A GEOMETRIC APPROACH TO BETWEENNESS *

BENNY CHOR† AND MADHU SUDAN‡

Abstract. An input to the *betweenness* problem contains m constraints over n real variables (points). Each constraint consists of three points, where one of the point is specified to lie inside the interval defined by the other two. The order of the other two points (i.e., which one is the largest and which one is the smallest) is not specified. This problem comes up in questions related to physical mapping in molecular biology. In 1979, Opatrny [14] showed that the problem of deciding whether the n points can be totally ordered while satisfying the m betweenness constraints is NP–complete. Furthermore, the problem is MAX SNP complete, and for every $\alpha > 47/48$ finding a total order that satisfies at least α of the m constraints is NP–hard (even if all the constraints are satisfiable). It is easy to find an ordering of the points that satisfies $1/3$ of the m constraints (e.g. by choosing the ordering at random).

This paper presents a polynomial time algorithm that either determines that there is no feasible solution, or finds a total order that satisfies at least $1/2$ of the m constraints. The algorithm translates the problem into a set of quadratic inequalities, and solves a semidefinite relaxation of them in $\mathcal{R}^n$. The n solution points are then projected on a random line through the origin. The claimed performance guarantee is shown using simple geometric properties of the SDP solution.

Key words. Approximation algorithm, Semidefinite programming, NP-completeness, Computational biology.

AMS subject classifications. 68Q20, 68Q25

1. Introduction.

An input to the *betweenness* problem consists of a finite set of n elements (or *points*) $S = \{x_1, \ldots, x_n\}$, and a finite set of m constraints. Each constraint consists of a triplet $(x_i, x_j, x_k) \in S \times S \times S$. A candidate solution to the betweenness problem is a total order $<$ on its points. A total order $x_{i_1} < x_{i_2} < \ldots < x_{i_n}$ satisfies the constraint (x_i, x_j, x_k) if either $x_i < x_j < x_k$ or $x_k < x_j < x_i$. That is, each constraint forces the second variable x_j to be between the two other variables x_i and x_k, but does not specify the relative order of x_i and x_k. The decision version of the betweenness problem is to decide if all constraints can be simultaneously satisfied by a total order of the variables.

In 1979, Opatrny [14] showed that the decision version of the betweenness problem is NP–complete. This problem arises naturally when analyzing certain mapping problems in molecular biology. For example, it arises when trying to order markers on a chromosome, given the results of a radiation hybrid experiment [6, 3]. A computational task of practical significance in this context is to find a total ordering of the markers (the x_i in our terminology) that maximizes the number of satisfied constraints. Indeed, betweenness is central in the recent software package RHMAPPER [15, 16]. At the heart of this package is a method for producing the order of *framework markers*, based on betweenness constraints (obtained from a statistical analysis of the biological data). Slonim *et. al.* successfully employ two greedy heuristics for solving the betweenness problem.

*A preliminary version of this paper appeared in the proceedings of the third annual European Symposium on Algorithms, ESA '95, Lecture Notes in Computer Science 979, Paul Spirakis (Ed.), Springer, September 1995, pp. 227–237.

†Dept. of Computer Science, Technion, Haifa 32000, Israel (benny@cs.technion.ac.il). Research partially supported by Technion V.P.R. funds.

‡Laboratory for Computer Science, MIT, 545 Technology Square, Cambridge, MA 02139 (madhu@lcs.mit.edu). Part of the work was done when this author was at the IBM Thomas J. Watson Research Center.

Opatrny gave two reductions in his proof of NP–completeness. One of these reductions is from 3SAT. Following his construction, we show, in Section 2, an approximation preserving reduction from MAX 3SAT. This implies that there exists an $\varepsilon > 0$, such that finding a total order that satisfies at least $m(1-\varepsilon)$ of the constraints (even if they are all satisfiable) is NP–hard. In particular this holds for every $\varepsilon < 1/48$ (see Corollary 2.5). On the other hand, it is easy to find a total order that satisfies one third of the m constraints (even if they are not all satisfiable). Simply arrange the points in a random order along the line. The probability that a specific constraint (x_i, x_j, x_k) is satisfied by such a randomly chosen order is 1/3, since exactly two of the six permutations on i, j, k have j in the middle. Thus the expected number of constraints satisfied by a random order is at least a third of the m constraints. On the other hand, it is easy to construct examples where at most $m/3$ constraints are satisfiable. Thus to achieve better approximation factors, one needs to be able to recognize instances of the betweenness problems that are not satisfiable.

We present a polynomial time algorithm that either determines that there is no feasible solution, or finds a total order that satisfies at least 1/2 of the m constraints. Our algorithm translates the problem into a set of quadratic inequalities, and solves a semidefinite programming relaxation (SDP) of them in $\mathcal{R}^n$. Let $v_1, \ldots, v_n \in \mathcal{R}^n$ be a feasible solution to the SDP, where each v_i corresponds to the real variable x_i. The n solution points are then projected on a random line through the origin. We show that if "x_j between x_i and x_k" is one of the betweenness constraints, then the angle between the lines $v_i v_j$ and $v_k v_j$ (in $\mathcal{R}^n$) is obtuse. Using this property, we prove that the random projection satisfies each constraint with probability at least 1/2. This gives a randomized algorithm with the claimed performance guarantee. Next, we show how to derandomize the algorithm. In addition, we demonstrate that our analysis of the semidefinite program is tight. There is an infinite family of inputs to the betweenness problem, such that the resulting SDP is feasible, but any total order of the variables satisfies at most $1/2 + o(1)$ of the m constraints.

