Exact Clustering via Integer Programming and Maximum Satisfiability

Atsushi Miyauchi,1 Tomohiro Sonobe,2,3 Noriyoshi Sukegawa4
1RIKEN AIP, Tokyo, Japan
2National Institute of Informatics, Tokyo, Japan
3JST, ERATO, Kawarabayashi Large Graph Project, Tokyo, Japan
4Chuo University, Tokyo, Japan

Abstract
We consider the following general graph clustering problem: given a complete undirected graph \( G = (V, E, c) \) with an edge weight function \( c : E \rightarrow \mathbb{Q} \), 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 2013).

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 \( c : E \rightarrow \mathbb{Q} \), 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 \( V_l \cap V_j = \emptyset \) for \( i \neq j \)) of \( V \) that maximizes the sum of edge weights within the clusters in \( \mathcal{C} \), i.e., \( \sum_{\{i,j\} \in E} c_{ij} \delta(C(i), C(j)) \), where \( C(i) \) is the (unique) cluster to which \( i \in V \) belongs and \( \delta \) 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., Aggarwal 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 \( \binom{n}{2} = \Theta(n^3) \) 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—that is, 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 \( O(nm_{\geq 0}) \), where \( m_{\geq 0} \) is the number of positive-weighted edges in \( G \), i.e., \( m_{\geq 0} = |\{(i,j) \in E : c_{ij} > 0\}| \).

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 \( O(nm_{\geq 0}) \) 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.

Next, we 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 formulation 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 $G = (V, E, c)$ 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 $C$ 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 $C$ 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 of 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 $C$ of $V'$, can be written as $Q_B(C) = \frac{1}{|E'|} \sum_{i \in V'_L} \sum_{j \in V'_R} \left( A_{ij} - \frac{d_i d_j}{|E'|} \right) \delta(C(i), C(j))$, 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 $A_{ij} - \frac{d_i d_j}{|E'|}$. 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 Miyachi 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:

$$P(G) : \text{max} \sum_{(i,j) \in P} c_{ij} x_{ij}$$

s. t. $x_{ij} + x_{jk} - x_{ik} \leq 1$ $\forall (i, j, k) \in T$,

$x_{ij} - x_{jk} + x_{ik} \leq 1$ $\forall (i, j, k) \in T$,

$-x_{ij} + x_{jk} + x_{ik} \leq 1$ $\forall (i, j, k) \in T$,

$x_{ij} \in \{0, 1\}$ $\forall (i,j) \in P$,

where $T = \{(i, j, k) : 1 \leq i < j < k \leq n\}$. The triangle inequality constraints stipulate that for any $i, j, k \in V$,
where \( T \) and \( \tilde{T} \) are the number of triangle inequality constraints in \( G \) and \( \tilde{G} \), respectively, in \( \text{RP}(\tilde{G}) \). Since the objective value of \( \tilde{G} \) is integer-valued by multiplying an appropriate value for each \( (i, j, k) \) component, we can immediately obtain \( \tilde{c}_{ij} \) so that for each \( (i, j) \in E \),
\[
\tilde{c}_{ij} = \begin{cases} 
-\epsilon & \text{if } (i, j) \in E_0, \\
\tilde{c}_{ij} & \text{otherwise},
\end{cases}
\]
where \( \epsilon \in (0, 1/\binom{n}{2}) \). Let us introduce a new instance \( \tilde{G} = (V, E, \tilde{c}) \). Then the number of triangle inequality constraints in \( \text{RP}(\tilde{G}) \) is \( O(nm_{\geq 0}) \), where \( m_{\geq 0} \) is the number of nonnegative-weighted edges in \( G \), i.e., \( m_{\geq 0} = |\{(i, j) \in E : c_{ij} \geq 0\}| \).

