A Unified Framework for Obtaining Improved Approximation Algorithms for Maximum Graph Bisection Problems

Eran Halperin[1] and *Uri Zwick*[1]

Department of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel.
E-mail: {heran,zwick}@post.tau.ac.il.

Abstract. We obtain improved semidefinite programming based approximation algorithms for *all* the natural maximum bisection problems of graphs. Among the problems considered are: MAX $\frac{n}{2}$-BISECTION – partition the vertices of a graph into two sets of equal size such that the total weight of edges connecting vertices from different sides is maximized; MAX $\frac{n}{2}$-VERTEX-COVER – find a set containing half of the vertices such that the total weight of edges touching this set is maximized; MAX $\frac{n}{2}$-DENSE-SUBGRAPH – find a set containing half of the vertices such that the total weight of edges connecting two vertices from this set is maximized; and MAX $\frac{n}{2}$-UNCUT – partition the vertices into two sets of equal size such that the total weight of edges that do *not* cross the cut is maximized. We also consider the directed versions of these problems, MAX $\frac{n}{2}$-DIRECTED-BISECTION and MAX $\frac{n}{2}$-DIRECTED-UNCUT. These results can be used to obtain improved approximation algorithms for the unbalanced versions of the partition problems mentioned above, where we want to partition the graph into two sets of size k and $n-k$, where k is not necessarily $\frac{n}{2}$. Our results improve, extend and unify results of Frieze and Jerrum, Feige and Langberg, Ye, and others.

1 Introduction

Goemans and Williamson [GW95] use semidefinite programming to obtain a 0.878-approximation algorithm for the MAX CUT problem: Given an undirected graph $G = (V, E)$ with nonnegative edge weights, partition the vertex set V into two sets S and $T = V - S$ such that the total weight of edges in the cut (S, T) is maximized. The MAX $\frac{n}{2}$-BISECTION problem is a variant of the MAX CUT problem in which the sets S and T are required to be of size $\frac{n}{2}$, where $n = |V|$ is assumed to be even. In other words, we are required to partition the vertices of the graph into two blocks of equal cardinality, such that the total weight of edges connecting vertices from different blocks is maximized. Frieze and Jerrum [FJ97] use extensions of the ideas set forth by Goemans and Williamson [GW95] to obtain a 0.651-approximation algorithm for the MAX $\frac{n}{2}$-BISECTION problem. Ye [Ye99] recently improved this result and obtained a 0.699-approximation algorithm for the problem. As one of our results here, we obtain another small improvement to the performance ratio that can be obtained for MAX $\frac{n}{2}$-BISECTION.

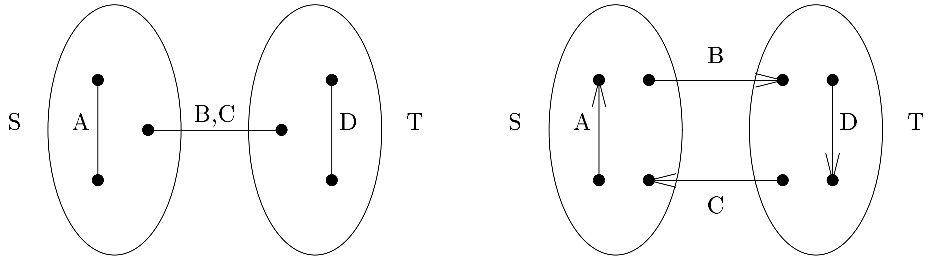

Fig. 1. Types of edges obtained by partitioning the vertices of a graph into two sets in the undirected case (on left-hand side) and the directed case (on right-hand side).

An improved approximation algorithm for the MAX $\frac{n}{2}$-BISECTION problem for *regular* graphs was recently obtained by Feige *et al.* [FKL00b].

Several graph bisection problems that are similar to MAX $\frac{n}{2}$-BISECTION problem were also considered in the literature. In Sect. 2, we consider all possible bisection problems for both undirected and directed graphs. For undirected graphs, there are three other natural bisection problems, which are the MAX $\frac{n}{2}$-VERTEX-COVER, the MAX $\frac{n}{2}$-DENSE-SUBGRAPH, and the MAX $\frac{n}{2}$-UNCUT problems. (These problems were defined in the abstract and will be defined again in Sect. 2). Approximation algorithms for these problems were obtained by Feige and Langberg [FL99], Ye and Zhang [YZ99] and Han *et al.* [HYZ00], [HYZZ00]. We obtain improved performance ratios for all three problems. We also consider maximum bisection problems of directed graphs, namely, the MAX $\frac{n}{2}$-DIRECTED BISECTION, which was considered by Ageev, Hassin and Sviridenko [AHS00], and the MAX $\frac{n}{2}$-DIRECTED-UNCUT, which to the best of our knowledge was not considered before. We obtain improved approximation algorithms also for these problems. A table comparing our results to the previously known results is given in Sect. 2.

Our results and techniques may be used to obtain improved approximation algorithms for the unbalanced version of all these problems. See Feige and Langberg [FL99] and Han, Ye and Zhang [HYZ00] for details.

To obtain our results we use a collection of old and new techniques. We show, for the first time, that adding the 'triangle inequalities' to the semidefinite relaxations of maximum bisection problems provably helps. To demonstrate this, we use a *global* argument, as opposed to local arguments used in previous results. We also introduce the somewhat counterintuitive idea of 'flipping' the bisection obtained by rounding the solution of the semidefinite programming relaxation.

2 Graph Bisection Problems

Let $G = (V, E)$ be a directed graph. Let $n = |V|$. Whenever needed, we assume that n is even. Let $S \subset V$ be such that $|S| = \frac{n}{2}$ and let $T = V - S$. The partition of the vertices of G into S and T partitions the edges of G into the four classes:

$$\begin{aligned} A &= \{\, (i,j) \in E \mid i,j \in S \,\} \;, \\ B &= \{\, (i,j) \in E \mid i \in S, j \in T \,\} \;, \\ C &= \{\, (i,j) \in E \mid i \in T, j \in T \,\} \;, \\ D &= \{\, (i,j) \in E \mid i,j \in T \,\} \;. \end{aligned}$$

(See the right-hand size of Fig. 1). If we ignore the direction of the edges (or if the graph is initially undirected), then the edges fall into the classes A, $B \cup C$ and D, as shown on the left-hand size of Fig. 1.

We let $w_{ij} \geq 0$ denote the weight of the edge (i,j). (If $(i,j) \notin E$, then $w_{ij} = 0$.) For a subset of edges $F \subseteq E$, we let $w(F) = \sum_{(i,j)\in F} w_{ij}$. The MAX (a,b,c,d)-BISECTION problem calls for maximizing $a\,w(A) + b\,w(B) + c\,w(C) + d\,w(D)$. When a,b,c,d range over 0 and 1 we obtain all the natural bisection problems. When $b = c$, the direction of the edges makes no difference and the resulting problem is a bisection problem of undirected graphs. If, for example, $a = d = 0$ and $b = c = 1$, then the resulting problem is the well studied MAX $\frac{n}{2}$-BISECTION. (It is possible to consider more general bisection problems in which a, b, c and d assume arbitrary nonnegative values, but the problems obtained are less natural. Our techniques could be used to devise approximation algorithms also for such problems.)