Our use of semidefinite programming is inspired by the recent success in using this methodology to find improved approximation algorithms for several optimization problems. The applicability of SDP in combinatorial optimization was demonstrated by Grötschel, Lovász and Schrijver [7] to show that the Theta function of Lovász [12] was polynomial time computable. This application was then turned into exact coloring and independent set finding algorithms for perfect graphs. The use of SDP in approximation algorithms was innovated by the work of Goemans and Williamson [5] who broke longstanding barriers in the approximability of MAX CUT and MAX 2SAT by their SDP based algorithm. Further evidence of the applicability of the SDP approach is provided by the works of Karger, Motwani and Sudan [10], who use it to approximate graph coloring, Alon and Kahale [1] (independent set approximation) and by Feige and Goemans [4] (improvements to MAX 2SAT).

Thus the semidefinite programming method has now been used successfully to solve many optimization problems — exactly and approximately. However all the cases where SDP has been used to find approximation algorithms seem to be essentially partition problems (MAX CUT, Coloring, Multicut etc.). Our solution seems to be (to the best of our knowledge) the only case where SDP has been used to solve an ordering problem. This syntactic difference between ordered structures and unordered ones, and the ability of SDP to help optimize over both, offers critical additional evidence on the power of the SDP methodology.

The remaining of this paper is organized as following: Section 2 presents the

approximation preserving reduction from MAX 3SAT, as well as other observations about the betweenness problem. Semidefinite programming is briefly reviewed in Section 3. The algorithm is presented in Section 4. Section 5 shows the tightness of our analysis. Finally, Section 6 contains some concluding remarks and open problems.

2. Preliminaries. We start this section with some preliminary observations about the betweenness problem. We begin by defining the notion of an approximate solution to the betweenness problem and analyzing the complexity of finding such a solution.

Definition 2.1. *Given an instance of the betweenness problem on m constraints and $\alpha \leq 1$, an* α-approximate solution *is one that satisfies at least αk constraints, where k is the maximum number of constraints satisfied by any solution. For $\alpha \leq 1$, the α-approximation (version of the betweenness) problem is the task of finding an α-approximate solution for every instance. An algorithm which solves such a problem is said to be an α-approximation algorithm. For $\alpha \leq 1$, the α-approximation problem for* satisfiable *instances is the task of finding a total order that satisfies αm constraints, or determining that the instance is not satisfiable. An algorithm which solves this problem is an α-approximation algorithm for satisfiable instances.*

The complexity of solving the betweenness problem exactly (i.e., for $\alpha = 1$) is well-settled. Opatrny [14] has shown that it is NP-hard to decide if a given instance of the betweenness problem is satisfiable. We now turn our attention to the complexity of the problem for $\alpha < 1$. We first present a hardness result based on a simple reduction from MAX CUT, due to Michel Goemans. An instance of the MAX CUT problem is an undirected graph. The goal of the problem is to find a partition $(S, \bar{S})$ of the vertex set so as to maximize the number of edges with one endpoint in S and one in $\bar{S}$. This problem was shown to be hard to approximate to within some factor $\alpha < 1$ by Arora et al. [2]. The best result known to date, due to Håstad [9] (see also Trevisan et al. [17]), is that α-approximating MAX CUT is NP-hard for every $\alpha > 16/17$.

Proposition 2.2. *For every α, the α-approximation version of the MAX CUT problem reduces to the α-approximation version of the betweenness problem.*

Proof. Given an instance G of the MAX CUT problem, we create an instance of the betweenness problem as follows: For every vertex v_i in the graph, create a point p_i. In addition we introduce one special point s. For every edge (v_i, v_j) in the graph, we introduce the betweenness constraint (p_i, s, p_j) (i.e., s is between p_i and p_j). Now, given a cut $(S, \bar{S})$ in the graph that has k edges crossing the cut any ordering that places the points corresponding to the vertices in S to the left of s and the rest of the points to the right of s is an ordering that satisfies k of the betweenness constraints. In the reverse direction, any ordering of the points that satisfies k betweenness constraints can be converted into a cut with k edges crossing the cut, by letting S be the set of vertices corresponding to points to the left of s. Thus the optima of the two problems are exactly equal; furthermore, given an α-approximate solution to the betweenness instance, we can construct an α-approximate solution to the MAX CUT instance. Thus an α-approximation algorithm for the betweenness problem yields an α-approximation algorithm for the MAX CUT problem. ∎

Corollary 2.3. *The α-approximation version of the betweenness problem is NP-hard for $\alpha > 16/17$.*

While the above reduction provides some insight about the hardness of the betweenness problem on general instances, it does not quite provide a hardness result for the problem of interest to us. This is because the instances of the betweenness

problem that we typically consider are fully satisfiable. In the reduction above, the only instances of the MAX CUT problem that reduce to fully satisfiable instances of betweenness are when the input graph is bipartite. But in such cases it is easy to find the MAX CUT, and thus the instance of betweenness produced are not necessarily hard.

In what follows we present an approximation preserving reduction from MAX 3SAT to the betweenness problem. This reduction follows Opatrny's original reduction and addresses the α-approximation problem for satisfiable instances. It is well-known that there exists a constant $\varepsilon > 0$ such that the $(1-\varepsilon)$-approximation version of the MAX 3SAT problem is NP-hard. The best results known to date, due to Håstad [9], show this is true for every $\varepsilon < 1/8$. Based on our reduction we conclude that there exists a constant $\varepsilon' > 0$ such that finding an ordering that satisfies $(1-\varepsilon')$ fraction of the constraints in a *satisfiable* instance of the betweenness problem is NP-hard.

