Deterministic pivoting algorithms for constrained ranking and clustering problems

Anke van Zuylen[*] Rajneesh Hegde[†] Kamal Jain[‡] David P. Williamson[§]

Abstract

We introduce new problems of finding minimum-cost rankings and clusterings which must be consistent with certain constraints (e.g. an input partial order in the case of ranking problems); we give deterministic approximation algorithms for these problems. Randomized approximation algorithms for unconstrained versions of these problems were given by Ailon, Charikar, and Newman [2] and by Ailon and Charikar [1]. Finding deterministic approximation algorithms for these problems answers an open question of Ailon et al. [2].

In particular, we give deterministic algorithms for constrained weighted feedback arc set in tournaments, constrained correlation clustering, and constrained hierarchical clustering related to finding good ultrametrics. Our algorithms follow the paradigm of Ailon et al. [2] of choosing a particular vertex as a pivot and partitioning the graph according to the pivot; unlike their algorithms, we do not choose the pivot randomly but rather use an LP relaxation to choose a good pivot deterministically. Additionally, the use of the LP relaxation allows us to impose constraints easily and analyze the results. In several cases we are able to find approximation factors for the constrained problems that improve on the factors they obtained for the unconstrained cases. We also give a combinatorial algorithm for constrained weighted feedback arc set in tournaments with weights satisfying probability constraints. This algorithm improves on the best known factor given by deterministic combinatorial algorithms for the unconstrained case.

1 Introduction

In this paper, we consider several problems related to the constrained aggregation of inconsistent information. The problems we consider can be divided into two categories: ranking and clustering problems. In the ranking problems, we are given a set of objects and (possibly contradictory) information about the relative ranking of each pair of objects, along with a partial order. We wish to find a ranking of the objects consistent with the partial order that minimizes the sum of pairwise discrepancies with the input information (this optimality criterion is due to Kemeny [7]). In rank aggregation, for example, we are given k rankings of the same n objects, and want to combine these into one ranking that minimizes the sum over all pairs i, j such that i is ordered before j of the number of input rankings that ordered j before i. The partial order constraint allows us to specify a priori information about the output ordering. In the clustering problems, we wish to partition a set of objects into clusters, and are given (again, possibly contradictory) information about the relation between any pair of objects. Consensus clustering is similar to rank aggregation in that we want to combine multiple input clusterings into a single clustering. In correlation clustering, the input information consists of a '+' indicating that a pair would prefer to be clustered together or '−' indicating that a pair would prefer to be separated. We also can specify consistent information that pairs of objects must be clustered together or must not be clustered together in the final clustering; this constraint plays the same role as the input partial order in the ranking problems. Hierarchical clustering is a generalization of correlation clustering. We want to find a nested clustering of the elements, and are given information for every pair of elements and for every level of the clustering. This problem can also be viewed as finding an ultrametric that is close to given pairwise distances. Again, we can specify constraints about pairs that must be clustered together or must not be clustered together at each level. These kinds of problems arise in many contexts, for example, in building meta-search engines for Web search, where we want to get robust rankings that are not sensitive to the various shortcomings and biases of individual search engines by combining the rankings of the individual search engines [5]; constraints on the output ordering may reflect a priori beliefs about the output ordering (e.g. the top-level page of a website should be ordered before subpages of the site). In an example from biology, the goal is to find classifications of genes by integrating data from different experiments [6]; again, the input constraints can reflect prior beliefs about the output classification.

We will model these problems as graphs, where each vertex represents an object, and the information

[*]School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14853. Email: avz2@cornell.edu. Supported by NSF grant CCF-0514628.

[†]Microsoft Corporation, Redmond, WA. Email: rajneeshhegde@gmail.com.

[‡]Microsoft Research, Redmond, WA 98052. Email: kamalj@microsoft.com.

[§]School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14853. Email: dpw@cs.cornell.edu. Supported by NSF grant CCF-0514628.

relating objects i and j is represented by nonnegative weights on the edge i, j. In the case of correlation or consensus clustering problems, we define weights w_{ij}^+ and w_{ij}^-. In consensus clustering, w_{ij}^+ (w_{ij}^-) gives the fraction of input clusterings that put i and j in the same cluster (in separate clusters). Correlation clustering can be modeled as a 0/1 weighted case. A set of pairs of vertices that must or must not be clustered together is part of the input. The goal of these problems is to minimize the sum of w_{ij}^- over all i, j in the same cluster plus the sum of w_{ij}^+ over all i, j that are not in the same cluster, subject to the given constraints. In the case of hierarchical clustering, we want to find a correlation clustering at each level, with the additional constraint that the clustering at level $\ell - 1$ is a refinement of the clustering at level ℓ. We also specify constraints for each level that specify pairs that should be clustered together or apart (respectively) at that level. For ranking problems, we will have a weight w_{ij} and a weight w_{ji} and wish to minimize the sum of w_{ji} over all pairs i, j such that i is ranked before j. For example in the case of rank aggregation, w_{ij} gives the fraction of input rankings that rank i before j. We are also given an input partial order P. We will refer to this problem as the *constrained weighted feedback arc set problem on tournaments*.

Ailon, Charikar, and Newman [2] and Ailon and Charikar [1] recently considered unconstrained versions of these problems in which any ordering/clustering is a feasible output. As in their papers, we will give approximation algorithms for the case when the weights satisfy probability constraints (i.e. $w_{ij} + w_{ji} = 1$ or $w_{ij}^+ + w_{ij}^- = 1$) and/or the triangle inequality (i.e. $w_{ij} + w_{jk} \geq w_{ik}$ or $w_{ij}^- + w_{jk}^- \geq w_{ik}^-$ and $w_{ij}^- + w_{jk}^+ \geq w_{ik}^+$). Note that the weights in all applications mentioned above satisfy either probability constraints, or both probability and triangle inequality constraints. Ailon et al. [2] give algorithms for the unconstrained problems that all fall into one general framework. The algorithm recursively generates a solution, by choosing a random vertex as "pivot" and ordering all other vertices with respect to the pivot vertex according to some criteria. In the first algorithm they give for the ranking problem, a vertex j is placed on the left of the pivot k if $w_{jk} \geq w_{kj}$ or on the right otherwise. Next, the algorithm recurses on the two instances induced by the vertices on each side. In the case of a clustering problem, a vertex j is placed in the same cluster as the pivot vertex k if $w_{jk}^+ \geq w_{jk}^-$. The algorithm recurses on the instance induced by the vertices that are not placed in the same cluster as the pivot vertex.

We answer an open question of Ailon et al. [2] by giving deterministic versions of their algorithms. We will show that by solving an LP relaxation of the problem to be solved, we can deterministically choose a good pivot. Furthermore, our algorithms extend to our new, constrained variants of their problems. In several cases, we are able to obtain improved guarantees compared to the pivoting algorithms by Ailon et al. [2] for the unconstrained problems; in remaining cases, our guarantees are the same as theirs for the unconstrained problems. Although it is possible to use our method to give a deterministic version of the algorithm from [1] for hierarchical clustering, we give a simpler, top-down algorithm which achieves the same approximation guarantee. The algorithm calls a slight variant of the algorithm for constrained correlation clustering to find the top level clustering, and then recurses on all clusters found. Our analysis of this algorithm is simpler, too, since the performance guarantee of [1] uses two different LPs in its analysis, and we use a single LP for both the algorithm and analysis.

In related work, Coppersmith, Fleischer and Rudra [4] have also given a deterministic algorithm for the unconstrained feedback arc set problem in tournaments where the weights obey the probability constraints. They show that sorting by indegree is a 5-approximation algorithm. We give another combinatorial algorithm that first removes all directed triangles, and then uses a simple recursive algorithm by Chudnovsky, Seymour and Sullivan [3] to generate a ranking. We show that this gives a factor 4 in the unconstrained case. Furthermore, we generalize these results to give a combinatorial 6-approximation algorithm for the constrained case.

