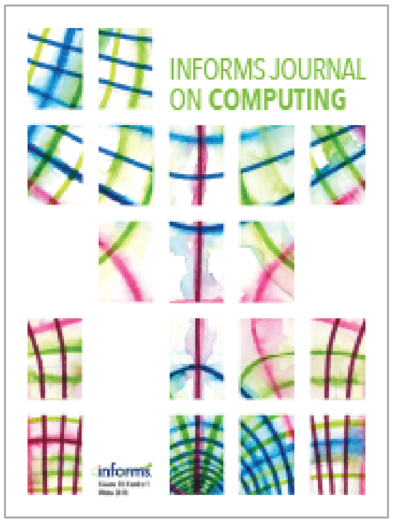


INFORMS Journal on Computing

A Lagrangian Bound on the Clique Number and an Exact Algorithm for the Maximum Edge Weight Clique Problem

Seyedmohammadhossein Hosseinian, Dalila B. M. M. Fontes, Sergiy Butenko

http://pubsonline.informs.org/journal/ijoc

INFORMS JOURNAL ON COMPUTING
Articles in Advance, pp. 1–16
ISSN 1091-9856 (print), ISSN 1526-5528 (online)

A Lagrangian Bound on the Clique Number and an Exact Algorithm for the Maximum Edge Weight Clique Problem

Seyedmohammadhossein Hosseinian,[a] Dalila B. M. M. Fontes,[b] Sergiy Butenko[a]

[a] Industrial & Systems Engineering, Texas A&M University, College Station, Texas 77843; [b] Institute for Systems and Computer Engineering, Technology and Science and Faculdade de Economia, Universidade do Porto, 4200-465 Porto, Portugal

Contact: hosseinian@tamu.edu, http://orcid.org/0000-0001-7016-5925 (SH); fontes@fep.up.pt, http://orcid.org/0000-0002-9402-2088 (DBMMF); butenko@tamu.edu, http://orcid.org/0000-0002-6662-9552 (SB)

Received: May 29, 2018
Revised: December 26, 2018; March 5, 2019; March 29, 2019
Accepted: March 30, 2019
Published Online in Articles in Advance: January 24, 2020

https://doi.org/10.1287/ijoc.2019.0898

Abstract. This paper explores the connections between the classical maximum clique problem and its edge-weighted generalization, the maximum edge weight clique (MEWC) problem. As a result, a new analytic upper bound on the clique number of a graph is obtained and an exact algorithm for solving the MEWC problem is developed. The bound on the clique number is derived using a Lagrangian relaxation of an integer (linear) programming formulation of the MEWC problem. Furthermore, coloring-based bounds on the clique number are used in a novel upper-bounding scheme for the MEWC problem. This scheme is employed within a combinatorial branch-and-bound framework, yielding an exact algorithm for the MEWC problem. Results of computational experiments demonstrate a superior performance of the proposed algorithm compared with existing approaches.

History: Accepted by Andrea Lodi, Area Editor for Design and Analysis of Algorithms—Discrete.
Funding: This work was supported by the National Science Foundation [Grant CMMI-1538493], Office of Naval Research [Grant N00014-13-1-0635], and Portuguese Foundation for Science and Technology [Grant POCI-01-0145-FEDER-031821].

Keywords: graph theory • analytic upper bound on the clique number • maximum edge weight clique problem

1. Introduction

Given a simple and undirected graph $G = (V, E)$, where $V = \{1, 2, \cdots, n\}$ is the set of vertices and E is the set of edges, a *clique* is a subset of vertices $C \subseteq V$ that induces a complete subgraph. A clique is called *maximal* if it is not a subset of a larger clique, and *maximum* if there is no larger clique in the graph. The cardinality of a maximum clique in G is called the *clique number* of the graph and is denoted by $\omega(G)$. The maximum clique (MC) problem, which is to find a maximum clique in the graph, is one of the most popular problems of combinatorial optimization and a great deal of research has been dedicated to its study; see, e.g., the survey papers by Bomze et al. (1999) and Wu and Hao (2015).

The maximum edge weight clique (MEWC) problem is a generalization of the MC problem to edge-weighted graphs. In an edge-weighted graph $G = (V, E, w^E)$, every edge $\{i, j\} \in E$ has a positive weight w_{ij}, and the edge weight of $C \subseteq V$ is defined as $W(C) = \sum_{\{i,j\}\in E(C)} w_{ij}$, where $E(C)$ denotes the set of edges with both endpoints in C. The MEWC problem is to find a clique with the maximum edge weight in G. If the weight of each edge is equal to 1, finding a maximum edge weight clique is equivalent to finding a maximum clique C^* with $W(C^*) = \binom{\omega(G)}{2}$. Therefore, the MEWC problem is at least as hard to solve as the MC problem, which is $\mathcal{NP}$-hard (Karp 1972). The MC problem is also known to be hard to approximate to within $n^{1-\epsilon}$ for every $\epsilon > 0$ (Zuckerman 2006). Similar to the MC problem, the MEWC problem finds applications in various fields, including computational biology (Akutsu et al. 2004, Brown et al. 2006), computer graphics (Pavan and Pelillo 2003, Ma and Latecki 2012), marketing (Cavique 2007), healthcare (Jabbar et al. 2013), and materials science (Hosseinian et al. 2017).

In this paper, we investigate some close connections between the MEWC and MC problems that lead to new results for both problems. First, we use a Lagrangian relaxation of an integer (linear) programming formulation of the MEWC problem to derive an analytic upper bound on the clique number of a graph. Then, upper bounds on the clique number are used in an upper-bounding scheme for the MEWC problem, which is employed in a combinatorial branch-and-bound (B&B) procedure for this problem. The performance of the proposed algorithm strongly depends on our ability to quickly compute nontrivial upper bounds on the clique number.

Several analytic (i.e., closed-form) upper bounds on the clique number of a graph have been proposed in the literature. The first bound of this type appeared in the work of Wilf (1967), who showed that $\omega(G) \le \rho(A_G) + 1$, where $\rho(A_G)$ denotes the spectral radius of the adjacency matrix A_G of graph G. Later, Amin and Hakimi (1972) presented two bounds. They noted that $\omega(G) \le (3 + \sqrt{9 - 8(n - m)})/2$ for connected graphs, with n and m denoting the number of vertices and edges, respectively. They also proved that $\omega(G) \le N_{-1}(A_G) + 1$, where $N_{-1}(A_G)$ is the number of eigenvalues of the adjacency matrix of the graph not exceeding -1. Budinich (2003) used complex analysis to show that $\omega(G) \le n - \text{rank}(A_{\bar{G}})/2$, where $A_{\bar{G}}$ denotes the adjacency matrix of the complement graph $\bar{G}$ of G and $\text{rank}(A_{\bar{G}})$ is the rank of this matrix. Among these bounds, only the first one of Amin and Hakimi (1972) is computable in linear time. The others involve spectral calculations on the adjacency matrix of the graph (or its complement) and require $\mathcal{O}(n^3)$ arithmetic operations. The upper bound derived in this paper provides another linear-time computable alternative to the first Amin-Hakimi bound. It should be noted that our bound was obtained as a result of an attempt of obtaining a tight Lagrangian relaxation-based upper bound for the MEWC problem. Consequently, coloring-based bounds proved to be superior within the proposed B&B framework for the MEWC problem; however, the obtained analytic upper bound on the clique number is still of interest as a nontrivial, closed-form, and easily computable bound for the classical MC problem.

A closely related problem to the MEWC problem is the maximum diversity problem, which is to find a maximum edge weight clique of cardinality not exceeding a given bound k in a complete edge-weighted input graph. As suggested by some authors (Padberg 1989, Mehrotra and Trick 1998), an instance of the MEWC problem can be transformed to a complete graph by adding dummy edges with sufficiently large negative weights and then solved as an instance of the maximum diversity problem with $k = |V|$. In this regard, these two names have been used interchangeably in the literature to refer to the maximum diversity problem. The exact solution methods proposed for the maximum diversity problem include combinatorial B&B (Martí et al. 2010) and branch-and-cut algorithms based on integer (linear) programming formulations (Dijkhuizen and Faigle 1993, Park et al. 1996, Macambira and de Souza 2000, Hunting et al. 2001, Sørensen 2004). Several heuristic and metaheuristic methods have also been applied to this problem, including tabu search (Alidaee et al. 2007, Palubeckis 2007, Aringhieri and Cordone 2011), memetic search (Wang et al. 2014), scatter search (Gallego et al. 2007), and GRASP (de Andrade et al. 2005, Silva et al. 2007).

To the best of our knowledge, the only works that focused on the MEWC problem (i.e., with non-complete input graphs) at the time of completion of the present work were Pullan (2008), Gouveia and Martins (2015), Hosseinian et al. (2016, 2018), and Fontes et al. (2018). More specifically, Pullan (2008) extended the phased local search method proposed for the MC problem in Pullan (2006) to address the MEWC problem. Gouveia and Martins (2015) studied existing integer programming formulations of the maximum diversity problem to adapt them for sparse graphs and introduced a set of new formulations for the MEWC problem. Recently, Hosseinian et al. (2016) developed a construction heuristic algorithm based on solving a quadratically constrained quadratic problem. In Hosseinian et al. (2018), the same authors introduced a quadratic programming formulation for the MEWC problem and showed the correspondence between local optima of the continuous problem and special structures in the underlying graph. They also presented an exact combinatorial B&B algorithm that uses a relaxation of the new formulation to prune the search tree. In addition, Fontes et al. (2018) proposed a biased random-key genetic algorithm for this problem. We refer to Hosseinian et al. (2017) for a detailed review of these methods.

Several months after submission of the original version of the present paper, Shimizu et al. (2018, 2019) presented two more works dedicated to exact methods for the MEWC problem. Similarly to our work, their approaches are based on a combinatorial B&B framework. Shimizu et al. (2018, 2019) also report excellent computational results, thus providing further evidence of the validity of this methodological direction.

The remainder of this paper is organized as follows. Section 2 presents derivation of the analytic upper bound on $\omega(G)$ from a Lagrangian relaxation of an integer (linear) programming formulation of the MEWC problem. In Section 3, we first explain how the Lagrangian relaxation problem visited in Section 2 motivates a new upper-bounding method for the MEWC problem, then we present an exact algorithm that employs this method within a combinatorial B&B framework. Section 4 provides results of computational experiments, and Section 5 concludes the paper.

