A 0.5-APPROXIMATION ALGORITHM FOR MAX DICUT WITH GIVEN SIZES OF PARTS

ALEXANDER AGRACHEV1, RAFAEL RASSIN2, AND MAXIM SUBEROVNI3

Abstract. Given a directed graph $G$ and an arc weight function $w : E(G) \rightarrow \mathbb{R}_+$, the maximum directed cut problem (MAX DICUT) is that of finding a directed cut $S \cup \bar{S}$ of maximum total weight. In this paper we consider a version of MAX DICUT—MAX DICUT with given sizes of parts of MAX DICUT with $\alpha$—basic instance is that of MAX DICUT plus a positive integer $p$, and it is required to find a directed cut $S \cup \bar{S}$ having maximum weight over all cuts $S \cup \bar{S}$ with $|S| = p$. Our main result is a 0.5-approximation algorithm for solving this problem. The algorithm is based on a trickly application of the pigeons-converting two-ways developed in some earlier papers by two of the authors and a remarkable structural property of basic solutions to a linear relaxation. The property is that each component of any basic solution is an element of a set $\{0, \frac{1}{2}, 1 - \frac{1}{32}, \frac{3}{4}, 1\}$, where $\frac{1}{2}$ is a constant that satisfies $0 < \frac{1}{2} < \frac{1}{2}$ and is the same for all components.

Key words. approximation algorithm, directed cut, linear relaxation, basic solution

AMS subject classifications. 68W25, 60C38, 60C27, 90C57

1. Introduction. Let $G$ be a directed graph. A directed cut in $G$ is defined to be the set of arcs leaving some vertex subset $X$ (we denote it by $\partial(X)$). Given a directed graph $G$ and an arc weight function $w : E(G) \rightarrow \mathbb{R}_+$, the maximum directed cut problem (MAX DICUT) is that of finding a directed cut $S \cup \bar{S}$ with $|S|$ with maximum total weight. In this paper we consider a version of MAX DICUT (MAX DICUT with given sizes of parts of MAX DICUT with GRP), whose instance is that of MAX DICUT plus a positive integer $p$, and it is required to find a directed cut $S \cup \bar{S}$ having maximum weight over all cuts $S \cup \bar{S}$ with $|S| = p$. MAX DICUT is well known to be NP-hard and so is MAX DICUT with GRP as the former evidently reduces to the latter.

The NP-hardness of MAX DICUT follows from the observation that the well-known undirected version of MAX DICUT—the maximum cut problem (MAX CUT), which is on the original Karö's list of NP-complete problems [K77]—reduces to MAX DICUT by substituting each edge for two oppositely oriented arcs. This means that for both problems there is no choice but to develop approximation algorithms. Nevertheless, this task turned out to be highly nontrivial, as for a long time it was an open problem whether it is possible to design approximations with factors better than trivial $1/2$ for MAX CUT and $1/4$ for MAX DICUT. Only quite recently, using a novel technique of rounding semidefinite relaxations, Goemans and Williamson [GW95] worked out an algorithm solving MAX CUT and MAX DICUT approximately within factors of 0.878 and 0.796, respectively. A half later Feige and Goemans [FG95] developed an algorithm for MAX DICUT with a better approximation ratio of 0.855. Recently, using a new

---

1. Revised by the editors February 18, 2000; accepted for publication (in revised form) January 10, 2001; published electronically April 3, 2003.

[1] Schofield Institute of Mathematics, pr. Krymskaya 4, Novosibirsk, Russia, (ag@math. necr.ru). The research of this author was partially supported by the Russian Foundation for Basic Research, grant 99-01-00631.

[2] Department of Statistics and Operations Research, School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, Israel (facebook/math.tau.ac.il).

[3] IBM T. J. Watson Research Center, P.O. Box 707, Yorktown Heights, NY 10608 (rafael@wrc. com). The research of this author was partially supported by the Russian Foundation for Basic Research, grant 99-01-00510.

