Flow Metrics

Claudson F. Bornstein and Santosh Vempala [1,2]

Instituto de Matemática and
COPPE/Sistemas, UFRJ
21945-970 Rio de Janeiro RJ Brazil

Department of Mathematics and
Laboratory for Computer Science, MIT
Cambridge MA 02139 USA

Abstract

We introduce *flow metrics* as a relaxation of path metrics (i.e. linear orderings). They are defined by polynomial-sized linear programs and have interesting properties including *spreading*. We use them to obtain relaxations for several NP-hard linear ordering problems such as minimum linear arrangement and minimum path-width. Our approach has the advantage of achieving the best-known approximation guarantees for these problems using the same relaxation and essentially the same rounding algorithm for all the problems while varying only the objective function from problem to problem. This is in contrast to the current state of the literature where each problem either has a new relaxation or a new rounding or both. We also characterize a natural projection of the flow polyhedron.

1 Introduction

Many problems in combinatorial optimization can be expressed in the framework of finding an assignment of pairwise distances of a specified type which minimizes (or maximizes) a prescribed cost function on the distances. For example, in the *maximum cut* problem [8], the distance is restricted to be a cut metric [4] and the cost function is defined as the sum of the distances between

Email addresses: `cfb@cos.ufrj.br`, `vempala@math.mit.edu` (Claudson F. Bornstein and Santosh Vempala).

[1] Supported in part by CNPq grant 300083/99-8 and a Protem CNPq/NSF grant.
[2] Supported in part by NSF Career Award CCR-9875024.

Preprint submitted to Elsevier Science

all pairs of vertices that are adjacent in the input graph. Similarly in the *sparsest cut* problem [9] the distance is also a cut metric. Other metrics that model interesting problems include Euclidean metrics (possibly of bounded dimension)[8,2], path metrics [3,6,10] and tree metrics [1].

The problems modeled in this way are often NP-hard (as in the case of the examples above) and one approach to solving them is to consider relaxations of the associated distances. In [9], cut metrics were relaxed to just metrics, i.e. distances satisfying the triangle inequality and this led to an $O(\log n)$ approximation for the sparsest cut. In [8] cut metrics were relaxed to Euclidean metrics (of finite but unbounded dimension) and this led to a 0.87856 approximation for the maximum cut. In these relaxations, the key is to ensure that the optimization problem over the relaxed metric can be solved in polynomial time and the cost function remains an approximation of the original (i.e. the integrality ratio of the relaxation is small).

In this paper, we consider problems modeled by path metrics, i.e. the metrics induced by line graphs. An example is *minimum linear arrangement* which asks for a linear ordering of the vertices of a given graph that minimizes the sum of differences between endpoints of edges of the graph. In our framework, it asks for a path metric that minimizes the sum of the edge lengths. Other interesting problems that can be modeled by this framework include *minimum interval graph completion*, *minimum pathwidth* and *minimum bandwidth*. These problems are all NP-hard, and a variety of approaches have been used to obtain efficient approximation algorithms for them.

In [6], path metrics were relaxed to *spreading* metrics while in [2,5] they were relaxed to Euclidean spreading metrics. Roughly speaking, a spreading metric satisfies the property that the average distance between vertices in any subset of k vertices is about k, as in the case of a path. This property can be modeled using a set of linear constraints. Although the number of constraints is exponential in the size of the input graph, they admit an efficient separation oracle [7] and so the resulting relaxations can be solved in polynomial time.

In this paper, we present a new relaxation for a path metric, which we call a *flow* metric. The relaxation can be viewed as a polyhedral relaxation of a linear ordering. Further, it is succinct, in that it only has a polynomial number of linear inequalities. Flow metrics provide a uniform framework for all the optimization problems associated with path metrics mentioned above. The cost functions for all these problems can be modeled using the variables of the relaxation.

To prove approximation guarantees for the problems, we use (extensions of) the rounding algorithm of [10] to map a fractional solution of the relaxation to a true linear ordering. While we are unable to improve any approximation bounds, our approach has the advantage of achieving the best-known approximation guarantees for all these problems using the same relaxation and essentially the same rounding algorithm for all the problems and varying only the objective function from problem to problem. This is in contrast to the current state of the literature where each problem warrants either a new relaxation or a new rounding or both.

