A Flexible ILP Formulation for Hierarchical Clustering

Sean Gilpin[a], Ian Davidson[a]

[a] *Computer Science Department*
University of California Davis
One Shields Road
Davis, California, 95616, United States of America

Abstract

Hierarchical clustering is a popular approach in a number of fields with many well known algorithms. However, all existing work to our knowledge implements a greedy heuristic algorithm with no explicit objective function. In this work we formalize hierarchical clustering as an integer linear programming (ILP) problem with a natural objective function and the dendrogram properties enforced as linear constraints. Our experimental work shows that even for small data sets finding the global optimum produces more accurate results. Formalizing hierarchical clustering as an ILP with constraints has several advantages beyond finding the global optima. Relaxing the dendrogram constraints such as transitivity can produce novel problem variations such as finding hierarchies with overlapping clusterings. It is also possible to add constraints to encode guidance such as `must-link`, `cannot-link`, `must-link-before` etc. Finally, though exact solvers exists for ILP we show that a simple randomized algorithm and a linear programming (LP) relaxation can be used to provide approximate solutions faster.

Keywords: hierarchical clustering, constraints, integer linear programming

1. Introduction

A recent survey [1] comparing non-hierarchical and hierarchical clustering algorithms showed that in published scientific articles, hierarchical algorithms are used far

Email addresses: `sagilpin@ucdavis.edu` (Sean Gilpin), `davidson@cs.ucdavis.edu` (Ian Davidson)

more than non-hierarchical clustering. However, most research and papers focus on non-hierarchical (partitional) clustering. Applications of hierarchical clustering typically can be divided into those that build large trees so that, for instance, a user can navigate a large collection of documents, and those that build trees to represent a scientific process, such as phylogenetic trees (evolutionary trees). We can further differentiate these works by noting that the data collected for the first type of application is easy to collect and hence voluminous, whilst the later application typically takes much time to collect and are typically small.

Our own earlier work [2] showed how to scale hierarchical clustering using hashing to huge data sets but in this paper the focus is the latter category of applications involving a small number of instances taking a long time to collect and which must be thoroughly analyzed. This means that spending hours for an algorithm to run is not uncalled for since a precise answer is worth the wait. Colloquially, going back to our examples, no one will complain if a dendrogram places documents in a good but non-optimal ordering but will if species are shown to evolve in the wrong order.

Hierarchical clustering algorithms remain relatively under-studied with most algorithms being relatively straight-forward greedy algorithms implemented in procedural language. For example, in the above mentioned survey two thirds of the implementations mentioned were simple agglomerative algorithms (see Figure 1) that start with each instance in a cluster by itself and then the closest two clusters are merged at each and every level. Even more advanced methods published in the database literature such as CLARANS and DBSCAN [3] still use this base approach but have more complex distance measures or methods to build the tree iteratively. Whereas non-hierarchical clustering algorithms have moved to elegant linear algebra formulations such as spectral clustering [4] hierarchical clustering has not.

Contributions. In this work we provide the first formalization of agglomerative clustering as an ILP problem (for a general introduction to the topic see [5] and for an example of an application of ILP to the non-hierarchical clustering data mining problem see [6]). Formulating the problem as an ILP allows the use of high quality solvers (free and commercial) such as CPLEX and Gurobi (used in all our experiments) to find solutions and automatically take advantage of multi-core architectures. Formulating

the problem as an ILP has a number of additional benefits that we now briefly discuss and provide details of later.

- *Explicit Global Objective Function Optimizing.* As mentioned most existing work greedily determines the best join at each and every level of the hierarchy. At no time is it possible to reset or revisit an earlier join. This is adequate when a "near enough" dendrogram is required, such as building a tree to organize song lists. Finding the global optima is more important when the data represents a physical phenomenon. This is discussed in the Section *Hierarchical Clustering as an ILP,* and we show this produces better quantitative results for scientific data sets such as language evolution in Table 3 and for hierarchical organization of fMRI scans in Table 4 .

- *Novel Problem Variations with Relaxing Constraints.* A side benefit of formalizing hierarchy learning is that the properties of a legal dendrogram are explicitly modeled as constraints to the optimization. We will show how novel problem variations can arise if some constraints are relaxed. In particular we show that relaxing the transitivity property allows for overlapping hierarchical clustering, that is, an instance can appear multiple times at the same level in the hierarchy and is akin to overlapping clustering. To our knowledge the problem of building dendrograms when an object appears multiple times is novel. This topic is explored in the Section *Relaxed Problem Settings*. In Figure 8 we show empirically that our method is capable of discovering overlapping hierarchies.

- *Novel Forms of Guidance By Adding Constraints.* The area of constrained clustering [7] has typically added constraints to procedural implementations. Here we show how we can add a number of typically used data mining constraints `must-link`, `cannot-link`, and `must-link-before` as linear constraints. This topic is explored in the Section *Adding Guidance* and in Figure 9 we show this can obtain even better empirical results.

- *Approximation Schemes* Finding the global optima to an intractable problem by definition must be time consuming. However, a large literature exists on general

methods to create approximate methods for ILPs. We provide two initial steps in that direction by exploiting a simple randomized algorithm that can create a factor two approximation scheme. We also explore using LP relaxations and randomized rounding. Figure 10 shows the empirical results of how well our LP relaxation and randomized rounding scheme compares to the optimal solution.

We organize the paper as follows. In Section 2 we describe our ILP formulation of hierarchical clustering. In Section 3 we describe an interesting variant of this formulation that allows us to learn overlapping hierarchical clustering by relaxing some of the constraints. Adding guidance that can encode domain knowledge or problem constraints in the form of additional ILP constraints is explored in Section 4. Section 5 describes two approximation algorithms to our overlapping hierarchical clustering algorithms. Finally in Section 6 we describe the questions we attempted to answer through empirical analysis and described the methods we devised to solve them as well as our results.

2. Hierarchical Clustering as an ILP

In this section, we discuss how to formulate hierarchical clustering as an ILP problem. We begin with a brief introduction of hierarchical clustering that can be skipped by the informed reader.

2.1. A Brief Introduction to Agglomerative Hierarchical Clustering

Agglomerative hierarchical clustering algorithms take as input a distance function and create a **dendrogram**, which is a tree structure which can be interpreted as a k-block set partition for each value of k between 1 and n, where n is the number of data points to cluster. These algorithms not only allow a user to choose a particular clustering granularity, but in many domains [8, 9, 10] clusters naturally form a hierarchy; that is, clusters are part of other clusters such as in the case of phylogenetic (evolutionary) trees. The popular agglomerative algorithms are easy to implement as they just begin with each point in its own cluster and progressively join two closest clusters to

Standard agglomerative clustering

Input:

- Distance matrix for n points $D \in n \times n$.
- Cluster distance function $d(c_i, c_j)$, e.g. single or complete linkage.

Output: Block partitioning Dendrogram$_k$, for each k, $1 \leq k \leq n$.