---
method of rounding linear relaxations—the piping rounding—Agar and Srivasteho [AS99] developed a 0.5-approximation algorithm for the version of Max Cut in which the parts of a vertex set bipartition are constrained to have given sizes (Max Cut with given sizes of parts of Max Cut with GIP). Later Henry and Rubinstein [HR06] presented a different 0.5-approximation with a better running time. The paper [AS99] presents an extension of the algorithm in [AS99] to a hypergraph generalization of Max Cut with GIP. Feige and Langberg [FL09] combined the method in [AS99] with the semidefinite programming approach to design a 0.5+ε-approximation for Max Cut with GIP, where ε is some unspecified small positive number.

It is easy to see that Max Cut with GIP reduces to Max Cut with GIP in the same way as Max Cut reduces to Max Cut. However, unlike Max Cut with GIP, Max Cut with GIP provides no possibilities for a straightforward application of the piping rounding since the F/L lower bound condition in the description of the method (see section 2) does not, in general, hold.

Fortunately, the other main condition, ε-convexity, always holds. In the final section of this paper, we show that the F/L lower bound condition is still satisfied with C = 0.5 in the case where the arc weights form a circulation in the given graph as well as when the parts of a cut are restricted to have the same size (the Digraph Orientation problem). Thus, these cases can be approximated within a factor of 0.5 by the direct application of the piping rounding method.

The main result of this paper is an algorithm that finds a feasible cut of weight within a factor of 0.5 in the case of arbitrary weights. It turns out that to construct such an algorithm one needs to carry out a more profound study of the problem structure. A house-sat opportunity is provided by a remarkable structural property of basic solutions to a linear relaxation (Theorem 4.1). At this point we should notice the papers of Jain [Ja98] and Mellkonian and Taras [MT99], where exploiting structural properties of basic solutions was also crucial in designing better approximations for many network design problems.

The resulting algorithm (DIRCUT) is of rounding type and as such consists of two phases: the first phase is to find an optimal (fractional) solution to a linear relaxation; the second (rounding) phase is to transform this solution to a feasible (integral) solution. A special feature of the rounding phase is that it uses two different rounding algorithms (ROUND, and Rounding) based on the piping rounding method and takes the best solution for the output. The worst-case analysis of the algorithm relies heavily on Theorem 4.1.

2. Piping rounding: A general scheme. In this section, to make the paper self-contained, we give a general description of the piping rounding method as it was presented in [AS99].

Assume that a problem P can be formulated as the following nonlinear binary program:

\[
\begin{align*}
\text{max} & \quad F(x) \\
\text{subject to} & \quad \sum_{i=1}^{n} x_i = p, \\
& \quad 0 \leq x_i \leq 1, \quad i = 1, \ldots, n, \\
& \quad x_i \in \{0, 1\}, \quad i = 1, \ldots, n,
\end{align*}
\]

where \( p \) is a positive integer and \( F(x) \) is a function defined on the rational points \( x = (x_i) \) of the n-dimensional cube [0, 1]^n and computable in polynomial time. Further
assumption that one can associate with \( F(x) \) another function, \( L(x) \), which is defined on the same set, coincides with \( F(x) \) on binary \( x \) satisfying (2.2), and the program

\[
\begin{align*}
& \max L(x) \\
& \text{s.t. } \sum_{i=1}^{n} x_i = p, \\
& \quad 0 \leq x_i \leq 1, \quad i = 1, \ldots, n,
\end{align*}
\]

which we call a nice relaxation, is polynomially solvable. Next assume that the following two main conditions hold:

- **F/L lower bound condition:** there exists a constant \( C \) such that \( 0 < C \leq 1 \) and \( F(x) \geq C L(x) \) for each rational point \( x \in \mathbb{N}^{n} \).

- **\( \varepsilon \)-convexity condition:** the function

\[
F(x, x_{i}, j) = F(x_{1}, \ldots, x_{i} + \varepsilon, \ldots, x_{j} - \varepsilon, \ldots, x_{n})
\]

is convex with respect to \( \varepsilon \in [-\min(x_{i}, 1-x_{j}), \min(1-x_{i}, x_{j})] \) for each pair of indices \( i \) and \( j \) and each \( x \in \mathbb{N}^{n} \).

