The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Exact Clustering via Integer Programming and Maximum Satisfiability

Atsushi Miyauchi,[1] Tomohiro Sonobe,[2,3] Noriyoshi Sukegawa[4]
[1]RIKEN AIP, Tokyo, Japan
[2]National Institute of Informatics, Tokyo, Japan
[3]JST, ERATO, Kawarabayashi Large Graph Project, Tokyo, Japan
[4]Chuo University, Tokyo, Japan
atsushi.miyauchi.hv@riken.jp, tomohiro_sonobe@nii.ac.jp, sukegawa.15k@g.chuo-u.ac.jp

Abstract

We consider the following general graph clustering problem: given a complete undirected graph $G = (V, E, c)$ with an edge weight function $c : E \to \mathbb{Q}$, we are asked to find a partition $\mathcal{C}$ of V that maximizes the sum of edge weights within the clusters in $\mathcal{C}$. Owing to its high generality, this problem has a wide variety of real-world applications, including correlation clustering, group technology, and community detection. In this study, we investigate the design of mathematical programming formulations and constraint satisfaction formulations for the problem. First, we present a novel integer linear programming (ILP) formulation that has far fewer constraints than the standard ILP formulation by Grötschel and Wakabayashi (1989). Second, we propose an ILP-based exact algorithm that solves an ILP problem obtained by modifying our above ILP formulation and then performs simple post-processing to produce an optimal solution to the original problem. Third, we present maximum satisfiability (MaxSAT) counterparts of both our ILP formulation and ILP-based exact algorithm. Computational experiments using well-known real-world datasets demonstrate that our ILP-based approaches and their MaxSAT counterparts are highly effective in terms of both memory efficiency and computation time.

1 Introduction

Clustering is a fundamental tool in data analysis. Roughly speaking, the task of clustering is to divide a given set of objects into subsets of homogeneous objects. To date, various problem settings and optimization algorithms have been extensively studied (Aggarwal and Reddy 2013; Jain, Murty, and Flynn 1999; Xu and Wunsch 2005).

Let us consider the following general graph clustering problem. An instance is a complete undirected graph $G = (V, E, c)$ with an edge weight function $c : E \to \mathbb{Q}$, where $\mathbb{Q}$ is the set of rational numbers. For simplicity, we denote $c_{ij} = c(\{i, j\})$ for each $\{i, j\} \in E$. The edge weight c_{ij} expresses the degree of preference that $i, j \in V$ are assigned to the same cluster; if c_{ij} is positive, we wish to assign $i, j \in V$ to the same cluster, whereas if c_{ij} is negative, we wish to assign $i, j \in V$ to different clusters. The goal is to find a partition $\mathcal{C} = \{V_1, V_2, \ldots, V_k\}$ (i.e., $\bigcup_{l=1}^{k} V_l = V$ and $V_i \cap V_j = \emptyset$ for $i \neq j$) of V that maximizes the sum of edge weights within the clusters in $\mathcal{C}$, i.e., $\sum_{\{i,j\} \in E} c_{ij} \delta(\mathcal{C}(i), \mathcal{C}(j))$, where $\mathcal{C}(i)$ is the (unique) cluster to which $i \in V$ belongs and δ is Kronecker's symbol, which is equal to 1 if two arguments are the same and 0 otherwise. Note here that in this problem, there is no restriction on the number of clusters in the output partition; thus, the algorithms are allowed to specify the optimal number of clusters endogenously. This problem is known as the *clique partitioning problem* (CPP), which was originally introduced by Grötschel and Wakabayashi (1989). As described in Section 2, the high generality of CPP leads to a wide variety of real-world applications, including correlation clustering, group technology, and community detection. The NP-hardness was demonstrated in Wakabayashi (1986).

In the field of artificial intelligence, *mathematical programming* and *constraint satisfaction* are known to be key frameworks to solve NP-hard optimization problems. In these frameworks, we first formulate a problem at hand as a mathematical programming problem or a constraint satisfaction problem such as an integer programming (IP) problem or a maximum satisfiability (MaxSAT) problem, and then solve it using powerful mathematical programming or constraint satisfaction solvers. For example, in the case of IP problems, we may use Gurobi Optimizer or IBM ILOG CPLEX. The problem we address, CPP, is not an exception; in fact, both of mathematical programming formulations and constraint satisfaction formulations have been actively developed.

Mathematical programming formulations. Grötschel and Wakabayashi (1989) introduced a 0-1 integer linear programming (ILP) formulation for CPP, which has been employed by many algorithms for CPP and its variants (e.g., (Agarwal and Kempe 2008; Bruckner et al. 2013; Jaehn and Pesch 2013; Miyauchi and Miyamoto 2013; Nowozin and Jegelka 2009; Oosten, Rutten, and Spieksma 2001; Van Gael and Zhu 2007)). The ILP formulation is simple and intuitive, but not sufficiently scalable for real-world applications in terms of both memory requirements and computation time. In particular, the issue of memory requirements is quite serious. Letting $n = |V|$, the ILP formulation has $3\binom{n}{3} = \Theta(n^3)$ constrains, which grows rapidly as n increases. For example, if $n =$ 1,000, the number of con-

straints reaches the order of half a billion; it is quite difficult to store such an ILP formulation on a standard computer.

To overcome this issue, much effort has been dedicated to constructing ILP formulations for CPP with fewer constraints. Dinh and Thai (2015) addressed a special case of CPP, which is called the *modularity maximization problem* (Fortunato 2010; Newman and Girvan 2004), and derived a set of redundant constraints in the ILP formulation by Grötschel and Wakabayashi (1989). By removing the constraints in advance, they introduced an ILP formulation with fewer constraints for the special case. Recently, Miyauchi and Sukegawa (2015b) generalized Dinh and Thai's result to CPP. If $m_{\geq 0}$ denotes the number of nonnegative-weighted edges in G, i.e., $m_{\geq 0} = |\{\{i,j\} \in E : c_{ij} \geq 0\}|$, the ILP formulation by Miyauchi and Sukegawa (2015b) has $O(nm_{\geq 0})$ constraints, which improves on the ILP formulation by Grötschel and Wakabayashi (1989) for the case in which $m_{\geq 0}$ is not large (e.g., $m_{\geq 0} = O(n)$).

However, for most real-world instances of CPP, the parameter $m_{\geq 0}$ is large owing to the large number of edges with weight zero; thus, the benefit of the above reformulation is quite limited for real-world applications. In fact, computational experiments in Miyauchi and Sukegawa (2015b) demonstrated that the decrease in the number of constraints—which is about 20% at most—is not significant. Moreover, their reformulation does not reduce computation time significantly; in fact, in some cases computation time increases.

