Towards Efficient and Improved Hierarchical Clustering With Instance and Cluster Level Constraints

Ian Davidson* S. S. Ravi†

ABSTRACT

Many clustering applications use the computationally efficient non-hierarchical clustering techniques such as k-means. However, less efficient hierarchical clustering is desirable as by creating a dendrogram the user can choose an appropriate value of k (the number of clusters) and in some domains cluster hierarchies (i.e. clusters within other clusters) naturally exist. In many situations apriori constraints/information are available such as in the form of a small amount of labeled data. In this paper we explore using constraints to improve the efficiency of agglomerative clustering algorithms. We show that just finding feasible (satisfying all constraints) solutions for some constraint combinations is NP-complete and should be avoided. For a given set of constraints we derive upper (k_{max}) and lower bounds (k_{min}) on the value of k where feasible solutions exist. This allows a restricted dendrogram to be created but its creation is not straight-forward. For some combinations of constraints, starting with a feasible clustering solution ($k = r$) and joining the two closest clusters results in a "dead-end" feasible solution which cannot be further refined to create a feasible solution with $r - 1$ clusters even though $k_{min} < r - 1 < k_{max}$. For such situations we introduce constraint driven hierarchical clustering algorithms that will create a complete dendrogram. When traditional algorithms can be used, we illustrate the use of the triangle inequality and a newly defined γ constraint to further improve performance and use the Markov inequality to bound the expected performance improvement. Preliminary results indicate that using constraints can improve the dendrogram quality.

1. INTRODUCTION AND MOTIVATION

Non-hierarchical clustering algorithms are extensively used in the data mining community. The *k-Means* **problem** for a given set of data points, $S = \{x_1, \ldots, x_n\}$ is to form a k-block set partition of S so as to minimize the vector quantization error (distortion). However, this problem is known to be **NP**-hard via showing that the distortion minimization problem for a binary distance function is intractable [7]. Instead, the k-Means **algorithm** is used to find good local minima and has linear complexity $O(kmni)$ with respect to the number of instances, attributes, clusters and iterations of the algorithm: n, m, k, i respectively. However, the algorithm is sensitive to initial starting conditions [3] and hence must be randomly restarted many times.

*Department of Computer Science, University at Albany - State University of New York, Albany, NY 12222. Email: davidson@cs.albany.edu.

†Department of Computer Science, University at Albany - State University of New York, Albany, NY 12222. Email: ravi@cs.albany.edu.

Agglomerative($S = \{x_1, \ldots, x_n\}$) **returns** $Dendrogram_k$ *for* $k = 1$ *to* $|S|$.

1. $C_i = \{x_i\}$, $\forall i$.
2. **for** $k = |S|$ *to* 1
   $Dendrogram_k = \{C_1 ... C_k\}$
   $d(i,j) = D(C_i, C_j)$, $\forall i, j$.
   $l, m = argmin_{a,b} d(a,b)$.
   $C_l = Join(C_l, C_m)$.
   $Remove(C_m)$.
   endloop

Figure 1: Standard Agglomerative Clustering

Conversely, hierarchical clustering algorithms are run once and create a dendrogram which is a tree structure containing a k-block set partition for each value of k between 1 and n. This allows the user to choose a particular clustering granularity and there even exists mature work that provides visual support [5] for the task. Hierarchical clustering also offers many measures of distance between two clusters such as: 1) the cluster centroids, 2) the closest points not in the same cluster and 3) the furthest points not in the same cluster. Furthermore, there are many domains [15] where clusters naturally form a hierarchy; that is, clusters are part of other clusters. However, these added benefits come at the cost of time and space efficiency since the typical implementation requires $O(mn^2)$ computations and $O(n^2)$ space. Hence, hierarchical clustering is not used as often in data mining applications particularly for large data sets.

The popular agglomerative algorithms are easy to implement as they just begin with each point in its own cluster and progressively join the closest clusters to reduce the number of clusters by 1 until $k = 1$. The basic agglomerative hierarchical clustering algorithm we will improve upon in this paper is shown in Figure 1.

In this paper we shall explore the use of instance and cluster level constraints to improve the efficiency of hierarchical clustering algorithms. This use of *prior knowledge* to improve efficiency is in contrast to other recent work that use just algorithmic enhancements such multi-stage algorithms with approximate distance functions [9]. We believe our use of constraints with *hierarchical* clustering to improve efficiency is the first though there exists work that uses spatial constraints to find specific types of clusters and avoid others [13, 14]. The

Constraint	Complexity
Must-Link	**P** [8, 4]
Cannot-Link	**NP**-Complete [8, 4]
δ-constraint	**P** [4]
ϵ-constraint	**P** [4]
Must-Link and δ	**P** [4]
Must-Link and ϵ	**NP**-complete [4]
δ and ϵ	**P** [4]

Table 1: Results for Feasibility Problems for a Given k (partitional clustering)

Constraint	Complexity	Dead-Ends
Must-Link+Cannot-link+ϵ+δ	**NP**-complete	Yes
Any combination including Cannot-Link (except above)	**P**	Yes
Any combination not including Cannot-Link (except above)	**P**	No

Table 2: Results for Feasibility Problems - Unbounded k (hierarchical clustering)

similarly named *constrained hierarchical clustering* [15] is actually a method of combining partitional and hierarchical clustering algorithms; the method does not incorporate apriori constraints.

Recent work [1, 2, 11] in the non-hierarchical clustering literature has explored the use instance level constraints. The **must-link** and **cannot-link** constraints require that two instances must both be part of or not part of the same cluster respectively. They are particularly useful in situations where a large amount of unlabeled data to cluster is available along with some labeled data from which the constraints can be obtained [11]. These constraints were shown to improve cluster purity when measured against an extrinsic class label not given to the clustering algorithm [11]. Our own recent work [4] explored the computational complexity of the *feasibility* problem of finding a clustering solution for a **given** value of k under a variety of constraints, including must-link and cannot-link constraints. For example, there is no feasible solution for the three cannot-link constraints $\{(a,b),(b,c),(a,c)\}$ for $k < 3$. We explored the complexity of this decision problem for a **given** value of k with must-link, cannot-link and two additional cluster level constraints: δ and ϵ. The δ **constraint** requires the distance between any pair of points in two different clusters to be at least δ. For any two point or bigger cluster C_i, the **ϵ-constraint** requires that for each point $x \in C_i$, there must be another point $y \in C_i$ such that the distance between x and y is at most ϵ. The complexity results of this work are shown in Table 1. These complexity results are important for data mining because when problems are shown to be intractable in the worst-case, we should avoid them or should not expect to find an exact solution efficiently.

In this paper we extend our previous work by exploring the complexity of the **hierarchical** clustering under the above four mentioned instance and cluster level constraints. This problem is different from the feasibility problems considered in our previous work in a number of ways. Firstly, with hierarchical clustering, k is unbounded and hence we must find the upper and lower bounds on k that contain feasible solutions. (Throughout this paper we shall use the term "feasible solution" to indicate that a clustering solution satisfies all constraints.) Secondly, for *some* combination of constraints, given an initial feasible clustering solution joining the two closest clusters may yield feasible solutions but "dead-end" solutions from which no other feasible solutions of lesser k value can be obtained even though they are known to exist. Therefore, the created dendrograms will be incomplete.

