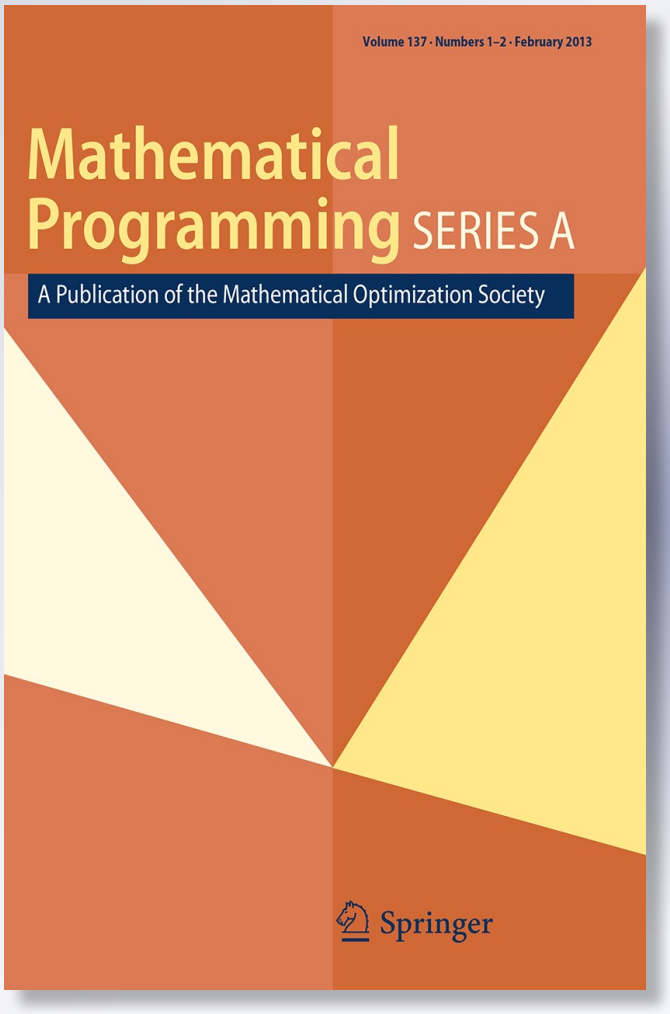
An exact algorithm for graph partitioning

William W. Hager, Dzung T. Phan & Hongchao Zhang

Mathematical Programming
A Publication of the Mathematical Optimization Society

ISSN 0025-5610
Volume 137
Combined 1-2

Math. Program. (2013) 137:531-556
DOI 10.1007/s10107-011-0503-x



Springer

Math. Program., Ser. A (2013) 137:531–556
DOI 10.1007/s10107-011-0503-x

FULL LENGTH PAPER

An exact algorithm for graph partitioning

William W. Hager · Dzung T. Phan · Hongchao Zhang

Received: 16 March 2011 / Accepted: 8 November 2011 / Published online: 30 November 2011

Abstract An exact algorithm is presented for solving edge weighted graph partitioning problems. The algorithm is based on a branch and bound method applied to a continuous quadratic programming formulation of the problem. Lower bounds are obtained by decomposing the objective function into convex and concave parts and replacing the concave part by an affine underestimate. It is shown that the best affine underestimate can be expressed in terms of the center and the radius of the smallest sphere containing the feasible set. The concave term is obtained either by a constant diagonal shift associated with the smallest eigenvalue of the objective function Hessian, or by a diagonal shift obtained by solving a semidefinite programming problem. Numerical results show that the proposed algorithm is competitive with state-of-the-art graph partitioning codes.

November 23, 2009. Revised October 13, 2011. This research was partly supported by National Science Foundation Grant 0620286 and by Office of Naval Research Grant N00014-11-1-0068.

W. W. Hager (✉)
Department of Mathematics, University of Florida, PO Box 118105,
Gainesville, FL 32611-8105, USA
e-mail: hager@ufl.edu
URL: http://www.math.ufl.edu/∼hager

D. T. Phan
Department of Business Analytics and Mathematical Sciences, IBM T.J. Watson Research Center,
PO Box 218, Yorktown Heights, NY 10598, USA
e-mail: phandu@us.ibm.com

H. Zhang
Department of Mathematics, Center for Computation and Technology, Louisiana State University,
140 Lockett Hall, Baton Rouge, LA 70803-4918, USA
e-mail: hozhang@math.lsu.edu
URL: http://www.math.lsu.edu/∼hozhang

Keywords Graph partitioning · Min-cut · Quadratic programming · Branchand bound · Affine underestimate

Mathematics Subject Classification (2000) 90C35 · 90C20 · 90C27 · 90C46

1 Introduction

Given a graph with edge weights, the graph partitioning problem is to partition the vertices into two sets satisfying specified size constraints, while minimizing the sum of the weights of the edges that connect the vertices in the two sets. Graph partitioning problems arise in many areas including VLSI design, data mining, parallel computing, and sparse matrix factorizations [17,27,32,43]. The graph partitioning problem is NP-hard [14].

There are two general classes of methods for the graph partitioning problem, exact methods which compute the optimal partition, and heuristic methods which try to quickly compute an approximate solution. Heuristic methods include spectral methods [22], geometric methods [15], multilevel schemes [23], optimization-based methods [11], and methods that employ randomization techniques such as genetic algorithms [41]. Software which implements heuristic methods includes Metis [29–31], Chaco [21], Party [37], PaToH [6], SCOTCH [36], Jostle [44], Zoltan [9], and HUND [16].

This paper develops an exact algorithm for the graph partitioning problem. In earlier work, Brunetta et al. [5] propose a branch-and-cut scheme based on a linear programming relaxation and subsequent cuts based on separation techniques. A column generation approach is developed by Johnson, Mehrotra, and Nemhauser [26], while Mitchell [33] develops a polyhedral approach. Karisch, Rendl, and Clausen [28] develop a branch-and-bound method utilizing a semidefinite programming relaxation to obtain a lower bound. Sensen [39] develops a branch-and-bound method based on a lower bound obtained by solving a multicommodity flow problem. Armbruster et al. [1,2] develop methods based on linear and semidefinite relaxations combined with separation routines for valid inequalities associated with the bisection cut polytope.

In this paper, we develop a branch-and-bound algorithm based on a quadratic programming (QP) formulation of the graph partitioning problem. The objective function of the QP is expressed as the sum of a convex and a concave function. We consider two different techniques for making this decomposition, one based on eigenvalues and the other based on semidefinite programming. In each case, we give an affine underestimate for the concave function, which leads to a tractable lower bound in the branch and bound algorithm.

The paper is organized as follows. In Sect. 2 we review the continuous quadratic programming formulation of the graph partitioning problem developed in [17] and we explain how to associate a solution of the continuous problem with the solution to the discrete problem. In Sect. 3 we discuss approaches for decomposing the objective function for the QP into the sum of convex and a concave functions, and in each case, we show how to generate an affine lower bound for the concave part. Section 4 gives the branch-and-bound algorithm, while Sect. 5 provides necessary and

sufficient conditions for a local minimizer. Section 6 compares the performance of the new branch-and-bound algorithm to earlier results given in [1,28,38], and [39].

Notation Throughout the paper, $\|\cdot\|$ denotes the Euclidian norm. $\mathbf{1}$ is the vector whose entries are all 1. The dimension will be clear from context. If $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{A} \succeq \mathbf{0}$ means that $\mathbf{A}$ is positive semidefinite. We let $\mathbf{e}_i$ denote the i-th column of the identity matrix; again, the dimension will be clear from context. If $\mathcal{S}$ is a set, then $|\mathcal{S}|$ is the number of elements in $\mathcal{S}$. The gradient $\nabla f(\mathbf{x})$ is a row vector.

2 Continuous quadratic programming formulation

Let G be a graph with n vertices

$$\mathcal{V} = \{1, 2, \ldots, n\},$$

and let a_{ij} be a weight associated with the edge connecting vertices i and j. When there is no edge between i and j, we set $a_{ij} = 0$. For each i and j, we assume that $a_{ii} = 0$ and $a_{ij} = a_{ji}$; in other words, we consider an undirected graph without self loops (a simple, undirected graph). The sign of the weights is not restricted, and in fact, a_{ij} could be negative, as it would be in the max-cut problem. Given integers l and u such that $0 \le l \le u \le n$, we wish to partition the vertices into two disjoint sets, with between l and u vertices in one set, while minimizing the sum of the weights associated with edges connecting vertices in different sets. The edges connecting the two sets in the partition are referred to as the cut edges, and the optimal partition minimizes the sum of the weights of the cut edges. Hence, the graph partitioning problem is also called the min-cut problem.

In [17] we show that for a suitable choice of the diagonal matrix $\mathbf{D}$, the graph partitioning problem is equivalent to the following continuous quadratic programming problem:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) := (\mathbf{1} - \mathbf{x})^{\mathsf{T}}(\mathbf{A} + \mathbf{D})\mathbf{x} \\ \text{subject to} \quad & \mathbf{0} \le \mathbf{x} \le \mathbf{1}, \quad l \le \mathbf{1}^{\mathsf{T}}\mathbf{x} \le u, \end{aligned} \tag{2.1}$$

where $\mathbf{A}$ is the matrix with elements a_{ij}. Suppose $\mathbf{x}$ is binary and let us define the sets

$$\mathcal{V}_0 = \{i : x_i = 0\} \quad \text{and} \quad \mathcal{V}_1 = \{i : x_i = 1\}. \tag{2.2}$$

It can be checked that $f(\mathbf{x})$ is the sum of the weights of the cut edges associated with the partition (2.2). Hence, if we add the restriction that $\mathbf{x}$ is binary, then (2.1) is exactly equivalent to finding the partition which minimizes the weight of the cut edges. Note, though, that there are no binary constraints in (2.1). The equivalence between (2.1) and the graph partitioning problem is as follows (see [17, Thm. 2.1]):

Theorem 2.1 *If the diagonal matrix* $\mathbf{D}$ *is chosen so that*

$$d_{ii} + d_{jj} \ge 2a_{ij} \quad \textit{and} \quad d_{ii} \ge 0 \tag{2.3}$$

for each i and j, then (2.1) has a binary solution $\mathbf{x}$ and the partition given by (2.2) is a min-cut.

The generalization of this result to multiset partitioning is given in [18]. The condition (2.3) is satisfied, for example, by the choice

$$d_{jj} = \max \{0, a_{1j}, a_{2j}, \ldots, a_{nj}\}$$

for each j. The proof of Theorem 2.1 was based on showing that any solution to (2.1) could be transformed to a binary solution without changing the objective function value. With a modification of this idea, any feasible point can be transformed to a binary feasible point without increasing the objective function value. We now give a constructive proof of this result, which is used when we solve (2.1).