We also answer another open question of Ailon et al. [2]: whether there exists an approximation algorithm for the unconstrained weighted correlation clustering when the weights satisfy only the triangle inequality and not probability constraints. It seems appropriate in the case of weighted correlation clustering to have the following two inequalities: (1) $w_{ik}^- \leq w_{ij}^- + w_{jk}^-$ and (2) $w_{ik}^+ \leq w_{ij}^- + w_{jk}^+$, which say (1) the cost of clustering i and k together (w_{ik}^-) cannot be more than the cost of clustering i and j together and clustering j and k together ($w_{ij}^- + w_{jk}^-$), and (2) the cost of separating i and k (w_{ik}^+) cannot be more than the cost of clustering i and j together and separating j and k ($w_{ij}^- + w_{jk}^+$). We show that under these assumptions on the weights our algorithm yields a 2-approximation. Ailon et al. [2] only assume the first type of constraints.

Ailon et al. [2] also propose a way to use pivoting for a randomized rounding algorithm. The (expected) approximation guarantee of this LP rounding algorithm is better than ours in the case when the weights satisfy only probability constraints, and we note that our algorithm is not faster than their LP rounding algorithm, as

	Ranking				Clustering		
	ours (cons)	ACN [2] (unc)	ACN-LP [2] (unc)	CFR [4] (unc)	ours (cons)	ACN [2] (unc)	ACN-LP [2] (unc)
Probability cons.	3 (unc:4,cons:6)	5	$\frac{5}{2}$	5	3	5	$\frac{5}{2}$
Triangle ineq.	2	3	-		2	-	-
Prob.cons.+Triangle ineq.	2	2	2		2	2	2
Aggregation	2	$\frac{11}{7}$	$\frac{4}{3}$		2	$\frac{11}{7}$	$\frac{4}{3}$

Table 1: The first column under each heading summarizes the results in this paper except hierarchical clustering; (cons) denotes result for constrained problems, (unc) unconstrained problems. The next two columns give results for unconstrained problems, given by randomized pivoting (ACN) and randomized LP rounding (ACN-LP). The result in column (CFR) is obtained by a deterministic combinatorial algorithm. The approximation guarantees for our combinatorial algorithm for unconstrained and constrained ranking with probability constraints are given in parentheses. The last row gives the results when taking the best of the algorithm generated solution and a random input permutation/clustering.

we solve the same LP relaxation. Finally, Ailon et al. [2] show that better (< 2) performance guarantees can be achieved for the unconstrained case when the weights are a convex combination of actual rankings or clusterings, by taking the best of their pivoting algorithm and picking a random permutation/clustering from the input permutations/clusterings. We have not been able to prove a similar result for our deterministic algorithm, even in the unconstrained case.

2 Constrained Weighted Minimum Feedback Arc Set in Tournaments

Given a set of vertices V, nonnegative weights w_{ij} and w_{ji} for each pair of vertices i and j, and a partial order (V, P), we want to find a linear extension of P, i.e. a permutation π such that if $(i,j) \in P$ then π ranks i before j, that minimizes the weight of the backward arcs, i.e. the sum over all i,j such that i is ranked before j of w_{ji}.

We will give an approximation algorithm for the case when the weights satisfy *probability constraints*, i.e. for any pair of vertices i,j, $w_{ij}+w_{ji} = 1$, or the *triangle inequality*, i.e. for any triplet i,j,k, $w_{ij} + w_{jk} \geq w_{ik}$.

If we let $x_{ij} = 1$ denote that i is ranked before j, then any feasible ranking satisfies $x_{ij} + x_{ji} = 1$ and $x_{ij} + x_{jk} + x_{ki} \geq 1$ (since if $x_{ij} + x_{jk} + x_{ki} = 0$, then j is ranked before i, k is ranked before j but i is ranked before k, which is not possible). Hence the following linear program gives a lower bound on the minimum weight feedback arc set:

$$\begin{array}{llll} & \min & \displaystyle\sum_{i<j} \Big(x_{ij} w_{ji} + x_{ji} w_{ij} \Big) & \\ & \text{s.t.} & x_{ij} + x_{jk} + x_{ki} \geq 1 & \text{for all distinct } i,j,k \\ (LP) & & x_{ij} + x_{ji} = 1 & \text{for all } i \neq j \\ & & x_{ij} = 1 & \text{for all } (i,j) \in P \\ & & x_{ij} \geq 0 & \text{for all } i \neq j \end{array}$$

A similar linear program was also used by Ailon et al. [2] for the unconstrained case, where $P = \emptyset$.

Given an optimal solution x to (LP) for instance (V, w, P), we form a tournament $G = (V, A)$, where $(i,j) \in A$ only if $x_{ij} \geq \frac{1}{2}$. We will break ties in such a way as to ensure that there are no "triangles" containing an arc in P, i.e. there is no $(i,j),(j,k),(k,i) \in A$ such that $(i,j) \in P$. We will use the optimal solution x to (LP) to find a vertex to pivot on, and given a pivot vertex k, we will put vertex j to the left or right of k depending on whether $(j,k) \in A$ or $(k,j) \in A$. As in Ailon et al. [2], we then recurse on the set of vertices to the left and right of k. In the recursive calls, we do not resolve the LP, but use the optimal solution x to the LP on the complete instance and the corresponding tournament $G = (V, A)$.

Note that the only way a pair of vertices i,j with $(i,j) \in A$ will have j before i in the ranking that is returned by the algorithm, would be if i and j are in the same recursive call, and a pivot k is chosen such that $(k,i) \in A, (j,k) \in A$. In other words, $(i,j),(j,k)$ and (k,i) form a triangle in A. Hence as long as we ensure that there are no triangles in A that contain an arc in P, then our algorithm returns a ranking that is a linear extension of P.

Obviously, for pairs $\{j,k\}$ where k is the pivot, the cost we incur is at most twice the cost for $\{j,k\}$ in the optimal solution to (LP). However, if k is the pivot vertex, then for pairs (j,i) that are in a triangle with k in A, i.e. pairs such that $(i,k),(k,j)$ and $(j,i) \in A$, the algorithm orders i before j, even though $(j,i) \in A$, or $x_{ij} \leq \frac{1}{2}$. Let $T_k(A)$ denote the set of pairs (j,i) that are in a triangle with k, so $T_k(A) = \{(j,i) | (i,k),(k,j),(j,i) \in A\}$. To bound the cost for the pairs in $T_k(A)$, we choose a pivot that minimizes the ratio of the cost incurred by the algorithm for these pairs and the cost for these pairs in the optimal solution to (LP).

In the following, for $V' \subseteq V$ let $A_{V'} = \{(i,j) \in A | i \in V', j \in V'\}$, $w_{V'} = \{w_{ij} | i \in V', j \in V'\}$, $x_{V'} = \{x_{ij} | i \in V', j \in V'\}$, and let $c_{ij} = x_{ij}w_{ji} + x_{ji}w_{ij}$. We give our algorithm, FAS-Pivot, in Figure 1.

Constrained-FAS(V, w, P)

```
Let $x$ be an optimal solution to $(LP)$ on instance $(V, w, P)$.
Label the nodes in $V$ such that $label(i) \neq label(j)$ for $i \neq j$,
    and $(i,j) \in P \Rightarrow label(i) < label(j)$. [1]
Let $A = \{(i,j) : x_{ij} > \frac{1}{2}\} \cup$
    $\{(i,j) : x_{ij} = \frac{1}{2}$ and $label(i) < label(j)\}$.
Return FAS-Pivot$(V, A, w, x)$.
```

FAS-Pivot(V, A, w, x)

```
Pick pivot $k \in V$ minimizing $\frac{\sum_{(j,i)\in T_k(A)} w_{ji}}{\sum_{(j,i)\in T_k(A)} c_{ji}}$.
Set $V_L \leftarrow \emptyset, V_R \leftarrow \emptyset$.
For all $j \in V \backslash \{k\}$
    If $(j,k) \in A$ then $V_L \leftarrow V_L \cup \{j\}$, else $V_R \leftarrow V_R \cup \{j\}$.
Let $Ordered(V_L)$ = FAS-Pivot $(V_L, A_{V_L}, w_{V_L}, x_{V_L})$.
Let $Ordered(V_R)$ = FAS-Pivot$(V_R, A_{V_R}, w_{V_R}, x_{V_R})$.
Return $Ordered(V_L), k, Ordered(V_R)$.
```

[1] This can be done recursively by taking a vertex that has no incoming arcs in P, labeling it 0, deleting this vertex and all arcs in P that are adjacent to it, increasing the label by 1, and recursing.