We now describe the pigeo rounding procedure. Its input is a fractional solution \( z \) satisfying (2.2)-(2.3) and its output is an integral solution satisfying (2.2)-(2.4) and having the property that \( F(z) \geq F(x) \). The pigeon rounding consists of uniform "pigeon steps." We describe the first step. If the solution \( z \) is not binary, then due to (2.2) it has at least two different components \( x_{i} \) and \( x_{j} \) with values lying strictly between 0 and 1. By \( \varepsilon \)-convexity condition, \( F(x, x_{i}, j) \geq F(z) \) either for \( \varepsilon = \min(1-x_{i}, x_{j}) \) or \( \varepsilon = -\min(x_{i}, 1-x_{j}) \). Then we obtain a new feasible solution \( z' = (x_{1}, \ldots, x_{i} + \varepsilon, \ldots, x_{j} - \varepsilon, \ldots, x_{n}) \) with a smaller number of noninteger components and such that \( F(z') \geq F(z) \). After repeating the "pigeon" step at most \( n-1 \) times we arrive at a binary feasible solution \( z \) with \( F(z') \geq F(z) \). Since each step can be performed in polynomial time, the overall running time of the described procedure is polynomially bounded.

Now suppose that \( z \) is an optimal solution to (2.5)-(2.7), satisfying both the \( \varepsilon \)-convexity and the \( F/L \) lower bound condition. Then \( F(z) \geq F(x) \geq C L(x) \geq CF^{*} \), where \( F^{*} \) is the optimal value of (2.2)-(2.4). Thus the algorithm consisting of a polynomial-time procedure to solve the nice relaxation (2.5)-(2.7) and the pigeon rounding finds a feasible solution to (2.1)-(2.4) of weight within a factor of \( C \) of the optimum. Note that if instead of a procedure for solving (2.5)-(2.7) we use any polynomial-time procedure to find a solution \( z \) satisfying (2.2)-(2.3) and \( F(z) \geq CF^{*} \), then we also obtain a \( C \)-approximation algorithm.

3. Application: MAX DICUT WITH GFP. In this section, we show an implementation of the above scheme in the case of MAX DICUT WITH GFP and, on the side, specify the character of obstacles to the direct application of the pigeon rounding method.

In what follows, \( G = (V, A) \) stands for the graph in the input of MAX DICUT WITH GFP. Since assigning zero weights to missing arcs yields an equivalent problem, we may assume that \( A \) contains all possible arcs. Let \( |V| = n \).

First, note that MAX DICUT WITH GFP can be formulated as the following non-
linear binary program:

$$\text{max } F(x) = \sum_{i \in A} w_i x_i (1 - x_i)$$

s.t. $$\sum_{i \in V} x_i = p,$$

$$x_i \in \{0, 1\}$$ for all $$i \in V.$$

Second, just like MAX CUT with CSP in [AS99], MAX CUT with CSP can be formulated as the following integer program:

$$\text{max } \sum_{i \in A} w_i z_{ij}$$

s.t. $$z_{ij} \leq x_i$$ for all $$ij \in A,$$

$$z_{ij} \leq 1 - x_j$$ for all $$ij \in A,$$

$$\sum_{i \in V} x_i = p,$$

$$0 \leq z_{ij} \leq 1$$ for all $$i \in V, \forall j \in A.$$ 

Now observe that the variables $$z_{ij}$$ can be eliminated from (3.1)–(3.5) by setting

$$z_{ij} = \min\{x_i, (1 - x_j)\}$$ for all $$ij \in A.$$

Hence (3.1)–(3.5) is equivalent to minimizing

$$L(x) = \sum_{i \in A} w_i \min\{x_i, (1 - x_i)\}$$

subject to (3.4), (3.5).

Thus we have functions $$F$$ and $$L$$ that can be considered as those involved in the description of the pipage rounding (see section 2). Notice now that for each pair of indices $$i$$ and $$j$$ the function $$\varphi(c, x, i, j)$$ defined by (2.8) is the sum of $$w_i (x_i + c) [1 - (x_j - c)] + w_j (x_j - c) [1 - (x_i + c)]$$ and a term linear in $$c$$. It follows that $$\varphi(c, x, i, j)$$ is a quadratic polynomial in $$c$$ having a nonnegative leading coefficient for each pair of indices $$i$$ and $$j$$ and each $$c \in [0, 1]^p$$. Thus, the function $$F$$ obeys the c-convexity condition. Unfortunately, the other, $$F/L$$ lower bound condition, does not hold for any $$C > 0$$. We present below an example showing that the ratio $$F(x)/L(x)$$ may be arbitrarily close to 0 even when the underlying graph is bipartite.