1. $c_i = \{x_i\}$, $1 \leq i \leq n$. Dendrogram$_n$ = $\{c_1,\ c_2,\ \ldots,\ c_n\}$.
2. **for** $k = n - 1$ **down to** 1 **do**
   (a) /* Find a closest pair of clusters. */
   Let (a, b) = argmin$_{(i,j)}\{d(c_i, c_j)\ :\ 1 \leq i < j \leq k + 1\}$.
   (b) Obtain Dendrogram$_k$ from Dendrogram$_{k+1}$ by merging c_b into c_a and then deleting c_b.

   endfor

Figure 1: Standard agglomerative clustering. Typical variations differ by the linkage/distance function used $d(i, j)$ whose details can drastically change the algorithm's behavior and efficient implementation. For example, the single linkage distance between two clusters is defined as the smallest distance between all pairs of points from the two different clusters.

reduce the number of clusters by 1 until $k = 1$. The basic agglomerative hierarchical clustering, which our method is compared to in this paper, is shown in Figure 1.

This algorithm is very popular and was first postulated in the 1950's and 60's in the numerical taxonomy literature [11]. Though useful they are unlike most modern machine learning methods since they were not derived to optimize an objective function. In comparison consider non-hierarchical clustering methods such as spectral clustering, k-means and mixture models. Each has an explicit object function: graph min-cut, vector quantization error, maximum likelihood respectively, and their algorithms are derived to optimize this function. Our purpose in this paper is to move hierarchical clustering forward by explicitly deriving an objective function and optimizing it. We have chosen an ILP formulation since our focus is on exact optimization.

2.2. Solving hierarchical clustering

Using ILP allows us to formally define an objective function and then find globally optimal solutions to hierarchical clustering problems. The two challenges to an ILP formulation are: i) ensuring that the resultant solution is a legal dendrogram; and ii) encoding a useful objective function. In the former, though we must model the dendrogram as a collection of variables, those variables are clearly dependent on each other and we must formalize this relationship. In the later, we must use these same variables to define a meaningful objective function.

2.3. Enforcing a Legal Dendrogram

In this section, we describe how a function (referred to as `merge` or as $\mathcal{M}$ within the ILP formulation) , representable using $O(n^2)$ integer variables (where n is the number of instances), can represent any dendrogram. The variables represent what level a pair of instances are **first** joined at and the constraints described in this section enforce that these variables obey this intended meaning. The `merge` function is described in Definition 1 as follows.

Definition 1. ***`merge` function***

$$\texttt{merge} : (\mathsf{Instances} \times \mathsf{Instances}) \rightarrow \mathsf{Levels}$$

$$(a, b) \mapsto \textit{first level } a, b \textit{ are in same cluster}$$

In this work we will learn this function. For a particular instance pair a, b the intended meaning of $\texttt{merge}(a, b) = \ell$ is that instances a, b are in the same cluster at level ℓ (and higher) of the corresponding dendrogram, but not at level $\ell - 1$ and lower. Therefore the domain of the `merge` function is all pairs of instances and the range is the integers between zero (tree bottom) and the maximum hierarchy level L (tree top).

The fact that any dendrogram from standard hierarchical clustering algorithms can be represented using this scheme is clear, but it is not the case that any such `merge` function represents a legal dendrogram. Consider a simple example with three instances a, b, c and with $\texttt{merge}(a, b) = 1$, $\texttt{merge}(a, c) = 2$ and $\texttt{merge}(b, c) = 3$. Such a `merge` function does not represent a legal dendrogram because it does not obey transitivity. To ensure we learn a legal dendrogram we must encode the following partition properties: reflexivity, symmetry, and transitivity (Definitions 2,3,4). Later we shall see how to enforce these requirements as linear inequalities in an ILP.

Definition 2. ***Reflexivity*** *An instance is always in the same cluster as itself.*

$$\forall a \left[\texttt{merge}(a, a) = 0\right]$$

Definition 3. ***Symmetry*** *If instance a is in the same cluster as instance b then instance b is also in the same cluster as instance a.*

$$\forall a, b \left[\texttt{merge}(a, b) = \texttt{merge}(b, a)\right]$$

Definition 4. ***Hierarchy and Transitivity*** *If instance a is in the same cluster as instance b at level q and b is in the same cluster as c at level r, then a must be in the same cluster as c at level q or r (which-ever is largest).*

$$\forall a, b, c \left[max\left(\texttt{merge}(a, b), \texttt{merge}(b, c)\right) \geq \texttt{merge}(a, c)\right]$$

2.4. Hierarchical Clustering Objective

The objective for agglomerative hierarchical clustering is traditionally specified as a greedy process where clusters that are closest together are merged at each level as shown in Figure 1. These objectives are called linkage functions (e.g. single [12], complete [13], UPGMA [14],WPGMA [15]) and effectively define a new distance function

Variable	Description
$\mathcal{M}(a,b)/\texttt{merge}(a,b)$	The values $\mathcal{M}(a,b)$ can be interpreted as the first level that the two points are merged.
O_{abc}	A binary variable set to 1 (TRUE) if points a, b are merged before a, c.
w_{abc}	Specifies the reward/penalty of joining a, b over a, c. In our work set as: $D(a,c) - D(a,b)$ so can be either positive (reward) or negative (penalty).
L	The maximum number of levels in the hierarchy. Unlike standard agglomerative clustering where the number of levels equals the number of points, our method allows the number of levels to be specified, and to be explicitly modeled as part of the objective function.
$Z_{ab \geq ac}$	Each Z variable captures the relative ordering of of two pairs of points' hierarchical distances (as measured in $\mathcal{M}$).

Table 1: Description of variables used in ILP formulation (See Figures 3 and 4).

($d(i,j)$ in Figure 1) to decide which clusters are closest and should therefore be merged next. The intuition behind this process can be interpreted to mean that points that are close together should be merged together before points that are far apart. *Our work formalizes this intuition into an ILP.*

To formalize that intuition we created an objective function that favors hierarchies where closer points are joined before further-away points. For example if $D(a,c)$ (distance between a and c) is larger than $D(a,b)$ then the points a and c should be merged after the points a and b are merged. This would require the merge function learnt to have $\texttt{merge}(a,b) < \texttt{merge}(a,c)$. Though learning the merge function directly allows us to recover the dendrogram easily, it is difficult to define an objective around it. Hence, we introduce the binary variable O_{abc} which is set to TRUE (value one) if the instances a and b are merged before a and c are merged. In our formulation we must

constrain O and `merge` so they are consistent with each other. We need both sets of variables since O is useful for specifying the objective function but difficult to specify the properties of a hierarchy on, and `merge` is challenging to specify an objective function on but useful for specifying the properties of a hierarchy. To complete the objective function we introduce w_{abc} a **non**-binary variable which indicates how close a and b are compared to a and c. In our experiments we set $w_{abc} = D(a,c) - D(a,b)$ which means it is set to 0 if a and b are equal distance to a and c and a positive (negative) number if they are closer (further). Figure 2 gives an example of how our objective works and Equations 1, 2, and 3 define it formally.

$$f(\mathcal{M}) = \sum_{a,b,c \in Instances} w_{abc} O_{abc} \tag{1}$$

where:

$$O_{abc} = \begin{cases} 1 & : \mathcal{M}(a,b) < \mathcal{M}(a,c) \\ 0 & : \text{otherwise} \end{cases} \tag{2}$$

$$w_{abc} = D(a,c) - D(a,b) \tag{3}$$

It should be noted that in all our experiments O is optimized over and w is a constant that is derived from the distance function D. However, in practice, w could be provided by a human expert either partially or completely. In the former case we would optimize over the not specified values of w which can be considered as a semi-supervised setting in that for some triplets you don't know their distance relationships but learn them. Using these variables allows an objective function shown formally in Equation 1 which captures the intuition of agglomerative hierarchical clustering (join closer points together before further points). However, because we have formulated it and allowed it to be solved as an ILP it will find the global optimum rather than converge to a local optimum. This is so since the ILP does not gradually build up the solution as greedy heuristic methods do.