Our work makes several pragmatic contributions that can improve the applicability of agglomerative hierarchical clustering:

- We illustrate that for some combination of constraints that just finding feasible solutions is **NP**-complete and hence should be avoided (Section 2 and Table 2).
- We derive lower and upper bounds (k_{min} and k_{max}) on the number of clusters for feasible solutions. These bounds allow us to *prune* the dendrogram from both its bottom and top (Section 3).
- We formally show that for every value of k between k_{min} and k_{max} there exists a feasible clustering solution (Theorem 6.1).
- For some constraint combinations traditional (closest cluster join) algorithms are applicable and for these we illustrate the use of the new instance level γ constraint to perform geometric reasoning so as to prune the number of joins to perform at each level (Section 5).
- For some constraint combinations traditional (closest cluster join) algorithms lead to dead-end feasible solutions that do not allow a complete dendrogram to be completed. For these we develop a new algorithm that we call *constraint-driven agglomerative clustering* that will create a complete dendrogram (Section 6 and Table 2).
- We empirically illustrate the efficiency improvement over unconstrained hierarchical clustering and present preliminary results indicating that the quality of the hierarchical clustering also improves (Section 8).

2. FEASIBILITY FOR HIERARCHICAL CLUSTERING

In this section, we examine the feasibility problem, that is, the problem of determining whether the given

set of points can be partitioned into clusters so that all the specified constraints are satisfied. A precise statement of the **Feasibility problem for Hierarchical Clustering** (FHC) is given below.

Instance: A set S of nodes, the (symmetric) distance $d(x, y) \geq 0$ for each pair of nodes x and y in S and a collection C of constraints.

Question: Can S be partitioned into subsets (clusters) such that all the constraints in C are satisfied?

This section considers the complexity of the above problem for several different types of constraints. When the problem is efficiently solvable and the answer to the feasibility question is "yes", the corresponding algorithm also produces a partition of S satisfying the constraints. When the nodes in S are points in Euclidean space and the distance function is the Euclidean distance, we obtain geometric instances of FHC.

We note that the FHC problem considered here is different from the constrained clustering problem considered in [4]. The main difference between the two problems is that the feasibility problems considered in [4] were typically for a given number of clusters; that is, the number of clusters is, in effect, another constraint. In the formulation of FHC, there are *no* constraints on the number of clusters, other than the trivial ones (i.e., the number of clusters must be at least 1 and at most $|S|$).

We shall in this section begin with the same constraints as those considered in [4]. They are: (a) Must-Link (ML) constraints, (b) Cannot-Link (CL) constraints, (c) δ constraint and (d) ϵ constraint. In later sections we shall introduce another cluster level constraint to improve the efficiency of the hierarchical clustering algorithms. As observed in [4], a δ constraint can be efficiently transformed into an equivalent collection of ML-constraints. Therefore, we restrict our attention to ML, CL and ϵ constraints. We show that for any pair of these constraint types, the corresponding feasibility problem can be solved efficiently. The simple algorithms for these feasibility problems can be used to seed an agglomerative or divisive hierarchical clustering algorithm as is the case in our experimental results. However, when all three types of constraints are specified, we show that the feasibility problem is **NP**-complete.

2.1 Combination of ML and CL Constraints

When the constraint set C contains only ML and CL constraints, the FHC problem can be solved using the following simple algorithm.

1. Form the clusters implied by the ML constraints. (This can be done by computing the transitive closure of the ML constraints as explained in [4].) Let C_1, C_2, ..., C_p denote the resulting clusters.

2. If there is a cluster C_i $(1 \leq i \leq p)$ with nodes x and y such that x and y are also involved in a CL constraint, then there is no solution to the feasibility problem; otherwise, there is a solution.

When the above algorithm indicates that there is a feasible solution to the given FHC instance, one such solution can be obtained as follows. Use the clusters produced in Step 1 along with a singleton cluster for each node that is not involved in an ML constraint. Clearly, this algorithm runs in polynomial time.

2.2 Combination of CL and ϵ Constraints

There is always a trivial solution consisting of $|S|$ singleton clusters to the FHC problem when the constraint set involves only CL and ϵ constraints. Obviously, this trivial solution satisfies both CL and ϵ constraints.

2.3 Combination of ML and ϵ Constraints

For any node x, an **ϵ-neighbor** of x is another node y such that $d(x, y) \leq \epsilon$. Using this definition, the following algorithm solves the FHC problem when the constraint set consists only of ML and ϵ constraints.

1. Construct the set $S' = \{x \in S : x$ does not have an ϵ-neighbor$\}$.

2. If some node in S' is involved in an ML constraint, then there is no solution to the FHC problem; otherwise, there is a solution.

When the above algorithm indicates that there is a feasible solution, one such solution is to create a singleton cluster for each node in S' and form one additional cluster containing all the nodes in $S - S'$. It is easy to see that the resulting partition of S satisfies the ML and ϵ constraints and that the feasibility testing algorithm runs in polynomial time.

The following theorem summarizes the above discussion and indicates that we can use these combinations of constraint types to perform efficient hierarchical clustering. However, it does not mean that we can always use traditional agglomerative clustering algorithms as the closest-cluster-join operation can yield dead-end clustering solutions.

THEOREM 2.1. *The* FHC *problem can be solved in polynomial time for each of the following combinations of constraint types: (a) ML and CL (b) CL and ϵ and (c) ML and ϵ.* □

2.4 Feasibility Under ML, CL and ϵ Constraints

In this section, we show that the FHC problem is **NP**-complete when all the three constraint types are involved. This indicates that creating a dendrogram under these constraints is an intractable problem and the best we can hope for is an approximation algorithm that may **not** satisfy all constraints. The **NP**-completeness proof uses a reduction from the following problem which is known to be **NP**-complete [10].

One-in-Three 3SAT with Positive Literals (OPL)

Instance: A set $c = \{x_1, x_2, \ldots, x_n\}$ of n Boolean variables, a collection $Y = \{Y_1, Y_2, \ldots, Y_m\}$ of m clauses, where each clause $Y_j = (x_{j_1}, x_{j_2}, x_{j_3}\}$ has exactly three non-negated literals.

Question: Is there an assignment of truth values to the variables in C so that exactly one literal in each clause becomes true?

THEOREM 2.2. *The* FHC *problem is* ***NP****-complete when the constraint set contains ML, CL and ϵ constraints.*

Proof: See Appendix.

3. BOUNDING THE VALUES OF K FOR SOLUTIONS

The unconstrained version of agglomerative hierarchical clustering builds a dendrogram for all values of k, $1 \leq k \leq n$, giving rise to the algorithm complexity of $O(n^2)$. However, with the addition of constraints, not all values of k may contain feasible solutions as discussed in Section 2. In this section we explore deriving lower (k_{min}) and upper bounds (k_{max}) on the value of k that contain feasible solutions. For some constraint types (e.g. CL-constraints) the problem of computing exact values of these bounds is itself **NP**-complete, and we can only calculate bounds (in effect, bounds on the bounds) in polynomial time.

Therefore, when building the dendrogram we can **prune** the dendrogram by starting building clusters at k_{max} and stop building the tree after k_{min} clusters are reached.

