Combinatorial approximation algorithms for the maximum directed cut problem

Eran Halperin [*] *Uri Zwick* [*]

Abstract

We describe several combinatorial algorithms for the maximum directed cut problem. Among our results is a simple linear time $\frac{9}{20}$-approximation algorithm for the problem, and a somewhat slower $\frac{1}{2}$-approximation algorithm that uses a bipartite matching routine. No better combinatorial approximation algorithms are known even for the easier maximum cut problem for undirected graphs. Our algorithms do *not* use linear programming, nor semidefinite programming. They are based on the observation that the maximum directed cut problem is equivalent to the problem of finding a maximum independent set in the line graph of the input graph, and that the linear programming relaxation of the problem is equivalent to the problem of finding a maximum fractional independent set of that line graph. The maximum fractional independent set problem can be easily reduced to a bipartite matching problem. As a consequence of this relation, we also get that the maximum directed cut problem for bipartite digraphs can be solved in polynomial time.

1 Introduction

The MAX CUT and MAX DI-CUT problems are well known combinatorial optimization problems. An input to the MAX CUT problem is an undirected graph $G = (V, E)$. The goal is to find a cut of G of maximum size. A cut is a partition (S, T) of the vertices, where $S \cup T = V$ and $S \cap T = \phi$. An edge $e \in E$ is cut by this cut if and only if $|e \cap S| = 1$. The MAX DI-CUT problem is the analog problem for directed graphs. Here a cut (S, T) cuts a directed edge (u, v) if and only if $u \in S$ and $v \in T$.

MAX CUT and MAX DI-CUT are hard optimization problems. It is known that no approximation ratio better than $\frac{16}{17}$ can be obtained for the MAX CUT problem, and that no approximation ratio better than $\frac{12}{13}$ can be obtained for the MAX DI-CUT problem, unless $P = NP$ (Trevisan *et al.* [TSSW96], Håstad [Hås97]).

Obtaining a $\frac{1}{2}$-approximation algorithm for the MAX CUT problem is easy. A random cut cuts, on average, half of the edges. This immediately gives a randomized $\frac{1}{2}$-approximation algorithm. A deterministic algorithm can then be obtained using the method of conditional probabilities (see, e.g., Alon and Spencer [AS92]). A similar approach gives, however, only a $\frac{1}{4}$-approximation algorithm for the MAX DI-CUT problem.

Improved approximation algorithms for the MAX CUT and MAX DI-CUT problems can be obtained using semidefinite programming. Goemans and Williamson [GW95] obtain a 0.878-approximation algorithm for the MAX CUT problem. Feige and Goemans [FG95] (see also Zwick [Zwi00]), improving a result of [GW95], obtain a 0.859-approximation algorithm for the MAX DI-CUT problem. (See Klein and Lu [KL96] for fast implementations of these algorithms.) Further narrowing the gaps between these bounds and the hardness results stated above is a challenging open problem.

Here, however, we are more interested in two other problems. The first is, what performance ratio can be obtained for MAX CUT and MAX DI-CUT *without* using semidefinite programming? The second is, is the MAX DI-CUT problem really harder than the MAX CUT problem? We try to shed some light on these problems by considering combinatorial approximation algorithms for the MAX DI-CUT problem.

Trevisan [Tre98a], building on ideas of Goemans and Williamson [GW94], obtains a $\frac{1}{2}$-approximation algorithm for MAX DI-CUT using linear programming. Alimonti [Ali97] obtains a $\frac{1}{3}$ approximation algorithm for MAX DI-CUT using local search. (See [KMSV94] for further discussion of local search approximation algorithms.) Here we improve these results and obtain a linear time $\frac{9}{20}$-approximation algorithm, and a purely combinatorial $\frac{1}{2}$-approximation algorithm for the problem.

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

ALGORITHM DC1:

Input: A directed graph $G = (V, A)$.
Output: A directed cut whose size, on average, is at least $\frac{2}{5}$ of the size of an optimal directed cut.

