J Glob Optim (2010) 48:57–71
DOI 10.1007/s10898-009-9520-1

Linear and quadratic programming approaches for the general graph partitioning problem

Neng Fan · Panos M. Pardalos

Received: 15 December 2009 / Accepted: 18 December 2009 / Published online: 5 January 2010

Abstract The graph partitioning problem is to partition the vertex set of a graph into a number of nonempty subsets so that the total weight of edges connecting distinct subsets is minimized. Previous research requires the input of cardinalities of subsets or the number of subsets for equipartition. In this paper, the problem is formulated as a zero-one quadratic programming problem without the input of cardinalities. We also present three equivalent zero-one linear integer programming reformulations. Because of its importance in data biclustering, the bipartite graph partitioning is also studied. Several new methods to determine the number of subsets and the cardinalities are presented for practical applications. In addition, hierarchy partitioning and partitioning of bipartite graphs without reordering one vertex set, are studied.

Keywords Graph partitioning · Linear programming · Quadratic programming · Bipartite graph partitioning

1 Introduction

The graph partitioning problem is an NP-complete [12] combinatorial optimization problem, and it partitions the vertex set of a graph into several disjoint subsets so that the sum of weights of the edges between the disjoint subsets is minimized.

A very early method for graph partitioning is the Kernighan-Lin algorithm [18], which is a heuristic algorithm. There are several surveys [1,7,11] regarding exact and heuristic

This paper is dedicated to Professor Roman G. Strongin in honor of his 70th birthday.

N. Fan (✉) · P. M. Pardalos
Center for Applied Optimization, Department of Industrial and Systems Engineering,
University of Florida, Gainesville, FL, USA
e-mail: andynfan@ufl.edu

P. M. Pardalos
e-mail: pardalos@ufl.edu

algorithms for graph partitioning. Graph partitioning has been related to clustering [24], which is a method of unsupervised classification used in data mining and image analysis. Applications of graph partitioning can be found in VLSI design [16], biological or social networks [9] and data mining [29].

Let $G = (V, E)$ be an undirected graph with a set of vertices $V = \{v_1, v_2, \dots, v_N\}$ and a set of edges $E = \{(v_i, v_j) :$ edge between vertices v_i and v_j, $1 \le i, j \le N\}$, where N is the number of vertices. The weights of the edges are given by a matrix $W = (w_{ij})_{N \times N}$, where $w_{ij}(> 0)$ denotes the weight of edge (v_i, v_j) and $w_{ij} = 0$ if no edge (v_i, v_j) exists between vertices v_i and v_j. This matrix is symmetric for undirected graphs G and is the adjacency matrix of G if $w_{ij} \in \{0, 1\}$.

Let K be the number of disjoint sets that we want to partition V into, and $n_1, \dots, n_K$ be the numbers of vertices within each sets. The general graph partitioning problem can be described as partitioning the vertex set V into K disjoint subsets so that the sum of weights of edges that connect vertices among different subsets is minimized.

In previous research, the graph is partitioned into equal or different by 1 cardinalities for all subsets either by linear programming [20] or semidefinite programming [17,20]. The approaches by quadratic programming [13,14] and semidefinite programming [28] require the given cardinalities $n_1, \dots, n_K$. For the special case of $K = 2$, it is a bipartition studied by spectral methods [8] and quadratic programming [13]. However, these approaches are still based on relaxations of exact formulations.

For large graphs, the multilevel and hierarchical methods are used for partitioning [15]. In addition, parallel formulation of graph partitioning [15] and parallel computation in optimization [26,27] are useful for partitioning of large graphs with thousands of vertices and edges. In this case, graph partitioning is a problem with very high dimensions, and several approaches are presented in for parallel computing [26,27].

In this paper, the cardinalities for each subsets are not required and the number $K(1 < K < N)$ of nonempty subsets is the only required information for possible equal or unequal partitioning. The quadratic and linear programming models are constructed to solve this general graph partitioning problem by algorithms based on the exact formulations. Although we claim that the K should be considered as an input for general graph partitioning, we also present the methods for determination of K, and the given cardinalities $n_k(k = 1, \dots, K)$ with possible changes of level $\hat{n}_k$.

In addition, the bipartite graph $G = (V, U, E)$, consists of two sets V, U of vertices with all edges in E between vertices from different vertex sets. The weights of this graph can be stored in a biadjacency weighted matrix. The bipartite graph partitioning problem is to partition both V and U into K subsets, respectively. In data mining, biclustering is to partition a data matrix in both rows and columns. The relationship between biclustering and bipartite graph partitioning is studied in [8]. In this paper, the partitioning for bipartite graph is also formulated as a quadratic and three linear programming models based on the results for general graphs.

The rest of this paper is organized as follows: In Sect. 2, the constraints and objective function are constructed; In Sect. 3, the quadratic programming models and their relaxations are studied, and in addition, the quadratic models for bipartite graph partitioning are also presented; In Sect. 4, three linear programming approaches are introduced; In Sect. 5, several cases, such as the determination of the number or cardinalities of subsets, hierarchy partitioning and partitioning of bipartite graphs without reordering one vertex set, are presented; In Sect. 6, numerical experiments for many graphs under different conditions are performed after comparing these approaches; Sect. 7 concludes this paper.

2 Preliminary

Based on the graph $G = (V, E)$ described above, let x_{ik} be the decision variables denoting that vertex v_i belongs to kth subset if $x_{ik} = 1$, otherwise $x_{ik} = 0$. Let X be the matrix $X = (x_{ik})_{N\times K}$ and the constraints for general graph partitioning problem are constructed as follows:

(1) Exclusive constraints. Every vertex v_i can only belong to exactly one subset of V, and so every row sum of X is 1,

$$\mathscr{R} = \left\{ x_{ik} \in R^{N\times K} : \sum_{k=1}^{K} x_{ik} = 1 \right\}. \tag{1}$$

For $i = 1, \dots, N$, all vertices belong to some subsets, and we have partitioned the vertex set into several subsets.

