SpeeDP: a fast method for solving the SDP relaxation of Max Cut

Veronica Piccialli*
Joint work with Luigi Grippo†, Laura Palagi†, Mauro Piacentini†, Giovanni Rinaldi‡

ESI 2010-Klagenfurt

*University of Rome Tor Vergata
†DIS-Sapienza University of Rome
‡IASI-CNR

LRSDP relaxation

$$\begin{aligned} \min \quad & \langle Q, X \rangle \\ & \operatorname{diag}(X) = e \\ & X \succeq 0, X \in \mathcal{S}^n \end{aligned} \qquad (\mathrm{SDP_{MC}})$$

LRSDP relaxation

$$\begin{aligned} \min \quad & \langle Q, X \rangle \\ & \operatorname{diag}(X) = e \\ & X \succeq 0, X \in \mathcal{S}^n \end{aligned} \qquad (\mathrm{SDP_{MC}})$$

$X \in \mathcal{S}^n,\ X \succeq 0$, X of rank r if and only if $X = VV^T$ where $V = [v_1 \dots v_n]^T \in \mathbb{R}^{n \times r}$

LRSDP relaxation

$$\begin{array}{ll} \min & \langle Q, X \rangle \\ & \operatorname{diag}(X) = e \\ & X \succeq 0, X \in \mathcal{S}^n \end{array} \qquad (\mathrm{SDP_{MC}})$$

$X \in \mathcal{S}^n,\ X \succeq 0$, X of rank r if and only if $X = VV^T$ where $V = [v_1 \dots v_n]^T \in \mathbb{R}^{n \times r}$

If there exists a solution of problem (SDP$_{MC}$) of rank r, a global solution of problem

$$\begin{array}{ll} \min & q_r(V) := \langle Q, VV^T \rangle \\ & \langle E_{ii}, VV^T \rangle = 1, \quad i = 1, \dots, n, \qquad V \in \mathbb{R}^{n \times r} \end{array}$$

gives a solution of Problem (SDP$_{\mathrm{MC}}$) .

LRSDP relaxation

$$\begin{array}{ll} \min & \langle Q, X\rangle \\ & \mathrm{diag}(X) = e \\ & X \succeq 0, X \in \mathcal{S}^n \end{array} \qquad (\mathrm{SDP_{MC}})$$

$X \in \mathcal{S}^n,\ X \succeq 0$, X of rank r if and only if $X = VV^T$ where $V = [v_1 \dots v_n]^T \in \mathbb{R}^{n\times r}$

If there exists a solution of problem (SDP$_{MC}$) of rank r, a global solution of problem

$$\begin{array}{ll} \min & q_r(V) := \langle Q, VV^T\rangle \\ & \langle E_{ii}, VV^T\rangle = 1, \quad i = 1, \dots, n, \qquad V \in \mathbb{R}^{n\times r} \end{array}$$

gives a solution of Problem (SDP$_{\mathrm{MC}}$) .

[Barvinok 95, Pataki] there exists an $X \in \mathcal{S}^n$ optimal solution of (SDP$_{\mathrm{MC}}$) with rank r satisfying the inequality $r(r+1)/2 \le n$.

LRSDP relaxation

$$\begin{array}{ll} \min & \langle Q, X\rangle \\ & \mathrm{diag}(X) = e \\ & X \succeq 0, X \in \mathcal{S}^n \end{array} \qquad (\mathrm{SDP_{MC}})$$

$X \in \mathcal{S}^n,\ X \succeq 0$, X of rank r if and only if $X = VV^T$ where $V = [v_1 \dots v_n]^T \in \mathbb{R}^{n\times r}$

If there exists a solution of problem (SDP$_{MC}$) of rank r, a global solution of problem

$$\begin{array}{ll} \min & q_r(V) := \langle Q, VV^T\rangle \\ & \langle E_{ii}, VV^T\rangle = 1, \quad i = 1, \dots, n, \qquad V \in \mathbb{R}^{n\times r} \end{array}$$

gives a solution of Problem (SDP$_{\mathrm{MC}}$) .