Constraint satisfaction formulations. Very recently, Berg and Järvisalo (2017) developed MaxSAT formulations for an optimization problem called the *weighted correlation clustering problem* (WCC). It should be noted that WCC is equivalent to CPP from an exact optimization perspective (see Example 1 in Section 2). Thus, the MaxSAT formulations by Berg and Järvisalo (2017) for WCC can be seen as exact formulations for CPP. WCC and its variants have been actively studied in the field of artificial intelligence (Ahn et al. 2015; Awasthi, Balcan, and Voevodski 2014; Bansal, Blum, and Chawla 2004; Bonchi, Gionis, and Ukkonen 2013; Chierichetti, Dalvi, and Kumar 2014; Kim et al. 2014; Puleo and Milenkovic 2016; Van Gael and Zhu 2007).

Specifically, Berg and Järvisalo (2017) developed three MaxSAT formulations: MaxSAT-Transitive, MaxSAT-Unary, and MaxSAT-Binary. MaxSAT-Transitive is the MaxSAT counterpart of the ILP formulation by Grötschel and Wakabayashi (1989). MaxSAT-Unary and MaxSAT-Binary are MaxSAT formulations that employ the *unary encoding* and *binary encoding* techniques, respectively. In their experiments, the MaxSAT formulations were compared with the ILP formulation by Grötschel and Wakabayashi (1989). The results of their experiments showed that MaxSAT-Binary outperforms both the other MaxSAT formulations and the ILP formulation. Thus, MaxSAT-Binary is known to be state-of-the-art in terms of exact formulation for WCC and thus for CPP.

1.1 Our Contribution

In this study, we further investigate the design of mathematical programming formulations and constraint satisfaction formulations for CPP. Our contribution can be summarized as follows:

1. We present a novel ILP formulation for CPP in which the number of constraints is $O(nm_{>0})$, where $m_{>0}$ is the number of positive-weighted edges in G, i.e., $m_{>0} = |\{\{i,j\} \in E : c_{ij} > 0\}|$.
2. We also propose an ILP-based exact algorithm for CPP. The algorithm first solves an ILP problem obtained by modifying our above ILP formulation and then performs simple post-processing to obtain an optimal solution to CPP.
3. We present MaxSAT counterparts of both our ILP formulation and ILP-based exact algorithm.
4. We conduct thorough experiments to evaluate the effectiveness of our ILP-based approaches and their MaxSAT counterparts in terms of both memory efficiency and computation time.

We first describe our first result above in detail. To design our ILP formulation, we effectively use the above result by Miyauchi and Sukegawa (2015b). Recall that they proposed an ILP formulation for CPP that has $O(nm_{\geq 0})$ constraints. The serious problem with their formulation is that for most real-world instances of CPP, the parameter $m_{\geq 0}$ is large owing to the large number of edges with weight zero. The critical idea behind the design of our ILP formulation is to perturb the edge weight function of a given instance so that all edges with weight zero have some negative weight. By doing this, the resulting instance has small $m_{\geq 0}$; thus, the ILP formulation by Miyauchi and Sukegawa (2015b) for the resulting instance, which is our proposed ILP formulation for the original instance, has far fewer constraints. Our theoretical analysis demonstrates that if the negative values used for the perturbation are close to zero, the proposed formulation obtains an optimal solution to the original instance.

We describe our second result in detail. In our ILP formulation above, it is necessary to deal with some negative perturbation values very close to zero. Unfortunately, such values may cause numerical instability and therefore increase computation time in practice; hence, such perturbation should be avoided if possible. To this end, we introduce an ILP problem that is identical to our above ILP formulation for CPP except that it uses an unperturbed objective function. This modified ILP problem also has $O(nm_{>0})$ constraints but does not depend on the perturbation. However, the ILP problem itself is incomplete as an ILP formulation for CPP; in fact, an optimal solution to the ILP problem may be infeasible for CPP. Thus, to obtain an optimal solution to CPP, the algorithm also performs simple post-processing that runs in linear time.

We next describe our third result in detail. As mentioned above, Berg and Järvisalo (2017) introduced a MaxSAT formulation called MaxSAT-Transitive, which is the MaxSAT counterpart of the ILP formulation by Grötschel and Wakabayashi (1989). That is, MaxSAT-Transitive uses hard clauses to represent the constraints in the ILP formulation and soft clauses associated with appropriate weights to represent its objective function. Beginning with MaxSAT-Transitive, we

can reproduce our results for ILP in the context of MaxSAT; specifically, we can obtain MaxSAT counterparts of both our ILP formulation and ILP-based exact algorithm.

Finally, we describe our fourth result in detail. In a series of experimental assessments, we compare our ILP-based approaches and their MaxSAT counterparts with the previous formulations using well-known real-world datasets arising in the context of correlation clustering, group technology, and community detection. The results demonstrate that our approaches significantly outperform the state-of-the-art formulations in terms of both memory efficiency and computation time. In particular, our ILP-based approaches can solve a real-world instance with a few thousand vertices for which the ILP formulation by Grötschel and Wakabayashi (1989) has more than eight billion constraints.

2 Application Examples

CPP is a general clustering problem and therefore has a wide variety of applications. Here we provide some important application examples.

Example 1 (Correlation clustering)**.** Correlation clustering was introduced by Bansal, Blum, and Chawla (2004) for clustering with qualitative information. As an example, we consider document clustering, in which a set of documents is to be clustered into topics, with the hindering constraint that there is no exact prior definition of what a "topic" constitutes. Alternatively, it can be assumed that there exists qualitative similarity information indicating that a number of pairs of documents are similar or dissimilar. In such a situation, the goal of correlation clustering is to find a partition of the set of documents that agrees as much as possible with the similarity information.

A mathematical formulation of correlation clustering is as follows: Let $G' = (V', E_+, E_-)$ be an edge-labeled undirected graph in which each edge $e \in E_+$ is labeled as "+" (similar) and each edge $e \in E_-$ is labeled as "−" (dissimilar). Note that $E_+ \cap E_- = \emptyset$ holds. The maximization version, MAXAGREE, asks for a partition $\mathcal{C}$ of V' that maximizes *agreements* (the number of $+$ edges within clusters plus the number of $-$ edges across clusters). The minimization version, MINDISAGREE, asks for a partition $\mathcal{C}$ of V' that minimizes *disagreements* (the number of $-$ edges within clusters plus the number of $+$ edges across clusters). These problems are equivalent in terms of optimality and are both NP-hard (Bansal, Blum, and Chawla 2004). WCC, which was mentioned above, deals with edge-weighted generalizations of both MAXAGREE and MINDISAGREE.