Example 1. Consider the following instance of MAX CUT with CSP. Let $$V = V_1 \cup V_2 \cup V_3$$, where $$|V_1| = k$$, $$|V_2| = |V_3| = 2$$. Let $$A = A_1 \cup A_2$$, where $$A_1$$ is the set of $$2k$$ arcs from $$V_1$$ to $$V_2$$ inducing a complete bipartite graph on $$V_1, V_2$$ and $$A_2$$ is the set of 4 arcs from $$V_2$$ to $$V_3$$ inducing a complete bipartite graph on $$V_2, V_3$$ (see Figure 3.1). Assume the arc weights to be unit and $$\rho \leq k$$.

Then for any feasible solution $$x$$,

$$L(x) = \sum_{i \in A} \min\{x_i, 1 - x_i\} \leq \sum_{i \in A} x_i$$

$$= 2 \sum_{i \in V_1} x_i + 2 \sum_{i \in V_2} x_i \leq 2p.$$
In fact, $2p$ is the optimal value of the nice relaxation and it can be obtained in more than one way. One way is to let $x_i = r - \frac{p}{2}$ for $i \in V_1, x_i = 1 - r$ for $i \in V_2$, and $x_i = 0$ for $i \in V_3$. Then $P(x) = 2kr^2 + 4(1 - r)$. Now if $p/k$ tends to $0$ as $p$ tends to infinity (for example, set $k = p^2$), $P(x)/L(x)$ tends to $0$.

Note that the same can be done with $|V_3| > 2$ and then the above $x$ will be the unique optimal solution to the nice relaxation.

Thus straightforward application of the pipage rounding method does not provide a constant-factor approximation.

Moreover, Example 1 shows that the greedy algorithm (at each step add a vertex which increases most or decreases least the weight of the cut) also does not yield any constant-factor approximation. For this instance the greedy algorithm may first choose the vertices of $V_3$ and then no more arcs can be added and a solution with only four arcs will be the outcome (while the optimal one is to choose $p$ vertices from $V_1$, which gives a cut of size $2p$).

4. The structure of basic solutions. The following fact, which may be of interest beyond the purpose of this paper, is crucial in constructing a $0.5$-approximation for MAX DICUT with GAP in the general case.

**Theorem 4.1.** Let $(x, z)$ be a basic feasible solution to the linear relaxation (3.1)-(3.5). Then

$$x_i \in \{0, 0.5, 1/2, 1 - 0.5\} \text{ for each } i$$

for some $0 < \delta < 1/2$.

**Proof.** Let $(x, z)$ be a basic feasible solution. By definition, $(x, z)$ is the unique solution to the system of linear equations formed by those constraints in (3.2)-(3.5) which hold with equality. First, observe that for any $ij \in A$, either both $z_{ij} \leq x_i$ and $z_{ij} \leq 1 - x_j$ hold with equalities or exactly one holds with equality and the other with strict inequality. In the former case we exclude $z_{ij}$ by replacing the equations with the single equation $x_i + x_j = 1$. In the latter case we delete that equality from the linear system. In either case the variable $z_{ij}$ re-enters from the system while the number of equations reduces by one and, moreover, the value of $z_{ij}$ can be uniquely
determined by the values of \( x_i \) and \( x_j \). After performing this operation for each \( ij \in A \), we arrive at a system that can be written in the following form:

\[
\begin{align*}
02 & : y_i + y_j = 1 \text{ for } ij \in A' \subseteq A, \\
03 & : \sum_i y_i = p, \\
04 & : y_i = 0 \text{ for every } i \text{ such that } x_i = 0, \\
05 & : y_i = 1 \text{ for every } i \text{ such that } x_i = 1.
\end{align*}
\]

Since the removed \( z \) can be uniquely restored from the vector \( x \), \( x \) must be the unique solution to this system. Remove all components of \( A \) equal to either 0 or 1 or 1/2 and denote the set of the remaining indices by \( \Gamma \). Denote by \( A' \) the subvector of \( z \) consisting of the components with indices in \( \Gamma \). Note that the system is obtained from \((4.2)-(4.5)\) by substituting \( y_i = x_i \) for each \( i \) such that \( x_i \in \{0,1/2,1\} \).