As discussed earlier, the δ constraints can be converted into a conjunction of must-link constraints. Let the total number of points involved in must-link constraints be ml. For the set of must-link constraints (ML) and transformed δ constraints calculate the transitive closure (TC) (see [4] for an algorithm). For example for the must-link constraints ML=$\{(a,b), (b,c), (c,d)\}$, TC=$\{a,b,c,d\}$ and hence $|TC|$=1.

Let there be cl points that are part of cannot link constraints each being represented by a node in the graph $G_c = \{V_c, E_c\}$ with edges indicating a cannot-link constraint between two points.

Let the ϵ constraints result in a graph ($G_\epsilon = \{V_\epsilon, E_\epsilon\}$) with a vertex for each instance. The edges in the graph indicate two points that are within ϵ distance from each other.

We now derive bounds for each constraint type separately.

3.1 Bounds for ML Constraints

When only ML constraints are present, a single cluster containing all points is a feasible solution. Therefore, $k_{min} = 1$. Each set in the transitive closure cannot be split any further without violating one or more the ML constraints. However, each of the $n - ml$ points which are not involved in any ML constraint may be in a separate singleton cluster. Therefore, for just ML constraints, k_{max}=$|TC|+n-ml$. (More precisely, k_{max} = $\min\{n, |TC| + n - ml\}$.)

3.2 Bounds for CL Constraints

Determining the minimum number of clusters required to satisfy the cannot-link constraints is equivalent to determining the minimum number of different labels so that assigning a label to each of the nodes in G_c gives no adjacent pair of nodes having the same label. This parameter, that is, the minimum number of colors needed for coloring the nodes of G_c, is commonly denoted by $\chi(G_c)$ [12]. The problem of computing $\chi(G_c)$ (i.e., the graph coloring problem) is typically **NP**-complete [6]. Thus, for CL constraints, computing the exact value of $k_{min} = \chi(G_c)$ is intractable. However, a useful upper bound on $\chi(G_c)$ (and hence on k_{min}) is one plus the maximum degree of a node (the maximum number of edges incident on any single node) in G_c [12]. Obviously, for CL constraints, making n singleton clusters is a feasible solution. Therefore, when only CL constraints are present, $k_{max} = n$.

Calculate-Bounds(S,ML,ϵ, δ) **returns** k_{min}, k_{max} *where S is the dataset, ML is the set of must-link constraints, ϵ and δ values define their respective constraints.*

1. Calculate transitive closure of ML: TC = $\{\{m_1\}, ...\{m_p\}\}$ (see [4] for an algorithm)
2. Construct G_c (see section 3.2)
3. Construct G_ϵ (see section 3.3)
4. k_{min} = max{NumIsolatedNodes(G_ϵ) + 1, $\chi(G_c)$}.
5. k_{max}= min{n, $TC + n - ml$}.

Figure 2: Function for calculating k_{min} and k_{max}

3.3 Bounds for ϵ Constraints

The ϵ constraints can be represented as a graph $G_\epsilon = \{V_\epsilon, E_\epsilon\}$, with each vertex being an instance and links between vertices/instances that are within ϵ distance to each other. Let i denote the number of isolated nodes in G_ϵ. Each of these i nodes must be in a separate singleton cluster to satisfy the ϵ-constraint. Nodes in G_ϵ with a degree of one or more may be grouped into a single cluster. Thus, $k_{min} = i + 1$. (More precisely, $k_{min} = \min\{n, i+1\}$, to allow for the possibility that all the nodes of G_ϵ are isolated.) As with CL constraints, making n singleton clusters is a feasible solution for the ϵ constraint as well. Hence for ϵ constraints $k_{max} = n$.

When putting the above results together, it is worth noting that only must-link constraints can reduce the number of points to cluster and hence prune the base of the dendrogram. Cannot-link and ϵ constraints increase the minimum number of clusters.

Therefore, we find that for all constraint types, we can bound k_{min} and k_{max} as shown in the following observation.

OBSERVATION 3.1. $k_{min} \geq \max\{\chi(G_c),$ IsolatedNodes $(G_\epsilon) + 1\}$ *and* $k_{max} \leq \min\{n, TC + n - ml\}$.

A function to determine the bounds on the feasible values of k is shown in Figure 2.

4. CONSTRAINTS AND IRREDUCIBLE CLUSTERINGS

In the presence of constraints, the set partitions at each level of the dendrogram must be feasible. This section formally shows that for certain types of constraints (and combinations of constraints), if mergers are performed in an arbitrary fashion (including the traditional hierarchical clustering algorithm, see Figure 1), then the dendrogram may prematurely dead-end. A premature dead-end implies that the dendrogram reaches a stage where no pair of clusters can be merged without violating one or more constraints, even though other sequences of mergers may reach significantly higher levels of the dendrogram. We use the following definition to capture the informal notion of a "premature end" in the construction of a dendrogram.

DEFINITION 4.1. *Given a feasible clustering $C = \{C_1, C_2, \ldots, C_k\}$ of a set S, we say that C is **irreducible** if no pair of clusters in C can be merged to obtain a feasible clustering C' with $k-1$ clusters.*

The remainder of this section examines the question of which combinations of constraints can lead to premature stoppage of the dendrogram.

4.1 Individual Constraints

We first consider each of the ML, CL and ϵ-constraints separately. It is easy to see that when only ML-constraints are used, the dendrogram can reach all the way up to a single cluster, no matter how mergers are done. The following example shows that with CL-constraints, if mergers are not done correctly, the dendrogram may stop prematurely.

Example: Consider a set S with $4k$ nodes. To describe the CL constraints, we will think of S as made up of four pairwise disjoint sets X, Y, Z and W, each with k nodes. Let $X = \{x_1, x_2, \ldots, x_k\}$, $Y = \{y_1, y_2, \ldots, y_k\}$, $Z = \{z_1, z_2, \ldots, z_k\}$ and $W = \{w_1, w_2, \ldots, w_k\}$. The CL-constraints are as follows.

(a) There is a CL-constraint for each pair of nodes $\{x_i, x_j\}$, $i \neq j$.

(b) There is a CL-constraint for each pair of nodes $\{w_i, w_j\}$, $i \neq j$.

(c) There is a CL-constraint for each pair of nodes $\{y_i, z_j\}$, $1 \leq i, j \leq k$.

Assume that the distance between each pair of nodes in S is 1. Thus, nearest-neighbor mergers may lead to the following feasible clustering with $2k$ clusters: $\{x_1, y_1\}$, $\{x_2, y_2\}$, ..., $\{x_k, y_k\}$, $\{z_1, w_1\}$, $\{z_2, w_2\}$, ..., $\{z_k, w_k\}$. This collection of clusters can be seen to be irreducible in view of the given CL constraints.

However, a feasible clustering with k clusters is possible: $\{x_1, w_1, y_1, y_2, \ldots, y_k\}$, $\{x_2, w_2, z_1, z_2, \ldots, z_k\}$, $\{x_3, w_3\}$, ..., $\{x_k, w_k\}$. Thus, in this example, a carefully constructed dendrogram allows k additional levels. □

When only the ϵ-constraint is considered, the following lemma points out that there is only one irreducible configuration; thus, no premature stoppages are possible. In proving this lemma, we will assume that the distance function is symmetric.

