Semidefinite Programming Based Algorithms for the Ratio Cut Problem

Luis A. A. Meira *Flávio K. Miyazawa*

Technical Report - IC-05-032 - Relatório Técnico

December - 2005 - Dezembro

Semidefinite Programming Based Algorithms for the Ratio Cut Problem[*]

Luis A. A. Meira   Flávio K. Miyazawa

Abstract

In this paper we analyze a known relaxation for the Ratio Cut Problem based on positive semidefinite constraints and present a branch and bound algorithm and heuristics based on this relaxation. The relaxation and the algorithms were tested on small and moderate sized instances. The relaxation leads to values very close to the optimum solution values. The exact algorithm could obtain solutions for small and moderate sized instances and the best heuristics obtained optimum or almost optimum solutions for all tested instances. We prove interesting characteristics for each one of these heuristics. We also compared the semidefinite based branch and bound algorithm with a commercial integer programming solver.

1 Introduction

Since the work of Goemans and Williamson [10], presenting an approximation algorithm for the Max-Cut Problem, the use of semidefinite programming has increased and it turns out to be one of the main tools to obtain good relaxations and algorithms for combinatorial optimization problems [11, 10, 21].

In this paper we analyze the relaxation of a Ratio Cut Problem formulation presented by Arora et al. [2].

The Ratio Cut problem is NP-hard and has applications in image segmentation [19], metric labeling problem[3] and a natural application in graph conductance.

We propose four heuristics and an exact branch and bound algorithm for the Ratio Cut Problem, based on this relaxation. We couldn't prove an approximation factor to the heuristics, but we proved good characteristics for each one. These algorithms were tested on a set of random instances and, for all tests, they produced optimal or almost optimal solutions. We also compared the semidefinite based branch and bound algorithm with a known integer programming formulation [6] solved with the Xpress-MP solver [17].

We proved the existence of a bounded solution for the Ratio Cut Problem when the expectation of both parts of the ratio are bounded. This strategy can also be applied for other problems with a ratio in the objective function. Because it is not known a way to decompose the expectation of a ratio in terms of their parts, this result has a general utility.

[*]This research was partially supported by ProNEx–FAPESP/CNPq Proc. 2003/09925-5, and CNPq (Proc. 306526/2004-2, 470608/01–3, 478818/03–3 and 490333/04-4).

The first approximation algorithm for the Ratio Cut problem has an $O(\log n)$ factor due to Leighton and Rao [15] in 1988. Recently, the interest in the Ratio Cut Problem has increased. Freivalds [6] presents a non-polynomial time 2-approximation algorithm, with empirical running time in $O(n^{1.6})$ to sparses graphs. When all edge costs are in $\{0, 1\}$ and vertices weights are 1, Arora et al. [2] present an $O(\sqrt{\log n})$-approximation algorithm based on a semidefinite program. For the general case, Chawla et al. [4] present an $O(\log^{3/4} n)$-approximation algorithm, while the current best known algorithm is an $O(\sqrt{\log n} \log \log n)$-approximation algorithm due to Arora, Lee and Naor [1].

2 Problem Definition

Given a graph $G = (V, E)$, a cost function, $c : E \to \mathbb{R}^+$, a weight function, $w : V \to \mathbb{R}^+$, and a set $S \subset V$, $S \neq \emptyset$, we denote by $\delta(S)$ the set of edges with exactly one extremity in S, $\mathcal{C}(S)$ the sum $\sum_{e \in \delta(S)} c_e$, $\mathcal{W}(S)$ the product $w(S)w(V \setminus S)$, where $w(C) := \sum_{v \in C} w_v$, and $\rho(S)$ the value $\mathcal{C}(S)/\mathcal{W}(S)$. The Ratio Cut Problem can be defined as follows:

MIN RATIO CUT PROBLEM: Given a graph $G = (V, E)$, costs $c_e \in \mathbb{R}^+$ for each edge $e \in E$, and weights $w_v \in \mathbb{R}^+$ for each vertex $v \in V$, find a cut $S \subset V, S \neq \emptyset$, that minimizes $\rho(S)$. See Figure 1.

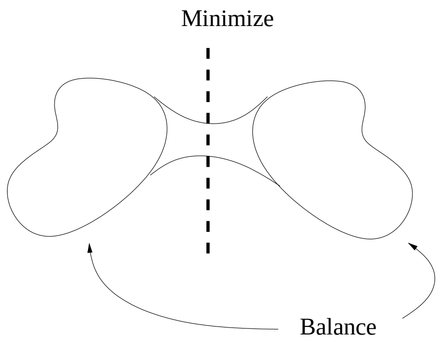

Figure 1: The objective of Ratio Cut Problem is to find a small balanced cut.

In the unweighted version, we have $c_e = 1$ for each $e \in E$, and $w_v = 1$ for each v in V.

The Ratio Cut problem is also referred to as the Sparsest Cut problem and Min Flux Cut problem [5]. Another problem that is strongly related to the Ratio Cut problem is the Quotient Cut Problem, which can be defined as follows:

MIN QUOTIENT CUT PROBLEM: Given a graph $G = (V, E)$, costs $c_e \in \mathbb{R}^+$ for each edge $e \in E$, and weights $w_v \in \mathbb{R}^+$ for each vertex $v \in V$, find a cut $S \subset V, S \neq \emptyset$, that minimizes $\mathcal{C}(S)/\min\{w(C), w(V \setminus C)\}$.

In [2], the Min Quotient Cut is also referred to as Sparsest Cut. In terms of approximability, an α-approximation algorithm for the Ratio Cut problem is a 2α-approximation algorithm for the Quotient Cut problem and *vice versa*.

The following lemma can be deduced from the work of Leigthon and Rao [15]. Because the proof is not straightforward, we present it here.

Lemma 2.1 *An α-approximation algorithm for the Ratio Cut problem is a 2α-approximation algorithm for the Quotient Cut problem, and* vice versa*.*

Proof.

Let $\mathcal{A}_1$ be an α-approximation algorithm and I an instance for the Ratio Cut problem. Let S be the cut produced by the algorithm $\mathcal{A}_1(I)$. Without loss of generality, we suppose that $w(S) \leq w(V)/2$ and $w(V \setminus S) \geq w(V)/2$. Because $\mathcal{A}_1$ is an α-approximation algorithm, we have

$$\frac{\mathcal{C}(S)}{w(S)w(V \setminus S)} \leq \alpha \frac{\mathcal{C}(S')}{w(S')w(V \setminus S')}, \quad \forall S' \subset V, S' \neq \emptyset.$$

That is,

$$\begin{aligned}
\frac{\mathcal{C}(S)}{w(S)} &\leq \alpha \frac{\mathcal{C}(S')w(V \setminus S)}{w(S')w(V \setminus S')}, \quad \forall S' \subset V, S' \neq \emptyset \\
&\leq \alpha \frac{\mathcal{C}(S')}{\min\{w(S'), w(V \setminus S')\}} \frac{w(V)}{\max\{w(S'), w(V \setminus S')\}}, \quad \forall S' \subset V, S' \neq \emptyset \\
&\leq 2\alpha \frac{\mathcal{C}(S')}{\min\{w(S'), w(V \setminus S')\}}, \quad \forall S' \subset V, S' \neq \emptyset, \qquad (1)
\end{aligned}$$

where inequality (1) is valid because $w(V) \leq 2\max\{w(S'), w(V \setminus S')\}$.

Now, suppose that $\mathcal{A}_2$ is an α-approximation algorithm and I an instance for the Quotient Cut problem. Let S be the cut produced by $\mathcal{A}_2(I)$. Without loss of generality, we suppose that $w(S) \leq w(V)/2$ and $w(V \setminus S) \geq w(V)/2$. Thus

$$\frac{\mathcal{C}(S)}{w(S)} \leq \alpha \frac{\mathcal{C}(S')}{\min\{w(S'), w(V \setminus S')\}}, \quad \forall S' \subset V, S' \neq \emptyset.$$

i.e.

$$\begin{aligned}
\frac{\mathcal{C}(S)}{w(S)w(V \setminus S)} &\leq \alpha \frac{\mathcal{C}(S')}{\min\{w(S'), w(V \setminus S')\}\, w(V \setminus S)}, \quad \forall S' \subset V, S' \neq \emptyset \\
&\leq \alpha \frac{\mathcal{C}(S')}{w(S')w(V \setminus S')} \frac{\max\{w(S'), w(V \setminus S')\}}{w(V \setminus S)}, \quad \forall S' \subset V, S' \neq \emptyset \qquad (2) \\
&\leq 2\alpha \frac{\mathcal{C}(S')}{w(S')w(V \setminus S')}, \quad \forall S' \subset V, S' \neq \emptyset. \qquad (3)
\end{aligned}$$

$\square$

Inequality (2) is obtained multiplying the previous inequality by $\frac{\max\{w(S'),w(V\setminus S')\}}{\max\{w(S'),w(V\setminus S')\}}$. Inequality (3) is valid because $\max\{w(S'), w(V \setminus S')\} \leq 2w(V \setminus S)$.