1. Let $G' \leftarrow G$.
2. While there is a directed path of length two in G', find such a path and remove its edges from G'.
3. Let $S_1 \leftarrow \{v \in V \mid d_{out}(G', v) > 0\}$,
   $S_2 \leftarrow \mathrm{RAND}(V, \frac{1}{2})$.
4. Output the larger of the cuts $(S_1, \bar{S_1})$, and $(S_2, \bar{S_2})$.

Figure 1: A very simple $\frac{2}{5}$-approximation algorithm for the MAX DI-CUT problem.

2 Preliminaries

Given a directed graph $G = (V, A)$, and a vertex $v \in V$, we let $d_{in}(G, v)$ and $d_{out}(G, v)$, respectively, denote the indegree and outdegree of v in G. If $S \subseteq V$ is a set of vertices, we let $\bar{S} = V - S$. Given a subset of arcs $B \subseteq A$, we let $opt(B)$ denote the maximum size of a directed cut in the subgraph (V, B). We also let $d_{in}(B, v)$ and $d_{out}(B, v)$, respectively, be the indegree and outdegree of v in the subgraph (V, B). Given a directed cut $(S, \bar{S})$ of $G = (V, A)$, we let $ratio(B, S) = |(S, \bar{S}) \cap B| / opt(B)$, where $(S, \bar{S})$ is interpreted here as the set of edges in the directed cut. Thus $ratio(B, S)$ is the local ratio achieved by the directed cut $(S, \bar{S})$ on the set B. We also use this notation when $(S, \bar{S})$ is a random cut produced by some algorithm, in which case $|(S, \bar{S}) \cap B|$ is replaced by $E[|(S, \bar{S}) \cap B|]$, the expected size of the intersection.

It is easy to see that if $A = \cup_{i=1}^{k} A_i$, where the A_i's are disjoint, and if an algorithm produces a (random) cut S such that $ratio(A_i, S) \geq \alpha$, for every $1 \leq i \leq k$, then the algorithm is an α-approximation algorithm for MAX DI-CUT.

Our first two algorithms use such a partition of the edges in which each A_i is either an edge, a directed path of length two, or a directed triangle. Note that in each one of these cases we have $opt(A_i) = 1$.

If X is a set and $0 \leq p \leq 1$, we let $\mathrm{RAND}(X, p)$ denote a random subset of X obtained by picking each element of X independently with probability p.

3 A very simple $\frac{2}{5}$-approximation algorithm

A very simple $\frac{2}{5}$ approximation algorithm for the MAX DI-CUT problem, called DC1, is given in Figure 1. The algorithm greedily finds directed paths of length 2 in the graph and removes them. Let P be the set of paths obtained in this way, and let $G' = (V, A')$ be the graph obtained after the removal of all these paths. It is easy to see that for each vertex $v \in V$, either $d_{in}(G', v) = 0$ or $d_{out}(G', v) = 0$, otherwise another directed path of length two can be found. The algorithm lets S_1 be the set of vertices with outgoing edges in G', and S_2 be a random subset of V. It then compares the sizes of the cuts $(S_1, \bar{S_1})$ and $(S_2, \bar{S_2})$ and outputs the larger one of them. It is easy to see that the algorithm can be implemented to run in linear time.

Theorem 3.1 *Algorithm* DC1 *is a linear time $\frac{2}{5}$-approximation algorithm for the MAX DI-CUT problem.*

Proof: Let m_1 be the number of edges left in G' after the removal of all the directed paths of length two, and let m_2 be the number of edges that belong to the removed paths. As a directed cut can contain at most one edge from each directed path of length two, we get that the size of the maximum directed cut of G is at most $m_1 + \frac{m_2}{2}$. The cut $(S_1, \bar{S_1})$ cuts all the edges left in G' and its size, therefore, is at least m_1. The cut, $(S_2, \bar{S_2})$, on the other hand, is a random cut so its expected size is $\frac{m_1+m_2}{4}$. The performance ratio achieved by the algorithm is therefore at least

$$\frac{\max\{\, m_1 \,,\, \frac{m_1+m_2}{4} \,\}}{m_1 + \frac{m_2}{2}} \;\geq\; \frac{\frac{1}{5}\, m_1 + \frac{4}{5}\, \frac{m_1+m_2}{4}}{m_1 + \frac{m_2}{2}} \;=\; \frac{2}{5}\,.$$