[Barvinok 95, Pataki] there exists an $X \in \mathcal{S}^n$ optimal solution of (SDP$_{\mathrm{MC}}$) with rank r satisfying the inequality $r(r+1)/2 \le n$.

$\widehat{r} = \max\{k \in N : k(k+1)/2 \le n\} = \left\lfloor \frac{\sqrt{1+8n}-1}{2} \right\rfloor$ **computable** upper bound on r

Relaxations

(NLP$_r$)

$$\begin{aligned} q_r^*(V) \quad &= \min \ \langle Q, VV^T \rangle \\ & \langle E_{ii}, VV^T \rangle = 1 \ i = 1, \ldots, n, \\ & V \in \mathbb{R}^{n \times r} \end{aligned}$$

Relaxations

(NLP$_r$)

$$\begin{aligned} q_r^*(V) \quad &= \min \ \langle Q, VV^T \rangle \\ & \langle E_{ii}, VV^T \rangle = 1 \ i = 1, \ldots, n, \\ & V \in \mathbb{R}^{n \times r} \end{aligned}$$

When $r = 1$,

$$\begin{aligned} &\min \ x^T Q x \\ &x^T E_{ii} x = x_i^2 = 1 \ i = 1, \ldots, n, \end{aligned} \qquad (\text{NLP}_1)$$

Relaxations

(NLP$_r$)

$$\begin{aligned} q_r^*(V) \quad &= \min \ \langle Q, VV^T \rangle \\ & \langle E_{ii}, VV^T \rangle = 1 \ i = 1, \ldots, n, \\ & V \in \mathbb{R}^{n \times r} \end{aligned}$$

When $r = 1$,

$$\begin{aligned} &\min \ x^T Q x \\ &x^T E_{ii} x = x_i^2 = 1 \ i = 1, \ldots, n, \end{aligned} \qquad (\mathrm{NLP}_1)$$

$=$(MC)

Relaxations

(NLP_r)

$$\begin{aligned} q_r^*(V) \quad &= \min \ \langle Q, VV^T \rangle \\ & \langle E_{ii}, VV^T \rangle = 1 \ i = 1, \ldots, n, \\ & V \in \mathbb{R}^{n \times r} \end{aligned}$$

When $r = 1$,

$$\begin{aligned} &\min \ x^T Q x \\ &x^T E_{ii} x = x_i^2 = 1 \ i = 1, \ldots, n, \end{aligned} \qquad (\text{NLP}_1)$$

$=$(MC)

$$q_r^* \leq q_1^* := z_{\text{MC}}^*.$$

Relaxations

(NLP_r)

$$\begin{aligned} q_r^*(V) \quad &= \min \ \langle Q, VV^T \rangle \\ & \langle E_{ii}, VV^T \rangle = 1 \ i = 1, \ldots, n, \\ & V \in \mathbb{R}^{n \times r} \end{aligned}$$

When $r = 1$,

$$\begin{aligned} &\min \ x^T Q x \\ &x^T E_{ii} x = x_i^2 = 1 \ i = 1, \ldots, n, \end{aligned} \qquad (\text{NLP}_1)$$

$=$(MC)

$$q_r^* \leq q_1^* := z_{\text{MC}}^*.$$

the feasible region is enlarging

Relaxations

(NLP_r)

$$q_r^*(V) = \min \langle Q, VV^T \rangle$$
$$\langle E_{ii}, VV^T \rangle = 1 \; i = 1, \ldots, n,$$
$$V \in \mathbb{R}^{n \times r}$$

(SDP_{MC})

$$z_{\text{SDP}}^* = \min \langle Q, X \rangle$$
$$\text{diag}(X) = e$$
$$X \succeq 0, X \in \mathcal{S}^n$$