All the interesting bisection problems that can be obtained in this way are listed in Table 1. MAX (a,b,c,d)-BISECTION problems, with $a,b,c,d \in \{0,1\}$ that are not listed in Table 1 are either equivalent to one of the problems listed there, or can be trivially solved in polynomial time. (For example, the optimal solution of the MAX $\frac{n}{2}$-DIRECTED-VERTEX-COVER, corresponding to $a = b = 1$ and $c = d = 0$, is obtained by letting S be the set the $\frac{n}{2}$ vertices with the largest outdegree.) For each of the bisection problems listed in Table 1 we obtain an improved performance ratio. For some problems the improvement is quite substantial. For others, such as MAX $\frac{n}{2}$-BISECTION, the improvement is quite small. Even this improvement is important, however, as it is obtained by adding the 'triangle inequalities' to the semidefinite programming relaxation of the problem, and using a global argument to show their contribution to the relaxation. The question of whether these inequalities can be used to obtain an improved performance ratio for the related MAX CUT problem is still open (see [GW95],[FG95],[Zwi99],[FKL00a],[FS01]).

3 Techniques Used by Our Approximation Algorithms

Our algorithms are based first and foremost on the ideas of Frieze and Jerrum [FJ97]. To these we add some ideas used by Feige and Langberg [FL99] and by Ye [Ye99], and some ideas that were not used before. Each one of these ingredients is briefly discussed below.

ABCD	problem	our result	previous result
0 1 1 0	MAX $\frac{n}{2}$-BISECTION	0.7016	0.6993 [Ye99]
1 0 0 0	MAX $\frac{n}{2}$-DENSE-SUBGRAPH	0.6221	0.5866 [YZ99]
1 0 0 1	MAX $\frac{n}{2}$-UNCUT	0.6436	0.6024 [YZ99]
1 1 1 0	MAX $\frac{n}{2}$-VERTEX-COVER	0.8452	0.8113 [HYZZ00]

Undirected bisection problems

ABCD	problem	our result	previous result
0 1 0 0	MAX $\frac{n}{2}$-DIRECTED-BISECTION	0.6440	0.5000 [AHS00]
1 0 1 1	MAX $\frac{n}{2}$-DIRECTED-UNCUT	0.8118	0.7500

Directed bisection problems

Table 1. maximum bisection problems of undirected and directed graphs and the performance ratios obtained for them here and in previous works.

3.1 Using The 'Triangle Inequalities'

All our algorithms start by solving a semidefinite programming relaxation of the problem at hand. To obtain our improved results, we strengthen the semidefinite programming relaxations by adding the so called 'triangle inequalities', i.e., the inequalities $v_i \cdot v_j + v_i \cdot v_k + v_j \cdot v_k \geq -1$, for $0 \leq i, j, k \leq n$. These inequalities were used already in some algorithms for maximum bisection problems (e.g., [FL99]), but were previously analyzed using local arguments. Here we show for the first time that a global argument that considers all triples (i, j, k) together can be used to improve the analysis of the algorithm.

3.2 Using outward rotations

By solving the semidefinite programming relaxation of the problem considered, we obtain an (almost) optimal solution $v_0, v_1, \ldots, v_n$. We then have to round these vectors to obtain a partition of the graph. The rounding procedure picks a random hyperplane, which partitions the vectors into two parts (as in [GW95]). Before rounding the vectors $v_0, v_1, \ldots, v_n$, we subject them to a rotation process. Rotations were used before by Feige and Langberg [FL99] and by Ye [Ye99]. Feige and Langberg [FL99] use Feige-Goemans [FG95] rotations. Ye [Ye99] uses a different type of rotations that is also used by Nesterov [Nes98] and by Zwick [Zwi99]. Following [Zwi99], we call these *outward rotations*. Outward rotations are also used by us here as they seem to be better suited for maximum bisection problems than Feige-Goemans rotations. Although such rotations were used before, they were not used yet in conjunction with the other ingredients.

3.3 Adding linear randomized rounding

For asymmetric problems we obtain better performance ratios by combining the random hyperplane rounding method with a rounding procedure similar to the one used when rounding solutions of linear programming relaxations. The relaxations of asymmetric problems contain a vector v_0 whose role is to break the symmetry. In the linear rounding technique, vertex i is placed in S with probability $(1 - v_0 \cdot v_i)/2$, and in T with probability $(1 + v_0 \cdot v_i)/2$. The choices made for the different vertices are *independent*.

3.4 'Flipping' the bisection

The rounding procedure returns a cut (S, T). Usually, this cut is not a bisection, i.e., $|S| \neq \frac{n}{2}$. A bisection (S', T') is then obtained by 'mending' the cut (S, T). Some of the maximum bisection problems are symmetric, i.e., the value of a cut (S', T') is equal to the value of a cut (T', S'). Some, like MAX $\frac{n}{2}$-VERTEX-COVER and $\frac{n}{2}$-DENSE-SUBGRAPH are not. For such asymmetric problems, our algorithms compare the values of (S', T') and of (T', S') and return the bisection with the larger value. Interestingly, this improves the performance ratio achieved. This is somewhat counterintuitive, at least at first, as all previous steps of the algorithm attempt to make (S', T'), and not (T', S'), a 'heavy' bisection.

4 A Generic Approximation Algorithm

We now describe a generic approximation algorithm that can be turned into an approximation algorithm for any problem from the family MAX (a, b, c, d)-BISECTION. Our algorithms for the different bisection problems are obtained by setting various parameters used by these generic algorithm.

The generic algorithm is given in Fig. 3. It starts by setting some parameters. (The meaning of these parameters is explained later.) It then solves the semidefinite programming relaxation of the problem described in Fig. 2. The relaxation used is the standard relaxation of these problems, augmented by the 'triangle inequalities'. Recall that $w_{ij} \geq 0$ is the weight of the edge (i, j). Each vertex $i \in V$ has a unit vector associated with it. There is also a unit vector v_0 that plays an important role for asymmetric problems. To see that this is indeed a relaxation, note that if (S, T) is a bisection of the graph then by letting $v_i = -v_0$, if $i \in S$, and $v_i = v_0$, if $i \in T$, where v_0 is an arbitrary unit vector, we obtain a feasible solution of the relaxation whose value is $value_{abcd}(S, T) = a\,w(A) + b\,w(B) + c\,w(C) + d\,w(D)$. This follows from the fact that each term of the objective function can be written in the following way:

$$\begin{aligned} &a\,\frac{1 - v_0 \cdot v_i - v_0 \cdot v_j + v_i \cdot v_j}{4} + b\,\frac{1 - v_0 \cdot v_i + v_0 \cdot v_j - v_i \cdot v_j}{4} \\ +\,&c\,\frac{1 + v_0 \cdot v_i - v_0 \cdot v_j - v_i \cdot v_j}{4} + d\,\frac{1 + v_0 \cdot v_i + v_0 \cdot v_j + v_i \cdot v_j}{4}\,, \end{aligned}$$

and the term multiplied by a, for example, assumes the value 1 only if $v_i = v_j = -v_0$, i.e., if $(i, j) \in A$. Note that if we set the vectors in this way then $\sum_{i=1}^{n} v_i = 0$, as $|S| = |T|$, and thus $\sum_{i,j \geq 1} v_i \cdot v_j = (\sum_{i=1}^{n} v_i) \cdot (\sum_{j=1}^{n} v_j) = 0$. Also note that the constraint $\sum_{i,j \geq 1} v_i \cdot v_j = 0$ implies that $\sum_{i=1}^{n} v_i = 0$.

