


A nonconvex quadratic optimization approach to the maximum edge weight clique problem

Syedmohammadhossein Hosseinian¹ ·
Dalila B. M. M. Fontes² · Sergiy Butenko¹ 

Received: 27 September 2017 / Accepted: 23 February 2018 / Published online: 7 March 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract The maximum edge weight clique (MEWC) problem, defined on a simple edge-weighted graph, is to find a subset of vertices inducing a complete subgraph with the maximum total sum of edge weights. We propose a quadratic optimization formulation for the MEWC problem and study characteristics of this formulation in terms of local and global optimality. We establish the correspondence between local maxima of the proposed formulation and maximal cliques of the underlying graph, implying that the characteristic vector of a MEWC in the graph is a global optimizer of the continuous problem. In addition, we present an exact algorithm to solve the MEWC problem. The algorithm is a combinatorial branch-and-bound procedure that takes advantage of a new upper bound as well as an efficient construction heuristic based on the proposed quadratic formulation. Results of computational experiments on some benchmark instances are also presented.

Keywords Maximum edge weight clique problem · Quadratic optimization · Continuous approaches to discrete problems

1 Introduction

Let $G = (V, E)$ be a simple, undirected graph, where V is the set of vertices and E is the set of edges. Given a subset of vertices $C \subseteq V$, let $G[C] = (C, E(C))$ denote the subgraph induced by C , where $E(C)$ is the set of edges with both endpoints in C . We call C a *clique* if

✉ Sergiy Butenko
butenko@tamu.edu

Syedmohammadhossein Hosseinian
hosseinian@tamu.edu

Dalila B. M. M. Fontes
dfontes@inesctec.pt

¹ Texas A&M University, College Station, TX, USA

² INESC TEC, Universidade do Porto, Rua Dr. Roberto Frias, 4200-464 Porto, Portugal

$G[C]$ is a complete graph. A clique is *maximal* if it is not a subset of any larger clique, and *maximum* if there is no larger clique in the graph. The *maximum clique problem* is to find a clique of maximum cardinality in G , which is called the *clique number* of G and is denoted by $\omega(G)$.

The maximum edge weight clique (MEWC) problem is a generalization of the classical maximum clique problem to edge-weighted graphs, which seeks a clique C with the maximum total weight of edges in the corresponding induced subgraph $G[C]$. More formally, given a weight w_{ij} associated with each edge $\{i, j\} \in E$, the weight of a clique C is defined as $W(C) = \sum_{\{i,j\} \in E(C)} w_{ij}$. The MEWC problem then seeks a clique C in G that maximizes $W(C)$.

If all edge weights of G are equal to 1, finding a maximum edge weight clique is equivalent to finding a maximum clique C^* in G with $W(C^*) = \binom{\omega(G)}{2}$. Therefore, the MEWC problem is at least as difficult to solve as the maximum clique problem, which is known to be \mathcal{NP} -hard [29]. The MEWC problem has a wide range of applications, including computer vision and pattern recognition [30,39], bioinformatics [2,9], retail industry [14], and health care [28].

Most of the works on the edge-weighted cliques in the literature deal with complete input graphs and look for a maximum edge weight clique of a cardinality not exceeding a given bound. The corresponding problem is referred to as the *maximum diversity problem* and is formally stated as follows: Given a complete (undirected) edge-weighted graph $G = (V, E)$, find a subset of vertices of cardinality at most $k \leq |V|$ with the maximum total weight of inter-connecting edges. An instance of the MEWC problem can be transformed into an instance of the maximum diversity problem by adding dummy edges with sufficiently large negative weights, and then solving for $k = |V|$. The exact solution methods proposed for this problem mainly involve branch-and-cut algorithms based on integer (linear) programming formulations; see for example [16,27,31,38,45]. Several heuristic and metaheuristic methods have also been applied to the maximum diversity problem, including tabu search [3,4,35], memetic search [48], scatter search [18], and greedy randomized adaptive search procedure [15,44]. Reviews of these methods can be found in [4,32,49].

The only previous works dealing with the MEWC problem we are aware of are [42] and [21], as well as our recent papers [25] and [26], the first of which is a preliminary workshop version of the present work and the second—a survey paper focusing on mathematical optimization formulations and existing solution approaches. In particular, Pullan [42] developed a phased local search heuristic, and Gouveia and Martins [21] recently presented a set of integer programming formulations by introducing new valid inequalities to classical formulations based on sparseness of the graph. In this paper, we also present an exact method to solve this problem. We introduce a quadratic formulation for the MEWC problem. Then, we use a relaxation of the new formulation to draw an upper bound for this problem and use this bound within a combinatorial branch-and-bound procedure. In addition, our method benefits from an efficient construction heuristic algorithm which is derived from the proposed formulation.

Our approach is based on a continuous characterization of the MEWC problem. A connection between cliques and a continuous optimization problem was first established by Motzkin and Straus [33], who showed a correspondence between maximum cliques in a graph and optima of a certain standard quadratic program. Pardalos and Phillips [37] were the first to use this connection in developing a global optimization algorithm for the maximum clique problem. Later, the Motzkin–Straus formulation and its generalizations have been extensively studied in [7,8,10,11,19,20,40,41]. In particular, Bomze [7] introduced its regularized version, which ensures one-to-one correspondence between local maxima of the

quadratic program and maximal cliques of the underlying graph. In a similar manner, this work introduces a quadratic programming formulation for the MEWC problem, and presents the relation between the continuous problem and the underlying graph in terms of global and local optima. Unlike the Motzkin–Straus formulation, which has the standard simplex as its feasible region, our formulation maximizes a quadratic function over a unit hypercube. Formulations of the maximum independent set problem (which is equivalent to the maximum clique problem in the complement graph) as a problem of maximizing a nonlinear function over a unit hypercube have been previously considered in [1, 5, 12, 22–24].

This paper is organized as follows. Section 2 presents the quadratic programming formulation for the MEWC problem. Section 3 explores characteristics of this continuous problem in terms of local and global optimality, and establishes the corresponding structures in the underlying graph. Section 4 introduces the new algorithm to solve the MEWC problem. Section 5 presents results of computational experiments on some benchmark instances and Sect. 6 concludes this paper.

Throughout this paper, we assume G is a simple, proper graph and all edge weights of it are positive. G is a *proper* graph if it is not complete and does not have an isolated vertex. By focusing on proper graphs, we exclude some trivial cases with respect to the MEWC problem. We denote vectors by boldface lowercase, and matrices by boldface uppercase letters. $\mathbf{0}$ denotes the vector all zeros, and $\mathbf{1}$ is the vector all ones. All vectors are column vectors, and $\|\cdot\|$ denotes the Euclidean norm.

2 Quadratic programming formulation for the MEWC problem

We consider a simple, undirected, edge-weighted graph $G = (V, E)$ with $|V| = n$ and the edge weights given by w_{ij} , $\{i, j\} \in E$. For a vertex $i \in V$, let $N(i)$ denote the neighborhood of i in G , defined as $N(i) = \{j \in V \mid \{i, j\} \in E\}$. The closed neighborhood $N[i]$ of i is $N[i] = N(i) \cup \{i\}$. Let $W^*(G)$ denote the edge weight of a MEWC in G . Recall that a *characteristic vector* of a subset of vertices $C \subseteq V$ is an n -dimensional vector $\mathbf{x} \in \mathbb{R}^n$ such that $x_i = 1$, $\forall i \in C$, and $x_i = 0$, $\forall i \notin C$.

Proposition 1 *The MEWC problem on G can be formulated as the following quadratic program (QP):*

$$W^*(G) = \max_{\mathbf{x} \in [0,1]^n} \left(\sum_{\{i,j\} \in E} w_{ij} x_i x_j - \sum_{\{i,j\} \notin E} \bar{w}_{ij} x_i x_j \right), \quad (1)$$

where $\mathbf{x} = (x_1, x_2, \dots, x_n)$ is the vector of variables corresponding to vertices of G and \bar{w}_{ij} is defined as follows:

$$\bar{w}_{ij} = \max \left\{ \sum_{k \in N(i)} w_{ik}, \sum_{k \in N(j)} w_{jk} \right\} + \xi \quad \forall \{i, j\} \notin E, i \neq j \quad (2)$$

for an arbitrarily small $\xi > 0$. Any global optimal solution of this QP is the characteristic vector of a MEWC of G .

Let \mathbf{Q} be a square matrix of order n with the following structure:

$$\mathbf{Q}(i, j) = \begin{cases} 0, & i = j \\ w_{ij}, & \{i, j\} \in E \\ -\bar{w}_{ij}, & \{i, j\} \notin E, \end{cases} \quad (3)$$

then (1) is equivalent to

$$\max_{\mathbf{x} \in [0, 1]^n} f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x}. \quad (\mathbf{P})$$

Note that \mathbf{Q} is symmetric and it is always indefinite by definition of a proper graph.

Proposition 1 is easy to establish directly, but it will also follow from the results characterizing the connection between local maxima of (P) and maximal cliques in G in the next section. First, we present the standard optimality conditions for (P).