When $r = 1$,

$$\min \; x^T Q x$$
$$x^T E_{ii} x = x_i^2 = 1 \; i = 1, \ldots, n,$$

(NLP_1) =(MC)

$$q_r^* \leq q_1^* := z_{\text{MC}}^*.$$

the feasible region is enlarging

Relaxations

(NLP$_r$)

$$\begin{aligned} q_r^*(V) \quad &= \min \ \langle Q, VV^T \rangle \\ & \langle E_{ii}, VV^T \rangle = 1 \ i = 1, \ldots, n, \\ & V \in \mathbb{R}^{n \times r} \end{aligned}$$

(SDP$_{\mathrm{MC}}$)

$$\begin{aligned} z_{\mathrm{SDP}}^* = \quad & \min \langle Q, X \rangle \\ & \mathrm{diag}(X) = e \\ & X \succeq 0, X \in \mathcal{S}^n \end{aligned}$$

When $r = 1$,

$$\begin{aligned} & \min \ x^T Q x \\ & x^T E_{ii} x = x_i^2 = 1 \ i = 1, \ldots, n, \end{aligned} \qquad (\mathrm{NLP}_1)$$

=(MC)

$$z_{\mathrm{SDP}}^* := q_n^* := q_{\hat{r}}^* := q_{r_{\min}}^* \leq q_r^* \leq q_1^* := z_{\mathrm{MC}}^*.$$

the feasible region is enlarging

A NS G.O. condition

Proposition [Grippo, Palagi, P. 09] A point $V^* \in \mathbb{R}^{n \times r}$ is a global minimizer of Problem (NLP$_r$) that solves Problem (SDP$_{\mathrm{MC}}$) if and only if there exists a $\lambda^* \in \mathbb{R}^n$ such that

$$\begin{aligned} &(Q + \Lambda^*)\, V^* = 0 \\ &Q + \Lambda^* \succeq 0 \\ &\langle E_{ii}, V^* \rangle = 1, \quad i = 1, \ldots, n. \end{aligned} \tag{1}$$

A NS G.O. condition

Proposition [Grippo, Palagi, P. 09] A point $V^* \in \mathbb{R}^{n \times r}$ is a global minimizer of Problem (NLP$_r$) that solves Problem (SDP$_{\mathrm{MC}}$) if and only if there exists a $\lambda^* \in \mathbb{R}^n$ such that

$$\begin{aligned} &(Q + \Lambda^*)\, V^* = 0 \\ &Q + \Lambda^* \succeq 0 \\ &\langle E_{ii}, V^* \rangle = 1, \quad i = 1, \ldots, n. \end{aligned} \tag{1}$$

The Lagrange multiplier associated to the solution of Problem (NLP$_r$) is a solution of the dual of problem (SDP$_{\mathrm{MC}}$).

A NS G.O. condition

Proposition [Grippo, Palagi, P. 09] A point $V^* \in \mathbb{R}^{n \times r}$ is a global minimizer of Problem (NLP$_r$) that solves Problem (SDP$_{\mathrm{MC}}$) if and only if there exists a $\lambda^* \in \mathbb{R}^n$ such that

$$\begin{aligned} &(Q + \Lambda^*)\, V^* = 0 \\ &Q + \Lambda^* \succeq 0 \\ &\langle E_{ii}, V^* \rangle = 1, \quad i = 1, \ldots, n. \end{aligned} \tag{1}$$

The Lagrange multiplier associated to the solution of Problem (NLP$_r$) is a solution of the dual of problem (SDP$_{\mathrm{MC}}$).

There is a closed form for computing the multiplier given a KKT point:

A NS G.O. condition

Proposition [Grippo, Palagi, P. 09] A point $V^* \in \mathbb{R}^{n \times r}$ is a global minimizer of Problem (NLP$_r$) that solves Problem (SDP$_{\mathrm{MC}}$) if and only if there exists a $\lambda^* \in \mathbb{R}^n$ such that

$$\begin{aligned} &(Q + \Lambda^*)\, V^* = 0 \\ &Q + \Lambda^* \succeq 0 \\ &\langle E_{ii}, V^* \rangle = 1, \quad i = 1, \ldots, n. \end{aligned} \tag{1}$$

The Lagrange multiplier associated to the solution of Problem (NLP$_r$) is a solution of the dual of problem (SDP$_{\mathrm{MC}}$).