$$\text{Max} \sum_{i,j\geq 1} w_{ij} \frac{\left\{ \begin{array}{c} (a+b+c+d)-(a+b-c-d)(v_0\cdot v_i) \\ -(a-b+c-d)(v_0\cdot v_j)+(a-b-c+d)(v_i\cdot v_j) \end{array} \right\}}{4}$$

$$\text{s.t.} \quad \sum_{i,j\geq 1} v_i\cdot v_j \;=\; 0$$

$$\begin{array}{rcll} v_i\cdot v_j + v_i\cdot v_k + v_j\cdot v_k & \geq & -1 \quad , & 0\leq i,j,k\leq n \\ v_i\cdot v_j - v_i\cdot v_k - v_j\cdot v_k & \geq & -1 \quad , & 0\leq i,j,k\leq n \\ -v_i\cdot v_j - v_i\cdot v_k + v_j\cdot v_k & \geq & -1 \quad , & 0\leq i,j,k\leq n \\ -v_i\cdot v_j + v_i\cdot v_k - v_j\cdot v_k & \geq & -1 \quad , & 0\leq i,j,k\leq n \end{array}$$

$$v_i\cdot v_i = 1 \quad , \quad 0\leq i\leq n$$

Fig. 2. A generic semidefinite programming relaxation of a weighted instance of the MAX (a,b,c,d)-BISECTION problem.

ALGORITHM LARGE-(a,b,c,d)-BISECTION:

Input: A weighted (directed) graph $G=(V,E)$.
Output: A large (a,b,c,d)-bisection.

1. **(Initialization)** Choose parameters $0\leq\tau_1,\tau_2$, a rotation coefficient $0\leq\rho\leq 1$, a probability $0\leq\nu\leq 1$, and a tolerance $\epsilon>0$. Let $\mu=\alpha_{abcd}(\rho,\nu)+\tau_1\beta(\rho,\nu)+\tau_2\gamma(\rho,\nu)$.
2. **(Solving the relaxation)** Solve the semidefinite programming relaxation of the instance given in Fig. 2 and obtain an (almost) optimal solution $v_0,v_1,\ldots,v_n$. Let z^* be the value of this solution.
3. **(Rotating the vectors)** Construct unit vectors $v'_0,v'_1,\ldots,v'_n$ such that $v'_i\cdot v'_j=\rho\,(v_i\cdot v_j)$, for every $0\leq i<j\leq n$.
4. **(Find initial partition)** Repeat
   (a) Choose a random vector r and let $S=\{\,i\mid sgn(v'_i\cdot r)\neq sgn(v'_0\cdot r)\,\}$ and $T=V-S$.
   (b) With probability ν override the choice of S made above, and for each $i\in V$, put i in S, independently, with probability $(1-v_0\cdot v_i)/2$, and in T, otherwise.
   (c) Let $X=value_{abcd}(S,T)$, $Y_1=|T|$, $Y_2=|S||T|$ and $Z=\frac{1}{z^*}X+\frac{2\tau_1}{n}Y_1+\frac{4\tau_2}{n^2}Y_2$.

   Until $Z\geq(1-\epsilon)\mu$.
5. **(Constructing a bisection)** If $|S|\geq\frac{n}{2}$, let S' be a random subset of S of size $\frac{n}{2}$ and let $T'=V-S'$. If $|S|\leq\frac{n}{2}$, let T' be a random subset of T of size $\frac{n}{2}$, and let $S'=V-T'$.
6. **(Flipping the bisection)** If $value_{abcd}(S',T')\geq value_{abcd}(T',S')$ then output (S',T'). Otherwise, output (T',S').

Fig. 3. A generic approximation algorithm for MAX (a,b,c)-BISECTION.

The relaxation given in Fig. 2 is equivalent to a semidefinite program. An (almost) optimal solution $v_0, v_1, v_2, \ldots, v_n$ of it can therefore be found in polynomial time using either the Ellipsoid algorithm [GLS81], or using a more efficient interior-point method algorithm (see [WSV00]). We let z^* denote the value of this (almost) optimal solution.

The algorithm then subjects the vectors $v_0, v_1, v_2, \ldots, v_n$ to outward-rotation, and obtains new vectors $v'_0, v'_1, v'_2, \ldots, v'_n$ such that $v'_i \cdot v'_j = \rho\,(v_i \cdot v_j)$, where $0 \le \rho \le 1$ is a rotation parameter. (See [Zwi99] for how this can be done.) A different value of ρ is used for each bisection problem, but this value is used for all the instances of that problem. We may assume, without loss of generality, that $v'_0, v'_1, v'_2, \ldots, v'_n \in \mathbb{R}^{n+1}$.

Following Frieze and Jerrum [FJ97], who follow Goemans and Williamson [GW95], the algorithm then chooses a random vector r according to the $(n+1)$-dimensional normal distribution. This vector r defines a hyperplane that partitions the vertices of the graph into the sets $S = \{\, i \in V \mid sgn(v'_i \cdot r) \neq sgn(v'_0 \cdot r) \,\}$ and $T = V - S$. Then, with probability ν, this choice of S is *ignored* and a new set S is constructed as follows: each vertex $i \in V$ is placed in S, independently, with probability $(1 - v_0 \cdot v_i)/2$, and in T, otherwise. (This is what we referred to in Sect. 3.3 as *linear rounding*.)

The partition (S,T) determines the quantities $X = value_{abcd}(S,T)$, $Y_1 = |T|$, $Y_2 = |S||T|$ and $Z = \frac{1}{z^*}X + \frac{2\tau_1}{n}Y_1 + \frac{4\tau_2}{n^2}Y_2$. Here, X is the value of this partition (which is not necessarily a bisection yet), while Y_1 and Y_2 measure, in some way, how close this partition is to a bisection. Finally, Z is a weighted combination of X, Y_1 and Y_2, where τ_1 and τ_2 are problem specific parameters, and z^* is the value of the optimal solution of the semidefinite program.