Figure 1: Our algorithm for constrained weighted feedback arc set on tournaments.

THEOREM 2.1. *Constrained-FAS is a 3 (2)-approximation algorithm for the constrained weighted minimum feedback arc set problem on tournaments when the weights satisfy the probability constraints (triangle inequality).*

Proof. We first note that the tournament A does not have any triangles that contain an arc in P: Suppose $(i,j) \in P$ and $(i,j), (j,k), (k,i)$ is a triangle in A. If $(i,j) \in P$, then $x_{ij} = 1$, and if $(j,k), (k,i)$ are in A, then we must have $x_{jk} \geq \frac{1}{2}, x_{ki} \geq \frac{1}{2}$. But by the first set of constraints of (LP), $x_{ik} + x_{kj} \geq 1 - x_{ji} = 1$, or $1 - x_{ki} + 1 - x_{jk} \geq 1$, hence x_{ki} and x_{jk} must be equal to $\frac{1}{2}$. But then $label(k) < label(i)$ and $label(j) < label(k)$, or $label(j) < label(i)$, which contradicts the property of the labeling that $(i,j) \in P \Rightarrow label(i) < label(j)$. Hence A does not have any triangles that contain an arc in P, and by the arguments given above, Constrained-FAS returns a feasible solution.

In an iteration where k is pivot, we decide the order between, and hence incur a cost for pairs $\{j,k\}$, and for pairs $\{i,j\}$ such that i and j do not both end up on the same side of k. Note that if a cost is incurred for a pair of vertices, then no other cost is incurred for this pair in later iterations. Clearly, the cost we incur for a pair $\{j,k\}$ when k is the pivot, is at most $2(w_{jk}x_{kj} + w_{kj}x_{jk})$. Similarly, if $(i,k), (k,j), (i,j) \in A$ then the cost for pair $\{i,j\}$ is at most $2(w_{ij}x_{ji} + w_{ji}x_{ij})$. Hence the only problematic pairs are those in $T_k(A)$, and if we show that it is possible in each iteration to choose a pivot such that $\frac{\sum_{(j,i)\in T_k(A)} w_{ji}}{\sum_{(j,i)\in T_k(A)} c_{ji}} \leq \alpha$, for α equals 3 (2) when the weights satisfy probability constraints (triangle inequality), then we are done.

Note that if x is feasible for (LP) on (V, w, P), then for any $V' \subset V$, x is also feasible for (LP) on the subgraph $(V', w_{V'}, P_{V'})$. We will show that for a feasible solution x to (LP) on (V, w_V, P_V) and a tournament A_V such that $(i,j) \in A_V$ only if $x_{ij} \geq \frac{1}{2}$, there exists a pivot k such that $\sum_{(j,i)\in T_k(A_V)} w_{ji} \leq \alpha \sum_{(j,i)\in T_k(A_V)} c_{ji}$, by showing that $\sum_{k\in V} \sum_{(j,i)\in T_k(A_V)} w_{ji} \leq \alpha \sum_{k\in V} \sum_{(j,i)\in T_k(A_V)} c_{ji}$.

Let T be the set of triangles $\{(i,k), (k,j), (j,i)\} \subset A_V$, and for a triangle $t \in T$, let $w(t) = \sum_{a\in t} w_a$ and $c(t) = \sum_{a\in t} c_a$. Then

$$\begin{aligned}
\sum_{k\in V} \sum_{(j,i)\in T_k(A_V)} w_{ji} &= \sum_{t\in T} \sum_{(j,i)\in t} w_{ji} = \sum_{t\in T} w(t), \\
\sum_{k\in V} \sum_{(j,i)\in T_k(A_V)} c_{ji} &= \sum_{t\in T} \sum_{(j,i)\in t} c_{ji} = \sum_{t\in T} c(t).
\end{aligned}$$

We will show that for any $t \in T$, $c(t) \geq \frac{1}{\alpha} w(t)$, where α is 3 (2) for the probability constraints (triangle inequality) case. For $a = (j,i)$, let $\bar{w}_a = w_{ij}$. Then for a given triangle t in T $c(t) = \sum_{a\in t}(\bar{w}_a x_a + w_a(1 - x_a)) = \sum_{a\in t} w_a + \sum_{a\in t}(\bar{w}_a - w_a)x_a$. Suppose without loss of generality, that $t = \{a_1, a_2, a_3\}$, with $\bar{w}_{a_1} - w_{a_1} \leq \bar{w}_{a_2} - w_{a_2} \leq \bar{w}_{a_3} - w_{a_3}$. To give a lower bound on $c(t)$, we consider the case that $\bar{w}_{a_1} - w_{a_1} \geq 0$ and the case that $\bar{w}_{a_1} - w_{a_1} < 0$.

In the first case, $\bar{w}_a - w_a \geq 0$ for all $a \in t$. Hence $c(t) = \sum_{a\in t} w_a + \sum_{a\in t}(\bar{w}_a - w_a)x_a \geq \sum_{a\in t} w_a = w(t)$. If $\bar{w}_{a_1} - w_{a_1} = \min_{a\in t}\{\bar{w}_a - w_a\} < 0$, we know from feasibility of x that $\sum_{a\in t} x_a \leq 2$, and again by the definition of T that $x_a \geq \frac{1}{2}$ for each $a \in t$. Therefore

$$\begin{aligned}
c(t) &= \sum_{a\in t} w_a + \sum_{a\in t}(\bar{w}_a - w_a)x_a \\
&\geq \sum_{a\in t} w_a + (\bar{w}_{a_1} - w_{a_1}) \\
&\quad + \frac{1}{2}(\bar{w}_{a_2} - w_{a_2}) + \frac{1}{2}(\bar{w}_{a_3} - w_{a_3}) \\
&= \bar{w}_{a_1} + \frac{1}{2}(\bar{w}_{a_2} + \bar{w}_{a_3}) + \frac{1}{2}(w_{a_2} + w_{a_3}).
\end{aligned}$$

In the case of probability constraints, $\bar{w}_a + w_a = 1$, and hence the above is equal to $1 + \bar{w}_{a_1} \geq 1$. Since $w(t) \leq 3$, it follows that $c(t) \geq \frac{1}{3}w(t)$. When the weights satisfy the triangle inequality, $\bar{w}_{a_2} + \bar{w}_{a_3} \geq w_{a_1}$, so the above is not less than $\bar{w}_{a_1} + \frac{1}{2}w(t) \geq \frac{1}{2}w(t)$. ■

3 A Combinatorial Approach to the Feedback Arc Set Problems

In this section, we describe a simple, combinatorial approach to the constrained weighted feedback arc set problem of Section 2. One reason for doing so is to show an interesting connection to a graph-theoretic conjecture related to the Caccetta-Häggkvist conjecture [9].

An *unordered* pair $\{i, j\}$ of vertices in a digraph (directed graph) G is called a *non-edge* if neither (i, j) nor (j, i) is an arc of G. The vertices i, j are called the *ends* of the non-edge.

