Cost-Optimal Constrained Correlation Clustering via Weighted Partial Maximum Satisfiability[☆]

Jeremias Berg, Matti Järvisalo[*]

HIIT, Department of Computer Science, University of Helsinki, Finland.
Emails: jeremias.berg@cs.helsinki.fi,matti.jarvisalo@cs.helsinki.fi

Abstract

Integration of the fields of constraint solving and data mining and machine learning has recently been identified within the AI community as an important research direction with high potential. This work contributes to this direction by providing a first study on the applicability of state-of-the-art Boolean optimization procedures to cost-optimal correlation clustering under constraints in a general similarity-based setting. We develop exact formulations of the correlation clustering task as Maximum Satisfiability (MaxSAT), the optimization version of the Boolean satisfiability (SAT) problem. For obtaining cost-optimal clusterings, we apply a state-of-the-art MaxSAT solver for solving the resulting MaxSAT instances optimally, resulting in cost-optimal clusterings. We experimentally evaluate the MaxSAT-based approaches to cost-optimal correlation clustering, both on the scalability of our method and the quality of the clusterings obtained. Furthermore, we show how the approach extends to *constrained* correlation clustering, where additional user knowledge is imposed as constraints on the optimal clusterings of interest. We show experimentally that added user knowledge allows clustering larger datasets, and at the same time tends to decrease the running time of our approach. We also investigate the effects of MaxSAT-level preprocessing, symmetry breaking, and the choice of the MaxSAT solver on the efficiency of the approach.

Keywords: Boolean optimization, Boolean satisfiability, Maximum satisfiability, correlation clustering, cost-optimal clustering, constrained clustering

1. Introduction

Integration of the fields of constraint solving and data mining and machine learning has recently been identified within the AI community as an important research direction with high potential. This work contributes to this direction by studying the applicability of Boolean optimization to cost-optimal correlation clustering under constraints.

[☆]This work is supported by Academy of Finland (grants #251170 COIN Centre of Excellence in Computational Inference Research, 276412, and 284591), Research Funds of the University of Helsinki, and Finnish Funding Agency for Technology and Innovation (project D2I: From Data to Intelligence). The authors thank Jessica Davies for providing the MaxHS solver version used in the experiments. A preliminary version of this work appeared as [1] and was presented at the 2013 ICDM workshops. This article thoroughly revises and extends the earlier workshop paper considerable, for examples by addressing the problem in a more general weighted setting, by introducing a third improved MaxSAT encoding, by extended experiments including comparisons with quadratic integer programming and several approximative algorithms, application of SAT-based preprocessing, symmetry breaking, and a MaxSAT solver comparison, as well as inclusion of full formal proofs and extended background and discussions.

[*]Corresponding author. Phone: +358 50 3199 248, Fax: +358 9 1915 1120.

A common problem setting in data analysis is a set of data points together with some information regarding their pairwise similarities from which some interesting underlying structure needs to be discovered. One way of approaching the problem is to attempt to divide the data into subgroups in a meaningful way, for example, so that data points in the same group are more similar to each other than to data points in other groups [2]. Discovering an optimal way of making such a division is in most settings computationally challenging and an active area of research [3]. A general term for problems of this kind is *clustering*: the groups the data is partitioned into are called *clusters*, and a partitioning of the dataset is called a *clustering* of the data.

In this work, we study the *correlation clustering* paradigm [4] in a general similarity-based setting. Correlation clustering is a well-studied [5, 6, 7, 8, 9] NP-hard problem. Given a labeled undirected graph with each edge labeled with either a positive or a negative label, the objective of correlation clustering is to cluster the nodes of the graph in a way which minimizes the number of positive edges between different clusters and negative edges within clusters. Taking a more general view to correlation clustering, we study the problem setting of *weighted* correlation clustering, in which each edge is associated with a weight (instead of merely a negative or positive label), indicating our confidence in that label. In the more general weighted case, the objective of correlation clustering is to minimize the sum of the weights of the positive edges between different clusters and the negative edges within clusters.

The correlation clustering paradigm is geared towards classifying data based on qualitative similarity information—as opposed to quantitative information—of pairs of data points. In contrast to other typical clustering paradigms, correlation clustering does not require the number of clusters as input. This makes it especially well-suited for settings in which the true number of clusters is unknown—which is often the case when dealing with real-world data. As a concrete example, consider the problem of clustering documents by topic without any prior knowledge on what those topics might be, based only on similarity information (edges) between pairs of different documents [4, 10]. Indeed, correlation clustering has various applications in biosciences [11], social network analysis and information retrieval [12, 13, 14]. Furthermore, the related problem of *consensus clustering* [15], with recent applications in bioinformatics and in particular microarray data analysis [16, 17, 18, 19], can also be naturally cast as correlation clustering.

Due to NP-hardness of correlation clustering [4], most algorithmic work on the problem has been heuristic, focusing on local search and approximative algorithms. While strong approximation algorithms have been proposed [4, 5, 6, 9]—providing up to constant-factor approximations in restricted settings—these algorithms are unable to provide actual cost-optimal solutions in general. In this work, we take a different approach: we study the applicability of state-of-the-art Boolean optimization techniques to *cost-optimally* solving real-world instances of the correlation clustering problem. A baseline motivation for this work are the recent advances in applying constraint programming for developing generic approaches to common data analysis problems [20, 21, 22, 23, 24, 25]. In a constraint programming based approach, the data analysis problem is stated in a declarative fashion within some constraint language, and then a generic solver for that language is used for solving the resulting instance.

Harnessing constraint solving for data analysis tasks has two key motivations. Firstly, declarative optimization systems allow for finding provably cost-optimal solutions. While heuristic approaches allow for scaling to very large datasets, quickly obtaining some hopefully meaningful clustering, the provably cost-optimal solutions obtained by the declarative approach can result in notably better clusterings which provide better insights into the data. This can be valuable especially when working on smaller scientific datasets which have taken years to collect [26]. Secondly, the declarative approach allows for easily integrating various types of additional constraints over the solution space at hand. This way, a user (domain data expert) may specify properties of solutions that are of interest to the user, without needing to extend

available specialized algorithms in a non-trivial way to cope with such additional constraints. A constraint-based framework for clustering problems is well-suited for problem instances where some form of domain specific knowledge might be required in order to obtain meaningful clusterings. The paradigm for clustering problems of this type is known as *constrained clustering* [27, 28, 29]. Recently, Boolean satisfiability (SAT) [30] based approaches to solving constrained clustering within other clustering problems have been proposed [23, 31], However, to the best of our knowledge the only work done on constrained correlation clustering is the linear programming based approach of [10]; this work is the first study on the applicability of Maximum Satisfiability (MaxSAT) [32], a well-known optimization version of SAT, to correlation clustering under constraints. The problem definition we study covers correlation clustering with additional constraints that, e.g., either force or forbid a pair of points from being assigned to the same cluster; known as must-link and cannot-link constraints [27].

1.1. Contributions

We present a novel and extensible MaxSAT-based approach to optimal correlation clustering. Using propositional logic as the declarative language, we formulate the correlation clustering task in an exact fashion as weighted partial MaxSAT [32] and apply a state-of-the-art MaxSAT solver to solve the resulting MaxSAT instance optimally. To our best knowledge this is the first practical approach to *exactly* solving correlation clustering for finding *cost-optimal* clusterings, i.e., optimal clusterings wrt the actual objective function of the problem, for real-world datasets with hundreds of elements. In contrast, most of the previous work on correlation clustering has mainly focused on approximation algorithms and greedy local-search techniques which cannot in general find optimal clusterings.

At the core of the approach, we present three different MaxSAT formulations of correlation clustering, and provide formal proofs for their correctness. We experimentally evaluate our approach on real-world datasets and compare the approach to both two alternative exact approaches, based on linear and quadratic integer programming [5, 33], and two approximation algorithms [5, 34]. The results show that our approach can provide cost-optimal solutions and scales better than competing exact integer and quadratic programming formulations. Furthermore, our approach performs especially well in terms of solution cost on sparse datasets (with many missing similarity entries), outperforming approximative methods even when the approximative methods are given full similarity information. Our approach easily extends to the task of *constrained correlation clustering*, which allows for the user to specify the clusterings of one's interest by imposing hard user-defined constraints over the search space of clusterings. We explain how different types of constraints can be handled within a MaxSAT-based approach to cost-optimally solving constrained correlation clustering instances. While approaches to constrained clustering have been proposed previously for different clustering paradigms [27, 28, 29, 35, 36, 23], the only previous work on constrained correlation clustering that we know of is [10]. However, their approach is approximative and the experiments are done on smaller datasets. In contrast, we show experimentally that added user knowledge allows clustering on larger datasets as it tends to notably decrease the running time of the approach. We also provide experimental results on MaxSAT-specific aspects of solving the correlation clustering instances, considering the effects of MaxSAT-level preprocessing, symmetry breaking, as well as the choice of the MaxSAT solver used on the efficiency of the approach.

1.2. Paper Organization

In Section 2 we provide a generic problem definition for correlation clustering that is used throughout this article. Our definition covers both correlation clustering and constrained correlation clustering. We also demonstrate how a symmetric similarity measure simplifies the objective function of the clustering problem

and show that a similarity measure can always be assumed to be symmetric. In Sections 3 and 4 we overview previously proposed linear and quadratic integer programs for solving correlation clustering exactly. In Section 5 we provide necessary background on Maximum Satisfiability. The MaxSAT encodings of correlation clustering are detailed in Sections 6, 7 and 8, respectively. Extensive experimental results are provided in Section 9. Finally, we present a short survey on related work in Section 10 and give some concluding remarks in Section 11. Formal proofs of the theorems presented in the paper are given in Appendix A.

2. Problem Setting

In this section, we present the general similarity-based problem setting under which we study correlation clustering in both unconstrained and constrained settings.

2.1. Problem Definition

Let $\overline{\mathbb{R}} = \mathbb{R} \cup \{\infty, -\infty\}$, $V = \{v_1, \ldots, v_N\}$ a set of N data points that we wish to cluster, and $W \in \overline{\mathbb{R}}^{N \times N}$ a *similarity matrix*. We denote the element on row i column j in W by $W(i, j)$. This input can be viewed as a weighted graph, as demonstrated by the following example.

Example 1. *Let $V = \{v_1, v_2, v_3, v_4\}$ be a set of data points and consider the similarity matrix W given in Figure 1 on the left. We can view this input as a directed graph $G = (V, E)$ where $(v_i, v_j) \in E$ if $W(i, j) \neq 0$, and the weight of each edge (v_i, v_j) is equal to $W(i, j)$. Figure 1 (right) illustrates the graph corresponding to W. In case the similarity matrix is symmetric, i.e., $W(i, j) = W(j, i)$ for all i and j, the graph underlying W is essentially undirected.*

$$W = \begin{bmatrix} \infty & 0 & -4.5 & 40 \\ -0.3 & \infty & -\infty & 65 \\ 2.5 & 0 & \infty & 0 \\ 0 & 4.5 & 0 & \infty \end{bmatrix}$$

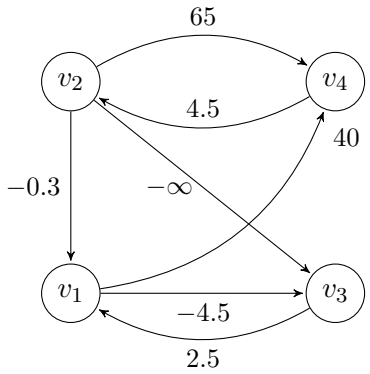

Figure 1: An example similarity matrix and its graph presentation.

The intuition behind the similarity matrix is that it expresses preferences on whether or not two points v_i and v_j should be assigned to the same cluster; a positive value indicates that v_i and v_j should be assigned to the same cluster, a negative value that they should not. We say that points v_i and v_j are *similar* if $W(i, j) \in \mathbb{R}$ and $W(i, j) > 0$. If $W(i, j) < 0$ and $W(i, j) \in \mathbb{R}$, we say that v_i and v_j are *dissimilar*. In the most general setting, neither the requirement of assigning pairs of points to the same (different) cluster(s) nor the notion of pairs of points being (dis)similar are required to be symmetric relations.

Any function $cl \colon V \to \mathbb{N}$ is a solution to the clustering problem, representing a clustering of the data points into clusters indexed with natural numbers. We say that two points v_i and v_j are *co-clustered* if $cl(v_i) = cl(v_j)$. Note that our formulation allows forcing two points to the same or different clusters. If

$W(i,j) = \infty$ for some i and j, then v_i and v_j have to be co-clustered. Analogously, if $W(i,j) = -\infty$, then v_i and v_j are not allowed to be co-clustered. If the infinite values are in conflict with each other, the problem instance is infeasible. These additional hard constraints are commonly referred to as *must-link* (ML) and *cannot-link* (CL) constraints [27]. We will use the following definition to incorporate the intended semantics of the infinite values onto the possible clusterings. Given a similarity matrix W, we say that a clustering cl *respects the infinite values of* W, if $cl(v_i) = cl(v_j)$ whenever $W(i,j) = \infty$ and $cl(v_i) \neq cl(v_j)$ whenever $W(i,j) = -\infty$.

Given a cost function G such that $G(W, cl) \in \mathbb{R}$ for every solution cl, we say that a clustering cl (of V) is *optimal* under W as measured by G, if cl respects the infinite values of W and $G(W, cl) \leq G(W, cl')$ holds for any clustering cl' (of V) that respects the infinite values of W. The definition is sufficient for all purposes as we can always turn a function G we wish to maximize into a minimization problem by considering the function $-G$. For a given similarity matrix W we use $\operatorname{argmin}_{cl}(G(W, cl))$ to denote the set of optimal clusterings under W as measured by G.

In this work we focus on the the cost function of *correlation clustering* with additional must-link and cannot-link constraints. In correlation clustering [4, 9] we are given a pairwise similarity measure over a set of data points. The task is then to cluster the nodes in a way that maximizes the number of similar points co-clustered and minimizes the number of dissimilar points co-clustered. More formally, given a symmetric similarity matrix W, the task is to find a clustering which minimizes the cost function

$$H(W, cl) = \sum_{\substack{cl(v_i)=cl(v_j) \\ i<j}} (\mathcal{I}[-\infty < W(i,j) < 0] \cdot |W(i,j)|) + \sum_{\substack{cl(v_i)\neq cl(v_j) \\ i<j}} (\mathcal{I}[\infty > W(i,j) > 0] \cdot W(i,j)) \tag{1}$$

where $\mathcal{I}[b]$ is an indicator function which takes the value 1 if the condition b is true, else $I[b] = 0$. Figure 2 gives a precise formulation of constrained correlation clustering used throughout this work.

Input: A set of N data points $V = \{v_1 \dots v_N\}$ and a symmetric similarity matrix $W \in \overline{\mathbb{R}}^{N \times N}$.

Output: A function $cl^* \colon V \to \mathbb{N}$ such that $cl^* \in \operatorname{argmin}_{cl \colon : V \to \mathbb{N}} (H(W, cl))$.

Figure 2: The constrained correlation clustering problem.

This definition covers all variants of correlation clustering that we are aware of. For example, the definition of [4] where the input consists of a complete graph with each edge labeled by a $+$ or $-$ is equivalent to restricting the input similarity matrix to only contain values from $\{-1, 1\}$ and specifically not to contain infinite values. Furthermore, the assumption of symmetric input can be made without loss of generality, as detailed in Section 2.2.

Example 2. *Let $V = \{v_1, v_2, v_3, v_4\}$ be a set of data points and consider the similarity matrix given in Figure 3 on the left. Figure 3 (right) illustrates one possible solution cl to the correlation clustering problem for this input data. In described solution, $cl(v_1) = cl(v_2) = cl(v_3) \neq cl(v_4)$. The cost of cl is*

$$\begin{aligned} H(W, cl) &= (\mathcal{I}[W(1,2) < 0] \cdot |W(1,2)| + \mathcal{I}[W(1,3) < 0] \cdot |W(1,3)| + \mathcal{I}[W(2,3) < 0] \cdot |W(2,3)|) + \\ &\quad (\mathcal{I}[W(1,4) > 0] \cdot W(1,4) + \mathcal{I}[W(2,4) > 0] \cdot W(2,4) + \mathcal{I}[W(3,4) > 0] \cdot W(3,4)) \\ &= |W(1,2)| + W(3,4) = 3.3. \end{aligned}$$

$$W = \begin{bmatrix} \infty & -0.3 & 2.5 & 0 \\ -0.3 & \infty & 4 & -5 \\ 2.5 & 4 & \infty & 3 \\ 0 & -5 & 3 & \infty \end{bmatrix}$$

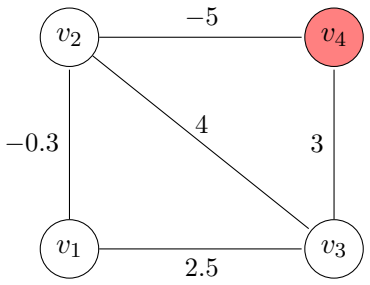

Figure 3: An example similarity matrix and the graphical representation of a solution to the correlation clustering problem

In contrast to many other clustering problems, deciding the number of clusters is in the most general case part of the correlation clustering problem. However, as every point is assigned to exactly one cluster, in practice it is enough to search over all functions $cl\colon V \to \{1, \ldots, N\}$.

2.2. On the Assumption of Symmetric Similarities

We will now show that the assumption of symmetric similarity matrices in our problem definition (Figure 2) can be done without loss of generality. Correlation clustering is often defined with 2 positive weights w_{ij}^+ and w_{ij}^- for each pair of data points v_i, v_j as the input [5]. The intuition behind these weights is that they give a separate measure for the costs of not assigning v_i and v_j to the same (w_{ij}^+) and to different (w_{ij}^-) cluster(s). A straightforward method of modeling this in terms of our clustering setting would be to use a cost function such as

$$H'(W, cl) = \sum_{cl(v_i)=cl(v_j)} (\mathcal{I}[-\infty < W(i,j) < 0] \cdot |W(i,j)|) + \sum_{cl(v_i)\neq cl(v_j)} (\mathcal{I}[\infty > W(i,j) > 0] \cdot W(i,j)), \tag{2}$$

and letting $W(i,j) = w_{ij}^+$ and $W(j,i) = -w_{ij}^-$ for all $i < j$. However, this turns out to be unnecessary.

Theorem 1. *Let $V = \{v_1 \ldots v_N\}$ be a set of data points and W an asymmetric similarity matrix over V. Assume that for all i and j, $W(i,j) = \infty$ implies $W(j,i) \neq -\infty$ (from which it also follows that $W(i,j) = -\infty$ implies $W(j,i) \neq \infty$). Then there is a symmetric similarity matrix W^S such that*

$$\operatorname*{argmin}_{cl}(H(W^S, cl)) = \operatorname*{argmin}_{cl}(H'(W, cl)).$$

We note that the assumption in the theorem is minor. The condition can be checked in polynomial time, and if it does not hold, there are no feasible solutions to the constrained problem. A detailed proof of Theorem 1 is provided in Appendix A. The simpler objective function H simplifies the exact declarative formulations considered in this work.

2.3. Constrained Clustering

In the clustering domain, the concept of a constraint is fairly abstract and the exact types of constraints that are feasible depend on the particular domain. A typical categorization of different types of constraints are *instance-level* constraints and *cluster-level constraints* [37]. Cluster-level constraints [36] deal with relationships between clusters. Examples of cluster-level constraints include constraints which enforce a predefined lower bounds for the similarity (distance) over clusters or a predefined upper bound on the dissimilarity of points within the clusters, as well as constraints requiring that the clustering contains at most/exactly/at

least some fixed number of clusters or that all clusters contain at least a certain number of data points. Instance-level constraints deal with relationships between pairs of points. Two very well known examples are the already discussed ML and CL constraints. ML and CL constraints are have been shown to be flexible in the sense that many different types of constraints can be expressed in terms of them [37].

2.4. Consensus clustering

As detailed in [15], another problem closely related to correlation clustering is *consensus clustering*. In consensus clustering we are given a set V of data points and K different *clusterings* of V. The task is then to find a single *consensus clustering* which agrees as well as possible with the input clusterings. Consensus clustering fits into our problem definition by the following construction. For each pair of points $v_i, v_j \in V$ let s_{ij} be the number of clusterings in which v_i and v_j are co clustered and $d_{ij} = K - s_{ij}$ be the number of clusterings in which they are not. Now construct a similarity matrix W by assigning $W(i,j) = s_{ij}$ and $W(j,i) = -d_{ij}$ for each pair of data points and apply Theorem 1 to obtain the equivalent (in terms of correlation clustering) symmetric similarity matrix. Then an optimal solution to the resulting correlation clustering problem corresponds to an optimal solution to the consensus clustering problem. Consensus clustering is indeed also NP-hard [38]. Recently the problem has received more attention due to applications in bioinformatics and in particular microarray data analysis [16, 17, 18, 19].

3. Correlation Clustering as Integer Linear Programming

An exact integer linear programming (ILP) formulation of correlation clustering has been proposed in [5, 10]. We will now restate this integer linear programming formulation in terms of our generic problem setting.