There is a closed form for computing the multiplier given a KKT point:

- $\hat{\lambda}_i = -\langle E_{ii} Q, \hat{V}\hat{V}^T \rangle, \quad i = 1, \ldots, n$

A NS G.O. condition

Proposition [Grippo, Palagi, P. 09] A point $V^* \in \mathbb{R}^{n\times r}$ is a global minimizer of Problem (NLP$_r$) that solves Problem (SDP$_{\mathrm{MC}}$) if and only if there exists a $\lambda^* \in \mathbb{R}^n$ such that

$$\begin{aligned} &(Q + \Lambda^*)\, V^* = 0 \\ &Q + \Lambda^* \succeq 0 \\ &\langle E_{ii}, V^* \rangle = 1, \quad i = 1, \ldots, n. \end{aligned} \tag{1}$$

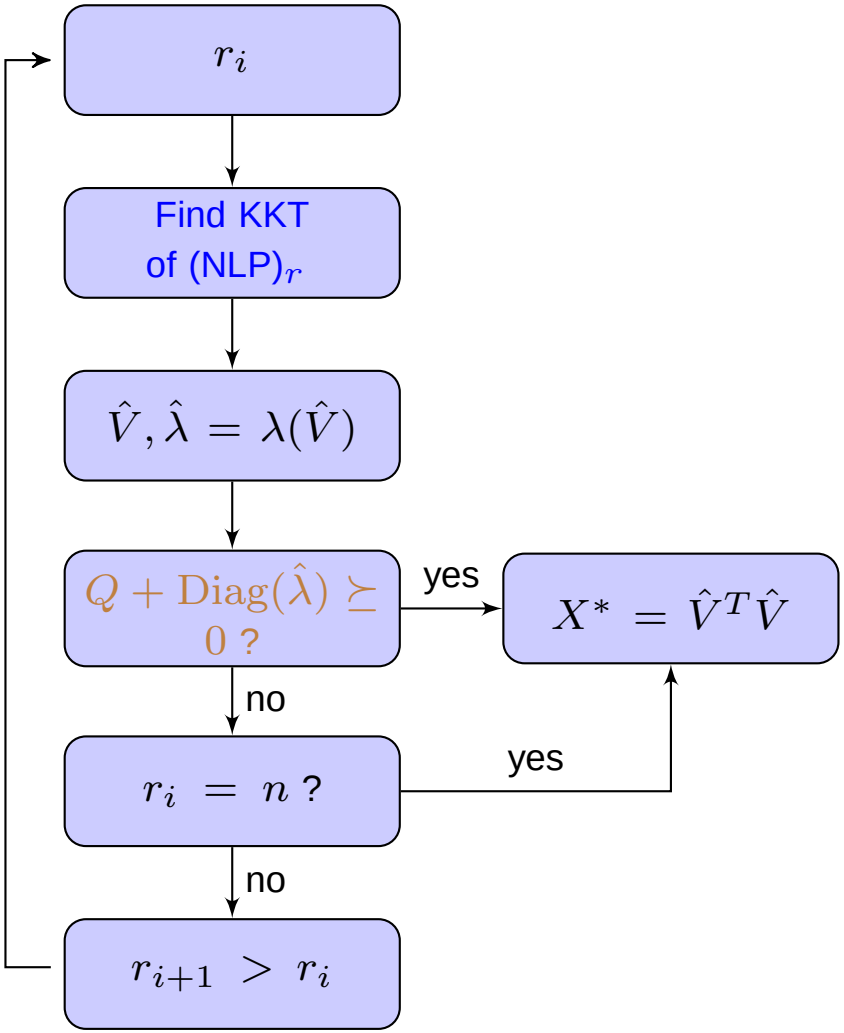
The Lagrange multiplier associated to the solution of Problem (NLP$_r$) is a solution of the dual of problem (SDP$_{\mathrm{MC}}$).