LEMMA 4.1. *Suppose S is a set of nodes to be clustered under an ϵ-constraint. Any irreducible and feasible collection C of clusters for S must satisfy the following two conditions.*

(a) C contains at most one cluster with two or more nodes of S.

(b) Every singleton cluster in C consists of a node x such that x does not have any ϵ-neighbor in S.

Proof: Consider Part (a). Suppose C has two or more clusters, say C_1 and C_2, such that each of C_1 and C_2 has two or more nodes. We claim that C_1 and C_2 can be merged without violating the ϵ-constraint. This is because each node in C_1 (C_2) has an ϵ-neighbor in C_1 (C_2) since C is feasible and distances are symmetric. Thus, merging C_1 and C_2 cannot violate the ϵ-constraint. This contradicts the assumption that C is irreducible and the result of Part (a) follows.

The proof for Part (b) is similar. Suppose C has a singleton cluster $C_1 = \{x\}$ and the node x has an ϵ-neighbor in some cluster C_2. Again, C_1 and C_2 can be merged without violating the ϵ-constraint. □

4.2 Combinations of Constraints

Lemma 4.1 can be seen to hold even for the combination of ML and ϵ constraints since ML constraints cannot be violated by merging clusters. Thus, no matter how clusters are merged at the intermediate levels, the highest level of the dendrogram will always correspond to the configuration described in the above lemma when ML and ϵ constraints are used.

In the presence of CL-constraints, it was pointed out through an example that the dendrogram may stop prematurely if mergers are not carried out carefully. It is easy to extend the example to show that this behavior occurs even when CL-constraints are combined with ML-constraints or an ϵ-constraint.

5. THE γ CONSTRAINT AND GEOMETRIC REASONING

In this section we introduce a new constraint, the γ constraint and illustrate how the triangle inequality can be used to perform geometric reasoning to further improve the run-time performance of hierarchical clustering algorithm. Though this improvement does not affect the worst-case analysis, we can perform a best case analysis and an expected performance improvement using the Markov inequality. Future work will investigate if tighter bounds can be used.

DEFINITION 5.1. *(The γ Constraint For Hierarchical Clustering) Two clusters whose centroids are separated by a distance greater than γ cannot be joined.*

The γ constraint allows us to specify how geometrically well separated the clusters should be.

Recall that the triangular inequality for three points A, B, C refers to the expression $|D(A,B) - D(B,C)| \leq D(A,C) \leq D(A,B) + D(C,B)$ where D is the Euclidean distance function. We can improve the efficiency of the

IntelligentDistance(γ, C=$\{C_1 ... C_k\}$) **returns** $d(i,j) \forall\ i,j$

1. for $i = 2...n-1$
   $d_{1,i} = D(C_1, C_i)$
   endloop
2. **for** $i = 2$ to $n-1$
   for $j = i+1$ to $n-1$
   $\hat{d_{i,j}} = |d_{1,i} - d_{1,j}|$
   endloop
   if $\hat{d_{i,j}} > \gamma$ then $d_{i,j} = \gamma + 1$ *; do not join*
   else $d_{i,j} = D(x_i, x_j)$
   endloop
3. return $d(i,j) \forall i,j$

Figure 3: Function for Calculating Distances Using the γ constraint

hierarchical clustering algorithm by making use of the lower bound in the triangle inequalities and the γ constraint. Let A, B, C now be cluster centroids. If we have already computed $D(A,B)$ and $D(B,C)$ then if $|D(A,B) - D(B,C| > \gamma$ then we need not compute the distance between A and C as the lower bound on $D(A,C)$ already exceeds γ. Formally the function to calculate distances using geometric reasoning at a particular level is shown in Figure 3.

If triangular inequality bound exceeds γ, then we save making m floating point power calculations and a square root calculation if the data points are in m dimensional space. Since we have to calculate the minimum distance we have already stored $D(A,B)$ and $D(B,C)$ so there is no storage overhead.

As mentioned earlier we have no reason to believe that there will be at least one situation where the triangular inequality saves computation in *all problem instances*, hence in the worst case, there is no performance improvement. We shall now explore the best and expected case analysis.

5.1 Best Case Analysis for Using the γ constraint

Consider the n points to cluster $\{x_1, ..., x_n\}$. The first iteration of the agglomerative hierarchical clustering algorithm using symmetrical distances is to compute the distance between each point and every other point. This involves the computation ($D(x_1, x_2)$, $D(x_1, x_3)$, ..., $D(x_1, x_n)$), ..., ($D(x_i, x_{i+1})$, $D(x_i, x_{i+2})$, ..., $D(x_i, x_n)$), ..., ($D(x_{n-1}, x_n)$), which corresponds to the arithmetic series $n-1+n-2+...+1$. Thus for agglomerative hierarchical clustering using *symmetrical* distances the number of distance computations is $n(n-1)/2$.

We can view this calculation pictorially as a tree construction as shown in Figure 4.

If we perform the distance calculation at the first level of the tree then we can obtain bounds using the triangular inequality for **all** branches in the second level as to bound the distance between two points the distance between these points and another common point need

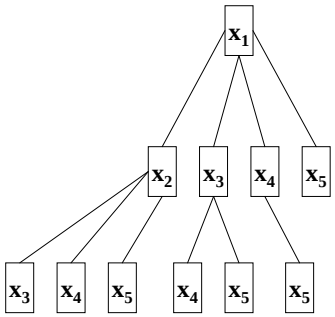

Figure 4: A Simple Illustration of How the Triangular Inequality Can Save Distance Computations

only be known. Thus in the best case there is only (n-1) distance computations instead of $kn(n-1)/2$.

5.2 Average Case Analysis for Using the γ constraint

However, it is highly unlikely that the best case situation will ever occur. We now focus on the average case analysis using the Markov inequality to determine the *expected* performance improvement which we later empirically verify. Let ρ be the average distance between any two instances in the data set to cluster.

The triangle inequality provides a lower bound which if exceeding γ will result in computational savings. Therefore, the proportion of times this occurs will increase as we build the tree from the bottom up. We can bound how often this occurs if we can express γ in terms of ρ, hence let $\gamma = c\rho$.

Recall that the general form of the Markov inequality is: $P(X = x \geq a) \leq \frac{E(X)}{a}$, where x is a single value of the continuous random variable X, a is a constant and $E(X)$ is the expected value of X. In our situation, X is distance between two points chosen at random, $E(X) = \rho$, a $= c\rho$. Therefore, at the lowest level of the tree ($k = n$) then the number of times the triangular inequality will save us computation time is $(n\rho)/(c\rho) = n/c$ indicating a saving of $1/c$. As the Markov inequality is a rather weak bound then in practice the saving may be substantially higher as we shall see in our empirical section. The computation saving that are obtained at the bottom of the tree are reflected at higher levels of the tree. When growing the entire tree we will save at least $n/c + (n-1)/c \ldots + 1/c$ time. This is an arithmetic sequence where the additive constant being $1/c$ and hence the total expected computations saved is at least $n/2(2/c + (n-1)/c) = (n^2 + n)/2c$. As the total computations for regular hierarchical clustering is $n(n-1)/2$, the computational saving is approximately $1/c$.