MAXAGREE and MINDISAGREE can be reduced to CPP. An instance $G = (V, E, c)$ of CPP is constructed as follows: Let $V = V'$. For each $\{i, j\} \in E$, we set $c_{ij} = 1$ if $\{i, j\} \in E_+$, $c_{ij} = -1$ if $\{i, j\} \in E_-$, and $c_{ij} = 0$ otherwise. Clearly, an optimal solution to CPP corresponds to an optimal solution to both MAXAGREE and MINDISAGREE. It should be noted that WCC can also be reduced to CPP and CPP can be reduced to WCC.

Example 2 (Group technology)**.** Group technology plays a key role in the design of efficient manufacturing systems (Groover 2007). Suppose that the goal is to develop a manufacturing system for some new product, comprising p parts that are processed by q machines. In such a situation, the goal of group technology is to find a suitable partition of the set of parts and machines needed to define an efficient cellular manufacturing system.

As mentioned in Oosten, Rutten, and Spieksma (2001), group technology can be modeled as CPP. An instance $G = (V, E, c)$ of CPP is constructed as follows: Let V be a union of the set of p parts and the set of q machines. An edge $\{i, j\} \in E$ between a part i and a machine j has weight 1 if i is processed by j and -1 otherwise. Each edge between two parts or two machines has weight zero.

Example 3 (Community detection)**.** Community detection is a fundamental task in network analysis that aims to find a partition of the set of vertices into communities (Fortunato 2010). Intuitively speaking, a community is a subset of vertices densely connected internally but sparsely connected with the rest of the network. Recently, the issue of community detection in bipartite networks has garnered a significant amount interest (Fortunato 2010). Barber (2007) introduced a quality function called the *bipartite modularity*, which is appropriate for community detection in bipartite networks. Let $G' = (V', E')$ be an undirected bipartite graph for which it is known that V' is divided into V'_L and V'_R so that each edge has one endpoint in V'_L and the other in V'_R. The bipartite modularity, a quality function for a partition $\mathcal{C}$ of V', can be written as $Q_b(\mathcal{C}) = \frac{1}{|E'|} \sum_{i \in V'_L} \sum_{j \in V'_R} \left(A_{ij} - \frac{d_i d_j}{|E'|} \right) \delta(\mathcal{C}(i), \mathcal{C}(j))$, where A_{ij} is the (i, j) component of the adjacency matrix of G' and d_i is the degree of $i \in V'$. The bipartite modularity maximization problem is NP-hard (Miyauchi and Sukegawa 2015a).

The problem can be reduced to CPP. An instance $G = (V, E, c)$ of CPP is constructed as follows: Let $V = V'$. An edge $\{i, j\} \in E$ between $i \in V'_L$ and $j \in V'_R$ has weight $\frac{A_{ij}}{|E'|} - \frac{d_i d_j}{|E'|^2}$. Each edge between two vertices in V'_L or two vertices in V'_R has weight zero.

3 ILP Formulation

We first revisit the standard formulation by Grötschel and Wakabayashi (1989) and the reformulation by Miyauchi and Sukegawa (2015b). Let $V = \{1, 2, \ldots, n\}$ and $P = \{(i, j) : 1 \leq i < j \leq n\}$. For each $(i, j) \in P$, we introduce a decision variable x_{ij} equal to 1 if $i, j \in V$ are in the same cluster and 0 otherwise. Then the ILP formulation by Grötschel and Wakabayashi (1989) can be represented as follows:

$$
\begin{aligned}
\mathrm{P}(G): \ \max. \quad & \sum_{(i,j) \in P} c_{ij} x_{ij} \\
\text{s. t.} \quad & x_{ij} + x_{jk} - x_{ik} \leq 1 && \forall (i, j, k) \in T, \\
& x_{ij} - x_{jk} + x_{ik} \leq 1 && \forall (i, j, k) \in T, \\
& -x_{ij} + x_{jk} + x_{ik} \leq 1 && \forall (i, j, k) \in T, \\
& x_{ij} \in \{0, 1\} && \forall (i, j) \in P,
\end{aligned}
$$

where $T = \{(i, j, k) : 1 \leq i < j < k \leq n\}$. The triangle inequality constraints stipulate that for any $i, j, k \in V$, if

$i, j \in V$ are in the same cluster and $j, k \in V$ are also in the same cluster, then $i, k \in V$ must be in the same cluster. The ILP formulation has $\binom{n}{2} = \Theta(n^2)$ variables and $3\binom{n}{3} = \Theta(n^3)$ triangle inequality constraints.

Miyauchi and Sukegawa (2015b) derived a set of redundant triangle inequality constraints in P(G). By removing the constraints in advance, they introduced the following ILP formulation:

$$\begin{aligned} \mathrm{RP}(G):\ \max.\quad & \sum_{(i,j)\in P} c_{ij}x_{ij} \\ \text{s. t.}\quad & x_{ij} + x_{jk} - x_{ik} \leq 1 && \forall (i,j,k) \in T^1_{\geq 0}, \\ & x_{ij} - x_{jk} + x_{ik} \leq 1 && \forall (i,j,k) \in T^2_{\geq 0}, \\ & -x_{ij} + x_{jk} + x_{ik} \leq 1 && \forall (i,j,k) \in T^3_{\geq 0}, \\ & x_{ij} \in \{0,1\} && \forall (i,j) \in P, \end{aligned}$$

where $T^1_{\geq 0} = \{(i,j,k) \in T : c_{ij} \geq 0 \text{ or } c_{jk} \geq 0\}$, $T^2_{\geq 0} = \{(i,j,k) \in T : c_{ij} \geq 0 \text{ or } c_{ik} \geq 0\}$, and $T^3_{\geq 0} = \{(i,j,k) \in T : c_{jk} \geq 0 \text{ or } c_{ik} \geq 0\}$. They proved the following theorem:

Theorem 1 (Theorem 1 in Miyauchi and Sukegawa (2015b))**.** *Let $G = (V, E, c)$ be an arbitrary instance of CPP. P(G) and RP(G) share the same set of optimal solutions.*

Therefore, we can solve RP(G) instead of P(G) to obtain an optimal solution to CPP. Note that the number of triangle inequality constraints in RP(G) can be evaluated as $O(nm_{\geq 0})$, where $m_{\geq 0}$ is the number of nonnegative-weighted edges in G, i.e., $m_{\geq 0} = |\{(i,j) \in P : c_{ij} \geq 0\}|$.

3.1 Our Formulation

Here we present our ILP formulation. Without loss of generality, we assume that the edge weight function c is integer-valued. When c is rational-valued, we can immediately obtain an equivalent instance in which c is integer-valued by multiplying an appropriate value for each c_{ij}.