(2) Cardinality (size) constraints. Let the cardinality of the kth subset be $n_k = \sum_{i=1}^{N} x_{ik}$. In order to eliminate the case that all vertices belong to only one subset, every subset should be nonempty ($\mathscr{S}_a$). Some partitioning results require the order of subsets ($\mathscr{S}_b$), and $MaxCard$ denotes the maximum size of each subset ($\mathscr{S}_c$). The constraints $\mathscr{S}_c$ can be considered as an equipartition if $MaxCard$ is chosen as an average cardinality of N/K. In the following models, we always choose the most general case $\mathscr{S}_a$ so that the graph can be partitioned according to its structure. Since we have relaxed the size constraints for graph partitioning, the term "general graph partitioning" is used in this paper.

$$\mathscr{S}_a = \left\{ x_{ik} \in R^{N\times K} : n_k \geq 1 \right\}, \tag{2a}$$

$$\mathscr{S}_b = \left\{ x_{ik} \in R^{N\times K} : n_1 \geq n_2 \geq \cdots \geq n_k \geq 1 \right\}, \tag{2b}$$

$$\mathscr{S}_c = \left\{ x_{ik} \in R^{N\times K} : n_k \leq MaxCard \right\}. \tag{2c}$$

(3) Anti-degeneracy constraints. The subset indexes of the vertices can be restricted that the first vertex must belong to first subset, the second vertex must belong either to first or to second subset, and continually, the kth vertex must belong to one of the first k subsets.

$$\mathscr{D} = \left\{ x_{ik} \in R^{N\times K} : x_{ik} = 0, \quad i = 1, \dots K, \quad k = i+1, \dots, K \right\}. \tag{3}$$

With these constraints, the formulations are less degenerated because of the constraints on permutations of vertex assignment.

(4) Binary constraints. Every x_{ik} indicates that vertex v_i belongs to kth subset or not.

$$\mathscr{B} = \left\{ x_{ik} \in R^{N\times K} : x_{ik} = 0 \text{ or } 1 \right\}. \tag{4}$$

(5) Nonnegative constraints. The binary constraints can be relaxed to nonnegative ones which ensure the partitioning of G under some conditions.

$$\mathscr{P} = \left\{ x_{ik} \in R^{N\times K} : x_{ik} \geq 0 \right\}. \tag{5}$$

The sum of weights of edges that connect vertices among different subsets is inter-similarity, and correspondingly, the sum of weights of edges whose both ends belong to same subset is intra-similarity. Thus, the sum of inter- and intra-similarities covering the weights of all edges is a constant once the weighted matrix W is known. Therefore, the objective of minimizing the inter-similarity is equivalent to maximizing the intra-similarity.

If edge (v_i, v_j) has two ends in the same subset, $x_{ik} = x_{jk}$ for all $k = 1, \ldots, K$ and only one k exists such that $x_{ik} = x_{jk} = 1$. Thus the intra-similarity can be expressed as $\sum_{(v_i,v_j)\in E} \sum_{k=1}^{K} w_{ij} x_{ik} x_{jk}$ or

$$\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{K} w_{ij} x_{ik} x_{jk} = \frac{1}{2} trace(X^T W X), \tag{6}$$

since $w_{ij} = 0$ if no edge between vertices v_i and v_j and $w_{ij} = w_{ji}$ for the same edges (v_i, v_j) and (v_j, v_i). Then, the inter-similarity is $\frac{1}{2}(e^T W e - trace(X^T W X))$, where $e^T W e$ is the total weights of all edges and e is a vector of all elements being 1.

3 Quadratic programming approaches

From (6), the objective function can be expressed in the form max $\frac{1}{2} trace(X^T W X)$. As descriptions above, the constraints (1) and (4) are prerequisite for graph partitioning problem. In our general case, the constraints (2a) are chosen so that the requirement of the cardinality of each subset is not necessarily to be given.

3.1 Quadratic programming models and relaxations

The problem can be formulated as a zero-one quadratic program as follows.

$$\begin{aligned} \max \quad & \frac{1}{2} trace(X^T W X) \\ s.t. \quad & x_{ik} \in \mathscr{R} \cap \mathscr{S}_a, \\ & x_{ik} \in \mathscr{B}, \qquad i = 1, \ldots, N, k = 1, \ldots, K. \end{aligned} \tag{7}$$

The objective in (7) is not standard in a quadratic programming. Denoting $X = (x_{ij})_{N\times K} = (x_1, \ldots, x_k, \ldots, x_K)$, where x_ks are column vectors, $\hat{X} = (x_1^T, x_2^T, \ldots, x_K^T)^T$, and

$$\hat{W} = \begin{pmatrix} W & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & W & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & W \end{pmatrix},$$

with dimension $(NK) \times (NK)$, where $\mathbf{0}$ is a matrix with all elements being 0 with proper dimension, the standard formulation for the objective in (7) is

$$\frac{1}{2} trace(X^T W X) = \frac{1}{2}\sum_{k=1}^{K} x_k^T W x_k = \frac{1}{2}\hat{X}^T \hat{W} \hat{X}. \tag{8}$$

Usually, the constrains $\mathscr{D}$ in (3) are added to reduce the degeneracy, and (2c) are added as a requirement of loose cardinalities, where $MaxCard$ can be chosen loosely (for example, $MaxCard = N/(K-1)$). Let $D = (d_{ij})_{N\times N}$ be a diagonal matrix. By properly choosing the elements d_{ii}, the binary constraints (4) can be relaxed to nonnegative ones (5) by the following Theorem 1 and for the proof details, we refer to [13]. Thus, the following continuous formulation can be obtained.