The functions $\alpha_{abcd}(\rho,\nu)$, $\beta(\rho,\nu)$ and $\gamma(\rho,\nu)$, to be defined later, are such that $E[X] \ge \alpha_{abcd}(\rho,\nu)z^*$, $E[Y_1] \ge \beta(\rho,\nu)\frac{n}{2}$, and $E[Y_2] \ge \gamma(\rho,\nu)\frac{n^2}{4}$. (The quantities X, Y_1 and Y_2 are random variables as they are determined the partition (S,T), which is determined in turn by the random vector r.) Thus, $\mu = \alpha_{abcd}(\rho,\nu) + \tau_1\beta(\rho,\nu) + \tau_2\gamma(\rho,\nu)$, computed at the initialization phase of the algorithm, is a lower bound on $E[Z]$.

If $Z \ge (1-\epsilon)\mu$, then the partition (S,T) obtained using the random vector r, or using the linear rounding, is accepted. Otherwise, a new partition is generated. Here, $\epsilon > 0$ is a tolerance parameter set at the initialization phase of the algorithm. As $E[Z] \ge \mu$, $Z \le 1 + \tau_1 + \tau_2$, and $\epsilon > 0$, it is easy to see, using Markov's inequality, that the expected number of random vectors that have to be chosen until $Z \ge (1-\epsilon)\mu$ is only $O(\frac{1}{\epsilon})$.

The partition (S,T) obtained in this way is usually not a bisection. It is mended is the following way. If $|S| \ge \frac{n}{2}$, we let S' be a random subset of S of size $\frac{n}{2}$ and then let $T' = V - S'$. Otherwise, if $|S| \le \frac{n}{2}$, we let T' be a random subset of T of size $\frac{n}{2}$, and then let $S' = V - T'$. The partition (S',T') obtained in this way is now a bisection.

Finally, the algorithm compares $value_{abcd}(S',T')$ and $value_{abcd}(T',S')$, the values of the bisections (S',T') and (T',S') and outputs the better bisection. This step is helpful only for asymmetric problems.

5 The Functions $\alpha_{abcd}(\rho,\nu)$, $\beta(\rho,\nu)$ and $\gamma(\rho,\nu)$

The functions $\alpha_{abcd}(\rho,\nu)$, $\beta(\rho,\nu)$ and $\gamma(\rho,\nu)$ play a central role in our algorithms and in their analysis. The function $\alpha_{abcd}(\rho,\nu)$ is problem specific. The functions $\beta(\rho,\nu)$ and $\gamma(\rho,\nu)$ are the same for all the bisection problems. As mentioned in the previous section, these functions should satisfy $E[X] \geq \alpha_{abcd}(\rho,\nu)z^*$, $E[Y_1] \geq \beta(\rho,\nu)\frac{n}{2}$, and $E[Y_2] \geq \gamma(\rho,\nu)\frac{n^2}{4}$. The larger we could make these functions, while satisfying the required conditions, the larger the performance ratios obtained by our algorithms would be.

The bounds on $\alpha_{abcd}(\rho,\nu)$ and $\beta(\rho,\nu)$ are obtained using standard local ratio techniques (see [GW95], [FG95] and [Zwi99]). Thus, for example,

$$\alpha_{0110}(\rho,0) \;=\; \min_{-1\leq x<1} \frac{2}{\pi}\,\frac{\arccos(\rho x)}{1-x}\;.$$

We specify the function $\alpha_{0110}(\rho,\nu)$ only for $\nu=0$, as $\nu=0$ is the optimal choice for symmetric problems, i.e., problems with $abcd=dcba$. A full description of how the functions $\alpha_{abcd}(\rho,\nu)$ are computed is given in the full version.

Let β'' and γ'' be the expectations of $\frac{2Y_1}{n}$ and $\frac{4Y_2}{n^2}$, respectively, given that step 4(b) is executed, i.e., when vertex i is independently put in S, with probability $\frac{1-v_0\cdot v_i}{2}$. Note that $\beta''=\beta(\rho,1)$, $\gamma''=\gamma(\rho,1)$, for any $0\leq\rho\leq 1$. Also let $\beta'(\rho)=\beta(\rho,0)$ and $\gamma'(\rho)=\gamma(\rho,0)$. It is easy to see that we can let:

$$\begin{aligned}\beta(\rho,\nu) &= (1-\nu)\beta'(\rho)+\nu\beta''\;,\\ \gamma(\rho,\nu) &= (1-\nu)\gamma'(\rho)+\nu\gamma''\;.\end{aligned}$$

We next show that we may assume that $\beta''=\gamma''=1-O(1/n)$. This follows as

$$\beta'' \;=\; \tfrac{2}{n}\textstyle\sum_{i=1}^n \Pr[i\in T] = \tfrac{2}{n}\sum_{i=1}^n \frac{1+v_0\cdot v_i}{2} = 1\;.$$

(Recall that $\sum_{i,j} v_i\cdot v_j = 0$ implies $\sum_{i=1}^n v_i = 0$, and therefore $\sum_{i=1}^n v_0\cdot v_i = 0$.) Next, we have

$$\gamma'' \;=\; \tfrac{4}{n^2}\textstyle\sum_{i,j}\Pr[i\in S\;,\;j\in T] \;=\; \tfrac{4}{n^2}\sum_{i=1}^n \frac{1-v_0\cdot v_i}{2}\sum_{j\neq i}\frac{1+v_0\cdot v_j}{2} \;\geq\; 1-\frac{2}{n}\;,$$

as for any i, we have $\sum_{j\neq i}\frac{1+v_0\cdot v_j}{2} = \sum_{j=1}^n \frac{1+v_0\cdot v_j}{2} - \frac{1+v_0\cdot v_i}{2} \geq \frac{n}{2}-1$. In the full version of the paper we show that

$$\beta'(\rho) \;=\; \min_{-1\leq x\leq 0}\frac{2}{\pi}\,\frac{\arccos(\rho x)-x\arccos(\rho)}{1-x}\;,$$

is a valid choice for $\beta'(\rho)$.

It is not difficult to verify that $\gamma'(\rho)=\beta'(\rho)$ is a valid choice for $\gamma'(\rho)$. One of the contributions of this paper is a demonstration that

$$\gamma'(\rho) \;=\; \min_{-1\leq x\leq -\frac{1}{3}}\frac{1}{\pi}\left(\arccos(\rho x)+\frac{3(x+1)}{4}\arccos(-\frac{\rho}{3})+\frac{1-3x}{4}\arccos(\rho)\right)$$

is also a valid choice. This is a better choice as it is not difficult to check that $\beta'(\rho) < \gamma'(\rho)$, for every $0 < \rho \leq 1$. The proof that this choice is valid in given in Appendix A. We show there that obtaining good choices for $\gamma'(\rho)$ is closely related to the behavior of the MAX CUT algorithms of Goemans and Williamson [GW95] and of Zwick [Zwi99], strengthened by the 'triangle inequalities', on *complete* graphs. To show that the above choice of $\gamma'(\rho)$ is valid, we rely on Turán's theorem that states that any undirected graph on n vertices with more than $\frac{n^2}{4}$ edges contains a triangle (Turán [Tur41]). Thus, if $v_0, v_1, \ldots, v_n$ is a feasible solution of the relaxation given in Fig. 2, then for at most $\frac{n^2}{4}$ pairs (i,j), where $1 \leq i < j \leq n$, we can have $v_i \cdot v_j < -\frac{1}{3}$. We *conjecture* that the function

$$\bar{\gamma}'(\rho) \;=\; \frac{1}{2\pi}\left(3\arccos(-\frac{\rho}{3}) + \arccos(\rho)\right) \;,$$

which satisfies $\gamma'(\rho) < \bar{\gamma}'(\rho)$, for any $0 < \rho \leq 1$, is also a valid choice for $\gamma'(\rho)$. If true, then further improvements to all the bounds given in Table 1 would be obtained. Furthermore, $\bar{\gamma}'(\rho)$ would then be the optimal choice for $\gamma'(\rho)$.