2.5. ILP Transformation

Now that we have presented a global objective function for hierarchical clustering we will address how to find globally optimum legal hierarchies by translating the prob-

lem into an integer linear program (ILP). Figure 3 shows at a high level the optimization problem we are solving.

Figure 4 shows the ILP equivalent of the problem in Figure 3. To ensure that $\mathcal{M}$ is a legal merge function we need to ensure the resultant hierarchy is legal. This requires encoding the constraints to enforce the previously mentioned properties of reflexivity, symmetry, and transitivity properties (Definitions 2 - 4). Recall that $\mathcal{M}$ can be represented as square matrix of $n \times n$ variables indicating at what level a pair of points are joined. Then reflexivity and symmetry are both easily enforced by removing redundant variables from the matrix, in particular removing all diagonal variables for reflexivity, and all lower triangle variables for symmetry. Transitivity can be turned into a set of linear constraints by noticing that the previously mentioned inequality $max(\mathcal{M}(a,b), \mathcal{M}(b,c)) \geq \mathcal{M}(a,c)$ is logically equivalent to:

$$(\mathcal{M}(a,b) \geq \mathcal{M}(a,c)) \vee (\mathcal{M}(b,c) \geq \mathcal{M}(a,c)) \tag{4}$$

In the final ILP (Figure 4) there is a variable Z introduced for each clause/inequality from Equation 4 (i.e. $Z_{ab \geq ac}$ and $Z_{bc \geq ac}$) and there are three constraints added to enforce that the disjunction of the inequalities is satisfied. Constraints are also added to ensure that the $\mathbf{O}$ variables and the variables in $\mathcal{M}$ are consistent.

3. Relaxed Problem Settings - Removing Constraints

A benefit of formalizing hierarchy building is that individual parts of the formulization can be relaxed. Here we explore two aspects of relaxation. Firstly, we can relax the formal definitions of a legal hierarchy to obtain novel problem settings. In particular we consider the effect of relaxing the transitivity constraint of our ILP formulation of hierarchical clustering. Relaxing transitivity allows a form of hierarchical clustering where instances can appear in multiple clusters at the same level (i.e. overlapping clustering). Secondly, we relax the integer constraints so that we can create approximation algorithms by relaxing the problem into an LP and using randomized rounding schemes (see Section 5).

3.1. Overlapping Hierarchies

If the transitivity property is satisfied then the clusters at each level in the hierarchy can be obtained by computing the connected components over the pairwise `together` relations obtained from the relation $\mathcal{M}$. For any level ℓ we say two points a and b are `together` if $\mathcal{M}(a, b) \leq \ell$ and then compute transitive closure over the `together` relation. Furthermore, for a legal/standard hierarchy the set of maximal cliques will be the same as the set of connected components (because the `together` relation will be transitive for a hierarchy), which motivates our use of maximal cliques for non-transitive hierarchies. When the transitivity property of the `merge` function is relaxed, the clusterings corresponding to each level of the dendrogram found by taking the maximal cliques of the level's `together` relation will no longer necessarily be set a partitions. Clustering over non-transitive relations/graphs is common in the community detection literature [16] where overlapping clusters are learned by finding all maximal cliques in the graph (or weaker generalizations of maximal clique). We use this same approach in our work, finding maximal cliques for each level in the hierarchy to create a sequence of overlapping clusterings.

3.2. Overlapping Clustering Objective

Rather than completely relaxing the transitivity requirement we allow transitivity to be violated with a penalty. Intuitively graphs in which transitivity is better satisfied will lead to simpler overlapping clusterings. We therefore added transitivity in the objective function (rather than having it as a constraint) as shown in Equation 5 which introduces a new variable T_{abc} (Equation 6) whose value reflects whether the instance triple a, b, c obeys transitivity. We also introduce a new weight w_t which specifies how important transitivity will be to the objective.

$$f(\mathcal{M}) = \sum_{a,b,c \in Instances} [w_{abc} O_{abc} + w_t T_{abc}] \tag{5}$$

$$T_{abc} = \begin{cases} 1 & : max\left(\mathcal{M}(a,b), \mathcal{M}(b,c)\right) \geq \mathcal{M}(a,c) \\ 0 & : \text{otherwise} \end{cases} \tag{6}$$

The full ILP, with the new objective presented in Equation 5 and relaxed transitivity, is presented in Figure 5.

4. Adding Guidance - Adding Constraints

Guidance constraints such as `must–link`, `cannot–link`, and `must–link–before` constraints can be used to encode domain knowledge or problem constraints into otherwise unsupervised problems such as hierarchical clustering. Previous work showed that hierarchical clustering results can benefit from the addition of these constraints [17, 18, 19], but all previous work relied on ad-hoc schemes to integrate constraint solving into agglomerative clustering algorithms. An advantage to our ILP formulation of hierarchical clustering is that data mining constraints can be added naturally by encoding them as linear constraints, which is very straight forward.

Encoding Global Must-Link and Cannot-Link Constraints. A global `cannot–link` (CL) or `must–link` (ML) constraint, specifies that two instances must be together or apart throughout the entire hierarchy. For `must–link`, this implies that the points must have be joined from the very first level of the hierarchy as shown in Equation 7. Typically a global `cannot–link` constraint would specify that two points can never be together anywhere in the hierarchy, and thus a single `cannot–link` constraint will lead to a non-full tree hierarchy (a forest). Since in this work, we do not attempt to model non-full tree hierarchies we instead use this constraint to specify that two points cannot be merged until the highest level (the root) as show in Equation 8.

$$ML(a, b) \equiv \mathcal{M}(a, b) = 1 \tag{7}$$

$$CL(a, b) \equiv \mathcal{M}(a, b) = L \tag{8}$$

Encoding Local Must-Link and Cannot-Link Constraints. While global `cannot–link` or `must–link` constraints have been shown to improve the performance of hierarchical clustering, they were designed for flat clustering and do not take into account the hierarchical nature of a dendrogram. Local `cannot–link` (CL_i) or `must–link` (ML_i) constraints were created to allow these constraints to address particular levels in a hierarchy. For example $ML_i(a, b)$ specifies that points a and b must be together at level i in the hierarchy (or sooner) which is shown in Equation 9. Likewise $CL_i(a, b)$ specifies that a and b cannot be together at level i (in other words they

must be merged later) as is shown in Equation 10.

$$ML_i(a,b) \equiv \mathcal{M}(a,b) \leq i \tag{9}$$

$$CL_i(a,b) \equiv \mathcal{M}(a,b) > i \tag{10}$$

Encoding Local Must-Link-Before Constraints. Another constraint type that was specifically devised for hierarchical clustering is the `must-link-before` (MLB) constraints [20]. In some ways this constraint is less powerful tha local `must-link` constraints because they only specifies the relative ordering of how points should be merged. However in practice local constraints are difficult to specify whereas it is generally easy to specify a `must-link-before` constraint. As Equation 11 shows, the definition of $MLB(a,b,c)$ is that points a and b must be merged before a and c or b and c.

$$\begin{aligned} MLB(a,b,c) &\equiv [\mathcal{M}(a,b) < \mathcal{M}(a,c)] \wedge [\mathcal{M}(a,b) < \mathcal{M}(b,c)] \\ &\equiv \mathcal{M}(a,b) < \mathcal{M}(a,c) \end{aligned} \tag{11}$$

The simplification of the conjunction in Equation 11 is logically equivalent because of the properties of a hierarchy that $\mathcal{M}$ obeys. If $\mathcal{M}(a,b) < \mathcal{M}(a,c)$ then $\mathcal{M}(a,c) = \mathcal{M}(b,c)$ due to the nature of hierarchies, and therefore $\mathcal{M}(a,b) < \mathcal{M}(b,c)$.