Consider the 150 instance IRIS data set (n=150) where the average distance (with attribute value ranges all being normalized to between 0 and 1) between two instances is 0.6 then $\rho = 0.6$. If we state that we do not wish to join clusters whose centroids are greater than 3.0 then $\gamma = 3.0 = 5\rho$. By not using the γ constraint

and the triangular inequality the total number of computations is (11175) and the number of computations that are saved is at least $(150^2 + 150)/10 = 2265$ and hence the saving is about 20%.

6. CONSTRAINT DRIVEN HIERARCHICAL CLUSTERING

6.1 Preliminaries

In agglomerative clustering, the dendrogram is constructed by starting with a certain number of clusters, and in each stage, merging two clusters into a single cluster. In the presence of constraints, the clustering that results after each merge must be feasible; that is, it must satisfy all the specified constraints. The merging process is continued until either the number of clusters has been reduced to one or no two clusters can be merged without violating one or more of the constraints. The focus of this section is on the development of an efficient algorithm for constraint-driven hierarchical clustering. We recall some definitions that capture the steps of agglomerative (and divisive) clustering.

DEFINITION 6.1. *Let S be a set containing two or more elements.*

*(a) A **partition** $\pi = \{X_1, X_2, \ldots, X_r\}$ of S is a collection of pairwise disjoint and non-empty subsets of S such that $\cup_{i=1}^{r} X_i = S$. Each subset X_i in π is called a **block**.*

*(b) Given two partitions $\pi_1 = \{X_1, X_2, \ldots, X_r\}$ and $\pi_2 = \{Y_1, Y_2, \ldots, Y_t\}$ of S, π_2 is a **refinement** of π_1 if for each block Y_j of π_2, there is a block X_p of π_1 such that $Y_j \subseteq X_p$. (In other words, π_2 is obtained from π_1 by splitting each block X_i of π_1 into one or more blocks.)*

*(c) Given two partitions $\pi_1 = \{X_1, X_2, \ldots, X_r\}$ and $\pi_2 = \{Y_1, Y_2, \ldots, Y_t\}$ of S, π_2 is a **coarsening** of π_1 if each block Y_j of π_2 is the union of one or more blocks of π_1. (In other words, π_2 is obtained from π_1 by merging one or more blocks of π_1 into a block of π_2.)*

It can be seen from the above definitions that successive stages of agglomerative clustering correspond to coarsening of the initial partition. Likewise, successive stages of divisive clustering correspond to refinements of the initial partition.

6.2 An Algorithm for Constraint-Driven Hierarchical Clustering

Throughout this section, we will assume that the constraint set includes all three types of constraints (ML, CL and ϵ) and that the distance function for the nodes is symmetric. Thus, a feasible clustering must satisfy all these constraints. The main result of this section is the following.

THEOREM 6.1. *Let S be a set of nodes with a symmetric distance between each pair of nodes. Suppose*

Input: Two feasible clusterings $C_1 = \{X_1, X_2, \ldots, X_{k_1}\}$ and $C_2 = \{Z_1, Z_2, \ldots, Z_{k_2}\}$ of a set of nodes S such that $k_1 < k_2$ and C_2 is a refinement of C_1.
Output: A feasible clustering $D = \{Y_1, Y_2, \ldots, Y_{k_1+1}\}$ of S with $k_1 + 1$ clusters such that D is a refinement of C_1 and a coarsening of C_2.
Algorithm:

1. Find a cluster X_i in C_1 such that two or more clusters of C_2 were merged to form X_i.
2. Let $Y_{i_1}, Y_{i_1}, \ldots, Y_{i_t}$ (where $t \geq 2$) be the clusters of C_2 that were merged to form X_i.
3. Construct an undirected graph $G_i(V_i, E_i)$ where V_i is in one-to-one correspondence with the clusters $Y_{i_1}, Y_{i_1}, \ldots, Y_{i_t}$. The edge $\{a, b\}$ occurs if at least one of a and b corresponds to a singleton cluster and the singleton node has an ϵ neighbor in the other cluster.
4. Construct the connected components (CCs) of G_i.
5. If G_i has two or more CCs, then form D by splitting X_i into two clusters X_i^1 and X_i^2 as follows: Let R_1 be one of the CCs of G_i. Let X_i^1 contain the clusters corresponding to the nodes of R_1 and let X_i^2 contain all the other clusters of X_i.
6. If G_i has only one CC, say R, do the following.
   (a) Find a spanning tree T of R.
   (b) Let a be a leaf of T. Form D by splitting X_i into two clusters X_i^1 and X_i^2 in the following manner: Let X_i^1 contain the cluster corresponding to node a and let X_i^2 contain all the other clusters of X_i.

Figure 5: Algorithm for Each Stage of Refining a Given Clustering

---

ConstrainedAgglomerative(S,ML,CL,ϵ, δ, γ) **returns** $Dendrogram_i$, $i = k_{min}$ ... k_{max}

1. if(CL $\neq \emptyset$) return *empty* //Use Figure 5 algorithm
2. $k_{min}, k_{max} = calculateBounds(S, ML, \epsilon, \delta, \gamma)$
3. **for** k=k_{max} to k_{min}
   $Dendrogram_k = \{C_1 ... C_k\}$
   d = IntelligentDistance($C_1 ... C_k$)
   l, m = closest legally joinable clusters
   $C_l = Join(C_l, C_m)$ $remove(C_m)$
   endloop

Figure 6: Constrained Agglomerative Clustering

there exist two feasible clusterings C_1 and C_2 of S with k_1 and k_2 clusters such that the following conditions hold: (a) $k_1 < k_2$ and (b) C_1 is a coarsening of C_2. Then, for each integer k, $k_1 \leq k \leq k_2$, there is a feasible clustering with k clusters. Moreover, given C_1 and C_2, a feasible clustering for each intermediate value k can be obtained in polynomial time.

Proof: We prove constructively that given C_1, a clustering D with $k_1 + 1$ clusters, where D is a refinement of C_1 and a coarsening of C_2, can be obtained in polynomial time. Clearly, by repeating this process, we can obtain all the intermediate stages in the dendrogram between C_1 and C_2. The steps of our algorithm are shown in Figure 5. The correctness of the algorithm is established through a sequence of claims.

CLAIM 6.1. *The clustering D produced by the algorithm in Figure 5 has $k_1 + 1$ clusters. Further, D is a refinement of C_1 and a coarsening of C_2.*

Proof: We first observe that Step 1 of the algorithm chooses a cluster X_i from C_1 such that X_i is formed by merging two or more clusters of C_2. Such a cluster X_i must exist since C_2 is a refinement of C_1 and C_2 has more clusters than C_1. The remaining steps of the algorithm form D by splitting cluster X_i into two clusters X_i^1 and X_i^2 and leaving the other clusters of C_1 intact. Thus, clustering D has exactly k_1+1 clusters. Also, the splitting of X_i is done in such a way that both X_i^1 and X_i^2 are formed by merging one or more clusters of C_2. Thus, D is a refinement of C_1 and a coarsening of C_2 as stated in the claim.