3 Semidefinite Programming Formulation and Relaxation

Given an instance (G, c, w) for the Ratio Cut Problem, we can formulate it as follows:

$$\begin{array}{rcll} \rho_F = \min \sum_{i<j} |v_i - v_j|^2 c_{ij} & & & \\ \text{s.t.} & & & \\ |v_i - v_j|^2 & \leq & |v_i - v_k|^2 + |v_k - v_j|^2, \quad \forall i,j,k \in V, & \\ \sum_{i<j} |v_i - v_j|^2 w_i w_j & = & 1, & (F) \\ v_i^2 & = & r^2, \quad \forall i \in V, & \\ r & \geq & 0 & \end{array}$$

For each vertex $i \in V$, v_i have a value r or $-r$. This fact allow us to scale v_i in a way that the denominator of the ratio is 1 without change the objective function. Note that $|v_i - v_j|^2$ is non zero if and only if edge (i, j) is considered in the ratio.

Lemma 3.1 *F is a* formulation *for the Ratio Cut Problem.*

Proof.

Let I be an instance for the Ratio Cut Problem, $\mathcal{O}^* \subset V$ be an optimum solution to I and ρ_F the value of the objective function of F solved with instance I.

We first prove that $\rho_F \leq \rho(\mathcal{O}^*)$ and then $\rho_F \geq \rho(\mathcal{O}^*)$.

Given a cut S, let $r = \frac{1}{\sqrt{4|S||V \setminus S|}}$, $v_i = r$ for each $i \in S$ and $v_i = -r$ for each $i \in V \setminus S$. The obtained attribution is feasible to F and has cost $\rho(S)$ for unweighted graphs. This is valid for any set S, including $\mathcal{O}^*$, so $\rho_F \leq \rho(\mathcal{O}^*)$. In the weighted version, the same conclusion is valid for $r = \frac{1}{\sqrt{4\mathcal{W}(S)\mathcal{W}(V \setminus S)}}$.

To conclude that F is a formulation it is sufficient to show that any solution has a associated cut with the same cost. Let $\mathcal{V} = (v, r)$ be a solution to F. Consider the associated cut $S_\mathcal{V} = \{i : v_i = -r\}$. As $\mathcal{V}$ is feasible, $r = \frac{1}{\sqrt{4|S_\mathcal{V}||V \setminus S_\mathcal{V}|}}$ (in the weighted version $r = \frac{1}{\sqrt{4\mathcal{W}(S_\mathcal{V})\mathcal{W}(V \setminus S_\mathcal{V})}}$) and $\rho_F(\mathcal{V})$ values $\rho(S_\mathcal{V})$. $\square$

For the rest of the paper, we consider $n = |V|$.

If we relax v_i to an n-dimensional vector in $\mathbb{R}^n$, we have a relaxation, named *A1*. To write the relaxation *A1* as a semidefinite program, we observe that $|v_i - v_j|^2 = (v_i - v_j)^2 = v_i^2 - 2v_iv_j + v_j^2$ and $x_{ij} = v_iv_j$. For more details on semidefinite programming see [14, 18, 12, 11]. In the following we present the (weighted) relaxation *A1* and its corresponding semidefinite program.

$$\begin{array}{l}
\textit{A1} \\
\min \sum_{i<j} |v_i - v_j|^2 c_{ij} \\
\text{s.t.} \\
|v_i - v_j|^2 \leq |v_i - v_k|^2 + |v_k - v_j|^2 \ \ \forall i,j,k \in V, \\
\sum_{i<j} |v_i - v_j|^2 w_i w_j \ = \ 1, \\
v_i \ \in \ \mathbb{R}^n.
\end{array}$$

$$\begin{array}{l}
\textit{A1} \text{ in SDP} \\
\min \sum_{i<j} (x_{ii} + x_{jj} - 2x_{ij}) c_{ij} \\
\text{s.t.} \\
x_{ij} + x_{jk} - x_{ik} \leq x_{jj} \ \ \forall i,j,k \in V, \\
\sum_{i<j} (x_{ii} + x_{jj} - 2x_{ij}) w_i w_j \ = \ 1, \\
X \ \succeq \ 0.
\end{array}$$

For unweighted graphs, Arora et al. [2] prove an integrality gap of $O(\sqrt{\log n})$ for the relaxation $A1$. The largest gap known up to now for this relaxation is 10/9.

The relaxation $A1$ can be made tighter adding the constraint $|v_i| = |v_j|$, for each $i, j \in V$. In semidefinite programming, these constraints are given by $x_{ii} = x_{jj}$, for each $i, j \in V$. We denote by $A1^+$ the relaxation $A1$ with these new constraints. Observe that $A1^+$ is still a relaxation to F.

Remark: Note that $A1$ has only relative distances between vectors in the objective function and all constraints. This implies that the objective function value does not change when the vectors v_j, for each $j \in V$, get the same translation. On the other side, in the relaxation $A1^+$ there may exist translations that do not change the objective function. See Figure 2.

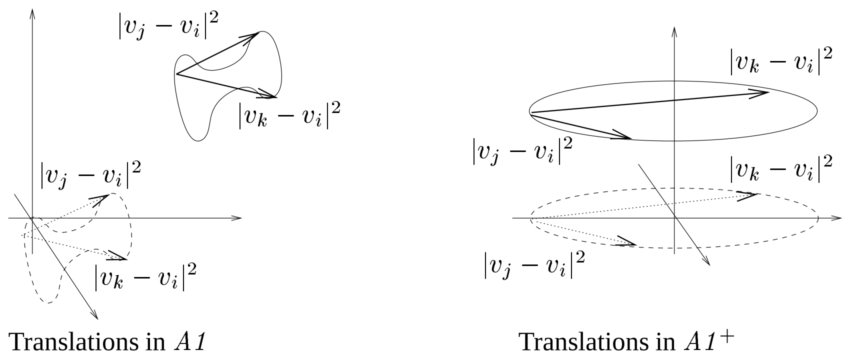

Figure 2: Translations in formulations $A1$ and $A1^+$.

Definition 3.2 *An optimal solution X^* of $A1$ or $A1^+$ is said to be* small *if it is optimal and minimizes $\sum_{i \in V} |v_i|$.*

To convert a non-small solution of $A1$ to a small one with the same objective value, it is sufficient to translate its barycenter to the origin. In the relaxation $A1^+$ it is sufficient to translate the hyperplane of small dimension that contains all vectors of the solution to the origin.

Given an instance I for the Ratio Cut Problem, we denote by $A1(I)$ and $A1^+(I)$ the relaxations $A1$ and $A1^+$ defined with the corresponding values of I.

4 Heuristics for the Ratio Cut Problem

In this section we propose some heuristics for the Ratio Cut problem. These heuristics receive as an input parameter a feasible small solution $X^* = (x^*_{ij})$ to relaxation $A1(G, c, w)$ or $A1^+(G, c, w)$ and then apply some rounding strategy to obtain a feasible solution.

In the following we present the heuristics *H1*, *H2*, *H3* and *H4*. The heuristics *H1* and *H2* use a value $\xi(X^*, i, j)$ defined as $\xi(X^*, i, j) = \frac{x^*_{ij}}{\sqrt{x^*_{ii}x^*_{jj}}} = \cos(v_i, v_j)$. Note that a smaller value for $\xi(X^*, i, j)$ means that vectors v_i and v_j are more separated.