2. A Lagrangian Relaxation Bound on the Clique Number

A well-known integer (linear) programming formulation of the MEWC problem on a graph $G = (V, E, w^E)$

with positive edge weights is as follows (Park et al. 1996):

$$
\begin{aligned}
W^* = \max \quad & \sum_{\{i,j\}\in E} w_{ij}\, y_{ij} \\
\text{s.t.} \quad & y_{ij} \le x_i && \forall \{i,j\} \in E \\
& y_{ij} \le x_j && \forall \{i,j\} \in E \\
& x_i + x_j \le 1 && \forall \{i,j\} \notin E \\
& x_i \in \{0,1\} && \forall i \in V \\
& y_{ij} \in \{0,1\} && \forall \{i,j\} \in E.
\end{aligned} \tag{1}
$$

In this formulation, every vertex $i \in V$ is represented by a variable x_i and every edge $\{i,j\} \in E$ by a variable y_{ij}. Every feasible solution of formulation (1) induces a clique $C = \{i \in V \mid x_i = 1\}$ with $E(C) = \{\{i,j\} \in E \mid y_{ij} = 1\}$. Therefore, an optimal solution of formulation (1) characterizes a clique with the maximum total sum of edge weights in the induced subgraph.

Consider the full Lagrangian relaxation of Equation (1), except for integrality of the variables, that is,

$$
\begin{aligned}
\mathfrak{L}(\boldsymbol{\mu}^1,\boldsymbol{\mu}^2,\boldsymbol{\mu}^3) = \max_{\mathbf{x},\mathbf{y}} & \sum_{\{i,j\}\in E} w_{ij}y_{ij} + \sum_{\{i,j\}\in E} \mu^1_{ij}(x_i - y_{ij}) \\
& + \sum_{\{i,j\}\in E} \mu^2_{ij}(x_j - y_{ij}) + \sum_{\{i,j\}\notin E} \mu^3_{ij}(1 - x_i - x_j) \\
\text{s.t. } & x_i \in \{0,1\} \qquad \forall i \in V \\
& y_{ij} \in \{0,1\} \qquad \forall \{i,j\} \in E,
\end{aligned} \tag{2}
$$

where $\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \boldsymbol{\mu}^3 \ge \mathbf{0}$ denote the vectors of Lagrangian multipliers (dual variables) corresponding to the first, second and third set of constraints in Equation (1), respectively. Restricting the dual variables in Equation (2) to the line $\mu^1_{ij} = \mu^2_{ij} = \mu^3_{ij} = \mu, \forall i,j \in V$, we obtain a simpler problem,

$$
\begin{aligned}
z(\mu) = \max_{\mathbf{x},\mathbf{y}} & \sum_{\{i,j\}\in E} \big(w_{ij}y_{ij} + \mu(x_i + x_j - 2y_{ij})\big) \\
& + \sum_{\{i,j\}\notin E} \mu(1 - x_i - x_j) \\
\text{s.t. } & x_i \in \{0,1\} \quad \forall i \in V \\
& y_{ij} \in \{0,1\} \quad \forall \{i,j\} \in E.
\end{aligned} \tag{3}
$$

Let $\mathcal{L}(\mathbf{x},\mathbf{y},\mu)$ denote the objective function of Equation (3). Then,

$$
\begin{aligned}
\mathcal{L}(\mathbf{x},\mathbf{y},\mu) &= \sum_{\{i,j\}\in E} \big(w_{ij}y_{ij} + \mu(x_i + x_j - 2y_{ij})\big) + \sum_{\{i,j\}\notin E} \mu(1 - x_i - x_j) \\
&= \sum_{\{i,j\}\in E} y_{ij}(w_{ij} - 2\mu) + \mu\left(\sum_{\{i,j\}\in E} (x_i + x_j) + \sum_{\{i,j\}\notin E} (1 - x_i - x_j)\right) \\
&= \sum_{\{i,j\}\in E} y_{ij}(w_{ij} - 2\mu) + \mu\left(\bar{m} + \sum_{i\in V} x_i(2d_i - n + 1)\right),
\end{aligned} \tag{4}
$$

where d_i denotes the degree of a vertex $i \in V$, n is the number of vertices, and $\bar{m} = \binom{n}{2} - |E|$ represents the number of edges in the complement graph of G. Equation (4) implies that the restricted Lagrangian relaxation problem of Equation (3) is separable with respect to variables corresponding to vertices and edges of G, so it can be rewritten as follows:

$$
\begin{aligned}
z(\mu) = & \max_{\mathbf{y}\in\{0,1\}^{|E|}} \sum_{e\in E} y_e(w_e - 2\mu) \\
& + \mu\left(\bar{m} + \max_{\mathbf{x}\in\{0,1\}^n} \sum_{v\in V} x_v(2d_v - n + 1)\right).
\end{aligned} \tag{5}
$$

Optimal solutions of both optimization problems in Equation (5) are determined by the signs of the coefficients of the binary decision variables. Therefore, $z(\mu)$ can be calculated according to Equation (6) for every $\mu \ge 0$,

$$
z(\mu) = \sum_{e\in E^+} (w_e - 2\mu) + \mu\left(\bar{m} + \sum_{v\in V^+} (2d_v - n + 1)\right), \tag{6}
$$

where $V^+ \subseteq V$ and $E^+ \subseteq E$ are defined as follows:

$$
V^+ = \{v \in V \mid d_v \ge (n-1)/2\}, \tag{7}
$$

$$
E^+ = \{e \in E \mid w_e \ge 2\mu\}. \tag{8}
$$

Note that V^+ only depends on the structure of G, whereas E^+ is a function of weights and the Lagrangian multiplier μ. Minimizing $z(\mu)$ over the nonnegative values of μ gives an upper bound on the optimal value of the MEWC problem. That is,

$$
\begin{aligned}
W^* \le \min_{\boldsymbol{\mu}\ge\mathbf{0}} \mathfrak{L}(\boldsymbol{\mu}) \le \min_{\mu\ge 0} & \sum_{e\in E^+} (w_e - 2\mu) \\
& + \mu\left(\bar{m} + \sum_{v\in V^+} (2d_v - n + 1)\right),
\end{aligned} \tag{9}
$$

where $\boldsymbol{\mu} = (\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \boldsymbol{\mu}^3)$. In Equation (9), the first inequality is the weak duality and the second one is due to the fact that $z(\mu)$ is a restriction of $\mathfrak{L}(\boldsymbol{\mu})$. We use this result to get an upper bound on the clique number of the graph. The following proposition serves this purpose.

Proposition 1. *Given an undirected graph $G = (V,E)$ with $|V| = n$ and $|E| = m$, let $b = \binom{n}{2} - m + \sum_{v\in V^+} (2d_v - n + 1)$, where d_v denotes degree of a vertex $v \in V$ and V^+ is defined in Equation (7). Then, $a = \lfloor (1 + \sqrt{4b+1})/2 \rfloor$ gives an upper bound on $\omega(G)$.*

Proof. Consider an edge-weighted version of G with $w_{ij} = 1, \forall \{i,j\} \in E$. Then, by Equation (9),

$$
\bar{W} = \min_{\mu\ge 0} \sum_{e\in E^+} (1 - 2\mu) + \mu b \tag{10}
$$

is an upper bound on the optimal value of the MEWC problem on G, which equals $\binom{\omega(G)}{2}$. Note that the parameter b depends only on the structure of the graph and is invariant of the edge weights. Given the definition of E^+, the objective function of Equation (10) is increasing for $\mu \geq 1/2$, so we can restrict the optimization interval to $[0, 1/2]$. Since $E^+ = E$ for all values of $\mu \in [0, 1/2]$, the objective function of Equation (10) is linear and reaches its minimum at $\mu = 0$ or $\mu = 1/2$. Therefore, $\binom{\omega(G)}{2} \leq \min\{|E|, b/2\} \leq b/2$, which implies that $\omega(G) \leq \lfloor (1 + \sqrt{4b+1})/2 \rfloor$. □

This bound equals $\omega(G)$ if the graph can be partitioned into two cliques that are either very densely or very sparsely connected to each other. Two extreme cases are complete graphs and independent union of two cliques, on which the proposed bound can be easily checked to be sharp. Clearly, the bound is loose on sparse graphs due to the term $\bar{m} = \binom{n}{2} - m$.

Example 1. We illustrate Proposition 1 on the graph of Figure 1. This graph has $n = 7$ vertices and $m = 13$ edges, and its clique number is $\omega(G) = 3$:

$$\begin{aligned} b &= \binom{n}{2} - m + \sum_{v \in V^+} (2d_v - n + 1) \\ &= 21 - 13 + 5(8 - 7 + 1) = 18, \end{aligned} \tag{11}$$

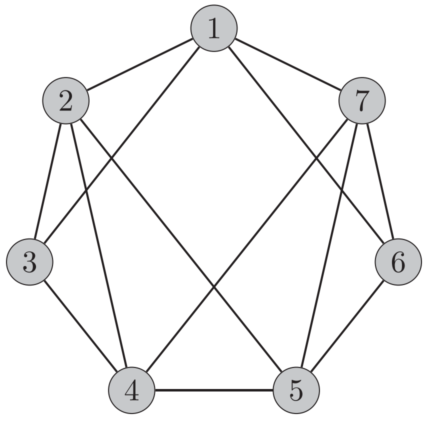
$$a = \left\lfloor \frac{1 + \sqrt{4b+1}}{2} \right\rfloor = \left\lfloor \frac{1 + \sqrt{73}}{2} \right\rfloor = \lfloor 4.77 \rfloor = 4 \geq \omega(G). \tag{12}$$

Note that this graph has five vertices of degree 4, so absence of a clique of cardinality 5 is not evident without exploring adjacency of the vertices. In particular, using the first bound by Amin and Hakimi (1972), we obtain $\omega(G) \leq \lfloor (3 + \sqrt{9 - 8(7 - 13)})/2 \rfloor = 5$. None of the spectral bounds, presented in Section 1, will generate an upper bound better than 4 for this graph.