### 4.1 Correctness
Here we demonstrate that our algorithm (i.e., \( \text{RP}^*(G) + \text{pp} \)) returns an optimal solution to \( P(G) \). To this end, it suffices to show the optimality of \( \tilde{x} \) to \( P(G) \). Let \( x = (x_{ij})_{(i,j) \in E} \) be an optimal solution to \( P(G) \) and \( \epsilon \) its objective value in \( P(G) \). Since all the constraints in \( \text{RP}(\tilde{G}) \) are also contained in \( P(G) \), the solution \( x \) is feasible for \( \text{RP}(\tilde{G}) \). The objective value of \( x \) in \( \text{RP}(\tilde{G}) \), i.e., \( \sum_{(i,j) \in E} \tilde{c}_{ij}x_{ij} \), is strictly greater than \( \epsilon - 1 \) because the decrement, due to the change from \( P(G) \) to \( \text{RP}(\tilde{G}) \), of the objective value of \( x \) is at most \( c_{ij}E_0| < (1/\binom{n}{2}) \cdot \binom{n}{2} \). As for the objective value of \( \tilde{x} \) in \( P(G) \), i.e., \( \sum_{(i,j) \in E} c_{ij}\tilde{x}_{ij} \), we have \( \alpha \geq \sum_{(i,j) \in E} c_{ij}\tilde{x}_{ij} \geq \sum_{(i,j) \in E} c_{ij}x_{ij} - 1 > a - 1 \), where the first inequality follows from the feasibility of \( \tilde{x} \) in \( P(G) \), and the third inequality follows from the optimality and feasibility of \( x \) and \( \tilde{x} \), respectively, in \( \text{RP}(\tilde{G}) \). Since the objective value of \( \tilde{x} \) in \( P(G) \) is an integer, we have \( \sum_{(i,j) \in E} c_{ij}\tilde{x}_{ij} = \alpha \). Therefore, \( \tilde{x} \) is optimal to \( P(G) \).
to show that the objective value of \( x^* \) remains the same as that of \( \mathcal{F}^* \) (in \( P(G) \) and \( \mathcal{R}^* \) (G) because \( x^* \) is feasible for \( P(G) \) and \( \mathcal{F}^* \) is optimal to a relaxation \( \mathcal{R}^* \) (G) of \( P(G) \). For convenience, we define \( P^*_m = \{(i,j) \in P : x^* \}_{ij} = 1\) and \( P^*_o = P \setminus P^*_m \). We have the following lemmata:

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

**Proof.** It suffices to show that for any \( q \in \{1,2,\ldots,k\} \), it holds that \( \mathcal{F}^*_{ij} = 1 \) 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 \( P^*_m \), there exists a path on \( P^*_m \) that connects \( i \) and \( j \) if we ignore the direction of edges. Denote this (undirected) path by \( i = v_0, v_1, \ldots, v_q = j \). Since \( c_{v_0v_1} > 0 \) and \( c_{v_2v_3} > 0 \), \( \mathcal{R}^* (G) \) has the constraint \( x_{v_0v_1} + x_{v_2v_3} - x_{v_0v_3} \leq 1 \). 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 \( \mathcal{F}^*_{v_0v_1} = \mathcal{F}^*_{v_1v_2} = 1 \) to this constraint, we have \( \mathcal{F}^*_{v_0v_3} = 1 \). Moreover, since \( c_{v_2v_3} > 0 \), \( \mathcal{R}^* (G) \) also has the constraint \( x_{v_0v_2} + x_{v_2v_3} - x_{v_0v_3} \leq 1 \). Substituting \( \mathcal{F}^*_{v_0v_2} = \mathcal{F}^*_{v_2v_3} = 1 \) to this constraint, we have \( \mathcal{F}^*_{v_0v_3} = 1 \). Repeating this operation, we finally have \( \mathcal{F}^*_{v_0v_3} = \mathcal{F}^*_{ij} = 1 \).

**Lemma 2.** It holds that \( \sum_{(i,j) \in P^*_o} c_{ij} \mathcal{F}^*_{ij} \leq 0 \).

**Proof.** For each \((i,j) \in P^*_o \), we have \((i,j) \notin P^*_m \). Therefore, for each \((i,j) \in P^*_o \), we have \( \mathcal{F}^*_{ij} = 0 \) or \( \mathcal{F}^*_{ij} \leq 0 \), which proves the lemma.

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

**Theorem 3.** Let \( G = (V,E,c) \) be an arbitrary instance of \( \mathcal{C} \) such that \( c \) is integer-valued. Any \( 0\)-1 vector returned by our algorithm (i.e., \( \mathcal{R}^* (G) \) + \( \mathcal{P} \)) is optimal to \( P(G) \).

5 **MaxSAT Counterparts**

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

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 \( \text{True} \) if \( i,j \in V \) are in the same cluster and \( \text{False} \) otherwise. Then MaxSAT-Transitive, which we refer to as \( \mathcal{S}(P(G)) \) in the present study, can be represented as follows:

**Hard clauses:**
\[
\begin{align*}
&\neg x_{ij} \lor \neg x_{jk} \lor x_{ik} \quad \forall (i,j,k) \in T, \\
&\neg x_{ij} \lor x_{jk} \lor \neg x_{ik} \quad \forall (i,j,k) \in T, \\
&x_{ij} \lor \neg x_{jk} \lor \neg x_{ik} \quad \forall (i,j,k) \in T,
\end{align*}
\]

**Soft clauses:**
\[
(\forall (i,j) \in P \text{ with } \{i,j\} \in E_+ \text{ with weight } c_{ij}) \lor (\forall (i,j) \in P \text{ with } \{i,j\} \in E_\rightarrow \text{ with weight } c_{ij})
\]

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 \( \mathcal{S}(P(G)) \), we can reproduce our results for ILP in the context of MaxSAT; specifically, we can obtain \( \mathcal{S}(P(G)) \), \( \mathcal{S}(P(G)) + \mathcal{P} \), and \( \mathcal{S}(P(G)) + \mathcal{P} \) + \( \mathcal{P}(G) + \mathcal{P} \), 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 \( G = (V,E_+,E_\rightarrow,c) \) be an arbitrary instance of \( \mathcal{C} \) such that \( c \) is integer-valued. Any optimal solution to \( \mathcal{S}(P(G)) \) is also optimal to \( P(G) \).