Corollary 2.2 *If $\mathbf{x}$ is feasible in (2.1) and the diagonal matrix $\mathbf{D}$ satisfies (2.3), then there exists a binary $\mathbf{y}$ with $f(\mathbf{y}) \le f(\mathbf{x})$ and $y_i = x_i$ whenever x_i is binary.*

Proof We first show how to find $\mathbf{z}$ with the property that $\mathbf{z}$ is feasible in (2.1), $f(\mathbf{z}) \le f(\mathbf{x})$, $\mathbf{1}^{\mathsf{T}}\mathbf{z}$ is integer, and the only components of $\mathbf{z}$ and $\mathbf{x}$ which differ are the fractional components of $\mathbf{x}$. If $\mathbf{1}^{\mathsf{T}}\mathbf{x} = u$ or $\mathbf{1}^{\mathsf{T}}\mathbf{x} = l$, then we are done since l and u are integers; hence, we assume that $l < \mathbf{1}^{\mathsf{T}}\mathbf{x} < u$. If all components of $\mathbf{x}$ are binary, then we are done, so suppose that there exists a nonbinary component x_i. Since $a_{ii} = 0$, a Taylor expansion of f gives

$$f(\mathbf{x} + \alpha \mathbf{e}_i) = f(\mathbf{x}) + \alpha \nabla f(\mathbf{x})_i - \alpha^2 d_{ii},$$

where $\mathbf{e}_i$ is the i-th column of the identity matrix. The quadratic term in the expansion is nonpositive since $d_{ii} \ge 0$. If the first derivative term is negative, then increase α above 0 until either $x_i + \alpha$ becomes 1 or $\mathbf{1}^{\mathsf{T}}\mathbf{x} + \alpha$ is an integer. Since the first derivative term is negative and $\alpha > 0$, $f(\mathbf{x} + \alpha \mathbf{e}_i) < f(\mathbf{x})$. If $\mathbf{1}^{\mathsf{T}}\mathbf{x} + \alpha$ becomes an integer, then we are done. If $x_i + \alpha$ becomes 1, then we reach a point $\mathbf{x}_1$ with one more binary component and with an objective function value no larger than $f(\mathbf{x})$. If the first derivative term is nonnegative, then decrease α below 0 until either $x_i + \alpha$ becomes 0 or $\mathbf{1}^{\mathsf{T}}\mathbf{x} + \alpha$ is an integer. Since the first derivative term is nonnegative and $\alpha < 0$, $f(\mathbf{x} + \alpha \mathbf{e}_i) \le f(\mathbf{x})$. If $\mathbf{1}^{\mathsf{T}}\mathbf{x} + \alpha$ becomes an integer, then we are done. If $x_i + \alpha$ becomes 0, then we reach a point $\mathbf{x}_1$ with one more binary component and with a smaller value for the cost function. In this latter case, we choose another nonbinary component of $\mathbf{x}_1$ and repeat the process. Hence, there is no loss of generality in assuming that $\mathbf{1}^{\mathsf{T}}\mathbf{x}$ is an integer.

Suppose that $\mathbf{x}$ is not binary. Since $\mathbf{1}^{\mathsf{T}}\mathbf{x}$ is an integer, $\mathbf{x}$ must have at least two nonbinary components, say x_i and x_j. Again, expanding f in a Taylor series gives

$$f(\mathbf{x} + \alpha(\mathbf{e}_i - \mathbf{e}_j)) = f(\mathbf{x}) + \alpha(\nabla f(\mathbf{x})_i - \nabla f(\mathbf{x})_j) + \alpha^2(2a_{ij} - d_{ii} - d_{jj}).$$

By (2.3), the quadratic term is nonpositive for any choice of α. If the first derivative term is negative, then we increase α above 0 until either $x_i + \alpha$ reaches 1 or $x_j - \alpha$ reach 0. Since the first derivative term is negative and $\alpha > 0$, we have $f(\mathbf{x} + \alpha(\mathbf{e}_i - \mathbf{e}_j)) < f(\mathbf{x})$.

If the first derivative term is nonnegative, then we decrease α below 0 until either $x_i + \alpha$ reaches 0 or $x_j - \alpha$ reach 1. Since the first derivative term is nonnegative and $\alpha < 0$, it follows that $f(\mathbf{x} + \alpha(\mathbf{e}_i - \mathbf{e}_j)) \leq f(\mathbf{x})$. In either case, the value of the cost function does not increase, and we reach a feasible point $\mathbf{x}_1$ with $\mathbf{1}^\mathsf{T}\mathbf{x}_1$ integer and with at least one more binary component. If $\mathbf{x}_1$ is not binary, then $\mathbf{x}_1$ must have at least two nonbinary components; hence, the adjustment process can be continued until all the components of $\mathbf{x}$ are binary. These adjustments to $\mathbf{x}$ do not increase the value of the cost function and we only alter the fractional components of $\mathbf{x}$. This completes the proof. □

3 Convex lower bounds for the objective function

We compute an exact solution to the continuous formulation (2.1) of graph partitioning problem using a branch and bound algorithm. The bounding process requires a lower bound for the objective function when restricted to the intersection of a box and two half spaces. This lower bound is obtained by writing the objective function as the sum of a convex and a concave function and by replacing the concave part by the best affine underestimate. Two different strategies are given for decomposing the objective function.

3.1 Lower bound based on minimum eigenvalue

Let us decompose the objective function $f(\mathbf{x}) = (\mathbf{1} - \mathbf{x})^\mathsf{T}(\mathbf{A} + \mathbf{D})\mathbf{x}$ in the following way:

$$f(\mathbf{x}) = (f(\mathbf{x}) + \sigma\|\mathbf{x}\|^2) - \sigma\|\mathbf{x}\|^2,$$

where σ is the maximum of 0 and the largest eigenvalue of $\mathbf{A} + \mathbf{D}$. This represents a DC (difference convex) decomposition (see [24]) since $f(\mathbf{x}) + \sigma\|\mathbf{x}\|^2$ and $\sigma\|\mathbf{x}\|^2$ are both convex. The concave term $-\|\mathbf{x}\|^2$ is underestimated by an affine function ℓ to obtain a convex underestimate f_L of f given by

$$f_L(\mathbf{x}) = \left(f(\mathbf{x}) + \sigma\|\mathbf{x}\|^2\right) + \sigma\ell(\mathbf{x}). \tag{3.1}$$

We now consider the problem of finding the best affine underestimate ℓ for the concave function $-\|\mathbf{x}\|^2$ over a given compact, convex set denoted $\mathcal{C}$. The set of affine underestimators for $-\|\mathbf{x}\|^2$ is given by

$$\mathcal{S}_1 = \{\ell : \mathbb{R}^n \to \mathbb{R} \text{ such that } \ell \text{ is affine and } -\|\mathbf{x}\|^2 \geq \ell(\mathbf{x}) \text{ for all } \mathbf{x} \in \mathcal{C}\}.$$

The best affine underestimate is a solution of the problem

$$\min_{\ell \in \mathcal{S}_1} \max_{\mathbf{x} \in \mathcal{C}} -\left(\|\mathbf{x}\|^2 + \ell(\mathbf{x})\right). \tag{3.2}$$

The following result generalizes Theorem 3.1 in [19] where we determine the best affine underestimate for $-\|\mathbf{x}\|^2$ over an ellipsoid.

Theorem 3.1 *Let $\mathcal{C} \subset \mathbb{R}^n$ be a compact, convex set and let $\mathbf{c}$ be the center and r be the radius of the smallest sphere containing $\mathcal{C}$. This smallest sphere is unique and a solution of (3.2) is*

$$\ell^*(\mathbf{x}) = -2\mathbf{c}^\mathsf{T}\mathbf{x} + \|\mathbf{c}\|^2 - r^2.$$

Furthermore,

$$\min_{\ell \in \mathcal{S}_1} \max_{\mathbf{x} \in \mathcal{C}} \; -\left(\|\mathbf{x}\|^2 + \ell^*(\mathbf{x})\right) = r^2.$$

Proof To begin, we will show that the minimization in (3.2) can be restricted to a compact set. Clearly, when carrying out the minimization in (3.2), we should restrict our attention to those ℓ which touch the function $h(\mathbf{x}) := -\|\mathbf{x}\|^2$ at some point in $\mathcal{C}$. Let $\mathbf{y} \in \mathcal{C}$ denote the point of contact. Since $h(\mathbf{x}) \geq \ell(\mathbf{x})$ and $h(\mathbf{y}) = \ell(\mathbf{y})$, a lower bound for the error $h(\mathbf{x}) - \ell(\mathbf{x})$ over $\mathbf{x} \in \mathcal{C}$ is

$$h(\mathbf{x}) - \ell(\mathbf{x}) \geq |\ell(\mathbf{x}) - \ell(\mathbf{y})| - |h(\mathbf{x}) - h(\mathbf{y})|.$$

If M is the difference between the maximum and minimum value of h over $\mathcal{C}$, then we have

$$h(\mathbf{x}) - \ell(\mathbf{x}) \geq |\ell(\mathbf{x}) - \ell(\mathbf{y})| - M. \tag{3.3}$$

An upper bound for the minimum in (3.2) is obtained by the linear function ℓ_0 which is constant on $\mathcal{C}$, with value equal to the minimum of $h(\mathbf{x})$ over $\mathbf{x} \in \mathcal{C}$. If $\mathbf{w}$ is a point where h attains its minimum over $\mathcal{C}$, then we have

$$\max_{\mathbf{x} \in \mathcal{C}} \; h(\mathbf{x}) - \ell_0(\mathbf{x}) = \max_{\mathbf{x} \in \mathcal{C}} \; h(\mathbf{x}) - h(\mathbf{w}) = M.$$

Let us restrict our attention to the linear functions ℓ which achieve an objective function value in (3.2) which is at least as small as that of ℓ_0. For these ℓ and for $\mathbf{x} \in \mathcal{C}$, we have

$$h(\mathbf{x}) - \ell(\mathbf{x}) \leq \max_{\mathbf{x} \in \mathcal{C}} \; h(\mathbf{x}) - \ell(\mathbf{x}) \leq \max_{\mathbf{x} \in \mathcal{C}} \; h(\mathbf{x}) - \ell_0(\mathbf{x}) = M. \tag{3.4}$$

Combining (3.3) and (3.4) gives

$$|\ell(\mathbf{x}) - \ell(\mathbf{y})| \leq 2M. \tag{3.5}$$

Thus, when we carry out the minimization in (3.2), we should restrict our attention to linear functions which touch h at some point $\mathbf{y} \in \mathcal{C}$ and with the change in ℓ across

$\mathcal{C}$ satisfying the bound (3.5) for all $\mathbf{x} \in \mathcal{C}$. This tells us that the minimization in (3.2) can be restricted to a compact set, and that a minimizer must exist.

Suppose that ℓ attains the minimum in (3.2). Let $\mathbf{z}$ be a point in $\mathcal{C}$ where $h(\mathbf{x}) - \ell(\mathbf{x})$ achieves its maximum. A Taylor expansion around $\mathbf{x} = \mathbf{z}$ gives

$$h(\mathbf{x}) - \ell(\mathbf{x}) = h(\mathbf{z}) - \ell(\mathbf{z}) + (\nabla h(\mathbf{z}) - \nabla \ell)(\mathbf{x} - \mathbf{z}) - \|\mathbf{x} - \mathbf{z}\|^2.$$

Since $\ell \in \mathcal{S}_1$, $h(\mathbf{x}) - \ell(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in \mathcal{C}$. It follows that

$$h(\mathbf{z}) - \ell(\mathbf{z}) \geq -(\nabla h(\mathbf{z}) - \nabla \ell)(\mathbf{x} - \mathbf{z}) + \|\mathbf{x} - \mathbf{z}\|^2. \tag{3.6}$$

Since $\mathcal{C}$ is convex, the first-order optimality conditions for $\mathbf{z}$ give

$$(\nabla h(\mathbf{z}) - \nabla \ell)(\mathbf{x} - \mathbf{z}) \leq 0$$

for all $\mathbf{x} \in \mathcal{C}$. It follows from (3.6) that

$$h(\mathbf{z}) - \ell(\mathbf{z}) \geq \|\mathbf{x} - \mathbf{z}\|^2 \tag{3.7}$$

for all $\mathbf{x} \in \mathcal{C}$. There exists $\mathbf{x} \in \mathcal{C}$ such that $\|\mathbf{x} - \mathbf{z}\| \geq r$ or else $\mathbf{z}$ would be the center of a smaller sphere containing $\mathcal{C}$. Hence, (3.7) implies that

$$h(\mathbf{z}) - \ell(\mathbf{z}) \geq r^2.$$

It follows that

$$\max_{\mathbf{x} \in \mathcal{C}} \; h(\mathbf{x}) - \ell(\mathbf{x}) \geq h(\mathbf{z}) - \ell(\mathbf{z}) \geq r^2. \tag{3.8}$$

We now observe that for the specific linear function ℓ^* given in the statement of the theorem, (3.8) becomes an equality, which implies the optimality of ℓ^* in (3.2). Expand h in a Taylor series around $\mathbf{x} = \mathbf{c}$ to obtain

$$\begin{aligned} h(\mathbf{x}) &= -\|\mathbf{c}\|^2 - 2\mathbf{c}^{\mathsf{T}}(\mathbf{x} - \mathbf{c}) - \|\mathbf{x} - \mathbf{c}\|^2 \\ &= -2\mathbf{c}^{\mathsf{T}}\mathbf{x} + \|\mathbf{c}\|^2 - \|\mathbf{x} - \mathbf{c}\|^2. \end{aligned}$$

Subtract $\ell^*(\mathbf{x}) = -2\mathbf{c}^{\mathsf{T}}\mathbf{x} + \|\mathbf{c}\|^2 - r^2$ from both sides to obtain

$$h(\mathbf{x}) - \ell^*(\mathbf{x}) = r^2 - \|\mathbf{x} - \mathbf{c}\|^2. \tag{3.9}$$

If $\mathbf{c} \in \mathcal{C}$, then the maximum in (3.9) over $\mathbf{x} \in \mathcal{C}$ is attained by $\mathbf{x} = \mathbf{c}$ for which

$$h(\mathbf{c}) - \ell^*(\mathbf{c}) = r^2.$$

Consequently, (3.8) becomes an equality for $\ell = \ell^*$, which implies the optimality of ℓ^* in (3.2).