■

4 A simple $\frac{9}{20}$-approximation algorithm

An improved approximation algorithm for the MAX DI-CUT, called DC2, is given in Figure 2. Although its description is a bit longer than that of DC1, it is also quite simple, and can also be implemented to run in linear time. It achieves a performance ratio of $\frac{9}{20}$.

Algorithm DC2 maintains a set B of edges, a set P of directed paths of length two, and a set T of directed triangles. Initially, $B \leftarrow A$, the set of edges of the input graph, while P and T are empty. Just like algorithm DC1, it begins by greedily constructing paths of length two, as long as that is possible (rule (a)). Unlike algorithm DC1, it is willing to break up a path that was already constructed, if two new paths could be formed. For example, if $(u, v, w) \in P$ and $(u', u), (v', v) \in B$ are edges that were not used yet in paths, then rule (b) allows the algorithm to replace the path (u, v, w) with the two paths (u', u, v) and (v', v', w). The other rules are similar in nature. Rule (c) is analogous to rule (b). Rule (d) essentially breaks a path of length four into two paths of length two. Rule (e) allows the algorithm to construct triangle. Rules (f) and (g) allow the algorithm to break triangles if two paths could be constructed.

We note that step 2 of the algorithm could be implemented to run in linear time. To see this note that each rule reduces the number of edges in B. Thus, the total number of rule applications is bounded by the number of edges of the graph. Now, as the set B only gets smaller, for each path or triangle constructed by the algorithm we only have to check once whether further rules could be applied to it.

When no further rules could be applied, the algorithm partitions the vertices of G into three disjoint sets X_1, X_2 and X_3. The set X_1 is composed of all the vertices that have outgoing edges in the subgraph (V, B). The set X_2 is composed of all the vertices that have incoming edges in (V, B), while X_3 is the set of isolated vertices in (V, B). The sets X_1, X_2 and X_3 are disjoint as otherwise another paths of length two could formed. We claim the following:

Lemma 4.1 *(i) If $(u, v, w) \in P$ then ($u \notin X_2$ or $v \notin X_2$) and ($v \notin X_1$ or $w \notin X_1$) and ($v \notin X_2$ or $w \notin X_1$). (ii) If $(u, v, w) \in T$ then $u, v, w \in X_3$.*

Proof: It is easy to see that if one of the specified conditions does not hold than at least one rule could be applied. For example, if $(u, v, w) \in P$ and $u, v \in X_2$, then rule (b) could be applied. ∎

Lemma 4.2 *For every path $p \in P$, at least one of the following holds:*

1. $\text{ratio}(p, S_1) \geq \frac{1}{2}$.
2. $\text{ratio}(p, S_1) \geq \frac{1}{4}$ *and* $\text{ratio}(p, S_2) \geq \frac{3}{4}$.

Proof: Let $p = (u, v, w) \in P$. As we have seen, a directed path can cut at most one of the edges of p and therefore $opt(p) = 1$. Thus $ratio(p, S_i)$ is the expected number of arcs of p which are in the cut $(S_i, \bar{S}_i)$. Consider the following cases:

- $\underline{u \in X_1}$: If $v \in X_2 \cup X_3$, then we clearly have $ratio(p, S_1) \geq \frac{1}{2}$. Otherwise, $v \in X_1$ and by Lemma 4.1 we get that $w \in X_2 \cup X_3$ and we again have $ratio(p, S_1) \geq \frac{1}{2}$.
- $\underline{w \in X_2}$: This case is analogous to the previous case.
- $\underline{u \in X_2}$: By Lemma 4.1, we get that $v \notin X_2$ and $w \notin X_1$. We may also assume that $w \notin X_2$, otherwise the previous case would apply. Thus $w \in X_3$. If $v \in X_1$ then we again get $ratio(p, S_1) \geq \frac{1}{2}$. Otherwise, $v \in X_3$ and it is easy to check that $ratio(p, S_1) \geq \frac{1}{4}$ and $ratio(p, S_2) \geq \frac{3}{4}$.
- $\underline{w \in X_1}$: This case is analogous to the previous case.
- $\underline{u \in X_3}$: We may again assume that $w \in X_3$ and no matter where v is, we have $ratio(p, S_1) \geq \frac{1}{2}$.
- $\underline{w \in X_3}$: This case is analogous to the previous case.

This completes the proof of the lemma. ∎

Lemma 4.3 *For every triangle $t \in T$ we have* $\text{ratio}(t, S_1) = \text{ratio}(t, S_2) = \text{ratio}(t, S_3) = \frac{3}{4}$.

Proof: Let $t = (u, v, w) \in T$. It is easy to see that $opt(t) = 1$. By Lemma 4.1, we get that $u, v, w \in X_3$. Thus, $ratio(t, S_i) = \frac{3}{4}$, for $1 \leq i \leq 3$, as required. ∎

Theorem 4.4 *Algorithm* DC2 *is a linear-time $\frac{9}{20}$-approximation algorithm for the MAX DI-CUT problem.*

Proof: If $e \in B$, then $ratio(e, S_1) = 1$, $ratio(e, S_2) = 0$ and $ratio(e, S_3) = \frac{1}{4}$. Also note that if $p \in P$ then an

ALGORITHM DC2:

Input: A directed graph $G = (V, A)$.
Output: A directed cut whose size, on average, is at least $\frac{9}{20}$ of the size of an optimal directed cut.

1. Let $B \leftarrow A$, $P \leftarrow \phi$, $T \leftarrow \phi$.

2. While possible, apply one of the following rules:

$$
\begin{array}{lccc}
(a) & (u,v),(v,w) \in B & \Longrightarrow & \begin{array}{c} P \leftarrow P \cup \{(u,v,w)\} \\ B \leftarrow B - \{(u,v),(v,w)\} \end{array} \\
(b) & \begin{array}{c} (u,v,w) \in P \\ (u',u),(v',v) \in B \end{array} & \Longrightarrow & \begin{array}{c} P \leftarrow P \cup \{(u',u,v),(v',v,w)\} - \{(u,v,w)\} \\ B \leftarrow B - \{(u',u),(v',v)\} \end{array} \\
(c) & \begin{array}{c} (u,v,w) \in P \\ (v,v'),(w,w') \in B \end{array} & \Longrightarrow & \begin{array}{c} P \leftarrow P \cup \{(u,v,v'),(v,w,w')\} - \{(u,v,w)\} \\ B \leftarrow B - \{(v,v'),(w,w')\} \end{array} \\
(d) & \begin{array}{c} (u,v,w) \in P \\ (u',u),(w,w') \in B \end{array} & \Longrightarrow & \begin{array}{c} P \leftarrow P \cup \{(u',u,v),(v,w,w')\} - \{(u,v,w)\} \\ B \leftarrow B - \{(u',u),(w,w')\} \end{array} \\
(e) & \begin{array}{c} (u,v,w) \in P \\ (w,u) \in B \end{array} & \Longrightarrow & \begin{array}{c} P \leftarrow P - \{(u,v,w)\} \\ T \leftarrow T \cup \{(u,v,w)\}\ ;\ B \leftarrow B - \{(w,u)\} \end{array} \\
(f) & \begin{array}{c} (u,v,w) \in T \\ (u,u') \in B \end{array} & \Longrightarrow & \begin{array}{c} P \leftarrow P \cup \{(u,v,w),(w,u,u')\} \\ T \leftarrow T - \{(u,v,w)\}\ ;\ B \leftarrow B - \{(u,u')\} \end{array} \\
(g) & \begin{array}{c} (u,v,w) \in T \\ (w',w) \in B \end{array} & \Longrightarrow & \begin{array}{c} P \leftarrow P \cup \{(u,v,w),(w',w,u)\} \\ T \leftarrow T - \{(u,v,w)\}\ ;\ B \leftarrow B - \{(w',w)\} \end{array}
\end{array}
$$

3. Let $X_1 \leftarrow \{v \in V \mid d_{out}(B,v) > 0\}$,
   $X_2 \leftarrow \{v \in V \mid d_{in}(B,v) > 0\}$,
   $X_3 \leftarrow V - (X_1 \cup X_2)$.