Given a set $V = \{v_1, \ldots, v_N\}$ of N data points and a symmetric similarity matrix W, the integer program involves binary variables $x_{ij} \in \{0, 1\}$, where $1 \leq i < j \leq N$. The intended interpretation of these variables is that $x_{ij} = 1$ iff v_i and v_j are co-clustered in any clustering. We note that the variables are only required whenever $i < j$. However, for notational convenience, we use x_{ij} and x_{ji} to denote the same variable. Using these variables, the set of optimal solutions to the following integer linear program represents the set of optimal clusterings of V under W [5].

$$\text{MINIMIZE} \sum_{\substack{-\infty < W(i,j) < 0 \\ i<j}} (x_{ij} \cdot |W(i,j)|) - \sum_{\substack{\infty > W(i,j) > 0 \\ i<j}} (x_{ij} \cdot W(i,j))$$

$$\begin{aligned} \text{where} \quad x_{ij} + x_{jk} &\leq 1 + x_{ik} && \text{for all distinct } i,j,k \\ x_{ij} &= 1 && \text{for all } W(i,j) = \infty \\ x_{ij} &= 0 && \text{for all } W(i,j) = -\infty \\ x_{ij} &\in \{0,1\} && \text{for all } i,j. \end{aligned} \tag{3}$$

The purpose of the *transitivity constraint* $x_{ij} + x_{jk} \leq 1 + x_{ik}$ is to ensure a *well-defined* clustering; for any $(v_i, v_j, v_k) \in V \times V \times V$, each of the points v_i, v_j, v_k must belong to exactly one cluster, and hence it follows that if points v_i, v_j are assigned to the same cluster and points v_j, v_k are assigned to the same cluster, by transitivity then points v_i, v_k should also be assigned to the same cluster. Stated as a linear constraint we require that if $x_{ij} + x_{jk} = 2$ then $x_{ik} = 1$, which is exactly what the transitivity constraint in the integer program demands. The purpose of the two other constraints is to ensure that the solution clustering respects

the infinite values of W. Whenever $W(i,j) = \infty$, v_i and v_j have to be co-clustered, which in terms of the integer program is equivalent to $x_{ij} = 1$. Analogously, $W(i,j) = -\infty$ is equivalent to $x_{ij} = 0$. This formulation consists of $\mathcal{O}(N^2)$ variables and $\mathcal{O}(N^3)$ constraints. In terms of practical considerations, this suggests poor scalability for larger datasets.

4. Correlation Clustering as Quadratic Integer Programming

A quadratic integer programming formulation of correlation clustering was proposed in [33]. In addition to the number of data points N, the quadratic integer programming (QIP) formulation requires one additional parameter K, an upper limit for the number of clusters that the solution clustering should contain. The formulation allows $K = N$ in which case the set of possible solutions to the quadratic program exactly matches the set of possible solutions to the integer linear programming formulation of correlation clustering and our general definition of correlation clustering (recall Figure 2). We next restate the quadratic program in terms of our generic problem setting and the parameter K.

Given a set $V = \{v_1, \ldots, v_N\}$ of N data points, an upper bound on the number of clusters K, and a symmetric similarity matrix W, the quadratic program involves binary variables $y_i^k \in \{0,1\}$, where $1 \leq i \leq N$ and $1 \leq k \leq K$. The intended interpretation of the variables is that $y_i^k = 1$ iff data point v_i is assigned to cluster k. Using these variables, the set of optimal solutions to the following quadratic integer program represents the set of optimal clusterings of V under W [33].

$$\text{MINIMIZE} \sum_{\substack{-\infty < W(i,j) < 0 \\ i<j}} \left(\sum_{k=1}^{K} \left(y_i^k y_j^k \right) \cdot |W(i,j)| \right) - \sum_{\substack{\infty > W(i,j) > 0 \\ i<j}} \left(\sum_{k=1}^{K} \left(y_i^k y_j^k \right) \cdot W(i,j) \right)$$

$$\begin{aligned} \text{where} \quad \sum_{k=1}^{K} y_i^k &= 1 \quad \text{for all } i \\ \sum_{k=1}^{K} \left(y_i^k y_j^k \right) &= 1 \quad \text{for all } W(i,j) = \infty \\ \sum_{k=1}^{K} \left(y_i^k y_j^k \right) &= 0 \quad \text{for all } W(i,j) = -\infty \\ y_i^k &\in \{0,1\} \quad \text{for all } i,k. \end{aligned} \tag{4}$$

For some intuition, note that the sum $\sum_{k=1}^{K} \left(y_i^k y_j^k \right)$ is equal to 1 only if points v_i and v_j are assigned to the same cluster in the solution clustering. The purpose of the $\sum_{k=1}^{K} y_i^k = 1$ for all i constraint is to ensure that the solution to the quadratic program corresponds to a well-defined clustering of the data. As all the variables used are binary, the constraint forces exactly one of the variables $y_i^1, \ldots, y_i^K$ to 1 for all i, which in turn ensures that the corresponding data point v_i is assigned to exactly one cluster, as required for a well-defined clustering. This non-convex QIP consists of $\mathcal{O}(NK)$ variables and $\mathcal{O}(N + I)$ constraints where I is the number of infinite values in the input similarity matrix. We note that the non-convexity of the quadratic program can follow both from the integrality constraints as well as the similarity values themselves, as demonstrated by the following example.

Example 3. *Consider the set $V = \{v_1, v_2, v_3\}$ of data points and the following similarity matrix over V:*

$$W = \begin{bmatrix} \infty & -1 & -10 \\ -1 & \infty & 1 \\ -10 & 1 & \infty \end{bmatrix}.$$

For this similarity matrix and $K = N = 3$, the QIP in matrix form is

$$\text{MINIMIZE} \quad \frac{1}{2}(\mathbf{y})^T \mathbb{W}(\mathbf{y})$$

$$\textit{subject to:} \quad A\mathbf{y} = b$$
$$\mathbf{y} \in \{0,1\}^9,$$

where

$$\mathbf{y} = \begin{bmatrix} y_1^1 \\ y_1^2 \\ y_1^3 \\ y_2^1 \\ y_2^2 \\ y_2^3 \\ y_3^1 \\ y_3^2 \\ y_3^3 \end{bmatrix} \textit{ and } \mathbb{W} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 10 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 10 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & -1 & 0 & 0 & 0 \end{bmatrix}$$

with A and b chosen to fit the constraints $\sum_{k=1}^{K} y_i^k = 1$ for all i. For this instance of correlation clustering, the matrix $\mathbb{W}$ is indefinite. To see this, observe that it has both negative and positive eigenvalues, for example 10 and $-5 - 3\sqrt{3}$. Hence the objective function of the quadratic program in itself is not convex.

5. Maximum Satisfiability

Before describing our MaxSAT formulations of correlation clustering, we review necessary basic concepts related to Maximum Satisfiability.

5.1. Syntax and Semantics

For a Boolean variable x, there are two literals, x and $\neg x$. A clause is a disjunction ($\vee$, logical OR) of literals and a truth assignment is a function from Boolean variables to $\{0, 1\}$. A clause C is satisfied by a truth assignment τ ($\tau(C) = 1$) if $\tau(x) = 1$ for a literal x in C, or $\tau(x) = 0$ for a literal $\neg x$ in C. A set F of clauses is satisfiable if there is an assignment τ satisfying all clauses in F ($\tau(F) = 1$), and unsatisfiable ($\tau(F) = 0$ for any assignment τ) otherwise.

An instance $F = (F_h, F_s, c)$ of the *weighted partial MaxSAT* problem consists of two sets of clauses, a set F_h of *hard* clauses and a set F_s of *soft* clauses, and a function $c : F_s \to \mathbb{R}^+$ that associates a non-negative cost with each of the soft clauses[1]. Any truth assignment τ that satisfies F_h is a *solution* to F. The *cost*

[1]Our definition for the function c is more general than the standard $c : F_s \to \mathbb{N}^+$, which restricts the costs of soft clauses to be integral.

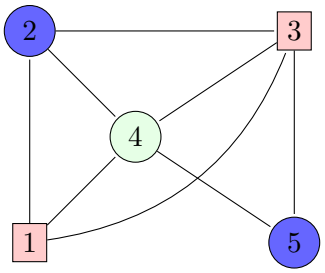

Figure 4: An example of representing graph coloring as MaxSAT.

$\text{COST}(F, \tau)$ of a solution τ to F is defined as

$$\text{COST}(F, \tau) = \sum_{C \in F_s} c(C) \cdot (1 - \tau(C)),$$

i.e., as the sum of the costs of the soft clauses not satisfied by τ. A solution τ is (globally) *optimal* for F if $\text{COST}(F, \tau) \leq \text{COST}(F, \tau')$ holds for any solution τ' to F. The cost of the optimal solutions of F is denoted by $\text{OPT}(F)$. Given a weighted partial MaxSAT instance F, the weighted partial MaxSAT problem asks to find an optimal solution to F. For simplicity, we will from here on drop the term "weighted partial" when referring to weighted partial MaxSAT instances, and simply refer to them as MaxSAT instances.

Example 4. *As an example of modeling problems with MaxSAT, consider the* 3*-coloring problem for the graph in Figure 4. The coloring problem can be modeled with MaxSAT by forming a MaxSAT instance* $F = (F_h, F_s, c)$ *using a set of* 15 *boolean variables,* $\{r_i, b_i, g_i \mid i = 1..5\}$*. The intended semantics of a variable* r_x *is that the node* x *is colored red, similarly for* g_x *(green) and* b_x *(blue). The hard clauses in* F *restrict each node to be colored with exactly one color and the soft clauses represent the constraints forcing each pair of nodes sharing an edge to be colored with different colors. As clauses, this corresponds to*

$$\begin{aligned} F_h \quad = \quad & \{(r_1 \vee b_1 \vee g_1), (r_2 \vee b_2 \vee g_2), (r_3 \vee b_3 \vee g_3), (r_4 \vee b_4 \vee g_4), (r_5 \vee b_5 \vee g_5), \\ & (\neg r_1 \vee \neg g_1), (\neg r_1 \vee \neg b_1), (\neg b_1 \vee \neg g_1), (\neg r_2 \vee \neg g_2), (\neg r_2 \vee \neg b_2), (\neg b_2 \vee \neg g_2), \\ & (\neg r_3 \vee \neg g_3), (\neg r_3 \vee \neg b_3), (\neg b_3 \vee \neg g_3), (\neg r_4 \vee \neg g_4), (\neg r_4 \vee \neg b_4), (\neg b_4 \vee \neg g_4), \\ & (\neg r_5 \vee \neg g_5), (\neg r_5 \vee \neg b_5), (\neg b_5 \vee \neg g_5)\}, \end{aligned}$$

and

$$\begin{aligned} F_s \quad = \quad & \{(\neg r_1 \vee \neg r_2), (\neg b_1 \vee \neg b_2), (\neg g_1 \vee \neg g_2), (\neg r_1 \vee \neg r_3), (\neg b_1 \vee \neg b_3), (\neg g_1 \vee \neg g_3), \\ & (\neg r_1 \vee \neg r_4), (\neg b_1 \vee \neg b_4), (\neg g_1 \vee \neg g_4), (\neg r_2 \vee \neg r_3), (\neg b_2 \vee \neg b_3), (\neg g_2 \vee \neg g_3), \\ & (\neg r_2 \vee \neg r_4), (\neg b_2 \vee \neg b_4), (\neg g_2 \vee \neg g_4), (\neg r_3 \vee \neg r_4), (\neg b_3 \vee \neg b_4), (\neg g_3 \vee \neg g_4), \\ & (\neg r_3 \vee \neg r_5), (\neg b_3 \vee \neg b_5), (\neg g_3 \vee \neg g_5), (\neg r_4 \vee \neg r_5), (\neg b_4 \vee \neg b_5), (\neg g_4 \vee \neg g_5)\} \end{aligned}$$

with $c(w) = 1$ *for all* $w \in F_s$*. An optimal solution* τ *to* F *is* $\tau(r_1) = \tau(r_3) = \tau(b_5) = \tau(b_2) = \tau(g_4) = 1$ *and* $\tau(x) = 0$ *for all other variables. The cost of this solution is* 1*, proving that any* 3*-coloring of the graph in Figure 4 has to assign the same color to at least one pair of nodes sharing an edge.*

5.2. Solving MaxSAT

Recent advances in MaxSAT solvers make MaxSAT a viable approach to finding globally (cost-)optimal solutions to various optimization problems with successful real-world applications such as hardware design

debugging [39], post-silicon and C-code fault localization [40, 41], reasoning over biological networks [42], and optimal Bayesian network structure learning [43]. As both SAT solvers and MaxSAT solvers continue improving, it is becoming commonly accepted that large problems can be solved in practice [44] and that the computational time is very much an empirical question and often not dominated by theoretical worst-case complexity. Indeed, MaxSAT is an active area of research [45, 46, 47, 48, 49]. We next provide a short overview of MaxSAT solvers. For a more comprehensive discussion, we refer the reader to [50, 51].

Many of the state-of-the-art MaxSAT solvers aimed at efficiently solving real-world instances in practice make use of a SAT solver as a subroutine. By *relaxing* the soft clauses in the input formula, the MaxSAT solver can linearly search for the optimal solution to the instance by querying the SAT solver for the existence of a truth assignment (not) satisfying at least (at most) k soft clauses for different values of k. Intuitively, k can then either be an upper [46] or a lower bound [52, 53] for the optimal solution. Another often used search strategy is binary search [49, 47]. This basic idea of the algorithm has been improved by exploiting the fact that whenever invoked on an unsatisfiable set of clauses, a modern SAT-solver can produce proof of unsatisfiability in the form of a (small) subset of the input clauses that in itself is unsatisfiable. These subsets are commonly referred to as *unsatisfiable cores* [52, 47, 54]. By using the information provided by the cores, MaxSAT solver can relax soft cores on demand, instead of having to relax all of them upfront. Solvers following this strategy are referred to as *core-guided solvers*. Other proposed methods for MaxSAT solving include incorporating integer linear programming techniques, either as one part of the solving algorithm [55] or by directly encoding the MaxSAT instance as an instance of integer linear programming [56].

In this work, we extend the application domains of MaxSAT to correlation clustering by presenting three different encodings for finding optimal solutions to the correlation clustering problem. Given a symmetric similarity matrix W over a set V of data points (recall Section 2.1), the basic idea behind all of our MaxSAT formulations of correlation clustering is that hard clauses are used to enforce that any solution to the MaxSAT instance represents a well-defined clustering (i.e., a mapping $cl\colon V \to \mathbb{N}$). The soft clauses are used to encode the cost function in a faithful way, so that each solution to the MaxSAT instance can be mapped into a clustering with exactly the same cost. In this way the optimal solution of the created MaxSAT instance can be mapped into the optimal clustering of the correlation clustering problem. Next we will present all three encodings in detail.

6. A MaxSAT Formulation of Correlation Clustering: Transitive Encoding

Our first MaxSAT formulation, the *transitive encoding*, of correlation clustering can be viewed as a simple reformulation of the integer linear programming formulation (recall Section 3) in terms of MaxSAT.

Similarly as in the ILP formulation, we use boolean variables x_{ij}, where $1 \leq i < j \leq N$, with the interpretation that $x_{ij} = 1$ iff points v_i and v_j are co-clustered[2]. We again adopt the notational convenience $x_{ij} = x_{ji}$. Now the transitive encoding forms the MaxSAT instance $F^1 = (F^1_h, F^1_s, c)$ summarized in Figure 5.

We next describe the different parts of F^1 in detail.

6.1. Hard Clauses

The hard clauses F^1_h of the transitive encoding are a clausal formulation of the *transitivity constraints* ($x_{ij} + x_{jk} \leq 1 + x_{ik}$ for all distinct i, j, k) of the ILP formulation. In terms of propositional logic, these can

[2]Unlike the two other MaxSAT encodings considered in this work, the transitive encoding does not directly allow for enforcing an upper bounds of less than N on the number of clusters.

Hard Clauses F_h^1:	$(\neg x_{ij} \vee \neg x_{jk} \vee x_{ik})$	for all $(v_i, v_j, v_k) \in V^3$ where i, j, k are distinct
Must-Link	(x_{ij})	for all $i < j$ s.t. $W(i,j) = \infty$
Cannot-Link	$(\neg x_{ij})$	for all $i < j$ s.t. $W(i,j) = -\infty$
Soft Clauses F_s^1:	(x_{ij})	for all similar v_i, v_j s.t. $i < j$
	$(\neg x_{ij})$	for all dissimilar v_i, v_j s.t. $i < j$
Cost c of soft clauses	$c((x_{ij})) = W(i,j)$	for all similar v_i, v_j s.t. $i < j$
	$c((\neg x_{ij})) = \lvert W(i,j) \rvert$	for all dissimilar v_i, v_j s.t. $i < j$

Figure 5: MaxSAT instance $F^1 = (F_h^1, F_s^1, c)$ produced by the transitive encoding.

be stated as $(x_{ij} \wedge x_{jk}) \rightarrow x_{ik}$, which in clausal form corresponds to

$$(\neg x_{ij} \vee \neg x_{jk} \vee x_{ik})\,.$$

6.2. Soft Clauses

The soft clauses F_s^1 encode the cost function. Each dissimilar pair of points v_i and v_j ($-\infty < W(i,j) < 0$) that are co-clustered corresponds to exactly one unsatisfied soft clause with weight $-W(i,j)$, and similarly, each similar pair of points v_i and v_j ($\infty > W(i,j) > 0$) that are assigned to different clusters corresponds to one unsatisfied soft clause with weight $W(i,j)$. These conditions are captured by the unit soft clauses $(\neg x_{ij})$ and (x_{ij}), respectively, with weights set to $|W(i,j)|$.

6.3. Encoding Constrained Clustering

The transitive encoding extends naturally to constrained correlation clustering with ML and CL constraints. For each $W(i,j) = \infty$, v_i and v_j are forced to be co-clustered. This is achieved with the *hard* clause (x_{ij}). Similarly, for each $W(i,j) = -\infty$, points v_i and v_j are forced to different clusters, which is achieved by the hard clause $(\neg x_{ij})$. In addition to ML and CL, various types of other constraints can be expressed.

Example 5. *(Running example of further constraints) We will use a running example of encoding additional constraints under the three MaxSAT encodings considered in the work, highlighting some of the differences between the encodings. As an example, consider the constraint* NOTCOCLUSTERED(i,j,t) *forbidding a triple of points v_i, v_j and v_t from being co-clustered. Under the transitive encodings, this constraint can be encoded as a single clause* NOTCOCLUSTERED$(i,j,t) := (\neg x_{ij} \vee \neg x_{jt} \vee \neg x_{it})$*. As another example, consider the cluster-level constraint* ATMOSTINALL(k) *requiring each cluster to contain at most k data points. This constraint can be reformulated as requiring that each data point v_i is co-clustered with at most $k-1$ other data points. For a fixed data point v_i the latter formulation can be encoded as a* cardinality constraint $\sum_{j=\{1,\ldots,N\}\setminus\{i\}} x_{ij} \leq (k-1)$ *requiring at most $k-1$ of the variables $x_{i1}, \ldots x_{iN}$ to be set to true, which can further be encoded with hard clauses using one of the several compact cardinality constraints; see e.g. [57, 58]. The whole* ATMOSTINALL(k) *constraint decomposes in to a conjunction of such cardinality constraints over i.*

6.4. Constructing a Clustering from a MaxSAT Solution to the Transitive Encoding

Any solution τ to F^1 represents a valid clustering cl_τ of V, constructed in an iterative manner as follows.

While there still are unassigned points left:

1. Let i be the smallest index for which $cl_\tau(v_i)$ is not defined yet and let j be the iteration number ($j = 1...$).
2. Assign $cl_\tau(v_i) = j$.
3. Assign $cl_\tau(v_k) = j$ for all still unassigned v_k for which $\tau(x_{ik}) = 1$.

The fact that cl_τ is well-defined follows from the observation that each point gets assigned to at most one cluster and each iteration of the procedure assigns at least one point to a cluster. Furthermore, the hard transitivity constraints in F^1 ensure that the intended semantics of the x_{ij} variables hold in cl_τ. Hence it follows that the optimal solutions of F^1 correspond to the optimal clusterings of V. The correctness of the transitive encoding can be formalized as follows.

Theorem 2. *Given a set V of data points and a symmetric similarity matrix W over V, let F^1 be the MaxSAT instance produced by the transitive encoding on W. The clustering $cl_{\tau^*} : V \to \mathbb{N}$ constructed from an optimal solution τ^* to F^1 is an optimal clustering of V.*

A detailed proof of the theorem is given in Appendix A.

We note that the transitive encoding does not require a predefined number of clusters. This is avoided by the definition of the x_{ij} variables, interpreted as pairwise indicator variables for two data points v_i, v_j being assigned to the same cluster. However, the encoding is not very compact. Its size is similar to the ILP presented earlier, $\mathcal{O}(N^2)$ variables and $\mathcal{O}(N^3)$ clauses, suggesting that also this encoding does not scale well. Next we will present a *unary encoding* of correlation clustering into MaxSAT, which to some extent addresses the compactness issue of the transitive encoding.

7. An Unary Encoding of Correlation Clustering into MaxSAT

We now consider a more compact *unary encoding*, which to some extent resembles the quadratic integer programming formulation presented in Section 4. Similarly to the QIP, the unary encoding allows an upper bound K on the number of available clusters. By letting $K = N$, the set of clusterings produced by the unary encoding is exactly the same as for the transitive encoding. The size of the unary encoding is $\mathcal{O}(E \cdot K + N \cdot K)$ variables and $\mathcal{O}(E \cdot K)$ clauses where E is the number of nonzero values in the input similarity matrix W. Due to the dependence on E, in practice the unary encoding is more compact than the transitive encoding whenever the input matrix contains 0-entries or $K < N$.

The unary encoding involves $N \cdot K$ boolean variables y_i^k, where $i = 1..N$ (the number of data points) and $k = 1..K$ (the number of clusters). The intended interpretation of these variables is that $y_i^k = 1$ iff point v_i belongs to cluster k. Furthermore, the encoding employs two types of auxiliary variables.

