15
Comparison of separator costs $C(S)$ obtained by BLP\_HEFM and METIS\_HE.

| Graph Type | % BLP Wins | % Improvement |
|------------|-------------|---------------|
|            |             | 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 |
| Graph Type | BLP_R | METIS_R |
|------------|-------|---------|
|           | 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.