2.1 First order necessary conditions (FONC)

Let $\mathbf{x}^* \in [0, 1]^n$ be a local maximum point of (P). Then there are two non-negative vectors $\boldsymbol{\mu} \in \mathbb{R}^n$ and $\boldsymbol{\lambda} \in \mathbb{R}^n$ such that:

$$\mathbf{Q} \mathbf{x}^* = \boldsymbol{\lambda} - \boldsymbol{\mu} \quad (\boldsymbol{\lambda} \geq \mathbf{0}, \boldsymbol{\mu} \geq \mathbf{0}) \quad (4)$$

$$\mu_i x_i^* = 0 \quad \text{and} \quad \lambda_i (1 - x_i^*) = 0 \quad \forall i \in \{1, 2, \dots, n\}, \quad (5)$$

where μ_i , λ_i and x_i^* are the i th components of vectors $\boldsymbol{\mu}$, $\boldsymbol{\lambda}$ and \mathbf{x}^* , respectively. The condition (4) enforces dual feasibility, and (5) is the complementary slackness condition.

2.2 Second order necessary conditions (SONC)

Suppose \mathbf{x}^* is a local maximum point of (P). Let

$$Z(\mathbf{x}^*, \boldsymbol{\mu}, \boldsymbol{\lambda}) = \{i \mid (x_i^* = 0 \text{ and } \mu_i > 0) \text{ or } (x_i^* = 1 \text{ and } \lambda_i > 0)\}, \quad (6)$$

then, in addition to the first order necessary conditions, \mathbf{x}^* also satisfies

$$\mathbf{y}^T \mathbf{Q} \mathbf{y} \leq 0 \quad \forall \mathbf{y} \in \mathbb{R}^n \quad \text{s.t.} \quad y_i = 0 \quad \forall i \in Z(\mathbf{x}^*, \boldsymbol{\mu}, \boldsymbol{\lambda}). \quad (7)$$

2.3 Second order sufficient conditions (SOSC)

If a point \mathbf{x}^* satisfies FONC, SONC, and

$$\mathbf{y}^T \mathbf{Q} \mathbf{y} < 0 \quad \forall \mathbf{y} \in \mathbb{R}^n \setminus \{\mathbf{0}\} \quad \text{s.t.} \quad y_i = 0 \quad \forall i \in Z(\mathbf{x}^*, \boldsymbol{\mu}, \boldsymbol{\lambda}), \quad (8)$$

then \mathbf{x}^* is a strict local maximizer of (P).

3 Optimality characterizations

Lemma 1 Every local maximizer \mathbf{x}^* of (P) is a binary vector (i.e., $\mathbf{x}^* \in \{0, 1\}^n$) with at least two components equal to 1.

Proof Suppose there exists $i \in \{1, 2, \dots, n\}$ such that $0 < x_i^* < 1$. By (5), $\mu_i = \lambda_i = 0$ must hold for this solution. By (4), this implies that

$$\mathbf{Q}^i \mathbf{x}^* = 0, \quad (9)$$

where \mathbf{Q}^i denotes the i th row of matrix \mathbf{Q} . Regarding the structure of matrix \mathbf{Q} , Eq. (9) can be written as

$$\sum_{j \in N(i)} w_{ij} x_j^* - \sum_{k \notin N(i)} \bar{w}_{ik} x_k^* + 0 x_i^* = 0. \quad (10)$$

In order for \mathbf{x}^* to satisfy (10), exactly one of the following conditions must hold at this point:

$$x_j^* = x_k^* = 0 \quad \forall j, k \in \{1, 2, \dots, n\} \setminus \{i\}; \quad (11)$$

$$\exists j \in N(i) \text{ s.t. } x_j^* \neq 0 \text{ and } \exists k \notin N(i) \text{ s.t. } x_k^* \neq 0. \quad (12)$$

In the above notation, i , j , and k are distinct indices. Now we show that either of the aforementioned cases leads to a contradiction, implying that such a point \mathbf{x}^* does not exist. Consider the former case. Since \mathbf{x}^* is a local maximum point of (P), $f(\mathbf{x}^*) \geq f(\mathbf{x})$ for all feasible points $\mathbf{x} \in B_\epsilon(\mathbf{x}^*)$, where $B_\epsilon(\mathbf{x}^*) = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x} - \mathbf{x}^*\| \leq \epsilon\}$ denotes an ϵ -neighborhood of \mathbf{x}^* in \mathbb{R}^n . It is apparent that $f(\mathbf{x}^*) = 0$ because f is a homogeneous quadratic function. However, increasing the value of $x_j^* : j \in N(i)$ from 0 to ϵ (a single variable) results in a distinct feasible point $\hat{\mathbf{x}} \in B_\epsilon(\mathbf{x}^*)$ with $f(\hat{\mathbf{x}}) = w_{ij} x_j^* \epsilon > 0$ which contradicts the local optimality of \mathbf{x}^* . Note that by the problem assumption (i.e., there is no isolated vertex in G) it is guaranteed that such a vertex $j \in N(i)$ exists.

In the latter case, we use SONC to draw a contradiction. Let Y denote the set of all vectors \mathbf{y} that are considered for SONC at the local maximum point \mathbf{x}^* . That is,

$$Y = \{\mathbf{y} \mid \mathbf{y} \in \mathbb{R}^n : y_i = 0 \quad \forall i \in Z(\mathbf{x}^*, \boldsymbol{\mu}, \boldsymbol{\lambda})\}. \quad (13)$$

Consider the vector $\tilde{\mathbf{y}}$, only two components of which, \tilde{y}_i and \tilde{y}_k , are non-zero. Then, $\tilde{\mathbf{y}} \in Y$ if $x_i^*, x_k^* \neq 0$ and $x_i^*, x_k^* \neq 1$. These conditions on x_i^* are already met by assumption (i.e., $0 < x_i^* < 1$). We pick vertex $k \notin N(i)$ such that $x_k^* \neq 0$. Note that, existence of such a vertex k is assumed under this case. Besides, $x_k^* < 1$ is enforced by the magnitude of \bar{w}_{ik} and (10). We assign $\tilde{y}_i = 1$ and $\tilde{y}_k = -1$. By this assignment, $\tilde{\mathbf{y}} \in Y$ and we have

$$\tilde{\mathbf{y}}^T \mathbf{Q} \tilde{\mathbf{y}} = 2(-\bar{w}_{ik})(1)(-1) = 2\bar{w}_{ik} > 0. \quad (14)$$

Equation (14) indicates violation of SONC, which contradicts the local optimality of \mathbf{x}^* . This proves that each local maximizer is a binary vector.

Since every variable in (P) corresponds to a vertex in the graph, every local maximum point of (P) is the characteristic vector of a subset of vertices in G . Next, we show that at least two components are equal to 1 in any local maximizer \mathbf{x}^* of (P). Since f is a homogeneous quadratic function, $f(\mathbf{x}^*) = 0$ if less than two coordinates of \mathbf{x}^* are nonzero. For $\mathbf{0}$ as the characteristic vector of an empty set of vertices, observe that increasing the values of two variables x_i and x_j such that $\{i, j\} \in E$ from 0 to $\epsilon/\sqrt{2}$ results in a new point $\hat{\mathbf{x}} \in B_\epsilon(\mathbf{0})$ with $f(\hat{\mathbf{x}}) = \frac{1}{2}w_{ij}\epsilon^2 > 0$. Similarly, for the characteristic vector of a single-vertex set $\{i\}$, increasing the variable of an adjacent vertex j from 0 to ϵ increases the function value to $w_{ij}\epsilon > 0$. This completes the proof. \square

Theorem 1 \mathbf{x}^* is a local maximizer of (P) if and only if it is the characteristic vector of a maximal clique in G .

Proof First we show that if \mathbf{x}^* is a local maximizer of **(P)**, then it is characteristic vector of a maximal clique in G . By Lemma 1 all components of \mathbf{x}^* have to be 0 or 1. Let $V = S \cup T$ be a partition of vertices with respect to the components of \mathbf{x}^* as follows:

$$S = \{j \in V \mid x_j^* = 0\} \quad \text{and} \quad T = \{i \in V \mid x_i^* = 1\}. \quad (15)$$

Note that a proper graph has at least three vertices and we have already shown that cardinality of T is at least two.

By (5),

$$\mu_i = 0 \quad \text{and} \quad \lambda_i \geq 0 \quad \forall i \in T \quad (16)$$

$$\lambda_j = 0 \quad \text{and} \quad \mu_j \geq 0 \quad \forall j \in S. \quad (17)$$

Equations (16)–(17) along with (4) imply that

$$\mathbf{Q}^i \mathbf{x}^* = \lambda_i \geq 0 \quad \forall i \in T \quad (18)$$

$$\mathbf{Q}^j \mathbf{x}^* = -\mu_j \leq 0 \quad \forall j \in S, \quad (19)$$

where \mathbf{Q}^i and \mathbf{Q}^j denote rows of matrix \mathbf{Q} corresponding to vertices $i \in T$ and $j \in S$, respectively.

Taking into account the structure of matrix \mathbf{Q} and magnitude of its components, (18) implies that for every vertex $i \in T$, all vertices that are not adjacent to i must have variables equal to zero. That is,

$$i \in T, k \notin N[i] \Rightarrow k \in S, \quad (20)$$

which implies that T is a clique. Similarly, (19) implies that for every vertex $j \in S$, there must exist at least one vertex i with $x_i^* = 1$ such that i and j are not adjacent in G . That is,

$$\forall j \in S \exists i \in T : \{i, j\} \notin E, \quad (21)$$

so T is a maximal clique.