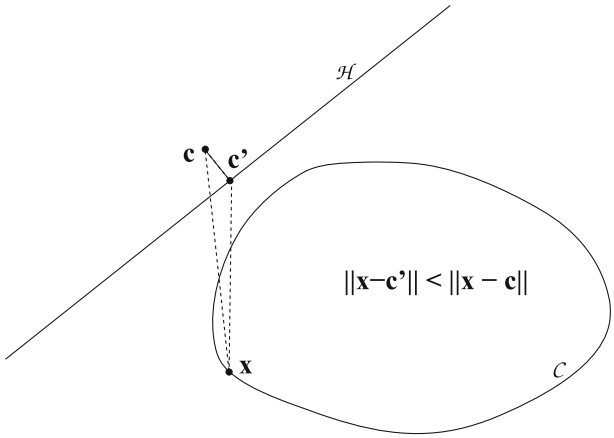


Fig. 1 Suppose $\mathbf{c} \notin \mathcal{C}$

We can show that $\mathbf{c} \in \mathcal{C}$ as follows: Suppose $\mathbf{c} \notin \mathcal{C}$. Since $\mathcal{C}$ is compact and convex, there exists a hyperplane $\mathcal{H}$ strictly separating $\mathbf{c}$ and $\mathcal{C}$—see Fig. 1 If $\mathbf{c}'$ is the projection of $\mathbf{c}$ onto $\mathcal{H}$, then

$$\|\mathbf{x} - \mathbf{c}'\| < \|\mathbf{x} - \mathbf{c}\| \quad \text{for all} \quad \mathbf{x} \in \mathcal{C}. \tag{3.10}$$

Let $\mathbf{x}' \in \mathcal{C}$ be the point which is farthest from $\mathbf{c}'$ and let $\mathbf{x} \in \mathcal{C}$ be the point farthest from $\mathbf{c}$. Hence, $\|\mathbf{x} - \mathbf{c}\| = r$. By (3.10), we have $\|\mathbf{x}' - \mathbf{c}'\| < \|\mathbf{x} - \mathbf{c}\| = r$; it follows that the sphere with center $\mathbf{c}'$ and radius $\|\mathbf{x}' - \mathbf{c}'\|$ contains $\mathcal{C}$ and has radius smaller than r. This contradicts the assumption that r was the sphere of smallest radius containing $\mathcal{C}$.

The uniqueness of the smallest sphere containing $\mathcal{C}$ is as follows: Suppose that there exist two different smallest spheres $\mathcal{S}_1$ and $\mathcal{S}_2$ containing $\mathcal{C}$. Let $\mathcal{S}_3$ be the smallest sphere containing $\mathcal{S}_1 \cap \mathcal{S}_2$. Since the diameter of the intersection is strictly less than the diameter of $\mathcal{S}_1$ or $\mathcal{S}_2$, we contradict the assumption that $\mathcal{S}_1$ and $\mathcal{S}_2$ were spheres of smallest radius containing $\mathcal{C}$. □

Remark 1 Although the smallest sphere containing $\mathcal{C}$ in Theorem 3.1 is unique, the best linear underestimator of $h(\mathbf{x}) = -\|\mathbf{x}\|^2$ is not unique. For example, suppose $\mathbf{a}$ and $\mathbf{b} \in \mathbb{R}^n$ and $\mathcal{C}$ is the line segment

$$\mathcal{C} = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} = \alpha\mathbf{a} + (1 - \alpha)\mathbf{b}, \quad \alpha \in [0, 1]\}.$$

Along this line segment, h is a concave quadratic in one variable. The best affine underestimate along the line segment corresponds to the line connecting the ends of the quadratic restricted to the line segment. Hence, in $\mathbb{R}^{n+1}$, any hyperplane which contains the points $(h(\mathbf{a}), \mathbf{a})$ and $(h(\mathbf{b}), \mathbf{b})$ leads to a best affine underestimate.

Remark 2 Let $\mathcal{C}$ be the box

$$\mathcal{B} = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{p} \le \mathbf{x} \le \mathbf{q}\}.$$

The diameter of $\mathcal{B}$, the distance between the points in $\mathcal{B}$ with greatest separation, is $\|\mathbf{p}-\mathbf{q}\|$. Hence, the smallest sphere containing $\mathcal{B}$ has radius at least $\|\mathbf{p}-\mathbf{q}\|/2$. If $\mathbf{x} \in \mathcal{B}$, then

$$|x_i - (p_i + q_i)/2| \leq (q_i - p_i)/2$$

for every i. Consequently, $\|\mathbf{x} - (\mathbf{p}+\mathbf{q})/2\| \leq \|\mathbf{p}-\mathbf{q}\|/2$ and the sphere with center $\mathbf{c} = (\mathbf{p}+\mathbf{q})/2$ and radius $r = \|\mathbf{p}-\mathbf{q}\|/2$ contains $\mathcal{B}$. It follows that this is the smallest sphere containing $\mathcal{B}$ since any other sphere must have radius at least $\|\mathbf{p}-\mathbf{q}\|/2$.

Remark 3 Finding the smallest sphere containing $\mathcal{C}$ may not be easy. However, the center and radius of any sphere containing $\mathcal{C}$ yields an affine underestimate for $\|\mathbf{x}\|^2$ over $\mathcal{C}$. That is, if $\mathcal{S}$ is a sphere with $\mathcal{C} \subset \mathcal{S}$, then the best affine underestimate for $-\|\mathbf{x}\|^2$ over $\mathcal{S}$ is also an affine underestimate for $-\|\mathbf{x}\|^2$ over $\mathcal{C}$.

3.2 Lower bound based on semidefinite programming

A different DC decomposition of $f(\mathbf{x}) = (\mathbf{1}-\mathbf{x})^{\mathsf{T}}(\mathbf{A}+\mathbf{D})\mathbf{x}$ is the following:

$$f(\mathbf{x}) = (f(\mathbf{x}) + \mathbf{x}^{\mathsf{T}}\mathbf{\Lambda}\mathbf{x}) - \mathbf{x}^{\mathsf{T}}\mathbf{\Lambda}\mathbf{x},$$

where $\mathbf{\Lambda}$ is a diagonal matrix with i-th diagonal element $\lambda_i \geq 0$. We would like to make the second term $\mathbf{x}^{\mathsf{T}}\mathbf{\Lambda}\mathbf{x}$ as small as possible while keeping the first term $f(\mathbf{x}) + \mathbf{x}^{\mathsf{T}}\mathbf{\Lambda}\mathbf{x}$ convex. This suggests the following semidefinite programming problem

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} \lambda_i \\ \text{subject to} \quad & \mathbf{\Lambda} - (\mathbf{A}+\mathbf{D}) \succeq \mathbf{0}, \quad \mathbf{\Lambda} \succeq \mathbf{0}, \end{aligned} \tag{3.11}$$

where $\boldsymbol{\lambda}$ is the diagonal of $\mathbf{\Lambda}$. If the diagonal of $\mathbf{A}+\mathbf{D}$ is nonnegative, then the inequality $\mathbf{\Lambda} \succeq \mathbf{0}$ can be dropped since it is implied by the inequality $\mathbf{\Lambda} - (\mathbf{A}+\mathbf{D}) \succeq \mathbf{0}$.

As before, we seek the best linear underestimate of the concave function $-\mathbf{x}^{\mathsf{T}}\mathbf{\Lambda}\mathbf{x}$ over a compact, convex set $\mathcal{C}$. If any of the λ_i vanish, then reorder the components of $\mathbf{x}$ so that $\mathbf{x} = (\mathbf{y}, \mathbf{z})$ where $\mathbf{z}$ corresponds to the components of λ_i that vanish. Let $\mathbf{\Lambda}_+$ be the principal submatrix of $\mathbf{\Lambda}$ corresponding to the positive diagonal elements, and define the set

$$\mathcal{C}_+ = \{\mathbf{y} : (\mathbf{y}, \mathbf{z}) \in \mathcal{C} \text{ for some } \mathbf{z}\}.$$

The problem of finding the best linear underestimate for $-\mathbf{x}^{\mathsf{T}}\mathbf{\Lambda}\mathbf{x}$ over $\mathcal{C}$ is essentially equivalent to finding the best linear underestimate for $-\mathbf{y}^{\mathsf{T}}\mathbf{\Lambda}_+\mathbf{y}$ over the $\mathcal{C}_+$. Hence, there is no loss of generality in assuming that the diagonal of $\mathbf{\Lambda}$ is strictly positive. As a consequence of Theorem 3.1, we have

Corollary 3.2 *Suppose the diagonal of* $\mathbf{\Lambda}$ *is strictly positive and let* $\mathbf{c}$ *be the center and* r *the radius of the unique smallest sphere containing the set*

$$\mathbf{\Lambda}^{1/2}\mathcal{C} := \{\mathbf{\Lambda}^{1/2}\mathbf{x} : \mathbf{x} \in \mathcal{C}\}.$$

The best linear underestimate of $-\mathbf{x}^{\mathsf{T}}\mathbf{\Lambda}\mathbf{x}$ *over the compact, convex set* $\mathcal{C}$ *is*

$$\ell^*(\mathbf{x}) = -2\mathbf{c}^{\mathsf{T}}\mathbf{\Lambda}^{1/2}\mathbf{x} + \|\mathbf{c}\|^2 - r^2.$$

Furthermore,

$$\min_{\ell \in \mathcal{S}_2} \max_{\mathbf{x} \in \mathcal{C}} \; -\left(\mathbf{x}^{\mathsf{T}}\mathbf{\Lambda}\mathbf{x} + \ell^*(\mathbf{x})\right) = r^2,$$

where

$$\mathcal{S}_2 = \{\ell : \mathbb{R}^n \to \mathbb{R} \text{ such that } \ell \text{ is affine and } -\mathbf{x}^{\mathsf{T}}\mathbf{\Lambda}\mathbf{x} \geq \ell(\mathbf{x}) \text{ for all } \mathbf{x} \in \mathcal{C}\}.$$

Proof With the change of variables $\mathbf{y} = \mathbf{\Lambda}^{1/2}\mathbf{x}$, an affine function in $\mathbf{x}$ is transformed to an affine function in $\mathbf{y}$ and conversely, an affine function in $\mathbf{y}$ is transformed to an affine function in $\mathbf{x}$. Hence, the problem of finding the best affine underestimate for $-\mathbf{x}^{\mathsf{T}}\mathbf{\Lambda}\mathbf{x}$ over $\mathcal{C}$ is equivalent to the problem of finding the best affine underestimate for $-\|\mathbf{y}\|^2$ over $\mathbf{\Lambda}^{1/2}\mathcal{C}$. Apply Theorem 3.1 to the transformed problem in $\mathbf{y}$, and then transform back to $\mathbf{x}$. □

Remark 4 If $\mathcal{C}$ is the box $\{\mathbf{x} \in \mathbb{R}^n : \mathbf{0} \leq \mathbf{x} \leq \mathbf{1}\}$, then $\mathbf{\Lambda}^{1/2}\mathcal{C}$ is also a box to which we can apply the observation in Remark 2. In particular, we have

$$\mathbf{c} = \frac{1}{2}\mathbf{\Lambda}^{1/2}\mathbf{1} = \frac{1}{2}\boldsymbol{\lambda}^{1/2} \quad \text{and} \quad r = \|\mathbf{\Lambda}^{1/2}\mathbf{1}\|/2 = \|\boldsymbol{\lambda}^{1/2}\|/2. \tag{3.12}$$

Hence, $\|\mathbf{c}\|^2 - r^2 = 0$ and we have $\ell^*(\mathbf{x}) = -\boldsymbol{\lambda}^{\mathsf{T}}\mathbf{x}$.