6 The Performance Ratio of the Generic Algorithm

We continue with the analysis of the generic approximation algorithm for bisection problems given in Fig. 3. We have already reached a stage in which the algorithm finds a partition (S,T) of the graph for which $Z \geq (1-\epsilon)\mu$. Recall that $X = value_{abcd}(S,T)$, $Y_1 = |T|$, $Y_2 = |S||T|$, that $Z = \frac{1}{z^*}X + \frac{2\tau_1}{n}Y_1 + \frac{4\tau_2}{n^2}Y_2$, and that $\mu = \alpha_{abcd}(\rho,\nu) + \tau_1\beta(\rho,\nu) + \tau_2\gamma(\rho,\nu)$. Let $\lambda = \frac{X}{z^*}$ and $\delta = \frac{|S|}{n}$, so that $X = \lambda z^*$, $|S| = \delta n$, $|T| = (1-\delta)n$, and thus $Y_1 = (1-\delta)n$ and $Y_2 = \delta(1-\delta)n^2$. To simplify the analysis, we assume that $\epsilon = 0$. As ϵ may be arbitrarily small, we may get arbitrarily close to the ratio obtained by this analysis. Thus,

$$\lambda + 2(1-\delta)\tau_1 + 4\delta(1-\delta)\tau_2 \;\geq\; \alpha_{abcd}(\rho,\nu) + \tau_1\beta(\rho,\nu) + \tau_2\gamma(\rho,\nu) \;.$$

The partition (S,T) is usually not a bisection. To covert it into a bisection, we choose either a random subset of T of size $\frac{n}{2}$, if $\delta \leq \frac{1}{2}$, and call it T', or a random subset of S of size $\frac{n}{2}$, and call it S', if $\delta \geq \frac{1}{2}$. We let (S',T') be the bisection obtained in this way.

Suppose, at first, that $\delta \geq \frac{1}{2}$. Each element of S has a probability of $p = \frac{1}{2\delta}$ of being chosen to S'. However, as we are choosing a subset of size exactly $\frac{n}{2}$, these events are *not* independent. But, as n may be assumed to be large, these events are almost independent. To simplify the analysis, we assume that these events are independent. This can be justified as follows. We first construct an almost-bisection (S',T'), where S' is obtained by choosing each element of S independently with probability p. (We still assume that $\delta \leq \frac{1}{2}$.) With overwhelming probability $|S'| = \frac{n}{2} + O(\sqrt{n})$. We then convert (S',T') into an exact bisection (S'',T'') by choosing subsets of size exactly $\frac{n}{2}$. It is easy to see that the loss incurred by moving from (S',T') to (S'',T'') is negligible. Similarly, if $\delta \leq \frac{1}{2}$, we assume that each element of T is chosen to be in T' independently with probability $q = \frac{1}{2(1-\delta)}$.

We find it more convenient now to replace the names A, B, C, D and a, b, c, d by indexable names. We let $(F_1, F_2, F_3, F_4) = (A, B, C, D)$ and $(a_1, a_2, a_3, a_4) = (a, b, c, d)$. With our new notation, we have $value_{abcd}(S, T) = \sum_{i=1}^{4} a_i w(F_i)$. It is easy to check that if F_1, F_2, F_3, F_4 is the partition of the edges induced by (S, T), and F_1', F_2', F_3', F_4' is the partition induced by (S', T'), then

$$\begin{pmatrix} E[w(F_1')] \\ E[w(F_2')] \\ E[w(F_3')] \\ E[w(F_4')] \end{pmatrix} = \begin{pmatrix} p^2 & 0 & 0 & 0 \\ p(1-p) & p & 0 & 0 \\ p(1-p) & 0 & p & 0 \\ (1-p)^2 & 1-p & 1-p & 1 \end{pmatrix} \begin{pmatrix} w(F_1) \\ w(F_2) \\ w(F_3) \\ w(F_4) \end{pmatrix} \quad , \quad \text{if } \delta \geq \tfrac{1}{2} \, ,$$

$$\begin{pmatrix} E[w(F_1')] \\ E[w(F_2')] \\ E[w(F_3')] \\ E[w(F_4')] \end{pmatrix} = \begin{pmatrix} 1 & 1-q & 1-q & (1-q)^2 \\ 0 & q & 0 & q(1-q) \\ 0 & 0 & q & q(1-q) \\ 0 & 0 & 0 & q^2 \end{pmatrix} \begin{pmatrix} w(F_1) \\ w(F_2) \\ w(F_3) \\ w(F_4) \end{pmatrix} \quad , \quad \text{if } \delta \leq \tfrac{1}{2} \, .$$

Denote the matrices appearing in these expressions by $M_1(p)$ and $M_2(q)$. Let

$$M(\delta) = \begin{cases} M_1(\frac{1}{2\delta}) & \text{if } \delta \geq \frac{1}{2} \, , \\ M_2(\frac{1}{2(1-\delta)}) & \text{if } \delta \leq \frac{1}{2} \, . \end{cases}$$

Recall, again, that $X = \sum_{i=1}^{4} a_i w(F_i) = \lambda z^*$. Let λ_i, for $1 \leq i \leq 4$, be such that $w(F_i) = \lambda_i z^*$. Then, we have $\sum_{i=1}^{4} a_i \lambda_i = \lambda$. Also, as $\sum_{i=1}^{4} w(F_i) = w(E) \geq z^*$, we also have $\sum_{i=1}^{4} \lambda_i \geq 1$. Define

$$f_{abcd}(\delta, \lambda_1, \lambda_2, \lambda_3, \lambda_4) = (a_1 \; a_2 \; a_3 \; a_4) \; M(\delta) \; (\lambda_1 \; \lambda_2 \; \lambda_3 \; \lambda_4)^T \, .$$

We are finally able to state the performance ratio of our algorithm:

Theorem 1. *The performance ratio of the approximation algorithm* LARGE-(a, b, c, d)-PARTITION, *given in Fig. 3, when run with the parameters* τ_1, τ_2 *and* ρ *and with the functions* $\alpha_{abcd}(\rho, \nu)$, $\beta(\rho, \nu)$ *and* $\gamma(\rho, \nu)$, *is at least*

$$g_{abcd}(\tau_1, \tau_2, \rho, \nu) =$$

$$\left[\begin{array}{ll} \min & \max\{ \, f_{abcd}(\delta, \lambda_1, \lambda_2, \lambda_3, \lambda_4) \, , \, f_{dcba}(\delta, \lambda_1, \lambda_2, \lambda_3, \lambda_4) \, \} \\ \text{s.t.} & \lambda + 2(1-\delta)\tau_1 + 4\delta(1-\delta)\tau_2 \; \geq \; \alpha_{abcd}(\rho, \nu) + \tau_1 \beta(\rho, \nu) + \tau_2 \gamma(\rho, \nu) \\ & a_1\lambda_1 + a_2\lambda_2 + a_3\lambda_3 + a_4\lambda_4 = \lambda \\ & \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 \geq 1 \\ & 0 \leq \delta \leq 1 \, , \; 0 \leq \lambda, \lambda_1, \lambda_2, \lambda_3, \lambda_4 \end{array} \right] .$$

Proof. It is not difficult to verify that the expected value of the bisection (S', T') constructed by the algorithm is $f_{abcd}(\delta, \lambda_1, \lambda_2, \lambda_3, \lambda_4) \, z^*$, and that the expected value of the flipped bisection (T', S') is $f_{dcba}(\delta, \lambda_1, \lambda_2, \lambda_3, \lambda_4) \, z^*$. (Note that flipping the bisection is equivalent to changing *abcd* to *dcba*.) As z^* is the optimal value of the relaxation, it is an upper bound on the value of the optimal bisection. Thus, $\max\{ \, f_{abcd}(\delta, \lambda_1, \lambda_2, \lambda_3, \lambda_4) \, , \, f_{dcba}(\delta, \lambda_1, \lambda_2, \lambda_3, \lambda_4) \, \}$ is the performance ratio achieved when $|S| = \delta n$ and $w(F_i) = \lambda_i z^*$, for $1 \leq i \leq 4$. Finally, it was argued above that $\delta, \lambda, \lambda_1, \lambda_2, \lambda_3, \lambda_4$ satisfy all the stated inequalities. The claim of the theorem follows. $\square$

Although the functions $f_{abcd}(\delta, \lambda_1, \lambda_2, \lambda_3, \lambda_4)$ and $g_{abcd}(\tau_1, \tau_2, \rho, \nu)$ are fairly complicated functions, it is possible to evaluate them numerically for any value of their arguments. To see this, let us write the function $g_{abcd}(\tau_1, \tau_2, \rho, \nu)$ in the following form:

$$g_{abcd}(\tau_1 . \tau_2, \rho, \nu) = \min_{0 \le \delta \le 1} \begin{bmatrix} \min & z \\ \text{s.t.} & \lambda \ge h(\tau_1, \tau_2, \rho, \nu, \delta) \\ & a_1\lambda_1 + a_2\lambda_2 + a_3\lambda_3 + a_4\lambda_4 = \lambda \\ & \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 \ge 1 \\ & z \ge f_{abcd}(\delta, \lambda_1, \lambda_2, \lambda_3, \lambda_4) \\ & z \ge f_{dcba}(\delta, \lambda_1, \lambda_2, \lambda_3, \lambda_4) \\ & 0 \le \lambda, \lambda_1, \lambda_2, \lambda_3, \lambda_4 \end{bmatrix} ,$$

where

$$h(\tau_1, \tau_2, \rho, \nu, \delta) = \alpha_{abcd}(\rho, \nu) + \tau_1\beta(\rho, \nu) + \tau_2\gamma(\rho, \nu) - (2(1-\delta)\tau_1 + 4\delta(1-\delta)\tau_2).$$

Thus, for every fixed $\tau_1, \tau_2, \rho, \delta$, we get a linear program whose optimal value is attained at one of its vertices. Thus, for any τ_1, τ_2, and ρ, the performance of the algorithm is easy to compute. 'Flipping' the bisection will improve the performance of the algorithm if two of the inequalities defining the optimal value of the linear program are $z = f_{1110}(\delta, \lambda_1, \lambda_2, \lambda_3)$, and $z = f_{0111}(\delta, \lambda_1, \lambda_2, \lambda_3)$.

To obtain our best approximation algorithms for the various problems we choose the parameters τ_1, τ_2, ρ and ν that maximize the function $g_{abcd}(\tau_1, \tau_2, \rho, \nu)$. The computations here were done numerically. The performance guarantees obtained in this way may be made rigorous with the help of a rigorous global optimization system. See Zwick [Zwi00] for further details.

7 Parameter Setting

The setting of the parameters τ_1, τ_2, ρ and ν using which we obtain the results stated in Table 1 are given in Table 2. It is interesting to note that $\rho < 1$, i.e., outward rotations are used, for all the bisection problems. Also linear rounding is used, i.e., $\nu > 0$, by the algorithms for all the *asymmetric* problems. Finally, $\tau_1 > 0$ only for the asymmetric undirected problems.

8 Concluding Remarks

We presented a unified approach for developing approximation algorithms for all maximum bisection problems of directed and undirected graphs. Our approach uses almost all previously used techniques together with some techniques that are used here for the first time. Using this unified approach we obtained improved approximation algorithms for *all* the maximum bisection problems.

It is possible to obtain some small improvements to the performance ratios reported here. These improvements are obtained by adding some additional features to the generic approximation algorithm presented in Fig. 3. Some improvements may be obtained, for some of the problems, by combining outward

ABCD	problem	ratio	τ_1	τ_2	ρ	ν
0 1 1 0	MAX $\frac{n}{2}$-BISECTION	0.7016	0	1.83	0.89	0
1 0 0 0	MAX $\frac{n}{2}$-DENSE-SUBGRAPH	0.6221	0.69	2.15	0.87	0.23
1 0 0 1	MAX $\frac{n}{2}$-UnCUT	0.6436	0	3.86	0.81	0
1 1 1 0	MAX $\frac{n}{2}$-VERTEX-COVER	0.8452	0.5	0.37	0.88	0.06
0 1 0 0	MAX $\frac{n}{2}$-DIRECTED-BISECTION	0.6440	0	1.61	0.93	0.16
1 0 1 1	MAX $\frac{n}{2}$-DIRECTED-UnCUT	0.8118	0	2.10	0.74	0.04

Table 2. The settings of the parameters τ_1, τ_2, ρ and ν using which the results stated in Table 1 are obtained.

rotations with Feige-Goemans rotations [FG95]. The improvements, however, are quite small while the analysis becomes substantially more complicated.

Obtaining substantially improved ratios, especially for MAX $\frac{n}{2}$-BISECTION, which is perhaps the most natural bisection problem, remains a challenging open problem. It would also be interesting to prove, or disprove, our conjecture, made in Sect. 5, regarding the function $\bar{\gamma}(\rho)$.

References

[AHS00] A. Ageev, R. Hassin, and M. Sviridenko. An 0.5-approximation algorithm for MAX DICUT with given sizes of parts. Manuscript, 2000.