While [10] also uses essentially the same rounding algorithm for the minimum linear arrangement and the minimum interval graph completion problems, the variables in the spreading metric are not readily suitable to express the objective functions for the pathwidth and minimum cut linear arrangement problems. The excluded $K_{r,r}$ algorithms for minimum linear arrangement and minimum interval completion for [10] can also be directly applied to flow metrics yielding an $O(\log\log n)$ approximation for these problems.

The new relaxation has some interesting structural properties as well. We prove that the flow distances are (approximately) polyhedral liftings of spreading distances. We do this by constructing a flow whose distances approximate any given set of spreading distances. As a result, we obtain a succinct description of the spreading polyhedron. In conclusion, we mention some directions for strengthening flow metrics that might lead to improved approximation guarantees for these and other problems.

2 A new relaxation for linear orderings

Consider a path on n vertices numbered $1, 2, \ldots, n$. The distance between u and v in the corresponding path metric is $|u - v|$.

Now imagine sending a flow of one unit from each vertex u to every vertex v to the right of u (i.e $v > u$). These flows satisfy the following properties:

(i) For each pair of vertices u, v, the value of the flow from u to v plus the flow from v to u is one.

(ii) For each triple u, v, w, the flow between u and v that goes through w, the flow between v and w that goes through u and the flow between w and u that goes through v sum to one.

Flow distances are a set of distances satisfying the above properties. A flow metric is a set of Flow distances that is also a metric, i.e., that satisfy the triangle inequality. To define this formally, let the variables $f_{i,j}^{u,v}$ represent the value of the flow from u to v on the edge (i, j). These variables satisfy flow conservation at every vertex except the source and the sink. Define a set of auxiliary variables

$$g_w^{u,v} = \sum_i f_{i,w}^{u,v}.$$

In other words, $g_w^{u,v}$ represents the flow from u to v that goes into the vertex w.

Definition 2.1 *A* flow metric *is a solution to the following linear program (FP), where f and g are flow variables as defined above.*

$$g_v^{u,v} + g_u^{v,u} = 1 \qquad \forall u, v \in V \tag{1}$$

$$(g_w^{u,v} + g_w^{v,u}) + (g_u^{v,w} + g_u^{w,v}) + (g_v^{w,u} + g_v^{u,w}) = 1 \qquad \forall u, v, w \in V \tag{2}$$

$$d_f(u,v) = \sum_{i,j} f_{i,j}^{u,v} \quad \forall u, v \in V \tag{3}$$

$$d_f(u,v) + d_f(v,w) \geq d_f(u,w) \quad \forall u, v, w \in V \tag{4}$$

Constraint (3) defines a set of directed distances that form the “metric” and constraint (4) imposes the triangle inequality on these distances. It would be natural to add an additional constraint on the flow variables that would ensure that the collection of all flows would conform to a path when all variables were integral. This additional constraints are not needed in our formulation, even though they could be also imposed.

An important property of a flow metric is that it satisfies the 1-dimensional spreading constraints [6,2].

Theorem 2.2 *For any subset S of vertices,*

$$\sum_{u,v \in S} d_f(u,v) \geq \binom{|S|}{3}.$$

Proof.

$$\begin{aligned}
\sum_{u,v \in S} d_f(u,v) &= \sum_{u,v \in S} \sum_{i,w} f_{i,w}^{u,v} \\
&\geq \sum_{u,v,w \in S} \sum_{i} f_{i,w}^{u,v} \\
&= \sum_{u,v,w \in S} g_w^{u,v} \\
&= \sum_{\{u,v,w\} \subset S} g_w^{u,v} + g_w^{v,u} + g_u^{v,w} + g_u^{w,v} + g_v^{w,u} + g_v^{u,w} \\
&= \sum_{\{u,v,w\} \subset S} 1 = \binom{|S|}{3}.
\end{aligned}$$

□

3 Rounding algorithms

Here we describe two variants of the rounding algorithm of Rao and Richa [10] which we will use either directly or as the main subroutine in our approximation algorithms for various NP-hard problems (described in the next section). The rounding algorithm takes as input the set of pairwise distances d_f defined by a flow metric and produces a linear ordering of the vertices.