CONJECTURE 4. ([9]) *Let G be a digraph with no parallel edges, no directed cycles of length ≤ 3, and k non-edges. Then G has a feedback arc set of at most $k/2$ arcs.*

This conjecture is due to Chudnovsky, Seymour and Sullivan and is related to the Caccetta-Häggkvist conjecture (see [9]). (There are tight examples for the conjecture; for instance, the directed 4-cycle, any acyclic tournament, or products of the two.) The following weakening of the conjecture is known.

THEOREM 4.1. ([3]) *Let G be a digraph as in Conjecture 4. Then G has a feedback arc set of at most k arcs.*

The proof of Theorem 4.1 above also gives a simple recursive algorithm to find a feedback arc set of the given size.

Given a digraph, a *directed triangle cover* is a set of arcs such that every directed triangle in the digraph shares at least one arc with this set.

THEOREM 4.2. *There is a 2-approximation algorithm to find a directed triangle cover. This holds even for weighted graphs.*

Proof (sketch). Krivelevich [8] gave a factor 2 algorithm to find an *undirected* triangle cover. A similar algorithm gives a factor 2 algorithm to find directed triangle cover. The details are reserved for an extended version.

COROLLARY 4.1. *There exists a 4-approximation algorithm for the unconstrained unweighted minimum feedback arc set problem for tournaments.*

Proof. Given a tournament A, use Theorem 4.2 to find a directed triangle cover of A of size $\leq 2\tau$, where τ is the minimum size of a directed triangle cover of A. Delete this triangle cover, and apply Theorem 4.1 to the resulting digraph. This gives a feedback arc set of size $\leq 4\tau$. Since τ is a lower bound on the size of a feedback arc set, we get a 4-approximation algorithm. ■

Theorem 4.1 can be generalized to a weighted version suitable for the weighted feedback arc set problem for tournaments where the weights satisfy the probability constraints (but not necessarily the triangle inequality). We omit the details here, but the approach is similar to the proof of Theorem 4.1.

Finally, we describe the generalization of this approach to the constrained case. Again, for simplicity, we describe the unweighted case. Let G be a digraph and P be a partial order defined on its vertex set. We call an arc (u, v) of G a *hard arc* if $(u, v) \in P$, and a *soft arc* otherwise.

By an *induced* P_2, we mean an ordered triple (u, v, w) of vertices such that G has the arcs (u, v) and (v, w), but $\{u, w\}$ is a non-edge.

THEOREM 4.3. *Let G be a digraph, and P a partial order on its vertex set, such that $\forall(u, v) \in P$, (u, v) is an arc of G. Suppose G has no parallel edges, no directed cycles with ≤ 3 soft arcs, and k non-edges. Then there exist k soft arcs in G whose removal gives an acyclic digraph.*

Proof. The proof generalizes the proof of Theorem 4.1. For any non-edge $\{u, v\}$ such that G has a (directed) u-v path containing exactly one soft arc, *add* the arc (u, v) to G. (It follows that $(u, v) \notin P$, thus (u, v) is a soft arc.) Do this repeatedly till there are no non-edges as above. Let G' be the resulting digraph. Note that each step preserves the hypothesis that there are no directed cycles with ≤ 3 soft arcs, since we only add a soft arc (u, v) if there is a u-v path containing exactly one soft arc, hence G' satisfies that hypothesis too. Also, note that G' has no more non-edges than G, so it suffices to prove the theorem for G'.

As in Section 2, we recursively construct a ranking of the elements by pivoting on a vertex. Pick a pivot vertex v such that the number of induced P_2's of the form (u, v, w) is at least as big as the number of induced P_2's of the form (v, x, y). (The existence of such a vertex follows from a simple counting argument: the sums, over all vertices v, of the numbers of induced P_2's of the above two forms, both amount to the same quantity, that is, the total number of induced P_2's in G'.)

Now let I, O and N be the set of in-neighbors, out-neighbors and non-neighbors of v, respectively, in G'. We recurse on the instance induced by $I \cup N$ and O, and return the ranking that has the ordered vertices in $I \cup N$, followed by v, followed by the ordered vertices in O.

The only arcs that become backarcs (or are removed) in this iteration are of the form (x, y) with $x \in O, y \in N$, since there are no arcs from v to $I \cup N$ by the definition of I and N, and there are no arcs (x, z) with $x \in O, z \in I$, since otherwise v, x and z are in a directed cycle with ≤ 3 soft arcs. The choice of v now implies that the number of non-edges with one end each in I and O is at least as large as the number of arcs of the form (x, y) with $x \in O, y \in N$, so we can charge the deletion to the non-edges with one end each in I and O.

We claim that all arcs deleted in this step are soft arcs. Suppose not, and let (x, y) be a hard arc with $x \in O, y \in N$. If (v, x) is a hard arc, then by the transitivity of P, (v, y) must also be in P. In particular, (v, y) must be an arc of G', contrary to the definition of N. On the other hand, if (v, x) is a soft arc, then (v, x, y) gives a directed path containing exactly one soft arc, hence the step described in the first paragraph of the proof should have added a soft arc (v, y). Again, this contradicts the definition of N. This proves the theorem. ■

Theorem 4.2 can also be generalized to the constrained case but with a weaker factor of 3 instead of 2. On the positive side, factor 3 can be obtained by a combinatorial (primal-dual) algorithm for the constrained (edge weighted) case. The following theorem is easy to prove in the setting of set cover: we let each directed cycle with ≤ 3 soft arcs correspond to an element in our ground set, and let each soft arc correspond to a set that contains the cycles in which the arc appears. Then each element belongs to at most three sets, and the following is well known.

THEOREM 4.4. *There is a combinatorial 3-approximation algorithm to find a set of arcs to cover all the directed cycles with ≤ 3 soft arcs.*

COROLLARY 4.2. *There is a combinatorial 6-approximation algorithm for the constrained unweighted minimum feedback arc set problem for tournaments.*

As before, Theorem 4.3 can be generalized to a weighted version, where the weights satisfy the probability constraints. This gives an analogous 6-approximation algorithm for the constrained weighted version of the problem, when the weights satisfy the probability constraints. Even though the approximation factor is worse than the factor 3 proven in Section 2, the algorithm here is a purely combinatorial algorithm without any need to solve linear programs.

5 Correlation and Consensus Clustering

Given a set of vertices V, nonnegative weights w_{ij}^+, w_{ij}^- for each pair $i, j \in V$, and sets P^+, P^- we want to find a clustering that has i and j in the same cluster if $\{i, j\} \in P^+$, and i and j in separate clusters if $\{i, j\} \in P^-$, that minimizes the sum of w_{ij}^+ over all i, j in different clusters plus the sum of w_{ij}^- over all i, j in the same cluster. We again consider two kinds of constraints on the weights: probability constraints ($w_{ij}^+ + w_{ij}^- = 1$) and the triangle inequality ($w_{ij}^- + w_{jk}^- \geq w_{ik}^-$ and $w_{ij}^+ + w_{jk}^- \geq w_{ik}^+$).

Let $x_{ij}^+ = 1$ denote that i and j are in the same cluster, $x_{ij}^+ = 0$ that i are j are not in the same cluster, and let $x_{ij}^- = 1 - x_{ij}^+$. For three vertices i, j, k, it is impossible that i and j are in the same cluster ($x_{ij}^- = 0$), j and k are in the same cluster ($x_{jk}^- = 0$), but i and k are not in the same cluster ($x_{ik}^+ = 0$), hence for any feasible clustering $x_{ij}^- + x_{jk}^- + x_{ik}^+ \geq 1$. The following linear program thus gives a lower bound on the value of an optimal clustering:

$$
\begin{array}{llll}
 & \min & \sum_{i<j} (x_{ij}^+ w_{ij}^- + x_{ij}^- w_{ij}^+) & \\
 & \text{s.t.} & x_{ij}^- + x_{jk}^- + x_{ik}^+ \geq 1 & \text{for all distinct } i, j, k \\
 & & x_{ij}^+ + x_{ij}^- = 1 & \text{for all } i \neq j \\
(LP_{CC}) & & x_{ij}^+ = 1 & \text{for all } \{i, j\} \in P^+ \\
 & & x_{ij}^- = 1 & \text{for all } \{i, j\} \in P^- \\
 & & x_{ij}^+ = x_{ji}^+ & \text{for all } i \neq j \\
 & & x_{ij}^+, x_{ij}^- \geq 0 & \text{for all } i \neq j
\end{array}
$$

The version of this LP where $P^+ \cup P^-$ is empty was also used by Ailon et al. [2].