5. Polynomial Time Approximation Algorithms

In this section we consider some theoretical results for approximations of the ILP formulation presented in Figure 5 (i.e. the ovelapping hierarchy formulation). This formulation allows transitivity to be violated, and has an interpretation that allows any variable assignment to be translated into a valid hierarchy (with overlapping clusters).

5.1. Factor Two Approximation

Theorem 1. *Let $\mathcal{M}_0$ be created by independently sampling each value from the uniform distribution $\{1 \dots L\}$, and $\mathcal{M}^*$ be the optimal solution to ILP in Figure 5. Then $E[f(\mathcal{M}_0)] \geq \frac{L-1}{2L} f(\mathcal{M}^*)$.*

Proof.

$$
\begin{aligned}
E[f(\mathcal{M}_0)] &= E\left[\sum_{abc}\left(w_{abc}O_{abc} + w'_{abc}T_{abc}\right)\right] \\
&= \sum_{abc} w_{abc}E\left[O_{abc}\right] + \sum_{abc} w'_{abc}E\left[T_{abc}\right] \\
&= \sum_{abc} w_{abc}\frac{L-1}{2L} + \sum_{abc} w'_{abc}\frac{4L^3+3L^2-L}{6L^3} \\
&= \frac{L-1}{2L}\sum_{abc} w_{abc} + \frac{4L^3+3L^2-L}{6L^3}\sum_{abc} w'_{abc} \\
&\geq \frac{L-1}{2L}f(\mathcal{M}^*) \approx \frac{1}{2}f(\mathcal{M}^*)
\end{aligned}
$$

□

Note that in the proof we use the following results for the values of $E\left[O_{abc}\right]$ and $E\left[T_{abc}\right]$:

$$E\left[O_{abc}\right] = p(\mathcal{M}_0(a,b) > \mathcal{M}_0(a,c)) = \frac{L-1}{2L}$$

$$
\begin{aligned}
E\left[T_{abc}\right] &= p(\mathcal{M}_0(a,b) \geq \mathcal{M}_0(a,c) \vee \mathcal{M}_0(b,c) \geq \mathcal{M}_0(a,c)) \\
&= 1 - \frac{2L^3-3L^2+L}{6L^3} = \frac{4L^3+3L^2-L}{6L^3}
\end{aligned}
$$

The bound $E[f(\mathcal{M}_0)] \geq \frac{L-1}{2L}f(\mathcal{M}^*)$ is a constant given L (parameter specifying number of levels in hierarchy), and will generally be close to one half since the number of levels L is typically much greater than 2.

5.2. LP Relaxation

The problem in Figure 5 can be solved as a linear program by relaxing the integer constraints, but the resulting solution, $\mathcal{M}_f^*$ will not necessarily have all integer values. Given such a solution, we can independently round each value up with probability $\mathcal{M}_f^*(a,b) - \lfloor \mathcal{M}_f^*(a,b) \rfloor$ and down otherwise. The expectation of the objective value for the LP relaxation can be calculated and in the experimental section we calculate both optimal integer solutions and the expectation for this simple rounding scheme.

Figure 10 shows that their difference is usally very small. If $\mathcal{M}_0$ is the solution created by rounding $\mathcal{M}_f^*$, then the expectation can be calculated as:

$$\begin{aligned} E[f(\mathcal{M}_0)] &= E\left[\sum_{abc}\left(w_{abc}O_{abc} + w'_{abc}T_{abc}\right)\right] \\ &= \sum_{abc} w_{abc}E\left[O_{abc}\right] + \sum_{abc} w'_{abc}E\left[T_{abc}\right] \end{aligned}$$

The expectation for each variable (T and O) breaks down into a piecewise function. The expectation for O_{abc} is listed below. This variable relies on the ordering of $\mathcal{M}_f^*(a,b)$ and $\mathcal{M}_f^*(a,c)$. When those variables are far apart (e.g. distance of 2 or greater) then rounding them will not change the value of O_{abc}. When they are close the value of O_{abc} will depend on how they are rounded, which breaks down into two cases depending on whether $\mathcal{M}_f^*(a,b)$ and $\mathcal{M}_f^*(a,c)$ are in the same integer boundary.

$$E\left[O_{abc}\right] = \begin{cases} 1 & : \lfloor \mathcal{M}_f^*(a,b) \rfloor > \lceil \mathcal{M}_f^*(a,c) \rceil \\ 1 - p_1 & : \lfloor \mathcal{M}_f^*(a,b) \rfloor = \lceil \mathcal{M}_f^*(a,c) \rfloor \\ p_2 & : \lfloor \mathcal{M}_f^*(a,b) \rfloor = \lfloor \mathcal{M}_f^*(a,c) \rfloor \\ 0 & : \lceil \mathcal{M}_f^*(a,b) \rceil < \lfloor \mathcal{M}_f^*(a,c) \rfloor \end{cases}$$

$$p_1 = \left(\lceil \mathcal{M}_f^*(a,b) \rceil - \mathcal{M}_f^*(a,b)\right)\left(\mathcal{M}_f^*(a,c) - \lfloor \mathcal{M}_f^*(a,c) \rfloor\right)$$

$$p_2 = \left(\mathcal{M}_f^*(a,b) - \lfloor \mathcal{M}_f^*(a,b) \rfloor\right)\left(\lceil \mathcal{M}_f^*(a,c) \rceil - \mathcal{M}_f^*(a,c)\right)$$

The expectation for T_{abc} can be calculated using similar reasoning but it is not listed here because it breaks down into a piecewise function with more cases and is very large.

6. Experiments

Here we overview our data sets, research questions, and experiments in that order. As in our previous work the code and data sets will be made available to aid in reproduciblity of our results.

6.1. Data Sets and Performance Measures

In keeping with the motivation of our work, these data sets are difficult to obtain so the time to find the global optimum is justified. For example, the fMRI study took many years to collect (it is a longitudinal study), the language data set is unique, and the movie data set can only grow by a few hundred instances per year.

Languages Data Set. The language data set contains phonological, morphological, lexical character usage of twenty four historic languages [21] (`http://www.cs.rice.edu/~nakhleh/CPHL/`). We chose this data set because it allowed us to test our method's ability to find a ground truth hierarchy from high dimensional data (each instance is represented by over 200 features). These languages are known to have evolved from each other with scientists agreeing upon the ground truth. In such problem settings, finding the exact order of evolution of the entities (in this case languages) is important and even small improvements are considered significant.

fMRI Data Set. An important problem in cognitive research is determining how brain behavior contributes to mental disorders such as dementia. We use a set of fMRI brain scans of patients who had also been given a series of cognitive tests/scores by doctors. These scores can be used to organize the patients into a natural hierarchy based on their mental health status and forms our ground truth hierarchy. We can then cluster the fMRI **scans** and see how well the hierarchy obtained from the scans matches the hierarchy obtained from the cognitive scores. Once again finding the best hierarchy is important since it can be used to determine those most at risk without the need for time consuming tests/scores. Each scan consists of brain activity mapped onto a $63 \times 52 \times 72$ grid over 200+ time steps. Thus each instance consists of over 500,000 columns.