- A_{ij}^k, where $i = 1..N$, $j = 2..N$, $i < j$, $W(i, j) > 0$, and $k = 1..K$, with the interpretation $A_{ij}^k = 1$ iff points v_i and v_j are both assigned to cluster k.
- D_{ij}, where $i = 1..N$, $j = 2..N$, $i < j$, and $W(i, j) < 0$, with the interpretation that if $D_{ij} = 0$, then points v_i and v_j are assigned to different clusters.

These variables are used for compactly encoding the similarity and dissimilarity constraints. We will next present details on the clauses used in the unary encoding. As with the transitive encoding, the hard clauses limit the set of solutions to well-defined clusterings, and the soft clauses encode the cost function in a faithful way. However, the hard and soft clauses differ significantly from the clauses in the transitive encoding. Most notably, both hard and soft clauses are included in the unary encoding for encoding the similarity and dissimilarity constraints.

Concretely, the unary encoding forms the MaxSAT instance $F^2 = (F_h^2, F_s^2, c)$ summarized in Figure 6.

Hard Clauses F_h^2:	$\textsc{ExactlyOne}(i)$	for all $v_i \in V$
	$\textsc{HardSimilar}(i,j,k)$	for all similar v_i, v_j s.t. $i < j$ and $1 \leq k \leq K$
	$\textsc{HardDissimilar}(i,j,k)$	for all dissimilar v_i, v_j s.t. $i < j$ and $1 \leq k \leq K$
Must-Link	$\text{ML}^U(v_i, v_j)$	for all $i < j$ s.t. $W(i,j) = \infty$
Cannot-Link	$\text{CL}^U(v_i, v_j)$	for all $i < j$ s.t. $W(i,j) = -\infty$
Soft Clauses F_s^2:	$\textsc{SoftSimilar}(i,j)$	for all similar v_i, v_j s.t. $i < j$
	$\textsc{SoftDissimilar}(i,j)$	for all dissimilar v_i, v_j s.t. $i < j$
Cost c of soft clauses	$c(\textsc{SoftSimilar}(i,j)) = W(i,j)$	for all similar v_i, v_j s.t. $i < j$
	$c(\textsc{SoftDissimilar}(i,j)) = \lvert W(i,j) \rvert$	for all dissimilar v_i, v_j s.t. $i < j$

Figure 6: MaxSAT instance $F^2 = (F_h^2, F_s^2, c)$ produced by the unary encoding.

We next describe the different parts of F^2 in detail.

7.1. Ensuring Well-Defined Clusterings

The hard constraints $\textsc{ExactlyOne}(i)$ constrain the search to well-defined clusterings by enforcing that each data point v_i is assigned into exactly one cluster k. In terms of the variables in the encoding this means that, for each i, exactly one of the variables $y_i^1, \ldots, y_i^K$ should be assigned to 1, i.e.,

$$\textsc{ExactlyOne}(i) := \sum_{k=1}^{K} y_i^k = 1.$$

A number of different encodings of this cardinality constraint as clauses have been previously developed [59]. In our experiments, we used the so-called *sequential encoding* [60] which is linear, or more precisely, introduces $3K - 4$ clauses and $K - 1$ auxiliary variables for each i. We refer the interested reader to [60] for a detailed description of this encoding.

7.2. Encoding Similarity

For a similar pair of data points v_i and v_j, the constraints $\textsc{HardSimilar}(i,j,k)$ for each $k = 1..K$ and $\textsc{SoftSimilar}(i,j)$ together enforce the requirement that v_i and v_j are assigned to the same cluster whenever the soft constraint $\textsc{SoftSimilar}(i,j)$ is satisfied. In terms of propositional logic, this requirement can be expressed as the formula

$$(y_i^1 \wedge y_j^1) \vee (y_i^2 \wedge y_j^2) \vee \ldots \vee (y_i^K \wedge y_j^K).$$

In order to the express this propositional formula as clauses, we employ the auxiliary variables A_{ij}^k and define the semantics of these to be $\tau(A_{ij}^k) = 1$ iff $\tau(y_i^k \wedge y_j^k) = 1$. In terms of propositional logic, the defining constraint is $A_{ij}^k \leftrightarrow (y_i^k \wedge y_j^k)$, which can be expressed as

$$\text{HARDSIMILAR}(i, j, k) := \{(\neg A_{ij}^k \vee y_i^k), (\neg A_{ij}^k \vee y_j^k), (A_{ij}^k \vee \neg y_i^k \vee \neg y_j^k)\}.$$

We note that the definitions of the auxiliary variables do not yet enforce points v_i and v_j to be assigned to cluster k. Instead, the clauses $\text{HARDSIMILAR}(i, j, k)$ state that the variable A_{ij}^k is set to true if and only if points v_i and v_j are both assigned to cluster k. This must hold in every solution to F^2, hence the clauses are *hard*.

Using the auxiliary variables, the soft constraint expressing that the points v_i and v_j are assigned to the same cluster can be encoded as the clause

$$\text{SOFTSIMILAR}(i, j) := (A_{ij}^1 \vee \cdots \vee A_{ij}^K) \text{ with weight } c(\text{SOFTSIMILAR}(i, j)) = W(i, j).$$

For some intuition, we note that if this clause is satisfied in a solution τ, then for some k, $\tau(A_{ij}^k) = 1$. Since all hard clauses are satisfied in any solution, it follows that points v_i and v_j will be assigned to cluster k, exactly as required. Similarly, if points v_i and v_j are not assigned to the same cluster, then due to the hard constraints we have $\tau(A_{ij}^k) = 0$ for all k, and the soft clause will not be satisfied. Each unsatisfied clause must increase the cost of a MaxSAT solution according to the similarity values of the corresponding points, which is why the weight of the clause is set to $W(i, j)$.

7.3. Encoding Dissimilarity

For a dissimilar pair of data points v_i and v_j, the clauses $\text{HARDDISSIMILAR}(i, j, k)$ for each $k = 1..K$ and $\text{SOFTDISSIMILAR}(i, j)$ together enforce the requirement that v_i and v_j are assigned to different clusters. This can be expressed by requiring for each cluster that at least one of v_i and v_j should not be assigned to that cluster, which in clausal form is expressed by $(\neg y_i^k \vee \neg y_j^k)$ for a cluster k. The whole constraint enforcing v_i and v_j to be assigned to different clusters is hence

$$(\neg y_i^1 \vee \neg y_j^1) \wedge \ldots \wedge (\neg y_i^K \vee \neg y_j^K). \tag{5}$$

Equation 5 is already in clausal form. However, we want to make sure that breaking any of the individual clauses corresponds to a cost of $|W(i, j)|$. To achieve this, we use the auxiliary variables D_{ij}, and define them in terms of propositional logic as $\neg D_{ij} \rightarrow (\neg y_i^k \vee \neg y_j^k)$ for each cluster $k = 1..K$. That is, if $\tau(D_{ij}) = 0$ for some solution τ to F^2, then v_i and v_j are not assigned to the same cluster[3]. The defining constraint can be expressed as the hard clauses

$$\text{HARDDISSIMILAR}(i, j, k) := (D_{ij} \vee \neg y_i^k \vee \neg y_j^k).$$

The auxiliary variable D_{ij} makes it possible to express the soft constraint requiring v_i and v_j to not be co-clustered simply as

$$\text{SOFTDISSIMILAR}(i, j) := (\neg D_{ij}) \text{ with weight } c(\text{SOFTDISSIMILAR}(i, j)) = |W(i, j)|.$$

[3]The formalism behind grouped soft clauses like the ones in Equation 5 is known as *group-MaxSAT*. An exact treatment of group-MaxSAT is beyond the scope of this work, we refer the interested reader to [61].

For some intuition, we have that if the clause $(\neg D_{ij})$ is satisfied in a solution to F^2, then the clauses $(\neg y_i^k \vee \neg y_j^k)$ also have to be satisfied for all k. Hence points v_i and v_j are not assigned to the same cluster. On the other hand, if v_i and v_j are assigned to the same cluster k, then the solution has to assign $D_{ij} = 1$ in order to satisfy the hard clause $(D_{ij} \vee \neg y_i^k \vee \neg y_j^k)$, resulting in one unsatisfied clause with weight $|W(i,j)|$, exactly as required for representing the correlation clustering cost function faithfully.

7.4. Encoding Constrained Clustering

By noticing that for each $k = 1..K$ we need to enforce that $cl(v_i) = k$ iff $cl(v_j) = k$, the must-link constraint over v_i and V_j can be encoded under the unary encoding as

$$\text{ML}^U(v_i, v_j) := \{(\neg y_i^1 \vee y_j^1), (y_i^1 \vee \neg y_j^1), \ldots, (\neg y_i^K \vee y_j^K), (y_i^K \vee \neg y_j^K)\},$$

where the clauses $(\neg y_i^k \vee y_j^k)$ and $(y_i^k \vee \neg y_j^k)$ correspond to $y_i^k \leftrightarrow y_j^k$. For some intuition, in any solution τ we have that, whenever $\tau(y_i^k) = 1$, the solution has to assign $\tau(y_j^k) = 1$ in order to satisfy $(\neg y_i^k \vee y_j^k)$. Furthermore, based on the other hard clauses, we know that there exists exactly one $k = 1..K$ for which $\tau(y_i^k) = 1$, and hence $\tau(y_i^{k'}) = 0$ for all $k' \neq k$. Thus τ has to assign $\tau(y_j^{k'}) = 0$ in order to satisfy the clause $(y_i^{k'} \vee \neg y_j^{k'})$; hence the points are assigned to the same cluster.

The benefit of encoding the must-link constraint in this way compared to the similarity constraint presented earlier is the elimination of the auxiliary variables A_{ij}^k and hence a decrease in the number of clauses generated. On the other hand, similarity constraints cannot be encoded directly in this way since they are soft. Furthermore, whenever a similarity constraint is not satisfied, the cost added to a MaxSAT solution should be exactly the corresponding similarity value, which is controlled in a simple way with the A_{ij}^k variables.

Cannot-link constraints in the unary encoding can also be encoded more compactly than the dissimilarity constraints. The variable D_{ij} in the encoding is used to ensure that an unsatisfied dissimilarity constraint corresponds exactly to cost $|W(i,j)|$. If we know that the dissimilarity constraint has to be satisfied (making it a hard cannot-link constraint), we can simply leave out the extra variable. The intuition between the cannot-link clauses is that for each $k = 1..K$ and any solution τ, either $\tau(y_i^k) = 0$ or $\tau(y_j^k) = 0$. Stated as clauses, we have

$$\text{CL}^U(v_i, v_j) := \{(\neg y_i^1 \vee \neg y_j^1), \ldots, (\neg y_i^K \vee \neg y_j^K)\}.$$

Example 6. *(Running example of further constraints continued.)*

Under the unary encoding, the NOTCOCLUSTERED(i,j,t) *constraint, forbidding all three of the points v_i, v_j and v_t from being co-clustered, can be encoded with a set of K clauses*

$$\text{NOTCOCLUSTERED}(i,j,t) := \{(\neg y_i^1 \vee \neg y_j^1 \vee \neg y_t^1), \ldots, (\neg y_i^K \vee \neg y_j^K \vee \neg y_t^K)\}.$$

The constraint includes one clause for each cluster $s = 1..K$, each forbidding all three points from being assigned to cluster s. The constraint ATMOSTINALL(k)*, requiring each cluster to contain at most k data points, can be encoded as a conjunction of K cardinality constraints, namely, by enforcing $\sum_{i=1}^{N} y_i^j \leq k$ over each cluster index j.*

7.5. Constructing a Clustering from a MaxSAT Solution to the Unary Encoding

Given a solution τ to F^2, we can easily construct a corresponding well-defined clustering cl_τ of the data points by assigning each point v_i into the cluster k for which $\tau(y_i^k) = 1$. Due to the hard constraints F_h^2, in any solution τ there is exactly one such k for every i. Especially, the clustering constructed from an optimal solution to F^2 will be an optimal clustering of the data, minimizing the correlation clustering objective function. This correctness of the unary encoding can be formalized as follows.

Theorem 3. *Given a set V of data points with $|V| = N$, a symmetric similarity matrix W over V, and an upper limit K on the available clusters such that $1 \leq K \leq N$, let F^2 be the MaxSAT instance produced by the unary encoding on W. The clustering $cl_{\tau^*} : V \to \{1, \dots K\}$ constructed from an optimal solution τ^* to F^2 is an optimal clustering of V over all clusterings $cl : V \to \{1, \dots K\}$. In other words, cl_{τ^*} is optimal over all clusterings of V that use at most K clusters.*

Intuitively, the theorem follows from the already discussed connections between cost incurred by a clustering and the weight of unsatisfied soft clauses in the unary encoding. A proof of Theorem 3 is provided in Appendix A.

8. A Binary Encoding of Correlation Clustering into MaxSAT

As the third encoding, we describe a *binary encoding* of correlation clustering as MaxSAT, which is essentially a bitwise reformulation of the unary encoding. Similarly to the unary encoding, the binary encoding allows an upper limit K on the available clusters. As is often the case with SAT and MaxSAT encodings, the binary encoding is more compact than both the unary and the transitive encoding, regardless of the input similarity matrix or the value of K. An instance formed by the binary encoding contains $\mathcal{O}(E + N \cdot \log_2 K)$ variables and $\mathcal{O}(E \cdot \log_2 K)$ clauses, where E is the number of nonzero values in the input similarity matrix W.

For simplicity, we first assume that K is a power of 2, more precisely $K = 2^a$ for some $a \in \mathbb{N}$. From this it follows that $\log_2 K = a$ is an integer. The encoding also works if this is not the case; we will describe the required adaptations in Section 8.5. The encoding uses a variables b_i^k where $1 \leq k \leq a$ for each point v_i. The intended semantics of these variables is that point v_i is assigned to cluster index $b_i^a..b_i^1$, interpreted as a binary number with the least significant bit to the right. Additionally, we employ two types of auxiliary variables.

- EQ_{ij}^k, where $1 \leq i < j \leq N$, $W(i,j) \in \mathbb{R} \setminus \{0\}$, and $1 \leq k \leq a$. The intended semantics of EQ_{ij}^k is $EQ_{ij}^k = 1$ iff $b_i^k = b_j^k$.
- S_{ij}, where $1 \leq i < j \leq N$ and $W(i,j) \in \mathbb{R} \setminus \{0\}$. $S_{ij} = 1$ iff points v_i and v_j are co-clustered. (Note the equivalence: also, if $S_{ij} = 0$, then points v_i and v_j are not assigned to the same cluster.)

Hard Clauses F_h^3:	$\textsc{Equality}(i,j,k)$	for all $W(i,j) \in \mathbb{R} \setminus \{0\}$
	$\textsc{SameCluster}(i,j)$	for all $W(i,j) \in \mathbb{R} \setminus \{0\}$
Must-Link	$\text{ML}^B(v_i, v_j)$	for all $i < j$ s.t. $W(i,j) = \infty$
Cannot-Link	$\text{CL}^B(v_i, v_j)$	for all $i < j$ s.t. $W(i,j) = -\infty$
If $K \neq N$ and $K \neq 2^a$ for any a	$\textsc{ClustersLessThan}(i,K)$	for all $v_i \in V$
Soft Clauses F_s^3:	(S_{ij})	for all similar v_i, v_j s.t. $i < j$
	$(\neg S_{ij})$	for all dissimilar v_i, v_j s.t. $i < j$
Cost c of soft clauses	$c((S_{ij})) = W(i,j)$	for all similar v_i, v_j s.t. $i < j$
	$c((\neg S_{ij})) = \lvert W(i,j) \rvert$	for all dissimilar v_i, v_j s.t. $i < j$

Figure 7: MaxSAT instance $F^3 = (F_h^3, F_s^3, c)$ produced by the binary encoding.

An instance F^3 produced by the binary encoding is summarized in Figure 7. This time, the only hard clauses required are the clauses defining the auxiliary variables. This is due to the fact that any MaxSAT

solution has to assign all the variables b_i^k in some unique way, and hence any solution will represent a well-defined clustering. We next describe the binary encoding in more detail.

8.1. Hard Clauses

As the b_i^k variables form the bit-representation of the cluster index of point v_i, the question of whether two points v_i and v_j are assigned to the same cluster is equivalent to whether the values of b_i^k and b_j^k are equal for all $1 \leq k \leq a$. In order to reason about the equality of individual bits, the binary encoding uses a "equality" variables EQ_{ij}^k for each pair of points v_i and v_j for which $i < j$ and $W(i,j) \in \mathbb{R} \setminus \{0\}$. These variables are defined to be equivalent to $\tau(b_i^k) = \tau(b_j^k)$ when τ is a solution to F^3. In terms of propositional logic, the defining constraint is $EQ_{ij}^k \leftrightarrow (b_i^k \leftrightarrow b_j^k)$, which corresponds to the set of clauses

$$\text{EQUALITY}(i,j,k) := \{(EQ_{ij}^k \vee b_i^k \vee b_j^k), (EQ_{ij}^k \vee \neg b_i^k \vee \neg b_j^k), (\neg EQ_{ij} \vee \neg b_i^k \vee b_j^k), (\neg EQ_{ij} \vee b_i^k \vee \neg b_j^k)\}.$$

Encoding the semantics of the S_{ij} variables is straightforward using the equality variables. Two points v_i and v_j are assigned to the same cluster iff the values at each bit-position in the bit representation of their cluster indices are the same. Stated in propositional logic, we have $S_{ij} \leftrightarrow (EQ_{ij}^1 \wedge \ldots \wedge EQ_{ij}^a)$, which corresponds to

$$\text{SAMECLUSTER}(i,j) := \{(\neg S_{ij} \vee EQ_{ij}^1), \ldots, (\neg S_{ij} \vee EQ_{ij}^a), (S_{ij} \vee \neg EQ_{ij}^1 \vee \ldots \vee \neg EQ_{ij}^a)\}.$$

8.2. Soft Clauses

As the variable S_{ij} has the exact same semantics as the variable x_{ij} in the transitive encoding, it can be used to formulate the soft clauses of the binary encoding in a very similar manner as the soft clauses in the transitive encoding. For every similar pair of points v_i and v_j, the cost of the clustering should increase by $W(i,j)$ whenever the points are not assigned to the same cluster. This condition is encoded by the unit soft clause (S_{ij}) with weight $c((S_{ij})) = W(i,j)$. Analogously, for every dissimilar pair the instance includes the soft clause $(\neg S_{ij})$ with weight $c((\neg S_{ij})) = |W(i,j)|$.

8.3. Encoding Constrained Clustering

For compactly encoding the must-link constraint in the binary encoding, we simplify the similarity constraint. We need to ensure that $\tau(b_i^k) = \tau(b_j^k)$ for all bits $k = 1..a$ and all MaxSAT solutions τ. For a fixed k, this can be stated as $(b_i^k \leftrightarrow b_j^k)$, which as clauses is expressed by $(\neg b_i^k \vee b_j^k), (b_i^k \vee \neg b_j^k)$. Hence the whole must-link constraint is

$$\text{ML}^B(v_i, v_j) := \{(\neg b_i^1 \vee b_j^1), (b_i^1 \vee \neg b_j^1), \ldots, (\neg b_i^a \vee b_j^a), (b_i^a \vee \neg b_j^a)\}.$$

The cannot-link constraint can be seen as a simplified dissimilarity constraint. The variable EQ_{ij}^k and the clauses defining it are still required for all bits. However, the cannot-link constraint can be stated as a single clause: we simply require that there exists a bit-position k such that the values b_i^k and b_j^k differ. The whole cannot-link constraint is

$$\text{CL}^B(v_i, v_j) := \{\text{EQUALITY}(i,j,1), \ldots, \text{EQUALITY}(i,j,a), (\neg EQ_{ij}^1 \vee \ldots \vee \neg EQ_{ij}^a)\}.$$

Example 7. *(Running example of further constraints continued.) Due to the similar semantics of the S_{ij} variables of the binary encoding and the x_{ij} variables of the transitive encoding, both of our example constraints can be encoded very similarly to the transitive encoding. The* NOTCOCLUSTERED(i,j,t) *constraint, forbidding all three of the points v_i, v_j and v_t from being co-clustered, can be encoded by a single*

clause NOTCOCLUSTERED$(i, j, t) := (\neg S_{ij} \vee \neg S_{it} \vee \neg S_{jt})$, *i.e., more compactly than in the unary encoding directly. Also, the* ATMOSTINALL(k) *constraint can be encoded similarly to the transitive encoding by using for each v_i, a cardinality constraint forbidding v_i from being co-clustered with mode than $k-1$ other points:* $\sum_{j=\{1\ldots N\}\setminus\{i\}} S_{ij} \leq k-1$.

8.4. Constructing a Clustering from a MaxSAT Solution to the Binary Encoding

Given a solution τ to F^3, there is again a very natural way of constructing a clustering of V. For each data point v_i, let $\tau(b_i^a)\tau(b_i^{a-1})\ldots\tau(b_i^1) = c$, where the left hand side is interpreted as a binary number, and assign $cl_\tau(v_i) = c+1$. Since the number of available bits is $\log_2 K$, it follows that $0 \leq c \leq K-1$, and hence $1 \leq cl_\tau(v_i) \leq K$ holds for all v_i. The clustering constructed from an optimal solution to F^3 is optimal amongst all clusterings using at most K clusters.

Theorem 4. *Given a set V of data points with $|V| = N$, a symmetric similarity matrix W over V, and an upper limit K on the available clusters such that $1 \leq K \leq N$, let F^3 be the MaxSAT instance produced by the binary encoding on W. The clustering $cl_{\tau^*}\colon V \to \{1,\ldots K\}$ constructed from an optimal solution τ^* to F^3 is an optimal clustering of V under W over all clusterings $cl\colon V \to \{1,\ldots K\}$. In other words, cl_{τ^*} is optimal over all clusterings of W that use at most K clusters.*

A proof of this theorem is provided in Appendix A.