Given an optimal solution x to LP_{CC}, we form two sets of edges E^+, E^- so that $(V, E^+ \cup E^-)$ is a complete graph, $E^+ \cap E^- = \emptyset$, and $\{i, j\} \in E^\pm$ only if $x_{ij}^\pm \geq \frac{1}{2}$. We will adapt the algorithm from the previous section, so that if k is chosen as pivot vertex, we put j into the same cluster as k if $\{j, k\} \in E^+$ and we separate j from k if $\{j, k\} \in E^-$. The algorithm then recurses on all vertices that are not put into the same cluster as k.

We call the counterpart of a triangle a "bad triplet" ([2]): a triplet (i, j, k) such that $\{i, j\} \in E^+, \{j, k\} \in E^+$ and $\{k, i\} \in E^-$. If one of the vertices of a bad triplet is chosen as pivot, then the "opposite" pair is not clustered according to $E^+ \cup E^-$. It is easily verified that for any triplet that is not bad, we can pick any of the vertices as pivot, and our clustering will obey E^+ and E^- for all three pairs. Hence to make sure that our algorithm returns a clustering that is feasible with respect to P^+ and P^-, it is enough to ensure that $E^+ \cup E^-$ does not contain any bad triplets that contain a pair in P^+ or P^-.

We let $T_k^+(E) = \{\{i, j\} \in E^+ | \{j, k\} \in E^+, \{k, i\} \in E^-\}$ and $T_k^-(E) = \{\{i, j\} \in E^- | \{j, k\} \in E^+, \{k, i\} \in$

$E^+\}$, i.e. if we pivot on k, then $T_k^+(E) \cup T_k^-(E)$ are the pairs of vertices that we break up even though they're in E^+, and the pairs that we join even though they're in E^-. In the algorithm we will choose a pivot k that minimizes the ratio of the cost for the pairs in $T_k^+(E) \cup T_k^-(E)$ in the algorithm and the cost for these pairs in the optimal solution to (LP_{CC}).

In the following, for $V' \subseteq V$, let $w_{V'} = \{w_{ij}^+, w_{ij}^- | i \in V', j \in V'\}$, $x_{V'} = \{x_{ij}^+, x_{ij}^- | i \in V', j \in V'\}$, $E_{V'}^{\pm} = \{\{i,j\} \in E^{\pm} | i \in V', j \in V'\}$ and let $c_{ij} = x_{ij}^+ w_{ij}^- + x_{ij}^- w_{ij}^+$. We give the formal statement of our algorithm, CC-Pivot, in Figure 5.

```
Constrained-CC$(V, w, P^+, P^-)$
  Let $x$ be an opt. sol. to $(LP_{CC})$ on $(V, w, P^+, P^-)$.
  Label the vertices in $V$ such that
      $\{i,j\} \in P^+ \Rightarrow label(i) = label(j)$ and
      $\{i,j\} \in P^- \Rightarrow label(i) \neq label(j)$. [2]
  Let $E^+ = \{\{i,j\} : x_{ij}^+ > \frac{1}{2}\} \cup$
      $\{\{i,j\} : x_{ij}^+ = \frac{1}{2}$ and $label(i) = label(j)\}$
  Let $E^- = \{\{i,j\} : x_{ij}^- > \frac{1}{2}\} \cup$
      $\{\{i,j\} : x_{ij}^- = \frac{1}{2}$ and $label(i) \neq label(j)\}$
  Return CC-Pivot$(V, w, E^+, E^-, x)$

CC-Pivot$(V, w, E^+, E^-, x)$
  Pick pivot $k \in V$ minimizing
      $\frac{\sum_{\{i,j\} \in T_k^+(E)} w_{ij}^+ + \sum_{\{i,j\} \in T_k^-(E)} w_{ij}^-}{\sum_{\{i,j\} \in T_k^+(E) \cup T_k^-(E)} c_{ij}}$.
  $C \leftarrow \{k\} \cup \{j \in V : \{j,k\} \in E^+\}$.
  $V' \leftarrow V \backslash C$
  Return C, CC-Pivot$(V', w_{V'}, E_{V'}^+, E_{V'}^-, x_{V'})$
```

[2] This can be done recursively by taking any vertex, labeling it 0, and also giving label 0 to all vertices j such that $\{i,j\} \in P^+$. (We assume that if $\{i,j\}, \{j,k\} \in P^+$, then $\{i,k\} \in P^+$.) We delete all labeled vertices, increase our label by 1, and repeat.

Figure 2: Our algorithm for correlation and consensus clustering.

Theorem 5.1. *Constrained-CC is a 3 (2)-approximation algorithm for constrained correlation and consensus clustering when the weights satisfy the probability constraints (triangle inequality).*

Proof. We first show that the algorithm returns a feasible clustering that obeys P^+ and P^- by showing that $E^+ \cup E^-$ does not have any bad triplets that contain a pair in $P^+ \cup P^-$. Suppose $\{i,j\} \in P^+$ and i, j and k form a bad triplet, i.e. $\{i,j\} \in E^+, \{j,k\} \in E^+, \{k,i\} \in E^-$. Then $x_{jk}^+ \geq \frac{1}{2}, x_{ki}^- \geq \frac{1}{2}$, but by the triangle inequality constraints of (LP_{CC}) and the fact that $x_{ij}^+ = 1$ if $\{i,j\} \in P^+$, we have $x_{jk}^- + x_{ki}^+ \geq 1$, or $1 - x_{jk}^+ + 1 - x_{ki}^- \geq 1$, so $x_{jk}^+ = \frac{1}{2}$ and $x_{ki}^- = \frac{1}{2}$. But then $label(j) = label(k)$ and $label(k) \neq label(i)$, so $label(j) \neq label(i)$ which contradicts the properties of our labeling, since $\{i,j\} \in P^+$. A similar contradiction can be derived if we assume there is a bad triplet containing $\{i,j\} \in P^-$. Hence the algorithm returns a feasible clustering that has i and j in one cluster if $\{i,j\} \in P^+$, and i and j in separate clusters if $\{i,j\} \in P^-$.

To bound the cost of the algorithm, we look at the cost incurred in one call to CC-Pivot. If k is pivot, we decide whether or not to break up into separate clusters pairs $\{j,k\}$, and pairs $\{i,j\}$ such that i is clustered with k and j is not, or both i and j are clustered with k (as the cluster containing k is not broken down into smaller clusters). If a cost is incurred for a pair of vertices, then no other cost is incurred for this pair in later iterations. As before, the cost we incur for pair $\{j,k\}$ is not more than $2(x_{jk}^+ w_{jk}^- + x_{jk}^- w_{jk}^+)$, because we either incur a cost of w_{jk}^-, in which case $x_{jk}^+ \geq \frac{1}{2}$, or we incur a cost of w_{jk}^+, but then $x_{jk}^- \geq \frac{1}{2}$. Similarly, for a pair $\{i,j\}$ that is not in a bad triplet with k, either i is clustered with k, j is not clustered with k, and $x_{ij}^- \geq \frac{1}{2}$, or i and j are both clustered with k and $x_{ij}^+ \geq \frac{1}{2}$, so the cost we incur is accounted for by twice the contribution of $\{i,j\}$ to the objective value of (LP_{CC}). If i and j are both separated from k, then they are in the next recursive call together, and no cost is incurred yet for this pair, as it is not decided yet whether they will be clustered together or not.