Movie Lens. The Move Lens data set [22] is a set of user ratings of movies. This data set is of interest because each movie also has a set of associated genres, and the sets of movies belonging to each genre typically have a very high overlap. For example romantic-comedy movies will belong to two genres as well as science fiction horror films. Thus traditional hierarchical algorithms cannot easily discover these overlapping hierarchies as they require each data point (movies in this case) to appear only once at

each level. In contrast our methods allow a data point to appear multiple times at each level and hence the one movie can appear in the romance part of the tree as well as the comedy part of the tree if it were a romantic comedy.

We created overlapping ground truth hierarchies from all the genres and cross-genres (e.g. romantic comedy, action comedy) and tested our method's ability to find these overlapping clusterings as compared to standard agglomerative clustering methods.

20 Newsgroups. The 20 Newsgroups dataset is a collection of documents that are known to be organized into a hierarchy that is shown in Table 2.

- **Computers**
  - Hardware
    - comp.os.mswindows.misc
    - comp.windows.x
    - comp.graphics
  - Software
    - comp.sys.ibm.pc.hardware
    - comp.sys.mac.hardware
- **Recreation**
  - Automobiles
    - rec.autos
    - rec.motorcycles
  - Sports
    - rec.sport.baseball
    - rec.sport.hockey
- **Sale**
  - misc.forsale
- **Science**
  - sci.med
  - Technology
    - sci.crypt
    - sci.electronics
    - sci.space
- **Politics**
  - talk.politics.guns
  - International
    - talk.politics.mideast
    - talk.politics.misc
- **Philosophy/Religion**
  - alt.atheism
  - Theism
    - talk.religion.misc
    - soc.religion.christian

Table 2: Hierarchical structure of the 20 Newsgroup data.

Artificial Data Set. We created three artificial hierarchical clustering data sets with a known ground truth hierarchies, so that we could use them to precisely answer important questions about our new hierarchical clustering formulation. Since we often injected noise into our experiments many times we created small data sets so experiments can be repeated many times.

1. The first artificial hierarchy we created had 80 instances, 4 levels, was a balanced tree, and did not have overlapping clusterings. The ground truth hierarchy for this dataset was therefore always the same. We can turn the hierarchy into a pairwise distance matrix with the distance between two points a and b being the level of their first common ancestor. We increased the complexity of this dataset by increasingly adding uniform error to the distances. We used this data set to evaluate our basic ILP formulation which enforced transitivity (Figure 4) and compared the results with standard hierarchical clustering algorithms (single, complete, UPGMA, WPGMA).
2. The second artificial data set was created very similarly to the first, in that it also had 80 instances, 4 levels and was balanced, but each of the clusters on the highest non-root level shared 50% of their instances with another cluster (overlap). We increased the challenge of the problem by increasingly adding uniform error to the distance matrix.
3. The third set of artificial datasets was different than the previous two. We created a random ground truth hierarchies by first creating a random distance matrix and then applying a single linkage agglomerative clustering algorithm to learn the ground truth hierarhcy. We then added increasing random uniform error to increase the complexity of the problem.

6.1.1. Performance Measures

We use two standard measurements to evaluate our algorithm.

H-correlation. The H-Correlation [20] measures the distance to a ground truth hierarchy with a value of 1 being an exact perfect match and is formally defined as:

$$H_{Corr} = \frac{1}{\text{\# of tripples}} \sum_{a,b,c \in Points} \delta\{\mathcal{M}_{GT}(a,b) - \mathcal{M}_{GT}(a,c), \mathcal{M}_L(a,b) - \mathcal{M}_L(a,c)\}$$

Where $\mathcal{M}_{GT}$ is the ground truth hierarchy, $\mathcal{M}_L$ is the learned hierarchy that is being evaluated and δ is a Kronecker delta function. The more the relative order of hierarchical distances agree between the learned and ground truth hierarchy, the higher the H-Correlation.

F1 Score. The F1 score is commonly used to compare clustering results to ground truth clusters by first matching the clusters from the two different sets and then calculating:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

We measured F1 score by first calculating the set of all clusters generated within the ground truth and learned dendrograms, and then matching the learned clusters with the best true clusters. The matching between the clusters from the learned hierarchy and the ground truth hierarchy was created so that it maximized the total F1 score.

6.2. Questions

Our contributions are formulating hierarchical clustering as an optimization problem, hence we begin our experiments with the question: **1. Does our global optimization formulation of hierarchical clustering provide better results than greedy algorithms?** Our results in Tables 3 and 4 show that our method outperforms standard hierarchical clustering for real world language evolution and fMRI data sets for finding the best hierarchy (matching the ground truth). Furthermore, Figure 6 show that for artificial data sets our method performs better than existing greedy algorithms when the data set contains noise in the pairwise distances. This is a likely situation as the distances are typically measured via the instance features which may be corrupt or contain errors.

We also explore the novel idea of overlapping hierarchies, which raises the next experimental question: **2. Can our method find overlapping clusters in datasets?** Figure 8 shows the results of our method on Artificial Dataset 2 which was created to contain overlapping clusters. Table 5 shows the results for a real data set of movies where overlapping clusters naturally occur due to multiple genres and the results show that our method had better average results as well as a smaller variance than all of the agglomerative clustering algorithms. Finally, Table 6 shows our methods ability to find crosspostings (messages that belong to two different newsgroups/forum topics) with the 20 Newsgroup dataset.

3. Does the addition of constraints improve performance and run time of the algorithms? Most work on adding constraints to clustering [7] improves clustering

results but at the cost of efficiency. We explore if adding constraints to our formulation improves the accuracy of recovering the ground truth hierarchy and also if it improves performance run-time. Figure 9 shows the results for our methods.

Finally, our approximation schemes have a bound on their performance which we test: **4. How tight is the theoretical bounds for the difference between the expected and optimal solutions. In practice how do the random solutions compare to optimal solutions?** The results for our experiment showing the difference between the ILP solutions to overlapping hierarchical clustering and the LP approximation scheme solutions are shown in Figure 10.

6.3. Experimental Methodology

We now address the research questions proposed above in order.

1. Does our global optimization formulation of hierarchical clustering provide better results than greedy algorithms?

The results of the first experiment we used to answer this question are shown in Figure 6, and they shows our method's ability to outperform many standard agglomerative clustering algorithms for standard hierarchical clustering (no overlapping clusters) on artificial data (Artificial Dataset 1). The results show that our algorithm performs better when there is increasing distance error which is expected since agglomerative algorithms are greedy and erroneous steps cannot be undone. It is significant to note that even in small data sets finding the global optimum is beneficial and we expect this improvement to be larger in bigger data sets. This experiment was performed by using the F1 score to measure the difference for learned and ground truth hierarchies and repeating each test 10 times for each error factor. The error factor is the quantity of error added, so that the ground truth hierarchical distances had uniform error added in the range $[0, \textit{error factor}]$.

We then tried our algorithm on Artificial Dataset 3, whose ground truth hierarchies were created using agglomerative clustering algorithms (single linkage), and were therefore potentially more difficult for our algorithm to compete on. To see this, note that in Figure 7, the single linkage algorithm performs better than the other agglomerative clustering algorithms, which is unusual. We used single linkage because

it is known to be most flexible in the types and shapes of hierarchies that is capable of learning. Figure 7 shows not only that our ILP formulation works better than standard agglomerative clustering in this setting, but also shows how more noisy problems effect the runtime of our ILP formulation. The plot on the right side of Figure 7 shows that the ILP formulation can be very quick for many problems, but that increasing the amount of noise leads to exponential increases in run time.