PROPOSITION 2.4. *For every $\varepsilon > 0$, the $(1-\varepsilon)$-approximation version of the MAX 3SAT problem on satisfiable instances reduces to the $(1-\varepsilon/6)$-approximation version of the betweenness problem on satisfiable instances.*

Proof. Given a 3-CNF formula ϕ on n variables and m clauses, we construct an instance I of the betweenness problem on $2+n+5m$ points with $6m$ constraints, such that: for every ℓ, there exists a total order satisfying $5m+\ell$ of the betweenness constraints in I if and only if there exists an assignment satisfying ℓ of the clauses in ϕ. The reduction proceeds as follows: For each Boolean variable x_i of ϕ, we add a point p_i to I. In addition we create two special points T and F. Without loss of generality, we consider orderings where T is to the right of F. An ordering of the points p_i, T and F is supposed to imply a truth assignment as follows: If p_i is to the left of F then it is false, if it is to the right of F then it is true. This interpretation will also apply to the additional "clause points" that are introduced in the rest of the construction.

Given a clause C_j, say $C_j = x_1 \vee \overline{x_2} \vee x_3$, we create five points $q_j^{(1)}$, $q_j^{(\bar{2})}$ and $q_j^{(3)}$ and $r_j^{(12)}$ and $r_j^{(123)}$. The points q_j are supposed to represent the assignment to the literals in the clause. For each literal in the clause, we include a constraint that forces the variable to be assigned consistently with the literal. We do so with the constraints F is between p_2 and $q_j^{(\bar{2})}$, whereas $q_j^{(1)}$ is between p_1 and F and $q_j^{(3)}$ is between p_3 and F. Thus, for example, an assignment satisfies $q_j^{(\bar{2})}$ if and only if it falsifies p_2. The points $r_j^{(12)}$ and $r_j^{(123)}$ are supposed to represent the OR of the first two and three literals in the clause, respectively. This is enforced with the betweenness constraints, $r_j^{(12)}$ is between $q_j^{(1)}$ and $q_j^{(\bar{2})}$; and $r_j^{(123)}$ is between $r_j^{(12)}$ and $q_j^{(3)}$. So, for example, if both literal points $q_j^{(1)}$ and $q_j^{(\bar{2})}$ are false, and $r_j^{(12)}$ is between $q_j^{(1)}$ and $q_j^{(\bar{2})}$ then $r_j^{(12)}$ must be false, while if at least one of the literal points is true, then $r_j^{(12)}$ can be placed so that it is true (to the right side of F). Lastly we add a betweenness constraint that attempts to ensure that a clause is assigned true. This is done with the constraint $r_j^{(123)}$ is between F and T.

Thus corresponding to each clause we have 6 betweenness constraints. Consider an assignment to the variables in ϕ satisfying ℓ clauses out of m. Without loss of generality assume that the assignment sets $x_1, \ldots, x_k = \mathsf{false}$ and $x_{k+1}, \ldots, x_n = \mathsf{true}$. Order the points p_i and T and F as follows:

$$p_1 \cdots p_k \; F \; p_{k+1} \cdots p_n \; T.$$

For j going from 1 to m, the literal points $q_j^{(1)}$, $q_j^{(2)}$ and $q_j^{(3)}$ are then placed between p_k and F or between F and p_{k+1}, depending on their truth value. (A true literal is placed between F and p_{k+1} while a false literal is between p_k and F.) Finally the points $r_j^{(12)}$ and $r_j^{(123)}$ are placed as far to the right as possible subject to the betweenness constraints. This tends to make $r_j^{(123)}$ lie between F and T if any one of the literals in the jth clause is true. This arrangement always satisfies at least five of the betweenness constraints associated with the k-th clause. The only constraint it may not satisfy is the constraint "$r_j^{(123)}$ is between F and T": and this constraint is satisfied if and only if at least one of the literals in the j-th clause is true. Thus this ordering satisfies $5m + \ell$ of the betweenness constraints. Conversely it may be verified that if an arrangement of the points (again, with F left of T) satisfies $5m + \ell$ betweenness constraints, then the assignment that assigns `true` to all those variables whose corresponding points lie to the right of F, satisfies at least ℓ clauses in the formula ϕ. (There must be at least ℓ values of j for which the arrangement satisfies all 6 betweenness constraints involving q_j's and r_j's. For these values of j, the corresponding assignment satisfies the j-th clause.)

Thus given a 3-CNF formula ϕ with m clauses we have constructed a betweenness instance I with $m' = 6m$ constraints. Further more given an ordering satisfying $(1-\varepsilon)m'$ constraints, we can reconstruct an assignment satisfying at least $(1-\varepsilon)m' - 5m = m(1-6\varepsilon)$ clauses of ϕ. □

Corollary 2.5. *The α-approximation version of the betweenness problem on satisfiable instances is NP-hard, for every $\alpha > 47/48$.*

Next we show what can achieved by the obvious randomized algorithm for the betweenness problem.

The natural randomized algorithm for the betweenness problem arranges the points in a random order along the line. The probability that a specific constraint is satisfied by such a randomly chosen order is 1/3. Thus the expected number of constraints satisfied by a random order is at least a third of all the constraints. By the method of conditional probabilities one can find such order in polynomial time. Since this order satisfies 1/3rd of all constraints, it is within 1/3rd of the optimal ordering. The result is summarized below.

Proposition 2.6. *The 1/3-approximation version of the betweenness problem can be solved in polynomial time.*

Before going on to more sophisticated techniques for solving this problem, let us examine the main weakness of the above algorithm. We first argue that no algorithm can do better than attempting to satisfy a third of all given constraints. Consider an instance of the betweenness problem on three points with three constraints insisting that each point be between the other two. Clearly we can satisfy only one of the above three constraints, which proves the claim. Thus the primary weakness of the above algorithm is not in the (absolute) number of constraints it satisfies, but in the fact that it attempts to do so for every instance of the betweenness problem; even those that are obviously not satisfiable. Thus to achieve better approximation factors, one needs to be able to recognize instances of the betweenness problems that are not satisfiable. But this is an NP-hard task. In fact, Corollary 2.5 indicates that one cannot even distinguish instances that are satisfiable from those for which an ε fraction of the constraints remain unsatisfied under any assignment. In what follows we use a semidefinite relaxation of our problem to distinguish cases that are not satisfiable from cases where at least 50% of the given clauses are satisfiable. We then go on to show that using this relaxation we can achieve a better approximation

than the naive randomized algorithm.