The remaining possibilities are the pairs $\{i,j\}$ in $T_k^+(E) \cup T_k^-(E)$. We need to show that there exists a pivot in each iteration such that

$$\sum_{\{i,j\} \in T_k^+(E)} w_{ij}^+ + \sum_{\{i,j\} \in T_k^-(E)} w_{ij}^- \leq \alpha \sum_{\{i,j\} \in T_k^+(E) \cup T_k^-(E)} c_{ij}$$

where α is 3 and 2 in the probability constraints case and triangle inequality case respectively. We will show that for any feasible solution x,

$$\sum_{k \in V} \Big(\sum_{\{i,j\} \in T_k^+(E)} w_{ij}^+ + \sum_{\{i,j\} \in T_k^-(E)} w_{ij}^- \Big) \leq \alpha \sum_{k \in V} \sum_{\{i,j\} \in T_k^+(E) \cup T_k^-(E)} c_{ij}.$$

Let T be the set of bad triplets (i,j,k) such that $\{i,j\} \in E^+, \{j,k\} \in E^+$ and $\{k,i\} \in E^-$. For a triplet $t = (i,j,k) \in T$, let $w(t) = w_{ij}^+ + w_{jk}^+ + w_{ki}^-$, $c(t) = c_{ij} + c_{jk} + c_{ki}$. Note that if one of the vertices (say vertex v) of a bad triplet is chosen, then the edge connecting the remaining two vertices is either in $T_v^+(E)$ or in $T_v^-(E)$. Hence we get

$$\sum_{k\in V}\Big(\sum_{\{i,j\}\in T_k^+(E)} w_{ij}^+ + \sum_{\{i,j\}\in T_k^-(E)} w_{ij}^-\Big) = \sum_{t\in T} w(t),$$
$$\sum_{k\in V}\sum_{\{i,j\}\in T_k^+(E)\cup T_k^-(E)} c_{ij} = \sum_{t\in T} c(t).$$

As in the proof of Theorem 2.1, we need to show $c(t) \geq \frac{1}{\alpha} w(t)$. Because the analysis is similar to that in the proof of Theorem 2.1, we omit the remainder of the analysis for space reasons. ■

6 Hierarchical Clustering

An M-level hierarchical clustering of a set V is a nested clustering of the elements in V, where the clustering at level ℓ is a refinement of the clustering at level $\ell + 1$. Given a set V and a matrix D with $D_{ij} \in \{0,\dots,M\}$ for any distinct $i,j \in V$, we want to find an M-level hierarchical clustering of V minimizing $\sum_{i,j\in V} |D_{ij} - \lambda_{ij}|$, where λ_{ij} is the number of levels in which i and j are in different clusters, or equivalently, since the clusterings are nested, i and j are in different clusters at levels $1,\dots,\lambda_{ij}$, and in the same cluster at levels $\lambda_{ij}+1,\dots,M$. Note that this is equivalent to finding an *ultrametric* that minimizes the ℓ_1 distance with D. An ultrametric is a tree metric in which all vertices are at the leaves of the tree, and the distance from each leaf to the root is the same.

In addition to the input data D, we allow the input to specify a lower bound L_{ij}, and an upper bound U_{ij} for each pair of vertices i,j. Any feasible hierarchical clustering should have i and j in different clusters at levels $1,\dots,L_{ij}$, and in the same cluster at levels $U_{ij}+1,\dots,M$. If we look at hierarchical clustering as fitting an ultrametric, then the constraints specify that the distance between a pair is at least L_{ij} and at most U_{ij}. The constraints have to be consistent, i.e. for any pair i,j we must have $L_{ij} \leq U_{ij}$ and for any $k \neq i,j$, $L_{ij} \leq \max\{U_{ik}, U_{jk}\}$.

We now observe that this problem is closely related to the correlation clustering problem. Let

$$d_{ij}^\ell = \begin{cases} 1 & \text{if } D_{ij} \geq \ell \\ 0 & \text{otherwise.}\end{cases}$$

Let $x_{ij}^\ell = 1$ if i and j are in separate clusters at level ℓ, and $x_{ij}^\ell = 0$ otherwise. We give below an integer programming formulation (HC) for the problem. The linear programming relaxation of (HC) was used by Ailon and Charikar [1] to give an $O(\log n \log\log n)^{1/p}$-approximation algorithm for fitting ultrametrics and tree metrics to general dissimilarity data (i.e. not necessarily in $\{0,\dots,M\}$) under the L_p-norm. We will denote by LP_{HC} the LP relaxation of the integer program (HC). The integer program essentially solves a correlation clustering problem at each level ℓ subject to the constraint that the clustering at level $\ell - 1$ is a refinement of that at level ℓ. The variables x_{ij}^ℓ give a correlation clustering at level ℓ with objective function weight $w_{ij}^+ = 0$ and $w_{ij}^- = 1$ if $D_{ij} \geq \ell$, and weight $w_{ij}^+ = 1$ and $w_{ij}^- = 0$ if $D_{ij} < \ell$, and with $P^+ = \{\{i,j\} : \ell > U_{ij}\}, P^- = \{\{i,j\} : \ell \leq L_{ij}\}$; without the consistency constraints the integer program reduces to M copies of the correlation clustering integer program of the previous section.

$$\begin{array}{lll}
 & \min \sum_{i<j}\sum_{\ell=1}^{M} \Big((1-d_{ij}^\ell)x_{ij}^\ell + d_{ij}^\ell(1-x_{ij}^\ell)\Big) & \\
 & \text{s.t.}\quad x_{ij}^\ell + x_{jk}^\ell \geq x_{ik}^\ell & \forall \text{ distinct } i,j,k, \forall \ell = 1,\dots,M \\
(HC) & x_{ij}^\ell \leq x_{ij}^{\ell-1} & \forall i\neq j, \forall \ell = 2,\dots,M \\
 & x_{ij}^\ell = 1 & \forall i\neq j, \forall \ell \leq L_{ij} \\
 & x_{ij}^\ell = 0 & \forall i\neq j, \forall \ell > U_{ij} \\
 & x_{ij}^\ell \in \{0,1\} & \forall i\neq j, \forall \ell = 1,\dots,M
\end{array}$$

For each level we form two disjoint sets of edges $E^{+,\ell}, E^{-,\ell}$ so that $(V, E^{+,\ell}\cup E^{-,\ell})$ is a complete graph and $\{i,j\} \in E_\ell^-$ only if $x_{ij}^\ell \geq \frac{1}{2}$, and $\{i,j\} \in E^{+,\ell}$ only if $x_{ij}^\ell \leq \frac{1}{2}$. We will use a slightly altered version of CC-Pivot to generate a clustering at level M, and then recursively call the algorithm to generate an $(M-1)$-level hierarchical clustering for each cluster found.

To find a clustering of V at level M, we pick a pivot k and we put j into the same cluster as k if $\{j,k\} \in E^{+,M}$ and we separate j from k if $\{j,k\} \in E^{-,M}$. Next we recurse on all vertices that are not in the same cluster as k. Once we have found a clustering $\mathcal{C}_M$ of V, we recurse on each cluster $C \in \mathcal{C}_M$ to find an $(M-1)$-level subclustering, this time using $E^{+,M-1}$ and $E^{-,M-1}$ to decide whether a vertex j is clustered together with pivot vertex k or not.