We next tried our algorithms on real world data sets where we know the ground truth. Here we measure the H-correlation between the ground truth hierarchy and the hierarchy found by the algorithms. Table 3 shows that our algorithm did a better job of learning the ground truth hierarchy of languages as defined by our current understanding of language evolution. Because of the small nature of this dataset we only performed this experiment once and reported the H-correlation. For the fMRI data however, we were able to compare our method against standard agglomerative clustering multiple times and report mean and standard deviations for the H-correlation measures (See Table 4). Our method was statistically significantly better than agglomerative clustering.

Table 3: Performance of our basic ILP formulation on the languages evolution data set evaluated using H-Correlation compared to the known ground truth dendrogram (higher is better). No variances are reported in this experiment because of its small size the experiment was only performed once on the entire dataset.

Algorithm	**H-Correlation**
ILP	0.1889
Single	0.1597
Complete	0.1854
UPGMA	0.1674
WPGMA	0.1803

2. Can our method find overlapping clusters in datasets? We evaluated our overlapping clustering formulation against standard hierarchical clustering, using Artificial Dataset 2 and presented the results in Figure 8. We used the same overlapping

Table 4: Performance of our basic ILP formulation on fMRI data to recreate an hierarchy that matches the ordering of a persons fMRI scans based on their cognitive scores. Our method has higher average H-correlation with greater than 95% confidence.

Algorithm	**H-Correlation**	**Standard Deviation**
ILP	0.3305	0.0264
Single	0.3014	0.0173
Complete	0.3149	0.0306
UPGMA	0.3157	0.0313
WPGMA	0.3167	0.0332

hierarchy to test how well our expected linear programming results compared to the optimal results found using an ILP solver. Those results are presented in Figure 10 and show that in practice using an LP solution along with a very simple rounding scheme leads to results very close to the optimal ILP objective.

Table 5: Performance of overlapping formulation on real world data, the Movie Lens data set with an ideal hierarchy described by genre. Our method has higher average F1 score with greater than 95% confidence.

Algorithm	**F1 Score**	**Standard Deviation**
LP	0.595	0.0192
Single	0.547	0.0242
Complete	0.569	0.0261
UPGMA	0.548	0.0245
WPGMA	0.563	0.0246

We also used the 20 Newsgroup dataset to test our methods ability to find overlapping clustering. The idea behind this experiment is that there are many newsgroup postings that can naturally fit into multiple newsgroups. For example the newsgroups ”comp.os.mswindows.misc” and ”comp.sys.ibm.pc.hardware” are very likely to contain posts that could be cross-posted since these software and hardware platforms have

traditionally been tied together. Since the 20 Newsgroups data set we used did not contain cross-postings, we simulated them by taking two messages from similar newsgroups and adding together their bag of word representations. We repeated this experiment 10 times for different samples of the 20 Newsgroup dataset of size 100, each time creating simulated cross postings and then used our overlapping hierarchical clustering formulation. We then looked at the number of clusters that each point belonged to in the entire hierarchy and measured the average of the standard points and the simulated cross-posted points. Table 6 shows that the simulated cross postings are in many more clusters than the standard points.

Document Types	**Average Number of Clusters Belonged To**
Hybrid/Cross-posted Articles	236
Regular Articles	137

Table 6: Experiment that shows that points composed of two posts from different categories are more likely to belong to more clusters (appear multiple times at each level).

3. Does the addition of constraints improve performance and run time of the algorithms? Figure 9 shows the results of our experiment that attempted to simulate how a data mining practitioner might add constraints representing domain knowledge to our formulation. At each iteration of this experiment we learned a hierarchy and found errors in the result by comparing to the ground truth (from Artificial Data Set 3). Five new constraints were created during each iteration which were then added to a list of constraints to be used in the next iteration of the experiment. As we expected, adding a small number of constraints lead to a significant increase in the quality of results as measured using H-correlation. Another side effect that we suspected might be the case, was that adding constraints representing domain knowledge also decreased the amount of time needed to solve the problems. These constraints added were all `must-link-before` constraints which apparently have the effect of decreasing the search space without increasing the difficulty of solving corresponding linear programs.

4. How tight is the theoretical bounds for the difference between the expected

and optimal solutions. In practice how do the random solutions compare to optimal solutions?

We used the artificial overlapping hierarchy (Artificial Dataset 2) to test how well our expected linear programming results compared to the optimal results found using an ILP solver. Those results are presented in Figure 10 and show that in practice using an LP solution along with a very simple rounding scheme leads to results very close to the optimal ILP objective.

7. Conclusion

Hierarchical clustering is an important method of analysis. Most existing work focuses on heuristics to scale these methods to huge data sets which is a valid research direction if data needs to be quickly organized into any meaningful structure. However, some (often scientific) data sets can take years to collect and although they are much smaller, they require more precise analysis at the expense of time. In particular, a good solution is not good enough and a better solution can yield significant insights. In this work we explored two such data sets: language evolution and fMRI data. In the former the evolution of one language from another is discovered and in the later the organization of patients according to their fMRI scans indicates the patients most at risk. Here the previously mentioned heuristic methods perform not as well, which is significant since even small improvements are worthwhile.

We present to the best of our knowledge the first formulation of hierarchical clustering as an integer linear programming problem with an explicit objective function that is globally optimized. Our formulation has several benefits. It can find accurate hierarchies because it finds the global optimum. We found this to be particularly important when the distance matrix contains noise (see Figure 8). Since we formalize the dendrogram creation with an objective function that is constrained we can easily relax and add constraints. We showed that relaxing the dendrogram properties/constraints can lead to novel problem settings. In particular we explored relaxing transitivity to discover overlapping clustering, where an instance can appear multiple times in the hierarchy. To our knowledge the problem of overlapping hierarchical clustering has not

been addressed before. We also showed that many of the guidance-style constraints (and more) used in hierarchical settings can be represented as linear inequalites and hence added to the ILP.

We believe this work will move forward hierarchical clustering from being implemented as heuristic by making it formally modeled and optimized. This occurred in the 1990's to non-hierarchical clustering with the introduction of linear algebra formulations such as spectral clustering but has not occurred in hierarchical clustering.

8. Acknowledgments

The authors gratefully acknowledge support of this research via ONR grant N00014-11-1-0108, NSF Grant NSF IIS-0801528 and a Google Research Gift. The fMRI data provided in this grant was created by NIH grants P30 AG010129 and K01 AG 030514 and is available.

[1] J. R. Kettenring, A Perspective on Cluster Analysis, Statistical Analysis and Data Mining 1 (1) (2008) 52–53.

[2] S. Gilpin, B. Qian, I. Davidson, Efficient hierarchical clustering of large high dimensional datasets, in: CIKM, 2013.

[3] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, (First Edition), Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[4] Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92), American Mathematical Society, 1996.

[5] G. L. Nemhauser, L. A. Wolsey, Integer and combinatorial optimization, Wiley, 1988.

[6] M. Mueller, S. Kramer, Integer linear programming models for constrained clustering, in: B. Pfahringer, G. Holmes, A. Hoffmann (Eds.), Discovery Science, Vol. 6332, Springer Berlin Heidelberg, 2010, pp. 159–173.