Rounding Algorithm A

(1) Select a vertex v for which $\sum_{w \in G} d_f(v, w)$ is maximum.
(2) Assign edges to *levels*: There are $n-1$ levels called $L(1), L(2), \ldots, L(n-1)$. An edge $e = (i, j) \in E$ is included in level $L(d)$ if $d_f(v, i) < d \leq d_f(v, j)$.
(3) Partition levels into *buckets*: There are $\lceil \log n \rceil$ buckets numbered $1, \ldots, \lceil \log n \rceil$. A level $L(d)$ is assigned to bucket b if $2^b < |L(d)| \leq 2^{b+1}$.
(4) Select the largest bucket. Let $L(d_1), L(d_2), \ldots, L(d_r)$ be the levels in that bucket in the order of increasing distance from v. Then define H_i to be the subgraph induced by the vertices u that satisfy $d_i \leq d_f(v, u) < d_{i+1}$ (where $d_0 = 0$ and $d_{r+1} = n + 1$).
(5) Recurse on the subgraphs $H_0, H_1, H_2, \ldots H_r$. Return the ordering obtained by concatenating the orderings of the subgraphs.

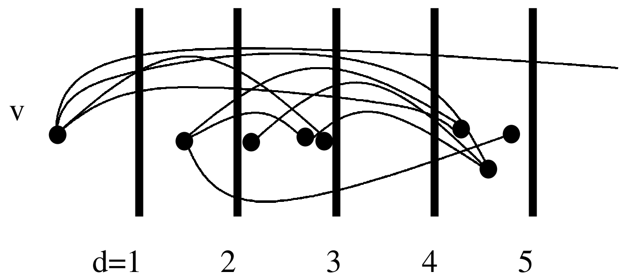

Fig. 1. All vertices are ordered with respect to their distances to v, and then the edges of the graph are distributed into levels. An edge starting at a vertex at distance less than d from v and ending at distance at most d is assigned to level d, for integer indices d.

Figure 1 illustrates an iteration of the algorithm. The vertices of the graph are ordered according to their distances from v and the edges of the graph can be distributed in levels.

The next algorithm differs from Algorithm A in that it assigns vertices to levels. In terms of Figure 1, a vertex will be assigned to a level d whenever it is at distance less than d from v and at least one of its neighbors is at distance greater than or equal to d.

Rounding Algorithm B

Use algorithm A with steps (2) and (4) replaced by the following:

2'. Assign vertices to levels: A vertex u is included in level $L(d)$ if $d_f(v,u) < d$ and there exists a vertex w adjacent to u such that $d \leq d_f(v,w)$.
4'. Select the largest bucket. Let $L(d_1), L(d_2), \ldots, L(d_r)$ be the levels in that bucket in the order of increasing distance from v. Delete the vertices in these levels from the graph. Then define H_i to be the subgraph induced by the remaining vertices u that satisfy $d_i \leq d_f(v,u) < d_{i+1}$.

4 Approximation algorithms and guarantees

Here we give algorithms for various linear ordering problems. All the algorithms start by solving the flow metric relaxation with a suitable objective function. We prove that the final solution found is within the best known approximation factor of the optimum for each of these problems.

4.1 Minimum Linear Arrangement

Given a graph $G = (V, E)$ the minimum linear arrangement problem asks for an ordering $v_1, v_2, \cdots, v_n$ of the vertices of G such that the sum of the lengths of the edges of G is minimized, where the length of an edge (v_i, v_j) is $|i - j|$.

The problem can be expressed in terms of a flow metric as obtaining an ordering that minimizes the sum of the distances $d_f(i,j)$ for edges $(i,j) \in E$. The next algorithm finds a linear arrangement of cost $O(\log n)$ times the minimum by rounding this flow metric, and matches the approximation factor of [10]. The proof is essentially the same.