Now we show that if \mathbf{x}^* is the characteristic vector of a maximal clique C in G , then it is a local maximizer of **(P)**. It is obvious that \mathbf{x}^* is a feasible point of **(P)**. We start by showing that \mathbf{x}^* satisfies FONC and SONC. Let

$$\mu_i = 0 \quad \text{and} \quad \lambda_i = \mathbf{Q}^i \mathbf{x}^* \quad \forall i \in C \quad (22)$$

$$\lambda_j = 0 \quad \text{and} \quad \mu_j = -\mathbf{Q}^j \mathbf{x}^* \quad \forall j \notin C. \quad (23)$$

This assignment satisfies (5). It also satisfies $\mathbf{Q}\mathbf{x}^* = \boldsymbol{\lambda} - \boldsymbol{\mu}$. Therefore, in order to prove \mathbf{x}^* is a stationary point, it suffices to show that

$$\lambda_i \geq 0 \quad \forall i \in C \quad \text{and} \quad \mu_j \geq 0 \quad \forall j \notin C. \quad (24)$$

Given the structure of matrix \mathbf{Q} and \mathbf{x}^* being the characteristic vector of C , we have

$$\mathbf{Q}^i \mathbf{x}^* = \sum_{k \in C \cap N(i)} w_{ik} - \sum_{l \in C \setminus N(i)} \bar{w}_{il} \quad \forall i \in C. \quad (25)$$

C being a maximal clique with at least two vertices implies that

$$\forall i \in C : C \setminus N(i) = \emptyset \quad \text{and} \quad C \cap N(i) \neq \emptyset. \quad (26)$$

Equations (22) and (25)–(26) imply that

$$\lambda_i = \sum_{k \in C \cap N(i)} w_{ik} > 0 \quad \forall i \in C. \quad (27)$$

Similarly,

$$\mathbf{Q}^j \mathbf{x}^* = \sum_{k \in C \cap N(j)} w_{jk} - \sum_{l \in C \setminus N(j)} \bar{w}_{jl} \quad \forall j \notin C, \quad (28)$$

and by maximality of C ,

$$C \setminus N(j) \neq \emptyset \quad \forall j \notin C. \quad (29)$$

Regarding the magnitude of \bar{w}_{jl} , (23) and (28)–(29) imply that

$$\mathbf{Q}^j \mathbf{x}^* < 0 \Rightarrow \mu_j = -\mathbf{Q}^j \mathbf{x}^* > 0 \quad \forall j \notin C. \quad (30)$$

Therefore, the assignment of λ and μ according to (22) and (23) satisfies FONC, so \mathbf{x}^* is a stationary point of (P). Furthermore, $Z(\mathbf{x}^*, \mu, \lambda) = V$, $Y = \{\mathbf{0}\}$, so SONC and SOSC are satisfied, implying that \mathbf{x}^* is a strict local maximum point of this problem. \square

Corollary 1 *A clique C is an optimal solution to the MEWC problem if and only if its characteristic vector is a global maximizer of (P).*

Proof The feasible region of (P) is a compact set, so $f(\mathbf{x})$ attains its global maximum in a local maximizer. By Theorem 1, every local maximizer of (P) is the characteristic vector of a maximal clique in G . The objective value of a local maximizer \mathbf{x}_C of (P), corresponding to a maximal clique C in G , is

$$f(\mathbf{x}_C) = \sum_{\{i,j\} \in E} w_{ij} x_i x_j - \sum_{\{i,j\} \notin E} \bar{w}_{ij} x_i x_j = \sum_{\{i,j\} \in E(C)} w_{ij} = W(C), \quad (31)$$

where $E(C)$ is the set of edges of the induced subgraph $G[C]$. Therefore, a global maximizer of (P) is the characteristic vector of a maximal clique in G with maximum total weight. \square

As stated previously, the maximum clique problem can be considered as a special case of the MEWC problem. Hence, the aforementioned results hold for this problem too. This is presented through the following corollary.

Corollary 2 (The Maximum Clique Problem) *The clique number $\omega(G)$ of a proper undirected and unweighted graph $G = (V, E)$ can be found through solving the following quadratic programming problem:*

$$\binom{\omega(G)}{2} = \max_{\mathbf{x} \in [0,1]^n} \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} \quad (32)$$

where matrix \mathbf{Q} is constructed according to (3), with $w_{ij} = 1$, $\forall \{i, j\} \in E$, and $\bar{w}_{ij} = \max\{\deg(i), \deg(j)\} + 1$, $\forall \{i, j\} \notin E$.

4 Solving the MEWC problem

The results in the last section establish the relation between (P) and the MEWC problem in terms of optimality characteristics. We use these results to develop a method for solving the MEWC problem. In this section, we first present a construction heuristic that is derived from a polynomial-time solvable approximation of (P). Then, we introduce an algebraic upper bound for this problem based on solving a quadratic relaxation of (P). Finally, we present our solution method, which is a combinatorial branch-and-bound (B&B) procedure that uses an initial lower bound provided by the heuristic algorithm, and applies the algebraic upper bound to prune the search tree.

4.1 Construction heuristic

(P) concerns maximization of a non-convex quadratic function subject to a set of linear constraints, which is \mathcal{NP} -hard [36]. However, quadratic optimization subject to an ellipsoid constraint is polynomial-time solvable [50]. We approximate (P) by replacing its unit hypercube constraint with a unit hypersphere and examining stationary points of the new problem. Similar approaches have been successfully exploited before by replacing the standard simplex in the Motzkin–Straus formulation [11, 19] or the unit hypercube in a QP formulation over a box [12] with an ellipsoid constraint to develop a heuristic algorithm for the maximum clique/independent set problem.

The new problem is obtained as follows. Assume that a MEWC consists of k vertices. Then $\mathbf{1}^T \mathbf{x}^* = k$, so adding this constraint and changing the variables by dividing each variable by k , we obtain a problem equivalent to (P),

$$\begin{aligned} \max_{\mathbf{x} \in \mathbb{R}^n} \quad & f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} \\ \text{s.t.} \quad & \mathbf{x} \in [0, 1/k]^n \\ & \mathbf{1}^T \mathbf{x} = 1. \end{aligned} \quad (33)$$

Finally, the feasible region of (33) is approximated with a unit hypersphere:

$$\begin{aligned} \max_{\mathbf{x} \in \mathbb{R}^n} \quad & f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} \\ \text{s.t.} \quad & \mathbf{x}^T \mathbf{x} = 1. \end{aligned} \quad (34)$$

Suppose \mathbf{x}^* is a local optimizer of (34). Then, the first order optimality conditions indicate

$$\mathbf{Q} \mathbf{x}^* = 2 \mu \mathbf{x}^* \quad (35)$$

$$\|\mathbf{x}^*\|^2 = 1, \quad (36)$$

where μ is the Lagrangian multiplier of the constraint.

Equations (35) and (36) imply that every local optimizer \mathbf{x}^* of (34) is a normalized eigenvector of matrix \mathbf{Q} corresponding to an eigenvalue $\lambda = 2\mu$. A heuristic solution to the MEWC problem is constructed by considering all eigenvectors of matrix \mathbf{Q} , each treated as an approximation of the characteristic vector of a subset of vertices in G . Algorithm 1 presents the corresponding procedure.

The QCH algorithm operates as follows: first, it calculates eigenvectors of matrix \mathbf{Q} . Each eigenvector \mathbf{x}_e is treated as an approximation of the characteristic vector of a maximal clique. Mapping from a real-valued vector \mathbf{x}_e to a maximal clique is done through the EXTRACT-CLIQUE function, demonstrated in Algorithm 2. This function sorts the vertices based on their values in vector \mathbf{x}_e . A clique is initiated by including the vertex corresponding to the first element of the sorted list, and is expanded by appending other vertices according to their orders in the list. To make sure the mapping output is indeed a clique, a vertex is added to the clique only if it is adjacent to all vertices that are already in the clique. Sorting components of \mathbf{x}_e in a decreasing (non-increasing) order follows the idea that the closer a value is to 1, the higher priority its vertex has to be included in the incumbent clique. Since \mathbf{x}_e components have signed values, an analogous mapping can be considered by giving priority to vertices whose values in \mathbf{x}_e are close to -1 . This mapping corresponds to sorting the \mathbf{x}_e components in an increasing (non-decreasing) order. The QCH algorithm examines both mappings by calling the EXTRACTCLIQUE function twice with different ordering arguments. The algorithm outputs a clique with the highest weight among all extracted cliques.