Since \( x \) is the unique solution to \((4.2)-(4.5)\), \( x' \) is the unique solution to the resulting system, which can be written in the following form:

\[
\begin{align*}
06 & : y_i + y_j = 1 \text{ for } ij \in A' \subseteq A', \\
07 & : \sum_i y_i = p',
\end{align*}
\]

where \( p' \leq p \). It follows that one can choose \( |\Gamma| \) independent equations from \((4.6)-(4.7)\). We claim that any subsystem of this sort must contain \((4.7)\). Assume to the contrary that \(|\Gamma| < |\Gamma| \) equations from the set \((4.6)\) form an independent system. Consider the (undirected) subgraph \( \Gamma \) of \( G \) (we ignore orientations) corresponding to these equations. Note that \( |E(\Gamma)| = |\Gamma| \). Since \( x_i \neq 1/2 \) for all \( i \in \Gamma \), \( \Gamma \) does not contain odd cycles. Moreover, \( \Gamma \) cannot have even cycles as the subsystem corresponding to such a cycle is clearly dependent. Thus \( \Gamma \) is acyclic. But then \( |E(\Gamma)| \leq |\Gamma| - 1 \), a contradiction.

Now fix \( |\Gamma| \) independent equations from \((4.6)-(4.7)\). Then, by the above claim, we obtain the following system:

\[
\begin{align*}
08 & : y_i + y_j = 1 \text{ for } ij \in A', \\
09 & : \sum_{i \in \Gamma} y_i = p',
\end{align*}
\]

where \( A' \subseteq A' \). Since all the equations in \((4.8)-(4.9)\) are independent, \(|\Gamma| = |\Gamma| + 1 \).

We have proved above that the subgraph spanned by \( A' \) is acyclic, which together with \(|\Gamma| = |\Gamma| + 1 \) implies that it is a tree. Recall also that \( x' \) is the unique solution to \((4.8)-(4.9)\) and \( x'_i = x_i \neq 0, 1/2, 1 \) for all \( i \in \Gamma \). All these facts together with the structure of equalities \((4.8)\) imply that the components of \( x' \) with indices in \( \Gamma \) split into two sets—those equal to some \( 0 < \delta < 1/2 \) and those equal to \( 1 - \delta \).

5. Algorithm DICTU: Section 3 demonstrates that \( \text{DICTU} \) with \( \text{GRP} \) in the general setting does not admit a direct application of the piping rounding method. In this section we show that by using Theorem 4.1 and some tricks one is able to design not only a constant factor but even a 0.5-approximation for solving \( \text{DICTU} \) with \( \text{GRP} \). Moreover, the performance bound of 0.5 cannot be improved using different methods of rounding as the integrality gap of \((3.1)-(3.5)\) can be arbitrarily close to 1/2 (this can be shown exactly in the same way as it was done for \( \text{MAX CUT} \) with \( \text{GRP} \) in [AS99]).
For \( ij \in A \), call the number \( w_{ij} \min\{x_i, (1 - x_j)\} \) the contributed weight of the arc \( ij \). Observe that for any \( a, b \in [0,1], \ ab = \max\{a, b\} \min\{a, b\} \). It follows that

\[
F(x) = \sum_{ij \in A} w_{ij}F_{ij}(1 - x_j) = \sum_{ij \in A} w_{ij} \max\{x_i, 1 - x_j\} \min\{x_i, 1 - x_j\}
\]

(5.1)

\[
= \sum_{ij \in A} \max\{x_i, 1 - x_j\} \text{"the contributed weight of (i,"} \]