4. Let $S_1 \leftarrow X_1 \cup \mathrm{RAND}(X_3, \frac{1}{2})$,
   $S_2 \leftarrow X_2 \cup \mathrm{RAND}(X_3, \frac{1}{2})$,
   $S_3 \leftarrow \mathrm{RAND}(V, \frac{1}{2})$.

5. Output the larger of the cuts $(S_1, \bar{S}_1)$, $(S_2, \bar{S}_2)$ and $(S_3, \bar{S}_3)$.

Figure 2: A simple $\frac{9}{20}$-approximation algorithm for the MAX DI-CUT problem.

addition to the conditions of Lemma 4.2 we also have $\mathit{ratio}(p, S_3) = \frac{1}{2}$.

Consider a variant of algorithm DC2 in which $(S_1, \bar{S}_1)$ is chosen with probability $\frac{3}{10}$, $(S_2, \bar{S}_2)$ is chosen with probability $\frac{1}{10}$ and $(S_3, \bar{S}_3)$ is chosen with probability $\frac{3}{5}$. Let $(S, \bar{S})$ be the random cut produced. Let $e \in B$, let $p_1 \in P$ be such that $\mathit{ratio}(p, S_1) \geq \frac{1}{2}$, let $p_2 \in P$ be such that $\mathit{ratio}(p, S_1) \geq \frac{1}{4}$ and $\mathit{ratio}(p, S_2) \geq \frac{3}{4}$, and let $t \in T$. Then it is easy to verify that

$$
\begin{array}{rcl}
\mathit{ratio}(e) & \geq & \frac{3}{10} \cdot 1 + \frac{1}{10} \cdot 0 + \frac{3}{5} \cdot \frac{1}{4} \;=\; \frac{9}{20} \\
\mathit{ratio}(p_1) & \geq & \frac{3}{10} \cdot \frac{1}{2} + \frac{1}{10} \cdot 0 + \frac{3}{5} \cdot \frac{1}{2} \;=\; \frac{9}{20} \\
\mathit{ratio}(p_2) & \geq & \frac{3}{10} \cdot \frac{1}{4} + \frac{1}{10} \cdot \frac{3}{4} + \frac{3}{5} \cdot \frac{1}{2} \;=\; \frac{9}{20} \\
\mathit{ratio}(t) & \geq & \frac{3}{10} \cdot \frac{3}{4} + \frac{1}{10} \cdot \frac{3}{4} + \frac{3}{5} \cdot \frac{3}{4} \;>\; \frac{9}{20}
\end{array}
$$

Thus, this variant is a $\frac{9}{20}$-approximation algorithm for the MAX DI-CUT problem. Algorithm DC2 is at least as good as it chooses the larger of the cuts $(S_1, \bar{S}_1)$, $(S_2, \bar{S}_2)$ and $(S_3, \bar{S}_3)$. ■

5 A combinatorial $\frac{1}{2}$-approximation algorithm

In this section we present a combinatorial $\frac{1}{2}$-approximation algorithm for the MAX DI-CUT problem. We do this by considering the standard linear programming relaxation of the problem and showing that an optimal solution of it can be found using combinatorial means.