Algorithm

(1) Minimize $\sum_{(i,j)\in E} d_f(i,j)$ over (FP) to get a set of distances d_f.
(2) Apply rounding algorithm A.

Theorem 4.1 *The algorithm finds an $O(\log n)$ approximation to the minimum linear arrangement.*

Proof. The algorithm removes certain edges of the graph in each iteration, and solves the problem recursively in each subgraph H_i. The total cost of the final solution is obtained by comparing the cost of the edges that are removed in each iteration with the difference in cost between the relaxation and the relaxation restricted to the components obtained by the removal of these edges. We show an $O(\log n)$ ratio between these two values, so that in fact the final solution is within $O(\log n)$ times the cost of the original relaxation and thus also a factor of at $O(\log n)$ times the optimal value.

Let bucket k be the bucket that was selected in the first iteration. Consider the set of edges removed in the first iteration of the algorithm, i.e., those edges that are in levels belonging to bucket k. First we bound the cost of these edges in the final ordering. For each component H_i, the number of edges that cross H_i is at most the number of edges in levels $L(d_i)$ and $L(d_{i+1})$. This is at most $2 \cdot 2^{k+1}$. In the final ordering the contribution of H_i to the lengths of these edges is at most the number of vertices in H_i. Summing the contribution of all H_i's we get that the total cost of these edges is at most

$$\sum_i 2 \cdot 2^{k+1} |H_i| = 2^{k+2} n.$$

Next we bound the contribution of these edges to the cost of the relaxation. Let (u, w) be any edge in E. Its contribution to the cost of the relaxation is $d_f(u, w) + d_f(w, u)$. If (u,w) belongs to levels $d_i, \ldots, d_j$ then $d_f(v, u) < d_i \leq d_j \leq d_f(v, w)$ and by the triangle inequality

$$d_f(u, w) \geq d_f(v, w) - d_f(v, u) \geq d_j - d_i.$$

By constraint (1) of (FP), $d_f(u, w) + d_f(w, u) \geq 1$ and so the edge (u,w) contributes at least $\max\{1, d_j - d_i\}$ to the objective function. In our analysis it will suffice to say that this is at least $(d_j - d_i + 1)/2$, or in other words, at least half the number of levels crossed by the edge (u, w), rounded up.

Theorem 2.2 guarantees that there exists at least one vertex v in G with the property that

$$\sum_{u \in V} d_f(v, u) \geq \frac{(n-1)(n-2)}{6}.$$

This implies that there is an edge (v, u) such that $d_f(v, u)$ is at least $(n-2)/6$ and hence there are at least as many levels. Since there are at most $\log n$ buckets the number of levels in the largest bucket is at least $(n-2)/6 \log n$.

Thus, when we add the contributions of all edges "cut" in the first iteration we get at least:

$$\sum_{\text{edges cut}} \frac{\#\text{levels crossed by edge}}{2} \geq \frac{(\text{min }\#\text{edges in a level})(\#\text{levels in chosen bucket})}{2}$$
$$\geq \frac{1}{2} \cdot 2^k \cdot \frac{(n-2)}{6 \log n} = \frac{2^k (n-2)}{12 \log n}.$$

As desired, this is a factor of $O(\log n)$ smaller than the actual cost of these edges in the rounded solution. □

4.2 Minimum Cut Linear Arrangement

Given a graph $G = (V, E)$ the minimum cut linear arrangement problem consists of obtaining a linear ordering $v_1, v_2, \ldots, v_n$ for the vertices of G that minimizes the maximum edge cut between the first i and the last $n-i$ vertices in the ordering. In terms of flow metrics, this can be expressed as the maximum over v of the sum of flows through v between all pairs of vertices i and j that are adjacent in G. The algorithm given here finds a solution of cost $O(\log^2 n)$ times the minimum, which matches the approximation factor due to [6].