Algorithm DRCUT consists of two phases: the first phase is to find an optimal (fractional) basic solution to the linear relaxation (3.1)-(3.5); the second (rounding) phase is to transform this solution to a feasible (integral) solution. The rounding phase runs two different rounding algorithms based on the pipage rounding method and takes the best solution for the output. Let \((z, x)\) denote a basic optimal solution to (3.1)-(3.5) obtained at the first phase. Recall that, by Theorem 4.1, the vector \( z \) satisfies (4.1). Set \( V_1 = \{i : x_i = \delta\}, V_2 = \{i : x_i = 1 - \delta\}, V_3 = \{i : x_i = 1/2\}, V_4 = \{i : x_i = 0 \text{ or } 1\}\). Denote by \( l_{ij} (k, \ell = 1, 2, 3, 4) \) the sum of contributed weights over all arcs going from \( V_k \) to \( V_{\ell} \). Set \( l_0 = l_{12} + l_{24} + l_{32} + l_{23} + l_{13} + l_{21} + l_{31} + l_{42}, l_0 = \sum_{ij \in A} (l_{ij} + l_{ji}) \) (Figure 5.1 might be helpful to the reader).

![Diagram showing the process of rounding](image)

**Fig. 5.1.** \( l_1 \) and \( l_2 \) are the sums of contributed weights over the displayed collections of arcs.

The second phase of the algorithm successively calls two rounding algorithms—ROUND1 and ROUND2—and takes a solution with maximum weight for the output.

ROUND1 is the pipage rounding applied to the optimal basic solution \( x \). Let \( \delta \) denote the output of ROUND1. Then by the description of pipage rounding, \( F(\delta) \geq F(x) \). Since the number \( \max\{x_i, 1 - x_j\} \) remains unchanged for any \( ij \in A \) such that \( i \in V_k \) and \( j \in V_{\ell} \), and can be easily computed, (5.1) implies that

\[
F(\delta) \geq F(x) \geq \delta l_{12} + (1 - \delta) l_0 + l_{13} + l_2.
\]

(5.2)

Algorithm ROUND2 is the pipage rounding applied to a different fractional solution \( x' \) which is obtained by an alteration of \( x \).

**Algorithm ROUND2.** Define a new vector \( x' \) by the following formulae:

\[
x'_i = \begin{cases} 
\min\{\delta, (1 - \delta) |V_k|/|V_i|\} & \text{if } i \in V_k, \\
\max\{0, (1 - \delta) - (1 - \delta) |V_k|/|V_i|\} & \text{if } i \in V_{\ell}, \\
x_i & \text{if } i \in V \setminus (V_k \cup V_{\ell}).
\end{cases}
\]

(5.3)
Apply the pipage rounding to \( z' \).

Analysis. The vector \( z' \) is obtained from \( z \) by redistributing uniformly the values from the components in \( V_2 \) to those in \( V_1 \) and keeping the same the remaining ones. It follows from the description of ROUNDD2 that \( z' \) is feasible. Applying the pipage rounding to \( z' \) results in an integral feasible vector of weight at least \( F(z') \). We claim that \( F(z') \geq l_2 + l_1/2 \). Consider first the case when \( |V_1| \geq |V_2| \). Then by (5.3), \( x_i = 0 \) for all \( i \in V_2 \) and \( x_i' \geq \delta \) for all \( i \in V_1 \). Therefore, by (5.1) and definitions of \( V_2 \) it follows that \( F(z') \geq l_2 + l_1/2 \). Now assume that \( |V_2| \leq |V_1| \). Then by (5.3), \( x_i = 1 \) and \( x_i' \leq 1 - \delta \) for all \( i \in V_1 \). Hence, again by (5.1), \( F(z') \geq l_2 + l_1/2 \). Therefore, in either case \( F(z') \geq l_2 + l_1/2 \). Thus ROUNDD2 outputs a solution of weight at least \( l_2 + l_1/2 \). Together with (5.2) this implies that the output of DICUT has weight at least

\[
\max(l_2 + l_1/2, \delta l_2 + (1 - \delta)l_0 + l_1/2 + l_2),
\]

which is bounded from below by

\[
q = \max(l_2, \delta l_2 + (1 - \delta)l_0 + l_1/2),
\]

where \( \delta = l_0 + l_2 \). Now recall that \( \delta < \delta < 1/2 \). Hence, if \( l_2 \geq l_1 \), then \( q = l_2 + l_1/2 \geq (l_2 + l_1 + l_2)/2 \) and if \( l_2 < l_1 \), then \( q = \delta l_2 + (1 - \delta)l_0 + l_1/2 > (l_2 + l_1)/2 + l_1/2 \). Thus, in either case the algorithm DICUT outputs a solution of weight at least \( l_2 + l_0 + l_1 + l_2/2 \), which is at least half of the optimum.