Following Goemans and Williamson [GW94] and Trevisan [Tre98a], we consider the following standard linear programming relaxation of the maximum directed cut problem:

$$(5.1)\qquad \boxed{\begin{array}{lll} \text{Maximize} & \displaystyle\sum_{(i,j)\in A} z_{ij} & \\ \text{subject to} & z_{ij} \le x_i & ,\ (i,j)\in A \\ & z_{ij} \le 1-x_j & ,\ (i,j)\in A \\ & 0 \le x_i \le 1 & ,\ i \in V \end{array}}$$

Trevisan [Tre98a] shows that a $\frac{1}{2}$-approximation algorithm for the MAX DI-CUT problem can be obtained by solving this linear program and the using a simple randomized rounding technique that puts vertex i in the set S with probability $f(x_i)$, where $f(x) = \sqrt{\frac{x}{2}}$, if $0 \le x \le \frac{1}{2}$, and $f(1-x) = 1 - f(x)$, for $\frac{1}{2} \le x \le 1$.

We shall describe a simple combinatorial algorithm for finding a half-integral optimal solution of this relaxation, i.e., $x_i \in \{0, \frac{1}{2}, 1\}$, for every $1 \le i \le n$. As $f(x) = x$, for $x \in \{0, \frac{1}{2}, 1\}$, we get that by putting each vertex $i \in V$ in S with probability x_i, we obtain a directed cut whose expected size is at least $\frac{z^*}{2}$, where z^* is the optimal value of the relaxation. As $z^* \ge opt(G)$, we would obtain a $\frac{1}{2}$-approximation algorithm. This is stated in the following lemma.

Lemma 5.1 *Let $\{x_i\}_{i=1}^n$ be a half-integral optimal solution of the linear programming relaxation of a MAX DI-CUT instance. Then, the expected size of the cut obtained by putting each vertex i in S, independently, with probability x_i is at least half of the optimum.*

How do we find, using combinatorial means, a half-integral optimal solution of (5.1)? We begin by eliminating the variables x_i from the program and obtain the following equivalent, but slightly larger, linear program:

$$(5.2)\qquad \boxed{\begin{array}{lll} \text{Maximize} & \displaystyle\sum_{(i,j)\in A} z_{ij} & \\ \text{subject to} & z_{ij} + z_{jk} \le 1 & ,\ (i,j),(j,k)\in A \\ & 0 \le z_{ij} \le 1 & ,\ (i,j)\in A \end{array}}$$

Lemma 5.2 *The linear programs (5.1) and (5.2) are equivalent.*

Proof: Suppose that $\{z_{ij}\}$ and $\{x_i\}$ is a feasible solution of (5.1). Let $(i,j),(j,k) \in A$. Then $z_{ij} \le 1 - x_j$ and $z_{jk} \le x_j$. Thus $z_{ij} + z_{jk} \le 1$, and z_{ij} is also a solution of (5.2).

Conversely, let $\{z_{ij}\}$ be a feasible solution of (5.2). If $\{j : (i,j) \in A\} \ne \phi$, set $x_i = \max_{j:(i,j)\in A} z_{ij}$. Otherwise set $x_i = 0$. Then, clearly, $z_{ij} \le x_i$, for all $(i,j) \in A$. For every $(i,j) \in A$, if $x_j = 0$, then clearly, $z_{ij} \le 1 - x_j$, and otherwise, let k be such that $x_j = z_{jk}$. Then, as $z_{ij} + z_{jk} \le 1$, we get that $z_{ij} \le 1 - x_j$. ■

We introduce now the following definitions:

Definition 5.3 (Line graph) *Let $G = (V,E)$ be a directed graph. The* line graph *of G is a graph $L(G) = (E,P)$, such that $((u_1,v_1),(u_2,v_2)) \in P$ if and only if $v_1 = u_2$. In other words, the line graph of G is a graph whose vertices are the edges of G, and two edges of G are connected by an edge in $L(G)$ if and only if they form a path of length two. The graph $L(G)$ is considered to be an undirected graph.*