Algorithm

(1) Minimize $\max_{v\in V} \sum_{(i,j)\in E} \left(\sum_{u\in V} f_{u,v}^{i,j}\right)$.
(2) Apply rounding algorithm A producing an ordering $v_1, v_2, \ldots, v_n$.
(3) Select an index i in the ordering such that $n/4 \leq i \leq 3n/4$ and for which the cost of the cut $(\{v_1, v_2, \ldots, v_i\}, \{v_{i+1}, \ldots, v_n\})$ is minimum.
(4) Recurse on the subgraphs induced by $v_1, \ldots, v_i$ and $v_{i+1}, \ldots, v_n$.

Theorem 4.2 *The algorithm finds an $O(\log^2 n)$ approximation to the minimum cut linear arrangement.*

Proof.

We note that

$$\max_{v\in V} \sum_{(i,j)\in E} \left(\sum_{u\in V)} f_{u,v}^{i,j}\right) \geq \frac{1}{n} \cdot \sum_{v\in V} \sum_{(i,j)\in E} \left(\sum_{u\in V} f_{u,v}^{i,j}\right) = F/n$$

So F/n as defined above is a lower bound on the objective function for the minimum cut linear arrangement. Given an ordering for the vertices of a graph, the sum of all the $n-1$ cuts of the form $(\{v_1, \ldots, v_i\}, \{v_{i+1}, \ldots, v_n\})$ is exactly the cost of the linear arrangement $v_1, \ldots, v_n$. Rounding algorithm A finds an ordering whose linear arrangement cost is $O(F \log n)$. Given an ordering for the vertices of a graph, the sum of all the $n-1$ cuts of the form $(\{v_1, \ldots, v_i\}, \{v_{i+1}, \ldots, v_n\})$ is exactly the cost of the linear arrangement $v_1, \ldots, v_n$. Hence the ordering found has average cut size $O(F \log n/n)$. When we restrict our attention only half the cuts they must also have an $O(F \log n/n)$ average cut size. In particular consider the cuts at the middle $n/2$ vertices. The smallest cut in the set cannot be larger then the average cut size and thus has an $O(F \log n/n)$ size.

Each iteration of the algorithm partitions the graph into subgraphs with at most $3n/4$ vertices while incurring a cost of $O(\log n)$ times the optimum for the minimum cut linear arrangement relaxation. The algorithm terminates in $O(\log n)$ iterations with a total cost of $O(\log^2 n)$ times the optimum. □

This algorithm can be used with any minimum linear arrangement algorithm instead of algorithm A, and will produce a solution for the minimum cut linear arrangement problem with an approximation factor of $O(\log n)$ times the approximation factor of the minimum linear arrangement algorithm used.

4.3 Minimum Interval Graph Completion

The minimal interval graph completion problem consists of obtaining an interval graph I from an input graph G by adding the smallest possible number of edges to make G into an interval graph. This is equivalent to obtaining an order $v_1, v_2, \cdots, v_n$ for the vertices of G which minimizes the total number of edges of the graph I that is obtained from G by inserting edges (v_i, v_j) in G whenever there exists an edge $(v_i, v_k) \in E$ such that $i < j < k$. In terms of a flow metric the number of edges of I can be expressed as the sum over all vertices of the maximum of the directed distances from that vertex to its neighbors in G.

Algorithm

(1) Minimize $\sum_{i \in V} \max_{j:(i,j) \in E} d_f(i, j)$ over (FP) to get a set of distances d_f.
(2) Apply rounding algorithm B.

Theorem 4.3 *The algorithm finds an $O(\log n)$ approximation to the minimum interval graph completion.*

Proof. Similar to the proof of Theorem 4.1, we start by bounding the cost in the final ordering of the vertices cut in the first iteration of the algorithm by separately bounding the contribution of each component H_i.

Let bucket k be the one that was choosen in the first iteration of the algorithm. Each subgraph H_i can contribute at most $|H_i|$ to the cost of the vertices that are in levels d_i and d_{i+1} so that the total cost of the vertices in the cuts is at most

$$\sum_i 2^{k+2} |H_i + d_i| \leq 2^{k+2} n.$$

Next we bound the contribution of these vertices to the cost of the relaxation. The total number of buckets is at most $\log n$ and the number of levels in the largest bucket is at least $(n-2)/6\log n$ by the same arguments used in the proof of Theorem 4.1.