6. Directly tractable special cases. In the final section we consider two special cases of MAX DICUT with GSP which admit direct application of the pipage rounding method.

6.1. The circulation case. We first consider the case when the weight function \( w \) is a circulation in the given graph. This means that the function \( w \) obeys the condition

\[
\sum_{j \in A} w_j = \sum_{k \in A} w_k \text{ for each vertex } i \in V. \]

We will show that the circulation case of MAX DICUT with GSP admits a 0.5-approximation algorithm which is a straightforward implementation of the scheme described in section 2. In the next subsection we will show that it also finds a cut of weight within a factor of 0.5 of the optimum in the case when the cuts are constrained to have equal parts (the DISJOINT DIRECTED PROBLEM).

Note first that for any \( a \) and \( b \) between 0 and 1,

\[
2a(1 - b) = a(1 - b) + (1 - a) + a - b, \quad 2\min(a, 1 - b) = \min(a + b, 2 - a - b) + a - b.
\]

Using these identities we can rearrange the functions \( F \) and \( L \) in the following way:

\[
F(x) = 1/2 \sum_{j \in A} w_j \left[ x_j (x_j - x_k) + x_k (1 - x_k) + x_k - x_j \right],
\]

\[
L(x) = 1/2 \sum_{j \in A} w_j \left[ \min(x_j + x_k, 2 - x_i - x_j) + x_i - x_j \right].
\]
Since
\[ \sum_{i \in A} w_i (x_i - x_j) = \sum_{i \in V} q_i x_i, \]
where \( q_i \) is the difference between the sum of weights of arcs leaving the node \( i \) and the sum of weights of arcs entering the node \( i \), both functions can be expressed as the sums of two surrogates

\[ F(x) = \frac{1}{2} \sum_{i \in A} w_i \left[ x_i (1 - x_j) + x_j (1 - x_i) \right] + \frac{1}{2} \sum_{i \in V} q_i x_i; \]

\[ L(x) = \frac{1}{2} \sum_{i \in A} w_i \left[ \min\{x_i + x_j, 2 - x_i - x_j\} \right] + \frac{1}{2} \sum_{i \in V} q_i x_i. \]

Aseev and Syrdenko [AS99] proved that for \( 0 \leq x_i, x_j \leq 1 \),

\[ x_i (1 - x_j) + x_j (1 - x_i) \leq \min\{x_i + x_j, 2 - x_i - x_j\} \leq \frac{1}{2}. \]

It follows that \( F(x)/L(x) \geq 1/2 \) if \( \sum_{i \in V} q_i x_i \geq 0 \). In the case when the weights \( w \) form a circulation in \( G \), each \( q_i \) is equal to zero. This proves the bound of 1/2 for this case.

6.2. Digraph Direction. The digraph direction problem is the special case of Max DICUT with GIP where \( n = 2p \). Let \( (x, y) \) be an optimal solution to the linear relaxation (3.1)–(3.5). We claim that in this case \( \sum_{i \in V} q_i x_i \geq 0 \), which means by (6.1), (6.2), and (6.3) that the algorithm presented for the circulation case also has an approximation ratio of 1/2 for digraph direction. Indeed, assume to the contrary that \( \sum_{i \in V} q_i x_i < 0 \). Since \( n = 2p \), the vector \( y \) defined by

\[ y_i = 1 - x_i \quad \text{for all } i \in V \]

is also feasible for the nice relaxation (3.7), (3.4), (3.5). Moreover,

\[ \sum_{i \in V} q_i x_i = - \sum_{i \in V} q_i y_i \]

and

\[ \min\{x_i + x_j, 2 - x_i - x_j\} = \min\{y_i + y_j, 2 - y_i - y_j\} \]

for every \( i \in A \). This means that \( L(y) > L(x) \), which contradicts the optimality of \( x \).

REFERENCES