Algorithm 1 Quadratic-Based Construction Heuristic (QCH)

```

1: function QCH( $G$ )
2:    $\tilde{C} = \{\}$  ;  $\tilde{W} = 0$ 
3:   construct matrix  $Q$  according to (3)
4:   for each eigenvector  $\mathbf{x}_e$  of  $Q$  do
5:      $C_d = \text{EXTRACTCLIQUE}(\mathbf{x}_e, \text{decreasing})$ 
6:     if  $W(C_d) > \tilde{W}$  then  $\triangleright W(C_d)$ : weight of the clique  $C_d$ 
7:        $\tilde{C} \leftarrow C_d$  ;  $\tilde{W} \leftarrow W(C_d)$ 
8:     end if
9:      $C_i = \text{EXTRACTCLIQUE}(\mathbf{x}_e, \text{increasing})$ 
10:    if  $W(C_i) > \tilde{W}$  then  $\triangleright W(C_i)$ : weight of the clique  $C_i$ 
11:       $\tilde{C} \leftarrow C_i$  ;  $\tilde{W} \leftarrow W(C_i)$ 
12:    end if
13:  end for
14:  return  $(\tilde{C}, \tilde{W})$ 
15: end function

```

Algorithm 2 Clique extraction

```

1: function EXTRACTCLIQUE( $\mathbf{x}, \text{order}$ )  $\triangleright \text{order: decreasing or increasing}$ 
2:   sort components of  $\mathbf{x}$  according to  $\text{order}$   $\triangleright$  sorted array:  $[x^{(1)}, x^{(2)}, \dots, x^{(n)}]$ 
3:    $C = \{v[x^{(1)}]\}$   $\triangleright v[x^{(1)}]$ : vertex corresponding to  $x^{(1)}$ 
4:   for  $j = 2$  to  $n$  do
5:     if  $v[x^{(j)}] \in \bigcap_{i \in C} N(i)$  then
6:        $C \leftarrow C \cup \{v[x^{(j)}]\}$ 
7:     end if
8:   end for
9:   return  $C$ 
10: end function

```

Algorithm 3 QCH-N, QCH used for closed neighborhoods

```

1: function QCH-N( $G$ )
2:    $\tilde{C} = \{\}$  ;  $\tilde{W} = 0$ 
3:   for each vertex  $i \in V$  do
4:     run QCH on  $G[N[i]]$  to obtain  $\tilde{C}_i$  and  $W(\tilde{C}_i)$ 
5:     if  $W(\tilde{C}_i) > \tilde{W}$  then
6:        $\tilde{C} \leftarrow \tilde{C}_i$  ;  $\tilde{W} \leftarrow W(\tilde{C}_i)$ 
7:     end if
8:   end for
9:   return  $(\tilde{C}, \tilde{W})$ 
10: end function

```

The heuristic results can potentially improve when the QCH algorithm is implemented for the closed neighborhood of each vertex. That is, if a vertex i was known to be contained in the optimal solution, then the problem would be simplified to finding a maximum edge weight clique in $G[N[i]]$, the subgraph induced by i and its neighbors. While it takes longer to examine all such subgraphs, the heuristic algorithm is likely to generate better solutions. Algorithm 3 demonstrates utilizing the QCH algorithm in this setting. The output of the QCH (or QCH- N) algorithm provides an initial lower bound for the combinatorial B&B procedure. We show the quality of these solutions in Sect. 5 through computational experiments.

4.2 Quadratic relaxation bound

As it will be shown in detail in the next section, the combinatorial B&B algorithm traverses a search tree to find a clique with the maximum edge weight in the graph. At every node of the tree, it considers a subgraph induced by the union of an incumbent clique C and a list of vertices L , called *candidate list*. The candidate list is composed of the vertices in $V \setminus C$ that are adjacent to all vertices in C , so appending each of them to C would result in a larger and heavier clique. A node of the tree is pruned (i.e., the corresponding subgraph is excluded from further investigation) if an upper bound on the maximum clique-weight in the subgraph is not greater than a known global lower bound. In this section, we present a new method to calculate such upper bounds based on solving a relaxation of (P).

Consider $G' = G[C \cup L]$, the subgraph induced by the union of a clique C and its candidate list L . Let \mathbf{Q}' be the corresponding matrix to G' as defined in Sect. 2. We seek an upper bound on $W_{G'}^*$, the maximum weight of a clique in G' . By Proposition 1,

$$W_{G'}^* = \max_{\mathbf{x} \in \{0,1\}^{n'}} \frac{1}{2} \mathbf{x}^T \mathbf{Q}' \mathbf{x}, \quad (37)$$

where $n' = |C \cup L|$. As every vertex in L is adjacent to all vertices of C , any maximal clique in G' must contain C and by Theorem 1,

$$x_i = 1 \quad \forall i \in C \quad (38)$$

in an optimal solution of (37). Therefore, the objective function of (37) can be expanded as follows:

$$f(\mathbf{x}) = W(C) + \sum_{i \in L} x_i \left(\sum_{j \in C} w_{ij} \right) + \sum_{i \in L} \sum_{j \in L} \mathbf{Q}'(i, j) x_i x_j. \quad (39)$$

Results of the last section also indicate that the rest of the variables in the optimal solution are at their bounds. That is,

$$x_i \in \{0, 1\} \quad \forall i \in L. \quad (40)$$

Let \mathbf{q} denote the vector of coefficients in the second term of (39), then (37) can be written as

$$W_{G'}^* = W(C) + \max_{\mathbf{x} \in \{0,1\}^{|L|}} \left(\mathbf{q}^T \mathbf{x} + \frac{1}{2} \mathbf{x}^T \mathbf{Q}'_L \mathbf{x} \right), \quad (41)$$

where \mathbf{Q}'_L is the principal submatrix of \mathbf{Q}' corresponding to vertices of L . Note that \mathbf{Q}'_L is not constructed directly from the subgraph induced by L because the negative components of this matrix should account for the edges connecting vertices of L and C as well as interconnecting edges of L .

We calculate an upper bound on $W_{G'}^*$ through the following formula:

$$\mathcal{Z}_{G'} = W(C) + \left(\begin{array}{l} \mathcal{Z}_L = \max_{\mathbf{x} \in \mathbb{R}^{|L|}} \quad \mathbf{q}^T \mathbf{x} + \frac{1}{2} \mathbf{x}^T \mathbf{Q}'_L \mathbf{x} \\ \text{s.t.} \quad (\mathbf{x} - \mathbf{b})^T (\mathbf{x} - \mathbf{b}) = \frac{|L|}{4} \end{array} \right), \quad (42)$$

where $\mathbf{b} \in \mathbb{R}^{|L|}$ is a vector with all components equal to $\frac{1}{2}$. It is clear that the feasible region of the quadratic optimization problem in (42) contains all 0–1 vectors of size $|L|$, so it is a relaxation of (41) and $\mathcal{Z}_{G'} \geq W_{G'}^*$. Note that even if G' is not proper (i.e., L is a clique), (41) remains valid and $\mathcal{Z}_{G'}$ will be an upper bound on $W_{G'}^*$.

Let $\mathbf{y} = \mathbf{x} - \mathbf{b}$ and $\mathbf{d} = \mathbf{Q}'_L \mathbf{1} + 2\mathbf{q}$. Then,

$$\mathcal{Z}_L = \frac{1}{2} \left(\max_{\mathbf{y}^T \mathbf{y} = |L|/4} \mathbf{d}^T \mathbf{y} + \mathbf{y}^T \mathbf{Q}'_L \mathbf{y} \right) + \text{const.} \quad (43)$$

By eigen-decomposition of matrix \mathbf{Q}'_L and a linear transformation to the eigenvector space, (43) can be written as

$$\mathcal{Z}_L = \frac{1}{2} \left(\max_{\mathbf{y}'^T \mathbf{y}' = |L|/4} \mathbf{d}'^T \mathbf{y}' + \mathbf{y}'^T \mathbf{\Lambda} \mathbf{y}' \right) + \text{const.} \quad (44)$$

In (44), $\mathbf{\Lambda}$ is a diagonal matrix composed of eigenvalues of \mathbf{Q}'_L and $\mathbf{y}' = \mathbf{E}^T \mathbf{y}$ and $\mathbf{d}' = \mathbf{E}^T \mathbf{d}$, where \mathbf{E} is a square matrix whose columns are orthonormal eigenvectors of \mathbf{Q}'_L with the same order as the corresponding eigenvalues in $\mathbf{\Lambda}$. Let μ denote the Lagrangian multiplier of the single hypersphere constraint in (44). Then, by the first order optimality conditions, every local maximizer of (44) is a solution to the following system of equations:

$$2(\mathbf{\Lambda} - \mu \mathbf{I})\mathbf{y}' + \mathbf{d}' = 0 \quad (45)$$

$$\|\mathbf{y}'\|^2 = \frac{|L|}{4} \quad (46)$$

The set of all μ for which there exists a vector \mathbf{y}' such that (μ, \mathbf{y}') satisfies (45)–(46) is called the *spectrum* of this system.

Forsythe and Golub [17] have studied optimization of a quadratic function subject to a single hypersphere constraint in detail. In particular, they have shown that the global maximizer of this problem corresponds to the greatest μ in the spectrum of (45)–(46). Let $\{\lambda_1, \lambda_2, \dots, \lambda_{|L|}\}$ be the set of eigenvalues of \mathbf{Q}'_L (possibly with multiplicity) and λ^{\max} denote the maximum eigenvalue of this matrix. If there exists $i \in \{1, \dots, |L|\}$ such that $\lambda_i = \lambda^{\max}$ and $d'_i \neq 0$, then the greatest μ in the spectrum of (45)–(46) is given by the greatest root of the following univariate function, which we denote by μ^* (i.e., $\mu^* = \max\{\mu \mid \Phi(\mu) = 0\}$):

$$\Phi(\mu) = \sum_{i \in \mathcal{I}} \frac{(\lambda_i b'_i + q'_i)^2}{(\lambda_i - \mu)^2} - \frac{|L|}{4}, \quad \mathcal{I} = \{i \mid \lambda_i b'_i + q'_i \neq 0\}, \quad (47)$$

where b'_i and q'_i are the i th components of vectors $\mathbf{b}' = \mathbf{E}^T \mathbf{b}$ and $\mathbf{q}' = \mathbf{E}^T \mathbf{q}$, respectively. It is easy to see that $d'_i = \lambda_i b'_i + q'_i$ and, under the above condition, μ^* always exists and belongs to the interval $(\lambda^{\max}, +\infty)$. In this case, \mathcal{Z}_L can be directly calculated as follows:

$$\mathcal{Z}_L = \sum_{i=1}^{|L|} \frac{(\mu^* b'_i + q'_i) [\lambda_i (\mu^* b'_i + q'_i) + 2\mu^* q'_i]}{2(\lambda_i - \mu^*)^2}. \quad (48)$$

Note that $\Phi(\mu)$ is monotonically decreasing in the interval $(\lambda^{\max}, +\infty)$, so μ^* can be calculated by line search methods to a desired precision.