Let u be vertex belonging to levels $d_i, \ldots, d_j$. Then there exists a vertex w adjacent to u such that $d(v,u) < d_i \leq d_j \leq d(v,w)$. Applying the triangle inequality and constraint (1) of (FP) we arrive at a total cost associated with all vertices belonging in the cut of at least

$$\frac{1}{2} \cdot 2^k \cdot \frac{(n-2)}{6\log n} = \frac{2^k(n-2)}{12\log n}.$$

Again, the cost of the vertices in the relaxation, and in fact the reduction of the value of the relaxation by the removal of these vertices is a factor of $O(\log n)$ smaller than the cost that these vertices have in the solution. Added over all the recursive subproblems, the cost of the solution is a factor of $O(\log n)$ of the cost of the relaxation. $\square$

4.4 Minimum Pathwidth

Given a graph $G = (V, E)$ the minimum pathwidth problem consists of obtaining an ordering $v_1, v_2, \ldots, v_n$ that minimizes over all v_i the maximum number of vertices in $\{v_1, v_2, \ldots, v_{i-1}\}$ which have neighbors in $\{v_{i+1}, \ldots, v_n\}$.

In terms of a flow metric this can be expressed as the minimum of the maximum over all vertices v of the number of vertices i that have flow through v to one of its neighbors j. The next algorithm matches the approximation factor of the best previously know algorithm for the minimum pathwidth problem due to Bodlaender et al. [3].

Algorithm

(1) Minimize $\max_{v\in V} \sum_{i\in V} \max_{j:(i,j)\in E} g_v^{i,j}$.
(2) Apply rounding algorithm B to get an ordering $v_1, v_2, \ldots, v_n$
(3) Select an index i in the ordering such that $n/4 \leq i \leq 3n/4$ and for which the number of vertices before v_i with neighbors after v_i is minimized.
(4) Recurse on the subgraphs induced by $v_1, \ldots, v_{i-1}$ and $v_{i+1}, \ldots, v_n$.

Theorem 4.4 *The algorithm finds an $O(\log^2 n)$ approximation to the minimum pathwidth.*

Proof. We note that we can relate the cost of this problem with the average width at each vertex:

$$\begin{aligned}\max_{v\in V}\sum_{i\in V}\max_{j:(i,j)\in E} g_v^{i,j} &\geq \frac{1}{n}\cdot\sum_{v\in V}\sum_{i\in V}\max_{j:(i,j)\in E} g_v^{i,j}\\ &\geq \frac{1}{n}\cdot\sum_{i\in V}\max_{j:(i,j)\in E}\sum_{v\in V} g_v^{i,j}\\ &= \frac{1}{n}\cdot\sum_{i\in V}\max_{j:(i,j)\in E} d_f(i,j) = F/n.\end{aligned}$$

With F as defined above, rounding algorithm B can round the flow metric to a linear ordering whose interval graph completion cost is $O(F\log n)$. The average width for the ordering produced is $O(F\log n/n)$ so we can find a vertex in the middle $n/2$ vertices whose width is also $O(F\log n/n)$. Each of the $O(\log n)$ steps incurs a cost of $O(\log n)$ times the optimum value of the pathwidth relaxation thus yielding the result. $\square$

Just like the algorithm for minimum cut linear arrangement, this one can also have algorithm B replaced by another minimum interval completion algorithm so as to produce an ordering with an approximation factor of $O(\log n)$ times the approximation factor of the algorithm used.

5 The connection with spreading metrics

Theorem 2.2 shows that flow metrics provide a succinct description of (exponentially many) spreading constraints. Here we consider a stronger set of spreading constraints that have turned out to be useful for many applications. A set of distances $d(u,v)$ satisfies the *strong* spreading constraints if for any vertex $u\in V$,

$$\sum_{v\in S} d_f(u,v) \geq |S|^2/K$$

where K is a fixed constant [2,6].