Remark 5 Let us consider the set

$$\mathcal{C} = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{0} \leq \mathbf{x} \leq \mathbf{1}, \quad \mathbf{1}^{\mathsf{T}}\mathbf{x} = b\},$$

where $0 < b < n$. Determining the smallest sphere containing $\mathbf{\Lambda}^{1/2}\mathcal{C}$ may not be easy. However, as indicated in Remark 3, any sphere containing $\mathbf{\Lambda}^{1/2}\mathcal{C}$ yields an underestimate for $\mathbf{x}^{\mathsf{T}}\mathbf{\Lambda}\mathbf{x}$. Observe that

$$\mathbf{\Lambda}^{1/2}\mathcal{C} = \{\mathbf{y} \in \mathbb{R}^n : \mathbf{0} \leq \mathbf{y} \leq \boldsymbol{\lambda}^{1/2}, \quad \mathbf{y}^{\mathsf{T}}\boldsymbol{\lambda}^{-1/2} = b\}.$$

As observed in Remark 4, the center $\mathbf{c}$ and radius r of the smallest sphere $\mathcal{S}$ containing the set

$$\{\mathbf{y} \in \mathbb{R}^n : \mathbf{0} \leq \mathbf{y} \leq \boldsymbol{\lambda}^{1/2}\}$$

are given in (3.12). The intersection of this sphere with the hyperplane $\mathbf{y}^T\boldsymbol{\lambda}^{-1/2} = b$ is a lower dimensional sphere $\mathcal{S}'$ whose center $\mathbf{c}'$ is the projection of $\mathbf{c}$ onto the hyperplane. $\mathcal{S}'$ contains $\mathcal{C}$ since $\mathcal{C}$ is contained in both the original sphere $\mathcal{S}$ and the hyperplane. With a little algebra, we obtain

$$\mathbf{c}' = \frac{1}{2}\boldsymbol{\lambda}^{1/2} + \left(\frac{b - .5n}{\sum_{i=1}^n \lambda_i^{-1}}\right)\boldsymbol{\lambda}^{-1/2}.$$

By the Pythagorean Theorem, the radius r' of the lower dimensional sphere $\mathcal{S}'$ is

$$r' = \sqrt{.25\left(\sum_{i=1}^n \lambda_i\right) - \frac{(b - .5n)^2}{\sum_{i=1}^n \lambda_i^{-1}}}.$$

Hence, by Corollary 3.2, an underestimate of $-\mathbf{x}^T\boldsymbol{\Lambda}\mathbf{x}$ is given by

$$\ell(\mathbf{x}) = -\boldsymbol{\lambda}^T\mathbf{x} + \left(\frac{n - 2b}{\sum_{i=1}^n \lambda_i^{-1}}\right)\mathbf{1}^T\mathbf{x} + \|\mathbf{c}'\|^2 - (r')^2.$$

Since $\mathbf{1}^T\mathbf{x} = b$ when $\mathbf{x} \in \mathcal{C}$, it can be shown, after some algebra, that $\ell(\mathbf{x}) = -\boldsymbol{\lambda}^T\mathbf{x}$ (all the constants in the affine function cancel). Hence, the affine underestimate ℓ^* computed in Remark 4 for the unit box and the affine underestimate ℓ computed in this remark for the unit box intersect the hyperplane $\mathbf{1}^T\mathbf{x} = b$ are the same.

4 Branch and bound algorithm

Since the continuous quadratic program (2.1) has a binary solution, the branching process in the branch and bound algorithm is based on setting variables to 0 or 1 and reducing the problem dimension (we do not employ bisections of the feasible region as in [19]). We begin by constructing a linear ordering of the vertices of the graph according to an estimate for the difficulty in placing the vertex in the partition. For the numerical experiments, the order was based on the total weight of the edges connecting a vertex to the adjacent vertices. If two vertices v_1 and v_2 have weights w_1 and w_2 respectively, then v_1 precedes v_2 if $w_1 > w_2$.

Let $v_1, v_2, \ldots, v_n$ denote the ordered vertices. Level i in the branch and bound tree corresponds to setting the v_i-th component of $\mathbf{x}$ to the values 0 or 1. Each leaf at level i represents a specific selection of 0 and 1 values for the v_1 through v_i-th components of $\mathbf{x}$. Hence, a leaf at level i has a label of the form

$$\tau = (b_1, b_2, \ldots, b_i), \quad b_j = 0 \text{ or } 1 \text{ for } 1 \le j \le i. \tag{4.1}$$

Corresponding to this leaf, the value of the v_j-th component of $\mathbf{x}$ is b_j for $1 \le j \le i$.

Let $\mathcal{T}_k$ denote the branch and bound tree at iteration k and let $\mathcal{E}(\mathcal{T}_k)$ denote the leaves in the tree. Suppose $\tau \in \mathcal{E}(\mathcal{T}_k)$ lies at level i in $\mathcal{T}_k$ as in (4.1). Let $\mathbf{x}_\tau$ denote the

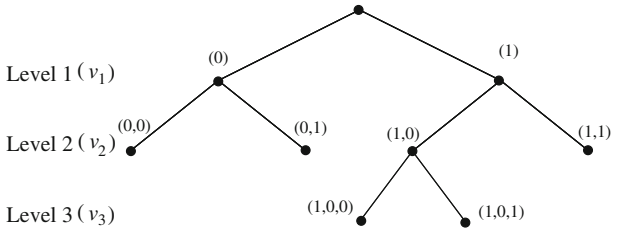


Fig. 2 Branch and bound tree

vector gotten by removing components v_j, $1 \le j \le i$, from $\mathbf{x}$. The v_j-th component of $\mathbf{x}$ has the pre-assigned binary value b_j for $1 \le j \le i$. After taking into account these assigned binary values, the quadratic problem reduces to a lower dimensional problem in the variable $\mathbf{x}_\tau$ of the form

$$\begin{aligned} &\text{minimize} \quad f_\tau(\mathbf{x}_\tau) \\ &\text{subject to} \quad \mathbf{0} \le \mathbf{x}_\tau \le \mathbf{1}, \quad l_\tau \le \mathbf{1}^\mathsf{T}\mathbf{x}_\tau \le u_\tau, \end{aligned}$$

where

$$u_\tau = u - \sum_{j=1}^{i} b_j \quad \text{and} \quad l_\tau = l - \sum_{j=1}^{i} b_j.$$

Using the techniques developed in Sect. 3, we replace f_τ by a convex lower bound denoted f_τ^L and we consider the convex problem

$$\begin{aligned} &\text{minimize} \quad f_\tau^L(\mathbf{x}_\tau) \\ &\text{subject to} \quad \mathbf{0} \le \mathbf{x}_\tau \le \mathbf{1}, \quad l_\tau \le \mathbf{1}^\mathsf{T}\mathbf{x}_\tau \le u_\tau. \end{aligned} \tag{4.2}$$

Let $M(\tau)$ denote the optimal objective function value for (4.2). At iteration k, the leaf $\tau \in \mathcal{E}(\mathcal{T}_k)$ for which $M(\tau)$ is smallest is used to branch to the next level. If τ has the form (4.1), then the branching processes generates the two new leaves

$$(b_1, b_2, \ldots, b_i, 0) \quad \text{and} \quad (b_1, b_2, \ldots, b_i, 1). \tag{4.3}$$

An illustration involving a 3-level branch and bound tree appears in Fig. 2.

During the branch and bound process, we must also compute an upper bound for the minimal objective function value in (2.1). This upper bound is obtained using a heuristic technique based on the gradient projection algorithm and sphere approximations to the feasible set. These heuristics for generating an upper bound will be described in a separate paper. As pointed out earlier, many heuristic techniques are available (for example, Metis [29–31], Chaco [21], and Party [37]). An advantage of our quadratic programming based heuristic is that we start at the solution to the lower bounding problem, a solution which typically has fractional entries and which is a

feasible starting point for (2.1). Consequently, the upper bound is no larger than the objective function value associated with the optimal point in the lower-bound problem.

Convex quadratic branch and bound (CQB)

1. Initialize $\mathcal{T}_0 = \emptyset$ and $k = 0$. Evaluate both a lower bound for the solution to (2.1) and an upper denoted U_0.
2. Choose $\tau_k \in \mathcal{E}(\mathcal{T}_k)$ such that $M(\tau_k) = \min\{M(\tau) : \tau \in \mathcal{E}(\mathcal{T}_k)\}$. If $M(\tau_k) = U_k$, then stop, an optimal solution of (2.1) has been found.
3. Assuming that τ_k has the form (4.1), let $\mathcal{T}_{k+1}$ be the tree obtained by branching at τ_k and adding two new leaves as in (4.3); also see Fig. 2. Evaluate lower bounds for the quadratic programming problems (4.2) associated with the two new leaves, and evaluate an improved upper bound, denoted U_{k+1}, by using solutions to the lower bound problems as starting guesses in a descent method applied to (2.1).
4. Replace k by $k + 1$ and return to step 2.

Convergence is assured since there are a finite number of binary values for the components of $\mathbf{x}$. In the worst case, the branch and bound algorithm will build all $2^{n+1} - 1$ nodes of the tree.

5 Necessary and sufficient optimality conditions

We use the gradient projection algorithm to obtain an upper bound for a solution to (2.1). Since the gradient projection algorithm can terminate at a stationary point, we need to be able to distinguish between a stationary point and a local minimizer, and at a stationary point which is not a local minimizer, we need a fast way to compute a descent direction.