Let $E_0 = \{\{i,j\} \in E : c_{ij} = 0\}$. We define an edge weight function $\widetilde{c}$ so that for each $\{i,j\} \in E$,

$$\widetilde{c}_{ij} = \begin{cases} -\epsilon & \text{if } \{i,j\} \in E_0, \\ c_{ij} & \text{otherwise,} \end{cases}$$

where $\epsilon \in (0, 1/\binom{n}{2})$. Let us introduce a new instance $\widetilde{G} = (V, E, \widetilde{c})$. Then the number of triangle inequality constraints in RP($\widetilde{G}$) is $O(nm_{>0})$, where $m_{>0}$ is the number of positive-weighted edges in G, i.e., $m_{>0} = |\{(i,j) \in P : c_{ij} > 0\}|$. It is expected that as $\epsilon > 0$ is sufficiently small, an optimal solution to RP($\widetilde{G}$) is also optimal to RP(G) and thus to P(G). In fact, we have the following theorem:

Theorem 2. *Let $G = (V, E, c)$ be an arbitrary instance of CPP such that c is integer-valued. Any optimal solution to RP($\widetilde{G}$) is also optimal to P(G).*

Proof. Let $\widetilde{\boldsymbol{x}} = (\widetilde{x}_{ij})_{(i,j)\in P}$ be an arbitrary optimal solution to RP($\widetilde{G}$). From Theorem 1, $\widetilde{\boldsymbol{x}}$ is an optimal solution to P($\widetilde{G}$), which implies that $\widetilde{\boldsymbol{x}}$ satisfies all the triangle inequality constraints for T. Thus, $\widetilde{\boldsymbol{x}}$ is also feasible for P(G).

We now show the optimality of $\widetilde{\boldsymbol{x}}$ to P(G). Let $\boldsymbol{x} = (x_{ij})_{(i,j)\in P}$ be an optimal solution to P(G) and a its objective value in P(G). Since all the constraints in RP($\widetilde{G}$) are also contained in P(G), the solution $\boldsymbol{x}$ is feasible for RP($\widetilde{G}$). The objective value of $\boldsymbol{x}$ in RP($\widetilde{G}$), i.e., $\sum_{(i,j)\in P} \widetilde{c}_{ij}x_{ij}$, is strictly greater than $a - 1$ because the decrement, due to the change from P(G) to RP($\widetilde{G}$), of the objective value of $\boldsymbol{x}$ is at most $\epsilon \cdot |E_0| < \left(1/\binom{n}{2}\right) \cdot \binom{n}{2} = 1$. As for the objective value of $\widetilde{\boldsymbol{x}}$ in P(G), i.e., $\sum_{(i,j)\in P} c_{ij}\widetilde{x}_{ij}$, we have $a \geq \sum_{(i,j)\in P} c_{ij}\widetilde{x}_{ij} \geq \sum_{(i,j)\in P} \widetilde{c}_{ij}\widetilde{x}_{ij} \geq \sum_{(i,j)\in P} \widetilde{c}_{ij}x_{ij} > a - 1$, where the first inequality follows from the feasibility of $\widetilde{\boldsymbol{x}}$ in P(G), and the third inequality follows from the optimality and feasibility of $\widetilde{\boldsymbol{x}}$ and $\boldsymbol{x}$, respectively, in RP($\widetilde{G}$). Since the objective value of $\widetilde{\boldsymbol{x}}$ in P(G) is an integer, we have $\sum_{(i,j)\in P} c_{ij}\widetilde{x}_{ij} = a$. Therefore, $\widetilde{\boldsymbol{x}}$ is optimal to P(G). □

4 ILP-Based Exact Algorithm

We introduce the following ILP problem:

$$\begin{aligned} \mathrm{RP}^*(G):\ \max.\quad & \sum_{(i,j)\in P} c_{ij}x_{ij} \\ \text{s. t.}\quad & x_{ij} + x_{jk} - x_{ik} \leq 1 && \forall (i,j,k) \in T^1_{>0}, \\ & x_{ij} - x_{jk} + x_{ik} \leq 1 && \forall (i,j,k) \in T^2_{>0}, \\ & -x_{ij} + x_{jk} + x_{ik} \leq 1 && \forall (i,j,k) \in T^3_{>0}, \\ & x_{ij} \in \{0,1\} && \forall (i,j) \in P, \end{aligned}$$

where $T^1_{>0} = \{(i,j,k) \in T : c_{ij} > 0 \text{ or } c_{jk} > 0\}$, $T^2_{>0} = \{(i,j,k) \in T : c_{ij} > 0 \text{ or } c_{ik} > 0\}$, and $T^3_{>0} = \{(i,j,k) \in T : c_{jk} > 0 \text{ or } c_{ik} > 0\}$. Note here that the set of constraints is the same as in RP($\widetilde{G}$), whereas the objective function is the same as in P(G) and RP(G), i.e., the unperturbed one.

Unfortunately, RP*(G) may fail to obtain an optimal solution to P(G). In fact, there exist instances such that an optimal solution to RP*(G) is infeasible for P(G). For example, consider $G = (V, E, c)$ with $V = \{1,2,3,4\}$, $c_{12} = 1$, $c_{13} = c_{23} = -1$, and $c_{14} = c_{24} = c_{34} = 0$. A 0-1 vector $\overline{\boldsymbol{x}}^* = (\overline{x}^*_{ij})$ such that $\overline{x}^*_{12} = \overline{x}^*_{14} = \overline{x}^*_{24} = \overline{x}^*_{34} = 1$ and $\overline{x}^*_{13} = \overline{x}^*_{23} = 0$ is one of the optimal solutions to RP*(G); however, the triangle inequality constraint $-x_{13} + x_{34} + x_{14} \leq 1$ in P(G) is violated.

To obtain an optimal solution to P(G), we perform the following simple post-processing, which we refer to as pp, for an optimal solution $\overline{\boldsymbol{x}}^*$ to RP*(G): Let $P^*_{>0} = \{(i,j) \in P : \overline{x}^*_{ij} = 1,\ c_{ij} > 0\}$. Obtain a set of weakly connected components $\{V_1, V_2, \ldots, V_k\}$ of $(V, P^*_{>0})$ by the depth-first search. Output a 0-1 vector $\boldsymbol{x}^*$ that corresponds to the partition $\{V_1, V_2, \ldots, V_k\}$, i.e., $\boldsymbol{x}^*$ such that $x^*_{ij} = 1$ if and only if $i, j \in V_q$ for some $q \in \{1, 2, \ldots, k\}$. Note that pp runs in time linear in the size of G.