$$\begin{aligned} \max \quad & \frac{1}{2} trace(X^T(W+D)X) \\ s.t. \quad & x_{ik} \in \mathscr{R} \cap \mathscr{S}_a, \\ & x_{ik} \in \mathscr{P}, \qquad i=1,\ldots,N, k=1,\ldots,K. \end{aligned} \tag{9}$$

Theorem 1 (Theorem 6.1, [13]) *If D is chosen to satisfy $d_{ii}+d_{jj} \geq 2a_{ij}$ for each i and j, then the continuous problem (9) has a maximizer contained in $\mathscr{B}$, and hence, this maximizer is a solution of the discrete problem (7). Conversely, every solution to (7) is also a solution to (9). Moreover, if $d_{ii}+d_{jj} > 2w_{ij}$ for each i and j, then every local maximizer for (9) lies in $\mathscr{B}$.*

There are many approaches for solving (7), including heuristic and exact optimization ones as reviewed in the paper [25] by Sherali and Smith. The heursitc approaches include the rank-two relaxation by Burer et al. [3], the evolutionary procedure by Lodi et al. [21] and the Tabu Search algorithm by Kochenberger et al. [19]. The exact optimization approaches include the branch-and-bound algorithm by Pardalos and Rodgers [23], and the Lagrangian decomposition algorithm by Chardaire and Sutter [5]. The most recent survey including heuristic and exact optimization approaches can be found for a similar quadratic assignment problem in [22] by Loiola et al. In this paper, CPLEX 11.0 is used to solve the zero-one quadratic programming problem, and CPLEX uses multiple types of most recent algorithms [6].

3.2 Quadratic approaches for bipartite graphs

Let $G=(V,U,E)$ be a bipartite graph with vertex sets $V=\{v_1,\ldots,v_N\}, U=\{u_1,\ldots,u_M\}$ and edge set $E=\{(v_i,u_j):$ edge between vertices v_i and $u_j, 1\leq i\leq N, 1\leq j\leq M\}$, where N and M are the numbers of vertices within two sets, respectively. Usually, instead of weighted matrix, the biadjacency weighted matrix $A=(a_{ij})_{N\times M}$ is given where $a_{i,j}$ is the weight of edge (v_i,u_j), and its corresponding weighted matrix W for G can be constructed as

$$W=\begin{pmatrix} \mathbf{0} & A \\ A^T & \mathbf{0} \end{pmatrix}.$$

By the weighted matrix W and considering $V\cup U$ as the vertex set, the previous formulations can be used. However, some problems (e.g., biclustering, [8]) based on bipartite graph models require partitioning of both vertex set V and U into K disjoint subsets so that the kth subsets of V and also of U as a whole subset. The sum of weights of edges among the union subsets has to be minimized.

Denoting $X=\begin{pmatrix} X_v \\ X_u \end{pmatrix}$, where $X_v=(x^v_{ik})_{N\times K}$ and $X_u=(x^u_{jk})_{M\times K}$ are indicators for vertices from V and U, respectively, the equivalent objective can be expressed as

$$\max \frac{1}{2} trace(X^TWX) = \max trace(X_v^TAX_u). \tag{10}$$

The constraints for X_v are still the ones (1), (2a) and (4), while the index should be changed to $j=1,\ldots,M$ instead of $i=1,\ldots,N$ for X_u. Let $\mathscr{R}'$, $\mathscr{S}'_a$ and $\mathscr{B}'$ be the constraints for X_u with changed indexes. The zero-one quadratic program can be formulated as

$$
\begin{aligned}
\max \quad & trace(X_v^T A X_u) \\
s.t. \quad & x_{ik}^v \in \mathscr{R} \cap \mathscr{S}_a \cap \mathscr{B}, \\
& x_{jk}^u \in \mathscr{R}' \cap \mathscr{S}_a' \cap \mathscr{B}', \\
& \quad i = 1, \ldots, N, j = 1, \ldots, M, k = 1, \ldots, K.
\end{aligned}
\tag{11}
$$

Theorem 1 still holds in this case by considering X_v and X_u as submatrices of X, and W is also symmetric. A continuous case of program (11) can be obtained by changing $\mathscr{B}$, $\mathscr{B}'$ to $\mathscr{P}$, $\mathscr{P}'$ by constructing a diagonal matrix D of dimension $(N + M) \times (N + M)$. In addition, the anti-degeneracy constraints $\mathscr{D}$ can only be put on one of X_v and X_u for bipartite graphs.

4 Linear programming approaches

4.1 Direct linear programming models

The objective function (6) is nonlinear. Here, two approaches, similarly as presented in [2], are used to transform this nonlinear objective into a linear one.

Let y_{ijk} denote indicator that edge (i, j) with two ends i, j belong to subset k if $y_{ijk} = 1$ or not if $y_{ijk} = 0$. Thus $y_{ijk} = x_{ik}x_{jk}, i, j = 1, \ldots, N, k = 1, \ldots, K$. The linearization is as follows

$$y_{ijk} \le x_{ik}, y_{ijk} \le x_{jk}, y_{ijk} \ge x_{ik} + x_{jk} - 1 \text{ and } y_{ijk} \ge 0.$$

Since y_{ijk} is in the objective to maximize, the constraints $y_{ijk} \ge 0$ can be eliminated for $w_{ij} \ge 0$. The linear programming formation for general graph partitioning problem is

$$
\begin{aligned}
\max \quad & \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{K} w_{ij} y_{ijk} \\
s.t. \quad & x_{ik} \in \mathscr{R} \cap \mathscr{S}_a \cap \mathscr{B}, \\
& \begin{cases} y_{ijk} \le x_{ik}, \\ y_{ijk} \le x_{jk}, \\ y_{ijk} \ge x_{ik} + x_{jk} - 1, \end{cases} \\
& i, j = 1, \ldots, N, k = 1, \ldots, K.
\end{aligned}
\tag{12}
$$