There is a closed form for computing the multiplier given a KKT point:

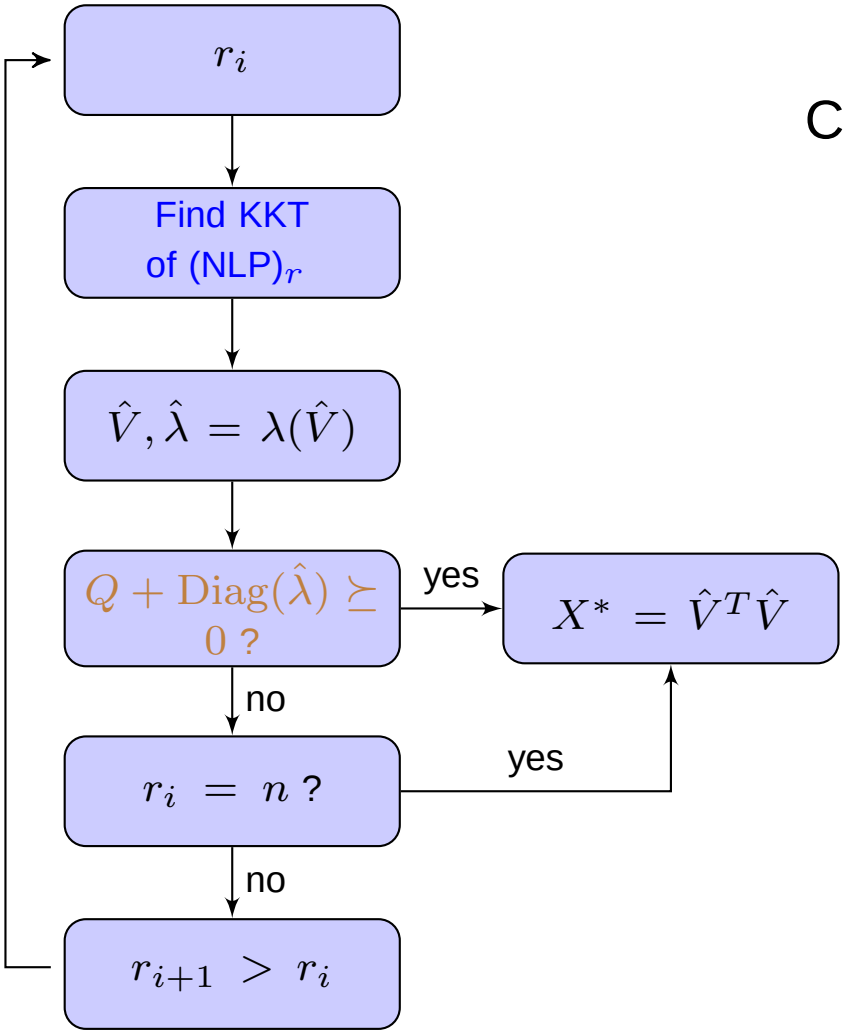
- $\hat{\lambda}_i = -\langle E_{ii} Q, \hat{V}\hat{V}^T \rangle, \quad i = 1, \ldots, n$
- $\sum_{i=1}^{n} \hat{\lambda}_i = e^T \lambda = -\langle Q, \hat{V}\hat{V}^T \rangle = -q_r(\hat{V})$

The closed expression of the multiplier can be generalized for problems satisfying $A_i A_j = 0$[JourneèBachAbsilSepulchre08]