Similar to before, we let $T_k^+(E^M) = \{\{i,j\} \in E^{+,M} | \{j,k\} \in E^{+,M}, \{k,i\} \in E^{-,M}\}$ and $T_k^-(E^M) = \{\{i,j\} \in E^{-,M} | \{j,k\} \in E^{+,M}, \{k,i\} \in E^{+,M}\}$. When k is chosen as pivot, then the pairs in $T_k^+(E^M)$ and $T_k^-(E^M)$ are the pairs that we do not cluster as suggested by rounding the LP variables. In the algorithm we will choose a pivot k that minimizes the ratio of the cost for the pairs in $T_k^+(E^M)\cup T_k^-(E^M)$ in the algorithm compared to the "budget" we have for these pairs, i.e. the cost for these pairs in the optimal solution to (LP_{HC}). In the following, let $c_{ij}^\ell = (1-d_{ij}^\ell)x_{ij}^\ell + d_{ij}^\ell(1-x_{ij}^\ell)$. Let $D = \{D_{ij} | i,j \in V\}$ and for $V' \subset V$, let $D_{V'} = \{D_{ij} | i, j \in V'\}$. We define $E^{+,\ell}_{V'}$ and $E^{-,\ell}_{V'}$ in a similar fashion. Similarly, we define x to contain x_{ij} for all pairs $\{i, j\}$ in V, and $x_{V'}$ is this set restricted to pairs in V'. Our algorithm, Hierarchical-Clustering, is given in Figure 3.

```
Hierarchical-Clustering($V, D, M, L, U$)
  Let $x$ be an opt. sol. to $(LP_{HC})$ on $(V, D, M, L, U)$.
  For $\ell = 1$ to $M$
      Label the vertices in $V$ such that
          $\ell > U_{ij} \Rightarrow \mathit{label}^\ell(i) = \mathit{label}^\ell(j)$ and
          $\ell \leq L_{ij} \Rightarrow \mathit{label}^\ell(i) \neq \mathit{label}^\ell(j)$.
      Let $E^{+,\ell} = \{\{i,j\} : x^\ell_{ij} < \frac{1}{2}\} \cup$
          $\{\{i,j\} : x^\ell_{ij} = \frac{1}{2}$ and $\mathit{label}^\ell(i) = \mathit{label}^\ell(j)\}$
      Let $E^{-,\ell} = \{\{i,j\} : x^\ell_{ij} > \frac{1}{2}\} \cup$
          $\{\{i,j\} : x^\ell_{ij} = \frac{1}{2}$ and $\mathit{label}^\ell(i) \neq \mathit{label}^\ell(j)\}$

  (Generate clusterings $\mathcal{C}_1, \ldots, \mathcal{C}_M$ such that $\mathcal{C}_\ell$ is
      a refinement of $\mathcal{C}_{\ell+1}$)
  $\mathcal{C}_M$ = HC-Pivot$(V, D, x, E^{+,M}, E^{-,M}, M)$
  For $\ell = M - 1$ down to 1
      $\mathcal{C}_\ell = \emptyset$
      for $C \in \mathcal{C}_{\ell+1}$
          $\mathcal{C}_\ell \leftarrow \{\mathcal{C}_\ell$, HC-Pivot$(C, D_C, x_C, E^{+,\ell}, E^{-,\ell}, \ell)\}$
  return $\{\mathcal{C}_1, \ldots, \mathcal{C}_M\}$.

HC-Pivot($V, D, x, E^+, E^-, M$)
  Pick pivot $k \in V$ minimizing
      $\frac{\sum_{\{i,j\} \in T^+_k(E)} \sum_{\ell=1}^M (1 - d^\ell_{ij}) + \sum_{\{i,j\} \in T^-_k(E)} d^M_{ij}}{\sum_{\{i,j\} \in T^+_k(E)} \sum_{\ell=1}^M c^\ell_{ij} + \sum_{\{i,j\} \in T^-_k(E)} c^M_{ij}}$.

  $C \leftarrow \{k\} \cup \{j \in V : \{j,k\} \in E^+\}$
  $V' \leftarrow V \backslash C$
  Return $\{C$, HC-Pivot$(V', D_{V'}, x_{V'}, E^+_{V'}, E^-_{V'}, M)\}$.
```

Figure 3: Our algorithm for finding a hierarchical clustering.

THEOREM 6.1. *Hierarchical-clustering is an $(M + 2)$-approximation algorithm for M-level hierarchical clustering.*

Proof. Let $P^{+,\ell} = \{\{i,j\} : \ell > U_{ij}\}, P^{-,\ell} = \{\{i,j\} : \ell \leq L_{ij}\}$. By the proof of Theorem 5.1, $E^{+,\ell}$ and $E^{-,\ell}$ do not contain any bad triplets that contain a pair in $P^{+,\ell} \cup P^{-,\ell}$. Hence our algorithm does not separate i, j at level $\ell \in \{U_{ij} + 1, \ldots, M\}$ since $\{i, j\} \in P^{+,\ell}$ for $\ell > U_{ij}$. On the other hand, if i, j are not separated already at some level $\ell > L_{ij}$, then the fact that $\{i, j\} \in P^{-,L_{ij}}$ ensures that i and j will be separated at level L_{ij} (and thus for all levels $\ell \leq L_{ij}$). Hence the algorithm returns a feasible solution. To bound the cost of the algorithm, we will bound the cost incurred in a call to HC-Pivot in terms of the value of the LP solution.

In a call to HC-Pivot at level ℓ with vertex set V' where k is pivot, we make the decision for each $j \in V'$ whether to cluster $\{j, k\}$ together (at level ℓ) or separating them (at all levels $1, \ldots, \ell$), plus we cluster i and j together at level ℓ if they are both clustered with k, and we separate i, j for all levels $1, \ldots, \ell$ if only one of them is clustered with k.

If a pair is clustered together, the cost incurred is d^ℓ_{ij}. If the pair is in $E^{+,\ell}$, then $x^\ell_{ij} \leq \frac{1}{2}$, and hence $c^\ell_{ij} \geq \frac{1}{2} d^\ell_{ij}$, so c^ℓ_{ij} pays for at least half the cost. If a pair is separated, the cost is $\sum_{\ell'=1}^{\ell} (1 - d^{\ell'}_{ij})$. If the pair is in $E^{-,\ell}$, then $x^\ell_{ij} \geq \frac{1}{2}$, and by the consistency constraints of (HC_{LP}) , $x^{\ell'}_{ij} \geq x^\ell_{ij}$ for each $\ell' \leq \ell$, hence $\sum_{\ell'=1}^{\ell} c^{\ell'}_{ij} \geq \frac{1}{2} \sum_{\ell'=1}^{\ell} (1 - d^{\ell'}_{ij})$, so $\sum_{\ell'=1}^{\ell} c^{\ell'}_{ij}$ pays half the cost of separating i and j at levels $1, \ldots, \ell$.

The remaining pairs for which we have not charged the cost yet are (1) the pairs $\{i, j\}$ that are both clustered with k, but that are not in $E^{+,\ell}$, and (2) the pairs $\{i, j\}$ for which i is clustered with k and j is separated from k, that are not in $E^{-,\ell}$. Note that these are exactly the pairs in $T^-_k(E^\ell)$ and $T^+_k(E^\ell)$. For the pairs in $T^-_k(E^\ell)$ we incur a cost of d^ℓ_{ij}, and for the pairs in $T^+_k(E^\ell)$ we incur a cost $\sum_{\ell'=1}^{\ell} (1 - d^{\ell'}_{ij})$.