Before stating the next claim, we have simple graph theoretic definition. A **connected subgraph** $G'(V', E')$ of G_i consists of a subset $V' \subseteq V$ of vertices and a subset $E' \subseteq E$ of edges. Thus, G' has only one connected component (CC). Note that G' may consist of just a single vertex.

CLAIM 6.2. *Let $G'(V', E')$, be any connected subgraph of G_i. Let Y denote the cluster formed by merging all the clusters of C_2 that correspond to vertices in V'. Then, the ϵ-constraint is satisfied for Y.*

Proof: Consider any vertex $v \in V'$. Suppose the cluster Z_j in C_2 corresponding to v has two or more nodes of S. Then, since C_2 is feasible and the distances are symmetric, each node in Z_j has an ϵ-neighbor in Z_j. Therefore, we need to consider only the case where Z_j contains just one node of S. If V' contains only the node v, then Z_j is a singleton cluster and the ϵ-constraint trivially holds for Z_j. So, we may assume that V' has two or more nodes. Then, since G' is connected, node v has a neighbor, say w. By the construction of G_i, the node in the singleton cluster Z_j has an ϵ-neighbor in the cluster Z_p corresponding to node w of G'. Nodes in Z_p are also part of Y. Therefore, the ϵ-constraint is satisfied for the singleton node in cluster Z_j.

CLAIM 6.3. *The clustering D produced by the algorithm in Figure 5 is feasible.*

Proof: Since C_1 is feasible and D is a refinement of C_1, D satisfies all the CL-constraints. Also, since C_2 is feasible and D is a coarsening of C_2, D satisfies all the ML-constraints. Therefore, we need only show that D satisfies the ϵ-constraint. Note also that D was obtained by splitting one cluster X_i of C_1 into two clusters X_i^1 and X_i^2 and leaving the other clusters of C_1 intact. Therefore, it suffices to show that the ϵ-constraint is satisfied by X_i^1 and X_i^2. To prove this, we consider two cases depending on how X_i^1 was chosen.

Case 1: X_i^1 was chosen in Step 5 of the algorithm.

In this case, the graph G_i has two or more CCs. From Claim 6.2, we know that the ϵ-constraint is satisfied for each CC of G_i. Since X_i^1 and X_i^2 are both made up of CCs of G_i, it follows that they satisfy the ϵ-constraint. This completes the proof for Case 1.

Case 2: X_i^1 was chosen in Step 6 of the algorithm.

In this case, the graph G_i has only one CC. Let x denote the leaf vertex from which X_i^1 was formed. Since x by itself is a (trivially) connected subgraph of G_i, by Claim 6.2, the ϵ-constraint is satisfied for X_i^1. Since x is a leaf vertex in T and removing a leaf does not disconnect a tree, the remaining subgraph T' is also connected. Cluster X_i^2 is formed by merging the clusters of C_2 corresponding to the vertices of T'. Thus, again by Claim 6.2, the ϵ-constraint is also satisfied for X_i^2. This completes the proof for Case 2.

It can be verified that given C_1 and C_2, the algorithm shown in Figure 5 runs in polynomial time. This completes the proof of the theorem. □

7. THE ALGORITHMS

The algorithm for traditional agglomerative clustering with constraints is shown in Figure 6.

The algorithm for constrain Driven Agglomerative clustering is shown in Figure 5.

8. EMPIRICAL RESULTS

In this section we present the empirical results of using the algorithm derived in this paper. We present the work on extensions to existing agglomerative clustering algorithms and illustrate the efficiency improvement over unconstrained hierarchical clustering and preliminary results indicating that the quality of dendrogram improves. We also verify the correctness of our average case performance bound. Since our constraint driven agglomerative clustering algorithm to our knowledge is the first of its kind, we could report no empirical results other than to show that the algorithm is correct which we have formally shown in Section 6.

8.1 Results for Extensions to Agglomerative Clustering Algorithms

In this sub-section we report results for six UCI data sets to verify the efficiency improvement and expected case performance bound. We will begin by investigating must and cannot link constraints. For each data set we clustered all instances but removed the labels from 90% of the data and used the remaining 10 % to provide an equal number of must-link and cannot-link constraints. The must-link constraints were between instances with

Data Set	Unconstrained	Constrained
Iris	11,175	9,996
Breast	243,951	212,130
Digit (3 vs 8)	1,999,000	1,800,050
Pima	294,528	260,793
Census	1,173,289,461	998,892,588
Sick	397,386	352,619.68

Table 3: Constructing a Restricted Dendrogram Using the Bounds from Section 3 (Mean Number of Join/Merger Ops.)

Data Set	Unconstrained	Constrained
Iris	3.2	2.7
Breast	8.0	7.3
Digit (3 vs 8)	17.1	15.2
Pima	9.8	8.1
Census	26.3	22.3
Sick	17.0	15.6

Table 4: Average Distortion per Instance of Entire Dendrogram

the same class label and cannot-link constraints between instances of differing class labels. We repeated this process twenty times and reported the average of performance measures. All instances with missing values were removed as hierarchical clustering algorithms do not easily handle such instances. Furthermore, all non-continuous columns were removed as there is no standard distance measure for discrete columns.

Table 3 illustrates the improvement due to the creation of a bounded/pruned dendrogram. We see that even a small number of must-link constraints effectively reduces the number of instances to cluster and this reduction will increase as the number of constraints increases.

Tables 4 and 5 illustrate the quality improvement that the must-link and cannot-link constraints provide. Note, we compare the dendrograms for k values between k_{min} and k_{max}. For each corresponding level in the unconstrained and constrained dendrogram we measure the average distortion $(1/n * \sum_{i=1}^{n} \text{Distance}(x_i - C_{f(x_i)})$, where $f(x_i)$ returns the index of the closest cluster to x_i) and present the average over all levels. It is important to note that we are not claiming that agglomerative clustering has the distortion as an objective function, rather that it is a good measure of cluster quality. We see that the distortion improvement is typically in the order of 15%. We also see that the average percentage purity of the clustering solution as measured by the class labels improves. We believe these improvement are due to the following. When many pairs of clusters have similar short distances the must-link constraints guide the algorithm to a better join. This type of improvement occurs at the bottom of the dendrogram. Conversely towards the top of the dendrogram the cannot-link constraints rule out ill-advised joins. However, this preliminary explanation requires further investigation which we intend to address in the future. In particular a study of the most informative constraints for hierarchical clustering remains an open question, though promising preliminary work for the area of non-hierarchical clustering exists [2].

Data Set	Unconstrained	Constrained
Iris	58%	66%
Breast	53%	59%
Digit (3 vs 8)	35%	45%
Pima	61%	68%
Census	56%	61%
Sick	50 %	59%

Table 5: Average Percentage Cluster Purity of Entire Dendrogram

Data Set	Unconstrained	Constrained
Iris	11,175	8,929
Breast	243,951	163,580
Digit (3 vs 8)	1,999,000	1,585,707
Pima	294,528	199,264
Census	1,173,289,461	786,826,575
Sick	397,386	289,751

Table 6: The Efficiency of an Unrestricted Dendrogram Using the Geometric Reasoning Approach from Section 5 (Mean Number of Join/Merger Ops.)