8.5. The Binary Encoding for General K

So far we have assumed that the upper limit on the available clusters is an power of 2, or, more precisely, that $K = 2^a$ for some a. This assumption simplifies the binary encoding since the values representable in binary with a bits are exactly 0 to $2^a - 1$. It is also possible to constraint K to an arbitrary value. A simple approach would be to encode a separate constraint for each point v_i and each value $j \in \{K, K+1, \ldots, 2^a - 1\}$ forbidding the value of the bit variables $b_i^a, \ldots, b_i^1$ (interpreted as a binary number) from being equal to j. However, this would result in $\mathcal{O}(N^2 \cdot \log_2 N)$ clauses, the same as the worst-case size of the whole encoding.

A more compact formulation can be obtained by observing that, for each data point we only need to encode a single constraint stating that the value of its assigned cluster index should be *less than* K. For a given K, let K^j denote the value of the jth bit in the binary representation of K. Note that as $2^{a-1} < K \leq 2^a$, there are exactly a bits in the binary representation of K. For any set of bit variables $b_i^a, \ldots, b_i^1$, denote the value represented by these variables in binary by $(b_i^a \ldots b_i^1)_2$. For a given datapoint v_i we can encode the constraint $(b_i^a \ldots b_i^1)_2 < K$ recursively using the observation that a binary number $(b_i^a \ldots b_i^1)_2$ is less than another binary number $(K^a \ldots K^1)_2$ iff

- $K^a = 1$ and $b_i^a = 0$, or
- $K^a = b_i^a$ and $(b_i^{a-1} \ldots b_i^1)_2 < (K^{a-1} \ldots K^1)_2$.

This formulation of inequality between binary numbers follows directly from the properties of binary numbers. We encode it as MaxSAT by introducing a fresh variables B_i^j, $1 \leq j \leq a$, and adding clauses defining them recursively as

$$\begin{aligned} \text{DEFB}(i,1) &:= B_i^1 \leftrightarrow \left(\neg b_i^1 \wedge (K^1 = 1)\right), \\ \text{DEFB}(i,j) &:= B_i^j \leftrightarrow \Big((\neg b_i^j \wedge (K^j = 1)) \vee ((b_i^j \leftrightarrow K^j) \wedge B_i^{j-1})\Big). \end{aligned} \tag{6}$$

As the value of K is known, we can simplify the definition accordingly when adding the clauses to the encoding. Using these variables, the whole constraint limiting the number of clusters is enforced by the clauses defining the semantics of the B_i^j variables, together with N unit clauses, one for each data point:

$$\textsc{ClustersLessThan}(i, K) := \{\textsc{DefB}(i, 1), \ldots, \textsc{DefB}(i, a) \mid i = 1..N\} \cup \{(B_1^a), \ldots, (B_N^a)\}.$$

The size of this formulation is $\mathcal{O}(N \cdot \log_2(N))$.

9. Experimental Evaluation

We will now describe an experimental evaluation of our MaxSAT-based approach to correlation clustering.

9.1. Benchmarks

We experiment on real-world datasets consisting of similarity values between amino-acid sequences of different proteins [62], as well as similarity matrices we obtained from standard UCI benchmark datasets. For each of the obtained similarity matrices, we normalized the matrix entries to the range $[-0.5, 0.5]$.

9.1.1. Protein Sequence Datasets

We obtained four protein sequence datasets from `http://www.paccanarolab.org/scps`. The data consists of similarity values between amino-acid sequences, originally computed using BLAST [63]. All values were originally in the range $[0, 1.0]$. Normalization of the similarity information to the range $[-0.5, 0.5]$ was done by subtracting 0.5 from each entry. Table 1 shows the number of data points for each data set.

9.1.2. UCI Datasets

In addition to the protein sequence datasets, we produced similarity matrices based on the following UCI datasets.

- ORL: the AT&T ORL database of images of faces, each of size 92×112. Obtained from `http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`.
- Ionosphere: the UCI ionosphere dataset, for classification of radar returns from the ionosphere, originally with 34 attributes. Obtained from `http://archive.ics.uci.edu/ml/`.
- Umist: the Sheffield (previously UMIST) Face Database, each face image of size 92×112. Obtained from `http://www.sheffield.ac.uk/eee/research/iel/research/face`.
- Breastcancer: the LIBSVM breast-cancer dataset, originally named "Wisconsin Breast Cancer in UCI". The set contains 10 features.
  Obtained from `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.
- Diabetes: the LIBSVM diabetes dataset, originally from UCI, containing 8 features. Obtained from `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.
- Ecoli: the UCI Ecoli dataset, containing protein localization sites, with 8 features. Obtained from `http://archive.ics.uci.edu/ml/`.

Table 1: Number of data points in datasets considered

Dataset	Number of points
Ecoli	327
Ionosphere	351
ORL	400
Prot 3	567
Umist	575
Prot 2	586
Prot 4	654
Prot 1	669
Breastcancer	683
Diabetes	768
Vowel	990

- Vowel: the LIBSVM Vowel dataset, originally from UCI, with 10 features. Obtained from `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

For these datasets, we first calculated the normalized Euclidean distance between each pair of points, and directly interpreted the distances as similarity values by linear inverse mapping to the range $[-0.5, 0.5]$. In order to simulate incomplete similarity information, we finally modified all similarity values in the range $[-0.25, 0.25]$ to be 0. The size of each dataset is reported in Table 1.

9.1.3. Setup

For solving the MaxSAT instances resulting from our encodings, we used the academic off-the-shelf MaxSAT solver MaxHS [64, 55, 65] (MaxSAT evaluation 2013 version) obtained from the authors. MaxHS implements a hybrid approach to MaxSAT solving, combining the logical reasoning power of a SAT solver with the arithmetic reasoning power of an integer linear programming solver. During its execution, MaxHS maintains a set of unsatisfiable cores (recall Section 5.2). At each iteration, the ILP solver is used for finding a *minimum-cost hitting set* over the soft clauses in the current set of cores. Clauses in the hitting set are then temporarily removed from the instance and the SAT solver is invoked again. MaxHS terminates when the working formula is satisfiable, at which point the assignment returned by the SAT solver is an optimal solution to the MaxSAT instance. We note that MaxHS is by no means the only possible choice for a MaxSAT solver to use. We also report on a comparison of different state-of-the-art MaxSAT solvers in Section 9.4.3, the results of which motivate the use of MaxHS.

We compare the MaxSAT-based approach with exactly solving the integer linear programming and the quadratic integer programming formulations of correlation clustering (recall Sections 3 and 4, respectively). We used the commercial state-of-the-art integer programming solvers IBM CPLEX (version 12.6) and Gurobi Optimizer (version 6.0) for solving the integer linear programs, and additionally, the non-commercial SCIP [66] framework for solving the quadratic integer programs. Furthermore, we also compare to two approximative algorithms in terms of the cost of solutions obtained: the approximation algorithm KwickCluster (KC) proposed in [5] and further considered in [67], and the SDPC approach based on a semi-definite relaxation of the quadratic integer programming formulation, proposed in [34]. More details on these algorithms are provided in Section 10.1. For solving the semi-definite programs, we used the Matlab package SeDuMi 1.3 [68].

On the protein data we also experimented with the algorithms described in [62], available from `http://www.paccanarolab.org/scps`, which are specialized algorithms for correlation clustering protein sequences. The authors provide two algorithms that allow an unrestricted number of clusters by default. One is based on spectral clustering (SCPS) and the other on connected component analysis (CCA).

In addition to the comparative results, we also report on MaxSAT-specific experiments on the effect of MaxSAT-level preprocessing (in Section 9.4.1) and symmetry breaking (in Section 9.4.2) on solving times. We employed MaxSAT preprocessing in all experiments due to its positive impact on solving times. As for symmetry breaking on the MaxSAT-level in the other experiments, we only applied partial symmetry breaking to all formulas by enforcing the point with the lowest index to always be assigned to the first cluster.

A timeout of 8 hours and a memory limit of 30 GB were enforced on each individual run of a solver. The experiments were run under Linux on eight-core Intel Xeon E5440 2.8-GHz cluster nodes each with 32 GB of RAM. In order to ensure repeatable results, only a single algorithm on a single benchmark instance was executed on each cluster node at each time.

9.2. Experiments on Unconstrained Correlation Clustering

We first focus on unconstrained correlation clustering, i.e., correlation clustering under the assumption that there are no infinite values in the input similarity matrices.

9.2.1. Comparison of Algorithms Providing Optimal Solutions

We start with a comparison of the exact approaches to correlation clustering: our three MaxSAT encodings, the integer linear programming formulation (ILP), and the quadratic integer programming formulation (QIP). As the size of the transitive encoding and the integer linear program does not depend on the number of non-zero elements in the similarity matrix, for these experiments we created instances by varying the number of points $n \geq 50$ in the four protein datasets (Prot1, Prot2, Prot3 and Prot4) by considering only the n first rows and columns of the original similarity matrix of each data set.

The results are shown in Figure 8. The reason for the absence of the QIP approach from the plot is that neither CPLEX, Gurobi, nor SCIP was able to solve any of the quadratic programs exactly within the time limit. For example, SCIP was able to solve the instances when using 20 points within seconds, but was unable to solve 50 points within 8 hours. While we do not have a definitive explanation for this poor behaviour, one possible explanation may deal with the non-convexity (recall Section 4) of the QIP formulation of correlation clustering.[4] The transitive and the unary MaxSAT encodings, as well as the ILP approach, are competitive with the binary encoding only when the number of points is small. However, all three MaxSAT encodings scale better than ILP and QIP. Both CPLEX and Gurobi ran out of memory on the ILP formulation for instances larger than 300 points, suggesting it will fail to solve larger instance irrespective of the timeout. Furthermore, the encodings for which the size of the instance is not dependent on the number of non-zero entries in the similarity matrix cannot benefit from any sort of pruning that one might be able to do on the similarity values of the input data.

Based on these observations, for the MaxSAT-based approach we focus on the binary encoding in the rest of the experiments.

[4] This would be inline with behavior observed in other problem domains as well, see e.g. [69].

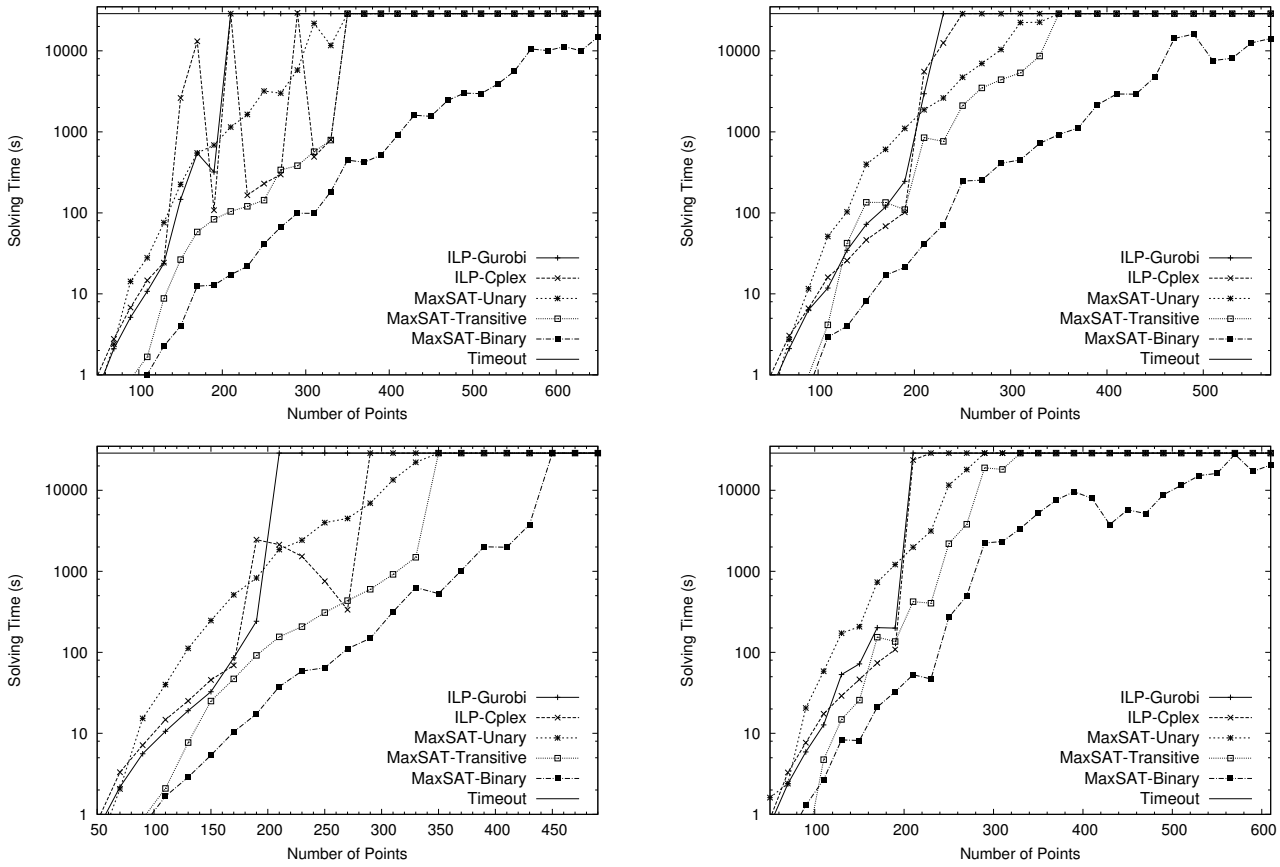

Figure 8: Point scalability of the exact approaches. Top left: Prot1, top right: Prot2, bottom left: Prot3, bottom right: Prot4.

9.2.2. Performance on Sparse Data

Next we simulate a setting in which the input data is sparse, that is, situations in which the similarity information available is incomplete. For $p \in \{0.05, 0.10, \ldots, 1\}$, we created instances from a given similarity matrix W by independently setting each non-zero element $W(i, j)$ to 0 with a probability $1 - p$. This results in a matrix W' where the expected number of non-zero entries is $p \cdot 100\%$ of the number of non zero entries in W.

We ran each of the approximative algorithms 100 times on each instance, and report the best values returned by them. We note that a single run of any of the approximative algorithms is very short, at most one minute for SDPC and within 10 seconds for the others.

Figures 9, 10 and 11 summarize the result of solving the sparse instances. A sparser matrix results in MaxSAT instances which are faster to solve. More importantly, however, we notice that MaxSAT is fairly robust when it comes to dealing with sparse data and the cost of the solution clustering. We experimentally compare the robustness of the different algorithms by calculating the cost $H(W, cl)$ for clusterings cl which were obtained with W' as input. This simulates a setting where there is some "true" objective function value that we would like the algorithms to optimize, but the amount of information available to the algorithms is limited/noisy. The cost of clusterings produced by the binary encoding is significantly lower than the other generic correlation clustering algorithms KwickCluster and SDPC for all values of p solvable by MaxSAT. For $p > 0.4$, the solutions obtained with MaxSAT have a clearly lower cost than the solutions provided by

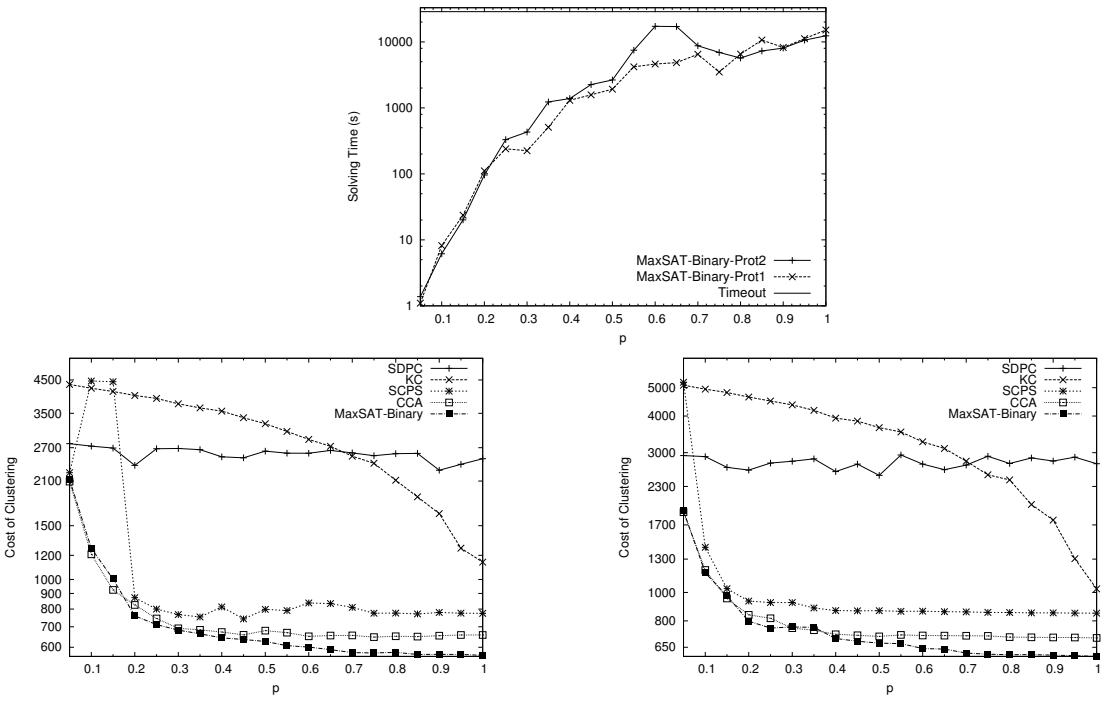

Figure 9: Top: Evolution of running times. Bottom: Cost of the clusterings obtained on sparse matrices. Bottom left: Prot1, bottom right: Prot2.

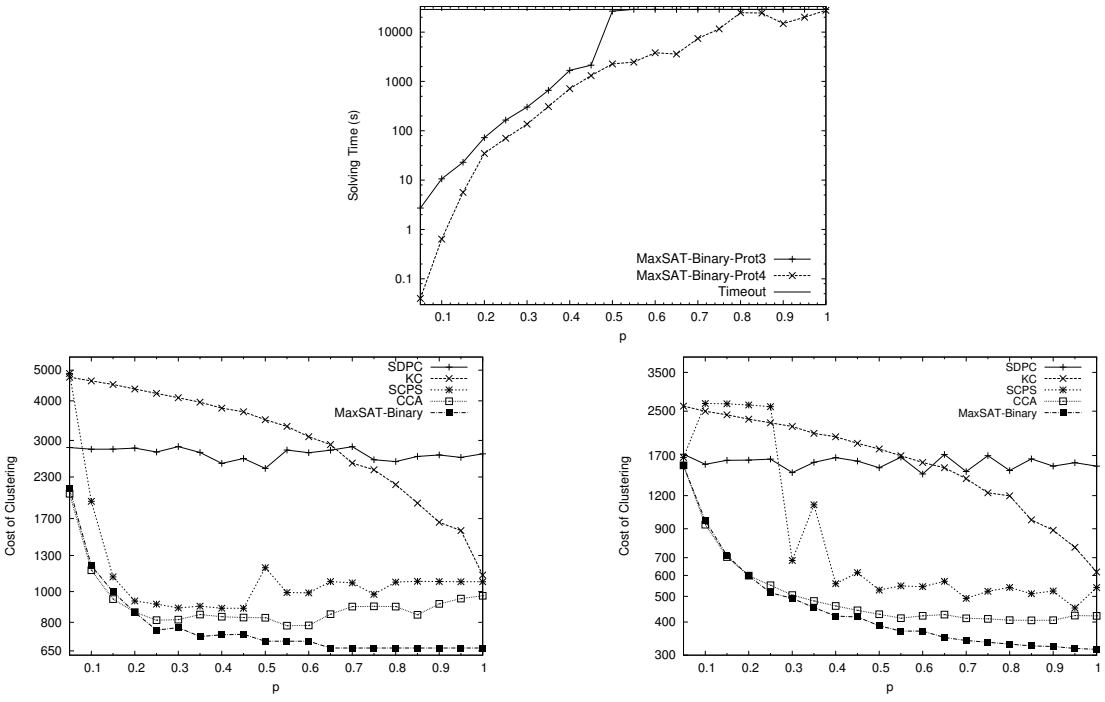

Figure 10: Top: evolution of running times. Bottom: cost of the clusterings obtained on sparse matrices. Bottom left: Prot3, Bottom right: Prot4. For the unsolvable MaxSAT instances we report the cost of the highest value of p still solvable.