Using one more set of constraints of polynomial size on the flow variables of a flow metric leads to flow distances satisfying the stronger spreading constraints. In other words, the projection of the flow metric polyhedron with the new constraints defined below is contained in the spreading metric polyhedron. In fact, the converse also holds approximately: for any set of spreading distances, there are corresponding flow distances that approximate the spreading distances to within a constant factor.

Consider a solution f to the linear program (FP) with the following additional constraint:

$$\sum_{i,j\in\{v,w,z\}} g_i^{u,j} + g_i^{j,u} \geq 1 \quad \forall u\in V, \quad v,w,z\in V\setminus u. \tag{5}$$

In terms of a path metric this constraint says that given a vertex u, and three other vertices v,w and z then either v, w or z must be in the path from between u one of the remaining vertices. This is enough to obtain strong spreading.

Theorem 5.1 *For any any subset S of vertices, and any vertex $u \in S$,*

$$\sum_{v \in S \setminus u} d_f(u,v) + d_f(v,u) \geq |S|(|S|-1)/6.$$

Proof.

$$\begin{aligned}
\sum_{v \in S \setminus u} d_f(u,v) + d_f(v,u) &= \sum_{v \in S \setminus u} \sum_{w \in V} g_w^{u,v} + g_w^{v,u} \\
&= \frac{1}{|S|-2} \sum_{v \in S \setminus u} \sum_{w \in V} \sum_{z \in S \setminus \{u,v\}} g_w^{u,v} + g_w^{v,u} \\
&\geq \frac{1}{|S|-2} \sum_{v,w,z \in S} \sum_{i,j \in \{v,w,z\}} g_i^{u,j} + g_i^{j,u} \\
&\geq \frac{1}{|S|-2} \binom{|S|}{3} = |S|(|S|-1)/6. \qquad \square
\end{aligned}$$

We proceed to show how to construct flow distances that approximate spreading distances. The distances obtained might not satisfy the triangle inequality even if the corresponding spreading distances do.

Theorem 5.2 *Let d be a set of distances satisfying the strong spreading constraints on a set V with n vertices, $1 \leq d(u,v) \leq n$ for all $u, v \in V$. Then there exist flow distances d_f defined on V such that for all $u, v \in V$,*

$$\frac{1}{2} d(u,v) \leq d_f(u,v) + d_f(v,u) \leq K \cdot d(u,v),$$

where K is the constant in the definition of the strong spreading property.

Proof. We construct $d_f(u,v)$ by first defining a unit flow from an arbitrary vertex u to all v in $V \setminus u$ so as to approximate $d(u,v)$, and set $d_f(v,u) = 0$. We then apply the same construction recursively to all $v \in V \setminus u$.

We construct a set of paths as follows. Order the vertices according to their distance from u. Ties are broken arbitrarily. Let $P = v_1, v_2, \ldots, v_n$ be the path with vertices in the order obtained. Let H be $\{v_{n/2}, \ldots, v_n\}$. We start by sending one unit of flow along P. This will modify this flow and reroute it through other paths via an iterative procedure. Each iteration starts by examining the current flow distances from each u to each vertex. We construct a modified path P' from P as follows: vertices $v \notin H$ at a distance of at least $\min\{d(u,v), n/4\}$ stay in the same positions in both P and P'. The remaining vertices are re-arranged. The vertices in H are inserted in P' in the first $n/2$ positions available. Finally the remaining vertices are assigned to the remaining positions in P', in the same order that they appeared in P. This is illustrated in figure 1. If we re-route a fraction p of flow from P to P' only the distances to the vertices that were "re-arranged" are affected. Vertices in H have their distances decreased. Vertices not in H that are re-arranged have their distances increased by $p \cdot n/2$. Reroute exactly as much flow as needed to make exactly one such vertex reach $d_f(u,v) = d(u,v)$. The process is repeated, until all vertices $v \notin H$ satisfy $d_f(u,v) \geq \min\{d(u,v), n/4\}$. At most 1/2 the flow from P will be re-routed before the process finishes.