All the heuristics can achieve an empty cut.

```
HEURISTIC $H1(G = (V, E), c, w, X^*)$
 1.  Let $s, t \in V$ where $\xi(X^*, s, t)$ is minimum.
 2.  For each $u \in V$ do
 3.      select a random number $\alpha$ uniformly distributed in $[-1, 1]$
 4.      if $\xi(X^*, s, u) \geq \alpha$ then
 5.          $S \leftarrow S \cup \{u\}$.
 6.  Return($S$).
```

```
HEURISTIC $H2(G = (V, E), c, w, X^*)$
 1.  Let $s, t \in V$ where $\xi(X^*, s, t)$ is minimum.
 2.  Select a random number $\alpha$ uniformly distributed in $[-1, 1]$.
 3.  For each $u \in V$ do
 4.      if $\xi(X^*, s, u) \geq \alpha$
 5.          $S \leftarrow S \cup \{u\}$.
 6.  Return($S$).
```

```
HEURISTIC $H3(G = (V, E), c, w, X^*)$
 1.  Let $\mathcal{V} = (v_i)$ be the Cholesky decomposition of $X^*$ ($X^* = \mathcal{V}^T\mathcal{V}$).
 2.  Let $S \leftarrow \emptyset$
 3.  Select a random vector $r \in \mathbb{R}^n$ uniformly distributed in the unit sphere.
 4.  For each $u \in V$ do
 5.      if $v_u \cdot r \geq 0$
 6.          $S \leftarrow S \cup \{u\}$.
 7.  Return($S$).
```

```
HEURISTIC $H4(G = (V, E), c, w, X^*)$
 1.  Let $\mathcal{V} = (v_i)$ be the Cholesky decomposition of $X^*$ ($X^* = \mathcal{V}^T\mathcal{V}$).
 2.  Let $A \leftarrow \emptyset$ and $B \leftarrow \emptyset$.
 3.  Select two vectors $r_1, r_2 \in \mathbb{R}^n$ uniformly distributed in the unit sphere.
 4.  For each $u \in V$ do
 5.      if $v_u \cdot r_1 >= 0$ and $v_u \cdot r_2 >= 0$ then $A \leftarrow A \cup \{u\}$
 6.      if $v_u \cdot r_1 < 0$ and $v_u \cdot r_2 < 0$ then $B \leftarrow B \cup \{u\}$
 7.  Let $G'$ obtained from $G$ contracting $A$ and $B$ to vertices $a$ and $b$, resp.
 8.  Let $S'$ be a cut with minimum cost separating $a$ and $b$ in $G'$
 9.  Let $S$ be the set of vertices in $G$ corresponding to $S'$.
10.  Return($S$).
```

4.1 Some Analysis of the Heuristics

It is not hard to show the following lemma:

Lemma 4.1 *Let $\mathcal{P}$ be a minimization problem with domain $\mathcal{D}$ and objective function $\rho_{\mathcal{P}}(S) = \mathcal{C}_{\mathcal{P}}(S)/\mathcal{W}_{\mathcal{P}}(S), \quad \forall S \in \mathcal{D}$, where $\mathcal{C}_{\mathcal{P}}(S) > 0$ and $\mathcal{W}_{\mathcal{P}}(S) > 0$ for each $S \in \mathcal{D}$. Let $\mathcal{A}$ be a randomized polynomial time algorithm for $\mathcal{P}$, I an instance and S a solution produced by $\mathcal{A}(I)$. If S^{opt} is an optimum solution for I and α_1, α_2, $\mathcal{C}^*$ and $\mathcal{W}^*$ are positive numbers such that $\mathcal{C}^*/\mathcal{W}^* \leq \rho_{\mathcal{P}}(S^{opt})$, $\alpha_1/\alpha_2 > 1$, the expectation $E[\mathcal{C}_{\mathcal{P}}(S)] \leq \alpha_1\mathcal{C}^*$ and $E[\mathcal{W}_P(S)] \geq \alpha_2\mathcal{W}^*$, then there exists a solution S' with $\rho_{\mathcal{P}}(S') \leq \frac{\alpha_1}{\alpha_2}\mathcal{C}^*/\mathcal{W}^*$.*

Proof.
Let $\rho^* = \mathcal{C}^*/\mathcal{W}^*$ and $\beta = \alpha_1/\alpha_2$. Because $E[\mathcal{C}(S)] \leq \alpha_1\mathcal{C}^*$ and $E[\mathcal{W}(S)] \geq \alpha_2\mathcal{W}^*$, we have

$$\frac{E[\mathcal{C}(S)]}{E[\mathcal{W}(S)]} \leq \rho^*\beta \Rightarrow E[\mathcal{C}(S)] - \rho^*\beta E[\mathcal{W}(S)] \leq 0. \tag{4}$$

By the linearity of the expectation, we have

$$E\left[\mathcal{C}(S) - \rho^*\beta\mathcal{W}(S)\right] \leq 0.$$

This proves the existence of at least one solution S^{gap} such that

$$\mathcal{C}(S^{gap}) - \rho^*\beta\mathcal{W}(S^{gap}) \leq 0.$$

That is, $\rho(S^{gap}) \leq \beta\rho^*$. □

Lemma 4.2 *If the hypothesis of Lemma 4.1 are valid for the Unweighted Ratio Cut Problem, with $\mathcal{C}$ as $\mathcal{C}_{\mathcal{P}}$, $\mathcal{W}$ as $\mathcal{W}_{\mathcal{P}}$, then it is also possible to find a solution S' with $\rho(S') \leq (1+\epsilon)\frac{\alpha_1}{\alpha_2}\mathcal{C}^*/\mathcal{W}^*$ in polinomial time with high probability, for any $\epsilon > 0$.*

Given a cut S for the Ratio Cut Problem, denote by $Z_S = \mathcal{C}(S) - \rho^*\beta\mathcal{W}(S)$.
We also need the following fact.

Fact 4.3 $\Pr[Z_S \geq \rho^*\dfrac{n}{b}] \leq p$ *for any* $p \in (0,1)$ *and* $b = \frac{p}{(\beta-1)(1-p)n}$.

Proof. (Fact 4.3)
The expectation $E[Z_S]$ is equal to $E[\mathcal{C}(S)] - \rho^*\beta E[\mathcal{W}(S)]$. Assuming the hypothesis of the lemma we have $E[Z_S] \leq 0$. We also have $n-1 \leq \mathcal{W}(S) \leq n^2$.
Because ρ^* is a lower for any optimal solution, we have $\frac{\mathcal{C}(S)}{\mathcal{W}(S)} \geq \rho^*$, for any set $S \subset V, S \neq \emptyset$. This implies $\mathcal{C}(S) - \rho^*\mathcal{W}(S) \geq 0$, for any set $S \subset V, S \neq \emptyset$.
Therefore,

$$\mathcal{C}(S) - \beta\rho^*\mathcal{W}(S) \geq -\left(\beta-1\right)\rho^*\mathcal{W}(S) \geq -\left(\beta-1\right)\rho^*n^2, \text{ for any } S \subset V, S \neq \emptyset.$$

From the above, we conclude that $Z_S \geq -\left(\beta-1\right)\rho^*n^2$.
Suppose that Fact 4.3 is false, that is, $\Pr[Z_S \geq \rho^*\frac{n}{b}] > p$. In this case, we obtain that

$$\begin{array}{rcl} E[Z_S] & \geq & -\left(\beta-1\right)\rho^*n^2\Pr[Z_s < \rho^*\frac{n}{b}] + \rho^*\frac{n}{b}\Pr[Z_s \geq \rho^*\frac{n}{b}]) \\ & > & -\left(\beta-1\right)\rho^*n^2(1-p) + \rho^*\frac{n}{b}p \\ & = & 0. \end{array}$$

Therefore, we obtain a contradiction that validates the lemma. □

Proof. (Lemma 4.2)

Consider a pair (p_1, b_1) that respect Fact 4.3 such as $b_1 = 2/\epsilon$ for some $\epsilon > 0$. With probability greater than $1 - p_1$, we have that

$$\mathcal{C}(S) - \beta\rho^*\mathcal{W}(S) \leq \rho^* \frac{n}{b_1}.$$

Because $n - 1 \leq \mathcal{W}(S)$ and $n \geq 2$ we have $\frac{n}{b_1} \leq 2\frac{\mathcal{W}(S)}{b_1}$. That is, with probability greater than $1 - p_1$ we have

$$\mathcal{C}(S) - \beta\rho^*\mathcal{W}(S) \leq 2\frac{\rho^*\mathcal{W}(S)}{b_1}.$$

Thus,

$$\rho(S) \leq \left(\beta + \frac{2}{b_1}\right)\rho^* = (\beta + \epsilon)\,\rho^*. \tag{5}$$

We have $p_1 = \dfrac{1}{1 + \frac{\frac{\epsilon}{2(\beta-1)}}{n}}$. If the algorithm is applied n times, and a solution S' with minimum value is returned, the probability that inequality 5 is not satisfied for S' is less than or equal to $(p_1)^n \leq \mathbf{e}^{-\frac{\epsilon}{2(\beta-1)}}$, where $\mathbf{e} \simeq 2.71828$. For β and ϵ constants, the probability to obtain the desired cut is greater than a constant value. An algorithm that finds the desired cut with high probability can be achieved for this result.

□

For the remaining of this section, we denote by $X^* = (x^*_{ij})$ a small optimum solution for $A1^+(G, c, w)$ and s and t two vertices were $\xi(X^*, s, t)$ is minimum. Given a solution X^* for relaxation $A1^+$, we denote by Y^* the solution X^* scaled by a constant factor k such that $Y^* = kX^*$ and $Y^*_{ii} = 1$ for each vertex i.

Let

$$\mathcal{C}(X) = \frac{1}{4}\sum_{i<j}(x_{ii} + x_{jj} - 2x_{ij})c_{ij} = \frac{1}{4}\sum_{i<j}|v_i - v_j|^2 c_{ij},$$

$$\mathcal{W}(X) = \frac{1}{4}\sum_{i<j}(x_{ii} + x_{jj} - 2x_{ij})w_i w_j = \frac{1}{4}\sum_{i<j}|v_i - v_j|^2 w_i w_j,$$

and $\rho(X) = \mathcal{C}(X)/\mathcal{W}(X)$, where $X = \mathcal{V}^T\mathcal{V}$.