We begin by stating the first-order optimality conditions. Given a scalar λ, define the vector

$$\boldsymbol{\mu}(\mathbf{x}, \lambda) = (\mathbf{A} + \mathbf{D})\mathbf{1} - 2(\mathbf{A} + \mathbf{D})\mathbf{x} + \lambda\mathbf{1},$$

and the set-valued maps $\mathcal{N} : \mathbb{R} \to 2^{\mathbb{R}}$ and $\mathcal{M} : \mathbb{R} \to 2^{\mathbb{R}}$

$$\mathcal{N}(\nu) = \begin{cases} \mathbb{R} & \text{if } \nu = 0 \\ \{1\} & \text{if } \nu < 0 \\ \{0\} & \text{if } \nu > 0 \end{cases}, \quad \mathcal{M}(\nu) = \begin{cases} \mathbb{R} & \text{if } \nu = 0 \\ \{u\} & \text{if } \nu > 0 \\ \{l\} & \text{if } \nu < 0 \end{cases}.$$

For any vector $\boldsymbol{\mu}$, $\mathcal{N}(\boldsymbol{\mu})$ is a vector of sets whose i-component is the set $\mathcal{N}(\mu_i)$. The first-order optimality (Karush-Kuhn-Tucker) conditions associated with a local minimizer $\mathbf{x}$ of (2.1) can be written in the following way: For some scalar λ, we have

$$\mathbf{0} \le \mathbf{x} \le \mathbf{1}, \quad \mathbf{x} \in \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}, \lambda)), \quad l \le \mathbf{1}^{\mathsf{T}}\mathbf{x} \le u, \quad \text{and} \quad \mathbf{1}^{\mathsf{T}}\mathbf{x} \in \mathcal{M}(\lambda). \tag{5.1}$$

The first and third conditions in (5.1) are the constraints in (2.1), while the remaining two conditions correspond to complementary slackness and stationarity of the Lagrangian.

In [17] we give a necessary and sufficient optimality conditions for (2.1), which we now review. Given any $\mathbf{x}$ that is feasible in (2.1), let us define the sets

$$\mathcal{U}(\mathbf{x}) = \{i : x_i = 1\}, \quad \mathcal{L}(\mathbf{x}) = \{i : x_i = 0\}, \quad \text{and} \quad \mathcal{F}(\mathbf{x}) = \{i : 0 < x_i < 1\}.$$

We also introduce subsets $\mathcal{U}_0$ and $\mathcal{L}_0$ defined by

$$\mathcal{U}_0(\mathbf{x}, \lambda) = \{i \in \mathcal{U}(\mathbf{x}) : \mu_i(\mathbf{x}, \lambda) = 0\} \quad \text{and} \quad \mathcal{L}_0(\mathbf{x}, \lambda) = \{i \in \mathcal{L}(\mathbf{x}) : \mu_i(\mathbf{x}, \lambda) = 0\}.$$

Theorem 5.1 *Suppose that $l = u$ and $\mathbf{D}$ is chosen so that*

$$d_{ii} + d_{jj} \geq 2a_{ij} \tag{5.2}$$

for all i and j. A necessary and sufficient condition for $\mathbf{x}$ to be a local minimizer in (2.1) is that the following all hold:

(P1) *For some λ, the first-order conditions (5.1) are satisfied at $\mathbf{x}$.*
(P2) *For each i and $j \in \mathcal{F}(\mathbf{x})$, we have $d_{ii} + d_{jj} = 2a_{ij}$.*
(P3) *Consider the three sets $\mathcal{U}_0(\mathbf{x}, \lambda)$, $\mathcal{L}_0(\mathbf{x}, \lambda)$, and $\mathcal{F}(\mathbf{x})$. For each i and j in two different sets, we have $d_{ii} + d_{jj} = 2a_{ij}$.*

In treating the situation $l < u$, an additional condition concerning the dual multipliers λ and $\boldsymbol{\mu}$ in the first-order optimality conditions (5.1) enters into the statement of the result:

(P4) *If $\lambda = \mu_i(\mathbf{x}, 0) = 0$ for some i, then $d_{ii} = 0$ in any of the following three cases:*
  (a) $l < \mathbf{1}^\mathsf{T}\mathbf{x} < u$.
  (b) $x_i > 0$ and $\mathbf{1}^\mathsf{T}\mathbf{x} = u$.
  (c) $x_i < 1$ and $\mathbf{1}^\mathsf{T}\mathbf{x} = l$.

Corollary 5.2 *Suppose that $l < u$ and $\mathbf{D}$ is chosen so that*

$$d_{ii} + d_{jj} \geq 2a_{ij} \quad \text{and} \quad d_{ii} \geq 0 \tag{5.3}$$

for all i and j. A necessary and sufficient condition for $\mathbf{x}$ to be a local minimizer in (2.1) is that (P1)–(P4) *all hold.*

Based on Theorem 5.1 and Corollary 5.2, we can easily check whether a given stationary point is a local minimizer. This is in contrast to the general quadratic programming problem for which deciding whether a given point is a local minimizer is NP-hard (see [34,35]). We now observe that when $\mathbf{x}$ is a stationary point and when any of the conditions (P2)–(P4) are violated, then a descent direction is readily available.

Proposition 5.3 *Suppose that $\mathbf{x}$ is a stationary point for (2.1) and (5.3) holds. If either* (P2) *or* (P3) *is violated, then $\mathbf{d} = \mathbf{e}_i - \mathbf{e}_j$, with an appropriate choice of sign, is a descent direction. If $l < u, \lambda = 0 = \mu_i(\mathbf{x}, 0)$, and $d_{ii} > 0$, then $\mathbf{d} = \mathbf{e}_i$, with an appropriate choice of sign, is a descent direction in any of the cases* (a)–(c) *of* (P4).

Proof The Lagrangian L associated with (2.1) has the form

$$L(\mathbf{x}) = f(\mathbf{x}) + \lambda(\mathbf{1}^{\mathsf{T}}\mathbf{x} - b) - \sum_{i\in\mathcal{L}} \mu_i x_i - \sum_{i\in\mathcal{U}} \mu_i (x_i - 1), \tag{5.4}$$

where $b = u$ if $\lambda > 0, b = l$ if $\lambda < 0$, and $\boldsymbol{\mu}$ stands for $\boldsymbol{\mu}(\mathbf{x}, \lambda)$. The sets $\mathcal{L}$ and $\mathcal{U}$ denote $\mathcal{L}(\mathbf{x})$ and $\mathcal{U}(\mathbf{x})$ respectively. By the first-order optimality conditions (5.1), we have $L(\mathbf{x}) = f(\mathbf{x})$ and $\nabla L(\mathbf{x}) = \mathbf{0}$. Expanding the Lagrangian around $\mathbf{x}$ gives

$$L(\mathbf{x}+\mathbf{y}) = L(\mathbf{x}) + \nabla L(\mathbf{x})\mathbf{y} + \frac{1}{2}\mathbf{y}^{\mathsf{T}}\nabla^2 L(\mathbf{x})\mathbf{y} = f(\mathbf{x}) - \mathbf{y}^{\mathsf{T}}(\mathbf{A}+\mathbf{D})\mathbf{y}.$$

We substitute for L using (5.4) to obtain

$$\begin{aligned} f(\mathbf{x}+\mathbf{y}) &= L(\mathbf{x}+\mathbf{y}) - \lambda(\mathbf{1}^{\mathsf{T}}(\mathbf{x}+\mathbf{y}) - b) + \sum_{i\in\mathcal{L}} \mu_i (x_i + y_i) + \sum_{i\in\mathcal{U}} \mu_i (x_i + y_i - 1) \\ &= f(\mathbf{x}) - \lambda\mathbf{1}^{\mathsf{T}}\mathbf{y} - \mathbf{y}^{\mathsf{T}}(\mathbf{A}+\mathbf{D})\mathbf{y} + \sum_{i\in\mathcal{L}} \mu_i y_i + \sum_{i\in\mathcal{U}} \mu_i y_i. \end{aligned} \tag{5.5}$$

If (P2) is violated, then there are indices i and $j \in \mathcal{F}(\mathbf{x})$ such that $d_{ii} + d_{jj} > 2a_{ij}$. We insert $\mathbf{y} = \alpha(\mathbf{e}_i - \mathbf{e}_j)$ in (5.5) to obtain

$$f(\mathbf{x} + \alpha(\mathbf{e}_i - \mathbf{e}_j)) = f(\mathbf{x}) + \alpha^2(2a_{ij} - d_{ii} - d_{jj}). \tag{5.6}$$

Since the coefficient of α^2 is negative, $\mathbf{d} = \mathbf{e}_i - \mathbf{e}_j$ is a descent direction for the objective function. Since $0 < x_i < 1$ and $0 < x_j < 1$, feasibility is preserved for α sufficiently small. In a similar manner, if (P3) is violated by indices i and j, then (5.6) again holds and $\mathbf{d} = \pm(\mathbf{e}_i - \mathbf{e}_j)$ is again a descent direction where the sign is chosen appropriately to preserve feasibility. For example, if $i \in \mathcal{L}_0(\mathbf{x})$ and $j \in \mathcal{U}_0(\mathbf{x})$, then $x_i = 0$ and $x_j = 1$. Consequently, $\mathbf{x} + \alpha(\mathbf{e}_i - \mathbf{e}_j)$ is feasible if $\alpha > 0$ is sufficiently small.

Finally, suppose that $l < u, \lambda = 0 = \mu_i(\mathbf{x}, 0)$, and $d_{ii} > 0$. Substituting $\mathbf{y} = \alpha\mathbf{e}_i$ in (5.5) yields

$$f(\mathbf{x} + \alpha\mathbf{e}_i) = f(\mathbf{x}) - \alpha^2 d_{ii}.$$

Since the coefficient d_{ii} of α^2 is positive, $\mathbf{d} = \pm\mathbf{e}_i$ is a descent direction. Moreover, in any of the cases (a)–(c) of (P4), $\mathbf{x} + \alpha\mathbf{d}$ is feasible for some $\alpha > 0$ with an appropriate choice of the sign of $\mathbf{d}$. □

We now give a necessary and sufficient condition for a local minimizer to be strict. When a local minimizer is not strict, it may be possible to move to a neighboring point which has the same objective function value but which is not a local minimizer.

Corollary 5.4 *If* $\mathbf{x}$ *is a local minimizer for* (2.1) *and* (5.3) *holds, then* $\mathbf{x}$ *is a strict local minimizer if and only if the following conditions hold:*

(C1) $\mathcal{F}(\mathbf{x})$ *is empty.*
(C2) $\nabla f(\mathbf{x})_i > \nabla f(\mathbf{x})_j$ *for every* $i \in \mathcal{L}(\mathbf{x})$ *and* $j \in \mathcal{U}(\mathbf{x})$.
(C3) *If* $l < u$, *the first-order optimality conditions* (5.1) *hold for* $\lambda = 0$, *and* $\mathcal{Z} := \{i : \nabla f(\mathbf{x})_i = 0\} \neq \emptyset$, *then either*
  (a) $\mathbf{1}^{\mathsf{T}}\mathbf{x} = u$ *and* $x_i = 0$ *for all* $i \in \mathcal{Z}$ *or*
  (b) $\mathbf{1}^{\mathsf{T}}\mathbf{x} = l$ *and* $x_i = 1$ *for all* $i \in \mathcal{Z}$.

Proof Throughout the proof, we let $\boldsymbol{\mu}$, $\mathcal{F}$, $\mathcal{L}$ and $\mathcal{U}$ denote $\boldsymbol{\mu}(\mathbf{x}, \lambda)$, $\mathcal{F}(\mathbf{x})$, $\mathcal{L}(\mathbf{x})$, and $\mathcal{U}(\mathbf{x})$ respectively, where $\mathbf{x}$ is a local minimizer for (2.1) and the pair $(\mathbf{x}, \lambda)$ satisfies the first-order optimality conditions (5.1). To begin, suppose that $\mathbf{x}$ is a strict local minimizer of (2.1). That is, $f(\mathbf{y}) > f(\mathbf{x})$ when $\mathbf{y}$ is a feasible point near $\mathbf{x}$. If $\mathcal{F}$ has at least two elements, then by (P2) of Theorem 5.1, $d_{ii} + d_{jj} = 2a_{ij}$ for each i and $j \in \mathcal{F}$. Since the first-order optimality conditions (5.1) hold at $\mathbf{x}$, it follows from (5.6) that

$$f(\mathbf{x} + \alpha(\mathbf{e}_i - \mathbf{e}_j)) = f(\mathbf{x}) \tag{5.7}$$

for all α. Since this violates the assumption that $\mathbf{x}$ is a strict local minimizer, we conclude that $|\mathcal{F}| \leq 1$. If $\mathbf{1}^{\mathsf{T}}\mathbf{x} = u$ or $\mathbf{1}^{\mathsf{T}}\mathbf{x} = l$, then since u and l are integers, it is not possible for $\mathbf{x}$ to have just one fractional component. Consequently, $\mathcal{F}$ is empty. If $l < \mathbf{1}^{\mathsf{T}}\mathbf{x} < u$, then by complementary slackness, $\lambda = 0$. Suppose that $|\mathcal{F}| = 1$ and $i \in \mathcal{F}$. By (P4) of Corollary 5.2, $d_{ii} = 0$. Again, by (5.5) it follows that

$$f(\mathbf{x} + \alpha\mathbf{e}_i) = f(\mathbf{x})$$

for all α. This violates the assumption that $\mathbf{x}$ is a strict local minimizer of (2.1). Hence, $\mathcal{F}$ is empty.