3. Semidefinite Programming. In this section we briefly introduce the paradigm of semidefinite programming. We describe why it is solvable in polynomial time. A complementary technique to that of semidefinite programming is the incomplete Cholesky decomposition. We describe how the combination allows one to find embeddings of points in finite-dimensional Euclidean space, subject to certain constraints.

DEFINITION 3.1. *For positive integers m and n, a semidefinite program is defined over a collection of n^2 real variables $\{x_{ij}\}_{i=1,j=1}^{n,n}$. The input consists of a set of mn^2 real numbers $\{a_{ij}^{(k)}\}_{i=1,j=1,k=1}^{n,n,m}$, a vector of m real numbers $\{b^{(k)}\}_{k=1}^{m}$ and a vector of n^2 real numbers $\{c_{ij}\}_{i=1,j=1}^{n,n}$. The objective is to find $\{x_{ij}\}_{i=1,j=1}^{n,n}$ so as to*

$$\begin{aligned} &\textit{maximize} && \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\ &\textit{subject to} && \\ \forall k \in \{1,\ldots,m\} \quad & && \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^{(k)}x_{ij} \le b^{(k)}. \\ &\textit{and} && \textit{The matrix } X = \{x_{ij}\} \textit{ is symmetric} \\ & && \textit{and positive semidefinite.} \end{aligned}$$

Recall that the following are equivalent ways of defining when a symmetric matrix X is positive semidefinite.

1. All the eigenvalues of X are non-negative.
2. For all vectors $y \in \mathcal{R}^n$, $y^T X y \ge 0$.
3. There exists a real matrix V such that $V^T \cdot V = X$.

It is well-known that the ellipsoid algorithm of Khaciyan [11] can be used to solve any semidefinite program approximately in the following sense: Given a parameter $\varepsilon > 0$, the algorithm runs in time polynomial in the input size and $\log(1/\varepsilon)$ and finds a feasible solution achieving an objective of at least optimum - ε (see, for instance [8]).

In order to use the semidefinite programming approach for solving combinatorial optimization problems, one more tool is useful. This is the ability to find a matrix V as guaranteed to exist in Part 3 of the definition of positive semidefiniteness above. The method that yields such a matrix is the incomplete Cholesky decomposition.

The matrix V can be used to interpret the solution obtained by the semidefinite programming problem geometrically. Interpret the columns of the $n \times n$ matrix V as n vectors $v_1, \ldots, v_n$ in $\mathcal{R}^n$. Now the variables x_{ij} of the matrix X simply correspond to the inner product of v_i and v_j. Thus a linear constraint on the x_{ij}'s is simply a linear constraint on the inner products of the vectors v_i's. And the objective function is simply a linear function on the inner products.

Thus the following provides an equivalent geometric interpretation of SDP:

> Find n vectors $v_1, \ldots, v_n$ so as to maximize the quantity $\sum_{i,j} c_{ij} < v_i, v_j >$,
> subject to the constraints $\sum_{i,j} a_{ij}^{(k)} < v_i, v_j > \ \le \ b^{(k)}$, for every $k \in \{1, \ldots, m\}$.

Alternately one can interpret SDP as solving an optimization problem that attempts to find n points in n-dimensional Euclidean space, subject to linear constraints

on the squares of the distance between the points. This is done by observing that the square of the distance between points v_i and v_j (denoted d_{ij}^2) is simply

$$< (v_i - v_j), (v_i - v_j) >=< v_i, v_i > + < v_j, v_j > -2 < v_i, v_j > \ .$$

Thus a linear inequality on the d_{ij}^2's is also a linear inequality on the inner products of the v_i's. (Actually the distance squared interpretation is equivalent to SDP since we can express $< v_i, v_j >$ as $(d_{i0}^2 + d_{j0}^2 - d_{ij}^2)/2$.)

From this interpretation of SDP we can solve any problem of the form:

> **Geometric SDP:** Embed n points in $\mathcal{R}^n$ such that the squares of the distance between the points, denoted d_{ij}, satisfy the constraints $\sum_{i,j} a_{ij}^{(k)} d_{ij}^2 \leq b^{(k)}$ while trying to maximize $\sum_{i,j} c_{ij} d_{ij}^2$. In the *$\varepsilon$-additive approximation* version the algorithm is allowed to return (for every feasible input) a solution such that each constraint is violated by at most ε, i.e., $\sum_{i,j} a_{ij}^{(k)} d_{ij}^2 \leq b^{(k)} + \varepsilon$, and the objective achieved is at least the optimum - ε.

In what follows we will use the last interpretation of SDP to solve the betweenness problem. In particular, we use the proposition below.

Proposition 3.2. *For every $\varepsilon > 0$, the ε-additive approximation version of the geometric SDP can be solved in time polynomial in the input size and* $\log(1/\varepsilon)$.

4. The Algorithm. The general idea of our algorithm is to express the betweenness constraints as a set of real quadratic inequalities. By considering an n–dimensional relaxation of the problem, we get an instance of semidefinite programming, and can find a feasible solution in $\mathcal{R}^n$ (if one exists). We study simple geometric properties of this solution set. We use them to argue that a projection of the set on a random line satisfies at least half the betweenness constraints (with high probability). Then we show how to derandomize the algorithm.