The linearized formulation (12) introduces K more variables and $3K$ more constraints for each edge comparing with the original one (7). Let z_{ij} be a binary variable such that $z_{ij} = 1$ if the vertices v_i, v_j of edge (v_i, v_j) belongs to the same subset and 0 otherwise. Thus $z_{ij} = \sum_{k=1}^{K} x_{ik}x_{jk}$ and $z_{ij} = 1$ if and only if for some k such that $x_{ik} = x_{jk} = 1$ and all others $x_{ik'} = x_{jk'} = 0$. The linearization of $z_{ij} = \sum_{k=1}^{K} x_{ik}x_{jk}$ can be formulated as

$$z_{ij} \le 1 + x_{ik} - x_{jk}, z_{ij} \le 1 - x_{ik} + x_{jk},$$

for all i, j, k and all other requirements of this formulations are eliminated for the objective to be maximized. Therefore, we obtain another linear programming formulation as

$$\begin{aligned} \max \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}z_{ij} \\ s.t. \quad & x_{ik} \in \mathscr{R} \cap \mathscr{S}_a \cap \mathscr{B}, \\ & \begin{cases} z_{ij} \le 1 + x_{ik} - x_{jk}, \\ z_{ij} \le 1 - x_{ik} + x_{jk}, \end{cases} \\ & i, j = 1, \ldots, N, k = 1, \ldots, K. \end{aligned} \tag{13}$$

In practice, the anti-degeneracy constraints $\mathscr{D}$ and the cardinality constraints $\mathscr{S}_c$ can be added to both programs (12) and (13).

4.2 An equivalent linear programming approach

The Laplacian matrix for $G = (V, E)$ with W is defined as $L = diag(\sum_j w_{1j}, \ldots, \sum_j w_{Nj}) - W$. As described in (6) and (8), the objective for partitioning the graph G can also be expressed as $\frac{1}{2}trace(X^T LX)$, and the equivalent quadratic programming formulation is

$$\begin{aligned} \min \quad & \frac{1}{2}trace(X^T LX) \\ s.t. \quad & x_{ik} \in \mathscr{R} \cap \mathscr{S}_a \cap \mathscr{B}, \\ & i = 1, \ldots, N, k = 1, \ldots, K. \end{aligned} \tag{14}$$

By the methods proposed in [4] and [25], defining $S = (s_{ik})_{N\times K} = (s_1, \ldots, s_K)$, $T = (t_{ik})_{N\times K} = (t_1, \ldots, t_K)$, where s_k, t_k are column vectors, the equivalent linear programming formulation is

$$\begin{aligned} \min \quad & e^T Se \\ s.t. \quad & x_{ik} \in \mathscr{R} \cap \mathscr{S}_a \cap \mathscr{B}, \\ & Lx_k - t_k - s_k + Ce = 0, \\ & t_k \le 2C(e - x_k), \\ & t_{ik}, s_{ik} \in \mathscr{P}, \\ & i = 1, \ldots, N, k = 1, \ldots, K. \end{aligned} \tag{15}$$

where the constant $C = 2\max_{i=1}^{N} \sum_{j=1}^{N} w_{ij}$.

From the methods in [4], the formulation (15) is equivalent to (14) after subtracting a constant in the objective of (15) by the following theorem.

Theorem 2 *Formulations* (14) *and* (15) *are equivalent.*

Proof Let X^0 be an optimal solution of the formulation (14). We claim that there exist $T, S(t_{ik} \ge 0, s_{ik} \ge 0)$ for all i, k such that

$$Lx_k^0 - t_k - s_k + Ce = 0, \tag{16}$$

$$t_k^T x_k^0 = 0. \tag{17}$$

The constraints $t_k \le 2C(e - x_k^0)$ in (15) is equivalent to (17) since x_{ik}^0 is binary and C is a positive constant. In the following, we use (17) to prove the equivalence.

Since $C = 2\max_{i=1}^{N}\sum_{j=1}^{N} w_{ij} = \|L\|_{\infty}, Lx_k + Ce \geq 0$, and there always exist $T, S(t_{ik} \geq 0, s_{ik} \geq 0)$ for all i, k such that both (16) and (17) hold. The claim is proved.

The variables S, T can be chosen as S^0, T^0 satisfying (16) and (17) so that $e^T Se$, the sum of s_{ik} for all i, k, is minimized. We claim that (X^0, S^0, T^0) is an optimal solution for the formulation (15).

Multiplying (16) by $(x_k^0)^T$, we have

$$\left(x_k^0\right)^T Lx_k^0 - \left(x_k^0\right)^T s_k + \left(x_k^0\right)^T Ce = 0, \tag{18}$$

by considering (17). Taking the sum over all $k = 1, \ldots, K$, and considering $trace(X^T LX) = \sum_{k=1}^{K} x_k^T Lx_k$, we obtain

$$trace((X^0)^T LX^0) - \sum_{k=1}^{K}\left(x_k^0\right)^T s_k^0 + \sum_{k=1}^{K}\left(x_k^0\right)^T Ce = 0. \tag{19}$$

In addition, $\sum_{k=1}^{K}(x_k^0)^T Ce = e^T Ce = C \cdot N$ is a constant since $x_{ik} \in \mathscr{R}$ in both (14) and (15). The equation

$$(x_k^0)^T s_k^0 = e^T s_k^0, \tag{20}$$

for $k = 1, \ldots, K$ can be proved by contradiction. Assume that for some i, we have $x_{ik}^0 = 0$ and $s_k^0 > 0$, where T^0, S^0 are chosen to minimized $e^T Se$. Defining the vectors $\tilde{t}_k, \tilde{s}_k$ and corresponding $\tilde{T}, \tilde{S}$ such that $\tilde{t}_{ik} = \tilde{t}_{ik}^0 + \tilde{s}_{ik}^0, \tilde{s}_{ik} = 0$ for $j \neq i, \tilde{t}_{jk} = t_{jk}, \tilde{s}_{jk} = s_{jk}$, in this case, $(X^0, \tilde{T}, \tilde{S})$ satisfies (16) and (17), and $e^T \tilde{S}e < e^T Se$, a contradiction to the assumption that (X^0, S^0, T^0) is optimal. Thus $(x_k^0)^T s_k^0 = e^T s_k^0$ holds for all $k = 1, \ldots, K$ and $\sum_{k=1}^{K}(x_k^0)^T s_k^0 = e^T S^0 e$, which leads (19) to

$$trace((X^0)^T LX^0) = e^T S^0 e - C \cdot N.$$

We have finished the proof that an optimal solution X^0 to (14) is also an optimal solution to (15) by choosing S^0, T^0 to minimize $e^T Se$. On the other direction, the proof is similar. □