3. An Exact Solution Method for the MEWC Problem

Consider the univariate Lagrangian relaxation problem of Equation (9). The objective function of this minimization problem is piecewise linear and convex, thus it can be solved efficiently to get an upper bound on W^*. Such an upper bound, however, is usually too loose to be used in a B&B algorithm. In fact, it can be shown that the optimal value of the univariate Lagrangian relaxation problem is no better than sum of the $b/2$ largest edge weights in the graph, where b is given in Proposition 1. A formal proof for this statement is presented in Appendix A. This result implies that a similar approach may be taken with any upper bound $\bar{\omega} \geq \omega(G)$ better than a (in Proposition 1) to tighten the bound on W^*. Evidently, given availability of such $\bar{\omega}$, this approach is disadvantageous as it will not generate an upper bound better than sum of the $\binom{\bar{\omega}}{2}$ largest edge weights in the graph. Motivated by this observation, in Section 3.1, we present an upper-bounding method for the MEWC problem that takes advantage of known upper bounds on the clique number and exploits adjacency of edges in the graph to draw a tighter bound on W^*.

Figure 1. The Example Graph



We use this upper-bounding method in the pruning subroutine of a combinatorial B&B procedure. The combinatorial B&B procedure is an implicit enumeration of cliques in the graph, which has been vastly used before in the algorithms proposed for the MC problem with different pruning methods and implementation details (see, e.g., Carraghan and Pardalos 1990, Östergård 2002, Tomita and Kameda 2007, Tomita et al. 2010, Batsyn et al. 2014, San Segundo et al. 2015). The procedure starts with a trivial clique (i.e., a single vertex) and continues by examining all cliques in the graph until a clique with the maximum weight is found. The systematic search is done through a tree traversal. Every node of the search tree corresponds to a subgraph induced by the union of a clique C and a list of vertices, called *candidate list*. The candidate list is composed of the vertices in $V \backslash C$ that are adjacent to all vertices in C, so appending any of them to C would result in a larger, and hence heavier, clique. A node is pruned (i.e., the corresponding subgraph is excluded from further investigation) if an upper bound on the maximum clique-weight in the corresponding subgraph is not greater than a known solution. If a node is not pruned, it branches by expanding the incumbent clique via appending a vertex from the candidate list to it and so generating a new node of the tree. In Section 3.2, we present details of the algorithm, including the pruning subroutine for the MEWC problem.

3.1. Upper-Bounding Method

Suppose $\bar{\omega} \geq \omega(G)$, an upper bound on the clique number of an edge-weighted graph $G = (V, E, w^E)$, is known. Then, sum of the $\binom{\bar{\omega}}{2}$ largest edge weights of

G gives a trivial upper bound for the MEWC problem. This approach is disadvantageous as it ignores the structure of the graph beyond the knowledge of $\bar{\omega}$. To obtain a better bound, we need to take into account the adjacency of edges.

Suppose C^* is a maximum edge weight clique (MEWC) of G. For every vertex $i \in V$, let

$$\Gamma(i) = \begin{cases} \frac{1}{2}\sum_{j\in C^*\setminus\{i\}} w_{ij} & i \in C^* \\ 0 & i \in V\setminus C^*. \end{cases} \tag{13}$$

Then, $W^* = W(C^*)$ can be expressed in terms of attributes of the vertices rather than edges, that is,

$$W(C^*) = \sum_{i\in C^*} \Gamma(i). \tag{14}$$

Since $|C^*| \le \omega \le \bar{\omega}$, the summations in Equations (13) and (14) are over at most $\bar{\omega}-1$ and $\bar{\omega}$ vertices, respectively. Therefore, $\Gamma(i)$ and $W(C^*)$ are bounded from above according to Equations (15) and (16):

$$\Gamma(i) \le \bar{\Gamma}(i) = \frac{1}{2}\sum_{k=1}^{\bar{\omega}-1} w_e^{(k)} : e \in \delta_i \quad \forall i \in V, \tag{15}$$

$$W(C^*) \le \sum_{k=1}^{\bar{\omega}} \bar{\Gamma}^{(k)}, \tag{16}$$

where δ_i is the set of edges incident to a vertex $i \in V$ and the superscript (k) denotes the kth largest value in the corresponding set. Note that computational complexity of calculating this upper bound on $W(C^*)$ is $\mathbb{O}(m \log n)$, which is no worse than sorting the edges according to their weights.

We refine Equations (15) and (16) further based on the special structure of the subgraphs visited throughout the search procedure. Recall that each subgraph is induced by the union of an incumbent clique C and its candidate list L, denoted by $G[C \cup L]$. Consider G', the minor of $G[C \cup L]$ generated by merging all vertices of C into a single vertex v. In this graph, the weights of the edges connecting v to vertices of L are defined as follows:

$$w'_{iv} = \sum_{j\in C} w_{ij}, \forall i \in L. \tag{17}$$

Clearly, the difference between the maximum edge weights of cliques in $G[C \cup L]$ and G' is equal to $W(C)$. Hence, we may look for an upper bound on the optimal value of the MEWC problem on G', which we denote by $W^*_{G'}$. Since v is adjacent to all vertices of L in G', we can restate Equation (15) as follows:

$$\Gamma'(i) \le \bar{\Gamma}'(i) = w'_{iv} + \frac{1}{2}\sum_{k=1}^{\bar{\omega}_L-1} w_e^{(k)} : e \in \delta_i^L \quad \forall i \in L, \tag{18}$$

where $\Gamma'(i)$ is the contribution of a vertex $i \in L$ to $W^*_{G'}$, $\bar{\omega}_L$ is an upper bound on the clique number of $G'[L] = G[L]$, and δ_i^L denotes the set of edges incident on a vertex $i \in L$ in $G[L]$. Therefore,

$$W^*_{G'} \le \sum_{k=1}^{\bar{\omega}_L} \bar{\Gamma}'^{(k)}. \tag{19}$$

This leads to the following bound, which is employed in our algorithm:

$$W^*_{G[C\cup L]} \le W(C) + \sum_{k=1}^{\bar{\omega}_L} \bar{\Gamma}'^{(k)}. \tag{20}$$

To calculate $\bar{\omega}_L$ in our algorithm, we consider two chromatic methods from the literature that underly recent advancements in exact algorithms of the MC problem: the method proposed by Tomita and Kameda (2007), and the one by San Segundo et al. (2015). The computational effort required by these methods is comparable to that of analytic bounds while results of experiments with benchmark instances have shown that they usually generate much tighter upper bounds on the clique number. High-quality upper bounds, such as Lovász number (Lovász 1979) and *η-bound* (Bulò and Pelillo 2010), on the other hand, require solving computationally expensive optimization problems and are not suited for B&B methods.

The first chromatic method that we consider is a sequential vertex-coloring heuristic based on a preordered list of vertices. It is well known that any proper coloring of vertices (i.e., assignment of colors such that every two adjacent vertices have different colors) generates an upper bound on the clique number of a graph. Although such upper bounds could be arbitrarily loose for special graphs (Mycielski 1955), several algorithms for the MC problem have successfully used this method (Östergård 2002, Tomita and Seki 2003, Tomita and Kameda 2007, Tomita et al. 2010, Batsyn et al. 2014). The second method is based on a MaxSAT encoding of the MC problem that was originally proposed by Li and Quan (2010). We use the implementation of this method presented by San Segundo et al. (2015) and refer to it as a *chromatic method*, as it was presented in the context of graph coloring without the overhead of an explicit MaxSAT encoding. It has been shown by San Segundo et al. (2015) that this bound could be even better than the chromatic number (i.e., the minimum number of colors in a proper coloring) of the graph. For the MC problem, the algorithm that employs the second method generally outperforms the ones that use the first method and its improvements (e.g., Tomita et al. 2010 and Batsyn et al. 2014). However, this is not the case for the MEWC problem, as our experimental results show, because of different pruning criteria used for these two problems and the overhead of calculating the pruning threshold in the latter method.

3.2. Algorithm

The algorithm starts with building two arrays of vertices, namely U and L, in the initialization step. The vertices are sorted according to an initial ordering and stored in array U. This ordering aims to keep the subproblems formed during the B&B process small. Vertex array L is a copy of U, in which the vertices are colored based on their positions in the array. These two arrays are then used by the Branch function, which is the main procedure of the combinatorial B&B algorithm. All input arguments of this function are global variables and are updated as the search proceeds. Algorithm 1 outlines the main procedure.

Algorithm 1 (Main Method)

```
1: function SOLVE MEWC($G$)
2:     $C = \emptyset; W = 0$
3:     $C^* = \emptyset; W^* = 0$
4:     $(U, L) = \text{Initialize}(G)$
5:     $\text{Branch}(G, U, L, C, C^*, W, W^*)$
6:     return $(C^*, W^*)$
7: end function
```

The Initialize function (see Algorithm B.1 in Appendix B) stores the vertices in array U such that the kth vertex in U, denoted by $U[k]$, has the smallest degree in the subgraph induced by $\{U[1], U[2], \cdots, U[k]\}$. If there are two (or more) vertices with minimum degree in that subgraph, their relative order in U is determined based on degrees of their neighbors. Let G^k denote the subgraph induced by the first k vertices in U. If more than one vertex in G^k have the minimum degree, then $U[k]$ will be the vertex with the minimum sum of neighbors' degrees, denoted by σ in Algorithm B.1. The vertex $U[k]$ is chosen arbitrarily if there is still a tie among some vertices. This method of initialization was presented as a part of the MCR algorithm (Tomita and Kameda 2007) and has been used in almost all succeeding combinatorial B&B algorithms for the MC problem, but its main idea was first proposed by Carraghan and Pardalos (1990). Initial coloring of the vertices follows the fact that the number of colors required to properly color a graph is no more than the maximum degree of the vertices plus one. Therefore, the number of colors to properly color G^k will be no more than $\min\{k, \Delta(G) + 1\}$, where $\Delta(G)$ denotes the maximum degree of a vertex in G. The Initialize function gives distinct colors to the first $\Delta(G) + 1$ vertices in the sorted array and repeats the last color for all the remaining vertices. Note that the initial coloring is not necessarily proper, but the color (i.e., natural number) assigned to the vertex $L[k]$ for every $k \in \{1, \cdots, n\}$ is indeed an upper bound on the clique number of G^k.