Observe that $\mathcal{C}(Y^*) = \sum_{i<j}\frac{1-y^*_{ij}}{2}c_{ij}$, $\mathcal{W}(Y^*) = \sum_{i<j}\frac{1-y^*_{ij}}{2}w_i w_j$ and $y^*_{ij} = \cos(v_i, v_j)$.

Lemma 4.4 *The value $\rho(Y^*)$ is a lower bound for $\rho(S)$, for any set $S \subseteq V$, $S \neq \emptyset$*

Proof.

Let X^* be an optimum solution to $A1^+$. As $A1^+$ is a relaxation of formulation F, $\rho(X^*)$ is a lower bound to the value of any optimum solution for the Ratio Cut. We have $\mathcal{C}(Y^*) = k\mathcal{C}(X^*)$, and $\mathcal{W}(Y^*) = k\mathcal{W}(X^*)$. Therefore, $\rho(Y^*) = \rho(X^*)$. □

We first consider that $\xi(X^*, s, t) = -1$. In this case, all heuristics generate non-empty cuts. We further present some ways to keep the same results when this condition is not true.

If each one of the four heuristics receives X^* as an input parameter, then they partially respect Lemma 4.1, but we couldn't obtain an approximation algorithm. In what follows, we show the strength and the weakness of each heuristic.

First consider heuristics $H1$ and $H2$. These heuristics are, in some way, symmetric. While the heuristic $H1$ guarantees that the expectation of $\mathcal{W}(S)$ is lower bounded by a constant factor of $\mathcal{W}(Y^*)$, see Lemma 4.5, the heuristic $H2$ guarantees that the expectation of $\mathcal{C}(S)$ is upper bounded by a constant factor of $\mathcal{C}(Y^*)$, see Lemma 4.6.

It is also possible to obtain X^* for which Lemma 4.1 does not hold for heuristics $H1$ and $H2$.

The following lemma shows that heuristic $H1$ produces a cut where $E[\mathcal{W}(S)]$ is at least a factor of $\mathcal{W}(Y^*)$.

Lemma 4.5 *If S is the solution produced by heuristic $H1$ and $\xi(X^*, s, t) = -1$, then $E[\mathcal{W}(S)] \geq \frac{\mathcal{W}(Y^*)}{2}$.*

Proof.

Consider two vertices u and v in $V \setminus \{s\}$.

The probability that edge us is in $\delta(S)$ is $\dfrac{1-\xi(X^*, u, s)}{2} = \dfrac{1-\cos(v_u, v_s)}{2} = \dfrac{1-y^*_{us}}{2}$.

The probability that edge uv is in $\delta(S)$ is the probability that $(s, u) \in \delta(S)$ and $(s, v) \notin \delta(S)$ plus the probability that $(s, u) \notin \delta(S)$ and $(s, v) \in \delta(S)$. That is,

$$\begin{aligned}
\Pr[(u,v) \in \delta(S)] &= \frac{1-y^*_{us}}{2}\left(1 - \frac{1-y^*_{vs}}{2}\right) + \left(1 - \frac{1-y^*_{us}}{2}\right)\frac{1-y^*_{vs}}{2} = \frac{1-y^*_{us}y^*_{vs}}{2} \\
&\geq \frac{1}{2}\min\{\frac{1-y^*_{us}}{2} + \frac{1-y^*_{vs}}{2}, \frac{1+y^*_{us}}{2} + \frac{1+y^*_{vs}}{2}\} && (6) \\
&\geq \frac{1}{2}\left(\frac{1-y^*_{uv}}{2}\right). && (7)
\end{aligned}$$

Inequality (6) can be verified. See Figure 3. Inequality (7) is valid from the triangle inequality constraints of $A1^+$. More precisely

$$\begin{gathered}
\frac{1-y^*_{uv}}{2} \leq \frac{1-y^*_{us}}{2} + \frac{1-y^*_{vs}}{2} \quad \text{and} \\
\frac{1-y^*_{uv}}{2} \leq \frac{1-y^*_{ut}}{2} + \frac{1-y^*_{vt}}{2} = \frac{1+y^*_{us}}{2} + \frac{1+y^*_{vs}}{2}. \qquad (8)
\end{gathered}$$

In inequality (8) we use the fact that $\xi(X^*, s, t) = -1$. In this case, the angle between s and t is π, and we have $\cos(v_{us}) = -\cos(v_{ut})$.

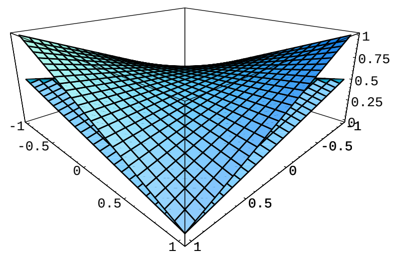

Figure 3: $\frac{1}{2}\min\{\frac{1-x_{us}}{2}+\frac{1-x_{vs}}{2},\frac{1+x_{us}}{2}+\frac{1+x_{vs}}{2}\}$ versus $\frac{1-x_{us}x_{vs}}{2}, \forall x_{us}, x_{vs} \in [-1,1]$.

The expectation of $\mathcal{W}(S)$ is

$$\begin{aligned}
E[\mathcal{W}(S)] &= \sum_{u<v} w_u w_v \Pr[uv \in \delta(S)] \\
&= \sum_{u\in V\setminus\{s\}} w_u w_s \Pr[us \in \delta(S)] + \sum_{u\in V\setminus\{s\}} \sum_{v\in V\setminus\{s\} \mid v<u} w_u w_v \Pr[uv \in \delta(S)] \\
&\geq \sum_{u\in V\setminus\{s\}} \frac{1-y^*_{us}}{2} w_u w_s + \frac{1}{2} \sum_{u\in V\setminus\{s\}} \sum_{v\in V\setminus\{s\} \mid v<u} \frac{1-y^*_{uv}}{2} w_u w_v \\
&\geq \frac{1}{2}\sum_{u<v} \frac{1-y^*_{uv}}{2} w_u w_v \\
&= \frac{\mathcal{W}(Y^*)}{2}.
\end{aligned}$$

□

Now, we present a solution X^* where the expectation of $\mathcal{C}(S)$ for the cut produced by heuristic *H1* is unbounded. Consider the case where $(1-y^*_{uv})/2 = 0$, that means no contribution to $\mathcal{C}(Y^*)$ and $(1-y^*_{us})/2 = 1/2$. In this case the probability that (u,v) is in $\delta(S)$ is $1/2$, that implies a half contribution to the expectation of $\mathcal{C}(S)$. Adding *edge by edge*, each edge could have a similar behavior, and the expectation of $\mathcal{C}(S)$ will grow unbounded in relation to the value of $\mathcal{C}(Y^*)$. See Figure 4.

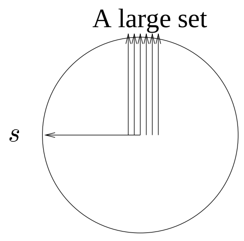

Figure 4: $\mathcal{C}(S)$ unbounded in *H1*.

Something symmetric happens for heuristic *H2*. In this case, we can obtain an upper bound to $E[\mathcal{C}(S)]$ within a factor to $\mathcal{C}(Y^*)$, but we couldn't obtain a lower bound to $E[\mathcal{W}(S)]$.

Lemma 4.6 *If S is the solution produced by heuristic $H2$, then $E[\mathcal{C}(S)] \leq \mathcal{C}(Y^*)$.*

Proof.

The probability that $(u, s) \in \delta(S)$ for all $u \in V \setminus \{s\}$ is given by

$$\Pr[(u,s) \in \delta(S)] = \frac{1 - \xi(X^*, u, s)}{2} = \frac{1 - \cos(v_u, v_s)}{2} = \frac{1 - y^*_{us}}{2}.$$

If u and v are not equal to s then

$$\Pr[(u,v) \in \delta(S)] = \left| \frac{y^*_{su} - y^*_{sv}}{2} \right| = \left| \frac{1 - y^*_{sv}}{2} - \frac{1 - y^*_{su}}{2} \right|,$$

that is exactly the probability to chose α values $\xi(X^*, s, v)$ and $\xi(X^*, s, u)$.