[7] S. Basu, I. Davidson, K. Wagstaff, Constrained Clustering: Advances in Algorithms, Theory, and Applications, Chapman & Hall/CRC, 2008.

[8] Y. Zhao, G. Karypis, U. Fayyad, Hierarchical clustering algorithms for document datasets, Data Min. Knowl. Discov.doi:10.1007/s10618-005-0361-3.

[9] P. P. Mohanta, D. P. Mukherjee, S. T. Acton, Agglomerative clustering for image segmentation., in: ICPR, 2002, pp. 664–667.

[10] L. Dragomirescu, T. Postelnicu, A natural agglomerative clustering method for biology, Biometrical Journal (1991) 841–849doi:10.1002/bimj.4710330712.

[11] J. A. Hartigan, Clustering Algorithms, John Wiley & Sons, Inc., New York, NY, USA, 1975.

[12] R. Sibson, SLINK: an optimally efficient algorithm for the single-link cluster method, The Computer Journal 16 (1) (1973) 30–34.

[13] T. Sørensen, A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons, Biol. Skr. 5 (1948) 1–34.

[14] R. R. Sokal, C. D. Michener, A statistical method for evaluating systematic relationships, University of Kansas Science Bulletin 38 (1958) 1409–1438.

[15] P. H. A. Sneath, R. R. Sokal, Numerical taxonomy: the principles and practice of numerical classification, Freeman, San Francisco, USA, 1973.

[16] C. C. Aggarwal (Ed.), Social Network Data Analytics, Springer, 2011.

[17] S. Gilpin, I. Davidson, Incorporating sat solvers into hierarchical clustering algorithms: An efficient and flexible approach, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '11, 2011. doi:10.1145/2020408.2020585.

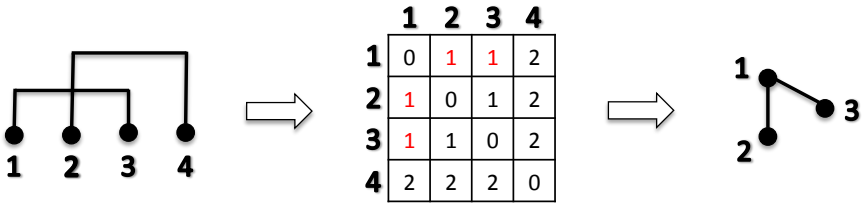
[18] I. Davidson, S. S. Ravi, Agglomerative hierarchical clustering with constraints: Theoretical and empirical results, in: Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD'05, 2005.

[19] K. Bade, A. Nurnberger, Personalized hierarchical clustering, in: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, WI '06, 2006. doi:10.1109/WI.2006.131.

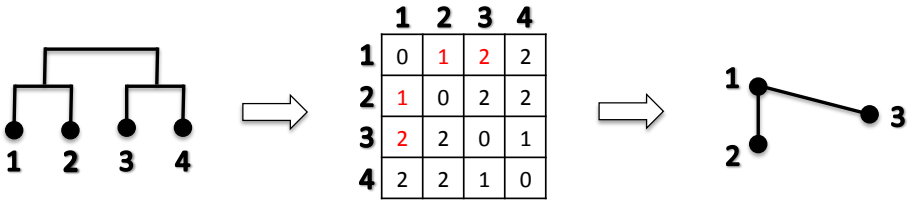
[20] K. Bade, D. Benz, Evaluation strategies for learning algorithms of hierarchies, in: Proceedings of the 32nd Annual Conference of the German Classification Society (GfKl'08), 2009.

[21] L. Nakhleh, D. Ringe, T. Warnow, Perfect phylogenetic networks : A new methodology for reconstructing the evolutionary history of natural languages, Networks 81 (2) (2002) 382–420.
URL http://www.cs.rice.edu/~nakhleh/PAPERS/NWRlanguage.pdf

[22] J. L. Herlocker, J. A. Konstan, A. Borchers, J. Riedl, An algorithmic framework for performing collaborative filtering, in: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '99, ACM, 1999.



(a) Ground truth hierarchy used as distance matrix D.



(b) Learned hierarchy and corresponding `merge` function.

Figure 2: Explanation of our objective function (see Equation 1). The top figure shows a ground truth hierarchy and its corresponding `merge` function encoding which in this hypothetical is used as the distance matrix as input to our method. The bottom figure shows a potential output from our method in terms of both the hierarchy and the corresponding `merge` matrix. If we look at the objective variable O_{123} that depends on the `merge` function of the bottom figure (whose relevant values are highlighted in red), then we can see the value should be 1 because $\mathcal{M}(1,2) < \mathcal{M}(1,3)$. However this will not lead to a reward in the objective function because $w_{123} = D(1,2) - D(1,3) = 1 - 1 = 0$, which is appropriate because we do not want to reward the solution for disagreeing with the input data.

$$\underset{\mathcal{M},\mathbf{O}}{\text{arg max}} \sum_{\{a,b,c \in Instances\}} w_{abc} * O_{abc}$$

subject to :

(1) $\mathcal{M}$ is a legal merge function

(2) $O_{abc} = \begin{cases} 1 & : \mathcal{M}(a,b) < \mathcal{M}(a,c) \\ 0 & : \text{otherwise} \end{cases}$

Figure 3: High level optimization problem for hierarchical clustering. Constraint 1 specifies that we find a legal dendrogram. Constraint 2 ensures that $\mathcal{M}$ and O are consistent with each other .

$$\underset{\mathcal{M},\mathbf{O},\mathbf{Z}}{\text{arg max}} \sum_{a,b,c \in Instances} w_{abc} * O_{abc}$$

subject to :

$\mathbf{O}, \mathbf{Z}, \mathcal{M}$ *are integers. See notation in Table 1.*

$0 \leq \mathbf{O} \leq 1,\ 0 \leq \mathbf{Z} \leq 1$

$1 \leq \mathcal{M} \leq L$

$Z_{ab \geq ac} + Z_{bc \geq ac} \geq 1$

$-L \leq \mathcal{M}(a,c) - \mathcal{M}(a,b) - (L+1)O_{abc} \leq 0$

$-L \leq \mathcal{M}(a,b) - \mathcal{M}(a,c) - (L+1)Z_{ab \geq ac} + 1 \leq 0$

$-L \leq \mathcal{M}(b,c) - \mathcal{M}(a,c) - (L+1)Z_{bc \geq ac} + 1 \leq 0$

Figure 4: ILP formulation of hierarchical clustering with global objective. The third constraint specifies the number of levels the dendrogram can have by setting the parameter L. The fourth constraint forces the O objective variables to have the meaning specified in Equation 2. Constraints 5-7 specify that $\mathcal{M}$ must be a valid dendrogram.

$$\arg\max_{\mathcal{M},\mathbf{O},\mathbf{Z}} \sum_{a,b,c \in Instances} [w_{abc} * O_{abc} + w_t * T_{abc}]$$

subject to :

$\mathbf{T}, \mathbf{O}, \mathbf{Z}, \mathcal{M}$ *are integers.*

$0 \leq \mathbf{T} \leq 1,\ 0 \leq \mathbf{O} \leq 1,\ 0 \leq \mathbf{Z} \leq 1$

$1 \leq \mathcal{M} \leq L$

$-L \leq \mathcal{M}(a,c) - \mathcal{M}(a,b) - (L+1)O_{abc} \leq 0$

$-L \leq \mathcal{M}(a,b) - \mathcal{M}(a,c) - (L+1)Z_{ab \geq ac} + 1 \leq 0$

$-L \leq \mathcal{M}(b,c) - \mathcal{M}(a,c) - (L+1)Z_{bc \geq ac} + 1 \leq 0$

$Z_{ab \geq ac} + Z_{bc \geq ac} \geq T_{abc}$

Figure 5: ILP formulation with relaxation on transitivity property that allows overlapping hierarches.