Consider a set of m betweenness constraints on n real variables $x_1, \ldots, x_n$. Suppose these constraints are satisfiable, and that $x_1 < x_2 < \ldots < x_n$ is a satisfying linear order. We can clearly embed the points in the unit interval, and assign $x_i = (i-1)/(n-1)$ $(i = 1, \ldots, n)$. Let x_i, x_j, x_k be a triplet such that x_j is required to be between x_i and x_k. For the assignment above, it is readily seen that $(x_i - x_j)^2 + (x_k - x_j)^2 < (x_i - x_k)^2$. Furthermore, the x's are at least $1/(n-1)$ apart and at most 1 apart. Thus, for every pair of distinct indices i, j, the x's satisfy the inequalities $1/(n-1)^2 \leq (x_i - x_j)^2 \leq 1$. This motivates the following geometric SDP relaxation for the betweenness problem.

Embed n points in $\mathcal{R}^n$ subject to the constraints

$$\begin{array}{ll} \frac{1}{(n-1)^2} \leq d_{ij}^2 \leq 1, & \forall\ i \neq j \\ d_{ij}^2 + d_{jk}^2 \leq d_{ik}^2, & \text{for every constraint } (x_i, x_j, x_k) \end{array} \qquad \text{(SDP1}(I)\text{)}$$

We strengthen this relaxation slightly before showing how to use it to find an approximate solution to the instance of the betweenness problem. Recall that the x's are at least $1/(n-1)$ apart and at most 1 apart. Therefore for any triple (x_i, x_j, x_k) the ratio between $(x_i - x_j)^2 + (x_j - x_k)^2$ and $(x_i - x_k)^2$ is maximized when x_i and x_k are extreme points (0 and 1), and x_j is as close as possible to one of them ($1/(n-1)$ or $(n-2)/(n-1)$). For these values, the ratio is

$$\left(\frac{1}{n-1}\right)^2 + \left(1 - \frac{1}{n-1}\right)^2 = 1 - \frac{2}{n-1} + \frac{2}{(n-1)^2} \ .$$

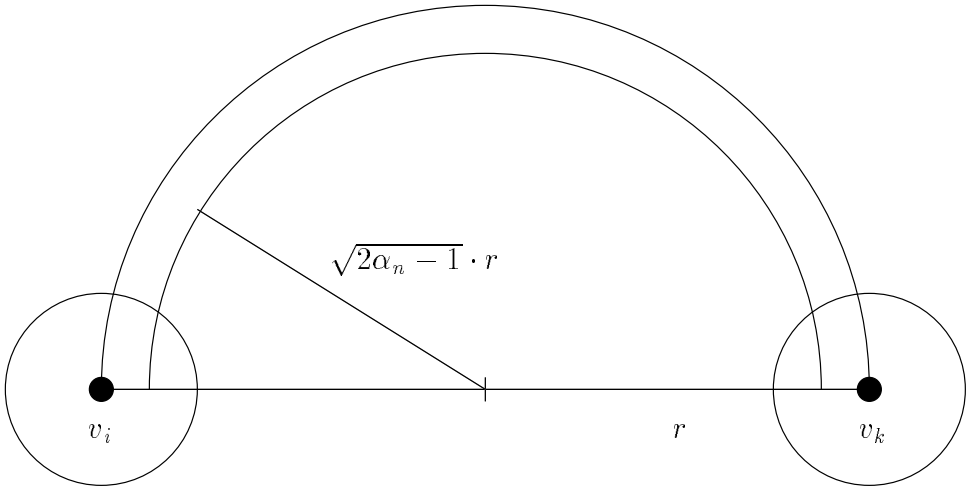

FIG. 1. *Possible location for the midpoint v_j*

Denote this value by α_n. Notice that $\alpha_n = 1 - 2/n + o(1/n)$ depends only on the number of variables.

We are now ready to set up our final SDP relaxation:

Embed n points in $\mathcal{R}^n$ subject to the constraints

$$\begin{aligned} &\tfrac{1}{(n-1)^2} \le d_{ij}^2 \le 1, && \forall\, i \neq j \\ &d_{ij}^2 + d_{jk}^2 \le \alpha_n d_{ik}^2, && \text{for every constraint } (x_i, x_j, x_k) \end{aligned} \qquad (\mathrm{SDP}(I))$$

The argument leading to the construction of the instance SDP(I) says that the SDP is feasible if the instance I is satisfiable and in fact there exists an embedding of the points in one dimension satisfying all the constraints. We summarize this below.

PROPOSITION 4.1. *For every instance I of the betweenness problem, if I is satisfiable, then the semidefinite program SDP(I) is feasible.*

As argued in Section 3 (see Proposition 3.2), we can use the ellipsoid algorithm to test the feasibility of SDP(I) and if it is feasible, to find an approximation of a feasible solution (if one exists). Let $v_1, \ldots, v_n \in \mathcal{R}^n$ be an approximately feasible solution, and let $v_i, v_j, v_k \in \mathcal{R}^n$ be a triplet that corresponds to a betweenness constraint. We first prove some geometric facts about the points v_i, v_j, v_k and then use this to design our approximation algorithm.

Consider any two dimensional plane through the points v_i, v_j, v_k. (If v_i, v_j, v_k are not collinear then this plane is unique, else we pick any such plane arbitrarily.) Let $2r$ be the distance between v_i and v_k $(1/(n-1) - \varepsilon \le 2r \le 1 + \varepsilon)$. In what follows we shall skip the term ε since it can be made arbitrarily small (and in particular, exponentially small in n).