[FG95] U. Feige and M.X. Goemans. Approximating the value of two prover proof systems, with applications to MAX-2SAT and MAX-DICUT. In *Proceedings of the 3nd Israel Symposium on Theory and Computing Systems, Tel Aviv, Israel*, pages 182–189, 1995.

[FJ97] A.M. Frieze and M. Jerrum. Improved approximation algorithms for MAX k-CUT and MAX BISECTION. *Algorithmica*, 18:67–81, 1997.

[FKL00a] U. Feige, M. Karpinski, and M. Langberg. Improved approximation of Max-Cut on graphs of bounded degree. Technical report, E-CCC Report number TR00-043, 2000.

[FKL00b] U. Feige, M. Karpinski, and M. Langberg. A note on approximating MAX-BISECTION on regular graphs. Technical report, E-CCC Report number TR00-043, 2000.

[FL99] U. Feige and M. Langberg. Approximation algorithms for maximization problems arising in graph partitioning. Manuscript, 1999.

[FS01] U. Feige and G. Schechtman. On the integrality ratio of semidefinite relaxations of MAX CUT. In *Proceedings of the 33th Annual ACM Symposium on Theory of Computing, Crete, Greece*, 2001. To appear.

[GLS81] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.

[GW95] M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

[HYZ00] Q. Han, Y. Ye, and J. Zhang. An improved rounding method and semidefinite programming relaxation for graph partition. Manuscript, 2000.

[HYZZ00] Q. Han, Y. Ye, H. Zhang, and J. Zhang. On approximation of Max-Vertex-Cover. Manuscript, 2000.

[Nes98] Y. E. Nesterov. Semidefinite relaxation and nonconvex quadratic optimization. *Optimization Methods and Software*, 9:141–160, 1998.

[Tur41] P. Turán. On an extremal problem in graph theory. *Mat. Fiz. Lapok*, 48:436–452, 1941.

[WSV00] H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors. *Handbook of semidefinite programming*. Kluwer, 2000.

[Ye99] Y. Ye. A 0.699-approximation algorithm for Max-Bisection. Manuscript, 1999.

[YZ99] Y. Ye and J. Zhang. Approximation of dense-$\frac{n}{2}$-subgraph and the complement of min-bisection. Manuscript, 1999.

[Zwi99] U. Zwick. Outward rotations: a tool for rounding solutions of semidefinite programming relaxations, with applications to max cut and other problems. In *Proceedings of the 31th Annual ACM Symposium on Theory of Computing, Atlanta, Georgia*, pages 679–687, 1999.

[Zwi00] U. Zwick. Analyzing the MAX 2-SAT and MAX DI-CUT approximation algorithms of Feige and Goemans. Manuscript, 2000.

A The Function $\gamma'(\rho)$

We show that the choice of $\gamma'(\rho)$ given in Sect. 5 is valid, i.e., that $E[|S||T|] \geq \gamma'(\rho)\frac{n^2}{4}$, for any sequence of unit vectors $v_0, v_1, \ldots, v_n$ that satisfies the constraints of the generic semidefinite programming relaxation given in Fig. 2. Note that these constraints do not depend on the specific problem considered. By the linearity of the expectation, we have $E[|S||T|] = \frac{1}{\pi}\sum_{i<j} \arccos(\rho(v_i \cdot v_j))$. Let

$$\gamma^*(\rho) \;=\; \left[\begin{array}{lcl} \min & \dfrac{4}{\pi n^2}\displaystyle\sum_{i<j} \arccos(\rho x_{ij}) & \\ \text{s.t.} & X \succeq 0 & \\ & x_{ii} = 1 & ,\ 1 \leq i \leq n \\ & \sum_{i<j} x_{ij} = -\frac{n}{2} & \\ & x_{ij} + x_{ik} + x_{jk} \geq -1 & ,\ 1 \leq i,j,k \leq n \\ & -1 \leq x_{ij} \leq 1 & ,\ 1 \leq i,j \leq n \end{array}\right] ,$$

where $X \succeq 0$ indicates that the matrix $X = (x_{ij})$ is a positive semidefinite matrix. It is clear from the definition of $\gamma^*(\rho)$ that $E[|S||T|] \geq \gamma^*(\rho)\frac{n^2}{4}$, as for any feasible solution $v_0, v_1, \ldots, v_n$ of the semidefinite program, the matrix $X = (x_{ij})$, where $x_{ij} = v_i \cdot v_j$, is a feasible solution of the optimization problem appearing in the definition of $\gamma^*(\rho)$. It is enough to show, therefore, that $\gamma^*(\rho) \geq \gamma'(\rho)$. If $X = (x_{ij})$ is a feasible point of the problem defining $\gamma^*(\rho)$, we let

$$\begin{aligned} \mathcal{A} &= \{\, (i,j) \mid x_{ij} < -\frac{1}{3} \;,\; 1 \leq i < j \leq n \,\} \;, \\ \mathcal{B} &= \{\, (i,j) \mid -\frac{1}{3} \leq x_{ij} < 0 \;,\; 1 \leq i < j \leq n \,\} \;, \\ \mathcal{C} &= \{\, (i,j) \mid 0 \leq x_{ij} \;,\; 1 \leq i < j \leq n \,\} \;. \end{aligned}$$

Consider the subgraph $G' = (V, \mathcal{A})$. As for every $(i,j) \in \mathcal{A}$ we have $x_{ij} < -\frac{1}{3}$, and as for every $1 \le i,j,k \le n$ we have $x_{ij} + x_{ik} + x_{jk} \ge -1$, we get that the graph $G' = (V, \mathcal{A})$ is *triangle-free*. A theorem of Turán [Tur41] states that a triangle-free n-vertex graph can have at most $\frac{n^2}{4}$ edges. It follows, therefore, that $|\mathcal{A}| \le \frac{n^2}{4}$. We now define the following function:

$$\gamma''(\rho) \;=\; \left[\begin{array}{ll} \min & \displaystyle\frac{4}{\pi n^2}\sum_{i<j} \arccos(\rho x_{ij}) \\ & \\ \text{s.t.} & \sum_{i<j} x_{ij} = -\frac{n}{2} \\ & -1 \le x_{ij} \le 1 \;,\; 1 \le i,j \le n \\ & |\{(i,j) \mid x_{ij} \le -\frac{1}{3}\}| \le \frac{n^2}{4} \end{array}\right] .$$

As all the feasible points of the minimization problem defining $\gamma^*(\rho)$ are also feasible points of the minimization problem defining $\gamma''(\rho)$, we get that $\gamma''(\rho) \le \gamma^*(\rho)$. We show next that $\gamma''(\rho) = \gamma'(\rho) + O(\frac{1}{n})$, for every $0 \le \rho \le 1$.

The proof relies on the convexity/concavity properties of the function $f(x) = \arccos(\rho x)$ (see Fig. 4). It is easy to see that $f(x)$ is concave in $[0,1]$ and convex in $[-1,0]$. As an immediate corollary, we get:

(i) If $0 < x-\epsilon \le x \le y \le y+\epsilon \le 1$, then $f(x-\epsilon)+f(y+\epsilon) \le f(x)+f(y)$.
(ii) If $-1 \le x \le x+\epsilon \le y-\epsilon \le y \le 0$, then $f(x+\epsilon)+f(y-\epsilon) \le f(x)+f(y)$.