By the first-order conditions (5.1), there exists λ such that

$$\mu_i(\mathbf{x}, \lambda) \geq 0 \geq \mu_j(\mathbf{x}, \lambda) \tag{5.8}$$

for all $i \in \mathcal{L}$ and $j \in \mathcal{U}$. If this inequality becomes an equality for some $i \in \mathcal{L}$ and $j \in \mathcal{U}$, then $\mu_i = 0 = \mu_j$, and by (P3) of Corollary 5.2, we have $d_{ii} + d_{jj} = 2a_{ij}$. Again, (5.7) violates the assumption that $\mathbf{x}$ is a strict local minimizer. Hence, one of the inequalities in (5.8) is strict. The λ on each side of (5.8) is cancelled to obtain (C2).

Suppose that $l < u$, $\lambda = 0$, and $\mathcal{Z} := \{i : \nabla f(\mathbf{x})_i = 0\} \neq \emptyset$. When $\lambda = 0$, we have $\boldsymbol{\mu}(\mathbf{x}, 0) = \nabla f(\mathbf{x})$. Hence, $\mathcal{Z} = \{i : \mu_i(\mathbf{x}, 0) = 0\} \neq \emptyset$. It follows from (P4) that in any of the cases (a)–(c), we have $d_{ii} = 0$. In particular, if $l < \mathbf{1}^{\mathsf{T}}\mathbf{x} < u$, then by (5.5), we have $f(\mathbf{x} + \alpha\mathbf{e}_i) = f(\mathbf{x})$ for all α. Again, this violates the assumption that $\mathbf{x}$ is a strict local minimum. Similarly, if for some $i \in \mathcal{Z}$, either $x_i > 0$ and $\mathbf{1}^{\mathsf{T}}\mathbf{x} = u$ or $x_i < 1$ and $\mathbf{1}^{\mathsf{T}}\mathbf{x} = l$, the identity $f(\mathbf{x} + \alpha\mathbf{e}_i) = f(\mathbf{x})$ implies that we violate the strict local optimality of $\mathbf{x}$. This establishes (C3).

Conversely, suppose that $\mathbf{x}$ is a local minimizer and (C1)–(C3) hold. We will show that

$$\nabla f(\mathbf{x})\mathbf{y} > 0 \text{ whenever } \mathbf{y} \neq \mathbf{0} \text{ and } \mathbf{x} + \mathbf{y} \text{ feasible in (2.1).} \tag{5.9}$$

As a result, by the mean value theorem, $f(\mathbf{x}+\mathbf{y}) > f(\mathbf{x})$ when $\mathbf{y}$ is sufficiently small. Hence, $\mathbf{x}$ is a strict local minimizer.

When $\mathbf{x}+\mathbf{y}$ is feasible in (2.1), we have

$$y_i \geq 0 \ \text{ for all } \ i \in \mathcal{L} \quad \text{and} \quad y_i \leq 0 \ \text{ for all } \ i \in \mathcal{U}. \tag{5.10}$$

By the first-order optimality condition (5.1), $\mu_i \geq 0$ for all $i \in \mathcal{L}$ and $\mu_i \leq 0$ for all $i \in \mathcal{U}$. Hence, we have

$$(\nabla f(\mathbf{x}) + \lambda \mathbf{1}^{\mathsf{T}})\mathbf{y} = \boldsymbol{\mu}^{\mathsf{T}}\mathbf{y} = \sum_{i\in\mathcal{L}} \mu_i y_i + \sum_{i\in\mathcal{U}} \mu_i y_i \geq 0. \tag{5.11}$$

We now consider three cases.

First, suppose that $\mathbf{1}^{\mathsf{T}}\mathbf{y} = 0$ and $\mathbf{y} \neq \mathbf{0}$. By (C1) $\mathcal{F}$ is empty and hence, by (5.10), $y_i > 0$ for some $i \in \mathcal{L}$ and $y_j < 0$ for some $j \in \mathcal{U}$. After adding λ to each side in the inequality in (C2), it follows that either

$$\min_{i\in\mathcal{L}} \mu_i \geq 0 > \max_{j\in\mathcal{U}} \mu_j \tag{5.12}$$

or

$$\min_{i\in\mathcal{L}} \mu_i > 0 \geq \max_{j\in\mathcal{U}} \mu_j. \tag{5.13}$$

Combining (5.11), (5.12), and (5.13) gives $\nabla f(\mathbf{x})\mathbf{y} \geq \mu_i y_i - \mu_j y_j > 0$ since either $\mu_i > 0$ or $\mu_j < 0$, and $y_i > 0 > y_j$.

Second, suppose that $\mathbf{1}^{\mathsf{T}}\mathbf{y} \neq 0$ and $\lambda \neq 0$. To be specific, suppose that $\lambda > 0$. By complementary slackness, $\mathbf{1}^{\mathsf{T}}\mathbf{x} = u$. Since $\mathbf{x}+\mathbf{y}$ is feasible in (2.1) and $\mathbf{1}^{\mathsf{T}}\mathbf{y} \neq 0$, we must have $\mathbf{1}^{\mathsf{T}}\mathbf{y} < 0$. Hence, by (5.11), $\nabla f(\mathbf{x})\mathbf{y} > 0$. The case $\lambda < 0$ is similar.

Finally, consider the case $\mathbf{1}^{\mathsf{T}}\mathbf{y} \neq 0$ and $\lambda = 0$. In this case, we must have $l < u$. If the set $\mathcal{Z}$ in (C3) is empty, then $\nabla f(\mathbf{x})_i = \mu_i \neq 0$ for all i, and by (5.11), $\nabla f(\mathbf{x})\mathbf{y} > 0$. If $\mathcal{Z} \neq \emptyset$, then by (C3), either $\mathbf{1}^{\mathsf{T}}\mathbf{x} = u$ and $x_i = 0$ for all $i \in \mathcal{Z}$ or $\mathbf{1}^{\mathsf{T}}\mathbf{x} = l$ and $x_i = 1$ for all $i \in \mathcal{Z}$. To be specific, suppose that $\mathbf{1}^{\mathsf{T}}\mathbf{x} = u$ and $x_i = 0$ for all $i \in \mathcal{Z}$. Again, since $\mathbf{x}+\mathbf{y}$ is feasible in (2.1) and $\mathbf{1}^{\mathsf{T}}\mathbf{y} \neq 0$, we have $\mathbf{1}^{\mathsf{T}}\mathbf{y} < 0$. If $\mathcal{U} = \emptyset$, then $\mathbf{x} = \mathbf{0}$ since $\mathcal{F} = \emptyset$. Since $\mathbf{1}^{\mathsf{T}}\mathbf{y} < 0$, we contradict the feasibility of $\mathbf{x}+\mathbf{y}$. Hence, $\mathcal{U} \neq \emptyset$. Since $\mathbf{1}^{\mathsf{T}}\mathbf{y} < 0$, there exists $j \in \mathcal{U}$ such that $y_j < 0$. Since $\mathcal{Z} \subset \mathcal{L}$, it follows from (5.12) that $\mu_j < 0$. By (5.11) $\nabla f(\mathbf{x})\mathbf{y} \geq \mu_j y_j > 0$. The case $\mathbf{1}^{\mathsf{T}}\mathbf{x} = l$ and $x_i = 1$ for all $i \in \mathcal{Z}$ is similar. This completes the proof of (5.9), and the corollary has been established. □

6 Numerical results

We investigate the performance of the branch and bound algorithm based on the lower bounds in Sect. 3 using a series of test problems. The codes were written in C and the experiments were conducted on an Intel Xeon Quad-Core X5355 2.66 GHz computer

using the Linux operating system. Only one of the 4 processors was used in the experiments. To evaluate the lower bound, we solve (4.2) by the gradient projection method with an exact linesearch and Barzilai-Borwein steplength [3]. The stopping criterion in our experiments was

$$\|P(\mathbf{x}_k - \mathbf{g}_k) - \mathbf{x}_k\| \le 10^{-4},$$

where P denotes the projection onto the feasible set and $\mathbf{g}_k$ is the gradient of the objective function at $\mathbf{x}_k$. The solution of the semidefinite programming problem (3.11) was obtained using Version 6.0.1 of the CSDP code [4] available at

https://projects.coin-or.org/Csdp/

We compare the performance of our algorithm with results reported by Karisch, Rendl, and Clausen in [28], by Sensen in [39], and by Armbruster in [1]. We also compared the performance of our algorithm to the max-cut algorithm of Rendl, Rinaldi, and Wiegele [38] as implemented in the code BiqMac. Since some earlier results were obtained on different computers, we obtained estimates for the corresponding running time on our computer using the LINPACK benchmarks [10]. Since our computer is roughly 30 times faster than the HP 9000/735 used in [28], it is roughly 7 times faster than the Sun UltrSPARC-II 400 Mhz machine used in [39], and it is roughly 1.3 times faster than the HP Compaq DC7100 3.2 Ghz Pentium IV used in [1], the earlier CPU times were divided by 30, 7 and 1.3 respectively to obtain the estimated running time on our computer. Note that the same interior-point algorithm that we use, which is the main routine in the CSDP code, was used to solve the semidefinite relaxation in [28]. Since the BiqMac code [38] of Rendl et al. was run on our machine, there was no need to scale the running times of this code.

Various data sets were used for the $\mathbf{A}$ matrices in the numerical experiments. Most of the test problems came from the library of Brunetta et al. [5]. Some of the test matrices were from the UF Sparse Matrix Library maintained by Timothy Davis:

http://www.cise.ufl.edu/research/sparse/matrices/

Since this second set of matrices is not directly connected with graph partitioning, we create an $\mathbf{A}$ for graph partitioning as follows: If the matrix $\mathbf{S}$ from the library was symmetric, then $\mathbf{A}$ was the adjacency matrix defined as follows: the diagonal of $\mathbf{A}$ is zero, $a_{ij} = 1$ if $s_{ij} \neq 0$, and $a_{ij} = 0$ otherwise. If $\mathbf{S}$ was not symmetric, then $\mathbf{A}$ was the adjacency matrix of $\mathbf{S}^T\mathbf{S}$. Finally, finite element mesh [8], KKT systems [20] and Johnson [25] graphs are available at

http://www.tu-chemnitz.de/mathematik/discrete/armbruster/diss

The test problems were based on the graph bisection problem where $l = u = n/2$ except for instances: KKT.lowt01, KKT.putt01, and KKT.capt09 in Table 9. For these three instances, we select (see page 149 of [1])

$$l = \lceil \frac{n-d}{2} \rceil, \quad u = \lfloor \frac{n+d}{2} \rfloor \text{ where } d = 2\lceil \frac{1.05n}{2} \rceil - n. \tag{6.1}$$

Table 1 Comparison of two lower bounds

Graph	n	LB_1	LB_2	Opt
Tina_Discal	11	0.31	**0.86**	12
jg1009	9	1.55	**1.72**	16
jg1011	11	**1.48**	0.94	24
Stranke94	10	1.76	**1.77**	24
Hamrle1	32	−1.93	**1.12**	17
4×5t	20	−21.71	**5.43**	28
8×5t	40	−16.16	**2.91**	33
t050	30	0.90	**18.54**	397
2×17 m	34	**1.33**	1.27	316
s090	60	−9.84	**13.10**	238

6.1 Lower bound comparison

Our numerical study begins with a comparison of the lower bound of Sect. 3.1 based on the minimum eigenvalue of $\mathbf{A}+\mathbf{D}$ and the best affine underestimate, and the lower bound of Sect. 3.2 based on semidefinite programming. We label these two lower bounds LB_1 and LB_2 respectively. In Table 1, the first 5 graphs correspond to matrices from the UF Sparse Matrix Library, while the next 5 graphs were from the test set of Brunetta, Conforti, and Rinaldi. The column labeled "Opt" is the minimum cut and while n is the problem dimension. The numerical results indicate that the lower bound LB_2 based on semidefinite programming is generally better (larger) than LB_1. In Table 1 the best lower bound is highlighted in bold. Based on these results, we use the semidefinite programming-based lower bound in the numerical experiments which follow.