4.1 Correctness

Here we demonstrate that our algorithm (i.e., RP*(G)+pp) returns an optimal solution to P(G). To this end, it suffices

to show that the objective value of $\boldsymbol{x}^*$ remains the same as that of $\overline{\boldsymbol{x}}^*$ (in P(G) and RP*(G)) because $\boldsymbol{x}^*$ is feasible for P(G) and $\overline{\boldsymbol{x}}^*$ is optimal to a relaxation RP*(G) of P(G). For convenience, we define $P^*_{\text{in}} = \{(i,j) \in P : x^*_{ij} = 1\}$ and $P^*_{\text{out}} = P \setminus P^*_{\text{in}}$. We have the following lemmas:

Lemma 1. *It holds that* $\sum_{(i,j)\in P^*_{in}} c_{ij} x^*_{ij} = \sum_{(i,j)\in P^*_{in}} c_{ij} \overline{x}^*_{ij}$.

Proof. It suffices to show that for any $q \in \{1, 2, \ldots, k\}$, it holds that $\overline{x}^*_{ij} = 1$ for each $i, j \in V_q$ with $i < j$. Fix $q \in \{1, 2, \ldots, k\}$. Let i, j be a pair of distinct vertices of V_q. Since V_q is weakly connected by $P^*_{>0}$, there exists a path on $P^*_{>0}$ that connects i and j if we ignore the direction of edges. Denote this (undirected) path by $i = v_0, v_1, \ldots, v_t = j$. Since $c_{v_0v_1} > 0$ (and $c_{v_1v_2} > 0$), RP*(G) has the constraint $x_{v_0v_1} + x_{v_1v_2} - x_{v_0v_2} \leq 1$. Note here that in this notation, it is necessary that $v_0 < v_1 < v_2$ holds. If it is not the case, we should swap the order of the indices appropriately. Substituting $\overline{x}^*_{v_0v_1} = \overline{x}^*_{v_1v_2} = 1$ to this constraint, we have $\overline{x}^*_{v_0v_2} = 1$. Moreover, since $c_{v_2v_3} > 0$, RP*(G) also has the constraint $x_{v_0v_2} + x_{v_2v_3} - x_{v_0v_3} \leq 1$. Substituting $\overline{x}^*_{v_0v_2} = \overline{x}^*_{v_2v_3} = 1$ to this constraint, we have $\overline{x}^*_{v_0v_3} = 1$. Repeating this operation, we finally have $\overline{x}^*_{v_0v_t} = \overline{x}^*_{ij} = 1$. □

Lemma 2. *It holds that* $\sum_{(i,j)\in P^*_{out}} c_{ij} \overline{x}^*_{ij} \leq 0$.

Proof. For each $(i,j) \in P^*_{\text{out}}$, we have $(i,j) \notin P^*_{>0}$. If otherwise, then $x^*_{ij} = 1$ and thus $(i,j) \in P^*_{\text{in}}$. Therefore, for each $(i,j) \in P^*_{\text{out}}$, we have $\overline{x}^*_{ij} = 0$ or $c_{ij} \leq 0$, which proves the lemma. □

By Lemmas 1 and 2, we have $\sum_{(i,j)\in P} c_{ij} x^*_{ij} = \sum_{(i,j)\in P^*_{\text{in}}} c_{ij} x^*_{ij} \geq \sum_{(i,j)\in P^*_{\text{in}}} c_{ij} \overline{x}^*_{ij} + \sum_{(i,j)\in P^*_{\text{out}}} c_{ij} \overline{x}^*_{ij} = \sum_{(i,j)\in P} c_{ij} \overline{x}^*_{ij}$. Therefore, we have the following theorem:

Theorem 3. *Let $G = (V, E, c)$ be an arbitrary instance of CPP such that c is integer-valued. Any 0-1 vector returned by our algorithm (i.e., RP*(G)+pp) is optimal to P(G).*

5 MaxSAT Counterparts

Here we present MaxSAT counterparts of both our ILP formulation (i.e., RP($\widetilde{G}$)) and ILP-based exact algorithm (i.e., RP*(G)+pp). It should be noted that we here consider WCC rather than CPP. As described above, WCC is equivalent to CPP from an exact optimization perspective. Let $G = (V, E_+, E_-, c)$ be an instance of WCC. Note that $c : E_+ \cup E_- \to \mathbb{Q}_{>0}$ is an edge weight function, where $\mathbb{Q}_{>0}$ is the set of positive rational numbers. The (positive) edge weights represent the strength of similarity and dissimilarity for $\{i,j\} \in E_+$ and $\{i,j\} \in E_-$, respectively. For simplicity, we denote $c_{ij} = c(\{i,j\})$ for each $\{i,j\} \in E_+ \cup E_-$.

We revisit MaxSAT-Transitive introduced by Berg and Järvisalo (2017), which is the MaxSAT counterpart of P(G). Let $V = \{1, 2, \ldots, n\}$ and $P = \{(i,j) : 1 \leq i < j \leq n\}$. For each $(i,j) \in P$, we introduce a Boolean variable x_{ij} equal to `True` if $i, j \in V$ are in the same cluster and `False` otherwise. Then MaxSAT-Transitive, which we refer to as S-P(G) in the present study, can be represented as follows:

Hard clauses:

$(\neg x_{ij} \vee \neg x_{jk} \vee x_{ik}) \quad \forall (i,j,k) \in T,$
$(\neg x_{ij} \vee x_{jk} \vee \neg x_{ik}) \quad \forall (i,j,k) \in T,$
$(x_{ij} \vee \neg x_{jk} \vee \neg x_{ik}) \quad \forall (i,j,k) \in T,$

Soft clauses:

(x_{ij}) with weight $c_{ij} \quad \forall (i,j) \in P$ with $\{i,j\} \in E_+,$
$(\neg x_{ij})$ with weight $c_{ij} \quad \forall (i,j) \in P$ with $\{i,j\} \in E_-,$

where $T = \{(i,j,k) : 1 \leq i < j < k \leq n\}$. The set of hard clauses is a clausal representation of the set of triangle inequality constraints in P(G), and the set of soft clauses is a clausal representation of the objective function in P(G).

By beginning with S-P(G), we can reproduce our results for ILP in the context of MaxSAT; specifically, we can obtain S-RP(G), S-RP($\widetilde{G}$), and S-RP*(G)+S-pp, which are the MaxSAT counterparts of RP(G), RP($\widetilde{G}$), and RP*(G)+pp, respectively. The detailed description of them is omitted owing to space limitations. We have the following theorems:

Theorem 4 (MaxSAT counterpart of Theorem 2)**.** *Let $G = (V, E_+, E_-, c)$ be an arbitrary instance of WCC such that c is integer-valued. Any optimal solution to S-RP($\widetilde{G}$) is also optimal to S-P(G).*

Theorem 5 (MaxSAT counterpart of Theorem 3)**.** *Let $G = (V, E_+, E_-, c)$ be an arbitrary instance of WCC such that c is integer-valued. Any `True`-`False` assignment returned by our algorithm (i.e., S-RP*(G)+S-pp) is optimal to S-P(G).*

6 Experimental Evaluation

The purpose of our experiments is to evaluate the effectiveness of our ILP-based approaches and their MaxSAT counterparts in terms of both memory efficiency and computation time. To this end, we use well-known real-world datasets arising in the context of correlation clustering, group technology, and community detection.