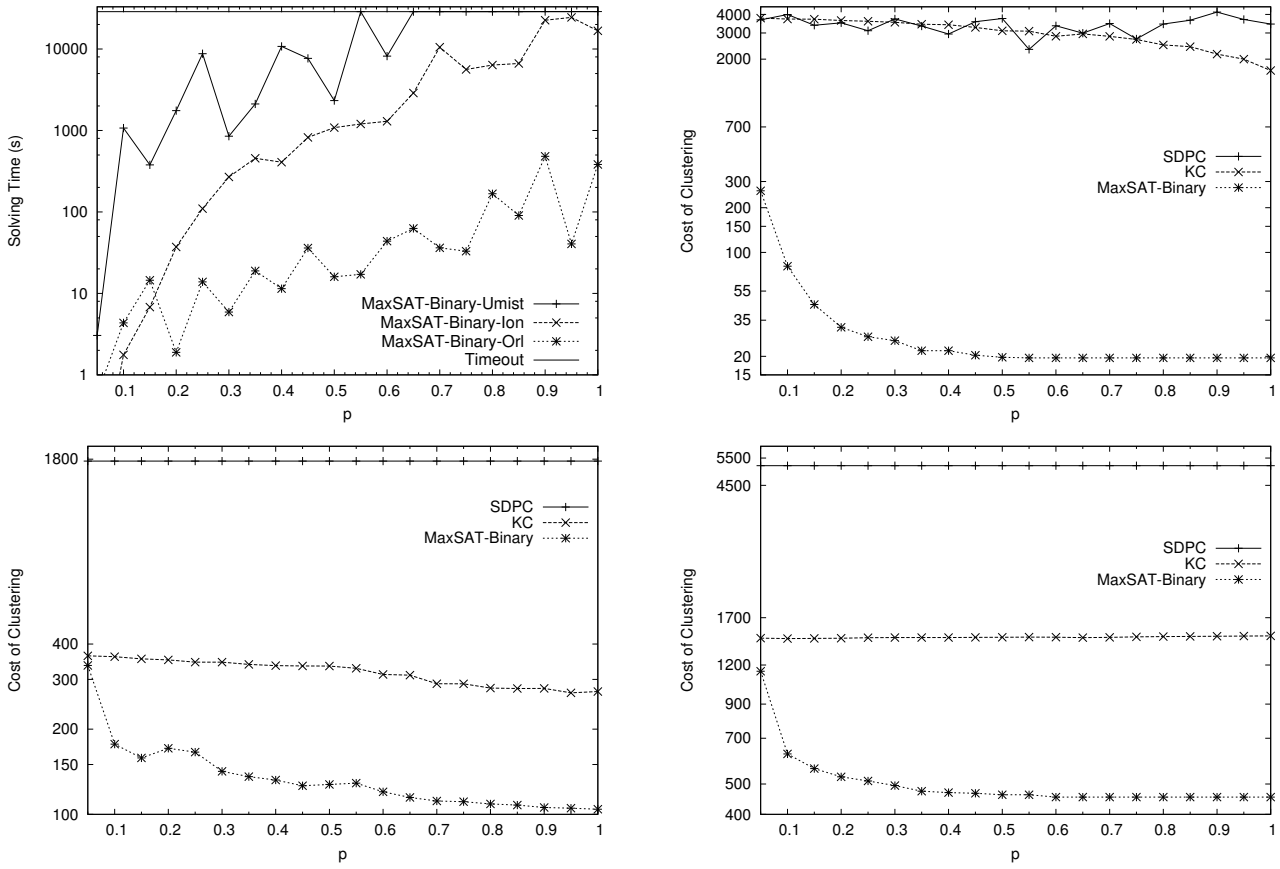

Figure 11: Evolution of running times (top left) and the cost of the clusterings obtained on sparse matrices. Top right: orl, bottom left: ionosphere, bottom right: umist. For the unsolvable MaxSAT instances we report the cost of the highest value of p still solvable.

two algorithms specialized for clustering protein sequences. Perhaps the most significant observation here is that, whenever the dataset was not solvable within the timeout, the clusterings obtained by MaxSAT on the highest value of p still solvable had in general a lower cost than any of the approximative algorithms at $p = 1.0$. This suggests that one can prune away a significant number of the non-zero entries in a matrix, hence speeding up MaxSAT solving, and still obtain clusterings of lower cost than those obtained with the approximative algorithms. We hypothesize that a more sophisticated method of pruning, perhaps taking into account the structure of the input matrix, could further improve the results. Comparing KwickCluster with SDPC, we observe that semi-definite programming performs slightly better on extremely sparse instances. However, when the density of the underlying graph increases, the performance of KwickCluster improves while the performance of SDPC remains fairly constant. One possible explanation for this could be that the relaxation of a quadratic program into a semi-definite program (see Section 10.1 for details) has a similar effect to the quality of the obtained clustering as pruning similarity information from the matrix.

9.3. Constrained Correlation Clustering

We now turn our attention to MaxSAT-based constrained correlation clustering.

9.3.1. Instance-Level Constraints

We first consider a situation in which an oracle, for example a domain expert, provides domain specific knowledge in the form of a set of must-link and cannot-link constraints the solution clusterings are expected to satisfy. By running several tests with an increasing number of constraints, we simulate a setting in which the current solution clustering is shown to the oracle, who is then allowed to add more constraints to the clustering algorithm in order to further restrict the set of acceptable clusterings. An iterative setting like this has previously been studied for example in [70, 71] and has been shown to greatly increase the clustering accuracy in other clustering problems [70, 71, 27].

We simulate this setting with the help of a (human created) "golden" clustering supplied with each of the datasets. Given the similarity matrix W based on a dataset, the golden clustering can be seen as a symmetric similarity matrix G_W of the same dimension where each element is either ∞ or $-\infty$. To simulate this iterative setting, we sampled an increasing number of pairs of indices $i < j$, and modify W by assigning $W(i,j) = G_W(i,j)$. Added $x\%$ user knowledge (UK) means that $x\%$ of available pairs of indices $i < j$ were sampled. This results in a setting in which at each iteration the MaxSAT algorithm has an increasing amount of information on the golden clustering. We note that, to the best of our knowledge, the considered approximative algorithms cannot handle such a constrained correlation clustering setting directly. Even though additional constraints could be included into the semi-definite program solved with SDPC, there are no guarantees that the clustering obtained after the rounding procedure within SDPC respects the added constraints (see Section 10.1 for more details). This is why all the values reported for those are for 0% added UK.

In these tests the performance of our encoding is evaluated using the well-known *rand index* [72] designed for measuring the similarity of two clusterings.

Definition 1. *Given a dataset $V = \{v_1 \dots v_N\}$, a clustering cl of V, and an example clustering g, let*

$$TP = |\{(v_i, v_j) \mid cl(v_i) = cl(v_j) \wedge g(v_i) = g(v_j)\}|$$

denote the number of pairs of points $i < j$ that are co-clustered in both cl and g (true positives). Let

$$TN = |\{(v_i, v_j) \mid cl(v_i) \neq cl(v_j) \wedge g(v_i) \neq g(v_j)\}|$$

denote the number of pairs of points $i < j$ that are assigned to different clusters in both cl and g (true negatives). The rand index of cl and g is then

$$R(cl, g) = \frac{TP + TN}{\binom{N}{2}} = \frac{2 \cdot (TP + TN)}{N \cdot (N-1)}.$$

Note that the denominator is the total number of unordered pairs of points over N data points.

As discussed in Section 2.3, ML and CL constraints are a non-trivial addition to the correlation clustering problem. Local search style clustering algorithms tend to suffer from over-constrainment in the sense that adding too many constraints can prevent such algorithms from converging. In contrast, the running time of the MaxSAT solver decreases with added constraints, see Figure 12[5]. As a consequence, using UK several additional datasets could be fully clustered with MaxSAT.

[5]Note that, especially when using a MaxSAT solver which searches bottom-up in the cost function (such as MaxHS), adding more constraints could also have a negative effect on ther running times of the solver.

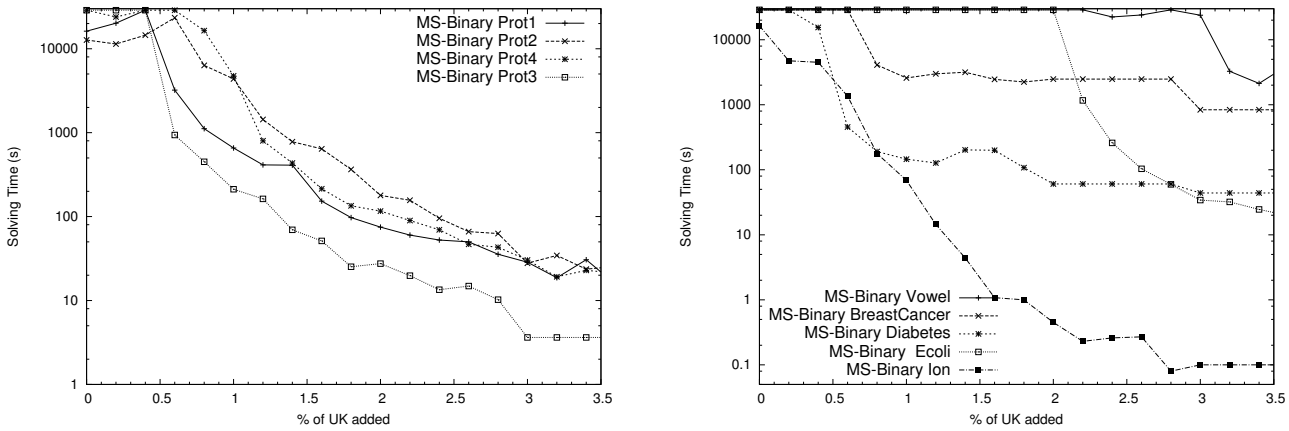

Figure 12: Evolution of running times of the MaxSAT solver with increasing amount of user knowledge added to the matrices.

Finally, we demonstrate that added UK constraints steer the clusterings produced by our MaxSAT encoding effectively towards the golden clustering. Figure 13 shows how the rand index increases as ML and CL constraints are added to the original similarity matrix. The number of extra constraints required for our algorithm to achieve rand indexes over 0.95 is for most datasets fairly small. The results suggest that extra constraints are highly beneficial and user knowledge should be taken advantage of whenever available.

9.3.2. Cluster-Level Constraints

To illustrate that the MaxSAT-based approach also enables obtaining optimal solutions which are guaranteed to satisfy cluster-level constraints, we consider a *Cluster Dissimilarity* constraint CL-DIS(k), closely related to constraints previously studied in distance-based clustering. Informally, a clustering cl satisfies the CL-DIS(k) constraint if no pair of points that are "more similar" than the threshold k are assigned to different clusters. More precisely, we require that $W(i,j) < k$ whenever $cl(v_i) \neq cl(v_j)$. This constraint is similar to the δ constraint in [37], where it was enforced by observing that the constraint can be decomposed into a set of ML constraints: whenever $W(i,j) > k$, we add a ML constraint over v_i and v_j.

Figure 14 demonstrates the running time of MaxHS on the four protein datasets (without any pruning) and an added CL-DIS(k) constraint for different threshold values k. Recall that all values in the benchmark similarity matrices were normalized to be between -0.5 and 0.5, which explains why we experimented with the threshold values $0.02, 0.04, \ldots, 0.5$. Note that $k = 0.5$ means that we are solving the original instance. All in all, the results show that the running time of MaxHS decreased drastically already for values only slightly below 0.5, all such instances being solvable in under a second.

9.4. MaxSAT-Specific Experiments

For the rest of this section, we will focus more MaxSAT-specific questions. We will consider the effects of MaxSAT-level preprocessing and symmetry breaking, as well as the performance of different state-of-the-art MaxSAT solvers, under the best-performing binary encoding. For these experiments we used the same set of benchmarks as in Section 9.2.2. We begin by considering preprocessing.

9.4.1. Effects of MaxSAT Preprocessing

Preprocessing is today an essential part of SAT solving. However, to date there has been few studies on MaxSAT preprocessing [73, 74]. As such, MaxSAT preprocessing is a relatively recent area of research,

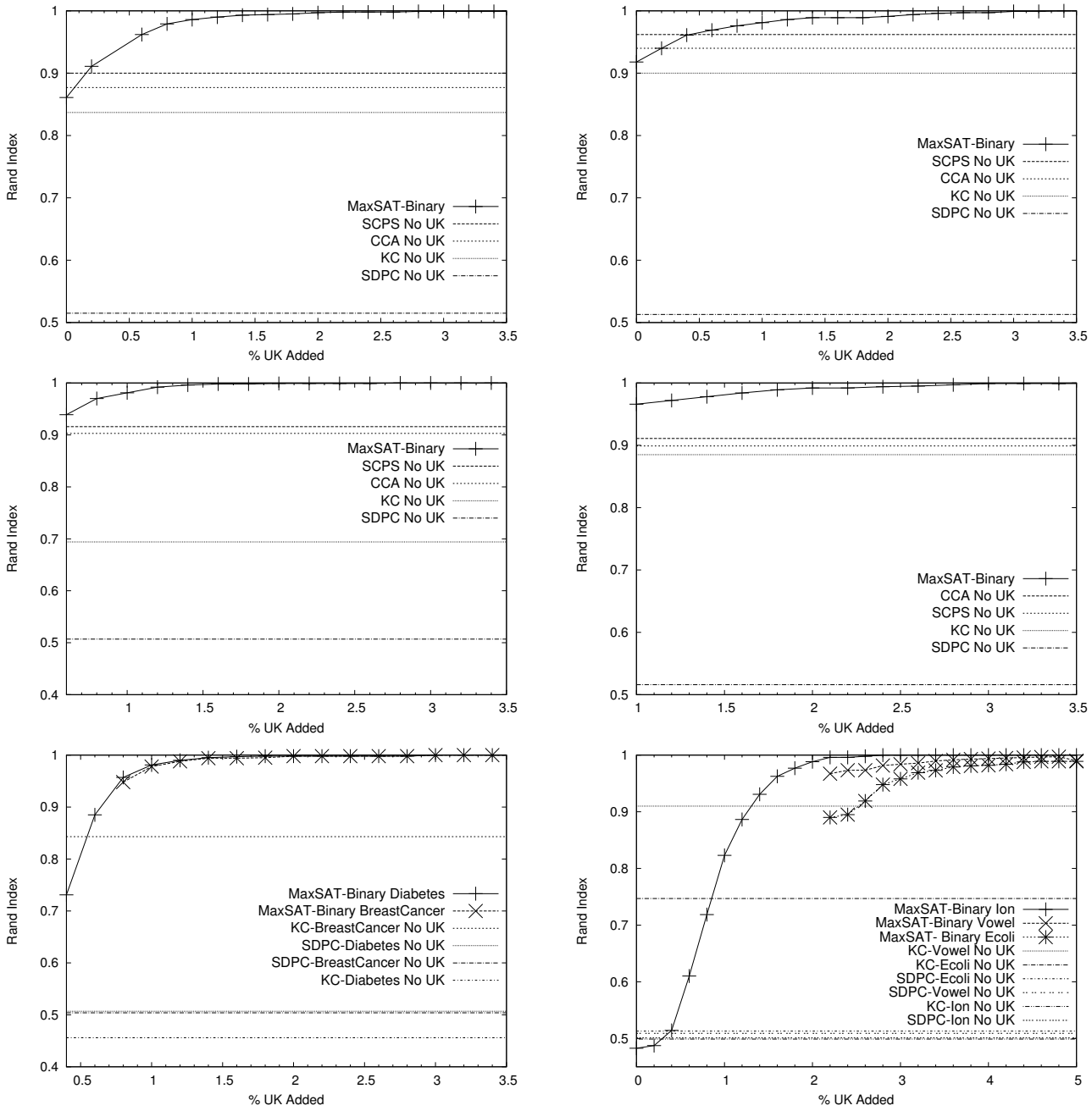

Figure 13: Evolution of the Rand index with increasing amount of user knowledge added to the matrices. The datasets from the top, left to right are: Prot1, Prot2, Prot3, Pro4, Breastcancer/Diabetes, Ecoli/Ionosphere/Vowel.

possibly due to the fact that not all popular SAT preprocessing techniques can be directly applied in the context of MaxSAT [73]. However, one recently proposed way of using SAT preprocessing on MaxSAT instances is through the so-called *labeled-CNF* framework [75, 73] which we also apply here.

We preprocess a given MaxSAT instance $F = (F_h, F_s, c)$ with N soft clauses in the following way.

1. Form the CNF formula $F^{SAT} = F_h \cup F_r$, where $F_r = \{(w_i \vee \neg r_i) \mid w_i \in F_s, i = 1..N\}$, with each

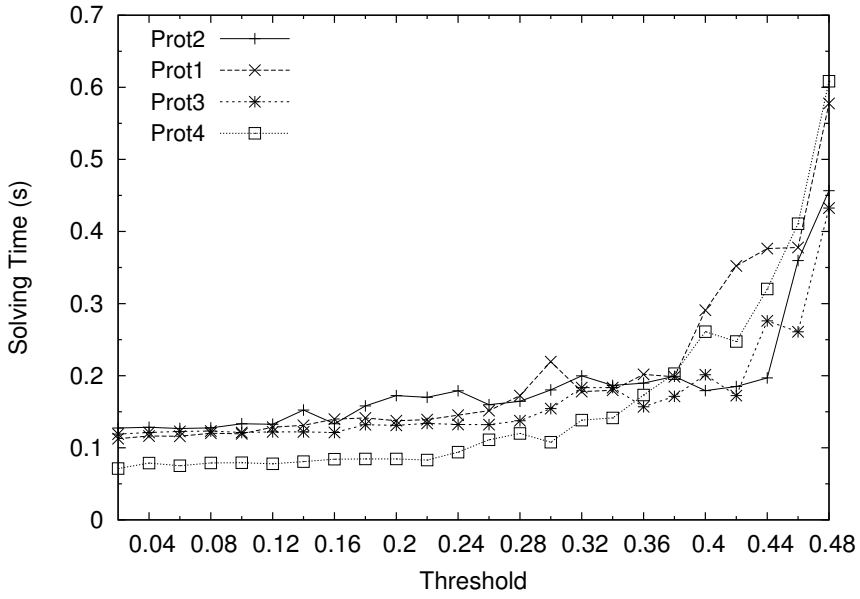

Figure 14: Running time of the MaxSAT solver with CL-DIS(k) constraint for different values of k.

of the variables r_i not appearing anywhere in F^{SAT} except the clause $(w_i \vee \neg r_i)$, thus obtaining the so-called *labeled-CNF* [75].

2. Apply the Coprocessor 2.0 [76] SAT preprocessor on F^{SAT}, obtaining the CNF formula F'^{SAT}. Coprocessor implements a large range of modern SAT preprocessing techniques, including unit propagation, variable elimination [77], clause elimination [78, 79], binary clause reasoning [80], etc. We used the white-listing option of Coprocessor to disable the preprocessor from removing any occurrences of any of the r_i variables. This is critical for ensuring correctness on the MaxSAT-level [75].
3. Finally, we constructed the MaxSAT instance $F^{PRE} = (F_h^{PRE}, F_s^{PRE}, c^{PRE})$ where $F_h^{PRE} = F'^{SAT}$, $F_s^{PRE} = \{(r_i) \mid i = 1..N\}$ and $c^{PRE}(r_i) = c(w_i)$. Now $\text{OPT}(F) = \text{OPT}(F^{PRE})$ and any solution τ to F^{PRE} can be extended into a solution for F of equal cost in polynomial (negligible) time, similarly as when applying SAT preprocessing on the SAT-level [81].

The preprocessing time of MaxSAT instances resulting from the binary encoding was negligible, less than 10 seconds for each instance.

Figure 15 demonstrates the difference in running time with and without preprocessing on instances consisting of 50% and 100% of the non-zero values of the datasets. On a majority of 94 out of 140 instances, preprocessing lowered the running time of the MaxSAT solver enough to compensate for the extra time spent on preprocessing. Furthermore, all instances that were solved without preprocessing were also solved after applying preprocessing. However, we did also observe instances on which preprocessing had a negative impact on the running time, exemplified in Figure 15 by the Prot1 dataset.

9.4.2. Effect of Symmetry Breaking

The solution space of correlation clustering is highly symmetric: given any clustering $cl\colon V \to \{1 \dots |V|\}$ of a set of data points, the cluster indices can be arbitrarily permuted without affecting the actual partitioning of the data points and hence its cost. This leads to the question of whether MaxSAT solving could be sped-up by breaking some of these symmetries on the MaxSAT-level.

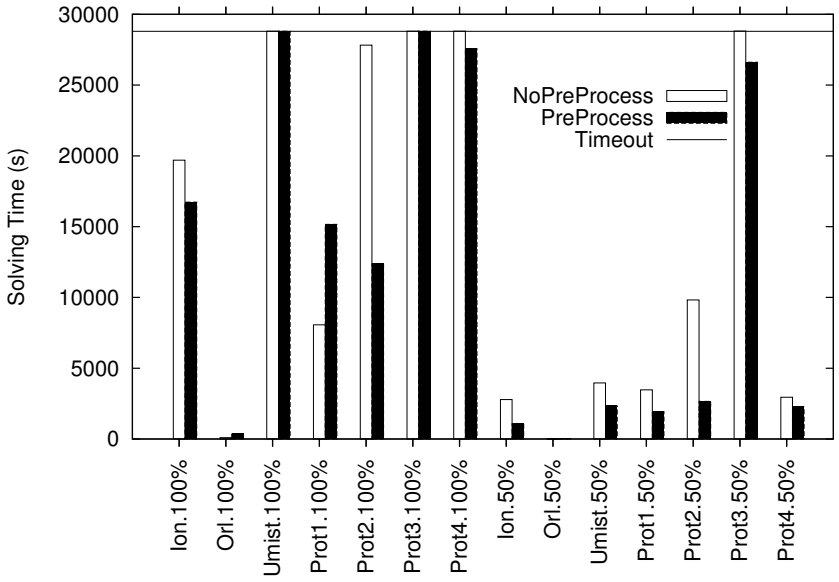

Figure 15: Effect of preprocessing MaxSAT instances. The percentage in the instance name is the expected number of nonzero entries (compared to the full instance). Solving times reported for MaxHS and the binary encoding.

Full symmetry breaking seems unlikely to be beneficial, due to the fact that a very large number of clauses—going far beyond the size of the binary encoding—would be needed. More formally, define a relation $\equiv$ over the set of possible clusterings of V by $cl \equiv cl'$ if $cl = \sigma \circ cl'$ for some permutation σ. It is simple to see that the relation $\equiv$ is an equivalence relation. Full symmetry breaking corresponds to adding constraints that forbid all but one member out of each equivalence class of $\equiv$. A straightforward approach to achieve this would require clauses that identify some representative point (e.g., the smallest) assigned to each cluster in a clustering and enforce an ordering over these points. This could be done by encoding constraints stating *If v_i is not co-clustered with v_j for any $j < i$, then $cl(v_i) > cl(v_k)$ for all $k < i$.* Such a constraint can be encoded into MaxSAT by techniques similar to the ones presented in Section 8. However, as there can be up to N clusters, this would introduce $\mathcal{O}(N^3)$ new clauses, which would evidently deteriorate the performance of a MaxSAT solver. As a related observation, note that the cubic transitive MaxSAT encoding breaks all symmetries, but does not perform as well as the binary encoding (without symmetry breaking).