The tree search is done by the Branch function in a recursive manner. In every call, this function inputs an incumbent clique C and its weight W, the best-known solution C^* and its weight W^*, and two lists of candidate vertices to expand the incumbent clique, denoted by U^p and L^p. Initially, C and C^* are empty, W and W^* are zero, and the candidate lists involve all vertices of the graph (i.e., U and L). Similar to U and L, the candidate lists at each node of the search tree (i.e., subproblems of the Branch function) are comprised of the same set of vertices. Vertices in U^p are uncolored and keep the relative order of vertices in U. Vertices in L^p are colored and reordered based on the subproblem coloring. Actually, the recursion can be implemented using a single array of candidate vertices, but experimental results on the MC problem have shown that keeping the initial ordering would generate, on average, tighter bounds on the clique number (Tomita et al. 2010, San Segundo et al. 2011). Implementation details of this function are presented in Algorithm 2.

Algorithm 2 (Combinatorial B&B Procedure)

```
 1: function Branch($G, U^p, L^p, C, C^*, W, W^*$)
 2:     while $U^p \neq \emptyset$ do
 3:         $q = \text{Prune}(G, U^p, L^p, C, W, W^*)$
 4:         if $q = $ true then                ▷ the node is pruned
 5:             return
 6:         else                                ▷ the node is not pruned
 7:             $v \leftarrow$ last vertex in $L^p$
 8:             $W_v = \sum_{u \in C} w_{uv}$
 9:             $C \leftarrow C \cup \{v\}$
10:             $W \leftarrow W + W_v$
11:             $U_v$ = an array of neighbors of $v$ in $U^p$ with
                    the same relative order
12:             if $U_v \neq \emptyset$ then
13:                 $L_v = \text{Subcolor}^{\bullet}(\cdots)$
14:                 $\text{Branch}(G, U_v, L_v, C, C^*, W, W^*)$
15:             else if $W > W^*$ then
16:                 $C^* \leftarrow C;\ W^* \leftarrow W$
17:             end if
18:             $C \leftarrow C \setminus \{v\}$
19:             $W \leftarrow W - W_v$
20:         end if
21:         $L^p \leftarrow L^p \setminus \{v\}$
22:         $U^p \leftarrow U^p \setminus \{v\}$
23:     end while
24:     return
25: end function
```

The Branch function operates as follows: first, it calls the Prune function to determine if the corresponding subgraph should be processed further. If so, the last vertex in array L^p is added to C, resulting in a larger clique. Then, a new subproblem is formed based on this new clique. The uncolored candidate list of the new subproblem, denoted by U_v, is comprised of all vertices in the parent candidate list U^p that are connected to the newly added vertex. Array U_v being empty implies that the current clique is maximal, thus its weight is compared against W^* and the better solution is kept. Otherwise, the Branch function calls itself on the new subproblem. Prior to this, the colored candidate list for the new subproblem, denoted by L_v, is generated through the Subcolor function. The two chromatic methods that we consider require different implementations of the Subcolor function. That is why we used a superscript • for this function in Algorithm 2 (line 13) to indicate a template function. We represent the functions corresponding to the first and second chromatic methods by superscripts 1 and 2, respectively.

The function Subcolor^1 (see Algorithm B.2 in Appendix B) implements the heuristic graph coloring method proposed by Tomita and Kameda (2007). This function inputs U_v and partitions it into the union of K independent sets. A proper coloring of $G[U_v]$ is attained by giving a distinct color to each independent set I_k for every $k \in \{1, \cdots, K\}$. The vertices are then stored in array L_v in a nondecreasing order of their colors. As a result, the color of the ith vertex in L_v is an upper bound on the clique number of the subgraph induced by the first i vertices in U_v. In particular, the color of the last vertex in L_v is an upper bound on the clique number of $G[U_v]$.

An improvement of this method was later presented by Tomita et al. (2010) via introducing a so-called "recoloring" procedure. More recently, San Segundo et al. (2015) proposed a novel recoloring method based on the MaxSAT encoding of the MC problem. We will also use this method to draw an upper bound on the clique number, and provide its details via Subcolor^2 function. But, we first present the pruning subroutine of our algorithm based on the upper-bounding method presented in Section 3.1.

In addition to computing the bounds on the clique number using Subcolor^1 and Subcolor^2 procedures, we also considered the sequential elimination method proposed by Gendron et al. (2008). However, the results of numerical experiments have shown that applying this method to enhance the bound on the clique number increases the CPU time spent on execution of the proposed B&B algorithm and hence are not reported here. The inferior performance can be explained by the fact that applying the sequential elimination algorithm increases the running time taken to compute the upper bound on the clique number by the factor of $\mathcal{O}(n^2)$ for a graph on n vertices.

Algorithm 3 (Pruning Procedure)

```
 1: function PRUNE($G, U^p, L^p, C, W, W^*$)
 2:     $q$ = false; $W' = 0$
 3:     $\bar{\omega}$ = color of the last vertex in $L^p$
 4:     $\Gamma$ = an array of size $|U^p|$ with all elements equal
            to zero
 5:     for $i = 1$ to $|U^p|$ do
 6:         $v \leftarrow U^p[i]$
 7:         for $j = 1$ to $|C|$ do
 8:             $u \leftarrow C[j]$; $\Gamma[i] \leftarrow \Gamma[i] + w_{uv}$
 9:         end for
10:        $\delta^i$ = an empty array of size $|U^p|$
11:        for $j = 1$ to $|U^p|$ do
12:            $u \leftarrow U^p[j]$
13:            if $\{u, v\} \in E$ then $\delta^i[j] = 1/2w_{uv}$
14:            else $\delta^i[j] = 0$
15:            end if
16:        end for
17:        sort $\delta^i$ in a nonincreasing order of elements
18:        for $k = 1$ to $\bar{\omega} - 1$ do
19:            $\Gamma[i] \leftarrow \Gamma[i] + \delta^i[k]$
20:        end for
21:    end for
22:    sort $\Gamma$ in a nonincreasing order of elements
23:    for $k = 1$ to $\bar{\omega}$ do
24:        $W' \leftarrow W' + \Gamma[k]$
25:    end for
26:    if $W' \leq W^* - W$ then $q$ = true
27:    end if
28:    return $q$
29: end function
```

Algorithm 3 demonstrates the pruning process. The Prune function operates as follows. The color of the last vertex in L^p gives an upper bound $\bar{\omega}$ on the clique number of the subgraph induced by the candidate list U^p. For every $i \in U^p$, $\bar{\Gamma}'(i)$ is calculated according to Equation (18) and is stored in array Γ (lines 4–21 of Algorithm 3). This array is then sorted in a nonincreasing order of elements, and the sum of its first $\bar{\omega}$ components is taken as W'. The corresponding subgraph is excluded from further investigation if $W + W'$ is not strictly larger than the weight of the best-known solution, i.e., W^*.

We conclude this section with details of the second chromatic method. A major parameter in implementation of the method proposed by San Segundo et al. (2015) is the pruning threshold, denoted by T in Algorithm 4. The pruning threshold is the maximum number of colors that can be used in a proper coloring of vertices such that the corresponding node is pruned. This method aims to exclude some vertices from the vertex-coloring process when the number of colors exceeds the pruning threshold. In an ideal case, the number of colors remains at the threshold and the node is pruned. Algorithm 4 demonstrates details of calculating the pruning threshold for the MEWC problem in our algorithm.

Algorithm 4 (Pruning Threshold for the MEWC Problem)

```
1: function THRESHOLD($G, U_v, C, W, W^*$)
2:     $\Gamma$ = an array of size $|U_v|$ with all elements equal
           to zero
3:     for $i = 1$ to $|U_v|$ do
```

```
4:       initialize $\Gamma[i]$ according to lines 7-9 of Algorithm 3
5:       form array $\delta^i$ according to lines 10-17 of Algorithm 3
6:    **end for**
7:    $W_t = \max\{\Gamma[i] : 1 \le i \le |U_v|\}$
8:    **if** $W_t > W^* - W$ **then** $T \leftarrow 0$
9:    **else**
10:      $t = 0$; $W_t = 0$
11:      **while** $W_t \le W^* - W$ **do**
12:         $t \leftarrow t + 1$
13:         **if** $t = |U_v|$ **then** go to line 28
14:         **else**
15:            **for** $i = 1$ to $|U_v|$ **do**
16:               $\Gamma[i] \leftarrow \Gamma[i] + \delta^i[t]$
17:            **end for**
18:            $\tilde{\Gamma} \leftarrow \Gamma$
19:           sort $\tilde{\Gamma}$ in a nonincreasing order of elements
20:            $W_t \leftarrow 0$
21:            **for** $k = 1$ to $t + 1$ **do**
22:               $W_t \leftarrow W_t + \tilde{\Gamma}[k]$
23:            **end for**
24:         **end if**
25:      **end while**
26:      $T \leftarrow t$
27:   **end if**
28:   **return** $T$
29: **end function**
```

Let $W'_t = \sum_{k=1}^{t} \bar{\Gamma}'^{(k)}$, then the threshold of our pruning method is given by $T = \operatorname{argmax}\{t \mid W'_t \le W^* - W\}$, where W^* is the weight of the best-known solution and W is the weight of the incumbent clique. As shown by Algorithm 4, computing T involves several iterations of a time-consuming part of the pruning process, i.e., updating and sorting contributions of vertices. Note that we are not replacing this part of Algorithm 3 with a simple comparison between the pruning threshold and the color of the last vertex in the candidate list. In fact, the PRUNE function may be called several times at every node of the search tree. The best-known solution may be updated in between these calls, which can result in pruning the subsequent children of the node.

Given a pruning threshold T, the SUBCOLOR² function (see Algorithm B.3 in Appendix B) operates as follows: consider a proper coloring of the graph (i.e., a partition of vertices into independent sets). A set of k colors is called *inconsistent* if there is no clique of cardinality k in the graph with one vertex in each set. If the number of colors is greater than the clique number, then there exists a color that forms an inconsistent set with some others. The procedure starts by assigning at most T colors to vertices such that each color identifies an independent set. If some vertices are still left, every time that a color $s \ge T + 1$ is assigned to an uncolored vertex v, the algorithm looks for two colors $k_1, k_2 \le T$ that form an inconsistent set with s (see Algorithm B.4 in Appendix B). If such a set is found, k_1 and k_2 are marked as *forbidden* and are not considered to find more inconsistent sets. More importantly, vertex v is excluded from the assigned color set. After all vertices are processed, the colored ones are placed in array L^p in a nondecreasing order of colors. Note that under this coloring method, the arrays U^p and L^p do not necessarily contain the same set of vertices.