6.2 Algorithm performance

Unless stated otherwise, the remaining test problems came from the library of Brunetta et al. [5]. Table 6 gives results for matrices associated with the finite element method [42]. In this section, the six methods being compared are labeled CQB (our convex quadratic branch and bound algorithm), BiqMac (algorithm of Rendl, Rinaldi, and Wiegele [38]), KRC (algorithm of Karisch, Rendl, and Clausen [28]), SEN (algorithm of Sensen [39]), and ARM-LP (linear programming relaxation) and ARM-SDP (semidefinite programming relaxation) (algorithms of Armbruster [1]). In particular, ARM-LP is based on the LP relaxation with separators (so-called `lp_all_small` in [1]), whereas ARM-SDP is related to the SDP relaxation along with the early branching criteria, odd cycle inequalities, the shortest-path and problem shrinking techniques (denoted by `oc_sp_s` in [1]). Note that some of numerical results from [1] were also presented in [2].

BiqMac is an algorithm designed to solve max-cut problems [38]. The minimum equipartitioning problems with n even can be formulated as

Table 2 Toroidal grid: a weighted $h \times k$ grid with hk vertices and $2hk$ edges that received integer weights uniformly drawn from [1,10]

			CQB		BiqMac		KRC	
Graph	n	(%)	#nodes	Time	#nodes	Time	#nodes	Time
8×5t	40	10	157	**0.12**	1	0.16	1	0.20
21×2t	42	10	21	**0.02**	1	0.22	1	0.17
23×2t	46	9	80	**0.16**	9	10.95	33	4.16
4×12t	48	9	76	**0.20**	1	0.29	3	0.56
5×10t	50	8	152	0.22	1	0.38	1	**0.20**
6×10t	60	7	1226	**0.43**	3	6.03	43	11.66
7×10t	70	6	1143	**0.72**	1	0.81	47	19.06
10×8t	80	5	584	**0.81**	3	4.76	45	31.46

$$\min \frac{1}{2} \sum_{i<j} a_{ij}(1 - x_i x_j) \quad \text{s.t.} \quad \sum_{i=1}^{n} x_i = 0, \mathbf{x} \in \{-1, 1\}^n.$$

By using an exact quadratic penalty function approach [40], we can recast the problem as an unconstrained integer programming problem

$$\min \frac{1}{2} \sum_{i<j} a_{ij}(1 - x_i x_j) + \tau \| \sum_{i=1}^{n} x_i \|^2 \quad \text{s.t. } \mathbf{x} \in \{-1, 1\}^n,$$

for any $\tau \geq \sum_{i<j} |a_{ij}|$, which is, in turn, equivalent to the following max-cut problem

$$\max \frac{1}{2} \sum_{i<j} (4\tau - a_{ij})(1 - x_i x_j) \quad \text{s.t. } \mathbf{x} \in \{-1, 1\}^n.$$

We chose $\tau = \sum_{i<j} |a_{ij}|$ in the numerical simulations for BiqMac. If n is odd, then we added a dummy node without any connections to the remaining nodes.

In the numerical results that follow, "n" is the problem dimension, "%" is the percent of the elements in the matrix that are nonzero, and "# nodes" is the number of nodes in the branch and bound tree. The CPU time is given in seconds. The best time is highlighted in bold.

Table 2 gives results for toroidal grid graphs. These graphs are connected with an $h \times k$ grid, the number of vertices in the graph is $n = hk$ and there are $2hk$ edges whose weights are chosen from a uniform distribution on the interval [1, 10]. We now compare between CQB, BiqMac, and KRC. We see in Table 2 that CQB was fastest in 7 of the 8 toroidal grid cases.

Results for planar grid graph are given in Table 3. These graphs are associated with an $h \times k$ grid. There are hk vertices and $2hk - h - k$ edges whose weights are integers uniformly drawn from [1,10]. For this relatively sparse test set, CQB was the fastest in 6 out of 8 problems.

Table 3 Planar grid

			CQB		BiqMac		KRC	
Graph	n	(%)	#nodes	Time	#nodes	Time	#nodes	Time
2×16 g	32	9	25	**0.06**	1	0.07	1	0.13
18×2 g	36	8	19	0.07	1	**0.05**	1	0.06
2×19 g	38	8	52	**0.11**	7	3.69	49	1.83
5×8 g	40	9	29	0.10	1	0.07	1	**0.06**
3×14 g	42	8	33	**0.13**	1	0.71	5	0.60
5×10 g	50	7	179	**0.28**	1	0.32	1	0.30
6×10 g	60	6	268	**0.33**	3	4.55	57	10.63
7×10 g	70	5	209	**0.52**	3	21.64	61	18.56

Table 4 Mixed grid graphs

			CQB		BiqMac		KRC	
Graph	n	(%)	#nodes	Time	#nodes	Time	#nodes	Time
2×17 m	34	100	42070	1.54	3	1.15	21	**0.96**
10×4 m	40	100	51635	3.80	1	**0.05**	2	0.06
5×10 m	50	100	3585782	283.17	1	0.14	1	**0.06**
4×13 m	52	100	10403116	725.36	1	**0.66**	5	1.13
13×4 m	52	100	16428323	1082.58	1	**0.63**	5	1.13
9×6 m	54	100	43513492	2967.04	1	0.53	1	**0.40**
10×6 m	60	100	36151948	2602.88	1	0.30	1	**0.26**
10×7 m	70	100	110759766	5044.75	1	**0.39**	1	0.46

Table 4 gives results for mixed grid graphs. These are complete graphs associated with an planar $h \times k$ planar grid; the edges in the planar grid received integer weights uniformly drawn from [1,100], while all the other edges needed to complete the graph received integer weights uniformly drawn from [1,10]. For these graphs, BiqMac and KRC were much faster than CQB. Notice that the graphs in this test set are completely dense. One trend that is seen in these numerical experiments is that as the graph density increases, the performance of CQB relative to the other methods degrades.

Table 5 gives results for randomly generated graphs. For these graphs, the density is first fixed and then the edges are assigned integer weights uniformly drawn from [1,10]. For this test set, CQB is fastest in 8 of 17 cases. Again, observe that the relative performance of CQB degrades as the density increases, mainly due to the large number of nodes in the branch and bound tree.

As can be seen in the mesh instances of Table 6, CQB was fastest in 6 out of the 10 problems even though the number of nodes in the branch and bound tree was much larger. Thus BiqMac, KRC and SEN provided much tighter relaxations, however, the time to solve their relaxed problems was much larger than the time to optimize our convex quadratics.

Table 5 Randomly generated graphs

			CQB		BiqMac		KRC	
Graph	n	(%)	#nodes	Time	#nodes	Time	#nodes	Time
q090	40	10	88	**0.11**	1	0.29	1	0.13
q080	40	20	1076	**0.20**	5	7.17	31	2.30
q030	40	70	781245	37.41	7	5.61	23	**2.06**
q020	40	80	1522011	66.02	3	3.84	7	**0.83**
q010	40	90	4096845	202.07	5	3.59	13	**1.36**
q000	40	100	7975274	373.24	1	0.24	1	**0.13**
c090	50	10	445	**0.23**	1	0.36	1	0.33
c080	50	20	19525	**1.85**	7	12.58	45	6.13
c070	50	30	217903	15.85	11	16.89	49	**8.06**
c030	50	70	33641339	2212.75	23	23.42	51	**5.46**
c010	50	90	24811307	1785.16	19	15.49	55	**5.30**
c000	50	100	40794151	2401.53	17	13.93	43	**4.67**
c290	52	10	380	**0.26**	1	0.46	1	0.40
c490	54	10	1591	**0.41**	1	0.66	15	3.30
c690	56	10	4675	0.66	1	**0.38**	3	1.00
c890	58	10	16883	**1.98**	13	28.45	71	17.53
s090	60	10	10674	**1.47**	3	5.56	37	9.90

Table 6 Mesh instances

			CQB		BiqMac		KRC		SEN	
Graph	n	(%)	#nodes	Time	#nodes	Time	#nodes	Time	#nodes	Time
m4	32	10	23	0.04	1	0.04	1	**0.03**	1	0.14
ma	54	5	9	0.17	1	**0.05**	1	0.10	1	0.28
me	60	5	16	0.20	1	0.14	1	**0.13**	1	0.28
m6	70	5	297	**0.45**	1	0.68	1	1.23	1	1.43
mb	74	4	90	**0.37**	1	0.62	1	0.98	1	1.14
mc	74	5	370	**0.41**	1	0.49	1	1.53	1	1.43
md	80	4	97	**0.53**	1	0.87	1	0.96	1	1.28
mf	90	4	105	**0.67**	1	1.42	1	0.80	1	1.85
m1	100	3	212	**0.99**	3	18.74	15	36.50	1	3.00
m8	148	2	4774	4.35	1	**3.06**	1	10.70	1	4.14

Table 7 gives results for binary de Bruijn graphs which arise in applications related to parallel computer architecture [7,12]. These graphs are constructed by the following procedure. We first build a directed graph using the Mathematica command:

Table 7 de Bruijn networks

			CQB		BiqMac		KRC		SEN	
Graph	n	(%)	#nodes	Time	#nodes	Time	#nodes	Time	#nodes	Time
debr5	32	12	46	0.07	1	0.08	3	0.20	1	**0.00**
debr6	64	6	3373	**0.48**	1	0.84	55	15.63	1	1.00
debr7	128	3	50314492	3092.34	71	471.94	711	2796.96	1	**10.28**

Table 8 Compiler design

			CQB		BiqMac		KRC		SEN		ARM-LP		ARM-SDP	
Graph	n	(%)	#nodes	Time	#nodes	Time	#nodes	Time	#nodes	Time	#nodes	Time	#nodes	Time
cd30	30	13	11	0.05	1	0.10	1	0.03	1	**0.00**	326	1.53	10276	196.92
cd45	45	10	34	0.25	1	0.26	1	**0.23**	1	0.57	49	3.84	2995	336.92
cd47a	47	9	42	0.31	1	**0.30**	1	0.33	7	1.00	100	2.03	4403	333.07
cd47b	47	9	55	**0.29**	1	0.81	35	3.73	3	1.43	12	3.84	963	86.92
cd61	61	10	78	0.80	7	6.31	1	**0.67**	6	6.00	785	62.30		failed

```
A = TableForm[ToAdjacencyMatrix[DeBruijnGraph[2, n]]]
```

To obtain the graph partitioning test problem, we add the Mathematica generated matrix to its transpose and set the diagonal to 0. For this test set, SEN had by far the best performance in 2 of 3 instances, while CQB was fastest in 1 instance.