Even though full symmetry breaking seems infeasible, we might still be able to boost solver performance by applying partial symmetry breaking to our encoding. Partial symmetry breaking refers to including clauses that remove (only) some symmetric solutions. As a simple example and the baseline in our experiments, we have the already mentioned the very simple *first point into the first cluster* constraint that can be enforced with $\log_2 N$ unit clauses of the form $(\neg b_1^i)$, $1 \leq i \leq \log_2 N$.

A more involved symmetry breaking constraint we consider is the CLUSTERSLESSTHAN(i, N) presented in Section 8.5. Without enforcing a limit on the number of clusters, the constraints CLUSTERSLESSTHAN(i, N) are not required in order for the encoding to be correct. However, including them does prune away a significant number of symmetric solutions. Furthermore, the constraints are relatively compact, in total only $\mathcal{O}(N \cdot \log_2(N))$ new clauses need to be added. In our experiments we call this type of symmetry breaking REMOVESLACK.

A further form of symmetry breaking deals with symmetries induced by the possibility of empty clusters. As an instance created by the binary encoding allows (at least) N different cluster indices for every point, the placement of empty clusters can potentially introduce symmetries into the solution space. Assume for example that some optimal clustering cl contains C clusters. Then cl is equivalent to at least $\binom{N}{N-C}$ other clusterings, depending on which of the N cluster indices are empty. We can remove some of these symmetries by forcing the empty clusters to occupy indices $C+1, \ldots N$, or, more generally, the largest (or equivalently, the smallest) available indices. We next describe the encoding of this constraint in terms of the binary encoding.

Assume that we are using a bits to represent the cluster indices in the binary encoding. We introduce new variables $E_1 \ldots E_{(2^a)}$, with the intended interpretation $\tau(E_j) = 1$ iff $c(v) \neq j$ for all $v \in V$, that is, cluster j is empty. Using these variables, the empty cluster indices are propagated with constraints of the form $E_i \rightarrow E_{i+1}$ which inductively require that the clusters of higher index than an empty cluster are also empty, and that all clusters of lower index than a non-empty cluster are also non-empty. To define the E_j variables for a given cluster j and data point v_i, let b_i^{t*} denote the literal corresponding to the value of the t:th bit in j, i.e., b_i^t if the t:th bit in j is 1, and $\neg b_i^t$ otherwise. Now the E_j variable can be defined as

$$E_j \leftrightarrow \bigvee_{i=1}^{N} \left(b_i^{1*} \wedge \ldots \wedge b_i^{a*} \right),$$

which can be compactly represented as clauses by introducing N new auxiliary variables $C_1^j, \ldots, C_N^j$ and defining them as $C_i^j \leftrightarrow \left(b_i^{1*} \wedge \ldots \wedge b_i^{a*} \right)$. Similar constraints are introduced for all of the E_i variables. We call this type of symmetry breaking PROPAGATEEMPTY. Compared to REMOVESLACK, the PROPAGATEEMPTY constraint breaks more symmetries. In particular, symmetries broken by PROPAGATEEMPTY include all of the symmetries broken by REMOVESLACK. However, PROPAGATEEMPTY is more costly in terms of encoding size. In total, the constraints introduce $\mathcal{O}(N^2 \cdot \log_2 N)$ new clauses. Recall that the total size of the binary encoding is $\mathcal{O}(E \cdot \log_2 N)$ where E is at most of order N^2, which means that enforcing the PROPAGATEEMPTY constraint might significantly increase the number of clauses on sparse instances.

Figure 16 demonstrates the effect of the REMOVESLACK constraint compared to the baseline. The PROPAGATEEMPTY constraint is missing from the figure due to the fact that, when enforcing it, no instances could be solved within the timeout. We hypothesize that the reason for this is the significant number of clauses required for encoding it. As a concrete example, the preprocessed Prot1 dataset with 100% of the non-zero values present contains 323 301 clauses without PROPAGATEEMPTY and 6 427 796 clauses when enforcing PROPAGATEEMPTY. However, as can be seen in Figure 16, the REMOVESLACK constraint actually does improve solver performance on most instances.[6]

9.4.3. A Comparison of MaxSAT solvers

In the main experiments reported in this work, we used the MaxHS MaxSAT solver, which has shown very good performance especially in the "crafted" category of the recent MaxSAT Evaluations[7]. Here we report on the performance of other state-of-the-art MaxSAT solvers, using the following solvers.

- Eva500 solver [82], obtained from `http://www.maxsat.udl.cat/14/solvers/`.

[6] We remind the reader that, apart from the *first point into the first cluster* constraint, symmetry breaking was not applied in the other experiments reported on in this article.

[7] `http://www.maxsat.udl.cat/14/results/index.html#wpms-crafted`

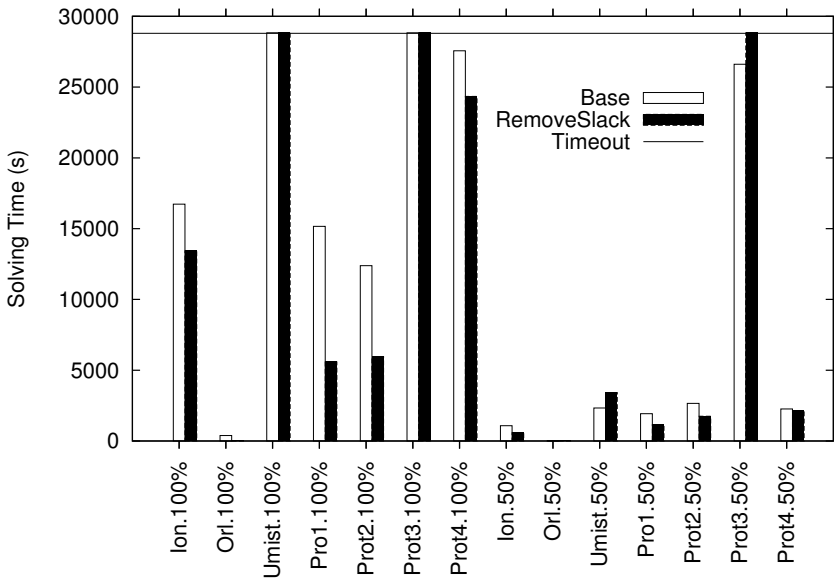

Figure 16: Effect of different symmetry breaking techniques on the solving time of MaxSAT. The percentage in the instance name is the expected number of nonzero entries (compared to the full instance). Solving times reported for MaxHS and the binary encoding.

- MsUnCore bcd2 version [49, 48] obtained from `http://www.csi.ucd.ie/staff/jpms/soft/soft.php`.
- OpenWBO [83, 84] version 1.1.1. obtained from `http://sat.inesc-id.pt/open-wbo/`.
- The ILP2013 solver [56]. We implemented the conversion to an integer program ourselves and used CPLEX to solve the resulting instances.

The first three in the list are core guided solvers (recall Section 5.2). Eva500 uses the identified cores and a restricted form of MaxSAT resolution [85] to relax the MaxSAT instance in a controlled way. MsUnCore performs binary search over the cost function and also maintains a set of already identified disjoint cores and relaxes each core separately whenever a new one is found. OpenWBO uses an incremental approach that allows it to pertain the state of the internal SAT solver more efficiently between the iterations. ILP2013 encodes the whole MaxSAT instances as an integer linear program and then calls an ILP solver. Since MsUnCore, OpenWBO and Eva500 accept only integral weights, for running these solvers we multiplied all similarity values by 10^{13} (the highest possible multiplier with which the trivial cost upper bound required as input by the solvers still stays within the 2^{63} range) and rounded afterwards to integers. Table 2 gives a performance comparison of the solvers. MaxHS scales significantly better than the other solvers. All in all, MaxHS solved 121 instances within the timeout while the second-best performing Eva500 solved 65. The other solvers in the comparison timed out on most instances. Note that, apart from the *first point into the first cluster* constraint, symmetry breaking was not applied in this experiment.

10. Related work

We continue with a survey on related work.

Table 2: Comparison of MaxSAT solvers

Solver	Number of solved instances	Number of timeouts
MaxHs	121	19
Eva500	65	75
ILP2013	7	133
MsUnCore	7	133
OpenWBO	0	140

10.1. Correlation Clustering

While the notion of producing good clusterings under inconsistent advice first appeared in [11], the formal definition of correlation clustering was proposed in [4] and shown to be NP-hard on complete graphs with each edge labeled with $+$ or $-$; or, in terms of the general problem definition considered in this work, on symmetric similarity matrices W where $W(i,j) = \{-1, 1\}$ for all i and j. NP-hardness motivated early work on approximative algorithms for the problem. Approximation algorithms for correlation clustering typically address one of three different objectives for the problem: *minimizing disagreements*, *maximizing agreements*, or *maximizing correlation*. Given a similarity matrix W over a set of data points $V = \{v_1, \ldots, v_N\}$, minimizing disagreements refers to minimizing the number of point pairs v_i, v_j whose cluster assignment does not agree with their similarity value $W(i,j)$, or more precisely, to finding a clustering cl minimizing

$$\sum_{\substack{i<j \\ \infty > W(i,j) > 0}} \mathcal{I}[cl(v_i) \neq cl(v_j)] W(i,j) + \sum_{\substack{i<j \\ -\infty < W(i,j) < 0}} \mathcal{I}[cl(v_i) = cl(v_j)] |W(i,j)|. \tag{7}$$

Maximizing agreements refers to maximizing the number of pairs of points v_i, v_j whose cluster assignment agrees with their similarity value $W(i,j)$, or more precisely, to finding a clustering cl maximizing

$$\sum_{\substack{i<j \\ \infty > W(i,j) > 0}} \mathcal{I}[cl(v_i) = cl(v_j)] W(i,j) + \sum_{\substack{i<j \\ -\infty < W(i,j) < 0}} \mathcal{I}[cl(v_i) \neq cl(v_j)] |W(i,j)|. \tag{8}$$

Maximizing correlation refers to maximizing the difference between agreements and disagreements, i.e., using the objective function obtained by substracting Equation 7 from Equation 8.

A polynomial-time approximation scheme for maximizing agreements on complete symmetric matrices with $\{-1, 1\}$ similarity values was presented in [4]. No such scheme is likely to exist for minimizing disagreements as the problem is APX-hard [86]. On general matrices, maximizing agreements is also APX-hard [6], except for when the ratio between the smallest and largest absolute value in the matrix is bounded by a constant [33]. To the best of our knowledge, the SDPC algorithm proposed in [34] and detailed below is the best known approximation algorithm for maximizing correlation.

The two approximative algorithms for correlation clustering we experimented with in this work are based on different techniques. KwickCluster [5] is a greedy combinatorial approximation algorithm that at each iteration picks one of the still unassigned nodes to be the *pivot node*, and forms a new cluster containing the pivot node and all still unassigned nodes that are similar to the pivot node (recall that nodes v_i and v_j are similar if $W(i,j) > 0$ in the similarity matrix under consideration). The algorithm terminates when all points have been assigned to some cluster. The same algorithm also appears in [67] under the name *PivotAlg*. As shown in [5, 67], KwickCluster is a factor-3 approximation algorithm for minimizing

disagreements under the assumption that $W(i,j) \in \{-1,1\}$ for all i and j, and a factor-5 approximation algorithm under the assumption that $-1 \leq W(i,j) \leq 1$ for all i, j.[8]

SDPC [34] is based on rounding solutions to a semidefinite program that itself is a relaxation of the quadratic programming formulation of correlation clustering restricted to two clusters. Restricting the correlation clustering problem search space to clusterings only containing two clusters, the quadratic program in Equation 4 (recall Section 4) can be formulated equivalently as

$$\begin{aligned} &\text{MAXIMIZE} && \sum_{i=1}^{N}\sum_{j=i+1}^{N}(W(i,j)z_iz_j) \\ &\text{where} && z_i \in \{-1,1\} \text{ for all } i, \end{aligned} \tag{9}$$

where N is the number of data points, and the value in the solution of the variable z_i indicates whether point v_i is assigned to cluster 1 or -1. This quadratic program can be relaxed into the semidefinite program

$$\begin{aligned} &\text{MAXIMIZE} && \sum_{i=1}^{N}\sum_{j=i+1}^{N}(W(i,j)u_i \cdot u_j) \\ &\text{where} && |u_i| = 1 \text{ for all } i \\ & && u_i \in \mathbb{R}^N \text{ for all } i, \end{aligned} \tag{10}$$

where each z_i binary variable from Equation 9 is represented by a vector u_i on the unit sphere in $\mathbb{R}^N$. The relaxation of the quadratic program (Equation 9) into the semidefinite program (Equation 10) is standard, being similar to the SDP relaxation for MaxCut presented in [87].

In [34] an algorithm that rounds a solution obtained to Equation 10 into a well-defined clustering is presented and shown to achieve a $\Omega\left(\log(N)^{-1}\right)$ approximation factor for maximizing correlation. The algorithm compares clusterings obtained from rounding the semi-definite program with the (unique) cost of the trivial clustering in which all data points are assigned to different clusters, and returns the better solution out of these two.

In [33] the authors develop a PTAS for maximizing agreements on general matrices under the assumption that the ratios between weights in the input matrices are bounded by a constant, which implies that the matrices cannot contain 0-entries, i.e. the available similarity information has to be complete. The PTAS is developed by using the smooth polynomial programming technique on the QIP formulation, which results in strong approximation bounds for the maximization problem on matrices satisfying the assumption.

In [88] a more restricted version of the approximative correlation clustering algorithm of [4] is presented, with experiments on identifying and resolving noun co-reference in texts. In [89] a greedy randomized adaptive search procedure (GRASP) based approximative algorithm is presented, with the motivation that the obtained solutions can be used as a criterion for determining the balance in social networks. Also, in [10] correlation clustering is used for crosslingual link detection between google news groups. The authors build on the results of [86] and present an algorithm based on relaxing the ILP formulation of correlation clustering into a linear program and then using region growing techniques for rounding of the solution of the linear program. As such, their algorithm is also approximative in nature and, in contrast to our approach, cannot provide optimality guarantees. The authors also provide some results on exactly solving the ILP with added must-link and cannot-link constraints[9]. As noted in [10] and supported by our experiments, the

[8] Recall that a factor α approximation algorithm on a minimization (maximization) problem is guaranteed to return a solution of cost lower (higher) than α times the cost of the optimal solution.

[9] Unfortunately, the authors were unable to provide an implementation of their algorithm.

ILP-based approach to correlation clustering suffers from the fact that the number of constraints is cubic in the number of data points, leading to memory problems in practice. The authors of [10] approach the issue by splitting the LP into smaller chunks and processing the chunks separately. In contrast, our experimental results suggest that using MaxSAT for solving correlation clustering is more memory-efficient without extra tuning. There has also been some work done on a variant of correlation clustering in which the the search is further restricted to $cl\colon V \to \{1, \dots K\}$ for some $K < N$ [8]. As explained in Sections 7 and 8, both the binary and the unary encoding can be used in this setting as well.

A few generalizations of correlation clustering have been proposed. In [12] the authors experiment with correlation clustering allowing overlapping clusters. The proposed solution to overlapping correlation clustering is a local search algorithm that locally adjusts the solution clustering as long as the cost function decreases. Out of the MaxSAT encodings presented in this work, the unary encoding extends naturally to overlapping clustering by changing the cardinality constraint $\textsc{ExactlyOne}(i)$ of each point to a more general $\sum_{k=1}^{K} y_i^k \leq p$, where p is the maximum number of clusters a single point can be assigned to. The resulting encoding can be shown to produce globally optimal solutions to the overlapping clustering problem. Another proposed generalization to correlation clustering is chromatic correlation clustering [13]. In the basic form of correlation clustering, there are two possible relationships between pairs of data points. A pair of data points can either be similar, dissimilar (or neither). Chromatic correlation clustering generalizes this by allowing more than two different categories of relationships. This can be visualized as an undirected graph in which each edge is colored. The task is then to find a clustering that maximizes color purity of edges within clusters. Our MaxSAT encodings can be extended to cover Chromatic Correlation Clustering by introducing variables which represent the principal color of each cluster.

10.2. Constrained Clustering

As exemplified in Section 9.3, the MaxSAT-based approach allows for obtaining solution which are guaranteed to satisfy additional hard constraints on the clusterings of interest. This includes both instance-level and, as exemplified in Section 9.3.2, even some distance-based cluster-level constraints which have been previously studied in the context of constrained clustering [90, 23]. The idea of adding constraints to the clustering problem was first introduced in [27, 28]. The introduction of constraints to the clustering problem allows the addition of domain knowledge to the problem and has been shown to increase clustering accuracy [27]. Much of the early work on constrained clustering concentrated on modifying existing heuristics and clustering algorithms in order to allow the addition of constraints. Examples include k-means and COB-WEB [27, 28], EM [91], hierarchical [92] and spectral clustering [93]. The problem of deciding if there exists a clustering satisfying a given set of must and cannot-link constraints was shown to be NP hard in [37]. In fact, many of the modified approximative algorithms are not even guaranteed to return a clustering satisfying all user constraints. The algorithms also have difficulties in handling too many extra constraints, they are easily over-constrained, preventing the algorithms from converging at all [37].

An alternative approach to constrained clustering is to cast the task as a constraint optimization problem, allowing for a very natural incorporation of added constraints. This is the approach which we employ in this work. A similar idea was proposed in [23] in a different clustering setting. The authors show that a satisfiability-based framework is well-suited for constrained clustering in the sense that constraints are easily added, the solutions returned are guaranteed to be globally optimal and satisfy all given constraints, and the search algorithm is not as easily over-constrained. Our approach to solving constrained *correlation* clustering is similar, but more generic as we do not restrict ourselves to only allowing two distinct clusters, which is a polynomial time special case of the general clustering problem. Furthermore, our encoding are on the MaxSAT-level (optimization instead of pure SAT), and employ a MaxSAT solver instead of a pure SAT solver.

Constrained clustering has also been approached via integer programming. In [94, 95] a variety of different possible constraints and optimization functions are considered. However, in practice their approach might be difficult to use as it requires a predetermined set of candidate clusters from which the algorithm searches for the best subset. In [90, 96] the authors use an integer programming and column generation based approach in order to exactly solve the minimum sum of squares clustering problem. Constrained clustering has also been approached, again in a different clustering setting, by constraint programming (CP) [97, 24][10]. In [97] different optimization criteria for clustering are studied and solved by casting the clustering problems as constraint programming problems. A SAT-based framework of constrained clustering has also been proposed for example in [31], but optimization criteria are not applied in their experiments. In [24] a general framework for *K-pattern set mining under constraints* is introduced. The authors present a general framework and explore the strengths and limitations of using constraint programming. Yet another recent example of using declarative programming in the context of clustering is [26], in which an ILP formulation of hierarchical clustering, with an explicit objective function that is globally optimized, was presented; that approach would similarly allow for satisfying hard constraints over the solution space.

11. Conclusions

This work contributes to the research direction of harnessing constraint solving for developing novel types of generic data analysis techniques. The focus of our study is the applicability of state-of-the-art Boolean optimization procedures to cost-optimal correlation clustering in both unconstrained and constrained settings. To this end, we presented a novel MaxSAT-based framework for solving correlation clustering. Our approach is based on casting the clustering problem declaratively as weighted partial maximum satisfiability, and using a generic MaxSAT solver for finding cost-optimal clusterings. We studied three different encodings of correlation clustering as MaxSAT, and reported on an experimental evaluation, comparing both the time required to solve the resulting MaxSAT instances, and the quality of the clusterings obtained. We compared the MaxSAT-based approach to previously proposed both exact (integer linear and quadratic programming based) and approximative (specialized local search and approximation algorithms and semi-definite programming) approaches on real-world datasets. The MaxSAT approach scales better than the exact integer linear and quadratic programming approaches, and provides clusterings of significantly lower cost than the approximative algorithms, especially when the input data is sparse. Due to the intrinsic computational hardness of correlation clustering, we acknowledge that a potential issue with our approach is scalability, especially scaling the MaxSAT-based approach to very large datasets (with tens of thousands of data points). Nevertheless, the approach can provide cost-optimal clusterings on real-world datasets with close to a thousand points. The approach is also flexible when it comes to satisfying user-specified constraints, i.e., in constrained correlation clustering. The running times of the approach can notably decrease in a constrained setting, allowing for solving larger datasets faster compared to the non-constrained setting. This is in stark contrast with local search algorithms which easily suffer from over-constraining in constrained settings. It is conceivable that our approach can be improved also by foreseeable improvements to generic MaxSAT solvers and by developing domain-specific parallelization schemes, as well as by specialized constraint optimization techniques and heuristics for the problem domain. Yet another interesting direction would be to study the applicability of Large Neighborhood Search which combine local search strategies for fixing a subspace of the search space to which to apply exact search techniques.

[10] The term constraint programming refers here to the declarative language as opposed to a general term of the paradigm.

References

[1] J. Berg, M. Järvisalo, Optimal correlation clustering via MaxSAT, in: Proc. 2013 IEEE ICDM Workshops, IEEE Press, 2013, pp. 750–757.

[2] D. H. Fisher, Knowledge acquisition via incremental conceptual clustering, Mach. Learn. 2 (2) (1987) 139–172.

[3] A. K. Jain, M. N. Murty, P. J. Flynn, Data clustering: A review, ACM Comput. Surv. 31 (3) (1999) 264–323.