The triangle inequality constraint implies that

$$\frac{1 - y^*_{sv}}{2} - \frac{1 - y^*_{su}}{2} \leq \frac{1 - y^*_{uv}}{2} \quad \text{and} \quad \frac{1 - y^*_{su}}{2} - \frac{1 - y^*_{sv}}{2} \leq \frac{1 - y^*_{uv}}{2}.$$

Therefore, we have

$$\Pr[(u,v) \in \delta(S)] = \left| \frac{1 - y^*_{su}}{2} - \frac{1 - y^*_{sv}}{2} \right| \leq \frac{1 - y^*_{uv}}{2}, \quad \text{for all } (u,v) \in V \times V,$$

and

$$E[\mathcal{C}(S)] = \sum_{u<v} \Pr[(u,v) \in \delta(S)] c_{uv} \leq \sum_{u<v} \frac{1 - y^*_{uv}}{2} c_{uv} = \mathcal{C}(Y^*).$$

□

To obtain a solution X^* where the expectation of $\mathcal{W}(S)$ is not lower bounded, consider an edge (u, v), where $y^*_{uv} = -1$ (full contribution to $\mathcal{W}(Y^*)$), and $y^*_{us} = y^*_{vs}$. In this case we have $\Pr[(u,v) \in \delta(S)] = 0$. Symmetrically to *H1*, adding *edge by edge*, each edge may have a similar behavior, and the expectation of $\mathcal{W}(S)$ does not increase, becoming unbounded relative with the value of $\mathcal{W}(Y^*)$. See Figure 5.

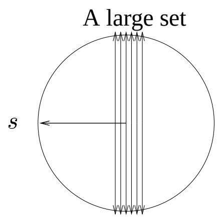

Figure 5: $\mathcal{W}(S)$ unbounded in $H2$.

The heuristic $H3$ uses the hyperplane rounding strategy presented by Goemans and Williamson [10] for the Max-Cut problem. They proved that the probability that an edge (u, v) is cut by a random hyperplane is $\arccos(y^*_{uv})/\pi$. As can be seen in Figure 6, this value is always greater than $0.878(1 - y^*_{uv})/2$. Therefore, the following lemma is straightforward.

Lemma 4.7 *If S is the solution produced by heuristic H3, then $E[\mathcal{W}(S)] \geq 0.878\mathcal{W}(Y^*)$.*

The difficult to obtain an upper bound to $\Pr[(u,v) \in \delta(S)] = \frac{\arccos(y^*_{uv})}{\pi}$ comes from the fact that $\frac{\arccos(y^*_{uv})}{\pi}$ cannot be bounded by $k\frac{1-y^*_{uv}}{2}$ for any constant k (see Figure 6).

When analyzing heuristic *H4*, we found out some interesting properties. The probability that (u,v) belongs to (A,B) is the probability that r_1 and r_2 cut (u,v), which values $(\arccos(x_{uv})/\pi)^2$, times the probability that $u \in A \cup B$, that is $1/2$.

$$\Pr[(u,v) \in (A,B)] = \left(\frac{\arccos(x_{uv})}{\pi}\right)^2 \frac{1}{2}.$$

We can note that this probability is upper and lower bounded by constant factors of the contribution in $\mathcal{C}(Y^*)$ and $\mathcal{W}(Y^*)$ (see Figure 7). More precisely, we have $E[\mathcal{C}(A,B)] \leq \frac{1}{2}\mathcal{C}(Y^*)$ and $E[\mathcal{W}(A,B)] \geq 0.2023\mathcal{W}(Y^*)$. Therefore, the following lemma is valid.

Lemma 4.8 *If S is the solution produced by heuristic H4 then $E[\mathcal{W}(S)] \geq 0.2023\mathcal{W}(Y^*)$.*

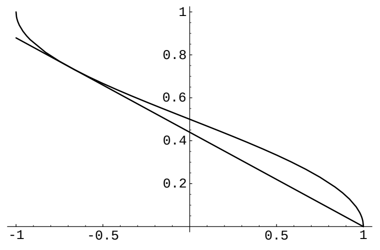

Figure 6: $\frac{\arccos(x)}{\pi}$ versus $0.878\left(\frac{1-x}{2}\right)$.

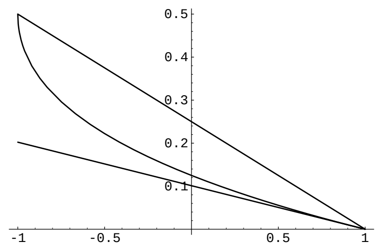

Figure 7: $0.2023\frac{1-x}{2}$ versus $\frac{1}{2}\frac{1-x}{2}$ versus $(\arccos(x)/\pi)^2\frac{1}{2}$.

Because the final cut S may have more edges than (A,B), we couldn't prove an upper bound for $E[\mathcal{C}(S)]$ by a factor of $\mathcal{C}(Y^*)$.

Consider now the case when $\xi(X^*,s,t) > -1$. One way to deal with this situation is to assure that one edge ij will be separated. In Section 5 we show in details how to do this separation. By solving the relaxation with this separation for an arbitrary vertex i and all $j \in V \setminus \{i\}$ and choosing the minimum, we can obtain a lower bound that contains a pair of vertices with $\xi(X^*,s,t) = -1$. Another way to deal with this situation is to discard empty cuts in the heuristics. In this case, $\mathcal{C}(S)$ and $\mathcal{W}(S)$ will be multiplied by the same factor and therefore we can maintain the same analysis, unless for heuristic *H1* that uses the fact $\xi(X^*,s,t) = -1$.

5 A Branch and Bound Algorithm for the Ratio Cut Problem

The idea used in the branch and bound algorithm, named $B\&B$, for the Ratio Cut problem is quite simple. It consists of using the relaxation as a dual bound and the heuristics as primal bounds. Although the relaxation has high time complexity, it leads to an excellent relaxation.

The first step consists to solve $A1^+$ relaxation. This leads the first lower bound. Apply the four heuristics over the result of the relaxation. Because the heuristics is much faster than solving the relaxation, each heuristic can be repeated a certain number of times without a significant waste of time.

While there exists a gap between the lower bound and the upper bound, we branch the execution. An arbitrary vertex is chosen, say vertex 0. We define a metric between each vertex $i \in V$ and the vertex 0, that is $\xi(X^*, i, 0) = \cos(v_i, v_0)$

If $\xi(X^*, i, 0) = 1$, we have the angle between v_i and v_0 as zero. In this case, as $A1^+$ has the constraint $|v_i| = |v_0|$, we have $v_i = v_0$. On the other hand, if $\xi(X^*, i, 0) = -1$ and, using similar arguments, we can conclude that $v_i = -v_0$.

When $\xi(X^*, i, 0)$ is in $\{-1, +1\}$ for all $i \in V$ all vector v_i is in $\{-v_0, v_0\}$ and the integer solution is reached. The constraint $|v_i| = |v_j|$ was necessary to conclude that the solution is integer. The formulation $A1$ does not contain this constraint and the $B\&B$ over $A1$ does not work correctly.

A node of the branch and bound tree is a semidefinite program Q composed by $A1^+$ more some integrality constraints. Let X_Q be an solution of Q. The "most" fractional variable for a given solution X_Q, say $ifrac(X_Q)$, is a vertex f such that $\xi(X^*, f, 0)$ is close to zero. The $B\&B$ algorithm maintains a heap sorted by $ifrac(X_Q)$ for each node Q. In each step, the algorithm selects a vertex f from the heap and, if Q value is smaller than the current lower bound, it produces new branching using the restrictions $\xi(X^*, f, 0) = 1$ and $\xi(X^*, f, 0) = -1$

If we add to $A1^+$ the constraint $x_{f0} = -x_{ff} = -x_{00}$, any further solution will satisfy $\xi(X^*, 0, f) = -1$. On the other hand, if we add the constraint $x_{f0} = x_{ff} = x_{00}$, any further solution will satisfy $\xi(X^*, 0, f) = 1$, so the branch and bound is possible.

Observe that the constraint $x_{ff} = x_{00}$ already belongs to $A1^+$.

The branch and bound algorithm beats the brute force algorithm (generate all possible cuts) to find an optimal solution and allow us to discover the integrality gap of $A1^+$ in graphs of the benchmark we have used.

In Figure 8, we present the branch and bound algorithm.

5.1 Generalization to Other Optimization Problems

The strategy of many Branch & Bound algorithms for a NP-hard problem is to consider an ILP formulation and to use additional restrictions to its relaxations. Because SDP is a generalization of LP, every formulation and relaxation valid in the previous is also valid to SDP. The difference is the fact that LP can be solved exactly and SDP can be solved within a small error ratio, both in polynomial time complexity.

To decide between SDP and LP, it is necessary to consider some important aspects.

The advantages of LP are considerable in many ways. There are many robust and stable implementations, such as CPLEX and XPRESS, that are comparatively faster than SDP solvers, many allowing integer programming formulations and features as pre-processing and branch-and-cut algorithms.