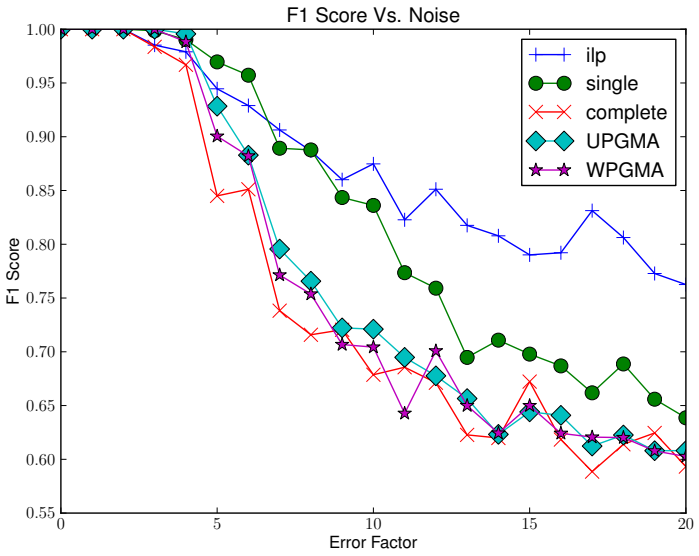


Figure 6: First Artificial Dataset. Effect of noise on prediction performance. F1 score versus noise in distance function. Comparison of basic ILP formulation on artificial data against a variety of standard competitors.

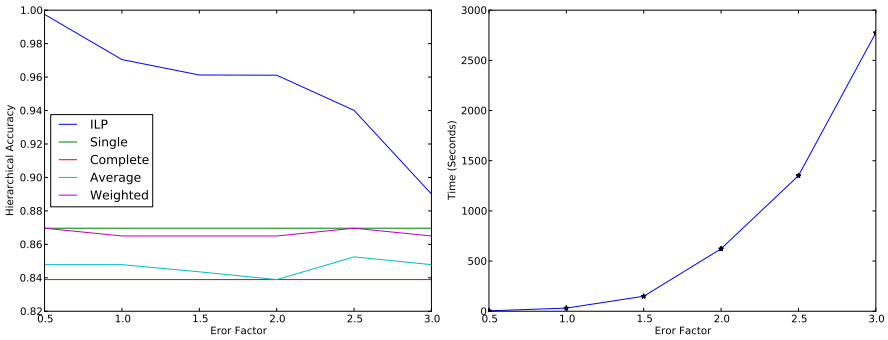


Figure 7: (Left) Average hierarchical accuracy (H-correlation) plotted against increasing amounts of error added to the artificial data set (Artificial Dataset 3). (Right) Average time required to solve problems plotted against increasing error.

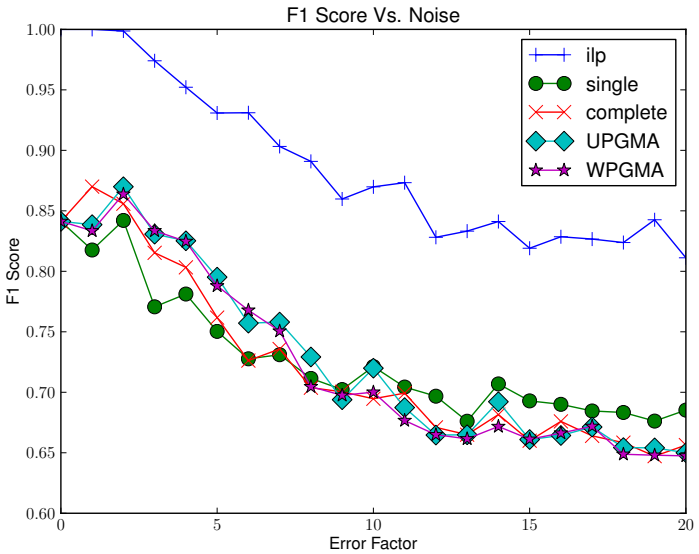


Figure 8: Ability to find overlapping clusterings. F1 score versus noise in distance function. Comparison of overlapping ILP formulation on artificial data against a variety of competitors.

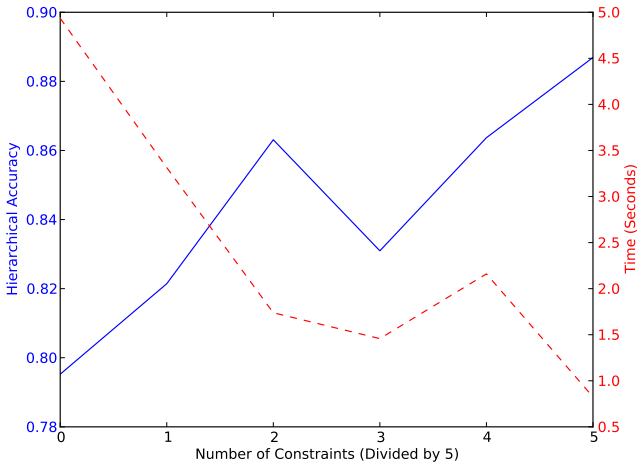


Figure 9: Hierarchical accuracy (H-correlation) and speed plotted against increasing amounts of `must-link-before` constraints generated from ground truth. At each iteration we add 5 new constraints based on discrepancies of what was learned from previous iteration, and the ground truth. These experiments were measured using the Artificial Dataset 3.

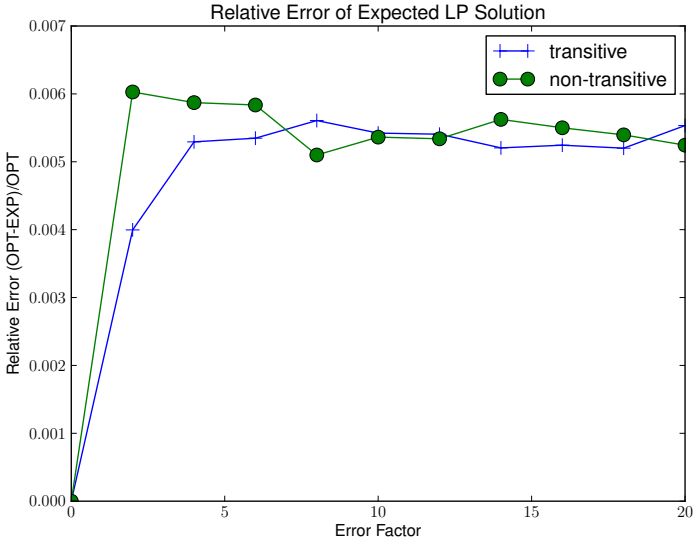


Figure 10: Measuring optimality of randomized solutions. Performance versus noise in distance function. Comparison of expected LP results versus optimal ILP results using two transitivity parameterizations (transitive $\rightarrow$ very large w_t, non-transitive $\rightarrow$ $w_t = 0$).