4. Computational Experiments

In this section, we present our computational results on 41 benchmark instances taken from the DIMACS data set (Johnson and Trick 1996). DIMACS instances are originally unweighted. Pullan (2008) proposed an edge-weighted version of these graphs, named DIMACS-EW, which has been used in later works on this problem. The weight of an edge $\{i,j\} \in E$ in DIMACS-EW graphs is given by $w_{ij} = (i + j) \bmod 200 + 1$. Table 1 presents characteristics of the test instances. In this table, $|V|$, $|E|$ and "Density" denote number of vertices, number of edges, and edge density of the graph, respectively.

All algorithms proposed in this paper were implemented in C++, compiled using Visual C++ and executed on an Intel®Core™ i7 CPU @ 2.90 GHz, 8 GB RAM computer with Windows 8.1 OS. The spectral upper bounds on the clique number were computed using Intel Math Kernel Library on the same system.

Table 2 provides results concerning the quality of the Lagrangian relaxation bound, i.e., a in Proposition 1, along with other analytic bounds proposed in the literature, as well as the coloring-based bound of Tomita and Kameda (2007). In addition, Table 3 reports the corresponding CPU times (in milliseconds). As mentioned earlier, three of these bounds require spectral calculations on the adjacency matrix of the graph (or its complement). Given the difference in the required computational effort, we distinguish them from the first bound of Amin and Hakimi (1972) and the Lagrangian bound proposed in this paper. In Tables 2 and 3, "Wilf," "A&H," and "Budinich" refer to the bounds proposed by Wilf (1967), Amin and Hakimi (1972), and Budinich (2003), respectively. The column "LRB" shows the results for the Lagrangian relaxation bound presented in Section 2. Finally, the column "T&K" reports results for the coloring-based bounds according to Tomita and Kameda (2007).

It is interesting to note that, with the exception of c-fat and p-hat families that mostly involve sparse graphs, the Lagrangian relaxation bound is relatively close to the

Table 1. Instances Characteristics

Name	$\|V\|$	$\|E\|$	Density	Name	$\|V\|$	$\|E\|$	Density
brock200-1	200	14,834	0.75	johnson8-2-4	28	210	0.56
brock200-2	200	9,876	0.50	johnson8-4-4	70	1,855	0.77
brock200-3	200	12,048	0.61	keller4	171	9,435	0.65
brock200-4	200	13,089	0.66	MANN-a9	45	918	0.93
C125.9	125	6,963	0.90	p-hat300-1	300	10,933	0.24
C250.9	250	27,984	0.90	p-hat300-2	300	21,928	0.49
c-fat200-1	200	1,534	0.08	p-hat300-3	300	33,390	0.74
c-fat200-2	200	3,235	0.16	p-hat500-1	500	31,569	0.25
c-fat200-5	200	8,473	0.43	p-hat500-2	500	62,946	0.50
c-fat500-1	500	4,459	0.04	p-hat700-1	700	60,999	0.25
c-fat500-10	500	46,627	0.37	p-hat700-2	700	121,728	0.50
c-fat500-2	500	9,139	0.07	san200-0.7-1	200	13,930	0.70
c-fat500-5	500	23,191	0.19	san200-0.7-2	200	13,930	0.70
DSJC500-5	500	62,624	0.50	san200-0.9-1	200	17,910	0.90
gen200-p0.9-44	200	17,910	0.90	san200-0.9-2	200	17,910	0.90
gen200-p0.9-55	200	17,910	0.90	san200-0.9-3	200	17,910	0.90
hamming6-2	64	1,824	0.91	san400-0.5-1	400	39,900	0.50
hamming6-4	64	704	0.35	sanr200-0.7	200	13,868	0.70
hamming8-2	256	31,616	0.97	sanr200-0.9	200	17,863	0.90
hamming8-4	256	20,864	0.64	sanr400-0.5	400	39,984	0.50
johnson16-2-4	120	5,460	0.77				

spectral bound proposed by Wilf (1967), that is, $\rho(A_G)+1$. Aside from these two families, the Lagrangian relaxation bound is better than the first bound of Amin and Hakimi (1972) on all the other graphs, with the same time complexity.

Next, we compare the performance of the presented algorithm for the MEWC problem with the results reported by Gouveia and Martins (2015) and Hosseinian et al. (2018), as well as the EWCLIQUE algorithm by Shimizu et al. (2019). The first two works have considered instances with no more than 300 vertices from brock, C, c-fat, hamming, johnson, keller, MANN, and p-hat families and applied an upper time limit of three hours to solve each instance. Shimizu et al. (2019) report results for 34 of the DIMACS instances, with a 1,000-second time limit. Our experiment involves all families of DIMACS-EW graphs. In addition to all the graphs with up to 300 vertices, we consider instances with up to 700 vertices with edge density no larger than 0.50 and use a three-hour time constraint for each instance.

Table 4 summarizes the recorded solution time. In this table, W^* shows the optimal solution value of the MEWC problem (if known) for each instance. The column "IP" provides the results of Gouveia and Martins (2015). They report the solution time of nine different integer programming models for the MEWC problem using the ILOG/CPLEX 11.2 solver on an Intel Core i7 CPU @ 3.40 GHz and 8 GB RAM. We consider the best model for each instance among them in our comparison. The column "CBQ" shows the computational results of the algorithm presented in Hosseinian et al. (2018), using the same system used to run the algorithms proposed in this paper. The column "EWC" presents the results for EWCLIQUE algorithm reported in Shimizu et al. (2019). The results for EWC were copied from Shimizu et al. (2019) and were obtained using a computer running Linux 4.4.0 OS with Intel Core i7-6700 CPU @ 3.40 GHz and 16 GB RAM. Shimizu et al. (2019) implemented their EWCLIQUE algorithm in C++ and used g++ 5.4.0 compiler with optimization option -O2. The last two columns concern the results of the algorithm presented in this paper. The column "Algorithm(1)" shows the solution time when the SUBCOLOR^1 function was used in the algorithm and "Algorithm(2)" corresponds to application of SUBCOLOR^2. We have used "t-lim" in this table to indicate that the corresponding algorithm was terminated due to time limit before completing the search. We have run our algorithm with the initial lower bound used in Hosseinian et al. (2018). The reported results in Table 4 include the computing time of the initial lower bounds.

We could solve 36 out of 41 instances within three hours. The only instance (among successful ones) that took more than one hour for our method to solve was san200-0.9-3. Our solution times are an order of magnitude better than the best results obtained by IP and CBQ methods for instances considered by all the three methods (except for small graphs, which are solved quickly by all methods). In comparison of our method with EWCLIQUE, neither performs uniformly better than the other. However, on the 30 instances solved by both methods, the total time

Table 2. Analytic and Coloring-Based Bounds on the Clique Number

		Spectral			Linear		Coloring
Name	$\omega(G)$	Wilf	A&H	Budinich	A&H	LRB	T&K
brock200-1	21	149	98	100	172	157	56
brock200-2	12	100	95	100	140	105	35
brock200-3	15	121	98	100	155	127	43
brock200-4	17	132	96	100	162	139	48
C125.9	34	112	63	62	118	115	55
C250.9	44	224	124	125	237	230	98
c-fat200-1	12	17	88	181	53	136	15
c-fat200-2	24	33	95	192	79	129	24
c-fat200-5	58	85	87	196	130	107	84
c-fat500-1	14	20	244	428	90	347	14
c-fat500-10	126	187	221	477	305	280	126
c-fat500-2	26	38	217	449	132	340	26
c-fat500-5	64	93	222	453	214	319	64
DSJC500-5	13	251	245	250	353	259	70
gen200-p0.9-44	44	180	96	100	189	184	62
gen200-p0.9-55	55	180	97	100	189	184	72
hamming6-2	32	58	33	42	60	59	37
hamming6-4	4	22	28	32	37	36	10
hamming8-2	128	248	131	163	251	249	136
hamming8-4	16	163	121	128	204	173	34
johnson16-2-4	8	92	16	60	104	96	14
johnson8-2-4	4	15	8	14	20	16	6
johnson8-4-4	14	53	28	35	61	56	19
keller4	11	111	63	85	137	117	34
MANN-a9	16	41	20	22	43	42	21
p-hat300-1	8	80	139	150	147	184	24
p-hat300-2	25	158	145	150	209	182	46
p-hat300-3	36	225	147	150	258	235	73
p-hat500-1	9	137	238	250	250	305	36
p-hat500-2	36	272	243	250	354	308	70
p-hat700-1	11	190	335	350	348	429	44
p-hat700-2	44	377	343	350	493	430	91
san200-0.7-1	30	140	95	100	167	148	30
san200-0.7-2	18	143	77	100	167	148	18
san200-0.9-1	70	180	98	100	189	184	70
san200-0.9-2	60	180	98	100	189	184	60
san200-0.9-3	44	180	96	100	189	184	44
san400-0.5-1	13	202	176	200	282	212	13
sanr200-0.7	18	139	97	100	166	147	50
sanr200-0.9	42	179	99	100	189	184	79
sanr400-0.5	13	201	196	200	282	208	59

to solve all these 30 instances taken by EWCLIQUE, Algorithm(1), and Algorithm(2) was 1,516.54, 837.866, and 1,122.228 seconds, respectively. The difference is mainly due to more challenging instances of this set. In fact, for most instances that the reported solution time of EWCLIQUE is less than Algorithm(1), the solution time for both algorithms is less than 10 seconds. Regarding the two vertex-coloring routines employed in our method, the algorithm performed uniformly better when the method of Tomita and Kameda (2007) was used in the pruning process. The difference is noticeable for instances that took more time to solve; see, for example, the results corresponding to san200-0.9-3.