Table 8 gives results for compiler design problems [13,26]. For this test set, CQB, BiqMac and KRC were fastest for 1 problem, when KRC was fastest in 2 out of 5 test problems. Note that the times for CQB were competitive with KRC even when KRC was faster.

Table 9 gives results for finite element meshes [8], KKT systems [20], and Johnson graphs [25]. We report the comparison of CQB, BiqMac and the algorithms of Armbruster [1]: ARM-LP and ARM-SDP. For KKT system graphs, the results reported in [1] is not for equicut problems, i.e. $u - l \geq 2$ in (6.1), hence BiqMac is not applicable. We see in Table 9 that CQB and ARM-LP were each fastest in 5 cases, while BiqMac and ARM-SDP were each fastest in 2 cases. Note that ARM-LP failed to solve 3 problems to optimality within 5 h on their computer, BiqMac reached the time limit of 36 h on our machine for the G250,2.5 graph.

7 Conclusions

An exact algorithm is presented for solving the graph partitioning problem with upper and lower bounds on the size of each set in the partition. The algorithm is based on a continuous quadratic programming formulation of the discrete partitioning problem. We show how to transform a feasible $\mathbf{x}$ for the graph partitioning QP (2.1) to a binary feasible point $\mathbf{y}$ with an objective function value which satisfies $f(\mathbf{y}) \leq f(\mathbf{x})$. The

Table 9 Finite element mesh, KKT systems, and Johnson graphs

			CQB		BiqMac		ARM-LP		ARM-SDP	
Graph	n	(%)	#nodes	Time	#nodes	Time	#nodes	Time	#nodes	Time
mesh.69.112	69	0.5	121	0.36	1	3.41	1	**0.02**	1	0.76
mesh.74.129	74	4.8	1462	**0.49**	1	0.61	1	11.95	1	1.20
mesh.137.231	137	2.5	9468	5.35	3	30.28	1	9.52	1	**2.20**
mesh.138.232	138	2.5	15240	**6.91**	1	11.36	1	36.37	1	7.86
mesh.148.265	148	2.5	4774	4.30	1	2.98	1	**0.67**	1	0.85
mesh.274.231	274	0.6	18628	7.88	3	76.43	1	**2.02**	1	8.51
mesh.274.469	274	1.3	24241	24.62	105	3581.05	1	153.53	1	**14.90**
KKT.lowt01	82	7.8	105	0.57			1	**0.15**	1	0.69
KKT.putt01	115	6.6	161	**1.51**			1	3.36	1	1.56
KKT.capt09	2063	0.5	549610	4658.59			634	**1467.25**		failed
G124,2.5	124	2.0	3120	**4.21**	539	2054.46	7	10.70	9	104.52
G124,5	124	4.2	4173417	**953.36**	225	1229.72		failed	31	3375.13
G124,10	124	8.1	500645440	101058.58	7069	**39685.91**		failed	1541	61449.53
G124,20	124	16.7	414725123	82380.11	4015	**18322.35**		failed	2153	49798.59
G250,2.5	250	1.1	9709688	10106.14		$>$ 36 h	3063	**1409.42**	13	7476.43

binary feasible point corresponds to a partition of the graph vertices and $f(\mathbf{y})$ is the weight of the cut edges. At any stationary point of (2.1) which is not a local minimizer, Proposition 5.3 provides a descent direction that can be used to strictly improve the objective function value.

In the branch and bound algorithm, the objective function is decomposed into the sum of a convex and a concave part. A lower bound for the objective function is achieved by replacing the concave part by an affine underestimate. Two different decompositions were considered, one based on the minimum eigenvalue of the matrix in the objective function, and the other based on the solution to a semidefinite programming problem. The semidefinite programming approach generally led to much tighter lower bounds. In a series of numerical experiments, the new algorithm CQB (convex quadratic branch and bound) was competitive with state-of-the-art partitioning methods; the relative performance of CQB was better for sparse graphs than for dense graphs.

Acknowledgments Constructive comments by the associate editor and reviewers are gratefully acknowledged. We would like to thank Franz Rendl and Angelika Wiegele for making available to us their BiqMac code.

References

1. Armbruster, M.: Branch-and-Cut for a Semidefinite Relaxation of Large-Scale Minimum Bisection Problems. PhD thesis, Chemnitz University of Technology, Chemnitz (2007)
2. Armbruster, M., Fügenschuh, M., Helmberg, C., Martin, A.: A comparative study of linear and semidefinite branch-and-cut methods for solving the minimum graph bisection problem. In: Lodi, A.,

Panconesi, A., Rinaldi, G. (eds.) Proceedings of the 13th International Integer Programming and Combinatorial Optimization Conference, vol. 5035 of Lecture Notes in Computer Science, pp. 112–124. Springer, (2008)
3. Barzilai, J., Borwein, J.M.: Two point step size gradient methods. IMA J. Numer. Anal. **8**, 141–148 (1988)
4. Borchers, B.: CSDP, A C library for semidefinite programming. Optim. Methods Softw. **11**(1), 613–623 (1999)
5. Brunetta, L., Conforti, M., Rinaldi, G.: A branch-and-cut algorithm for the equicut problem. Math. Program. **78**, 243–263 (1997)
6. Catalyürek, U.V., Aykanat, C.: Hypergraph-partitioning based decomposition for parallel sparse-matrix vector multiplication. IEEE Trans. Parallel Distrib. Syst. **10**, 673–693 (1999)
7. Collins, O., Dolinar, S., McEliece, R., Pollara, F.: A VLSI decomposition of the de Bruijn graph. J. ACM **39**, 931–948 (1992)
8. de Souza, C.C.: The Graph Equipartition Problem: Optimal Solutions, Extensions and Applications. PhD thesis, Université Catholique de Louvain, Louvain-la-Neuve (1993)
9. Devine, K., Boman, E., Heaphy, R., Bisseling, R., Catalyurek, U.: Parallel hypergraph partitioning for scientific computing. In: Proceedings of 20th International Parallel and Distributed Processing Symposium (IPDPS'06), IEEE (2006)
10. Dongarra, J.J.: Performance of various computers using standard linear equations software. Technical Report cs-89-85, University of Tennessee, Knoxville (2008)
11. Falkner, J., Rendl, F., Wolkowicz, H.: A computational study of graph partitioning. Math. Program. **66**, 211–240 (1994)
12. Feldmann, R., Monien, B., Mysliwietz, P., Tschöke, S.: A better upper bound on the bisection width of de Bruijn networks. In: STACS 97, Lecture Notes in Computer Science, vol. 1200, pp. 511–522. Springer, Berlin (1997)
13. Ferreira, C.E., Martin, A., Souza, C.C.de , Weismantel, R., Wolsey, L.A.: The node capacitated graph partitioning problem: a computational study. Math. Program. **81**, 229–256 (1998)
14. Garey, M.R., Johnson, D.S., Stockmeyer, L.: Some simplified NP-complete graph problems. Theor. Comput. Sci. **1**, 237–267 (1976)
15. Gilbert, J.R., Miller, G.L., Teng, S.H.: Geometric mesh partitioning: implementation and experiments. SIAM J. Sci. Comput. **19**, 2091–2110 (1998)
16. Grigori, L., Boman, E., Donfack, S., Davis, T.A.: Hypergraph-based unsymmetric nested dissection ordering for sparse LU factorization. SIAM J. Sci. Comput. (2008). under submission
17. Hager, W.W., Krylyuk, Y.: Graph partitioning and continuous quadratic programming. SIAM J. Discret. Math. **12**, 500–523 (1999)
18. Hager, W.W., Krylyuk, Y.: Multiset graph partitioning. Math. Meth. Oper. Res. **55**, 1–10 (2002)
19. Hager, W.W., Phan, D.T.: An ellipsoidal branch and bound algorithm for global optimization. SIAM J. Optim. **20**, 740–758 (2009)
20. Helmberg, C.: A cutting plane algorithm for large scale semidefinite relaxations. In: Grötschel, M. (ed.) The Sharpest Cut, MPS-SIAM Series Optimization, pp. 233–256. SIAM, Philadelphia (2004)
21. Hendrickson, B., Leland, R.: The Chaco user's guide—Version 2.0, Sandia National Laboratories. Technical Report SAND94-2692 (1994)
22. Hendrickson, B., Leland, R.: An improved spectral graph partitioning algorithm for mapping parallel computations. SIAM J. Sci. Comput. **16**, 452–469 (1995)
23. Hendrickson, B., Leland, R.: A multilevel algorithm for partitioning graphs. In: Proceedings of Supercomputing '95, ACM, November (1995)
24. Horst, R., Pardalos, P.M., Thoai, N.V.: Introduction to Global Optimization. Kluwer, Dordrecht (1995)
25. Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C.: Optimization by simulated annealing: an experimental evaluation. Part I, graph partitioning. Oper. Res. **37**, 865–892 (1989)
26. Johnson, E.L., Mehrotra, A., Nemhauser, G.L.: Min-cut clustering. Math. Program. **62**, 133–151 (1993)
27. Johnson, T.: A concurrent dynamic task graph. In: Proceedings of 1993 International Conference on Parallel Processing, (TR-93-011, CISE Department, University of Florida) (1993)
28. Karisch, S.E., Rendl, F., Clausen, J.: Solving graph bisection problems with semidefinite programming. INFORMS J. Comput. **12**, 177–191 (2000)
29. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J. Sci. Comput. **20**, 359–392 (1998)

30. Karypis, G., Kumar, V.: Parallel multilevel k-way partitioning scheme for irregular graphs. SIAM Rev. **41**, 278–300 (1999)
31. Karypis, G., Kumar, V.: Multilevel k-way hypergraph partitioning. VLSI Des. **11**, 285–300 (2000)
32. Lengauer, T.: Combinatorial Algorithms for Integrated Circuit Layout. Wiley, Chichester (1990)
33. Mitchell, J.: Branch-and-cut for the *k*-way equipartition problem. Technical report, Department of Mathematical Sciences, Rensselaer Polytechnic Institute (2001)
34. Murty, K.G., Kabadi, S.N.: Some NP-complete problems in quadratic and linear programming. Math. Program. **39**, 117–129 (1987)
35. Pardalos, P.M., Vavasis, S.A.: Quadratic programming with one negative eigenvalue is NP-hard. J. Glob. Optim. **1**, 15–22 (1991)
36. Pellegrini, F., Roman, J., Amestoy, P.R.: Hybridizing nested dissection and halo approximate minimum degree for efficient sparse matrix ordering. Concurr. Pract. Exp. **12**, 68–84 (2000)
37. Preis, R., Diekmann, R.: The PARTY partitioning library user guide—version 1.1 (1996)
38. Rendl, F., Rinaldi, G., Wiegele, A.: Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. Math. Program. **121**, 307–335 (2010)
39. Sensen, N.: Lower bounds and exact algorithms for the graph partitioning problem using multicommodity flows. In: Lecture Notes in Computer Science, vol. 2161, pp. 391–403. Springer, London (2001)
40. Sinclair, M.: An exact penalty function approach for nonlinear integer programming problems. Eur. J. Oper. Res. **27**, 50–56 (1986)
41. Soper, A.J., Walshaw, C., Cross, M.: A combined multilevel search and multilevel optimization approach to graph-partition. J. Glob. Optim. **29**, 225–241 (2004)
42. Souza, C., Keunings, R., Wolsey, L., Zone, O.: A new approach to minimizing the frontwidth in finite element calculations. Comput. Methods Appl. Mech. Eng. **111**, 323–334 (1994)
43. Teng, S.-H.: Provably good partitioning and load balancing algorithms for parallel adaptive N-body simulation. SIAM J. Sci. Comput. **19**, 635–656 (1998)
44. Walshaw, C., Cross, M., Everett, M.: Parallel dynamic graph partitioning for adaptive unstructured meshs. J. Parallel Distrib. Comput. **47**, 102–108 (1997)