Data Set	Unconstrained	Constrained
Iris	11,175	3,275
Breast	243,951	59,726
Digit (3 vs 8)	1,999,000	990,118
Pima	294,528	61,381
Census	1,173,289,461	563,034,601
Sick	397,386	159,801

Table 7: Cluster Level δ Constraint: The Mean Number of Joins for an Unrestricted Dendrogram

We now show that the γ constraint can be further used to improve efficiency in addition to Table 3. Table 6 illustrates the improvement that using a γ constraint equal to five times the average pairwise instance distance. We see that the average improvement is consistent with the average case bound derived in section 5 but as expected can produce significantly better than 20% improvement as the Markov inequality is a weak bound.

Finally, we use the cluster level δ constraint with an arbitrary value to illustrate the great computational savings that such constraints offer. Our earlier work [4] explored ϵ and δ constraints to provide background knowledge towards the "type" of clusters we wish to

find. In that paper we explored their use with the Aibo robot to find objects in images that were more than 1 foot apart as the Aibo can only navigate between such objects. For these UCI data sets no such background knowledge exists and how to set these constraint values for non-spatial data remains an active research area, hence we must test these constraints with arbitrary values. We set δ equal to 10 times the average distance between a pair of points and re-run our algorithms. Such a constraint will generate hundreds even thousands of must-link constraints that can greatly influence the clustering results and save efficiency as shown in Table 7. We see that the minimum improvement was 50% (for Census) and nearly 80% for Pima.

9. CONCLUSION

Non-hierarchical/partitional clustering algorithms such as k-means are used extensively in data mining due to their computational efficiency at clustering large data sets. However, they are limited in that the number of clusters must be stated apriori. Hierarchical agglomerative clustering algorithms instead present a dendrogram that allows the user to select a value of k and are desirable in domains where clusters within clusters occur. However, non-hierarchical clustering algorithms' complexity is typically linear in the number of instances (n) while hierarchical algorithms are typically $O(n^2)$.

In this paper we explored the use of instance and cluster level constraints to improve the efficiency of hierarchical clustering algorithms. Our previous work [4] studied the complexity of clustering for a given value of k under four types of constraints (must-link, cannot-link, ϵ and δ). We extend this work for unbounded k and find that clustering under all four types of constraints is **NP**-complete and hence creating a dendrogram that satisfies all constraints at each level is intractable. We also derived bounds in which all feasible solutions exist that allows us to create a restricted dendrogram.

An unexpected interesting result was that whenever using cannot-link constraints (either by themselves or in combination) traditional agglomerative clustering algorithms may yield dead-end or irreducible cluster solutions. For this case we create a constraint driven hierarchical clustering algorithm that is guaranteed to create a dendrogram that contains a complete range of feasible solutions.

We introduced a fifth constraint type, the γ constraint, that allows the performing geometric reasoning via the triangle inequality to save computation time. The use of the γ constraint in our experiments allows a computation saving of at least 20% and verified the correctness of our bound.

Our experimental results indicate that small amounts of labeled data can yield small but significant efficiency improvement via constructing a restricted dendrogram. A primary benefit of using labeled data is the improvement in the quality of the resultant dendrogram with respect to cluster purity and "tightness" (as measured by the distortion).

We find that our cluster level constraints which can apply to many data points offer the potential for large efficiency improvement. The δ and ϵ constraints can be efficiently translated into conjunctions and disjunctions of must-link constraints for ease of implementation. Just using the δ constraint provides saving of between two and four fold though how to set a value for this constraint remains an active research area.

10. ACKNOWLEDGMENTS

We would like to thank the anonymous SIAM Data Mining Conference reviewer who pointed out our earlier results [4] are applicable beyond non-hierarchical clustering.

11. REFERENCES

[1] S. Basu, A. Banerjee, and R. Mooney, Semi-supervised Clustering by Seeding, 19^{th} *ICML*, 2002.

[2] S. Basu, M. Bilenko and R. J. Mooney, Active Semi-Supervision for Pairwise Constrained Clustering, 4^{th} *SIAM Data Mining Conf.*. 2004.

[3] P. Bradley, U. Fayyad, and C. Reina, "Scaling Clustering Algorithms to Large Databases", 4^{th} *ACM KDD Conference*. 1998.

[4] I. Davidson and S. S. Ravi, "Clustering with Constraints and the k-Means Algorithm", 5^{th} *SIAM Data Mining Conf.* 2005.

[5] Y. Fua, M. Ward, E. Rundensteiner, Hierarchical parallel coordinates for exploration of large datasets, IEEE Viz. 1999.

[6] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman and Co., 1979.

[7] M. Garey, D. Johnson and H. Witsenhausen, "The complexity of the generalized Lloyd-Max problem", *IEEE Trans. Information Theory*, Vol. 28,2, 1982.

[8] D. Klein, S. D. Kamvar and C. D. Manning, "From Instance-Level Constraints to Space-Level Constraints: Making the Most of Prior Knowledge in Data Clustering", 19^{th} *ICML Conf.* 2002.

[9] A. McCallum, K. Nigam and L. Ungar. Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching, 6^{th} *ACM KDD Conf.*, 2000.

[10] T. J. Schafer, "The Complexity of Satisfiability Problems", *Proc.* 10^{th} *ACM Symp. Theory of Computing* (STOC'1978), 1978.

[11] K. Wagstaff and C. Cardie, "Clustering with Instance-Level Constraints", 17^{th} *ICML*, 2000.

[12] D. B. West, *Introduction to Graph Theory*, Second Edition, Prentice-Hall, 2001.

[13] K. Yang, R. Yang and M. Kafatos, "A Feasible Method to Find Areas with Constraints Using Hierarchical Depth-First Clustering", *Scientific and Statistical Database Management Conf.*, 2001.

[14] O. R. Zaiane, A. Foss, C. Lee, W. Wang, On Data Clustering Analysis: Scalability, Constraints and Validation, 6^{th} *PAKDD Conf.*, 2000.

[15] Y. Zho & G. Karypis, Hierarchical Clustering Algorithms for Document Datasets, University of Minnesota, Comp. Sci. Dept. *TR03-027*.

12. APPENDIX A

Proof of Theorem 2.2

Proof: It is easy to see that FHC is in **NP** since one can guess a partition of S into clusters and verify that the partition satisfies all the given constraints. We establish **NP**-hardness by a reduction from OPL.

Given an instance I of the OPL problem consisting of variable set X and clause set Y, we create an instance I' of the FHC problem as follows.

We first describe the nodes in the instance I'. For each Boolean variable x_i, we create a node v_i, $1 \leq i \leq n$. Let $V = \{v_1, v_2, \ldots, v_n\}$. For each clause $y_j = \{x_{j_1}, x_{j_2}, x_{j_3}\}$, $1 \leq j \leq m$, we do the following:

(a) We create a set $A_j = \{w_{j_1}, w_{j_2}, w_{j_3}\}$ containing three nodes. (The reader will find it convenient to think of nodes w_{j_1}, w_{j_2} and w_{j_3} as corresponding to the variables x_{j_1}, x_{j_2} and x_{j_3} respectively.) We refer to the three nodes in A_j as the **primary nodes** associated with clause y_j.