Thus, if $X = (x_{ij})$ is an optimal solution of the minimization problem that appears in the definition of $\gamma''(\rho)$, and if $(i,j),(k,l) \in \mathcal{A}$, or $(i,j),(k,l) \in \mathcal{B}$, then $x_{ij} = x_{kl}$, otherwise, we can decrease the objective function by moving x_{ij} and x_{kl} closer together. Similarly, if $(i,j),(k,l) \in \mathcal{C}$, and $x_{ij} \le x_{kl}$, then either $x_{ij} = 0$, or $x_{kl} = 1$, otherwise, we can decrease the objective function by moving x_{ij} and x_{kl} further apart. There can therefore be only a single point $(i,j) \in \mathcal{C}$ for which $x_{ij} \ne 0$ and $x_{ij} \ne 1$. As each point makes only an $O(\frac{1}{n^2})$ difference, we assume, for simplicity, that there is no such point. Next note that if $(i,j) \in \mathcal{B}$, $(k,l) \in \mathcal{C}$, and $x_{kl} = 0$, then we can move x_{ij} and x_{kl} closer together. It follows, therefore, that there exist $x < -\frac{1}{3}$ and $-\frac{1}{3} \le y \le 0$ (see Fig. 4) such that for every $(i,j) \in \mathcal{A}$ we have $x_{ij} = x$, for every $(i,j) \in \mathcal{B}$ we have $x_{ij} = y$, and for every $(i,j) \in \mathcal{C}$ we have $x_{ij} = 1$.

We next show that $\mathcal{A}$ is not empty. If $\mathcal{A}$ is empty and n is large enough, then $\mathcal{B}$ must contain at least four points. It is easy to check that for every $-\frac{1}{3} \le y \le 0$ we have $4f(y) > 3f(\frac{4y-1}{3}) + f(1)$, thus we can decrease the objective function by replacing the four points y, y, y, y by the points $\frac{4y-1}{3}, \frac{4y-1}{3}, \frac{4y-1}{3}, 1$. Using a very similar argument we in fact get that $|\mathcal{A}| = \frac{n^2}{4}$.

As $\mathcal{A}$ is not empty, we must have $y = -\frac{1}{3}$, as otherwise we can decrease the objective function by moving a point from $\mathcal{B}$ and a point from $\mathcal{A}$ closer together.

Let $a = |\mathcal{A}| = \frac{n^2}{4}$, $b = |\mathcal{B}|$ and $c = |\mathcal{C}|$. Then, we must have

$$a \cdot x + b \cdot (-\tfrac{1}{3}) + c \;=\; -\tfrac{n}{2} \quad , \quad a + b + c \;=\; \tbinom{n}{2} .$$

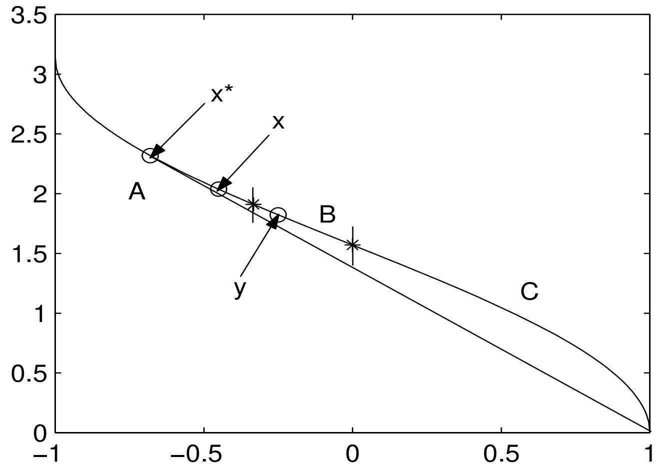

Fig. 4. The function $\arccos(x)$, the sets $\mathcal{A}, \mathcal{B}, \mathcal{C}$, and the points x^*, x and y.

Solving these equations, we get $a = \frac{n^2}{4}$, $b = \frac{3n^2(1+x)}{16}$, $c = \frac{n^2(1-3x)}{16} - \frac{n}{2}$. Thus,

$$\begin{aligned}\gamma''(\rho) &= \min_{-1 \le x \le -\frac{1}{3}} \frac{4}{\pi n^2} \left(\frac{n^2}{4} f(x) + \frac{3n^2}{16}(1+x) f(-\tfrac{1}{3}) + (\tfrac{n^2}{16}(1-3x) - \tfrac{n}{2}) f(1) \right) \\ &= \min_{-1 \le x \le -\frac{1}{3}} \frac{1}{\pi} \left(\arccos(\rho x) + \frac{3(x+1)}{4} \arccos(-\tfrac{\rho}{3}) + (\tfrac{1-3x}{4} - \tfrac{2}{n}) \arccos(\rho) \right) ,\end{aligned}$$

as required.

As mentioned in Sect. 5, we conjecture that $\gamma^*(\rho) = \bar{\gamma}'(\rho) + O(\frac{1}{n})$, where

$$\bar{\gamma}'(\rho) \;=\; \frac{1}{2\pi} \left(3 \arccos(-\tfrac{\rho}{3}) + \arccos(\rho) \right) ,$$

and that the optimal solution for the optimization problem defining $\gamma^*(\rho)$ is attained, when $4|n$, at

$$x_{ij} = \begin{cases} -\frac{1}{3} & \text{if } i \not\equiv j \pmod 4, \\ 1 & \text{otherwise.} \end{cases}$$

This correspond to the following spatial configuration: The unit vectors v_1, v_2, v_3 and v_4 point to the corners of regular tetrahedron, and thus $v_i \cdot v_j = -\frac{1}{3}$, for $i \neq j$, and each one of these vectors is duplicated $\frac{n}{4}$ times. We have numerical evidence that supports this conjecture, but not a proof.

Finally, we explain the comment made in Sect. 5, that obtaining a good choice for $\gamma'(\rho)$ is closely related to the study of the MAX CUT algorithms of Goemans and Williamson [GW95] and of Zwick [Zwi99], strengthened by the 'triangle inequalities', on *complete* graphs. In the semidefinite programming relaxation given in Fig. 2, we have the constraint $\sum_{i,j \ge 0} v_i \cdot v_j = 0$. As for every sequence of vectors $v_1, v_2, \ldots, v_n$ we have $\sum_{i,j \ge 0} v_i \cdot v_j = (\sum_{i=1}^n v_i) \cdot (\sum_{i=1}^n v_i) \ge 0$, we get that any solution of our semidefinite programming relaxation is an *optimal* solution of the semidefinite programming relaxation of the MAX CUT problem of the complete graph on n vertices, whose objective function in $\sum_{i<j} \frac{1 - v_i \cdot v_j}{2}$.