[4] N. Bansal, A. Blum, S. Chawla, Correlation clustering, Machine Learning 56 (1-3) (2004) 89–113.

[5] N. Ailon, M. Charikar, A. Newman, Aggregating inconsistent information: Ranking and clustering, J. ACM 55 (5).

[6] M. Charikar, V. Guruswami, A. Wirth, Clustering with qualitative information, J. Comput. Syst. Sci. 71 (3) (2005) 360–383.

[7] R. Shamir, R. Sharan, D. Tsur, Cluster graph modification problems, Discr. Appl. Math. 144 (1-2) (2004) 173–182.

[8] I. Giotis, V. Guruswami, Correlation clustering with a fixed number of clusters, Theory of Computing 2 (1) (2006) 249–266.

[9] E. D. Demaine, D. Emanuel, A. Fiat, N. Immorlica, Correlation clustering in general weighted graphs, Theor. Comput. Sci. 361 (2-3) (2006) 172–187.

[10] J. V. Gael, X. Zhu, Correlation clustering for crosslingual link detection, in: Proc. IJCAI, 2007, pp. 1744–1749.

[11] A. Ben-Dor, R. Shamir, Z. Yakhini, Clustering gene expression patterns, Journal of Computational Biology 6 (3/4) (1999) 281–297.

[12] F. Bonchi, A. Gionis, A. Ukkonen, Overlapping correlation clustering, in: Proc. ICDM, IEEE, 2011, pp. 51–60.

[13] F. Bonchi, A. Gionis, F. Gullo, A. Ukkonen, Chromatic correlation clustering, in: Proc. KDD, ACM, 2012, pp. 1321–1329.

[14] N. Cesa-Bianchi, C. Gentile, F. Vitale, G. Zappella, A correlation clustering approach to link classification in signed networks, in: Proc. COLT, Vol. 23 of JMLR Procedings, JMLR.org, 2012, pp. 34.1–34.20.

[15] P. Bonizzoni, G. D. Vedova, R. Dondi, T. Jiang, Correlation clustering and consensus clustering, in: Proc. ISAAC, Vol. 3827 of Lecture Notes in Computer Science, Springer, 2005, pp. 226–235.

[16] V. Filkov, S. Skiena, Integrating microarray data by consensus clustering, Int J Artif Intell T 13 (4) (2004) 863–880.

[17] V. Filkov, S. Skiena, Heterogeneous data integration with the consensus clustering formalism, in: Proc. DILS, Vol. 2994 of Lecture Notes in Computer Science, Springer, 2004, pp. 110–123.

[18] R. Giancarlo, F. Utro, Speeding up the consensus clustering methodology for microarray data analysis, Algorithms for Molecular Biology 6 (2011) 1.

[19] Z. Yu, H.-S. Wong, H.-Q. Wang, Graph-based consensus clustering for class discovery from gene expression data, Bioinformatics 23 (21) (2007) 2888–2896.

[20] T. Guns, S. Nijssen, L. D. Raedt, Itemset mining: A constraint programming perspective, Artif. Intell. 175 (12-13) (2011) 1951–1983.

[21] S. Nijssen, T. Guns, L. D. Raedt, Correlated itemset mining in ROC space: a constraint programming approach, in: Proc. KDD, ACM, 2009, pp. 647–656.

[22] L. D. Raedt, T. Guns, S. Nijssen, Constraint programming for data mining and machine learning, in: Proc. AAAI, AAAI Press, 2010.

[23] I. Davidson, S. S. Ravi, L. Shamis, A SAT-based framework for efficient constrained clustering, in: Proc. SDM, SIAM, 2010, pp. 94–105.

[24] T. Guns, S. Nijssen, L. D. Raedt, K-pattern set mining under constraints, IEEE Trans. Knowl. Data Eng. 25 (2) (2013) 402–418.

[25] B. Négrevergne, A. Dries, T. Guns, S. Nijssen, Dominance programming for itemset mining, in: Proc. ICDM, IEEE, 2013, pp. 557–566.

[26] S. Gilpin, S. Nijssen, I. N. Davidson, Formalizing hierarchical clustering as integer linear programming, in: Proc. AAAI, AAAI Press, 2013.

[27] K. Wagstaff, C. Cardie, Clustering with instance-level constraints, in: Proc. ICML, Morgan Kaufmann, 2000, pp. 1103–1110.

[28] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, Constrained K-means clustering with background knowledge, in: Proc. ICML, Morgan Kaufmann, 2001, pp. 577–584.

[29] I. Davidson, S. S. Ravi, Intractability and clustering with constraints, in: Proc. ICML, ACM, 2007, pp. 201–208.

[30] A. Biere, M. J. H. Heule, H. van Maaren, T. Walsh (Eds.), Handbook of Satisfiability, IOS Press, 2009.

[31] J.-P. Métivier, P. Boizumault, B. Crémilleux, M. Khiari, S. Loudni, Constrained clustering using SAT, in: Proc. IDA, Vol. 7619 of Lecture Notes in Computer Science, Springer, 2012, pp. 207–218.

[32] C. M. Li, F. Manyà, MaxSAT, hard and soft constraints, in: Handbook of Satisfiability, IOS Press, 2009, pp. 613–631.

[33] P. Bonizzoni, G. D. Vedova, R. Dondi, T. Jiang, On the approximation of correlation clustering and consensus clustering, J. Comput. Syst. Sci. 74 (5) (2008) 671–696.

[34] M. Charikar, A. Wirth, Maximizing quadratic programs: Extending grothendieck's inequality, in: Proc. FOCS, IEEE Computer Society, 2004, pp. 54–60.

[35] D. Klein, S. D. Kamvar, C. D. Manning, From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering, in: Proc. ICML, Morgan Kaufmann, 2002, pp. 307–314.

[36] I. Davidson, S. S. Ravi, Clustering with constraints: Feasibility issues and the k-means algorithm, in: Proc. SDM, SIAM, 2005, pp. 138–149.

[37] I. Davidson, S. S. Ravi, The complexity of non-hierarchical clustering with instance and cluster level constraints, Data Min. Knowl. Discov. 14 (1) (2007) 25–61.

[38] M. Křivánek, J. Morávek, NP-hard problems in hierarchical-tree clustering, Acta Informatica 23 (3) (1986) 311–323.

[39] Y. Chen, S. Safarpour, J. Marques-Silva, A. G. Veneris, Automated design debugging with maximum satisfiability, IEEE Trans. on CAD of Integrated Circuits and Systems 29 (11) (2010) 1804–1817.

[40] C. S. Zhu, G. Weissenbacher, S. Malik, Post-silicon fault localisation using maximum satisfiability and backbones, in: Proc. FMCAD, FMCAD Inc., 2011, pp. 63–66.

[41] M. Jose, R. Majumdar, Cause clue clauses: error localization using maximum satisfiability, in: Proc. PLDI, ACM, 2011, pp. 437–446.

[42] J. Guerra, I. Lynce, Reasoning over biological networks using maximum satisfiability, in: Proc. CP, Vol. 7514 of Lecture Notes in Computer Science, Springer, 2012, pp. 941–956.

[43] J. Berg, M. Järvisalo, B. Malone, Learning optimal bounded treewidth bayesian networks via maximum satisfiability, in: Proc. AISTATS, Vol. 33, JMLR, 2014, pp. 86–95.

[44] M. Järvisalo, D. Le Berre, O. Roussel, L. Simon, The international SAT solver competitions, AI Magazine 33 (1) (2012) 89–92.

[45] C. M. Li, F. Manyà, N. O. Mohamedou, J. Planes, Exploiting cycle structures in Max-SAT, in: Proc. SAT, Vol. 5584 of Lecture Notes in Computer Science, Springer, 2009, pp. 467–480.

[46] M. Koshimura, T. Zhang, H. Fujita, R. Hasegawa, QMaxSAT: A partial Max-SAT solver, JSAT 8 (1/2) (2012) 95–100.

[47] J. Marques-Silva, J. Planes, Algorithms for maximum satisfiability using unsatisfiable cores, in: Proc. DATE, IEEE, 2008, pp. 408–413.

[48] A. Morgado, F. Heras, J. Marques-Silva, Improvements to core-guided binary search for MaxSAT, in: Proc. SAT, Vol. 7317 of Lecture Notes in Computer Science, Springer, 2012, pp. 284–297.

[49] F. Heras, A. Morgado, J. Marques-Silva, Core-guided binary search algorithms for maximum satisfiability, in: Proc. AAAI, AAAI Press, 2011.

[50] C. Ansótegui, M. L. Bonet, J. Levy, SAT-based MaxSAT algorithms, Artif. Intell. 196 (2013) 77–105.

[51] A. Morgado, F. Heras, M. H. Liffiton, J. Planes, J. Marques-Silva, Iterative and core-guided MaxSAT solving: A survey and assessment, Constraints 18 (4) (2013) 478–534.

[52] Z. Fu, S. Malik, On solving the partial MaxSAT problem, in: Proc. SAT, Vol. 4121 of Lecture Notes in Computer Science, Springer, 2006, pp. 252–265.

[53] V. M. Manquinho, J. P. M. Silva, J. Planes, Algorithms for weighted boolean optimization, in: Proc. SAT, Vol. 5584 of Lecture Notes in Computer Science, Springer, 2009, pp. 495–508.

[54] A. Morgado, C. Dodaro, J. Marques-Silva, Core-guided MaxSAT with soft cardinality constraints, in: Proc. CP, Vol. 8656 of Lecture Notes in Computer Science, Springer, 2014, pp. 564–573.

[55] J. Davies, F. Bacchus, Exploiting the power of MIPs solvers in MaxSAT, in: Proc. SAT, Vol. 7962 of Lecture Notes in Computer Science, Springer, 2013, pp. 166–181.

[56] C. Ansótegui, J. Gabàs, Solving (weighted) partial MaxSAT with ILP, in: Proc. CPAIOR, Vol. 7874 of Lecture Notes in Computer Science, Springer, 2013, pp. 403–409.

[57] J. P. Marques-Silva, I. Lynce, Towards robust CNF encodings of cardinality constraints, in: Proc. CP, Vol. 4741 of Lecture Notes in Computer Science, Springer, 2007, pp. 483–497.

[58] I. Abío, R. Nieuwenhuis, A. Oliveras, E. Rodríguez-Carbonell, A parametric approach for smaller and better encodings of cardinality constraints, in: Proc. CP, Vol. 8124, Springer, 2013, pp. 80–96.

[59] S. Prestwich, CNF encodings, in: Handbook of Satisfiability, IOS Press, 2009, Ch. 2, pp. 75–97.

[60] C. Sinz, Towards an optimal CNF encoding of boolean cardinality constraints, in: Proc. CP, Vol. 3709 of Lecture Notes in Computer Science, 2005, pp. 827–831.

[61] F. Heras, A. Morgado, J. Marques-Silva, An empirical study of encodings for group MaxSAT, in: Proc. Canadian Conference on AI, Vol. 7310 of Lecture Notes in Computer Science, Springer, 2012, pp. 85–96.

[62] T. Nepusz, R. Sasidharan, A. Paccanaro, SCPS: a fast implementation of a spectral method for detecting protein families on a genome-wide scale, BMC Bioinformatics 11 (2010) 120.

[63] S. Altschul, W. Gish, W. Miller, E. Myers, D. Lipman, Basic local alignment search tool, J. Mol. Bio. 215 (3) (1990) 403–410.

[64] J. Davies, F. Bacchus, Solving MaxSAT by solving a sequence of simpler SAT instances, in: Proc. CP, Vol. 6876 of Lecture Notes in Computer Science, Springer, 2011, pp. 225–239.

[65] J. Davies, F. Bacchus, Postponing optimization to speed up maxsat solving, in: Proc. CP, Vol. 8124 of Lecture Notes in Computer Science, Springer, 2013, pp. 247–262.

[66] T. Achterberg, T. Berthold, T. Koch, K. Wolter, Constraint integer programming: A new approach to integrate CP and MIP, in: Proc. CPAIOR, Vol. 5015 of Lecture Notes in Computer Science, Springer, 2008, pp. 6–20.

[67] A. Wirth, Correlation clustering, in: Encyclopedia of Machine Learning, Springer, 2010, pp. 227–231.

[68] J. Sturm, Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones, Optimization Methods and Software 11–12 (1999) 625–653, version 1.05 available from `http://fewcal.kub.nl/sturm`.

[69] C. Buchheim, M. D. Santis, L. Palagi, A fast branch-and-bound algorithm for non-convex quadratic integer optimization subject to linear constraints using ellipsoidal relaxations, Operations Research Letters 43 (4) (2015) 384–388.

[70] D. Cohn, R. Caruana, A. McCallum, Semi-supervised clustering with user feedback, Tech. rep. (2003).

[71] I. Davidson, S. S. Ravi, M. Ester, Efficient incremental constrained clustering, in: P. Berkhin, R. Caruana, X. Wu (Eds.), KDD, ACM, 2007, pp. 240–249.

[72] W. Rand, Objective criteria for the evaluation of clustering methods, Journal of the American Statistical Association 66 (336) (1971) 846–850.

[73] A. Belov, A. Morgado, J. Marques-Silva, SAT-based preprocessing for MaxSAT, in: Proc. LPAR, Vol. 8312 of Lecture Notes in Computer Science, Springer, 2013, pp. 96–111.

[74] J. Berg, P. Saikko, M. Järvisalo, Improving the effectiveness of SAT-based preprocessing for MaxSAT, in: Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015), AAAI Press, 2015.

[75] A. Belov, M. Järvisalo, J. Marques-Silva, Formula preprocessing in MUS extraction, in: Proc. TACAS, Vol. 7795 of Lecture Notes in Computer Science, Springer, 2013, pp. 108–123.

[76] N. Manthey, Coprocessor 2.0 - a flexible CNF simplifier (tool presentation), in: Proc. SAT, Vol. 7317 of Lecture Notes in Computer Science, Springer, 2012, pp. 436–441.

[77] N. Eén, A. Biere, Effective preprocessing in SAT through variable and clause elimination, in: Proc. SAT, Vol. 3569 of Lecture Notes in Computer Science, Springer, 2005, pp. 61–75.

[78] M. Järvisalo, A. Biere, M. Heule, Blocked clause elimination, in: Proc. TACAS, Vol. 6015 of Lecture Notes in Computer Science, Springer, 2010, pp. 129–144.

[79] M. Heule, M. Järvisalo, A. Biere, Clause elimination procedures for CNF formulas, in: Proc. LPAR, Vol. 6397 of Lecture Notes in Computer Science, Springer, 2010, pp. 357–371.

[80] M. Heule, M. Järvisalo, A. Biere, Efficient CNF simplification based on binary implication graphs, in: Proc. SAT, Vol. 6695 of Lecture Notes in Computer Science, 2011, pp. 201–215.

[81] M. Järvisalo, A. Biere, Reconstructing solutions after blocked clause elimination, in: Proc. SAT, Vol. 6175 of Lecture Notes in Computer Science, Springer, 2010, pp. 340–345.

[82] N. Narodytska, F. Bacchus, Maximum satisfiability using core-guided MaxSAT resolution, in: Proc. AAAI, AAAI Press, 2014, pp. 2717–2723.

[83] R. Martins, V. M. Manquinho, I. Lynce, Open-WBO: A modular MaxSAT solver,, in: Proc. SAT, Vol. 8561 of Lecture Notes in Computer Science, Springer, 2014, pp. 438–445.

[84] R. Martins, S. Joshi, V. M. Manquinho, I. Lynce, Incremental cardinality constraints for MaxSAT, in: Proc. CP, Vol. 8656 of Lecture Notes in Computer Science, Springer, 2014, pp. 531–548.

[85] J. Larrosa, F. Heras, Resolution in Max-SAT and its relation to local consistency in weighted CSPs, in: Proc. IJCAI, Professional Book Center, 2005, pp. 193–198.

[86] E. D. Demaine, N. Immorlica, Correlation clustering with partial information, in: Proc. RANDOM-APPROX, Vol. 2764 of Lecture Notes in Computer Science, Springer, 2003, pp. 1–13.

[87] M. X. Goemans, D. P. Williamson, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming, J. ACM 42 (6) (1995) 1115–1145.

[88] A. McCallum, B. Wellner, Conditional models of identity uncertainty with application to noun coreference, in: Proc. NIPS, 2004, pp. 905–912.

[89] L. M. de A. Drummond, R. M. V. Figueiredo, Y. Frota, M. Levorato, Efficient solution of the correlation clustering problem: An application to structural balance, in: Proc. OTM Workshops, Vol. 8186 of Lecture Notes in Computer Science, Springer, 2013, pp. 674–683.

[90] B. Babaki, T. Guns, S. Nijssen, Constrained clustering using column generation, in: Proc. CPAIOR, Vol. 8451 of Lecture Notes in Computer Science, Springer, 2014, pp. 438–454.

[91] A. Bar-Hillel, T. Hertz, N. Shental, D. Weinshall, Learning distance functions using equivalence relations, in: Proc. ICML, AAAI Press, 2003, pp. 11–18.

[92] I. Davidson, S. S. Ravi, Using instance-level constraints in agglomerative hierarchical clustering: theoretical and empirical results, Data Min. Knowl. Discov. 18 (2) (2009) 257–282.

[93] T. Coleman, J. Saunderson, A. Wirth, Spectral clustering with inconsistent advice, in: Proc. ICML, ACM, New York, NY, USA, 2008, pp. 152–159.

[94] M. Mueller, S. Kramer, Integer linear programming models for constrained clustering, in: Proc. DS, Vol. 6332 of Lecture Notes in Computer Science, Springer, 2010, pp. 159–173.

[95] J. Schmidt, E. M. Brändle, S. Kramer, Clustering with attribute-level constraints, in: Proc. ICDM, IEEE, 2011, pp. 1206–1211.

[96] D. Aloise, P. Hansen, L. Liberti, An improved column generation algorithm for minimum sum-of-squares clustering, Math. Program. 131 (1-2) (2012) 195–220.

[97] T.-B.-H. Dao, K.-C. Duong, C. Vrain, A declarative framework for constrained clustering, in: Proc. ECML-PKDD, 2013, pp. 419–434.

Appendix A. Proofs

We provide detailed proofs of the fact that any similarity matrix can be symmetrized without affecting the set of optimal clusterings, as discussed in Section 2.2, and the correctness of the three encodings of correlation clustering as MaxSAT, presented in Sections 6, 7 and 8.

Appendix A.1. Proof of Theorem 1

Assume that $V = \{v_1 \dots v_N\}$ is a set of N data points and $W \in \overline{\mathbb{R}}^{N \times N}$ is an asymmetric similarity matrix. Let H' be the non-simplified cost function of correlation clustering (Equation 2) and H the simplified cost function (Equation 1). We will assume wlog that none of the considered matrices include contradicting infinite values.

The proof of Theorem 1 consists of considering the two different possible sources of asymmetries. The first are pairs of indices i and j for which $W(i,j) < 0 < W(j,i)$. Any such pair will always incur a cost of

at least $\min(|W(i,j)|, W(j,i))$ to any clustering. Thus the absolute value of both $W(i,j)$ and $W(j,i)$ can be decreased by this minimum without affecting the set of optimal clusterings. Notice that after this either $W(i,j) = 0$ or $W(j,i) = 0$. This observation is formalized in Lemma 1.

Based on the above, we can assume that all pairs $W(i,j)$ and $W(j,i)$ have the same sign. Now the existence of the symmetric W^S follows from the following observations. If both $W(i,j)$ and $W(j,i)$ are non-positive, the points v_i and v_j either incur a cost of $|W(i,j)| + |W(j,i)|$ or 0 to $H'(W, cl)$ under any clustering cl. Analogously, if both are non-negative, then the points either incur a cost of $W(i,j) + W(j,i)$ or 0. Hence, by letting $W^S(i,j) = W(i,j) + W(j,i)$, cl will incur the same cost under W^S (as measured by H) as under W (as measured by H'). This discussion is formalized in the proof of Theorem 1 given after the proof of Lemma 1.

Lemma 1. *There is a similarity matrix W^T such that $W^T(i,j) \cdot W^T(j,i) \geq 0$ (i.e., both have the same sign) for all i and j and $\operatorname{argmin}_{cl}(H'(W, cl)) = \operatorname{argmin}_{cl}(H'(W^T, cl))$.*

Proof. W^T can be constructed by repeatedly applying Lemma 2 to each pair of indices corresponding to elements of opposing signs in W. □

Lemma 2. *Let i and j be any pair of indices for which $W(i,j) < 0 < W(j,i)$. There exists a similarity matrix W^t for which $W^t(i,j) \cdot W^t(j,i) = 0$ and $\operatorname{argmin}_{cl}(H'(W, cl)) = \operatorname{argmin}_{cl}(H'(W^t, cl))$.*

Proof. Construct W^t as

$$\begin{aligned} W^t(i,j) &= W(i,j) + \min(|W(i,j)|, W(j,i)), \\ W^t(j,i) &= W(j,i) - \min(|W(i,j)|, W(j,i)) \text{ and} \\ W^t(i',j') &= W(i',j') \text{ whenever } i \neq i' \text{ or } j \neq j'. \end{aligned}$$

Now either $W^t(i,j) = 0$ or $W^t(j,i) = 0$, and hence $W^t(i,j) \cdot W^t(j,i) = 0$. Notice also that W^t includes exactly the same infinite values as W. This means that the set of feasible clusterings is the same for both matrices. We prove the second part of the lemma by showing that

$$H'(W, cl) = H'(W^t, cl) + \min(|W(i,j)|, W(j,i)) \tag{A.1}$$

for any feasible clustering cl of V. The fact that the set of optimal clusterings under W is the same as under W^t follows from $\min(|W(i,j)|, W(j,i))$ being independent of cl.