To compare the time to solve SDP and LP relaxations in instances of equivalent size see Table 2 (column *Av. CPU time Heu*) and Table 3 in Section 6. In these experiments, for instances with $n = 40$, the time to solve the associated LP was around 100 times faster than SDP.

```
ALGORITHM $B\&B(P)$,   where $P = A1^+(G, c, w)$
 1.  Let $X_{UB} \leftarrow Heuristics(P)$
 2.  Let $\mathcal{H}$ be a heap of semidefinite programs $Q$, indexed by the most
      fractionally variable (incident to vertex 0) in $X_Q$, say $ifrac(X_Q)$,
      where $X_Q$ is a solution of $Q$.
 3.  Let $\mathcal{H} \leftarrow \{P\}$.
 4.  While $\mathcal{H} \neq \emptyset$ do
 5.      let $Q \leftarrow RemoveMin(\mathcal{H})$ and $f \leftarrow ifrac(X_Q)$.
 6.      if $A1^+(Q) < \rho(X_{UB})$ then
 7.          let $Q' \leftarrow Q \cup \{x_{f0} = -x_{ff} = -x_{00}\}$
 8.              and $Q'' \leftarrow Q \cup \{x_{f0} = x_{ff} = x_{00}\}$
 9.          if $A1^+(Q') < \rho(X_{UB})$ then
10.              if $X_{Q'}$ is a feasible cut then $X_{UB} \leftarrow X_{Q'}$
11.              else
12.                  let $X' \leftarrow Heuristics(Q')$
13.                  if $\rho(X') < \rho(X_{UB})$ then $X_{UB} \leftarrow X'$
14.                  Let $\mathcal{H} \leftarrow \mathcal{H} \cup \{Q'\}$.
15.          if $A1^+(Q'') < \rho(X_{UB})$ then
16.              perform the corresponding steps executed in lines 10–14 for $Q''$.
17.  Return $X_{UB}$.
```

Figure 8: Semidefinite based branch and bound algorithm.

Another advantage of LP is the use of a *warm start* to solve the relaxations associated to each node. Although there are theoretical works that describe the possibility of a good initial point to a family of problems [16], some solvers, as the SDPA package, used in the experiments presented in this paper, has restrictions that do not allow this optimization. See more details in Section 6. In our implementation, the algorithm $B\&B$ solves the relaxation $A1^+$ from the beginning in each node of the branch and bound tree. Integer Linear Programming Solvers use a previous basis to obtain a new solution of a branching step.

On the other hand, SDP has some advantages. The use of SDP constraints can generate better lower bounds which may reduce the number of visited nodes. The result of a SDP, X^* in our notation, can be interpreted as a set of vectors, $\mathcal{V}$. In this case, we can use special rounding strategies, as done by heuristics $H3$ and $H4$.

Consider a general problem $\mathcal{P}$ described by an integer formulation. Any set of binary variables V^t can be described as $V^t = \{v_i^t \mid v_i^t \in \{-r, r\}\}$ for a constant $r \in \mathbb{R}$ and each variable v^t can be relaxed to a vector of length r and dimension $|V^t|$ by SDP constraints as

$$\begin{array}{rcl} X^t & \succeq & 0 \\ x_{ii} & = & r, \end{array}$$

where X^t is a square matrix with dimension $|V^t|$. Once we have solved the above semidefinite program, the vectors in V^t can be obtained by a Cholesky Decomposition of X^t.

The branching strategy described in Section 5 can be applied to construct a branch and bound algorithm. In many LP formulations that uses modulus, the relaxation usually does not give useful information and the use of a branch and bound over SDP may be competitive.

In Appendix A we have a list of problems where Semidefinite Programming is used as a tool to obtain an good approximation algorithms based on good relaxations, as Max Cut, Max *k*-Cut, *k*-Colouring and Satifiability problems.

When we relax natural integer linear formulations for Max Cut and Satifiability problems, the fractional solution can contain all edges or satisfy all clauses (see Subsection A.1 and A.2). Because of this weakness, the number of visited nodes in branch and bound algorithms based in these linear formulations can be very large. On the other hand, the SDP relaxation is more tight. For this reason, the number of visited nodes in a branch and bound tree is smaller.

For other problems, as Max *k*-Cut and *k*-Colouring, the relaxations of a natural ILP also don't give useful information, because it allows all edges to be in the cut or a 1-colouring fractional solutions (see Subsections A.4 and A.3). In these problems, SDP based branch and bound algorithms may be very promising. If two vertices have the same color or belong to the same set, we can use the following constraint:

$$x_{ij} = x_{ii} = x_{jj}.$$

By the other side, if two vertices must have different colors or must be in different sets we can use the following constraint:

$$x_{ji} = -1/(k-1).$$

6 Computational Results

The semidefinite programs were solved with the SDPA package [20], that is an implementation of a primal-dual interior-point method for semidefinite programming. Unfortunately, the *warm start* was not implemented in the SDPA package and we have to recompute the semidefinite program in each node of the branch and bound tree. Mitchell [16] presents a way to compute a new semidefinite point from the parent node in a semidefinite branch and bound tree. In theory, it is sufficient to start the resolution of SDP. If (X^0, Y^0) are initial primal and dual points, the SDPA solver looks for a solution (X, Y) such that $0 \preceq X \preceq omegaStar \times X^0$ and $0 \preceq Y \preceq omegaStar \times Y^0$, where $omegaStar$ is a parameter [8]. Our effort to compute a good initial point in the branch and bound procedure using the SDPA solver did not work, because the mentioned constraints limit the domain and non-optimal solutions were generated by the solver.

The experiments were performed following the same approach used by Goemans and Williamson [10] for the Max-Cut problem. We generated four types of instances: types A, B, C and D. In type A instances, each vertex has weight 1 and each edge is selected with probability 0.5 to receive cost 1. If the edge is not selected, its cost is zero. In instances of type B, the vertex weight is a random rational number uniformly distributed in $[0, 50]$, and the edge cost is also a rational number uniformly distributed in $[0, 50]$. In instances of type C, each vertex has weight 1, and each edge is selected with probability $9/|V|$ to receive cost 1. The edge cost is zero if the edge is not selected. While type A instances correspond to dense graphs, type C instances correspond to sparse ones, because the expected vertex degree is 9. In the instances of type D, we arbitrarily divided the set V in two disjoint sets, V_1 and V_2, where $|V_1| = |V_2|$. All vertices have weight 1. The edges inside V_1 or V_2 are selected with probability 0.5 to receive cost 1. Edges linking vertices in V_1 with vertices in V_2 are selected with probability 0.25 to receive cost 1. All edges not selected have cost zero.

We set the parameters of SDPA solver and the branch and bound algorithm to produce solutions with errors of at most 0.5%. Each heuristic was executed 1000 times on the same instance, returning the best obtained solution. Empty cuts were discarded.

For each instance and heuristic, we also saved a number k that is the iteration in which the respective heuristic found its best solution. The heuristic $H2$ does not have a k associated, because there are at most $|V| - 1$ possible solutions and we have obtained all of them. For the heuristics $H1$ and $H3$ we also count the empty cuts to compute the corresponding value of k, while for heuristic $H4$, we do not. The time to solve the heuristics without the time to solve the relaxation $A1^+$ is negligible (less than 0.3% of the time to solve $A1^+$).

We produced a set of instances containing 20, 30 and 40 vertices. For each instance, we obtained an optimum relaxed solution and applied the four heuristics, presented in Section 4, and the exact branch and bound algorithm. See Tables 1–2 for a summary of the results. The CPU times are given in seconds on a Xeon 2.4 GHz with 1024MB of RAM and Linux operational system. In the *Visited Nodes* column, we only considered non-leaf nodes in the branch tree. The *integrality gap* for an instance I is $\frac{B\&B(I)}{A1^+(I)}$. The *deviation factor* of a heuristic H_i is $\frac{H_i(I)}{B\&B(I)}$.

The relaxation $A1^+$ has integrality gap surprisingly good. For almost all tests the obtained relaxation value was equal to an optimum cut value. The quality of the relaxation leads to good heuristics and the branch and bound algorithm visited a small number of nodes in the branch and bound tree.

T y p	S i z	N. Inst	Integrality Gap		Average Deviation Factor				Maximum Deviation Factor			
			Av.	Max	*H1*	*H2*	*H3*	*H4*	*H1*	*H2*	*H3*	*H4*
A	20	400	1.005	1.01	1.05	1.00	1.00	1.00	1.54	1.12	1.00	1.00
	30	95	1.003	1.02	1.02	1.00	1.00	1.00	1.25	1.00	1.00	1.00
	40	5	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
B	20	400	1.00	1.002	1.008	1.00	1.00	1.00	2.78	1.00	1.00	1.00
	30	146	1.00	1.00	1.01	1.00	1.00	1.00	1.70	1.00	1.00	1.00
	40	26	1.00	1.00	1.01	1.00	1.00	1.00	1.27	1.00	1.00	1.00
C	20	400	1.007	1.015	1.045	1.00	1.00	1.00	1.75	1.00	1.00	1.00
	30	95	1.00	1.02	1.04	1.00	1.00	1.00	2.5	1.00	1.00	1.00
	40	2	1.00	1.00	1,00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
D	20	400	1.004	1.03	1.02	1.00	1.00	1.00	1.45	1.01	1.00	1.00
	30	93	1.00	1.01	1.02	1.00	1.00	1.00	1.17	1.00	1.00	1.00
	40	3	1.01	1.02	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 1: Tests using relaxation $A1^+$.