From Theorem 2, the objectives of (15) and (12), (14) have the relation as follows,

$$e^T S^0 e = trace((X^0)^T LX^0) + C \cdot N = -trace((X^0)^T WX^0) + e^T We + C \cdot N,$$

where X^0 is optimal for (14) and (X^0, S^0, T^0) is optimal for (15).

The programs obtained in (12), (13) and (15) are all binary linear programming. The most widely used method for this kind of programming is branch and bound algorithm besides heuristic approaches, and in CPLEX 11.0 [6], the combined heuristic and exact methods are used.

4.3 Linear approaches for bipartite graphs

In Eq. (10), the objective for partitioning bipartite graph $G = (V, U, E)$ with weighted biadjacency matrix A is

$$\max trace(X_v^T AX_u) = \max \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{K} a_{ij}x_{ik}^{v}x_{jk}^{u}.$$

Similarly, defining $y_{ijk} = x^v_{ik}x^u_{jk}$, the linear programming formulation for partitioning bipartite graph is as follows,

$$\begin{aligned}
\max \quad & \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{K} a_{ij}y_{ijk} \\
s.t. \quad & x^v_{ik} \in \mathscr{R} \cap \mathscr{S}_a \cap \mathscr{B}, \\
& x^u_{jk} \in \mathscr{R}' \cap \mathscr{S}'_a \cap \mathscr{B}', \\
& \begin{cases} y_{ijk} \le x^v_{ik}, \\ y_{ijk} \le x^u_{jk}, \\ y_{ijk} \ge x^v_{ik} + x^u_{jk} - 1, \end{cases} \\
& i = 1, \ldots N, j = 1, \ldots M, k = 1, \ldots, K.
\end{aligned} \tag{21}$$

This formulation (21) introduces K more variables and $3K$ more constraints for each edge comparing with the original formulation (11). As in the formulation (13), defining $z_{ij} = \sum_{k=1}^{K} x^v_{ik}x^u_{jk}$, another linear programming formulation with less variables and constraints is obtained in the following.

$$\begin{aligned}
\max \quad & \sum_{i=1}^{N}\sum_{j=1}^{M} a_{ij}z_{ij} \\
s.t. \quad & x^v_{ik} \in \mathscr{R} \cap \mathscr{S}_a \cap \mathscr{B}, \\
& x^u_{jk} \in \mathscr{R}' \cap \mathscr{S}'_a \cap \mathscr{B}', \\
& \begin{cases} z_{ij} \le 1 + x^v_{ik} - x^u_{jk} \\ z_{ij} \le 1 - x^v_{ik} + x^u_{jk} \end{cases} \\
& i = 1, \ldots N, j = 1, \ldots M, k = 1, \ldots, K.
\end{aligned} \tag{22}$$

As in (15), defining $X_v = (x^v_{ik})_{N\times K} = (x^v_1, \ldots, x^v_K)$, $X_u = (x^u_{jk})_{M\times K} = (x^u_1, \ldots, x^u_K)$, $S_v = (s^v_{ik})_{N\times K} = (s^v_1, \ldots, s^v_K)$, $S_u = (s^u_{jk})_{M\times K} = (s^u_1, \ldots, s^u_K)$, $T_v = (t^v_{ik})_{N\times K} = (t^v_1, \ldots, t^v_K)$, $T_u = (t^u_{jk})_{M\times K} = (t^u_1, \ldots, t^u_K)$ and $D_v = diag(\ldots, \sum_{j=1}^{M} a_{ij}, \ldots)$, $D_u = diag(\ldots, \sum_{i=1}^{N} a_{ij}, \ldots)$, the linear programming formulation for bipartite graph $G = (V, U, E)$ with matrix A is

$$\begin{aligned}
\min \quad & e^T S_v e + e^T S_u e \\
s.t. \quad & x^v_{ik} \in \mathscr{R} \cap \mathscr{S}_a \cap \mathscr{B}, \\
& x^u_{jk} \in \mathscr{R}' \cap \mathscr{S}'_a \cap \mathscr{B}', \\
& D_v x^v_k - A x^u_k - t^v_k - s^v_k + Ce = 0, \\
& D_u x^u_k - A^T x^v_k - t^u_k - s^u_k + Ce = 0, \\
& t^v_k \le 2C(e - x^v_k), \\
& t^u_k \le 2C(e - x^u_k), \\
& t^v_{ik}, s^v_{ik} \in \mathscr{P}, \\
& t^u_{jk}, s^u_{jk} \in \mathscr{P}', \\
& i = 1, \ldots, N, j = 1, \ldots, M, k = 1, \ldots, K.
\end{aligned} \tag{23}$$

where the constant $C = \max\{\max_i 2\sum_j a_{ij}, \max_j 2\sum_i a_{ij}\}$. The equivalence of (23) with original quadratic formulation for bipartite graph can be proved by similar methods in Theorem 2.

The constraints $\mathscr{D}$ can also be added to formulations (21), (22) and (23) to reduce the degeneracy on X_v.