(b) We create six additional nodes denoted by a_j^1, a_j^2, b_j^1, b_j^2, c_j^1 and c_j^2. We refer to these six nodes as the **secondary nodes** associated with clause y_j. For convenience, we let $B_j = \{a_j^1, b_j^1, c_j^1\}$. Also, we refer to a_j^2, b_j^2 and a_j^2 as the **twin node** of a_j^1, b_j^1 and c_j^1 respectively.

Thus, the construction creates a total of $n + 9m$ nodes. The distances between these nodes are chosen in the following manner, by considering each clause y_j, $1 \leq j \leq n$.

(a) Let the primary nodes w_{j_1}, w_{j_2} and w_{j_3} associated with clause y_j correspond to Boolean variables x_p, x_q and x_r respectively. Then $d(v_p, w_{j_1}) = d(v_q, w_{j_2}) = d(v_r, w_{j_3}) = 1$.

(b) The distances among the primary and secondary nodes associated with y_j are chosen as follows.

(i) $d(a_j^1, a_j^2) = d(b_j^1, b_j^2) = d(c_j^1, c_j^2) = 1$.
(ii) $d(a_j^1, w_{j_1}) = d(a_j^1, w_{j_2}) = 1$.
(iii) $d(b_j^1, w_{j_2}) = d(b_j^1, w_{j_3}) = 1$.
(iv) $d(c_j^1, w_{j_1}) = d(c_j^1, w_{j_3}) = 1$.

For each pair of nodes which are not covered by cases (a) or (b), the distance is set to 2. The constraints are chosen as follows.

(a) **ML-constraints:** For each j, $1 \leq j \leq n$, there are the following ML-constraints: $\{w_{j_1}, w_{j_2}\}$, $\{w_{j_1}, w_{j_3}\}$, $\{w_{j_2}, w_{j_3}\}$, $\{a_j^1, a_j^2\}$, $\{b_j^1, b_j^2\}$ and $\{c_j^1, c_j^2\}$.

(b) **CL-constraints:**

(i) For each pair of nodes v_p and v_q, there is a CL-constraint $\{v_p, v_q\}$.
(ii) For each j, $1 \leq j \leq m$, there are three CL-constraints, namely $\{a_j^1, b_j^1\}$, $\{a_j^1, c_j^1\}$ and $\{b_j^1, c_j^1\}$.

(c) **ϵ-constraint:** The value of ϵ is set to 1.

This completes the construction of the FHC instance I'. It can be verified that the construction can be carried out in polynomial time. We now prove that the FHC instance I' has a solution if and only if the OPL instance I has a solution.

If part: Suppose the OPL instance I has a solution. Let this solution set variables x_{i_1}, x_{i_2}, ..., x_{i_r} to true and the rest of the variables to false. A solution to the FHC instance I' is obtained as follows.

(a) Recall that $V = \{v_1, v_2, \ldots, v_n\}$. For each node v in $V - \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$, create the singleton cluster $\{v\}$.

(b) For each node v_{i_q} corresponding to variable x_{i_q}, $1 \leq q \leq r$, we create the following clusters. Let y_{j_1}, y_{j_2}, ..., y_{j_p} be the clauses in which variable x_{i_k} occurs.

(i) Create one cluster containing the following nodes: node v_{i_q}, the three primary nodes corresponding to each clause y_{j_l}, $1 \leq l \leq p$, and one pair of secondary nodes corresponding to cluster y_{j_l} chosen as follows. Since x_{i_q} satisfies clause y_{j_l}, one primary node corresponding to clause y_{j_l} has v_{i_q} as its ϵ-neighbor. By our construction, exactly one of the secondary nodes corresponding to y_{j_l}, say $a_{j_l}^1$, is an ϵ-neighbor of the other two primary nodes corresponding to y_{j_l}. The cluster containing v_{i_q} includes $a_{j_l}^1$ and its twin node $a_{j_l}^2$.

(ii) For each clause y_{j_l}, $1 \leq l \leq p$, the cluster formed in (i) does not include two secondary nodes and their twins. Each secondary node and its twin is put in a separate cluster. (Thus, this step creates $2p$ additional clusters.)

We claim that the clusters created above satisfy all the constraints. To see this, we note the following.

(a) ML-constraints are satisfied because of the following: the three primary nodes corresponding to each clause are in the same cluster and each secondary node and its twin are in the same cluster.

(b) CL-constraints are satisfied because of the following: each node corresponding to a variable appears in a separate cluster and exactly one secondary node (along with its twin) corresponding to a variable appears in the cluster containing the three primary nodes corresponding to the same variable.

(c) ϵ-constraints are satisfied because of the following. (Note that singleton clusters can be ignored here.)

(i) Consider each non-singleton cluster containing a node v_i corresponding to Boolean variable x_i. Let y_j be a clause in which x_i appears. Node v_i serves as the ϵ-neighbor for one of the primary nodes corresponding to y_j in the cluster. One of the secondary nodes, corresponding to y_j, say a_j^1, serves as the ϵ-neighbor for the other two primary nodes corresponding to y_j as well as its twin node a_j^2.

(ii) The other non-singleton clusters contain a secondary node and its twin. These two nodes are ϵ-neighbors of each other.

Thus, we have a feasible clustering for the instance I'.

Only if part: Suppose the FHC instance I' has a solution. We can construct a solution to the OPL instance I as follows. We begin with a claim.

CLAIM 12.1. *Consider any clause y_j. Recall that $A_j = \{w_{j_1}, w_{j_2}, w_{j_3}\}$ denotes the set of three primary nodes corresponding to y_j. Also recall that $V = \{v_1, v_2, \ldots, v_n\}$. In any feasible solution to I', the nodes in A_j must be in the same cluster, and that cluster must also contain exactly one node from V.*

Proof of claim: The three nodes in A_j must be in the same cluster because of the ML-constraints involving them. The cluster containing these three nodes may include only one node from the set $B_j = \{a_j^1, b_j^1, c_j^1\}$ because of the CL-constraints among these nodes. Each node in B_j has exactly two of the nodes in A_j as its ϵ-neighbor. Note also that the only ϵ-neighbors of each node in A_j are two of the nodes in B_j and one of the nodes from the set $V = \{v_1, v_2, \ldots, v_n\}$. Thus, to satisfy the ϵ-constraint for the remaining node in A_j, the cluster containing the nodes in A_j must also contain at least one node from V. However, because of the CL-constraints between each pair of nodes in V, the cluster containing the nodes in A_j must have exactly one node from V. This completes the proof of the claim.

CLAIM 12.2. *Consider the following truth assignment to the variables in the instance I: Set variable x_i to true if and only if node v_i appears in a non-singleton cluster. This truth assignment sets exactly one literal in each clause y_j to true.*

Proof of Claim: From Claim 12.1, we know that the cluster containing the three primary nodes corresponding to y_j contains exactly one node from the set $V = \{v_1, v_2, \ldots, v_n\}$. Let that node be v_i. Thus, v_i is the ϵ-neighbor for one of the primary nodes corresponding to y_j. By our construction, this means that variable x_i appears in clause y_j. Since x_i is set to true, clause y_j is satisfied. Also note that the cluster containing v_i does not include any other node from V. Thus, the chosen truth assignment satisfies exactly one of the literals in y_j.

From Claim 12.2, it follows that the instance I has a solution, and this completes the proof of Theorem 2.2. □