But if $d'_i = 0$ for all $\lambda_i = \lambda^{\max}$, then μ^* does not necessarily have the maximum value in the spectrum of (45)–(46). In this case, any eigenvalue $\lambda > \mu^*$ that satisfies (49) and (50) will generate a better objective value for this problem.

$$i \in \bar{\mathcal{I}} \quad \forall \lambda_i = \lambda \quad (49)$$

$$\sum_{i \notin \mathcal{I}_\lambda} \frac{(\lambda_i b'_i + q'_i)^2}{(\lambda_i - \lambda)^2} < \frac{|L|}{4}, \quad (50)$$

where $\bar{\mathcal{I}} = \{i \mid \lambda_i b'_i + q'_i = 0\}$ and $\mathcal{J}_\lambda = \{i \in \bar{\mathcal{I}} \mid \lambda_i = \lambda\}$. Let λ^* denote the largest eigenvalue of \mathbf{Q}'_L that satisfies these conditions. Then, \mathcal{Z}_L is given by

$$\mathcal{Z}_L = \frac{1}{2} \sum_{i=1}^{|L|} \lambda_i (y_i + b'_i)^2 + q'_i (y_i + b'_i), \quad (51)$$

where

$$y_i = \begin{cases} \frac{-(\lambda_i b'_i + q'_i)}{\lambda_i - \lambda^*} & i \notin \mathcal{J}_{\lambda^*} \\ \sqrt{\alpha} & i = j \text{ for a single } j \in \mathcal{J}_{\lambda^*} : \alpha = \frac{|L|}{4} - \sum_{i \notin \mathcal{J}_{\lambda^*}} y_i^2 \\ 0 & i \in \mathcal{J}_{\lambda^*} \setminus \{j\}. \end{cases} \quad (52)$$

We use these results to calculate an upper bound for the MEWC problem on the subgraphs visited throughout the combinatorial B&B process.

4.3 Combinatorial branch-and-bound procedure

Our algorithm utilizes a classical combinatorial method for implicit enumeration of maximal cliques in a graph. This framework has been vastly used in the algorithms proposed for the maximum clique problem with different pruning criteria and implementation details; see for example [6, 13, 34, 43, 46, 47]. We adapt this method for the MEWC problem by applying the upper bound of the last section, and present a combinatorial B&B algorithm for this problem. Algorithm 4 demonstrates the outline of our method.

Algorithm 4 Combinatorial B&B algorithm w/ quadratic relaxation bound

```

1: function CBQ( $G, \tilde{C}, \tilde{W}$ ) ▷  $\tilde{C}$ : initial solution,  $\tilde{W}$ : initial lower bound
2:    $C = \{\}$ ;  $W = 0$ 
3:    $C^* = \tilde{C}$ ;  $W^* = \tilde{W}$ 
4:    $L = \text{INITIALIZATION}(G)$ 
5:    $\text{BRANCH}(G, L, C, C^*, W, W^*)$ 
6:   return  $\langle C^*, W^* \rangle$ 
7: end function

```

The CBQ algorithm starts with making a sorted array of vertices L by calling the INITIALIZATION function. The order of vertices in this array determines the sequence of visiting cliques in the graph. This initial ordering aims to keep the subproblems formed by the BRANCH function small. BRANCH is a recursive function that examines maximal cliques in a systematic manner. All input arguments of this function are global and updated through the search process.

The INITIALIZATION function stores the vertices in L such that $L(k)$, the k th vertex in L , has the smallest degree in the subgraph induced by $\{L(1), L(2), \dots, L(k)\}$. If there are two (or more) vertices with minimum degree in this subgraph, their relative positions in L is determined based on degrees of their neighbors. Let G^k denote the subgraph induced by the first k vertices in L . If more than one vertex in G^k have the minimum degree, then $L(k)$ is the vertex with the minimum number of neighbor-of-neighbors, denoted by σ in Algorithm 5, among them.

Algorithm 5 Initialization

```

1: function INITIALIZATION( $G$ )
2:    $L$  = an empty array of size  $n$ 
3:   for  $k = n$  to 1 do
4:      $R \leftarrow$  the set of vertices with minimum degree in  $G$ 
5:     if  $|R| = 1$  then
6:        $u \leftarrow$  the vertex in  $R$ 
7:     else
8:       for every vertex  $v \in R$  do
9:          $\sigma(v) = \sum_{i \in N(v)} \deg(i)$ 
10:      end for
11:       $u \leftarrow$  a vertex in  $R$  with minimum  $\sigma$ 
12:    end if
13:     $L(k) \leftarrow u$ 
14:     $G \leftarrow G \setminus \{u\}$ 
15:  end for
16:  return  $L$ 
17: end function

```

Algorithm 6 Branch-and-bound procedure

```

1: function BRANCH( $G, L, C, C^*, W, W^*$ )
2:   while  $L \neq \{\}$  do
3:      $p = \text{PRUNE}(G, L, C, W, W^*)$ 
4:     if  $p = \text{true}$  then ▷ the branch is pruned
5:       return
6:     else ▷ the branch is not pruned
7:        $v \leftarrow$  last vertex in  $L$ 
8:        $\delta W = \sum_{i \in C} w_{iv}$ 
9:        $C \leftarrow C \cup \{v\}$ 
10:       $W \leftarrow W + \delta W$ 
11:       $L_v$  = an array of neighbors of  $v$  in  $L$  with the same relative order as in  $L$ 
12:      if  $L_v \neq \{\}$  then
13:        BRANCH( $G, L_v, C, C^*, W, W^*$ )
14:      else if  $W > W^*$  then
15:         $C^* \leftarrow C$ ;  $W^* \leftarrow W$ 
16:      end if
17:       $C \leftarrow C \setminus \{v\}$ 
18:       $W \leftarrow W - \delta W$ 
19:    end if
20:     $L \leftarrow L \setminus \{v\}$ 
21:  end while
22:  return
23: end function

```

Algorithm 6 shows details of the BRANCH function. In every call, this function's inputs are an incumbent clique C and its weight W , the best-known solution C^* and its weight W^* , and a list of candidate vertices to expand the incumbent clique, denoted by L . In fact, $L \subseteq V \setminus C$ contains vertices that are adjacent to all vertices of C . Initially, C is an empty set with its weight equal to zero, C^* and W^* are provided by the QCH heuristic solution, and the candidate list involves all vertices of the graph sorted according to the initial ordering.

The BRANCH function operates as follows: foremost, it calls the PRUNE function which determines whether the subgraph induced by $C \cup L$ could contain a clique heavier than the best-known solution. If the answer is negative, then the corresponding subgraph is excluded from further investigation and the function returns. Otherwise, it picks the last vertex in L and appends it to C . Correspondingly, a new candidate list for the expanded clique, denoted

by L_v in Algorithm 6, is formed. Then, the function calls itself on the subproblem defined by the new clique and the corresponding candidate list. If L_v is empty then the incumbent clique is maximal, thus its weight is compared against the best-known weight W^* and the best solution is kept. Since the BRANCH function traverses the search tree in a depth-first manner, the function returns only when it has investigated all possible outcomes of expanding C by appending $v \in L$. In other words, all cliques that contain $C \cup \{v\}$ have been examined upon solving the subproblem, so v is removed from the expanded clique as well as the candidate list.

Finally, presentation of the CBQ algorithm concludes by describing the pruning process. The PRUNE function determines whether or not a subproblem should be processed further in the BRANCH procedure. A subproblem is pruned if an upper bound on maximum weight of the cliques in the subgraph induced by $C \cup L$ is not greater than the best-known weight so far. We use the quadratic upper bound presented in the last section to prune the search tree. Algorithm 7 demonstrates details of the pruning procedure.