6.1 ILP-Based Approaches

We first compare our ILP formulation (i.e., RP($\widetilde{G}$)) and ILP-based exact algorithm (i.e., RP*(G)+pp) with the previous ILP formulations (i.e., P(G) and RP(G)) and the state-of-the-art MaxSAT formulation (i.e., MaxSAT-Binary). As for RP($\widetilde{G}$), we set the parameter ϵ as follows: $\epsilon = \frac{1}{n^2}$ and $\frac{2}{n^2}$, both of which are less than $1/\binom{n}{2}$. All ILP formulations were solved using Gurobi Optimizer 7.0.2 with default parameters. As for MaxSAT-Binary, we employed the preprocessing and symmetry-breaking operations mentioned in Berg and Järvisalo (2017). The preprocessing was simulated using Coprocessor 3.0 and the symmetry-breaking procedure called RemoveSlack was applied. MaxSAT-Binary was solved using MaxHS 2.9, as recommended in Berg and Järvisalo (2017).

Table 1: Instances used in our experiments.

ID	Name	n	$m_{>0}/\binom{n}{2}$	$m_{\geq 0}/\binom{n}{2}$
P1	Protein1	669	4.22%	98.49%
P2	Protein2	587	6.36%	98.21%
P3	Protein3	567	6.39%	97.95%
P4	Protein4	654	2.64%	98.87%
G9	Ch-8x20b	28	24.07%	81.75%
G14	Mc-16x24	40	11.47%	61.81%
G16	KI-16x43	59	7.36%	67.15%
G17	Ca-18x24	42	10.22%	60.05%
G18	MT-20x20	40	14.23%	62.95%
G19	Ku-20x23	43	12.51%	61.57%
G21	Bo-20x35	55	10.03%	62.90%
G25	CH5-24x40	64	6.50%	58.88%
G26	CH6-24x40	64	6.50%	58.88%
G27	CH7-24x40	64	6.50%	58.88%
G28	Mc-27x27	54	15.30%	64.36%
G29	Ca-28x46	74	7.81%	60.13%
G30	Ku-30x41	71	5.15%	55.65%
G31	St-30x50-1	80	4.87%	57.41%
G32	St-30x50-2	80	5.28%	57.82%
G33	Ki-36x90	126	4.23%	66.41%
G34	MC-37x53	90	24.39%	75.43%
G35	Ch-40x100	140	4.32%	63.21%
C1	Wafa-CEO	41	11.10%	63.54%
C2	Divorces	59	11.63%	85.33%
C3	Hollywood movies	102	3.73%	56.79%
C4	Scotland interlocks	244	1.21%	59.00%
C5	Graph product	674	0.27%	50.43%
C6	Network science	2,549	0.08%	53.11%

The time limit was set to 4 hours and the memory limit was set to 96 GB. The experiments were conducted on a Linux machine with Intel Xeon Processor E5-2650 v2 2.60 GHz CPU and 96 GB RAM.

Correlation clustering. Correlation clustering instances were first tested. Specifically, we considered MAXAGREE in the edge-weighted setting (i.e., WCC). The upper section of Table 1 lists the instances on which experiments were conducted, which were generated from protein sequence datasets on `http://www.paccanarolab.org/scps` in the same manner as in Berg and Järvisalo (2017). The data consist of similarity values between amino-acid sequences that were computed using BLAST (Altschul et al. 1990).

The results are detailed in Table 2. The number of variables is always equal to $\binom{n}{2} = n(n-1)/2$. The number of constraints in RP$^*(G)$ is omitted because it is exactly the same as in RP$(\widetilde{G})$. OM in some columns indicates that the memory requirements of the formulation (and the solving phase) exceed the limit (i.e., 96 GB). For the formulations that could be stored on the machine but could not be solved within the time limit, the relative gaps (i.e., $(\text{UB} - \text{LB})/\text{LB}$, where UB and LB, respectively, are the upper and lower bounds on the optimal value) obtained by the time limit are presented within parentheses if they are finite; otherwise OT is given. For each instance, the best computation time (or the relative gap) among the formulations is written in bold.

It is seen that neither P(G) nor RP(G) could be stored on the machine for instances P1 and P4 owing to a shortage of memory capacity. Although RP(G) had fewer constraints than P(G), the decrement was quite small, with at most 0.1% of the constraints removed. By contrast, our formulations, RP$(\widetilde{G})$ and RP$^*(G)$, had far fewer constraints, with about 90% of the constraints removed. Correspondingly, the memory limit was not exceeded, and optimal solutions were obtained for instances P1 and P2. The results also show that our formulations outperformed MaxSAT-Binary. In fact, only RP$(\widetilde{G})$ and RP$^*(G)$+pp could obtain an optimal solution to instance P2 and nearly-optimal solutions for instances P3 and P4, although MaxSAT-Binary solved instance P1 faster.

Group technology. An assessment of group technology instances was then conducted; some of these are listed in the middle section of Table 1. The instances were generated from manufacturing cell formation datasets on `http://mauricio.resende.info/data` in the manner described in Example 2; a detailed description of the datasets can be found in Gonçalves and Resende (2004). Although there were originally 35 instances, which were indexed from G1 to G35, some instances are omitted owing to space limitations. Our preliminary experiments showed that all formulations could solve the omitted instances within 10.0 s.

The results are summarized in Table 2. Because the instances are smaller than those used in correlation clustering, P(G) and RP(G) could always be stored on the machine. Although RP(G) had fewer constraints than P(G), the computation time (or the relative gap) increased in 7 out of 17 instances. Again, our formulations, RP$(\widetilde{G})$ and RP$^*(G)$, produced far fewer constraints. In fact, even for relatively small instances, at least 50% of the constraints were removed, with the figure increasing to above 90% for some large instances. Furthermore, the computation time was reduced significantly. In particular, RP$^*(G)$+pp could obtain optimal solutions for all instances within the time limit. The results also show that MaxSAT-Binary performed no better than P(G) and RP(G). In fact, for any instance that could not be solved by P(G) and RP(G), MaxSAT-Binary also failed to obtain an optimal solution, and moreover, only MaxSAT-Binary exceeded the memory limit for some instances owing to its significant memory requirements in the search phase.

Community detection. Community detection instances were then tested, with particular consideration given to the bipartite modularity maximization problem. The last section of Table 1 lists the instances on which experiments were conducted, which were generated from network datasets on `http://vlado.fmf.uni-lj.si/pub/networks/data` in the manner described in Example 3.