The performance of the branch and bound algorithm depends basically on the semidefinite programming solver, which is still a very active area. Clearly, the $B\&B$ algorithm with formulation $A1^+$ may also become faster using better solvers.

To investigate the formulation $A1^+$ more closely, we relaxed the semidefinite constraint. The new relaxation, named $A1^+_X := A1^+ \setminus \{X \succeq 0\}$, was solved by the Xpress-MP solver. The maximum integrality gap obtained with the optimum solution was 9%. The relaxation was solved around ninety times faster. See Table 3. The integrality gap for an instance I is $\frac{B\&B(I)}{A1^+_X(I)}$.

We also implemented the formulation used by Freivalds [6], which we denote by Fr, and solved it using the Xpress-MP solver [17] over instances of type A and size 20.

The formulation Fr is:

$$\begin{aligned} \min Fr(x) &= \sum_{i,j\in V} |x_i - x_j| c_{ij} \\ \text{subject to} \quad H(x) &= \sum_{i,j\in V} |x_i - x_j| d_i d_j = 1 \\ \sum_{i\in V} x_i &= 0, \end{aligned}$$

To convert the modulus to integer linear programing constraints, we used $N(N-1)/2$ additional binary variables.

Let the $MAX_{B\&B}$ and $AVG_{B\&B}$ be the maximum and the average time to $B\&B$ solve instances of size 20 in the experiment described by Table 2. We have $MAX_{B\&B} = 7000s$ and $AVG_{B\&B} = 323s$.

To solve instances of size 20 by Xpress over integer formulation Fr we set the maximum time as

Typ	Siz	N. Inst	Average k			Av. CPU time		Visited Nodes.	
			H1	*H3*	*H4*	Heu	$B\&B$	Av.	Max
A	20	400	135	10	2.3	155	439	1.2	21
	30	95	86	9.5	2	6668	6668	1.0	1
	40	5	29	7.8	1.2	30564	91510	1.0	1
B	20	400	18	4.3	2.2	108	108	1.0	1
	30	146	20	3.3	2.1	4305	4305	1.0	1
	40	26	6.6	3.2	2.0	18718	20761	1.0	1
C	20	400	118	8.6	2.3	137	473	1.4	6
	30	95	82	12	4.0	6568	6568	1.0	1
	40	2	68	6.0	1.5	26660	119623	1.5	2
D	20	400	95	10	2.4	166	273	1.0	3
	30	93	77	9.7	2.2	6695	6695	1.0	1
	40	3	22	5.0	1.6	28980	90928	1.0	1

Table 2: Tests using relaxation $A1^+$.

$2MAX_{B\&B}$ seconds. The Xpress-MP solver was configured to stop when the gap between primal and dual bounds is within 1%. We tested it in fourteen instances an for all of them Xpress-MP solver stopped in $2MAX_{B\&B}$ seconds without a solution with a gap within 1%.

We improve Fr with triangle inequalities, generating a formulation we denote by $Fr_{\triangle}$, and apply the same experiments to all instances of size 20. The Xpress-MP solver could found solutions (within 1% of the optimum) in a few seconds for the great majority of the instances, but for 12 of them, the solver visited more than $52{,}000$ nodes in the branch and bound tree and spent more than $MAX_{B\&B}$ seconds. For 6 instances, the solver spent more than $2MAX_{B\&B}$ seconds and stopped without a solution within 1% of the optimum.

Let T be the time to solve an instance with $Fr_{\triangle}$ formulation in the Xpress-MP solver, and Nod the number of visited nodes in the branch and bound tree. In Tables 4 we present the behavior of this formulation for the instances of size 20.

7 Conclusion

We analyzed a known relaxation for the Ratio Cut problem using semidefinite programming. We also presented an exact algorithm and heuristics based on this formulation. The presented heuristics obtained solutions with value very close to the optimum and the exact algorithm could be executed on small and medium sized instances. The solution of the branch and bound algorithm was competitive, compared to a known integer programming formulation solved with the Xpress-MP package.

The performed tests showed that the use of semidefinite programming for this problem is a very promising strategy. With the continuous advances on the theory of semidefinite programming (as occurred with the theory of linear programming) we hope the presented algorithms could obtain

Type	Size	N. Inst	Average Int. Gap	Max. Int. Gap	Average Time (s)
A	20	400	1.003	1.06	2.0
	30	95	1.008	1.09	38.7
	40	5	1.000	1.00	250.6
B	20	400	1.000	1.00	1.9
	30	146	1.000	1.00	46.3
	40	26	1.000	1.00	231.3
C	20	400	1.001	1.05	1.7
	30	95	1.004	1.09	35.2
	40	2	1.000	1.00	205.9
D	20	400	1.003	1.06	1.7
	30	93	1.010	1.07	36.5
	40	3	1.008	1.02	259.5

Table 3: Result of relaxation $A1_X^+$.

Time	N. Inst	Visited Nodes
Without solution in $2MAX_{B\&B}$	6	$Nod > 85000$
$MAX_{B\&B} < T \leq 2MAX_{B\&B}$	6	$52000 < Nod \leq 85000$
$AVG_{B\&B} < T \leq 2MAX_{B\&B}$	49	$2800 < Nod \leq 52000$
$AVG_{B\&B}/100 < T \leq AVG_{B\&B}$	591	$19 < Nod \leq 2800$
$T < AVG_{B\&B}/100$	948	$Nod \leq 19$
Total	1600	

Table 4: Execution time in and number of visited nodes in $Fr_\triangle$.

solutions for larger graphs in less time.

8 Acknowledges

We would like to thank Cristina G. Fernandes for helpful suggestions to this article.

References

[1] Sanjeev Arora, James R. Lee, and Assaf Naor. Euclidean distortion and the sparsest cut.

[2] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings, and graph partitionings. *ACM Symposium on Theory of Computing*, 2004.

[3] Evandro C. Bracht, Luis A. A. Meira, and Flavio Keidi Miyazawa. A greedy approximation algorithm for the uniform labeling problem analyzed by a primal-dual technique. In *WEA*, volume 3059 of *Lecture Notes in Computer Science*, pages 145–158. WEA, Springer, 2004.

[4] Shuchi Chawla, Anupam Gupta, and Harald Räcke. Embeddings of negative-type metrics and an improved approximation to generalized sparsest cut. *SODA*, 2005.

[5] P. Crescenzi and V. Kann. *A compendium of NP optimization problems*, 1994.

[6] K. Freivalds. A nondifferentiable optimization approach to ratio-cut partitioning. In *Proc. 2nd Workshop on Efficient and Experimental Algorithms*, Lectures Notes on Computer Science, LNCS 2647. Springer-Verlag, 2003.

[7] A. Frieze and M. Jerrum. Improved approximation algorithms for max k-cut and max bisection. *Algorithmica*, pages 67–81, 1997.

[8] Katsuki Fujisawa, Masakasu Kojima, Kasuhide Nakata, and Makoto Yamashita. *SDPA Users's Manual, Version 6.00* page 39, 2002.

[9] M.X. Goemans and D.P. Williamson. .878-approximation algorithms for MAX CUT and MAX 2SAT. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 422–431, Montréal, Québec, Canada, 23–25 May 1994.

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

[11] C. Helmberg. Semidefinite programming for combinatorial optimization. *Habilitationsschrift*, ZIB-Report ZR-00-34, Konrad-Zuse-Zentrum, October 2000.

[12] Thomas Hofmeister and Martin Hühne. An introduction to semidefinite programming and its applications to approximation algorithms, 1997. Reihe Computational Intelligence.

[13] D. Karger, R. Motwani, and M. Sudan. Approximate graph coloring by semidefinite programming. In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1994.

[14] M. Laurent and R. Rendl. Semidefinite programming and integer programming. Technical report, CWI, Amsterdam, 2002.

[15] T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. *Proc. 29th Ann. IEEE Symp. on Foundations of Comput. Sci., IEEE Computer Society, 422-431.*, 1988.

[16] J. E. Mitchell. Restarting after branching in the SDP approach to MAX-CUT and similar combinatorial optimization problems. *Journal of Combinatorial Optimization*, 5(2), 2001.

[17] Dash Optimization. *Xpress-MP Manual. Release 13.* 2002.

[18] Michael J. Todd. Semidefinite programming: Applications, duality, and interior-point methods, October 1998. INFORMS.

[19] S. Wang and J. Siskind. Image segmentation with ratio cut, 2003.

[20] Makoto Yamashita. Semidefinite programming algorithm, 2002.