We now consider the angle $\theta_{i,j,k} = \angle v_i v_j v_k$. We claim that this angle is obtuse (i.e., at least $\pi/2$). To see this, we project the points down to the two dimensional plane containing v_i, v_j and v_k. Furthermore, we rotate and translate the points so that $v_i = (-r, 0)$, $v_k = (r, 0)$ and $v_j = (x, y)$. Now we can use the explicit formulae $d_{ij}^2 = (x + r)^2 + y^2$, $d_{jk}^2 = (r - x)^2 + y^2$ and $d_{ik}^2 = 4r^2$. The constraint on these distances yields:

$$(x - r)^2 + y^2 + (x + r)^2 + y^2 \le 4\alpha_n r^2 \ ,$$

which implies

$$x^2 + y^2 \leq (2\alpha_n - 1)r^2 \ .$$

This means that v_j, the "midpoint" in the betweenness constraint, lies inside a ball of radius $r\sqrt{2\alpha_n - 1}$ whose center is the middle point $(v_i + v_k)/2$, and outside the two small balls of radius $1/(n-1)$ around v_i and v_k (see figure 1).

This proves that the angle $\theta_{i,j,k} = \angle v_i v_j v_k$ is indeed obtuse. The following claim proves a tighter bound on $\theta_{i,j,k}$.

Claim 4.2. *The angle $\theta_{i,j,k}$ satisfies $\theta_{i,j,k} \geq (1 + \Omega(1/n))\pi/2$.*

Proof. We apply the cosine rule

$$\begin{aligned}
\cos\theta_{i,j,k} = & \quad (d_{ij}^2 + d_{jk}^2 - d_{ik}^2)/(2d_{ij}d_{jk}) \\
= & (x^2 + y^2 - r^2)/\left(\sqrt{(x^2 + y^2 + r^2)^2 - 4r^2x^2}\right) \\
\leq & \quad (x^2 + y^2 - r^2)/(x^2 + y^2 + r^2) \\
\leq & \quad (\alpha_n - 1)/\alpha_n \\
< & \quad \alpha_n - 1 \\
= & \quad -\frac{2}{n} + \theta\left(\frac{1}{n^2}\right) \ .
\end{aligned}$$

Denoting $\theta_{i,j,k} = h + \pi/2$ and using the Taylor series expansion

$$\cos(h + \pi/2) = -h + \frac{h^3}{6} - \frac{h^5}{120} + \ldots$$

we get

$$-h + \theta(h^3) \ \leq \ -\frac{2}{n} + \theta\left(\frac{1}{n^2}\right)$$

so $h = \Omega(\frac{1}{n})$, namely $\theta_{i,j,k} \geq (1 + \Omega(1/n))\pi/2$. □

We are now ready to describe our algorithm. The algorithm proceeds by picking uniformly at random a line through the origin, and projecting the n points $v_1, \ldots, v_n$ on this random line. Let $x'_1, \ldots, x'_n$ be the n resulting points.

Claim 4.3. *Let $\theta_{i,j,k}$ denote the angle $\angle v_i v_j v_k$. Then the probability that x'_j lies between x'_i and x'_k equals $\theta_{i,j,k}/\pi$.*

Proof. Instead of considering an arbitrary line through the origin, we consider a parallel line that goes through the point v_j. This does not change the betweenness relation of the projections. Neither is this relation changed when considering the projection of this line on the two dimensional plane defined by v_i, v_j, v_k. Consider the section of the circle defined by the two lines that go through v_j and are perpendicular to the lines $v_i v_j$ and $v_k v_j$. It is not hard to see that only lines going through this section violate the betweenness constraint of the projections. This section occupies an angle of $\pi - \theta_{i,j,k}$ (see figure 2). The claim follows. □

Combining Claims 4.2 and 4.3, we get:

Corollary 4.4. *Suppose SDP(I) has a feasible solution. Then for any of the m constraints, the probability that x'_j lies between x'_i and x'_k is at least $1/2 + \Omega(1/n)$.*

As a consequence, the expected number of betweenness constraints satisfied by $x'_1, \ldots, x'_n$ is at least $m/2 + \Omega(m/n) = m(1/2 + \Omega(1/n))$. This yields the following lemma that forms a (weak) converse to Proposition 4.1.

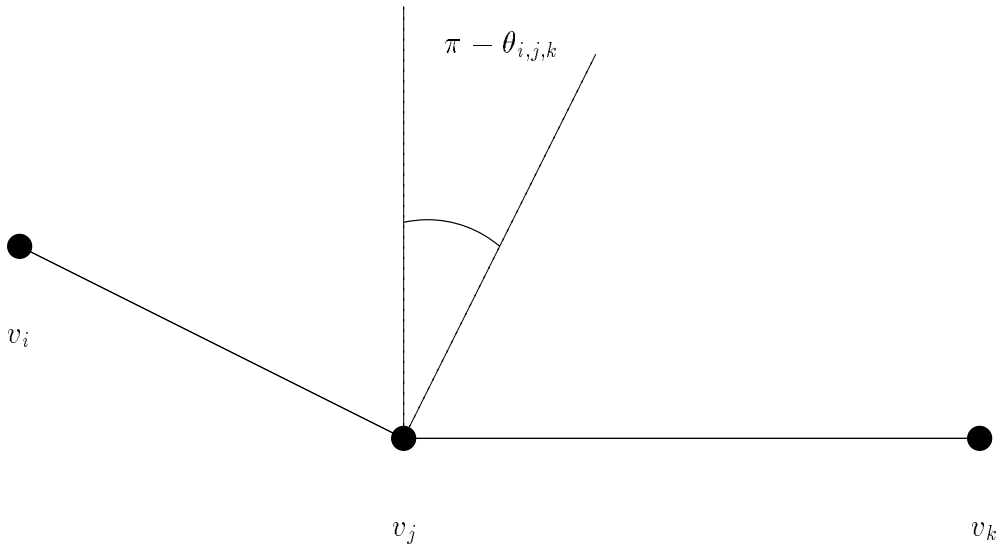

FIG. 2. *Lines going through the circular section violate the constraint.*

LEMMA 4.5. *For any instance I of the betweenness problem, if SDP(I) is feasible, then there exists a total order satisfying at least $m/2 + \Omega(m/n)$ of the betweenness constraints in I.*

Thus we get a randomized polynomial time algorithm that either finds that the constraints are infeasible, or generates a linear order that satisfies at least half the constraints.