The results are detailed in Table 2. It is seen that the trend of the results is similar to that produced in the correlation clustering and group technology assessments, i.e., RP$(\widetilde{G})$ and RP$^*(G)$+pp outperformed P(G), RP(G), and MaxSAT-Binary. Most strikingly, our approaches could obtain an optimal solution to instance C6 with 2,549 vertices, while P(G) and RP(G) required more than 8 billion and 6 billion constraints, respectively. Furthermore, MaxSAT-Binary left a very large relative gap (i.e., 6,350.3%).

Table 2: Results for the previous ILP formulations, our ILP-based approaches, and the state-of-the-art MaxSAT formulation.

ID	P(G)		RP(G)		RP($\widetilde{G}$)	$\epsilon = \frac{1}{n^2}$	$\epsilon = \frac{2}{n^2}$	RP*(G)+pp	MaxSAT-Binary		
	#constr.	time(s)	#constr.	time(s)	#constr.	time(s)	time(s)	time(s)	#var.	#clauses	time(s)
P1	149,038,482	OM	148,995,305	OM	12,106,852	4,010.3	3,505.6	3,796.2	167,184	684,568	**2,304.8**
P2	100,614,735	OM	100,571,045	OT	12,191,742	**2,774.3**	3,430.9	3,039.9	179,968	744,956	(48.1%)
P3	90,660,465	OT	90,611,160	OT	10,995,760	**(0.3%)**	**(0.3%)**	**(0.3%)**	173,928	719,876	(18.1%)
P4	139,222,212	OM	139,199,019	OM	7,161,805	**(0.2%)**	**(0.2%)**	**(0.2%)**	109,692	434,570	(119.2%)
G14	27,417	20.8	23,366	17.0	5,902	**4.2**	4.6	**4.2**	3,412	12,120	12.3
G16	97,527	233.3	84,297	551.1	13,560	14.2	13.9	**12.4**	6,212	22,440	64.3
G17	34,440	18.4	28,920	14.0	6,640	5.2	4.8	**3.2**	3,960	14,122	12.5
G18	29,640	804.4	25,661	803.9	7,839	433.6	343.6	**273.9**	3,680	12,962	OM
G19	37,023	66.6	31,606	52.4	8,646	18.9	18.6	**14.4**	4,196	15,128	219.5
G21	78,705	126.8	67,196	149.2	14,938	21.0	21.2	**19.1**	6,260	22,856	182.2
G25	124,992	474.5	102,860	245.2	15,750	16.6	**12.9**	14.7	9,536	31,718	110.4
G26	124,992	426.8	102,875	418.3	15,765	62.2	61.1	**25.5**	9,536	31,718	(1.4%)
G27	124,992	1,338.4	102,887	1,145.3	15,777	227.2	258.1	**168.9**	9,536	31,718	(160.6%)
G28	74,412	2,426.5	65,114	1,419.3	21,044	1,322.7	1,193.3	**570.1**	6,480	23,758	OM
G29	194,472	(6.4%)	161,752	(8.5%)	28,840	1,596.3	1,715.2	**1,296.5**	12,628	48,267	OM
G30	171,465	29.8	137,420	30.7	17,222	4.7	6.2	**4.3**	12,064	46,280	12.3
G31	246,480	78.4	199,476	85.0	23,508	10.7	12.0	**10.6**	14,620	55,683	14.9
G32	246,480	2,634.1	200,359	4,556.4	25,405	127.0	107.8	**47.0**	14,620	55,683	(137.2%)
G33	842,520	(151.3%)	716,019	(132.1%)	68,435	(6.0%)	(6.7%)	**11,963.4**	25,980	100,201	OM
G34	352,440	(13.2%)	326,247	(82.9%)	146,067	5,968.0	7344.6	**4,131.6**	18,909	73,342	(149.5%)
G35	1,342,740	16.3	1,121,684	13.4	112,904	**1.3**	**1.3**	**1.3**	42,240	168,918	17.8
C1	31,980	54.8	27,466	41.4	6,640	9.4	10.4	**7.5**	3,612	12,952	14.1
C2	97,527	4,927.8	93,475	4,749.9	20,116	75.6	74.5	**64.8**	4,308	15,232	138.9
C3	515,100	(11.6%)	413,487	(11.8%)	37,470	**8,711.4**	10,821.9	(0.4%)	23,190	90,651	OM
C4	7,174,332	(8.4%)	5,840,182	(8.4%)	172,070	**(1.3%)**	(1.9%)	(1.5%)	129,024	527,718	(141.9%)
C5	152,410,272	OM	114,839,168	OM	822,272	**15.9**	16.2	18.2	1,369,960	5,888,398	4,994.9
C6	8.3G	OM	6.3G	OM	13,130,379	**397.2**	407.3	402.4	21,417,336	94,669,067	(6,350.3%)

6.2 MaxSAT Counterparts

Next we compare our MaxSAT formulation (i.e., S-RP($\widetilde{G}$)) and MaxSAT-based exact algorithm (i.e., S-RP*(G)+S-pp) with the MaxSAT counterparts of P(G) and RP(G) (i.e., S-P(G) and S-RP(G)). As for S-RP($\widetilde{G}$), we set the parameter ϵ as in its ILP counterpart, i.e., $\epsilon = \frac{1}{n^2}$ and $\frac{2}{n^2}$. All MaxSAT formulations were solved using MaxHS 2.9.

The results are detailed in Table 3 with the same notations as in Table 2. As for S-RP($\widetilde{G}$), the left and right columns correspond to the results of $\epsilon = \frac{1}{n^2}$ and $\epsilon = \frac{2}{n^2}$, respectively. Note that the number of hard clauses in each MaxSAT formulation is equal to the number of constraints in its ILP counterpart (see Table 2). It is seen that our results are still effective for MaxSAT. In fact, S-RP($\widetilde{G}$) could solve 6 out of 11 instances that could be solved neither by S-P(G) nor by S-RP(G), and moreover, S-RP*(G) could solve instances P1, P2, and P4 much faster than S-P(G) and S-RP(G).

Table 3: Results for the MaxSAT counterparts.