A general algorithmic approach

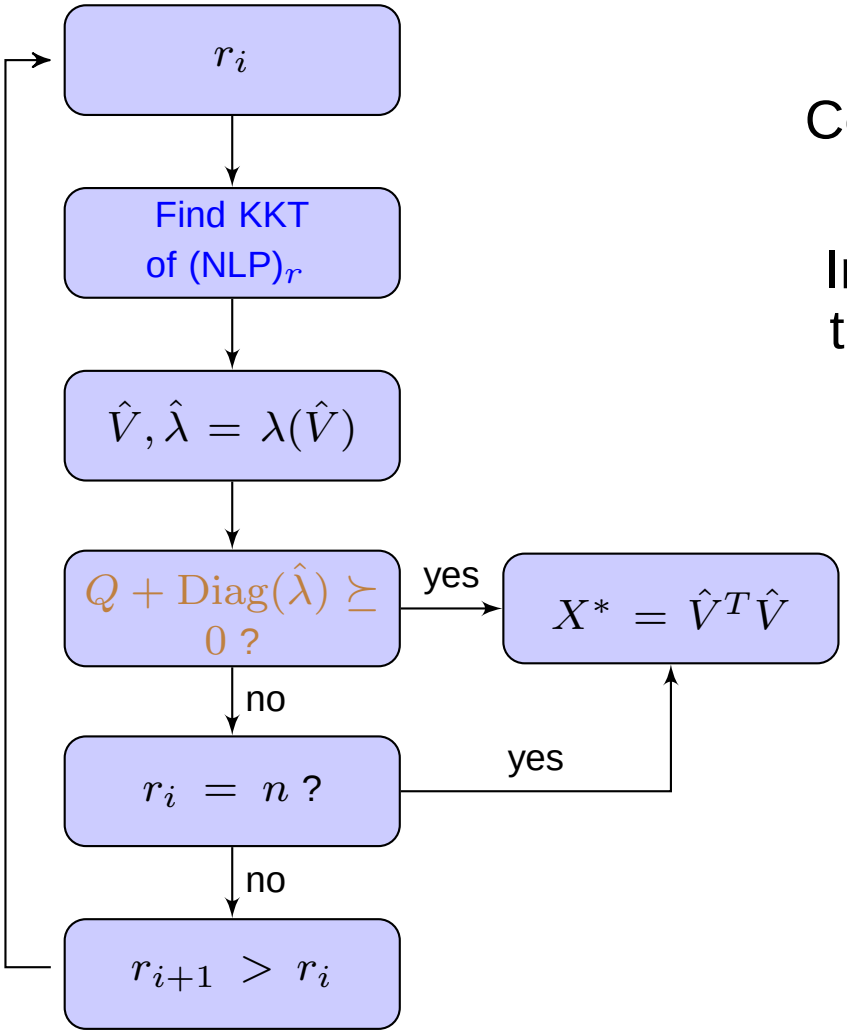


A general algorithmic approach



Computational evidence: $r^* << n$

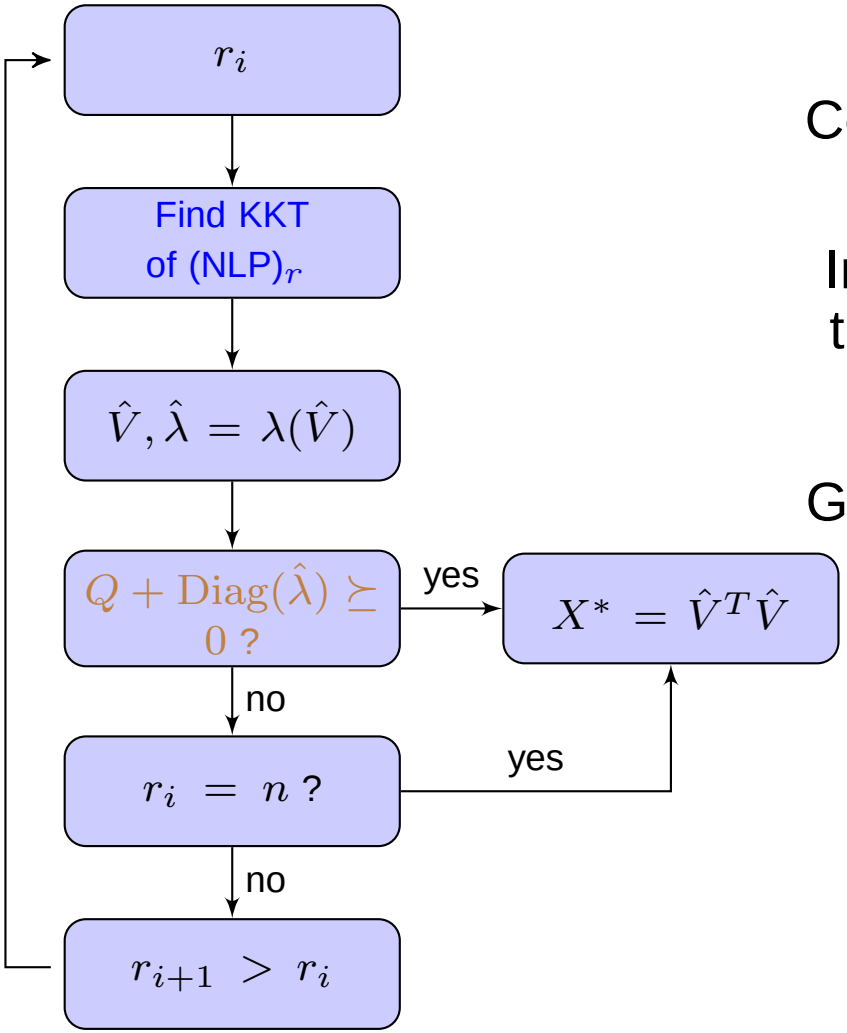
A general algorithmic approach



Computational evidence: $r^* << n$

Inner problem solution: transform the constrained problem into an unconstrained one

A general algorithmic approach



Computational evidence: $r^* << n$

Inner problem solution: transform the constrained problem into an unconstrained one

Global condition: find the minimum eigenvalue of $Q + \mathrm{Diag}(\hat{\lambda})$

Unconstrained formulations for (NLP$_r$)

Augmented Lagrangian [Burer-Monteiro, 2003]

$$\min_V L_a(V, \lambda^k, \epsilon^k) = L(V, \lambda^k) + \frac{1}{\epsilon^k} \sum_i^n \left(\langle E_{ii}, VV^T \rangle - 1 \right)^2$$

for a sequence $\{\lambda^k, \epsilon^k\}$, $\epsilon^k \to 0$

Unconstrained formulations for (NLP$_r$)

Augmented Lagrangian [Burer-Monteiro, 2003]

$$\min_V L_a(V, \lambda^k, \epsilon^k) = L(V, \lambda^k) + \frac{1}{\epsilon^k} \sum_i^n \left(\langle E_{ii}, VV^T \rangle - 1 \right)^2$$