Definition 5.4 [Fractional independent set and fractional vertex cover] *Let $G = (V,E)$, where $V = \{1,2,\ldots,n\}$, be an undirected graph. A fractional independent set of G is a vector $x = (x_1,x_2,\ldots,x_n)$ such that $0 \le x_i \le 1$, for every $1 \le i \le n$, and $x_i + x_j \le 1$, for every $(i,j) \in E$. The size of such a fractional independent set is $\sum_{i=1}^n x_i$. A fractional vertex cover of G is a vector $y = (y_1,y_2,\ldots,y_n)$ such that that $0 \le y_i \le 1$, for every $1 \le i \le n$, and $y_i + y_j \ge 1$, for every $(i,j) \in E$. The size of such a fractional vertex cover is $\sum_{i=1}^n y_i$. It is easy to see that x is a fractional independent set if and only if $1-x$ is a fractional vertex cover.*

Having another look at the linear program (5.2), we realize that

Lemma 5.5 *The optimal solution of (5.2) is a maximum fractional independent set of $L(G)$, the line graph of the input graph G.*

We now rely on the well known fact that any graph has a half-integral optimal fractional vertex cover, and therefore also a half-integral optimal fractional independent set. (See Nemhauser and Trotter [NT75], Lovász and Plummer [LP86])

Theorem 5.6 *A half-integral minimum fractional vertex cover of an undirected graph $G = (V, E)$ can be found by solving a bipartite matching problem on a graph with $2|V|$ vertices and $2|E|$ edges.*

Proof: (Sketch.) For the (easy) proof that a half-integral minimum fractional vertex cover always exists see [LP86]. To find such an optimal cover, construct a bipartite graph $G_{12} = (V_{12}, E_{12})$ in which each vertex $v \in V$ is replaced by two vertices $v_1, v_2 \in V_{12}$, and each edge (u, v) is replaced by the two edges $(u_1, v_2), (v_1, u_2) \in E_{12}$. Now, find a maximum matching in $G_{12} = (V_{12}, E_{12})$. From this maximum matching it is easy to construct a minimum vertex cover of G_{12}. Now, let $x^1 = (x^1_1, \ldots, x^1_n)$ be 0-1 vectors such that $x^1_v = 1$ if and only if v_1 belongs to the minimum vertex cover of G_{12}. The vector $x^2 = (x^2_1, \ldots, x^2_n)$ is defined similarly. It is easy to verify that $\frac{x^1+x^2}{2}$ is a half-integral optimal fractional vertex cover of G. ■

A maximum matching in a bipartite graph $G = (V, E)$ with $n = |V|$ and $m = |E|$ can be found in $O(m\sqrt{n})$ time (Micali and Vazirani [MV80]). As the line graph $L(G)$ may have at most mn edges, we get therefore the following result:

Theorem 5.7 *There is a combinatorial $\frac{1}{2}$-approximation algorithm for the MAX DI-CUT problem that runs in $O(n^{3/2}m)$ time.*

If we a willing to settle for $(\frac{1}{2} - \epsilon)$-approximation algorithms, for any ϵ we can obtain faster running times using a linear time $(1-\epsilon)$-approximation algorithm for the maximum matching problem (Hassin and Lahav [HL94]).

We note in passing that as maximum matching belongs to RNC (Mulmuley, Vazirani and Vazirani [MVV87]), the $\frac{1}{2}$-approximation algorithm described in this section can also be implemented in RNC. Trevisan [Tre98b] obtains, for any $\epsilon > 0$, an NC $(\frac{1}{2} - \epsilon)$-approximation algorithm.

6 Further results

In the previous section we considered the linear programming relaxation of the MAX DI-CUT problem and showed that it is equivalent to the linear programming relaxation of the maximum independent set problem of the line graph. The same is true also for the original problems.

Theorem 6.1 *Let $G = (V, E)$ be a directed graph. Then, the size of the maximum directed cut of G is equal to the size of the maximum independent set of the line graph $L(G)$.*

Proof: Let $(S, \bar{S})$ be a directed cut in G. If (u_1, v_1) and (u_2, v_2) are two edges in this cut then they do not form a directed path of length two. Thus, the edges in the directed cut form an independent set of $L(G)$.