We now outline a method for derandomizing our algorithm. Given an embedding of the betweenness problem, we can define a graph and an embedding of the graph in $\mathcal{R}^n$, such that the expected size of the MAX CUT found for this embedding of the graph equals the expected number of betweenness constraints that are satisfied by a random projection.

For every ordered pair of points (v_i, v_j) of the betweenness problem, introduce the vertex w_{ij} with embedding $v_i - v_j$. If i, j, k is a betweenness constraint, then put an edge between w_{ij} and w_{kj}. This defines the graph and its embedding.

Now consider any hyperplane through the origin that cuts across the edge between w_{ij} and w_{kj}. Let the slope of the normal to the hyperplane be the vector r. Assume w.l.o.g. that $r.w_{ij} < 0$ and $r.w_{jk} < 0$, then $r.v_i < r.v_j$ and $r.v_j < r.v_k$. Thus j lies between i and k. Conversely if projection onto the vector r satisfies the betweenness constraint for i, j, k; then the edge between w_{ij} and w_{jk} must be cut.

Mahajan and Ramesh [13] give a method to deterministically find a vector r whose cut value equals the expected cut value. They use this algorithm to derandomize the MAX CUT and Max 2SAT algorithm of Goemans and Williamson [5]. By using their algorithm we get a vector such that projection on to this vector satisfies as many constraints as the expected number satisfied by a random vector.

Remark: Observe that the above reduction is not a generic reduction from betweenness to MAX CUT. It uses the fact that the graph produced for the MAX CUT problem has a specified embedding, in order to map a solution of the MAX CUT problem to a solution of the betweenness problem.

We conclude the section by stating the main theorem of this paper.

Theorem 4.6. *The 1/2-approximation version of the betweenness problem can be solved in polynomial time. Specifically, there exists a polynomial time algorithm which takes as input an instance of the betweenness algorithm on n points and m constraints and either outputs "not feasible" or outputs a total order satisfying at least $m/2 + \Omega(m/n)$ constraints.*

5. Tightness of our analysis. In this section we show that our analysis of the semidefinite program is almost tight. We do so by exhibiting two families of instances of the betweenness problem on m constraints, such that the optimum value is at most $m(1/2 + o(1))$, but (a slight perturbation of) the SDP is nevertheless feasible.

The first example is related to the d-dimensional hypercube. For every integer $d > 1$, we construct the instance I_d as follows. I_d has 2^d points corresponding to the 2^d vertices of the d-dimensional hypercube. I_d has $m = \binom{d}{2}2^d$ constraints - one for every simple path of length 2 in the hypercube, with the betweenness constraint expecting the middle vertex of the path to be between the endpoints.

Consider a small perturbation of our SDP, where we set $d_{i,j}^2 + d_{j,k}^2 \leq d_{i,k}^2$ for each betweenness constraint. This SDP is clearly feasible — the natural embedding of the hypercube in d-dimensions (as a hypercube) ensures that every path of length 2 subtends an angle of 90^o at their midpoint.

Now consider a linear ordering of the points. Consider any point p and all the paths that have p as their midpoint: The number of such paths is $\binom{d}{2}$. Now let d_1 of the neighbors of p be on its left and d_2 of its neighbors be on its right (where $d_1 + d_2 = d$). The number of betweenness constraints expecting p to be in the middle that get satisfied is $d_1 d_2 \leq d^2/4$. Thus the fraction of betweenness constraints associated with any point that get satisfied is at most $(d^2/4)/(d(d-1)/2) = d/(2(d-1)) = 1/2 + 1/(2(d-1)) = 1/2 + o(1)$.

The second example, suggested to us by Michel Goemans, is related to the cuts in the complete graph K_n on n variables. For every integer $n > 1$, we construct the instance C_n as follows. C_n has $n + 1$ points, a "center point" v_0 and n "vertices" $v_1, \ldots, v_n$. C_n has $m = \binom{n}{2}$ constraints – one for every edge in the complete graph. For every $1 \leq i < k \leq n$, we have the betweenness constraint that v_0 is between v_i and v_k.

We now consider the following perturbation of our SDP, where $d_{i,j}^2 + d_{j,k}^2 \leq (1-1/n)d_{i,k}^2$ for each betweenness constraint. To see that this SDP is feasible, consider the following embedding: The vertex v_i is embedded as the point $(0, \ldots, 0, 1, 0, \ldots, 0)$, where the 1 occurs in the ith coordinate. The vertex v_0 is embedded as the point $(1/n, \ldots, 1/n)$. Observe that the distance between v_i and v_j is $\sqrt{2}$ and the distance between v_i and v_0 is $\sqrt{1 - 1/n}$. Thus for any two indices $i, k \neq 0$ the inequality $d_{i,0}^2 + d_{0,k}^2 \leq (1 - 1/n)d_{i,k}^2$, which corresponds to the betweenness constraints, is satisfied (in fact equality holds). Now in order to satisfy the SDP (recall that we required all pairwise distances to be at most 1) we simply scale down the simplex so that the distance between the vertices is 1, embed the center v_0 in the origin, and each vertex v_i in the corresponding simplex vertex. This embedding satisfies all the SDP constraints.