First, if either $W(i,j)$ or $W(j,i)$ is infinite, then it is infinite in W^t. Furthermore, the other element is 0 in W^t. Hence the pair i, j will incur cost $\min(|W(i,j)|, W(j,i))$ under W and 0 under W^t. As all other elements are equal in both matrices, we have $H'(W, cl) = H'(W^t, cl) + \min(|W(i,j)|, W(j,i))$.

Assume now that both $W(i,j)$ and $W(j,i)$ are finite. As the transformation from W to W^t maintains signs of all elements, Equation A.1 is equivalent to

$$\begin{aligned} & \mathcal{I}[cl(v_i) = cl(v_j)] \cdot |W(i,j)| + \mathcal{I}[cl(v_j) \neq cl(v_i)] \cdot W(j,i) \\ = \; & \mathcal{I}[cl(v_i) = cl(v_j)] \cdot |W^t(i,j)| + \mathcal{I}[cl(v_j) \neq cl(v_i)] \cdot W^t(j,i) + \min(|W(i,j)|, W(j,i)). \end{aligned}$$

This can be verified by considering the possible cases separately. □

Proof. (of Theorem 1) By Lemma 1 we can assume that $W(i,j) \cdot W(j,i) \geq 0$ for all i and j. Let $W^S(i,j) = W(i,j) + W(j,i)$. It is clear that W^S is symmetric. It remains to be shown that $\operatorname{argmin}_{cl}(H(W^S, cl)) = \operatorname{argmin}_{cl}(H'(W, cl))$. First note that $W^S(i,j) = \pm\infty$ iff either $W(i,j) = \pm\infty$ or $W(j,i) = \pm\infty$, so the set of feasible clusterings is the same for both matrices.

Let $i < j$ and cl be any feasible clustering of V. We show that $H(W^S, cl) = H'(W, cl)$. By decomposing both H and H' as in the proof of Lemma 1, is enough to show that

$$\begin{aligned} & \mathcal{I}[-\infty < W(i,j) < 0] \cdot |W(i,j)| + \mathcal{I}[-\infty < W(j,i) < 0] \cdot |W(j,i)| \\ = \; & \mathcal{I}[-\infty < W^S(i,j) < 0] \cdot |W^S(i,j)| \end{aligned}$$

and

$$\begin{aligned} & \mathcal{I}[\infty > W(i,j) > 0] \cdot W(i,j) + \mathcal{I}[\infty > W(j,i) > 0] \cdot W(j,i) \\ = \; & \mathcal{I}[\infty > W^S(i,j) > 0] \cdot W^S(i,j), \end{aligned}$$

corresponding to the two possible scenarios, $cl(v_i) = cl(v_j)$ and $cl(v_i) \neq cl(v_j)$, respectively. Both equations follow from the fact that the transformation from W to W^S preserves the signs of all elements. Thus $|W^S(i,j)| = |W(i,j) + W(j,i)|$. □

Appendix A.2. Correctness of the MaxSAT encodings

Next we move on to prove the correctness of the three MaxSAT encodings presented in this work, in other words, we prove Theorems 2, 3 and 4. Again, let $V = \{v_1, \dots v_N\}$ be a set of data points, $W \in \overline{\mathbb{R}}^{N \times N}$ a symmetric similarity matrix, and K an upper bound on the available clusters. Note that we allow $K = N$, so the proofs presented here cover the problem definition of [4, 12] and [9] as well as [8]. We first consider general conditions for correct MaxSAT encodings of correlation clustering. Recall that H is the cost function, Equation 1, of correlation clustering under minimization.

Proposition 1. *Let F be a MaxSAT instance and assume that a clustering $cl_\tau \colon V \to \{1 \dots K\}$ can be constructed from any solution τ to F. Further assume the following.*

1. *cl_τ is well-defined for all solutions τ to F.*
2. *For each solution τ to F, cl_τ respects the infinite values of W.*
3. *For each clustering cl that respects the infinite values of W, there exists some solution τ to F for which $H(W, cl) = H(W, cl_\tau)$.*
4. *$\text{COST}(F, \tau) = H(W, cl_\tau)$ for any solution τ to F.*

Now, if τ^* is an optimal solution to F, then cl_{τ^*} is an optimal clustering of V.

Proof. First note that Condition 1 ensures that cl_{τ^*} is well-defined and Condition 2 ensures that cl_{τ^*} is indeed a solution to the constrained problem. Now let cl be any clustering that respects the infinite values of W. Then by Condition 3 there exists a solution τ to F such that $H(W, cl_\tau) = H(W, cl)$. By the optimality of τ^* and condition Condition 4 it follows that

$$H(W, cl) = H(W, cl_\tau) = \text{COST}(F, \tau) \geq \text{COST}(F, \tau^*) = H(W, cl_{\tau^*}),$$

and hence cl_{τ^*} is optimal. □

Next we prove Theorems 2, 3 and 4 by showing that the instances generated with the transitive, unary and binary encodings fulfill the assumptions of Proposition 1.

Appendix A.2.1. Correctness of the Transitive Encoding

Let $F^1 = (F^1_h, F^1_s, c)$ be a MaxSAT instance generated by the transitive encoding, and cl_τ be the clustering constructed from a solution τ to F^1 by the procedure described in Section 6.4. The proof of Theorem 2, i.e., the fact that the transitive encoding produces optimal clusterings, follows from the following lemmas.

Lemma 3. *For any solution τ to F^1 and any $i < j$, we have $\tau(x_{ij}) = 1 \Leftrightarrow cl_\tau(v_i) = cl_\tau(v_j)$.*

Proof. Assume $cl_\tau(v_i) = k$. The lemma follows from the two possible scenarios that can occur when constructing cl_τ at iteration k.

(i) i is the smallest not yet assigned index. Then clearly $\tau(x_{ij}) = 1 \Leftrightarrow cl_\tau(v_i) = k = cl_\tau(v_j)$.

(ii) Some other index $t < i$ for which $\tau(x_{ti}) = 1$ is the smallest non assigned index. Now $\tau(x_{ij}) = 1 \Leftrightarrow \tau(x_{tj}) = 1 \Leftrightarrow cl_\tau(v_i) = k = cl_\tau(v_j)$. The first equivalence follows from τ being a solution to F^1. Thus $\tau((\neg x_{ij} \vee \neg x_{ti} \vee x_{tj})) = 1$, implying $\tau(x_{ij}) = 1 \Rightarrow \tau(x_{tj}) = 1$, and $\tau((\neg x_{ti} \vee \neg x_{tj} \vee x_{ij})) = 1$, implying $\tau(x_{ij}) = 0 \Rightarrow \tau(x_{tj}) = 0$. □

Lemma 4. *(Condition 1 of Proposition 1) cl_τ is well-defined for all solutions τ to F^1.*

Proof. Trivial, as each point is assigned to at most one cluster by the procedure in Section 6.4 and the procedure only terminates after all points have been assigned to a cluster. □

Lemma 5. *(Condition 2 of Proposition 1) cl_τ respects the infinite values of W for all solutions τ to F^1.*

Proof. First notice that, due to the hard unit clauses (x_{ij}) and $(\neg x_{ij})$, $\tau(x_{ij}) = 1$ for all $W(i,j) = \infty$, and $\tau(x_{ij}) = 0$ for all $W(i,j) = -\infty$. The rest follows from Lemma 3. □

Lemma 6. *(Condition 3 of Proposition 1) For each clustering cl that respects the infinite values of W there exists some solution τ to F for which $H(W, cl) = H(W, cl_\tau)$.*

Proof. We construct such a τ as follows:

$$\tau(x_{ij}) = \begin{cases} 1 \text{ if } cl(v_i) = cl(v_j) \\ 0 \text{ else} \end{cases} .$$

Notice that τ satisfies all hard transitivity clauses since cl is well-defined. Furthermore, τ satisfies all unit hard clauses since cl respects the infinite values of W. Finally, the claim $H(W, cl) = H(W, cl_\tau)$ follows from Lemma 3 and the construction of τ as $cl(v_i) = cl(v_j) \Leftrightarrow \tau(x_{ij}) = 1 \Leftrightarrow cl_\tau(v_i) = cl_\tau(v_j)$. □

Lemma 7. *(Condition 4 of Proposition 1) $\text{COST}(F^1, \tau) = H(W, cl_\tau)$ holds for any solution τ to F^1.*

Proof. We consider the part $H(W, cl_\tau) \leq \text{COST}(F^1, \tau)$. The other direction is almost identical. A similar pair of points v_i and v_j incurs a cost $W(i,j)$ to $H(W, cl_\tau)$ iff $cl_\tau(v_i) \neq cl_\tau(v_j)$. By Lemma 3, $\tau(x_{ij}) = 0$, and hence τ does not satisfy the unit soft clause (x_{ij}) of weight $W(i,j)$. Similarly, a dissimilar pair of points v_i, v_j incurring a cost $W(i,j)$ to $H(W, cl_\tau)$ corresponds to one unsatisfied soft clause $(\neg x_{ij})$ of the same weight. □

Appendix A.2.2. Correctness of the Unary Encoding

Let F^2 be a MaxSAT instance generated with the unary encoding and, given a solution τ to F^2, let cl_τ be the clustering constructed form τ by the procedure described in Section 7.5. The proof of Theorem 3 follows from the following lemmas.

Lemma 8. *(Condition 1 of Proposition 1) cl_τ is a well-defined clustering.*

Proof. Follows directly from the fact that, for any point v_i, $\tau(\text{EXACTLYONE}(i)) = 1$, and hence there exists exactly one $1 \leq k \leq K$ for which $\tau(y_i^k) = 1$. □

Lemma 9. *(Condition 2 of Proposition 1) cl_τ respects the infinite values of W for all solutions τ to F^2.*

Proof. Let v_i be an arbitrary data point. Assume $cl_\tau(v_i) = k$. It follows that $\tau(y_i^k) = 1$. The hard clause $(\neg y_i^k \vee y_j^k)$ included for all j s.t. $W(i,j) = \infty$. This implies $\tau(y_j^k) = 1$ and $cl_\tau(v_j) = k = cl_\tau(v_i)$. The hard clause $(\neg y_i^k \vee \neg y_j^k)$ included for all j s.t. $W(i,j) = -\infty$. This implies $\tau(y_j^k) = 0$ and $cl_\tau(v_j) \neq k = cl_\tau(v_i)$. □

Lemma 10. *(Condition 3 of Proposition 1) Let $cl\colon V \to \{1, 2, \ldots, K\}$ be any clustering of V that respects the infinite values of W. There is a solution τ to F^2 such that $cl = cl_\tau$.*

Proof. We construct such a τ. For each $1 \leq i \leq N$ and $1 \leq k \leq K$, let

$$\tau(y_i^k) = \begin{cases} 1 \text{ if } cl(v_i) = k \\ 0 \text{ else} \end{cases}, \quad \tau(A_{ij}^k) = \begin{cases} 1 \text{ if } cl(v_i) = cl(v_j) = k \\ 0 \text{ else} \end{cases} \text{ and}$$

$$\tau(D_{ij}) = \begin{cases} 1 \text{ if } cl(v_i) = cl(v_j) \\ 0 \text{ else} \end{cases}.$$

Clearly $cl = cl_\tau$ as long as τ is a solution to F^2. We show that it is by considering the different types of hard constraints present in F^2.

1. Since cl is well-defined, there is exactly one k for which $cl(v_i) = k$ for each v_i. Hence $\tau(\text{EXACTLYONE}(i)) = 1$ for all $v_i \in V$.
2. By construction $\tau(A_{ij}^k) = \tau(y_i^k \wedge y_j^k)$ for all similar v_i, v_j and k.
   Hence $\tau(\text{HARDSIMILAR}(i,j,k)) = 1$.
3. If $\tau(y_i^k) = 0$ or $\tau(y_j^k) = 0$ for a dissimilar pair of points v_i, v_j, then $\tau(\neg y_i^k \vee \neg y_j^k \vee D_{ij}) = 1$. If $\tau(y_i^k) = \tau(y_j^k) = 1$, then $cl(v_i) = cl(v_j)$. Hence $\tau(D_{ij}) = 1$ and $\tau(\neg y_i^k \vee \neg y_j^k \vee D_{ij}) = 1$. Thus $\tau(\text{HARDDISSIMILAR}(i,j,k)) = 1$ for all dissimilar v_i, v_j and k.
4. For all $W(i,j) = \infty$, we have that $cl(v_i) = cl(v_j)$. Hence there exists a k for which $\tau(y_i^k) = \tau(y_j^k) = 1$. since $\tau(\text{EXACTLYONE}(v_i)) = \tau(\text{EXACTLYONE}(v_j)) = 1$, $\tau(y_i^{k'}) = \tau(y_j^{k'}) = 0$ for all other k'. Hence $\tau(y_i^k \leftrightarrow y_j^k) = 1$ holds for all k and $\tau(\text{ML}^U(v_i, v_j)) = 1$.
5. For all $W(i,j) = -\infty$ we have that $cl(v_i) \neq cl(v_j)$. Hence either $cl(v_i) \neq k$ or $cl(v_j) \neq k$ for all k. By the construction of τ it follows that $\tau(\neg y_i^k \vee \neg y_j^k) = 1$ and $\tau(\text{CL}^U(v_i, v_j)) = 1$.

□

Lemma 11. *(Condition 4 of Proposition 1) $\text{COST}(F^2, \tau) = H(W, cl_\tau)$ for any solution τ to F^2.*

Proof. We consider the part $H(W, cl_\tau) \leq \text{COST}(F^2, \tau)$. The other direction is almost identical. A similar pair of points v_i, v_j incurs a cost $W(i,j)$ to $H(W, cl_\tau)$ iff $cl_\tau(v_i) \neq cl_\tau(v_j)$. Either $\tau(y_i^k) = 0$ or $\tau(y_j^k) = 0$ (or both) for all k, and hence $\tau(A_{ij}^k) = 0$ for all k. Thus τ does not satisfy the soft clause $\text{SOFTSIMILAR}(i,j)$ with weight $W(i,j)$. Similarly a dissimilar pair of points v_i v_j incurs cost $|W(i,j)|$ to $H(W, cl_\tau)$ iff $cl_\tau(v_i) = cl_\tau(v_j)$. There is a k for which $\tau(y_i^k \wedge y_j^k) = 1$. Thus τ does not satisfy the unit soft clause $(\neg D_{ij})$ with weight $|W(i,j)|$. □

Appendix A.2.3. Correctness of the Binary Encoding

Let F^3 be a MaxSAT instance generated with the binary encoding and, given a solution τ to F^3, let cl_τ be the clustering constructed from τ by the procedure described in Section 8.4. We prove the correctness of the binary encoding for an arbitrary K. Let $k = \lceil \log_2 K \rceil$ and assume that the encoding contains k bit variables for each data point. For any number $a \in \mathbb{N}$, let a^n denote the nth bit in the bit representation of a. For any set of bits $b^k, \ldots, b^1$m denote by $(b^k \ldots b^1)_2$ the value of the bit vector interpreted as a binary number, least significant bit to the right. Finally, let $b_i^{n*} = b_i^n$ if $K^n = 1$ and $b_i^{n*} = \neg b_i^n$ if $K^n = 0$. The proof of Theorem 4, i.e., of the fact that the binary encoding produces optimal clusterings, follows from the following lemmas.

Lemma 12. *(Condition 1 of Proposition 1) cl_τ is a well-defined clustering.*

Proof. Follows from the fact that for any point v_i, τ has to assign all the values $\tau(b_i^k), \ldots, \tau(b_i^1)$ in some unique way. Hence the value $cl_\tau(v_i)$ is uniquely defined. What remains to be shown is that $1 \leq cl_\tau(v_i) \leq K$. Assume for contradiction that $cl_\tau(v_i) = A$ for some $A > K$. Then $K - 1 < A - 1 = (\tau(b_i^k), \ldots, \tau(b_i^1))_2$. Based on the properties of binary numbers, we know that $(A-1)^j = 1$ and $(K-1)^j = 0$ at the most significant bit j where the values differ. As $\tau(\text{DEFB}(i, j')) = 1$, we have $\tau(B_i^k) = 0$, a contradiction. □

Lemma 13. *(Condition 2 of Proposition 1) cl_τ respects the infinite values of W for all solutions τ to F^3.*

Proof. If $W(i,j) = \infty$, then τ has to assign $\tau(b_i^n) = \tau(b_j^n)$ for each $n = 1..k$ in order to satisfy the hard clauses corresponding to $b_i^n \leftrightarrow b_j^n$. Hence $\tau(b_i^n) = \tau(b_j^n)$ for all bits and $cl_\tau(v_i) = cl_\tau(v_j)$. If $W(i,j) = -\infty$, then $\tau(EQ_{ij}^n) = 0$ for some $n = 1..k$ due to the hard clause $(\neg EQ_{ij}^1 \vee \ldots \vee \neg EQ_{ij}^k)$. It follows that $\tau(b_i^n \leftrightarrow b_j^n) = 0$. Thus $\tau(b_i^n) \neq \tau(b_j^n)$ and $cl_\tau(v_i) \neq cl_\tau(v_j)$. □

Lemma 14. *(Condition 3 of Proposition 1) Let $cl\colon V \to \{1, 2, \ldots, K\}$ be any clustering of V that respects the infinite values of W. There is a solution τ to F^3 such that $cl = cl_\tau$.*

Proof. Construct such τ as

$$\tau(b_i^n) = (cl(v_i) - 1)^n$$

$$\tau(EQ_{ij}^n) = \begin{cases} 1 \text{ if } (cl(v_i)-1)^n = (cl(v_j)-1)^n \\ 0 \text{ else} \end{cases}$$

$$\tau(S_{ij}) = \begin{cases} 1 \text{ if } (cl(v_i)-1)^t = (cl(v_j)-1)^t \text{ for all } 1 \leq t \leq k \\ 0 \text{ else} \end{cases}$$

$$\tau(B_i^1) = \begin{cases} 1 \text{ if } K^1 = 1 \text{ and } (cl(v_i)-1)^1 = 0 \\ 0 \text{ else} \end{cases}$$

$$\tau(B_i^n) = \begin{cases} 1 \text{ if } K^n = 1, (cl(v_i)-1)^n = 0 \text{ or } (cl(v_i)-1)^m = K^m \text{ and } \tau(B_i^{n-1}) = 1 \\ 0 \text{ else} \end{cases}.$$

Clearly $cl_\tau = cl$ as long as τ is a solution to F^3, so it remains to be shown that $\tau(F_h^3) = 1$. Consider the different types of hard constraints present in F^3.

1. For any $v_i \in V$ and any bit n, the fact that $\tau(\text{DEFB}(i,n)) = 1$ follows directly from the definition given in Equation 6, recalling that $(cl(v_i)-1)^n = \tau(b_i^n)$. Furthermore, $cl(v_i) \leq K \Leftrightarrow cl(v_i)-1 < K$. Hence there is a bit position n for which $K^n = 1$, $(cl(v_i)-1)^n = 0$ and $(cl(v_i)-1)^m = K^m$ for all $n < m \leq k$. Thus $\tau(B_i^n) = 1$ and $\tau(\text{CLUSTERSLESSTHAN}(i,K)) = 1$.
2. For any $W(i,j) < \infty$, $W(i,j) \neq 0$, and any bit n position, it holds that

$$\tau(EQ_{ij}^n) = 1 \Leftrightarrow (cl(v_i)-1)^n = (cl(v_j)-1)^n \Leftrightarrow \tau(b_i^n) = \tau(b_j^n).$$

   Hence $\tau(\text{EQUALITY}(i,j,m)) = 1$.
3. For any $W(i,j) \neq 0$ and $W(i,j) \in \mathbb{R}$, it holds that

$$\tau(S_{ij}) = 1 \Leftrightarrow \tau(b_i^t) = (cl(v_i)-1)^t = (cl(v_j)-1)^t = \tau(b_j^t) \quad 1 \leq t \leq k \Leftrightarrow$$
$$\tau(EQ_{ij}^t) = 1 \quad 1 \leq t \leq k.$$

   Hence $\tau(\text{SAMECLUSTER}(i,j)) = 1$.
4. For all $W(i,j) = \infty$, $cl(v_i) = cl(v_j)$, and hence $cl(v_i)-1 = cl(v_j)-1$. By the construction of τ, we have $\tau(b_i^n) = \tau(b_j^n)$ for all n. Thus $\tau(y_i^n \leftrightarrow y_j^n) = 1$ for all bit positions, and $\tau(\text{ML}^B(v_i,v_j)) = 1$.
5. For all $W(i,j) = -\infty$, $cl(v_i) \neq cl(v_j)$ and $cl(v_i)-1 \neq cl(v_j)-1$. Hence there is a bit position n for which $(cl(v_i)-1)^n \neq (cl(v_j)-1)^n$. By the construction of τ, $\tau(EQ_{ij}^n) = 0$ and $\tau(\neg EQ_{ij}^1 \vee \ldots \vee \neg EQ_{ij}^k) = 1$. As we already demonstrated that $\tau(\text{EQUALITY}(i,j,m)) = 1$ holds for all m, we conclude that $\tau(\text{CL}^B(v_i,v_j)) = 1$.

□

Lemma 15. *(Condition 4 of Proposition 1)* $\text{COST}(F^3,\tau) = H(W,cl_\tau)$ *for any solution* τ *to* F^3.

Proof. As the semantics of the S_{ij} variables exactly match the x_{ij} variables from the transitive encoding, the proof of this lemma is almost identical to the proof of the corresponding result for the transitive encoding. The key observation is that any pair of points v_i and v_j increases the cost of a MaxSAT solution by $|W(i,j)|$ iff it also increases the cost of cl_τ by $|W(i,j)|$. □