5 Several cases of graph partitioning problem

5.1 Determination of the number K of subsects

For controlling the cardinalities of subsets of V, the constraints $\mathscr{S}_a, \mathscr{S}_b, \mathscr{S}_c$ by notation $n_k = \sum_{i=1}^N x_{ik}$ are chosen under different situations. These constraints can be named nonempty constraints, order constraints and maximal cardinality constraints, respectively.

In previous research [13,17], either $n_1, \ldots, n_K$ are given or the equal partition with given K. In this paper, previous formulations require the input of K. However, the determination to input which $K \in \{2, 3, \ldots, N-1\}$ is still open. This section will present one method for determining K.

Let c_k be a binary variable such that $c_k = 1$ if the kth subset of V is nonempty, and $c_k = 0$ otherwise. Thus the cardinality constraints can be expressed as

$$\mathscr{S}_d = \left\{x_{ik} \in R^{N\times K} : n_k \geq c_k\right\}. \tag{24}$$

The term $\pm\sum_{k=1}^K c_k$ can be added to the objective function (add "+" when the original objective is to minimize, and "−" otherwise), in order to require the minimum possible number of subsets, by forcing some of the K clusters to be empty. Thus, K can be chosen as $N-1$ generally. When the constraints $\mathscr{S}_d$ is chosen to control the nonempty subsets, the maximal cardinality constraints $\mathscr{S}_c$ must be chosen at the same time, and nonempty constraints $\mathscr{S}_a$ cannot be chosen.

Determination of K has been discussed above to conquer the case of unknown number for the subsets in above. In the following, another method is presented for solving the general graph partitioning problem with the given cardinalities $n_1, \ldots, n_K$ but with some unknown changes. This will be an improvement of [13,14] and [28], which require only the exact cardinalities. Assume the cardinality of the kth subset changes over the integer interval $[n_k - \hat{n}_k, n_k + \hat{n}_k]$, where $\hat{n}_k \geq 0$ is an integer of the largest possible changes. The uncertainty can be fixed by robust optimization [10]. Here, a direct method is proposed.

From the constraint $\mathscr{R}$, we have

$$\sum_{k=1}^K\left(\sum_{i=1}^N x_{ik}\right) = \sum_{i=1}^N\left(\sum_{k=1}^K x_{ik}\right) = \sum_{i=1}^N 1 = N,$$

which guarantees that for any cardinality of each subset, the partitioning satisfies that all these N vertices should have their corresponding subsets. Therefore, the cardinality constraints, for given $n_1, \ldots, n_k$ with uncertain levels $\hat{n}_1, \ldots, \hat{n}_K$, respectively, can be established as

$$\mathscr{S}_e = \left\{x_{ik} \in R^{N\times K} : n_k - \hat{n}_k \leq \sum_{i=1}^N x_{ik} \leq n_k + \hat{n}_k, k = 1, \ldots, K\right\}. \tag{25}$$

In practice, n_k can be chosen as average cardinality for equipartition and $\hat{n}_k$ as a small number, to obtain a balanced partition.

5.2 Hierarchy graph partitioning

As pointed in Sect. 1, multilevel and hierarchical methods are used for large graphs. In Sect. 6, the implementations on CPLEX shows that the case $K = 2$ or $K = 3$ is very fast under the implementation of CPLEX. The hierarchy graph partitioning approach can be based on $K = 2$ or $K = 3$. At each level, after partitioning the vertex set into K subsets, the next partitioning can be computed on each subsets, and we can repeat these steps until we obtain the required number of subsets or structures.

5.3 Partitioning of bipartite graphs without reordering one vertex set

As mentioned above, the data matrix $A = (a_{ij})_{N\times M}$ is corresponding to the bipartite graph $G = (V, U, E)$ in Sect. 3.2. For the time series data matrix $A = (a_{it})_{N\times T}$ where index i corresponds to some objects and $t = 1, \dots, T$ to time points. In the partitioning process, the order of these time points should not be changed.

In order to include this condition, for any vertex $u_t \in U$, if it is partitioned into subset k, the next time point $u_{t+1} \in U$ should be in the same subset k or next subset $k + 1$. Mathematically, if $x^u_{11} = 1$, $x^u_{t+1,k} = 1$ or $x^u_{t+1,k+1} = 1$ must satisfy. Equivalently,

$$x^u_{t+1,k} + x^u_{t+1,k+1} \geq x^u_{tk}, \quad t = 1, \dots, T - 1, \quad k = 1, \dots, K - 1, \tag{26}$$

since all these variables are binary. In addition, for the last subset, another constraint

$$x^u_{t+1,K} \geq x^u_{tK}, \quad t = 1, \dots, T - 1, \tag{27}$$

should be added.

The constraints (26) and (27) can be added to any formulations (11), (21) or (22) to obtain the partitioning for bipartite graphs with one dimension as the type of time series.

6 Numerical experiments

6.1 Comparisons of two approaches

For the graph $G = (V, E)$ with N vertices and $|E|$ edges, the discrete quadratic programming (DQ) formulation(7), and the linear programming formulations L1 (12), L2 (13) and L3 (15) are compared in Table 1. The number of constraints for $\mathscr{R}$ is N while for $\mathscr{S}_a$ is K. For L3(15), the number of nonnegative constraints for S, T is not added.