Our best time improvements were obtained for c-fat200-5, C125.9, and san family instances. The CBQ algorithm failed to solve c-fat200-5 within three hours whereas it was solved by one of the integer programming models of Gouveia and Martins (2015) in about 155 seconds, and by EWCLIQUE in 74.31 seconds. We solved this instance in 0.077 seconds. All nine integer programming models of Gouveia and Martins (2015) failed to solve C125.9, whereas we solved this instance 42 times faster than the CBQ algorithm. For the five instances of the san family considered by our method and EWCLIQUE, the total time taken by Algorithm(1) to solve all these five instances was 264.089 seconds, whereas it took 979.15

Table 3. CPU Times (in Milliseconds) for the Considered Bounds on the Clique Number

	Spectral			Linear		Coloring
Name	Wilf	A&H	Budinich	A&H	LRB	T&K
brock200-1	29.687	23.437	23.437	0.000	0.000	3.125
brock200-2	25.000	23.437	21.875	0.000	0.000	3.125
brock200-3	23.437	23.437	20.312	1.562	0.000	1.562
brock200-4	23.437	23.437	23.437	0.000	1.562	1.562
C125.9	7.812	6.250	7.812	0.000	0.000	0.000
C250.9	40.625	43.750	40.625	0.000	0.000	3.125
c-fat200-1	17.187	17.187	21.875	0.000	0.000	0.000
c-fat200-2	10.937	12.500	23.437	0.000	1.562	1.562
c-fat200-5	15.625	17.187	25.000	0.000	0.000	3.125
c-fat500-1	151.563	142.188	232.813	1.562	0.000	6.250
c-fat500-10	168.750	164.063	292.188	0.000	0.000	32.812
c-fat500-2	142.188	142.188	273.438	1.562	0.000	7.812
c-fat500-5	137.500	139.063	245.313	1.562	0.000	17.187
DSJC500-5	267.188	260.938	257.813	4.688	3.125	17.187
gen200-p0.9-44	23.437	21.875	21.875	1.562	0.000	1.562
gen200-p0.9-55	23.437	23.437	21.875	1.562	0.000	1.562
hamming6-2	1.562	1.562	1.562	0.000	0.000	0.000
hamming6-4	1.562	1.562	1.562	0.000	0.000	0.000
hamming8-2	42.187	42.187	43.750	0.000	1.562	21.875
hamming8-4	42.187	40.625	39.062	0.000	1.562	14.062
johnson16-2-4	4.687	4.687	6.250	0.000	0.000	3.125
johnson8-2-4	0.000	0.000	0.000	0.000	0.000	0.000
johnson8-4-4	1.562	1.562	1.562	0.000	0.000	0.000
keller4	17.187	18.750	20.312	0.000	0.000	3.125
MANN-a9	0.000	1.562	0.000	0.000	0.000	0.000
p-hat300-1	64.062	64.062	64.062	1.562	0.000	4.687
p-hat300-2	65.625	65.625	68.750	1.562	1.562	4.687
p-hat300-3	67.187	67.187	65.625	1.562	0.000	6.250
p-hat500-1	262.500	264.063	257.813	1.562	3.125	14.062
p-hat500-2	264.063	262.500	262.500	3.125	3.125	15.625
p-hat700-1	667.188	667.188	675.000	3.125	4.687	29.687
p-hat700-2	668.750	667.188	667.188	6.250	6.250	29.687
san200-0.7-1	20.312	23.437	23.437	0.000	1.562	3.125
san200-0.7-2	35.937	34.375	37.500	0.000	0.000	4.687
san200-0.9-1	23.437	21.875	23.437	0.000	0.000	4.687
san200-0.9-2	21.875	25.000	23.437	0.000	0.000	3.125
san200-0.9-3	23.437	23.437	25.000	0.000	0.000	3.125
san400-0.5-1	135.938	143.750	135.938	1.562	1.562	18.750
sanr200-0.7	23.437	23.437	23.437	0.000	1.562	1.562
sanr200-0.9	25.000	21.875	23.437	0.000	0.000	1.562
sanr400-0.5	140.625	139.063	142.188	1.562	3.125	10.937

seconds for EWCLIQUE to solve them. It should be noted that both Shimizu et al. (2019) and Gouveia and Martins (2015) used computers with slightly faster processors (3.40 GHz) than that of the machine we used for our experiments (2.90 GHz).

Among the five solution methods in Table 4, four are combinatorial B&B algorithms, three of which (EWC being an exception) use the same initial lower bounds and vertex ordering. For each of these methods, we present the number of nodes of the corresponding search tree in Table 5. We exclude from this table the five instances that we could not solve. As can be seen in this table, the number of B&B nodes in either version of the algorithm presented in this paper is much less than that of the CBQ and EWCLIQUE algorithms, with one exception for each CBQ and EWCLIQUE. Namely, CBQ uses fewer nodes for johnson8-2-4, which is the smallest graph among the test instances; and EWCLIQUE uses fewer nodes for hamming6-4. Besides, the number of B&B nodes under the second chromatic method in our algorithm is less than (or equal to) the first method for all but one of the considered instances, san200-0.9-3. A better upper bound is expected, but is not guaranteed, to yield a smaller number of B&B nodes, as the effect of the bound's quality on the structure of the B&B tree is difficult to predict. Along with the results of Table 4, this implies that better quality of the upper

Table 4. Solution Time for the MEWC Problem

		CPU (sec.)				
Name	W^*	IP	CBQ	EWC	Algorithm(1)	Algorithm(2)
brock200-1	21,230	t-lim	3,047.565	338.31	196.359	282.281
brock200-2	6,542	9,464.240	7.436	0.10	0.765	0.983
brock200-3	10,303	t-lim	55.905	1.27	5.140	6.733
brock200-4	13,967	t-lim	188.031	4.84	16.311	21.765
C125.9	66,248	t-lim	4,558.170	—	108.109	173.719
C250.9	—	t-lim	t-lim	—	t-lim	t-lim
c-fat200-1	7,734	3.870	0.483	< 0.01	0.030	0.046
c-fat200-2	26,389	33.260	0.890	< 0.01	0.046	0.061
c-fat200-5	168,200	155.300	t-lim	74.31	0.077	0.124
c-fat500-1	10,738	—	—	< 0.01	0.343	0.390
c-fat500-10	804,000	—	—	t-lim	723.406	1,357.550
c-fat500-2	38,350	—	—	< 0.01	0.374	0.421
c-fat500-5	205,864	—	—	0.43	0.702	0.905
DSJC500-5	9,626	—	—	44.43	204.109	254.406
gen200-p0.9-44	—	—	—	—	t-lim	t-lim
gen200-p0.9-55	150,839	—	—	—	286.140	429.234
hamming6-2	32,736	0.300	4.437	< 0.01	0.015	0.031
hamming6-4	396	1.970	0.031	< 0.01	0.000	0.000
hamming8-2	800,624	t-lim	t-lim	0.23	t-lim	t-lim
hamming8-4	12,360	t-lim	439.437	1.46	14.390	18.421
johnson16-2-4	3,808	t-lim	84.687	0.25	12.203	15.984
johnson8-2-4	192	0.140	0.000	< 0.01	0.000	0.000
johnson8-4-4	6,552	2.340	0.687	< 0.01	0.046	0.078
keller4	6,745	t-lim	42.218	0.70	2.233	2.952
MANN-a9	5,460	9.390	1.906	0.02	0.203	0.359
p-hat300-1	3,321	1,273.050	3.281	0.01	0.327	0.374
p-hat300-2	31,564	t-lim	171.281	42.90	10.468	14.984
p-hat300-3	63,390	t-lim	t-lim	—	3,576.471	5,152.831
p-hat500-1	4,764	—	—	0.13	2.296	2.671
p-hat500-2	63,870	—	—	—	1,023.799	1,519.049
p-hat700-1	5,185	—	—	0.52	9.280	10.921
p-hat700-2	—	—	—	—	t-lim	t-lim
san200-0.7-1	45,295	—	—	54.88	1.827	2.421
san200-0.7-2	15,073	—	—	17.86	0.718	0.858
san200-0.9-1	242,710	—	—	12.56	89.249	124.234
san200-0.9-2	178,468	—	—	833.49	160.765	215.656
san200-0.9-3	96,764	—	—	—	4,037.105	8,740.675
san400-0.5-1	7,442	—	—	60.36	11.530	15.187
sanr200-0.7	16,398	—	—	18.67	45.233	62.936
sanr200-0.9	—	—	—	—	t-lim	t-lim
sanr400-0.5	8,298	—	—	9.04	52.827	66.046

bound (on the clique number) generated by the second chromatic method did not make up for the overhead of calculating the pruning threshold for the MEWC problem.

5. Conclusion

In this paper, we used a Lagrangian relaxation of an integer (linear) programming formulation of the MEWC problem to derive a new analytic (i.e., closed-form) upper bound on the clique number of graphs. The presented bound is characterized by the degrees of the vertices in the graph. We compared this bound with other analytic bounds proposed in the literature. Our computational results showed that the Lagrangian relaxation bound is in general tighter than the bound introduced by Amin and Hakimi (1972), which was the only existing analytic upper bound on the clique number computable in linear time.

We also presented a new exact algorithm for the MEWC problem. The algorithm is a combinatorial B&B procedure that employs an efficient pruning method. At every node of the search tree, the algorithm uses an upper bound on the clique number of the corresponding subgraph to calculate an upper bound for the MEWC problem. The calculation of this bound is based on contribution of the vertices of a clique to its weight, rather than the edges of the induced subgraph. We presented computational results of our algorithm on some benchmark instances, and compared them with the existing results in the