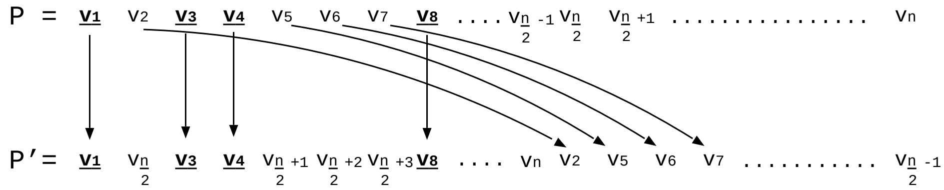

Fig. 2. The path P obtained by ordering the vertices according to their distance to a vertex u is used to route part of the flow. Paths P' are constructed from P by rearranging the vertices of P, so as to not move vertices v that are already at a distance $\min\{d(u,v), n/4\}$. The underlined vertices depict this set. The remaining vertices are rearranged, with the last $n/2$ vertices being moved to the first $n/2$ positions available.

As for the vertices in H, we now permute them in all paths that were constructed, and uniformly distribute the flow through each path among its permutation paths. This makes the distance from u to all vertices of H exactly the same.

At least half the flow goes through permutations of P, where these vertices are at an average distance of $3n/4$, while at most the other half of the flow flows though paths in which the average distance to these vertices is at least $n/4$, thus adding to a distance of at least $n/2$. Therefore for the final distances, we have

$$d_f(u,v) \geq \min\{d(u,v), \max\{d(u,v)/2, n/4\}\} \geq d(u,v)/2.$$

The strong spreading constraint implies that the maximum number of vertices at distance x from u is at most $K \cdot x$, so that in fact the distance from any vertex v to u in P is at most $K \cdot d(u,v)$. Since distance to any v is never increased beyond $d(u,v)$ by re-routing flow we have $d_f(u,v) \leq K \cdot d(u,v)$. $\square$

6 Further research

In our application of flow metrics we have used *independent* flows between pairs of vertices. This could be strengthened by using "capacity" variables $y_{i,j}$ on the edges (i, j) as upper bounds on edge flows and then restricting the sum of the capacities to be $n - 1$ (like a path).

One could also model trees in a similar manner, which might possibly lead to new approximation algorithms for tree-ordering problems, such as the minimum treewidth problem and the minimum chordal completion problem.

References

[1] Y. Bartal, "Probabilistic Approximation of Metric Spaces and its Algorithmic Applications," Proc. of the 37th Ann. IEEE Symp. on Foundations of Computer Science, 184-193, 1996.

[2] A. Blum, G. Konjevod, R. Ravi and S. Vempala, "Semi-Definite Relaxation for Minimum Bandwidth and other Vertex-Ordering Problems," *Theoretical Computer Science*, 235 (2000), 25-42. Preliminary version in *Proc. 30th ACM Symposium on the Theory of Computing*, Dallas, 1998.

[3] H. L. Bodlaender, J. R. Gilbert, H. Hafsteinsson, T. Klok, "Approximating Treewidth, Pathwidth, and Minimum Elimination Tree Height," Journal of Algorithms 18 (1995) 238-255.

[4] M. M. Deza and M. Laurent, *Geometry of cuts and metrics*, Springer-Verlag, 1997.

[5] J. Dunagan and S. Vempala, "On Euclidean embeddings and bandwidth minimization," *Proc. of the 5th Intl. Symp. on Randomization and Approximation techniques in Computer Science*, 229-240, 2001.

[6] G. Even. J. Naor, S. Rao and B. Schieber, "Divide-and-conquer approximation algorithms via spreading metrics," *Proceedings of the 35th Annual Conference on Foundations of Computer Science*, 62–71, 1995.

[7] M. Grötschel, L. Lovász, A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer, 1988.

[8] M. Goemans and D. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *JACM*, 42: 1115–1145, 1995.

[9] T. Leighton and S. Rao, "Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms", Journal of the ACM *(JACM)*, 46:6, 787–832, 1999.

[10] S. Rao and A. Richa, "New Approximation Techniques for Some Ordering Problems," Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 211-218, 1998.