From Table 1, the linear formulations introduce variables and constraints to eliminate quadratic variables in the objective, and the formulation (13) has less variables and constraints. The numbers of continuous variables and constraints in the formulations L1 and

Table 1 Comparisons of discrete programs

Formulation	Objective	No. 0–1 decisions	No. cont. variables	No. constraints
DQ (7)	Quadratic	$N \cdot K$	0	$N + K$
L1 (12)	Linear	$N \cdot K$	$\lvert E\rvert \cdot K$	$N + K + 3K \cdot \lvert E\rvert$
L2 (13)	Linear	$N \cdot K$	$\lvert E\rvert$	$N + K + 2K \cdot \lvert E\rvert$
L3 (15)	Linear	$N \cdot K$	$2N \cdot K$	$N + K + 2N \cdot K$

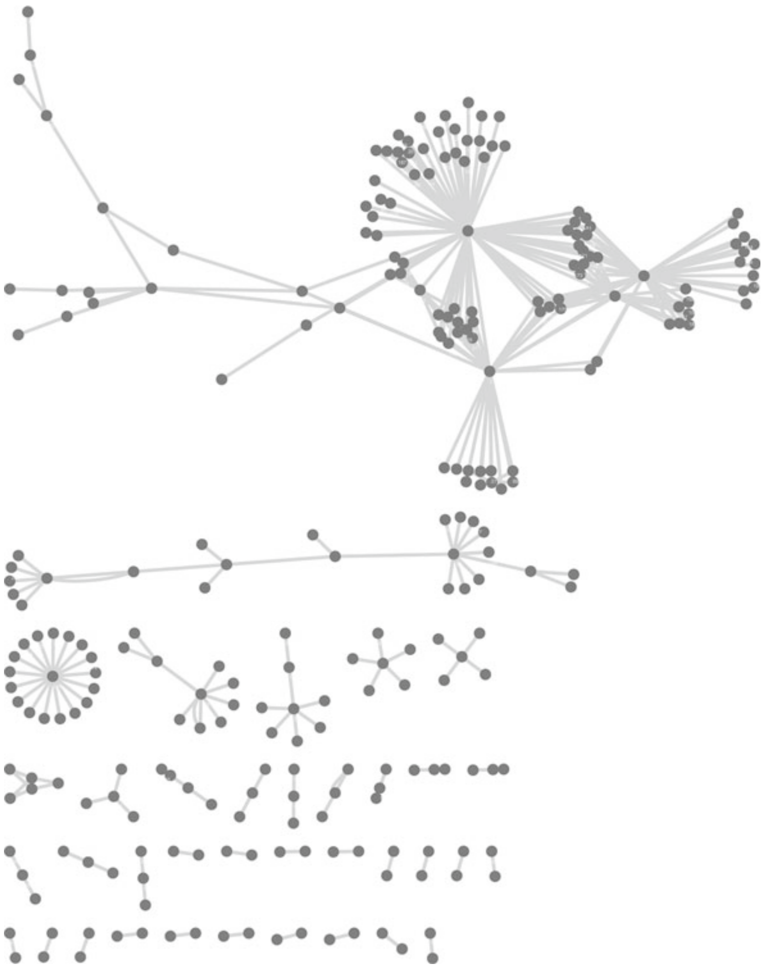

Fig. 1 An example for checking

L2 are related to the number of edges, while the number in formulation L3 is related to the number of vertices. Generally, L1 and L2 can be used for spare weighted matrix W. The cases for bipartite graph partitioning can be analyzed similarly.

6.2 Computational results

In this section, all programs are implemented using CPLEX 11.0 [6] via ILOG Concert Technology 2.5, and all computations are performed on a SUN UltraSpace- III with a 900 MHz processor and 2.0 GB RAM. Computational times are reported in CPU seconds.

Before comparing these different approaches, one example data set[1] from biological network (Fig. 1) is used to checking the models. Under these models, we can partitioning the vertices into connect components if the given K is the number of components. This graph

[1] http://biit.cs.ut.ee/graphweb/exampleInput/Cattle_protein_interactions_(IntAct).txt.

Table 2 Comparisons for general graphs

Graphs	Vertices N	Edges $\|E\|$	Subsets K	L1	L2	L3	DQ
Rand10	10	23	2	0.03	0.02	0.02	0.02
			3	2.01	0.27	0.03	0.08
			4	3.79	0.73	0.25	0.3
Rand20	20	98	2	4.9	0.24	0.06	0.03
Rand40	40	379	2	12.29	5.44	0.15	0.05
Rand100	100	2453	2	3456.69	2796.04	2.17	0.52

Table 3 Comparisons for bipartite graphs

Graphs	Vertices (N, M)	Edges $\|E\|$	K	BL1	BL2	BL3	BDQ
Rand10,10	(10,10)	48	2	0.4	0.22	0.22	0.11
			3	37	4.15	21.75	28.54
			4	373.87	33.51	371.53	1432.63
Rand5,15	(5,15)	40	2	0.38	0.16	0.21	0.1
			3	4.35	0.79	1.64	6.91
			5	0.02	0.05	0.03	0.01
Rand20,20	(20,20)	189	2	40.4	3.55	2.1	1.07

(Fig. 1) includes 37 connected components, and the inter-similarity among these components is 0. Our models for this graph with more than 600 edges works well, and by L2((13)) formulation, it took about 1300 s in CPLEX.

6.2.1 Graph $G = (V, E)$

In Table 2, discrete quadratic and three linear programming approaches are compared. The graphs are randomly generated in C++ with all weights being nonnegative. The gap is default in CPLEX and all approaches obtain the same optimal results. Under the cases of these graphs, the discrete quadratic formulation has the fast computation and the linear formulation (15) is the fastest one among three linear formulations though it has the most variables.

6.2.2 Bipartite graph $G = (V, U, E)$

For the case of bipartite graphs, the matrix A is also randomly generated with weights being nonnegative. The gap is default in CPLEX and all approaches obtain the same optimal results. Table 3 shows the time comparisons for discrete quadratic programming formulation BDQ(11) and three linear programming formulations, BL1(21), BL2(22) and BL3(23). The results show that the linear program (22), which has the fewest variables among three linear programs, is the most efficient one, and it is also better than quadratic formulations for these generated graphs.

7 Conclusion

In this paper, a zero-one quadratic programming formulation is used to model the general graph partitioning problem, and continuous quadratic programming formulation is also used. Two linear programming approaches can be derived from the zero-one quadratic programming, and another linear approach with introducing some variables in a different way is also presented. Several cases, such as determination of the number and cardinalities of subsets, hierarchy partitioning and partitioning of bipartite graphs without reordering one vertex set, are proposed. We have implemented the algorithms of these formulations for different graphs or networks and compared the time complexity.