Table 5. Size of the Search Tree

	Number of B&B nodes			
Name	CBQ	EWC	Algorithm(1)	Algorithm(2)
brock200-1	19,445,372	1,328,614,116	3,058,744	2,853,358
brock200-2	54,328	345,371	14,418	13,683
brock200-3	443,026	4,282,305	99,441	92,784
brock200-4	1,357,862	13,814,425	284,391	264,588
C125.9	13,276,576	—	1,022,493	1,018,056
c-fat200-1	186	632	186	186
c-fat200-2	2,431	6,780	178	178
c-fat200-5	—	138,193,445	176	176
c-fat500-1	—	1,605	486	486
c-fat500-10	—	—	586,791	586,791
c-fat500-2	—	4,679	472	472
c-fat500-5	—	1,227,023	750	750
DSJC500-5	—	200,152,687	2,994,499	2,856,281
gen200-p0.9-55	—	—	572,542	520,760
hamming6-2	16,134	896	223	213
hamming6-4	403	340	351	351
hamming8-4	2,568,843	2,475,100	155,100	145,634
johnson16-2-4	1,544,956	1,905,154	1,189,751	1,173,959
johnson8-2-4	113	150	127	127
johnson8-4-4	5,532	3,953	1,407	1,360
keller4	353,436	2,158,496	41,474	40,271
MANN-a9	40,407	116,041	26,159	26,159
p-hat300-1	11,278	50,151	3,939	3,862
p-hat300-2	854,903	134,486,327	114,726	110,475
p-hat300-3	—	—	23,035,890	21,907,679
p-hat500-1	—	468,371	25,159	23,945
p-hat500-2	—	—	6,211,268	5,893,921
p-hat700-1	—	1,678,557	113,239	108,136
san200-0.7-1	—	387,149,894	6,717	5,917
san200-0.7-2	—	48,732,878	3,983	3,765
san200-0.9-1	—	12,731,307	219,755	174,470
san200-0.9-2	—	303,169,816	377,156	303,253
san200-0.9-3	—	—	11,876,914	18,700,526
san400-0.5-1	—	43,132,933	41,090	40,953
sanr200-0.7	—	55,871,909	814,221	754,629
sanr400-0.5	—	36,003,126	928,826	881,902

literature. It was shown that the proposed algorithm outperforms the preceding solution methods of the MEWC problem due to effectiveness of its pruning process. Our solution times are at least an order of magnitude better than the best reported results for all nontrivial instances considered in previous works for this problem. The computational results also showed that recoloring approaches and sequential elimination enhancements to draw tight upper bounds on the clique number are not as effective as in the MC problem, due to the overhead of calculating the pruning threshold and enhanced bounds for the MEWC problem. At the same time, developing better coloring-based upper bounds on the clique number that exploit the structure of the colorings used may significantly speed up the proposed B&B algorithm, as is demonstrated by Shimizu et al. (2018). In addition, incorporating fast and effective heuristics could significantly improve the performance of the proposed approach, especially on dense graphs with large clique number, for which the classical MC algorithms tend to perform well.

Acknowledgments
The authors thank the three anonymous reviewers for their helpful suggestions.

Appendix A. The Univariate Lagrangian Relaxation Problem

Lemma A.1. *Suppose S is a set of objects, each of which has a weight $w_i > 0, \forall i \in S$. For every $x \geq 0$, let $S^+ = \{i \in S \mid w_i \geq 2x\}$. Then, for every positive integer N, the optimal value of the following optimization problem is always given by sum of the N largest weights of the objects in S:*

$$\min_{x \geq 0} \sum_{i \in S^+} (w_i - 2x) + 2\,Nx. \tag{A.1}$$

Proof. Let S_N denote the set of N objects with the largest weights in S, and $\tilde{w}$ be the average weight of the objects in S_N, i.e., $\tilde{w}{=}1/N\sum_{i\in S_N} w_i$. First, we show that

$N\tilde{w} \le \sum_{i\in S^+}(w_i - 2x) + 2Nx$. Patently, the inequality holds if $\tilde{w} \le 2x$. Hence, suppose $\tilde{w} > 2x$. Consider a partition of S_N as follows:

$$\begin{aligned} S_1 &= \{i \in S_N \mid \tilde{w} \le w_i\},\\ S_2 &= \{i \in S_N \mid 2x \le w_i < \tilde{w}\},\\ S_3 &= \{i \in S_N \mid w_i < 2x\}. \end{aligned} \tag{A.2}$$

Note that $S_1 \cup S_2 = S_N \cap S^+$. By definition of $\tilde{w}$, we have

$$\begin{aligned} \sum_{i\in S_1}(w_i - \tilde{w}) &= \sum_{i\in S_2}(\tilde{w} - w_i) + \sum_{i\in S_3}(\tilde{w} - w_i)\\ &= \sum_{i\in S_2}(\tilde{w} - w_i) + \sum_{i\in S_3}(\tilde{w} - 2x) + \sum_{i\in S_3}(2x - w_i)\\ &\ge \sum_{i\in S_2}(\tilde{w} - w_i) + \sum_{i\in S_3}(\tilde{w} - 2x), \end{aligned} \tag{A.3}$$

as $w_i < 2x, \forall i \in S_3$. Therefore,

$$\begin{aligned} \sum_{i\in S^+}(w_i - 2x) &\ge \sum_{i\in S_N\cap S^+}(w_i - 2x) = \sum_{i\in S_1}(w_i - 2x) + \sum_{i\in S_2}(w_i - 2x)\\ &= \sum_{i\in S_1}(w_i - \tilde{w} + \tilde{w} - 2x) + \sum_{i\in S_2}(w_i - \tilde{w} + \tilde{w} - 2x)\\ &\ge \sum_{i\in S_1}(\tilde{w} - 2x) + \sum_{i\in S_2}(\tilde{w} - 2x) + \sum_{i\in S_3}(\tilde{w} - 2x)\\ &= \sum_{i\in S_N}(\tilde{w} - 2x) = N(\tilde{w} - 2x), \end{aligned} \tag{A.4}$$

where the second inequality is due to Equation (A.3). Since the choice of x was arbitrary, this result implies that

$$N\tilde{w} \le \min_{x\ge 0} \sum_{i\in S^+}(w_i - 2x) + 2Nx. \tag{A.5}$$

Let x' be equal to one half of the smallest weight of an object in S_N. Then,

$$\begin{aligned} \sum_{i\in S^+}(w_i - 2x') + 2Nx' &= \sum_{i\in S_N}(w_i - 2x') + 2Nx'\\ &= \sum_{i\in S_N} w_i - 2Nx' + 2Nx' = N\tilde{w}. \end{aligned} \tag{A.6}$$

That is, by this choice of x, the univariate optimization problem reaches its lower bound. Hence, its optimal value is always given by $N\tilde{w} = \sum_{i\in S_N} w_i$. ☐

Appendix B. Algorithms

Algorithm B.1 (Initialization Step Proposed by Tomita and Kameda (2007))

```
1: function INITIALIZE($G$)
2:    $U$ = an empty array of size $n$
3:    for $k = n$ to 1 do                          ▷ initial sorting starts here
4:       $R \leftarrow$ 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 $i \in R$ do
9:             $\sigma(i) = \sum_{v\in N(i)} d_v$          ▷ $d_v$: degree of vertex $v$
10:         end for                               ▷ $N(i) = \{j \in V \mid \{i,j\} \in E\}$
11:         $u \leftarrow$ a vertex in $R$ with the minimum $\sigma$
12:      end if
13:      $U[k] \leftarrow u$
14:      $G \leftarrow G\backslash\{u\}$
15:   end for
16:   $L \leftarrow U$                                  ▷ initial coloring starts here
17:   for $k = 1$ to $\Delta(G)$ do
18:      color($L[k]$) $\leftarrow$ $k$
19:   end for
20:   for $k = \Delta(G) + 1$ to $n$ do
21:      color($L[k]$) $\leftarrow$ $\Delta(G) + 1$
22:   end for
23:   return $(U, L)$
24: end function
```

Algorithm B.2 (Vertex-Coloring Method Proposed by Tomita and Kameda (2007))

```
1: function Subcolor$^1$($G, U_v$)
2:    $L_v$ = an empty array of size $|U_v|$
3:    $K = 0$                          ▷ $K$: the number of colors used
4:    $I_1 = \emptyset$
5:    for $i = 1$ to $|U_v|$ do
6:       $u \leftarrow U_v[i]$
7:       $k = 1$
8:       while $N(u) \cap I_k \ne \emptyset$ do
9:          $k \leftarrow k + 1$
10:      end while
11:      if $k > K$ then
12:         $K \leftarrow k$
13:         $I_k = \emptyset$
14:      end if
15:      $I_k \leftarrow I_k \cup \{u\}$
16:   end for
17:   $i = 1$
18:   for $k = 1$ to $K$ do
19:      for $j = 1$ to $|I_k|$ do
20:         $L_v[i] \leftarrow I_k[j]$
21:         color($L_v[i]$) $\leftarrow k$
22:         $i \leftarrow i + 1$
23:      end for
24:   end for
25:   return $L_v$
26: end function
```

Algorithm B.3 (Recoloring Method Proposed by San Segundo et al. (2015))

```
1: function Subcolor$^2$ ($G, U_v, C, W, W^*$)
2:    $L_v$ = an empty array of size $|U_v|$
3:    $T$ = THRESHOLD($G, U_v, C, W, W^*$)
4:    $F = \emptyset$                        ▷ $F$: the set of forbidden colors
5:    $s = 1$; $I_1 = \emptyset$
6:    $Array \leftarrow U_v$
7:    while $Array \ne \emptyset$ do
8:       $I_s \leftarrow Array$
9:       while all vertices in $I_s$ have been selected do
10:         pick a vertex $v$ from $I_s$
11:         $result \leftarrow$ false
12:         if $s \ge T + 1$ then $result \leftarrow$ EXCLUDE($v, T, \{I_1, \cdots, I_T\}$,
             $I_s, F$)
```

```
13:         if $result$ = false then $I_s \leftarrow I_s \backslash N(v)$
14:         $Array \leftarrow Array \backslash \{v\}$
15:       end while
16:       $s \leftarrow s+1$; $I_s = \emptyset$
17:   end while
18:   $i = 1$
19:   for $k = 1$ to $s-1$ do
20:       for $j = 1$ to $|I_k|$ do
21:           $L_v[i] \leftarrow I_k[j]$
22:           color($L_v[i]$) $\leftarrow k$
23:           $i \leftarrow i+1$
24:       end for
25:   end for
26:   return $L_v$
27: end function
```

Algorithm B.4 (Vertex Exclusion Method (San Segundo et al. 2015))

```
 1: function EXCLUDE($v, T, \{I_1, \cdots, I_T\}, I_s, F$)
 2:   for $k_1 = 1$ to $T$ do
 3:       if $k_1 \notin F$ and $|I_{k_1} \cap N(v)| = 1$ then
 4:           $u \leftarrow I_{k_1} \cap N(v)$
 5:           for $k_2 = k_1 + 1$ to $T$ do
 6:               if $k_2 \notin F$ and $I_{k_2} \cap N(v) \cap N(u) = \emptyset$ then
 7:                   $F \leftarrow F \cup \{k_1\} \cup \{k_2\}$
 8:                   $I_s \leftarrow I_s \backslash \{v\}$
 9:                   return true
10:               end if
11:           end for
12:           for $k_2 = 1$ to $k_1 - 1$ do
13:               if $k_2 \notin F$ and $I_{k_2} \cap N(v) \cap N(u) = \emptyset$ then
14:                   $F \leftarrow F \cup \{k_1\} \cup \{k_2\}$
15:                   $I_s \leftarrow I_s \backslash \{v\}$
16:                   return true
17:               end if
18:           end for
19:       end if
20:   end for
21:   return false
22: end function
```

References

Akutsu T, Hayashida M, Tomita E, Suzuki J (2004) Protein threading with profiles and constraints. *Proc. 4th IEEE Sympos. Bioinformatics Bioengrg.* (IEEE Computer Society, Washington, DC), 537–544.