**Theorem 5 (MaxSAT counterpart of Theorem 3).** Let \( G = (V,E_+,E_\rightarrow,c) \) be an arbitrary instance of \( \mathcal{C} \) such that \( c \) is integer-valued. Any \( \text{True-False} \) assignment returned by our algorithm (i.e., \( \mathcal{S}(P(G)) + \mathcal{S}(\mathcal{P}) \) is optimal to \( P(G) \).

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., \( \mathcal{R}(G) \)) and ILP-based exact algorithm (i.e., \( \mathcal{R}^* (G) + \mathcal{P} \)) with the previous ILP formulations (i.e., \( P(G) \) and \( \mathcal{R}(G) \)) and the state-of-the-art MaxSAT formulation (i.e., MaxSAT-Binary). As for \( \mathcal{R}(G) \), we set the parameter \( \epsilon \) as follows: \( \epsilon = \frac{1}{n^2} \) and \( \frac{2}{n^2} \), both of which are less than \( \frac{1}{(n)} \). All ILP formulations were solved using Gurobi Optimizer 7.0.2 with default parameters. As for MaxSAT-Binary, we employed the pre-processing and symmetry-breaking operations mentioned in Berg and Järvisalo (2017). The preprocessing was simulated using Coprocessor 3.0 and the symmetry-breaking procedure called REMOVE SLACK was applied. MaxSAT-Binary was solved using MaxHS 2.9, as recommended in Berg and Järvisalo (2017).
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 but could not be solved within the time limit, the relative gaps (i.e., (UB − LB)/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.

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 MAXAGREED 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 Jánváls (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(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., (UB − LB)/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\(^∗\)(G) and RP\(^∗\)(G) + pp, 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(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(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(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>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>( n )</th>
<th>( m_{\geq 1}(n) )</th>
<th>( m_{\geq 2}(n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Protein1</td>
<td>669</td>
<td>4.22%</td>
<td>98.49%</td>
</tr>
<tr>
<td>P2</td>
<td>Protein2</td>
<td>587</td>
<td>6.36%</td>
<td>98.21%</td>
</tr>
<tr>
<td>P3</td>
<td>Protein3</td>
<td>567</td>
<td>6.39%</td>
<td>97.95%</td>
</tr>
<tr>
<td>P4</td>
<td>Protein4</td>
<td>654</td>
<td>2.64%</td>
<td>98.87%</td>
</tr>
<tr>
<td>G9</td>
<td>Ch-8x20b</td>
<td>28</td>
<td>24.07%</td>
<td>81.75%</td>
</tr>
<tr>
<td>G14</td>
<td>Mc-16x24</td>
<td>40</td>
<td>11.47%</td>
<td>61.81%</td>
</tr>
<tr>
<td>G16</td>
<td>KI-16x43</td>
<td>59</td>
<td>7.36%</td>
<td>67.15%</td>
</tr>
<tr>
<td>G17</td>
<td>Ca-18x24</td>
<td>42</td>
<td>10.22%</td>
<td>60.05%</td>
</tr>
<tr>
<td>G18</td>
<td>MT-20x20</td>
<td>40</td>
<td>14.23%</td>
<td>62.95%</td>
</tr>
<tr>
<td>G19</td>
<td>Ku-20x23</td>
<td>43</td>
<td>12.51%</td>
<td>61.57%</td>
</tr>
<tr>
<td>G21</td>
<td>Bo-20x35</td>
<td>55</td>
<td>10.03%</td>
<td>62.90%</td>
</tr>
<tr>
<td>G25</td>
<td>CH5-24x40</td>
<td>64</td>
<td>6.50%</td>
<td>58.88%</td>
</tr>
<tr>
<td>G26</td>
<td>CH6-24x40</td>
<td>64</td>
<td>6.50%</td>
<td>58.88%</td>
</tr>
<tr>
<td>G27</td>
<td>CH7-24x40</td>
<td>64</td>
<td>6.50%</td>
<td>58.88%</td>
</tr>
<tr>
<td>G28</td>
<td>M-27x27</td>
<td>54</td>
<td>15.30%</td>
<td>64.36%</td>
</tr>
<tr>
<td>G29</td>
<td>Ca-28x46</td>
<td>74</td>
<td>7.81%</td>
<td>60.13%</td>
</tr>
<tr>
<td>G30</td>
<td>Ku-30x41</td>
<td>71</td>
<td>5.15%</td>
<td>55.65%</td>
</tr>
<tr>
<td>G31</td>
<td>St-30x50-1</td>
<td>80</td>
<td>4.87%</td>
<td>57.41%</td>
</tr>
<tr>
<td>G32</td>
<td>St-30x50-2</td>
<td>80</td>
<td>5.28%</td>
<td>57.82%</td>
</tr>
<tr>
<td>G33</td>
<td>Ki-36x90</td>
<td>96</td>
<td>4.23%</td>
<td>66.41%</td>
</tr>
<tr>
<td>G34</td>
<td>MC-37x53</td>
<td>90</td>
<td>24.39%</td>
<td>75.43%</td>
</tr>
<tr>
<td>G35</td>
<td>Ch-40x100</td>
<td>140</td>
<td>4.32%</td>
<td>63.21%</td>
</tr>
</tbody>
</table>