ID	S-P(G)	S-RP(G)	S-RP($\widetilde{G}$)		S-RP*(G)+S-pp
	time(s)	time(s)	time(s)	time(s)	time(s)
P1	4,362.6	5,284.4	(0.5%)	(0.7%)	**1,058.4**
P2	2,690.6	2,779.7	2,049.8	2,366.6	**486.0**
P3	(11.0%)	**(3.1%)**	(253.1%)	(8.2%)	(3.9%)
P4	6,998.2	7,773.0	(31.8%)	(31.8%)	**3,200.5**
G14	14.9	**14.5**	16.6	18.0	21.8
G16	**19.6**	20.1	41.4	49.0	39.2
G17	**6.1**	6.2	11.4	13.1	10.5
G18	(140.0%)	(140.0%)	**3,782.8**	3,868.5	(140.0%)
G19	195.7	198.7	**45.4**	50.8	637.7
G21	124.3	126.6	**49.9**	60.2	132.4
G25	**31.9**	43.4	46.2	44.4	32.3
G26	2,017.7	1,986.3	**348.5**	484.6	4197.1
G27	(121.1%)	(121.1%)	**1,615.8**	2,338.5	(123.7%)
G28	OM	OM	**3,995.1**	6,949.2	OM
G29	OM	OM	**8,658.7**	11085.3	OM
G30	7.1	**6.1**	15.8	16.8	14.7
G31	**8.7**	9.3	26.9	28.2	8.9
G32	(118.0%)	(118.0%)	**772.0**	841.4	11,172.4
G33	OM	OM	OT	OT	OM
G34	(164.4%)	(164.4%)	OT	**13,421.1**	(154.9%)
G35	14.4	11.4	2.8	**2.4**	3.1
C1	13.8	13.4	14.5	18.0	**9.7**
C2	83.5	87.6	119.3	**82.9**	106.2
C3	(1,340.7%)	(1,340.7%)	OT	OT	**(802.1%)**
C4	**(65.8%)**	**(65.8%)**	OT	OT	(75.3%)
C5	4,162.6	2,794.7	525.7	555.7	**268.0**
C6	OM	OM	(18.3%)	(25.7%)	**(8.5%)**

7 Conclusion

In this study, we have investigated the design of mathematical programming formulations and constraint satisfaction formulations for CPP. More specifically, we have presented a novel ILP formulation, an ILP-based exact algorithm, and their MaxSAT counterparts. The experimental results demonstrate that our approaches significantly outperform the state-of-the-art formulations in terms of both memory efficiency and computation time.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable suggestions and helpful comments. The first author is supported by a Grant-in-Aid for Research Activity Start-up (No. 17H07357). This work was supported by JST ERATO Grant Number JPMJER1201, Japan.

References

Agarwal, G., and Kempe, D. 2008. Modularity-maximizing graph communities via mathematical programming. *European Physical Journal B* 66(3):409–418.

Aggarwal, C. C., and Reddy, C. K. 2013. *Data Clustering: Algorithms and Applications*. CRC Press.

Ahn, K. J.; Cormode, G.; Guha, S.; McGregor, A.; and Wirth, A. 2015. Correlation clustering in data streams. In *ICML '15: Proceedings of the 32nd International Conference on Machine Learning*, 2237–2246.

Altschul, S. F.; Gish, W.; Miller, W.; Myers, E. W.; and Lipman, D. J. 1990. Basic local alignment search tool. *Journal of Molecular Biology* 215(3):403–410.

Awasthi, P.; Balcan, M.-F.; and Voevodski, K. 2014. Local algorithms for interactive clustering. In *ICML '14: Proceedings of the 31st International Conference on Machine Learning*, 550–558.

Bansal, N.; Blum, A.; and Chawla, S. 2004. Correlation clustering. *Machine Learning* 56(1–3):89–113.

Barber, M. J. 2007. Modularity and community detection in bipartite networks. *Physical Review E* 76:066102.

Berg, J., and Järvisalo, M. 2017. Cost-optimal constrained correlation clustering via weighted partial maximum satisfiability. *Artificial Intelligence* 244:110–142.

Bonchi, F.; Gionis, A.; and Ukkonen, A. 2013. Overlapping correlation clustering. *Knowledge and Information Systems* 35(1):1–32.

Bruckner, S.; Hüffner, F.; Komusiewicz, C.; and Niedermeier, R. 2013. Evaluation of ILP-based approaches for partitioning into colorful components. In *SEA '13: Proceedings of the 12th International Symposium on Experimental Algorithms*, 176–187.

Chierichetti, F.; Dalvi, N.; and Kumar, R. 2014. Correlation clustering in MapReduce. In *KDD '14: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 641–650.

Dinh, T. N., and Thai, M. T. 2015. Toward optimal community detection: From trees to general weighted networks. *Internet Mathematics* 11(3):181–200.

Fortunato, S. 2010. Community detection in graphs. *Physics Reports* 486(3):75–174.

Gonçalves, J. F., and Resende, M. G. C. 2004. An evolutionary algorithm for manufacturing cell formation. *Computers & Industrial Engineering* 47(2–3):247–273.

Groover, M. P. 2007. *Automation, Production Systems, and Computer-Integrated Manufacturing*. Prentice Hall Press.

Grötschel, M., and Wakabayashi, Y. 1989. A cutting plane algorithm for a clustering problem. *Mathematical Programming* 45(1–3):59–96.

Jaehn, F., and Pesch, E. 2013. New bounds and constraint propagation techniques for the clique partitioning problem. *Discrete Applied Mathematics* 161(13–14):2025–2037.

Jain, A. K.; Murty, M. N.; and Flynn, P. J. 1999. Data clustering: A review. *ACM Computing Surveys* 31(3):264–323.

Kim, S.; Yoo, C. D.; Nowozin, S.; and Kohli, P. 2014. Image segmentation using higher-order correlation clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36(9):1761–1774.

Miyauchi, A., and Miyamoto, Y. 2013. Computing an upper bound of modularity. *European Physical Journal B* 86(7):302.

Miyauchi, A., and Sukegawa, N. 2015a. Maximizing Barber's bipartite modularity is also hard. *Optimization Letters* 9(5):897–913.

Miyauchi, A., and Sukegawa, N. 2015b. Redundant constraints in the standard formulation for the clique partitioning problem. *Optimization Letters* 9(1):199–207.

Newman, M. E. J., and Girvan, M. 2004. Finding and evaluating community structure in networks. *Physical Review E* 69:026113.

Nowozin, S., and Jegelka, S. 2009. Solution stability in linear programming relaxations: Graph partitioning and unsupervised learning. In *ICML '09: Proceedings of the 26th International Conference on Machine Learning*, 769–776.

Oosten, M.; Rutten, J. H. G. C.; and Spieksma, F. C. R. 2001. The clique partitioning problem: Facets and patching facets. *Networks* 38(4):209–226.

Puleo, G. J., and Milenkovic, O. 2016. Correlation clustering and biclustering with locally bounded errors. In *ICML '16: Proceedings of the 33rd International Conference on Machine Learning*.

Van Gael, J., and Zhu, X. 2007. Correlation clustering for crosslingual link detection. In *IJCAI '07: Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 1744–1749.

Wakabayashi, Y. 1986. *Aggregation of Binary Relations: Algorithmic and Polyhedral Investigations*. Ph.D. Dissertation, Universität Augsburg.

Xu, R., and Wunsch, D. 2005. Survey of clustering algorithms. *IEEE Transactions on Neural Networks* 16(3):645–678.