Algorithm 7 Pruning procedure

```

1: function PRUNE( $G, L, C, W, W^*$ )
2:    $upper\ bound = W(C \cup L)$ 
3:   for  $i = 1$  to  $|L|$  do
4:      $q_i = \sum_{j \in C} w_{ij}$ 
5:   end for
6:   construct matrix  $Q'_L$  according to (37)
7:   for  $i = 1$  to  $|L|$  do
8:      $\lambda_i$  = the  $i$ th smallest eigenvalue of  $Q'_L$ 
9:      $e^i$  = the  $i$ th eigenvector of  $Q'_L$  corresponding to  $\lambda_i$ 
10:  end for
11:   $\lambda^{max} \leftarrow \lambda_{|L|}$ 
12:  for  $i = 1$  to  $|L|$  do
13:     $q'_i = \sum_{j=1}^{|L|} e_j^i q_j$ 
14:     $b'_i = \frac{1}{2} \sum_{j=1}^{|L|} e_j^i$ 
15:  end for
16:   $i \leftarrow |L|$ 
17:  do
18:    if  $\lambda_i b'_i + q'_i \neq 0$  then
19:       $\mu^*$  = root of  $\Phi(\mu)$  according to (47) in the interval  $(\lambda^{max}, +\infty)$ 
20:       $\mathcal{Z}_L = \sum_{i=1}^{|L|} \frac{(\mu^* b'_i + q'_i) [\lambda_i (\mu^* b'_i + q'_i) + 2\mu^* q'_i]}{2(\lambda_i - \mu^*)^2}$ 
21:       $upper\ bound \leftarrow \min \{ \mathcal{Z}_L, upper\ bound \}$ 
22:      if  $upper\ bound \leq W^* - W$  then
23:        return true
24:      else
25:        return false
26:      end if
27:    end if
28:     $i \leftarrow i - 1$ 
29:  while  $\lambda_i = \lambda^{max}$ 
30:  if  $upper\ bound \leq W^* - W$  then
31:    return true
32:  else
33:    return false
34:  end if
35: end function

```

Clearly, sum of all edge weights in the subgraph induced by $C \cup L$, denoted by $W(C \cup L)$ in Algorithm 7, is a trivial upper bound for the MEWC problem on this subgraph. Although the algebraic bound presented in the last section could be much tighter, it cannot be guaranteed to reach $W(C \cup L)$ all the time. In this regard, the algorithm always compares the algebraic bound \mathcal{Z}_L with the combinatorial bound $W(C \cup L)$ and uses the tighter one in the pruning process. Besides, if $\lambda_i b'_i + q'_i = 0 \forall \lambda_i = \lambda^{max}$ we safely use $W(C \cup L)$ without calculating (51)–(52). We will show later, in the computational experiments section, that the frequency of using the combinatorial bound is negligible in comparison to that of the algebraic bound for most instances, implying effectiveness of the algebraic bound.

5 Computational experiments

In this section, we present computational results for the proposed approaches on 28 benchmark instances taken from DIMACS, RTN, and SC-NIP datasets. DIMACS instances are originally unweighted. We are using an edge-weighted version of them that was first introduced in [42] for the MEWC problem. In these graphs, a positive weight is assigned to an edge $\{i, j\} \in E$ as follows:

$$w_{ij} = (i + j) \bmod 200 + 1. \quad (53)$$

We refer to [21] for a description of RTN and SC-NIP graphs, and the associated edge weights. Table 1 presents characteristics of the test instances, where $|V|$, $|E|$ and *Density* denote number of vertices, number of edges and edge density of the graph, respectively.

We compare our results with the ones of Gouveia and Martins [21] as their work is the most recent and competitive one on solving the MEWC problem using exact solution methods. They have used sparseness of the graph to improve classic integer programming formulations of this problem. Their work presents computational results of solving the MEWC problem through 9 different integer programming formulations, namely F1, F11, F2, F21, F5, F52,

Table 1 Instances characteristics

Name	$ V $	$ E $	Density	Name	$ V $	$ E $	Density
brock200-1	200	14,834	0.754	johnson8-2-4	28	210	0.556
brock200-2	200	9876	0.496	johnson8-4-4	70	1855	0.768
brock200-3	200	12,048	0.605	keller4	171	9435	0.649
brock200-4	200	13,089	0.658	MANN-a9	45	918	0.927
C125.9	125	6963	0.898	p-hat300-1	300	10,933	0.244
C250.9	250	27,984	0.899	p-hat300-2	300	21,928	0.489
c-fat200-1	200	1534	0.077	p-hat300-3	300	33,390	0.744
c-fat200-2	200	3235	0.163	d1-RTN	2418	9317	0.003
c-fat200-5	200	8473	0.426	d3-RTN	4755	26,943	0.002
hamming6-2	64	1824	0.905	d7-RTN	6511	44,615	0.002
hamming6-4	64	704	0.349	d15-RTN	7965	62,136	0.002
hamming8-2	256	31,616	0.969	SC-NIP-m-t1	991	4161	0.008
hamming8-4	256	20,864	0.639	SC-NIP-r-t1	1393	56,276	0.058
johnson16-2-4	120	5460	0.765	SC-NIP-r-t2	1183	17,776	0.025

Table 2 Solution time

Name	W^*	CPU (s)			IP _{BASE}
		CBQ	Gouveia & Martins		
brock200-1	21,230	3047.565	>		>
brock200-2	6542	7.436	9464.240	(F1)	>
brock200-3	10,303	55.905	>		>
brock200-4	13,967	188.031	>		>
C125.9	66,248	4558.170	>		>
C250.9	–	>	>		>
c-fat200-1	7734	0.483	3.870	(F61)	31.296
c-fat200-2	26,389	0.890	33.260	(F2)	49.671
c-fat200-5	168,200	>	155.300	(F1)	134.578
hamming6-2	32,736	4.437	0.300	(F11)	17.000
hamming6-4	396	0.031	1.970	(F1)	6.468
hamming8-2	–	>	>		>
hamming8-4	12,360	439.437	>		>
johnson16-2-4	3808	84.687	>		>
johnson8-2-4	192	0.000	0.140	(F61)	0.421
johnson8-4-4	6552	0.687	2.340	(F11)	65.171
keller4	6745	42.218	>		>
MANN-a9	5460	1.906	9.390	(F1)	130.344
p-hat300-1	3321	3.281	1273.050	(F2)	8489.750
p-hat300-2	31,564	171.281	>		>
p-hat300-3	–	>	>		>
d1-RTN	4524	7.280	14.680	(F62)	1607.980
d3-RTN	5859	68.874	1565.550	(F62)	>
d7-RTN	7424	193.047	799.360	(F5)	>
d15-RTN	7424	389.672	>		>
SC-NIP-m-t1	343	1.374	5.770	(F62)	145.141
SC-NIP-r-t1	25,290	995.359	8.160	(F6)	5134.550
SC-NIP-r-t2	15,188	32.312	0.700	(F6)	146.484

F6, F61 and F62. We compare the performance of our algorithm with the best results among these models for each instance. In [21], the solution time for each instance and each model has been limited to 3 h (10,800 s). We set the same upper time limit in our experiment. In the results to follow, we have used $\xi = 1$ in the calculation of \bar{w}_{ij} according to (2).

Table 2 summarizes the main results in terms of solution time. In this table, W^* is the optimal weight for each instance (if known). The column *CBQ* presents the total CPU time in seconds taken by our algorithm to solve each instance on an Intel Core i-7 CPU @ 2.90 GHz. The next column, titled Gouveia & Martins, shows the best solution time among the 9 integer programming formulations of [21] along with the best formulation in parenthesis. These are the times reported in [21] obtained using the ILOG/CPLEX 11.2 solver on an Intel Core i-7 CPU @ 3.40 GHz. One of the most popular integer programming formulations of the MEWC problem is the one that was originally proposed in [38]. Gouveia and Martins have

considered this formulation with one additional constraint and presented the corresponding computational results in their paper. However, they have not used the automatic cut-generation option of the CPLEX solver in order to show the strength of the improved formulations. We also consider this classic formulation and present the corresponding solution times using the ILOG/CPLEX 12.7 package with the automatic cut generation on. This result is shown under the column IP_{BASE} in Table 2. In this table, we use “>” as a substitute of “> 10, 800” to indicate that the process was terminated due to the upper time limit without finding an optimal solution.

Given that the RTN and SC-NIP instances are very sparse, we added a preprocessing step to the algorithm and slightly modified the pruning process for these graphs. In the preprocessing step, vertices with degree of at most two are eliminated from the graph after recording the maximum weight of a clique that they contribute to. At the end, the result of the combinatorial B&B algorithm on the residual graph is compared with these values and the best solution is reported. Besides, at each node of the search tree, the pruning subroutine is called only if the edge density of the subgraph induced by the candidate list is at least 10%. This prevents the algorithm from performing the expensive spectral calculations on highly sparse subgraphs.

Computational results reveal the competitiveness of the CBQ algorithm. We could solve instances that all 9 formulations of [21] failed to solve, and among those that were solved by both methods we could reach much better solution times with a few exceptions. While Gouveia and Martins could solve 16 out of 28 instances, the CBQ algorithm successfully found an optimal solution for 24 instances. The quality of CBQ algorithm can be particularly observed through its performance on the *brock* and *p-hat* families. In the *brock* family, the CBQ algorithm could solve all four instances very fast. *brock200-2* is the only instance in this family that could be solved in [21], but with a solution time of more than 2.5 h. The CBQ algorithm solved this instance in less than 8 s. Among the instances of the *p-hat* family, only *p-hat300-1* could be solved in [21] with a solution time of more than 20 min; while the CBQ algorithm solved it in less than 4 s. Note that none of the formulations in [21] was the best uniformly over all solved instances. The only instance for which the CBQ algorithm failed to find an exact solution found by the best formulation of [21] was *c-fat200-5*.