Again, any linear ordering of the $n + 1$ points induces a cut in the graph K_n (vertices to the left of v_0, vertices to the right of v_0). An edge corresponds to a satisfied betweenness constraint if and only if the edge is across the cut. Therefore the maximum number of satisfiable constraints equals the sized of a maximum cut in K_n, namely $(n/2)^2 = m(1/2 + o(1))$.

The advantage of this maximum cut example is that it shows tightness of the

analysis with respect to quadratic inequalities of the form

$$d_{i,j}^2 + d_{k,j}^2 \leq \beta_n d_{i,k}^2 \ ,$$

where $\beta_n = 1 - 1/n - o(1/n)$. Our original SDP has the form

$$d_{i,j}^2 + d_{k,j}^2 \leq \alpha_n d_{i,k}^2$$

where $\alpha_n = 1 - 2/n + o(1/n)$. By starting with the complete graph example, and padding it with extra dummy variables that do not take part in any constraint, we can construct an example where only $1/2 + o(1)$ of the constraints are satisfiable, yet the original SDP (with α_n) is feasible (in fact any $\gamma_n = o(1)$ can work here). It is not clear how to come up with a non-artificial construction, i.e., without padding, having these properties.

6. Concluding Remarks. We remark that metric information can be easily incorporated into our algorithm. As a simple example, suppose that for some of the constraints, we know not only that x_j is between x_i and x_k, but that it is exactly in the middle, namely $x_j = (x_i + x_k)/2$. In this case, we add the inequality

$$d_{i,j}^2 + d_{k,j}^2 \leq d_{i,k}^2/4$$

instead of

$$d_{i,j}^2 + d_{k,j}^2 \leq \alpha_n d_{i,k}^2 \ .$$

Any feasible solution will have v_j exactly in the middle of v_i and v_k, and the same holds with respect to the final projections.

Finally, notice that our formulation of the problem as SDP only tested for feasibility of the constraints. It is interesting to see if the inclusion of an appropriate objective function, and possibly of additional inequalities, can be used to improve the performance guarantee of the algorithm. Other approaches to the problem, possibly purely combinatorial ones, are also of interest.

Acknowledgments. We are grateful to Michel Goemans for providing us with the max cut example, and for helpful discussions on semi-definite programming. Many thanks to Amir Ben-Dor for numerous helpful discussions on the betweenness problem. We would also like to thank Ron Shamir for acquainting us with reference [14] and for useful discussions, Oded Goldreich and the anonymous referee for their comments on earlier versions of this paper, and Amos Beimel and Dan Pelleg for their expert advice on xfig.

REFERENCES

[1] N. ALON AND N. KAHALE, *Approximating the independence number via the θ-function*, Manuscript, August 1994.

[2] S. ARORA, C. LUND, R. MOTWANI, M. SUDAN AND M. SZEGEDY, *Proof Verification and Hardness of Approximation Problems*, Journal of the ACM, to appear. An extended abstract appears in Proc. 33rd Annual IEEE Symposium on Foundations of Computer Science, pp. 14–23, 1992.

[3] D. COX, M. BURMEISTER, E. PRICE, S. KIM, R. MYERS, *Radiation Hybrid Mapping: A Somatic Cell Genetic Method for Constructing High Resolution Maps of Mammalian Chromosomes*, Science, Vol. 250, 1990, pp. 245–250.

[4] U. FEIGE AND M. GOEMANS, *Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT*, Proceedings of the Third Israel Symposium on Theory and Computing Systems, Tel Aviv, Israel, 1995, pp. 182-189.

[5] M. Goemans and D. Williamson, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, Journal of the ACM, 42(6):1115–1145, November 1995.

[6] S. Goss and H. Harris, *New Methods for Mapping Genes in Human Chromosomes*, Nature, Vol. 255, 1975, pp. 680–684.

[7] M. Grötschel, L. Lovász and A. Schrijver, *The ellipsoid method and its consequences in combinatorial optimization*, Combinatorica, 1:169–197, 1981.

[8] M. Grötschel, L. Lovász and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, Berlin, 1987.

[9] J. Håstad, *Some optimal inapproximability results*, Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, El Paso, Texas, 1997, pp. 1-10.

[10] D. Karger, R. Motwani and M. Sudan, *Approximate graph coloring via semidefinite programming*, Journal of the ACM, to appear. Extended abstract in Proc. of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, 1994, pp. 2-13.

[11] L. Khaciyan, *A polynomial algorithm in linear programming*, (English translation appears in) Soviet Mathematics Doklady, vol. 20, pp. 191–194, 1979.

[12] L. Lovász, *On the Shannon capacity of a graph*, IEEE Transactions on Information Theory, IT-25:1–7, 1979.

[13] S. Mahajan and H. Ramesh, *Derandomizing Semidefinite Programming Based Approximation Algorithms*, Proceedings of the 36th Annual Symposium on Foundations of Computer Science, pp. 162-169, 1995.

[14] J. Opatrny, *Total Ordering Problem*, SIAM Journal on Computing, vol. 8 no. 1, February 1979, pp. 111–114.

[15] D. Slonim, L. Stein, L. Kruglyak, and E. Lander, *RHMAPPER: An interactive computer package for constructing radiation hybrids maps*, 1996. Available at `http://www.genome.wi.mit.edu/ftp/pub/software/rhmapper/`.

[16] D. Slonim, L. Stein, L. Kruglyak, and E. Lander, *Building Human Genome Maps with Radiation Hybrids*, Journal of Computational Biology, vol. 4 no. 4, Winter 1997, pp. 487–504.

[17] L. Trevisan, G.B. Sorkin, M. Sudan, and D.P. Williamson, *Gadgets, approximation, and linear programming*, Proceedings of the 37th Annual Symposium on Foundations of Computer Science, Burlington, Vermont, 1996, pp. 617-626.