[21] Heng Yang, Yinyu Ye, and Jiawei Zhang. An approximation algorithm for scheduling two parallel machines with capacity constraints. *Discrete Appl. Math.*, 130(3):449–467, 2003.

A Formulations and Relaxations for a Series of Optimization Problems

In the following we present some semidefinite formulations for problems for which the strategy presented in this paper may be used. These formulations have been used to obtain good approximation algorithms for Max Cut, Max *k*-Cut, *k*-Colouring and Satifiability problems.

In what follows we show the definition, an integer or modulus formulation and a semidefinite formulation for each one of this problems.

A.1 Max Cut Problem

Bellow we present a Max Cut Problem definition:

MAX CUT PROBLEM: Given a graph $G = (V, E)$, let a cut S be a proper subset of V, and let $\delta(S)$ be the set of edges with exactly one endpoint in S. The objective is to find a cut S such as $|\delta(S)|$ is maximum. In the Weighted Max Cut Problem, there is an associated cost $c_e \in \mathbb{R}^+$ for each edge $e \in E$. The objective is to find a cut S such that $\sum_{e\in\delta(S)} w_e$ is maximum.

The Max Cut problem can be formulated as an integer linear program and an integer semidefinite program, as follows:

$$\begin{array}{llcll} & IP_{Max\ Cut} & & & SDP_{Max\ Cut} \\ \max & \displaystyle\sum_{\{i,j\}\in E} \frac{1}{2}|x_i - x_j|\ c(i,j) & & \max & \displaystyle\sum_{i<j} \frac{1}{2}(1 - v_i.v_j)\ c(i,j) \\ & \quad x_i \in \{-1, 1\} \quad \forall i \in V, & & & \quad v_i \in \{-e^1, e^1\}, \end{array}$$

where e^1 is a vector with all elements equal to zero, unless the first, that values 1. That is, $e^1 = (1, 0, 0, \ldots, 0)$.

If we relax $IP_{Max\ Cut}$ to a linear program, the objective function will achieve $\sum_{e\in E} c_e$, that means no useful information. On the other hand, the integrality gap of $SDP_{Max\ Cut}$, with v_i relaxed to unit sphere, is greater than 0.878 [9].

A.2 Satisfiability Problem

Bellow we present a Satifiability Problem definition:

SATISFIABILITY PROBLEM: Given a set U of variables, a collection $\mathcal{C}$ of disjunctive clauses, where a literal is a variable or a negated variable in U. A solution ϕ is a truth/false assignment for each variable $u \in U$, in other words, a solution is a function $\phi : U \to \{T, F\}$. The objective is to find an attribution ϕ where the number of clauses satisfied is maximnum

The Satifiability problem can be formulated as an integer linear program and an integer semidefinite program, as follows:

$$IP_{SAT}$$

$$\begin{array}{ll}
\max & \sum_{i} z_i \\
\text{s.t.} & \\
\multicolumn{2}{l}{\sum_{u\in I^+j} x_u + \sum_{v\in I_j^-} (1 - x_v) \geq z_i, \forall C_i \in \mathcal{C}} \\
& x_u \in \{0,1\}, \quad \forall u \in U \\
& z_i \in \{0,1\}, \quad \forall i : C_i \in \mathcal{C}
\end{array}$$

$$SDP_{SAT}$$

$$\begin{array}{l}
\max \sum_{C_j\in\mathcal{C}} z_j \\
\text{s.t.} \\
\sum_{i\in I_j^+} u(x_i) + \sum_{i\in I_j^-} u(\bar{x_i}) \geq z_j, \forall C_j \in \mathcal{C}, \\
u(C_j) \geq z_j, \forall C_j \in \mathcal{C},\ l(C_j) = 2, \\
\frac{1}{l(C_j)} \sum_{C\in P_j} u(C) \geq z_j, \forall C_j \in \mathcal{C},\ l(C_j) \geq 2, \\
z_j \in \{0,1\}, \\
v_i \in \{-e^1, e^1\},
\end{array}$$

where e^1 is a vector with all elements equal to zero, unless the first, that values 1, v_0 is a reference to the truth value, $u(x_i) = (1 - v_i.v_0)$, $u(\bar{x_i}) = (1 - v_i.v_0)$, $u(x_i \vee x_j) = \frac{1}{4}(1 + v_i.v_0) + \frac{1}{4}(1 + v_j.v_0) + \frac{1}{4}(1 + v_i.v_j)$, I_j^+ is the set of non-negated variables in clause C_j, I_j^- is the set of negated variables in clause C_j, $l(C_j)$ is the number os variables of C_j and P_j is a set of length 2 formed by the literals of C_j, two at time. For more details about SDP_{SAT} see [10].

If we relax IP_{SAT} to a linear program, the objective function will achieve all clauses satisfied, that means no useful information. On the other hand, the integrality gap of SDP_{SAT}, with v_i relaxed to a vector int he unit sphere and z_j in $[0, 1]$, is greater than 0.7584 [10].

A.3 Max k-Cut Problem

Bellow we present a Max *k*-Cut Problem definition:

MAX k-CUT PROBLEM: Given a graph $G = (V, E)$, a weight function, $w : E \to \mathbb{R}^+$, and an integer $k \in [2, \ldots, |V|]$, the objective is to find a partition F of V into k disjoint sets, $F = \{C_1, C_2, \ldots, C_k\}$, such that the sum of the weight of the edges between the disjoint sets,

$$\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \sum_{\substack{v_1\in C_i \\ v_2\in C_j}} w(v_1, v_2),$$

is maximum.

For $k = 2$ the problem is the Max Cut problem (see Section A.1). The best approximation algorithm for the Max *k*-Cut Problem is due to Frieze and Jerrum [7].

The Max *k*-Cut problem can be formulated as an integer linear program and an integer semidefinite program, as follows:

$$IP_{\text{Max}k\text{ Cut}}$$
$$\begin{array}{ll} \max & \sum_{e \in E} w_e y_e \\ \text{s.t.} & \\ & \sum_{i=1}^{k} x_{vi} = 1, \quad \forall v \in V, \\ & z_{uvi} = |x_{ui} - x_{vi}|, \quad \forall u, v \in V,\ i \in \{1, \ldots, k\}, \\ & y_{uv} = 1 - \frac{1}{2}\sum_{i=1}^{k} z_{uvi}, \quad \forall u, v \in V, \\ & x_{vi} \in \{0, 1\}, \quad \forall v \in V,\ i \in \{1, \ldots, k\}, \end{array}$$

$$SDP_{\text{Max}k\text{ Cut}}$$
$$\begin{array}{ll} \max & \frac{k-1}{k}\sum_{i<j} w_{ij}(1 - v_i.v_j) \\ \text{s.t.} & \\ & v_i.v_i = 1, \quad \forall i \in V, \\ & v_i.v_j = \{0, \frac{-1}{k-1}\}, \quad \forall i, j \in V. \end{array}$$

If we relax $IP_{\text{Max}k\text{ Cut}}$ to a linear programming, the objective function will achieve $\sum_{e \in E} w_e$, that means no useful information. On the other hand, the formulation $SDP_{\text{Max}k\text{ Cut}}$, with $v_i.v_j$ relaxed to $[-1, 1]$, is used as a tool to obtain an approximation algorithm to the problem [7].

A.4 k-Colouring Problem

Bellow we present a *k*-Colouring Problem definition:

> k-COLOURING PROBLEM: A legal vertex coloring of a graph $G(V, E)$ is an assignment of colors to its vertices such that no two adjacent vertices receives the same color. Equivalently, a legal coloring of G by k colors is a partition of its vertices into k independent sets. The *k*-Colouring problem consists in to find a legal coloring of G by k colors.

The *k*-Colouring problem can be formulated as an integer linear program and an integer semidefinite program, as follows:

$$IP_{k\text{ Colouring}}$$
$$\begin{array}{ll} \min & z \\ \text{s.t.} & \\ & x_u \leq z, \quad \forall u \in V, \\ & |x_u - x_v| \geq 1, \quad \forall \{u, v\} \in E, \\ & x_u \in \mathbb{Z}^+ \end{array}$$

$$SDP_{k\text{ Colouring}}$$
$$\begin{array}{ll} \min & z \\ \text{s.t.} & \\ & v_i.v_j \leq z, \quad \forall i, j \in V, \\ & v_i.v_i = 1, \quad \forall i \in V, \\ & v_i.v_j = \frac{-1}{k-1}, \quad \forall i, j \in E. \\ & v_i.v_j = \{0, \frac{-1}{k-1}\}, \quad \forall i, j \in V. \end{array}$$

If we relax $IP_{k\text{ Colouring}}$ to a linear program, the objective function will achieve zero, that means no useful information. On the other hand, the formulation $SDP_{k\text{ Colouring}}$, with $v_i.v_j$ relaxed to $[-1, 1]$, is used as a tool to obtain an approximation algorithm to the problem [13].