The results of the CBQ algorithm presented in Table 2 use the solutions provided by the QCH heuristic as initial lower bounds. As it was mentioned previously, better heuristic results could be attained by implementing the QCH algorithm for each closed neighborhood (see Algorithm 3). Spending more time to get better initial bounds is usually preferable in large and dense graphs, where it can result in a considerable reduction in size of the search tree. Table 3 compares the quality of heuristic solutions obtained from QCH and QCH-N algorithms. In this table, \tilde{W} denotes weight of the heuristic solution.

We repeated the experiments with the QCH-N lower bound. Although an optimal solution was reached in the heuristic phase for more instances, it did not lead to solving any new instance within the time limit of the B&B algorithm. Table 3 shows that the heuristic solution of QCH-N algorithm is optimal for 23 out of the 28 instances considered, while this number is only 16 for the QCH algorithm.

Finally, we present our results explicitly about quality of the quadratic relaxation (QR) bound in comparison with the sum of edge weights (SUM). In Table 4, $\#QR$ and $\#SUM$ denote the number of times that the algebraic bound and the combinatorial bound were used in Algorithm 7, respectively. The numbers in parentheses represent the corresponding percentage. The last column, titled *Avg. QR/SUM*, shows the average ratio of the algebraic bound (when calculated) to the sum of edge weights over all calls of the PRUNE function for each instance. This table includes only the instances that we could solve under the time constraint.

Table 3 Heuristic results

Name	W^*	QCH		QCH-N	
		\bar{W}	CPU (s)	\bar{W}	CPU (s)
brock200-1	21,230	21,230	0.015	21,230	1.984
brock200-2	6542	6542	0.015	6542	0.750
brock200-3	10,303	10,303	0.015	10,303	1.250
brock200-4	13,967	9634	0.015	13,736	1.562
C125.9	66,248	53,145	0.000	65,416	0.734
C250.9	–	69,977	0.015	83,780	7.062
c-fat200-1	7734	7734	0.015	7734	0.046
c-fat200-2	26,389	26,389	0.015	26,389	0.062
c-fat200-5	168,200	168,200	0.015	168,200	0.625
hamming6-2	32,736	32,736	0.000	32,736	0.046
hamming6-4	396	396	0.000	396	0.000
hamming8-2	–	800,624	0.046	800,624	9.796
hamming8-4	12,360	12,160	0.015	12,360	2.968
johnson16-2-4	3808	3608	0.000	3808	0.328
johnson8-2-4	192	192	0.000	192	0.000
johnson8-4-4	6552	6552	0.000	6552	0.046
keller4	6745	6745	0.015	6745	0.796
MANN-a9	5460	5445	0.000	5460	0.046
p-hat300-1	3321	3089	0.031	3321	0.578
p-hat300-2	31,564	25,412	0.031	31,564	3.328
p-hat300-3	–	50,995	0.031	59,425	8.218
d1-RTN	4524	4524	2.609	4524	0.265
d3-RTN	5859	5859	24.390	5859	2.328
d7-RTN	7424	7244	64.000	7424	6.687
d15-RTN	7424	6735	128.828	7424	13.578
SC-NIP-m-t1	343	343	0.609	343	0.234
SC-NIP-r-t1	25,290	25,290	2.640	25,290	14.859
SC-NIP-r-t2	15,188	15,188	0.500	15,188	2.421

Table 4 shows that, on DIMACS instances, the QR bound was much tighter than SUM except for c-fat200-1, c-fat200-2 and hamming6-2. Excluding these instances, the maximum average ratio is 0.76 for MANN-a9. The ratio is as good as 0.40 and 0.44 for johnson16-2-4 and keller4, respectively. For the brock, C, johnson, keller, MANN and p-hat families, the number of times that the algorithm applied the combinatorial bound is negligible compared to the application of QR bound. Looseness of the QR bound on the c-fat family could explain failure of our algorithm in solving c-fat200-5 under the time limit. On the RTN and SC-NIP, however, the QR bound did not perform as well as on the DIMACS graphs. While the number of times that the QR bound outperformed SUM is considerable—except for SC-NIP-r-t1—it was never the dominant pruning method and it performed very poorly in terms of the average ratio.

Table 4 Quality of the quadratic relaxation bound

Name	# <i>QR</i>	# <i>SUM</i>	Avg. <i>QR</i> / <i>SUM</i>
brock200-1	38,819,232 (99.82 %)	71,511 (0.18 %)	0.66
brock200-2	108,456 (99.82 %)	200 (0.18 %)	0.57
brock200-3	884,461 (99.82 %)	1590 (0.18 %)	0.61
brock200-4	2,711,652 (99.85 %)	4071 (0.15 %)	0.62
C125.9	26,539,316 (99.95 %)	13,835 (0.05 %)	0.72
c-fat200-1	5 (1.35 %)	366 (98.65 %)	1.27
c-fat200-2	0 (0.00 %)	4861 (100.00 %)	1.30
hamming6-2	11,836 (36.68 %)	20,431 (63.32 %)	1.02
hamming6-4	790 (80.28 %)	194 (19.72 %)	0.70
hamming8-4	4,990,643 (97.13 %)	147,215 (2.87 %)	0.65
johnson16-2-4	3,076,874 (99.49 %)	15,649 (0.51 %)	0.40
johnson8-2-4	215 (89.96 %)	24 (10.04 %)	0.51
johnson8-4-4	10,863 (98.19 %)	200 (1.81 %)	0.55
keller4	704,846 (99.71 %)	2061 (0.29 %)	0.44
MANN-a9	72,891 (90.18 %)	7938 (9.82 %)	0.76
p-hat300-1	22,217 (98.43 %)	354 (1.57 %)	0.58
p-hat300-2	1,694,985 (99.13 %)	14,820 (0.87 %)	0.74
d1-RTN	574 (39.42 %)	882 (60.58 %)	3.87
d3-RTN	1106 (32.22 %)	2327 (67.78 %)	4.90
d7-RTN	1449 (27.48 %)	3824 (72.52 %)	5.33
d15-RTN	2057 (21.74 %)	7406 (78.26 %)	4.61
SC-NIP-m-t1	407 (35.42 %)	742 (64.58 %)	2.35
SC-NIP-r-t1	116 (0.04 %)	298,088 (99.96 %)	3.98
SC-NIP-r-t2	131 (12.31 %)	933 (87.69 %)	10.89

6 Conclusion

This work introduces a nonconvex quadratic programming formulation for the maximum edge weight clique (MEWC) problem. Optimality characteristics of the new formulation were studied and the correspondence between local and global optima of the continuous problem and structures in the underlying graph was established. Mainly, it was shown that a feasible solution is a local maximizer of the continuous problem if and only if it is the characteristic vector of a maximal clique in the graph. Consequently, a global maximizer characterizes a maximum edge weight clique.

Based on the continuous formulation and its optimality characterization, a new exact algorithm for the MEWC problem was presented. The algorithm is a combinatorial branch-and-bound procedure that takes advantage of an algebraic upper bound. This bound is derived from a quadratic relaxation of the continuous problem. The branch-and-bound algorithm also uses an initial lower bound which is provided by a construction heuristic method. The heuristic algorithm extracts a maximal clique based on solving an approximation of the continuous problem. We conducted computational experiments on 28 benchmark instances and compared the solution time of our algorithm with the best results on the same instances in

the literature. The experiments show significant improvements. In particular, we could solve 24 instances within the time limit of 3 h (for each instance). We also used the computational results to show the quality of the presented algebraic bound explicitly. Our results indicate the effectiveness of our method, especially on relatively dense graphs.

In future research, one may devise other heuristic algorithms for the MEWC problem by different approximations of the continuous problem presented in this paper. Besides, other properties of the presented formulation may be explored to find tighter algebraic upper bounds for this problem.

Acknowledgements We would like to thank two anonymous referees, whose feedback helped us to improve the paper. This work was carried out while the 2nd author was a visiting scholar at Texas A&M University, College Station, TX, USA and is partially supported by scholarship SFRH/BSAB/113662/2015. Partial support by DOD-ONR (N00014-13-1-0635) and NSF (CMMI-1538493) Grants is also gratefully acknowledged.