Alidaee B, Glover F, Kochenberger G, Wang H (2007) Solving the maximum edge weight clique problem via unconstrained quadratic programming. *Eur. J. Oper. Res.* 181(2):592–597.

Amin AT, Hakimi SL (1972) Upper bounds on the order of a clique of a graph. *SIAM J. Appl. Math.* 22(4):569–573.

Aringhieri R, Cordone R (2011) Comparing local search metaheuristics for the maximum diversity problem. *J. Oper. Res. Soc.* 62(2):266–280.

Batsyn M, Goldengorin B, Maslov E, Pardalos PM (2014) Improvements to MCS algorithm for the maximum clique problem. *J. Combin. Optim.* 27(2):397–416.

Bomze IM, Budinich M, Pardalos PM, Pelillo M (1999) The maximum clique problem. Du DZ, Pardalos PM, eds. *Handbook of Combinatorial Optimization* (Kluwer Academic Publishers, Dordrecht, Netherlands), 1–74.

Brown JB, Bahadur D, Tomita E, Akutsu T (2006) Multiple methods for protein side chain packing using maximum weight cliques. *Genome Informatics* 17(1):3–12.

Budinich M (2003) Exact bounds on the order of the maximum clique of a graph. *Discrete Appl. Math.* 127(3):535–543.

Bulò SR, Pelillo M (2010) A new spectral bound on the clique number of graphs. Hancock E, Wilson R, Windeatt T, Ulusoy I, Escolano F, eds. *Structural, Syntactic, and Statistical Pattern Recognition*, Lecture Notes in Computer Science, vol. 6218 (Springer, Berlin, Heidelberg), 680–689.

Carraghan R, Pardalos P (1990) An exact algorithm for the maximum clique problem. *Oper. Res. Lett.* 9(6):375–382.

Cavique L (2007) A scalable algorithm for the market basket analysis. *J. Retailing Consumer Services* 14(6):400–407.

de Andrade MRQ, de Andrade PMF, Martins SL, Plastino A (2005) GRASP with path-relinking for the maximum diversity problem. Nikoletseas SE, ed. *Experimental and Efficient Algorithms*, Programming and Software Engineering, vol. 3503 (Springer, Berlin, Heidelberg), 558–569.

Dijkhuizen G, Faigle U (1993) A cutting-plane approach to the edge-weighted maximal clique problem. *Eur. J. Oper. Res.* 69(1):121–130.

Fontes DBMM, Gonçalves JF, Fontes FACC (2018) An evolutionary approach to the maximum edge weight clique problem. *Recent Adv. Electr. Electronic Engrg.* 11(3):260–266.

Gallego M, Duarte A, Laguna M, Martí R (2007) Hybrid heuristics for the maximum diversity problem. *Comput. Optim. Appl.* 44(3): 411–426.

Gendron B, Hertz A, St-Louis P (2008) A sequential elimination algorithm for computing bounds on the clique number of a graph. *Discrete Optim.* 5(3):615–628.

Gouveia L, Martins P (2015) Solving the maximum edge-weight clique problem in sparse graphs with compact formulations. *EURO J. Comput. Optim.* 3(1):1–30.

Hosseinian S, Fontes DBMM, Butenko S (2016) A quadratic approach to the maximum edge weight clique problem. Rocha AMAC, Costa MFP, Fernandes EMGP, eds. *Proc. XIII Global Optim. Workshop* (University of Minho, Braga, Portugal), 125–128.

Hosseinian S, Fontes DBMM, Butenko S (2018) A nonconvex quadratic optimization approach to the maximum edge weight clique problem. *J Global Optim.* 72(2):219–240.

Hosseinian S, Fontes DBMM, Butenko S, Buongiorno-Nardelli M, Fornari M, Curtarolo S (2017) The maximum edge weight clique problem: Formulations and solution approaches. Butenko S, Pardalos PM, Shylo V, eds. *Optimization Methods and Applications: In Honor of Ivan V. Sergienko's 80th Birthday* (Springer International Publishing, Cham, Switzerland), 217–237.

Hunting M, Faigle U, Kern W (2001) A Lagrangian relaxation approach to the edge-weighted clique problem. *Eur. J. Oper. Res.* 131(1):119–131.

Jabbar MA, Deekshatulu BL, Chandra P (2013) Graph based approach for heart disease prediction. Das VV, ed. *Proc. 3rd Internat. Conf. Trends Inform. Telecomm. Comput.* (Springer, New York), 465–474.

Johnson DS, Trick MA, eds. (1996) *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge* (American Mathematical Society, Providence, RI)

Karp RM (1972) Reducibility among combinatorial problems. Miller RE, Thatcher JW, eds. *Complexity of Computer Computations* (Plenum Press, New York), 85–103.

Li CM, Quan Z (2010) An efficient branch-and-bound algorithm based on MaxSAT for the maximum clique problem. *Proc. 24th AAAI Conf. Artificial Intelligence*, vol. 10 (Association for the Advancement of Artificial Intelligence, Atlanta), 128–133.

Lovász L (1979) On the Shannon capacity of a graph. *IEEE Trans. Inform. Theory* 25(1):1–7.

Ma T, Latecki LJ (2012) Maximum weight cliques with mutex constraints for video object segmentation. *2012 IEEE Conf. Comput. Vision Pattern Recognition* (IEEE Computer Society, Washington, DC), 670–677.

Macambira EM, de Souza CC (2000) The edge-weighted clique problem: Valid inequalities, facets and polyhedral computations. *Eur. J. Oper. Res.* 123(2):346–371.

Martí R, Gallego M, Duarte A (2010) A branch and bound algorithm for the maximum diversity problem. *Eur. J. Oper. Res.* 200(1):36–44.

Mehrotra A, Trick MA (1998) Cliques and clustering: A combinatorial approach. *Oper. Res. Lett.* 22(1):1–12.

Mycielski J (1955) Sur le coloriage des graphs. *Colloquium Math.* 3(2): 161–162.

Östergård PRJ (2002) A fast algorithm for the maximum clique problem. *Discrete Appl. Math.* 120(1–3):197–207.

Padberg M (1989) The Boolean quadric polytope: some characteristics, facets and relatives. *Math. Programming* 45(1–3):139–172.

Palubeckis G (2007) Iterated tabu search for the maximum diversity problem. *Appl. Math. Comput.* 189(1):371–383.

Park K, Lee K, Park S (1996) An extended formulation approach to the edge-weighted maximal clique problem. *Eur. J. Oper. Res.* 95(3): 671–682.

Pavan M, Pelillo M (2003) Generalizing the Motzkin-Straus theorem to edge-weighted graphs, with applications to image segmentation. Rangarajan A, Figueiredo M, Zerubia J, eds. *Energy Minimization Methods in Computer Vision and Pattern Recognition—EMMCVPR 2003*, Lecture Notes in Computer Science, vol. 2683 (Springer, Berlin, Heidelberg), 485–500.

Pullan W (2006) Phased local search for the maximum clique problem. *J. Combin. Optim.* 12(3):303–323.

Pullan W (2008) Approximating the maximum vertex/edge weighted clique using local search. *J. Heuristics* 14(2):117–134.

San Segundo P, Nikolaev A, Batsyn M (2015) Infra-chromatic bound for exact maximum clique search. *Comput. Oper. Res.* 64:293–303.

San Segundo P, Rodrıguez-Losada D, Jiménez A (2011) An exact bit-parallel algorithm for the maximum clique problem. *Comput. Oper. Res.* 38(2):571–581.

Shimizu S, Yamaguchi K, Masuda S (2018) A maximum edge-weight clique extraction algorithm based on branch-and-bound. Preprint, submitted October 24, https://arxiv.org/abs/1810.10258.

Shimizu S, Yamaguchi K, Masuda S (2019) A branch-and bound based exact algorithm for the maximum edge-weight clique problem. Lee R, ed. *Computational Science/Intelligence & Applied Informatics*, Studies in Computational Intelligence, vol. 787 (Springer International Publishing AG, Cham, Switzerland), 27–47.

Silva GC, de Andrade MRQ, Ochi LS, Martins SL, Plastino A (2007) New heuristics for the maximum diversity problem. *J. Heuristics* 13(4):315–336.

Sørensen MM (2004) New facets and a branch-and-cut algorithm for the weighted clique problem. *Eur. J. Oper. Res.* 154(1):57–70.

Tomita E, Kameda T (2007) An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments. *J. Global Optim.* 37(1):95–111.

Tomita E, Seki T (2003) An efficient branch-and-bound algorithm for finding a maximum clique. Calude CS, Dinneen MJ, Vajnovszki V, eds. *Discrete Mathematics and Theoretical Computer Science*, Lecture Notes in Computer Science, vol. 2731 (Springer, Berlin, Heidelberg), 278–289.

Tomita E, Sutani Y, Higashi T, Takahashi S, Wakatsuki M (2010) A simple and faster branch-and-bound algorithm for finding a maximum clique. Rahman MS, Fujita S, eds. *WALCOM: Algorithms and Computation*, Theoretical Computer Science and General Issues, vol. 5942 (Springer, Berlin, Heidelberg), 191–203.

Wang Y, Hao JK, Glover F, Lü Z (2014) A tabu search based memetic algorithm for the maximum diversity problem. *Engrg. Appl. Artificial Intelligence* 27:103–114.

Wilf HS (1967) The eigenvalues of a graph and its chromatic number. *J. London Math. Soc.* 42(1):330–332.

Wu Q, Hao JK (2015) A review on algorithms for maximum clique problems. *Eur. J. Oper. Res.* 242(3):693–709.

Zuckerman D (2006) Linear degree extractors and the inapproximability of max clique and chromatic number. *Proc. 38th Annual ACM Sympos. Theory Comput.* (ACM Press, New York), 681–690.