Conversely, let $I \subseteq A$ be an independent set of $L(G)$. By the definition of $L(G)$, there is no vertex $v \in V$ with an arc of I entering it, an arc in I leaving it. Let $S \subseteq V$ be the set of with arcs of I leaving them. Clearly, $|(S, \bar{S})| \geq |I|$, as required. ■

This theorem has the following interesting consequence:

Theorem 6.2 *The MAX DI-CUT problem for bipartite graphs can be solved in polynomial time.*

Proof: Suppose that $G = (V, E)$ is a directed bipartite graph. Let $V = X \cup Y$, where $X \cap Y = \phi$ be such that for any $(u, v) \in E$ either $u \in X$ and $v \in Y$, or $u \in Y$ and $v \in X$. We claim that $L(G)$, the line graph of G is also bipartite. Let (X, Y) be the set of arcs from X to Y, and let (Y, X) be the set of arcs from Y to X. Then, clearly, (X, Y) and (Y, X) are independent sets in the line graph G_l. Thus, $L(G)$ is indeed bipartite. As $L(G)$ is bipartite, a minimum vertex cover, and therefore a maximum independent set of it, could be found in polynomial time. From this independent set we can construct a maximum directed cut, as in the proof of Theorem 6.1. ■

7 Concluding remarks

We presented several combinatorial approximation algorithms for the MAX DI-CUT problem. We also discovered an interesting connection between the MAX DI-CUT problem and the problem of finding maximum independent sets in line graphs of directed graphs.

References

[Ali97] P. Alimonti. Non-oblivious local search for MAX 2-CCSP with application to MAX DICUT. In *Proceedings of the 23rd International Workshop on Graph-Theoretic Concepts in Computer Science*, 1997.

[AS92] N. Alon and J.H. Spencer. *The probabilistic method*. Wiley, 1992.

[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.

[GW94] M.X. Goemans and D.P. Williamson. New 3/4-approximation algorithms for the maximum satisfiability problem. *SIAM Journal on Discrete Mathematics*, 7:656–666, 1994.

[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.

[Hås97] J. Håstad. Some optimal inapproximability results. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing, El Paso, Texas*, pages 1–10, 1997. Full version available as E-CCC Report number TR97-037.

[HL94] R. Hassin and S. Lahav. Maximizing the number of unused colors in the vertex coloring problem. *Information Processing Letters*, 52:87–90, 1994.

[KL96] P. Klein and H. Lu. Efficient approximation algorithms for semidefinite programs arising from MAX CUT and COLORING. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing, Philadelphia, Pennsylvania*, pages 338–347, 1996.

[KMSV94] S. Khanna, R. Motwani, M. Sudan, and U. Vazirani. On syntactic versus computational views of approximability. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science, Santa Fe, New Mexico*, pages 819–830, 1994.

[LP86] L. Lovász and M.D. Plummer. *Matching theory*. North-Holland Publishing Co., Amsterdam, 1986. Annals of Discrete Mathematics, 29.

[MV80] S. Micali and V. V. Vazirani. An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In *Proceedings of the 21th Annual IEEE Symposium on Foundations of Computer Science, Syracuse, New York*, pages 17–27, 1980.

[MVV87] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7:105–113, 1987. An elegant parallel, randomized algorithm for finding a perfect matching in a simple graph based on Tutte's matrix is presented. The algorithm, which is made possible by a probabilistic lemma called the isolation lemma, requires inversion of a single integer matrix which can be parallelized.

[NT75] G.L. Nemhauser and L.E. Trotter. Vertex packings: structural properties and algorithms. *Math. Programming*, 8:232–248, 1975.

[Tre98a] L. Trevisan. Parallel approximation algorithms by positive linear programming. *Algorithmica*, 21, 1998.

[Tre98b] L. Trevisan. Recycling queries in PCPs and in linearity tests. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing, Dallas, Texas*, pages 299–308, 1998.

[TSSW96] L. Trevisan, G.B. Sorkin, M. Sudan, and D.P. Williamson. Gadgets, approximation, and linear programming (extended abstract). In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science, Burlington, Vermont*, pages 617–626, 1996.

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