CLAIM 6.1. *Let x be a feasible solution to (HC_{LP}). Let E^+, E^- be a partition of $E = V \times V$ such that $\{i, j\} \in E^+$ only if $x^M_{ij} \leq \frac{1}{2}$ and $\{i, j\} \in E^-$ only if $x^M_{ij} \geq \frac{1}{2}$. Then*

$$\begin{aligned}
&\sum_{k \in V} \Big(\sum_{\{i,j\} \in T^+_k(E)} \sum_{\ell=1}^{M} (1 - d^\ell_{ij}) + \sum_{\{i,j\} \in T^-_k(E)} d^M_{ij} \Big) \\
&\leq \quad (M + 2) \sum_{k \in V} \Big(\sum_{\{i,j\} \in T^+_k(E)} \sum_{\ell=1}^{M} c^\ell_{ij} + \sum_{\{i,j\} \in T^-_k(E)} c^M_{ij} \Big)
\end{aligned}$$

It follows from the claim that in each call to HC-Pivot, say at level ℓ, we can choose a pivot k such that the cost for the pairs in $T^+_k(E) \cup T^-_k(E)$ can be charged against $(\ell + 2)$ times the LP contribution of the variables corresponding to the decisions made for these pairs. So we have shown that c^ℓ_{ij} is charged either at most twice, or at most $\ell + 2$ times for each $\ell = 1, \ldots, M$. We conclude that the total cost of the algorithm is at most $(M + 2) \sum_{i,j} \sum_{\ell=1}^{M} c^\ell_{ij} = (M + 2)OPT$.

Proof of Claim 6.1. We define

$$A^M_k \quad = \quad \sum_{\{i,j\} \in T^+_k(E)} (1 - d^M_{ij}) + \sum_{\{i,j\} \in T^-_k(E)} d^M_{ij},$$

$$\begin{aligned} A_k^\ell &= \sum_{\{i,j\}\in T_k^+(E)} (1-d_{ij}^\ell), \\ B_k^M &= \sum_{\{i,j\}\in T_k^+(E)\cup T_k^-(E)} c_{ij}^M, \\ B_k^\ell &= \sum_{\{i,j\}\in T_k^+(E)} c_{ij}^\ell \end{aligned}$$

Let (i,j,k) be a bad triplet if $\{i,j\}\in E^+, \{j,k\}\in E^+, \{k,i\}\in E^-$, and let T be the set of all bad triplets. For a bad triplet $t=(i,j,k)$, let $c^M(t)=c_{ij}^M+c_{jk}^M+c_{ki}^M$, and let $d^M(t)=1-d_{ij}^M+1-d_{jk}^M+d_{ki}^M$. Note that if one of the vertices (say vertex v) of a bad triplet is chosen, then the edge connecting the remaining two vertices is either in $T_v^+(E)$ or in $T_v^-(E)$. Hence we get

$$\sum_{k\in V} A_k^M = \sum_{t\in T} d^M(t) \quad \text{and} \quad \sum_{k\in V} B_k^M = \sum_{t\in T} c^M(t).$$

Now, note that if we let $x_e^-=x_e^M, x_e^+=1-x_e^M$, and let $w_e^-=d_e^M, w_e^+=1-d_e^M$, then we see that $T_k^+(E), T_k^-(E)$ and T correspond exactly to how we defined them in the simple clustering case, and that $c^M(t)$ and $d^M(t)$ are exactly the same as $c(t)$ and $w(t)$. Since the weights satisfy $w_e^++w_e^-=1$, we have from the proof of Theorem 5.1 that $w(t)\le 3c(t)$. It follows that $d^M(t)\le 3c^M(t)$ and hence

$$\sum_{k\in V} A_k^M \le 3\sum_{k\in V} B_k^M$$

Next we will show that

$$\sum_{k\in V} A_k^\ell \le \sum_{k\in V} B_k^M + 4\sum_{k\in V} B_k^\ell. \tag{6.1}$$

It then follows that

$$\begin{aligned} &\sum_{k\in V}\Big(\sum_{\{i,j\}\in T_k^+(E)}\sum_{\ell=1}^{M}(1-d_{ij}^\ell)+\sum_{\{i,j\}\in T_k^-(E)} d_{ij}^M\Big) \\ = &\sum_{k\in V}\sum_{\ell=1}^{M-1} A_k^\ell + \sum_{k\in V} A_k^M \\ \le &\sum_{k\in V}\sum_{\ell=1}^{M-1} B_k^M + 4\sum_{k\in V}\sum_{\ell=1}^{M-1} B_k^\ell + 3\sum_{k\in V} B_k^M \\ \le &(M+2)\Big(\sum_{k\in V}\sum_{\ell=1}^{M-1} B_k^\ell + \sum_{k\in V} B_k^M\Big) \\ = &(M+2)\sum_{k\in V}\Big(\sum_{\{i,j\}\in T_k^+(E)}\sum_{\ell=1}^{M} c_{ij}^\ell + \sum_{\{i,j\}\in T_k^-(E)} c_{ij}^M\Big). \end{aligned}$$

To prove (6.1), note that in $\sum_{k\in V}\sum_{\{i,j\}\in T_k^+(E)}(1-d_{ij}^\ell)$, $(1-d_{ij}^\ell)$ is counted once for every bad triplet (i,j,k) or (k,i,j). For bad triplet $t=(i,j,k)\in T$, let $d_+^\ell(t)=(1-d_{ij}^\ell)+(1-d_{jk}^\ell)$, and let $c_+^\ell(t)=c_{ij}^\ell+c_{jk}^\ell$. Then

$$\sum_{k\in V} A_k^\ell = \sum_{t\in T} d_+^\ell(t), \quad \text{and} \quad \sum_{k\in V} B_k^\ell = \sum_{t\in T} c_+^\ell(t).$$

Thus we can prove (6.1) by showing that $d_+^\ell(t)\le 4c_+^\ell(t)+c^M(t)$. This is trivially true in the case that $d_+^\ell(t)=0$, so we consider the two cases $d_+^\ell(t)=1$ and $d_+^\ell(t)=2$. If $d_+^\ell(t)=1$, note that either d_{ij}^ℓ or d_{jk}^ℓ is 0. Suppose without loss of generality that $d_{ij}^\ell=0$. Then by the definition of d_{ij}^ℓ, $D_{ij}<\ell$. Since $\ell<M$, this implies that $d_{ij}^M=0$ also. So $d^M(t)\ge 1-d_{ij}^M\ge 1$. Going back to the proof of Theorem 5.1, we know that either $c^M(t)\ge d^M(t)$ or $c^M(t)\ge 1$. Hence $d^M(t)\ge 1$ implies that $c^M(t)\ge 1$, hence $d_+^\ell(t)\le c^M(t)$.

If $d_+^\ell(t)=2$, then $c_+^\ell(t)=x_{ij}^\ell+x_{jk}^\ell$. By the triangle inequality constraints in (LP_{HC}), $x_{ij}^\ell+x_{jk}^\ell\ge x_{ki}^\ell$. From the fact that (i,j,k) is a bad triplet, we know that $x_{ki}^M\ge\frac{1}{2}$. Additionally, by the refinement constraints of (LP_{HC}), $x_{ki}^\ell\ge x_{ki}^M$, so we get that $c_+^\ell(t)\ge\frac{1}{2}$, and hence $d_+^\ell(t)\le 4c_+^\ell(t)$. ■

References

[1] N. Ailon and M. Charikar. Fitting tree metrics: Hierarchical clustering and phylogeny. In *FOCS 2005*, pages 73–82.

[2] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: ranking and clustering. In *STOC 2005*, pages 684–693.

[3] M. Chudnovsky, P. Seymour, and B. Sullivan. Cycles in dense digraphs. Submitted, 2006.

[4] D. Coppersmith, L. Fleischer, and A. Rudra. Ordering by weighted number of wins gives a good ranking for weighted tournaments. In *SODA 2006*, pages 776–782.

[5] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *WWW 2001*, pages 613–622.

[6] V. Filkov and S. Skiena. Integrating microarray data by consensus clustering. In *ICTAI 2003*, pages 418–425.

[7] J. Kemeny. Mathematics without numbers. *Daedalus*, 88:575–591, 1959.

[8] M. Krivelevich. On a conjecture of Tuza about packing and covering of triangles. *Discrete Mathematics*, 142:281–286, 1995.

[9] B. D. Sullivan. A summary of results and problems related to the Caccetta-Häggkvist conjecture. URL: `http://www.aimath.org/WWN/caccetta/caccetta.pdf`, April 2006.