References

1. Abello, J., Butenko, S., Pardalos, P., Resende, M.: Finding independent sets in a graph using continuous multivariable polynomial formulations. *J. Glob. Optim.* **21**, 111–137 (2001)
2. Akutsu, T., Hayashida, M., Tomita, E., Suzuki, J.: Protein threading with profiles and constraints. In: *Proceedings of the Fourth IEEE Symposium on Bioinformatics and Bioengineering*, pp. 537–544 (2004 May)
3. Alidaee, B., Glover, F., Kochenberger, G., Wang, H.: Solving the maximum edge weight clique problem via unconstrained quadratic programming. *Eur. J. Oper. Res.* **181**(2), 592–597 (2007)
4. Aringhieri, R., Cordone, R.: Comparing local search metaheuristics for the maximum diversity problem. *J. Oper. Res. Soc.* **62**(2), 266–280 (2011)
5. Balasundaram, B., Butenko, S.: On a polynomial fractional formulation for independence number of a graph. *J. Glob. Optim.* **35**, 405–421 (2006)
6. Batsyn, M., Goldengorin, B., Maslov, E., Pardalos, P.M.: Improvements to MCS algorithm for the maximum clique problem. *J. Comb. Optim.* **27**(2), 397–416 (2014)
7. Bomze, I.M.: Evolution towards the maximum clique. *J. Glob. Optim.* **10**, 143–164 (1997)
8. Bomze, I.M., Budinich, M., Pelillo, M., Rossi, C.: A new “annealed” heuristic for the maximum clique problem. In: Pardalos, P.M. (ed.) *Approximation and Complexity in Numerical Optimization: Continuous and Discrete Problems*, pp. 78–96. Kluwer Academic Publishers, Dordrecht (2000)
9. Brown, J.B., Bahadur, D.K.C., Tomita, E., Akutsu, T.: Multiple methods for protein side chain packing using maximum weight cliques. *Genome Inform.* **17**, 3–12 (2006)
10. Bulò, S.R., Pelillo, M.: A generalization of the Motzkin–Straus theorem to hypergraphs. *Optim. Lett.* **3**(2), 287–295 (2009)
11. Busygin, S.: A new trust region technique for the maximum weight clique problem. *Discrete Appl. Math.* **154**, 2080–2096 (2006)
12. Busygin, S., Butenko, S., Pardalos, P.M.: A heuristic for the maximum independent set problem based on optimization of a quadratic over a sphere. *J. Comb. Optim.* **6**, 287–297 (2002)
13. Carraghan, R., Pardalos, P.: An exact algorithm for the maximum clique problem. *Oper. Res. Lett.* **9**, 375–382 (1990)
14. Cavique, L.: A scalable algorithm for the market basket analysis. *J. Retail. Consum. Serv.* **14**(6), 400–407 (2007)
15. de Andrade, M.R.Q., de Andrade, P.M.F., Martins, S.L., Plastino, A.: Grasp with path-relinking for the maximum diversity problem. In: Nikolettseas, S.E. (ed.) *Proceedings of the Experimental and Efficient Algorithms: 4th International Workshop, WEA 2005, Santorini Island, Greece, May 10–13, 2005*, pp. 558–569. Springer, Berlin (2005)
16. Dijkhuizen, G., Faigle, U.: A cutting-plane approach to the edge-weighted maximal clique problem. *Eur. J. Oper. Res.* **69**(1), 121–130 (1993)
17. Forsythe, G.E., Golub, G.H.: On the stationary values of a second-degree polynomial on the unit sphere. *J. Soc. Ind. Appl. Math.* **13**(4), 1050–1068 (1965)
18. Gallego, M., Duarte, A., Laguna, Manuel, Martí, Rafael: Hybrid heuristics for the maximum diversity problem. *Comput. Optim. Appl.* **44**(3), 411–426 (2009)

19. Gibbons, L.E., Hearn, D.W., Pardalos, P.M.: A continuous based heuristic for the maximum clique problem. In: Johnson, D.S., Trick, M.A. (eds.) *Cliques, Coloring, and Satisfiability: Second DIMACS Challenge, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 26, pp. 103–124. American Mathematical Society, Providence (1996)
20. Gibbons, L.E., Hearn, D.W., Pardalos, P.M., Ramana, M.V.: Continuous characterizations of the maximum clique problem. *Math. Oper. Res.* **22**, 754–768 (1997)
21. Gouveia, L., Martins, P.: Solving the maximum edge-weight clique problem in sparse graphs with compact formulations. *EURO J. Comput. Optim.* **3**(1), 1–30 (2015)
22. Harant, J.: A lower bound on the independence number of a graph. *Discrete Math.* **188**, 239–243 (1998)
23. Harant, J.: Some news about the independence number of a graph. *Discuss. Mathe. Graph Theory* **20**, 71–79 (2000)
24. Harant, J., Pruchnewski, A., Voigt, M.: On dominating sets and independent sets of graphs. *Comb. Probab. Comput.* **8**, 547–553 (1999)
25. Hosseinian, S., Fontes, D.B.M.M., Butenko, S.: A quadratic approach to the maximum edge weight clique problem. In: Rocha, A.M.A.C., Costa, M.F.P., Fernandes, E.M.G.P. (eds.) *XIII Global Optimization Workshop (GOW'16)*, pp. 125–128. University of Minho, Braga (2016)
26. Hosseinian, S., Fontes, D.B.M.M., Butenko, S., Buongiorno Nardelli, M., Fornari, M., Curtarolo, S.: The maximum edge weight clique problem: formulations and solution approaches. In: Butenko, S., Pardalos, P.M., Shylo, V. (eds.) *Optimization Methods and Applications*, pp. 217–237. Springer, Berlin (2017)
27. Hunting, M., Faigle, U., Kern, W.: A Lagrangian relaxation approach to the edge-weighted clique problem. *Eur. J. Oper. Res.* **131**(1), 119–131 (2001)
28. Jabbar, M.A., Deekshatulu, B. L., Chandra, P.: Graph based approach for heart disease prediction. In: Das, Vinu V (ed.) *Proceedings of the Third International Conference on Trends in Information, Telecommunication and Computing*, pp. 465–474. Springer, New York (2013)
29. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) *Complexity of Computer Computations*, pp. 85–103. Plenum Press, New York (1972)
30. Ma, T., Latecki, L.J.: Maximum weight cliques with mutex constraints for video object segmentation. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 670–677 (2012 June)
31. Macambira, E.M., de Souza, C.C.: The edge-weighted clique problem: valid inequalities, facets and polyhedral computations. *Eur. J. Oper. Res.* **123**(2), 346–371 (2000)
32. Martí, R., Gallego, M., Duarte, A., Pardo, E.G.: Heuristics and metaheuristics for the maximum diversity problem. *J. Heur.* **19**(4), 591–615 (2013)
33. Motzkin, T.S., Straus, E.G.: Maxima for graphs and a new proof of a theorem of Turán. *Can. J Math.* **17**, 533–540 (1965)
34. Östergård, P.R.J.: A fast algorithm for the maximum clique problem. *Discrete Appl. Math.* **120**, 197–207 (2002)
35. Palubeckis, G.: Iterated tabu search for the maximum diversity problem. *Applied Math. Comput.* **189**(1), 371–383 (2007)
36. Pardalos, P.M., Vavasis, S.A.: Quadratic programming with one negative eigenvalue is NP-hard. *J. Global Optim.* **1**(1), 15–22 (1991)
37. Pardalos, P.M., Phillips, A.T.: A global optimization approach for solving the maximum clique problem. *Int. J. Comput. Math.* **33**, 209–216 (1990)
38. Park, K., Lee, K., Park, S.: An extended formulation approach to the edge-weighted maximal clique problem. *Eur. J. Oper. Res.* **95**(3), 671–682 (1996)
39. Pavan, M., Pelillo, M.: Generalizing the motzkin-strauss theorem to edge-weighted graphs, with applications to image segmentation. In: Rangarajan, Anand, Figueiredo, Mário, Zerubia, Josiane (eds.) *Proceedings of the Energy Minimization Methods in Computer Vision and Pattern Recognition: 4th International Workshop, EMMCVPR 2003, Lisbon, Portugal, July 7–9, 2003*, pp. 485–500. Springer, Berlin (2003)
40. Pelillo, M., Jagota, A.: Feasible and infeasible maxima in a quadratic program for maximum clique. *J. Artif. Neural Netw.* **2**, 411–420 (1995)
41. Peng, Y., Peng, H., Tang, Q., Zhao, C.: An extension of the Motzkin–Straus theorem to non-uniform hypergraphs and its applications. *Discrete Appl. Math.* **200**, 170–175 (2016)
42. Pullan, W.: Approximating the maximum vertex/edge weighted clique using local search. *J. Heur.* **14**(2), 117–134 (2008)
43. Segundo, P.San, Nikolaev, A., Batsyn, M., Batsyn, M.: Infra-chromatic bound for exact maximum clique search. *Comput. Oper. Res.* **64**, 293–303 (2015)
44. Silva, G.C., de Andrade, M.R.Q., Ochi, L.S., Martins, S.L., Plastino, A.: New heuristics for the maximum diversity problem. *J. Heur.* **13**(4), 315–336 (2007)

45. Sorensen, M.M.: New facets and a branch-and-cut algorithm for the weighted clique problem. *Eur. J. Oper. Res.* **154**(1), 57–70 (2004)
46. Tomita, E., Kameda, T.: An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments. *J. Global Optim.* **37**(1), 95–111 (2007)
47. Tomita, E., Sutani, Y., Higashi, T., Takahashi, S., Wakatsuki, M.: A simple and faster branch-and-bound algorithm for finding a maximum clique. In: Rahman, Md. Saidur, Fujita, Satoshi (eds), *Proceedings of the WALCOM: Algorithms and Computation: 4th International Workshop, WALCOM 2010, Dhaka, Bangladesh, February 10–12, 2010*, pp. 191–203. Springer, Berlin (2010)
48. Wang, Y., Hao, J.K., Glover, F., Lü, Z.: A tabu search based memetic algorithm for the maximum diversity problem. *Eng. Appl. Artif. Intell.* **27**, 103–114 (2014)
49. Wu, Q., Hao, J.-K.: A review on algorithms for maximum clique problems. *Eur. J. Oper. Res.* **242**(3), 693–709 (2015)
50. Ye, Y.: A new complexity result on minimization of a quadratic function with a sphere constraint. In: Floudas, C., Pardalos, P. (eds.) *Recent Advances in Global Optimization*, pp. 19–31. Princeton University Press, Princeton (1992)