**Table 1: Instances used in our experiments.**
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.

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

<table>
<thead>
<tr>
<th>ID</th>
<th>P((\mathcal{G}))</th>
<th>(\text{const.})</th>
<th>time(s)</th>
<th>(\text{const.})</th>
<th>time(s)</th>
<th>(\text{const.})</th>
<th>time(s)</th>
<th>(\text{const.})</th>
<th>time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>149,038,482</td>
<td>OM</td>
<td>149,995,305</td>
<td>OM</td>
<td>12,106,824</td>
<td>4,010.3</td>
<td>3,505.6</td>
<td>3,796.2</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>100,614,735</td>
<td>OM</td>
<td>100,571,045</td>
<td>OT</td>
<td>12,191,742</td>
<td>2,774.3</td>
<td>3,430.9</td>
<td>3,039.9</td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>90,660,465</td>
<td>OT</td>
<td>90,611,160</td>
<td>OT</td>
<td>10,995,760</td>
<td>(0.3%)</td>
<td>(0.3%)</td>
<td>(0.3%)</td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>139,222,212</td>
<td>OM</td>
<td>139,199,019</td>
<td>OM</td>
<td>7,161,805</td>
<td>(0.2%)</td>
<td>(0.2%)</td>
<td>(0.2%)</td>
<td></td>
</tr>
</tbody>
</table>

### Table 3: Results for the MaxSAT counterparts.

<table>
<thead>
<tr>
<th>ID</th>
<th>S-P((\mathcal{G}))</th>
<th>S-RP((\mathcal{G}))</th>
<th>S-RP*((\mathcal{G}))</th>
<th>S-RP*((\mathcal{G})) + S-pp</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>4,362.6</td>
<td>5,284.4</td>
<td>(0.5%)</td>
<td>(0.7%)</td>
</tr>
<tr>
<td>P2</td>
<td>2,690.6</td>
<td>2,779.7</td>
<td>2,049.8</td>
<td>3,366.6</td>
</tr>
<tr>
<td>P3</td>
<td>(110%)</td>
<td>(31%)</td>
<td>(253.1%)</td>
<td>(8.2%)</td>
</tr>
<tr>
<td>P4</td>
<td>4,998.2</td>
<td>7,773.0</td>
<td>(31.8%)</td>
<td>(31.8%)</td>
</tr>
</tbody>
</table>

### 6.2 MaxSAT Counterparts

Next we compare our MaxSAT formulation (i.e., S-RP(\(\mathcal{G}\))) and MaxSAT-based exact algorithm (i.e., S-RP*(\(\mathcal{G}\)) + S-pp) with the MaxSAT counterparts of P(\(\mathcal{G}\)) and RP(\(\mathcal{G}\)) (i.e., S-P(\(\mathcal{G}\)) and S-RP(\(\mathcal{G}\))). As for S-RP(\(\mathcal{G}\)), we set the parameter \(\epsilon\) as in its ILP counterpart, i.e., \(\epsilon = \frac{1}{\sqrt{2}}\) and \(\epsilon = \frac{1}{\sqrt{2}}\). 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(\(\mathcal{G}\)), the left and right columns correspond to the results of \(\epsilon = \frac{1}{\sqrt{2}}\) and \(\epsilon = \frac{1}{\sqrt{2}}\), 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(\(\mathcal{G}\)) could solve 6 out of 11 instances that could be solved neither by S-P(\(\mathcal{G}\)) nor by S-RP(\(\mathcal{G}\)), and moreover, S-RP*(\(\mathcal{G}\)) could solve instances P1, P2, and P4 much faster than S-P(\(\mathcal{G}\)) and S-RP(\(\mathcal{G}\)).

### 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