Several formulations and their relaxations to continuous forms may work for different graphs, and choosing which of them to use for a given graph is still in discussion. The time series data $A = (a_{it})_{N\times T}$ is always random, such as stock market, and the stochastic programming may work in this situation. In addition, for the computational complexity of large graphs, the parallel computing may be useful and algorithms should be designed in such occasion for graph partitioning.

References

1. Alpert, C.J., Kahng, A.B.: Recent directions in netlist partitioning: a survey. Integr. VLSI J. **19**(1–2), 1–81 (1995)
2. Boulle, M.: Compact mathematical formulation for graph partitioning. Optim. Eng. **5**, 315–333 (2004)
3. Burer, S., Monteiro, R.D.C., Zhang, Y.: Rank-two relaxation heuristics for MAX-CUT and other binary quadratic programs. SIAM J. Optim. **12**, 503–521 (2001)
4. Chaovalitwongse, W., Pardalos, P.M., Prokopyev, O.A.: A new linearization technique for multi-quadratic 0–1 programming problems. Oper. Res. Lett. **32**, 517–522 (2004)
5. Chardaire, P., Sutter, A.: A decomposition method for quadratic zero-one programming. Manag. Sci. **41**(4), 704–712 (1995)
6. ILOG CPLEX 11.0 Users Manual, (2007)
7. Elsner, U.: Graph partitioning—A survey. Technische Universitat Chemnitz, SFB393-Preprint 97–27 (1997)
8. Fan, N., Chinchuluun, A., Pardalos, P.M. : Integer programming of biclustering based on graph models. In: Chinchuluun, A., Pardalos, P.M., Enkhbat, R., Tseveendorj, I. (eds.) Optimization and Optimal Control: Theory and Applications, Springer, Berlin (2009)
9. Fan, N., Pardalos, P.M., Chinchuluun, A., Pistikopoulos, E.N.: Graph partitioning approaches for analyzing biological networks. In: BIOMAT 2009—International Symposium on Mathematical and Computational Biology, submitted (2009)
10. Fan, N., Pardalos, P.M.: Robust optimization of graph partitioning involving uncertainty, In preparation (2009)
11. Fjallstrom, P.-O.: Algorithms for graph partitioning: a survey. Linkop. Elec. Articles Comput. Inf. Sci. **3**, 10 (1998)
12. Garey, M.R., Johnson, D.S., Stockmeyer, L.: Some simplified NP-complete graph problems. Theor. Comput. Sci. **1**, 237–267 (1976)
13. Hager, W., Krylyuk, Y.: Graph partitioning and continuous quadratic programming. SIAM J. Discret. Math. **12**(4), 500–523 (1999)
14. Hager, W., Krylyuk, Y.: Multiset graph partitioning. Math. Meth. Oper. Res. **55**, 1–10 (2002)
15. Karypis, G., Kumar, V.: Parallel multilevel graph partitioning. In: 10th International Parallel Processing Symposium (IPPS ’96), 314 (1996)
16. Karypis, G., Aggarwal, R., Kumar, V., Shekhar, S.: Multilevel hypergraph partitioning: applications in VLSI domain. IEEE Trans. Very Larg. Scale Integr. (VLSI) Syst. **7**(1), 69–79 (1999)
17. Karisch, S.E., Rendl, F. : Semidefinite programming and graph equipartition. In: Pardalos, P.M., Wolkowicz, H. (eds.) Topics in Semidefinite and Interior-Point Methods, pp. 77–95. American Mathematical Society, USA (1998)
18. Kernighan, B.W., Lin, S.: An Efficient Heuristic Procedure for Partitioning Graphs. Bell Syst. Tech. J. **49**(1), 291–307 (1970)

19. Kochenberger, G.A., Glover, F., Alidaee, B., Rego, C.: An unconstrained quadratic binary programming approach to the vertex coloring problem. Ann. Oper. Res. **139**(1), 229–241 (2005)
20. Lisser, A., Rendl, F.: Graph partitioning using linear and semidefinite programming. Math. Program. Ser. B **95**, 91–101 (2003)
21. Lodi, A., Allemand, K., Liebling, T.M.: An evolutionary heuristic for quadratic 0–1 programming. Eur. J. Oper. Res. **119**(3), 662–670 (1999)
22. Loiola, E.M., de Abreu, N.M.M., Boaventura-Netto, P.O., Hahn, P., Querido, T.: A survey for the quadratic assignment problem. Eur. J. Oper. Res. **176**(2), 657–690 (2007)
23. Pardalos, P.M., Rodgers, G.P.: Computational aspects of a branch and bound algorithm for quadratic zero-one programming. Computing **45**, 131–144 (1990)
24. Schaeffer, S.E.: Survey: graph clustering. Comput. Sci. Rev. **1**, 27–64 (2007)
25. Sherali, H.D., Smith, J.C.: An improved linearization strategy for zero-one quadratic programming problems. Optim. Lett. **1**, 33–47 (2007)
26. Strongin, R.G., Sergeyev, Ya, D.: Global multidimensional optimization on parallel computer. Parallel Comput. **18**, 1259–1273 (1992)
27. Strongin, R.G., Sergeyev, Ya, D.: Global optimization: fractal approach and non-redundant parallelism. J. Glob. Optim. **27**(1), 25–50 (2003)
28. Wolkowicz, H., Zhao, Q.: Semidefinite programming relaxations for the graph partitioning problem. Discret. Appl. Math. **96–97**, 461–479 (1996)
29. Zha, H., He, X., Ding, C., Simon, H., Gu, M.: Bipartite graph partitioning and data clustering. In: CIKM '01: Proceedings of the Tenth International Conference on Information and Knowledge Management, pp. 25–32. ACM Press, New York, NY (2001)