for a sequence $\{\lambda^k, \epsilon^k\}$, $\epsilon^k \to 0$

Exact Penalty Function [Grippo-Palagi-P., 2009]

$$\min_v P_\epsilon(V) = L(V, \lambda(V)) + \frac{1}{\epsilon} \sum_i \left(\langle E_{ii}, VV^T \rangle - 1 \right)^2$$

where $\lambda(V)$ is the closed-form expression and $\epsilon > 0$ is suff. small

Unconstrained formulations for (NLP$_r$)

Augmented Lagrangian [Burer-Monteiro, 2003]

$$\min_V L_a(V, \lambda^k, \epsilon^k) = L(V, \lambda^k) + \frac{1}{\epsilon^k} \sum_i^n \left(\langle E_{ii}, VV^T \rangle - 1 \right)^2$$

for a sequence $\{\lambda^k, \epsilon^k\}$, $\epsilon^k \to 0$

Exact Penalty Function [Grippo-Palagi-P., 2009]

$$\min_v P_\epsilon(V) = L(V, \lambda(V)) + \frac{1}{\epsilon} \sum_i \left(\langle E_{ii}, VV^T \rangle - 1 \right)^2$$

where $\lambda(V)$ is the closed-form expression and $\epsilon > 0$ is suff. small

Quotient-Type Function: using the change of variables $X_{ij} = (v_i/\|v_i\|)^T v_j/\|v_j\|$, $v_i \in \mathbb{R}^r, i = 1, \ldots, n$ (note :$V = (v_1 \ \ldots \ v_n)^T$)

$$\min_v \sum_{i,\ j} q_{ij} \frac{v_i^T v_j}{\|v_i\| \|v_j\|}$$

Quotient function

[Burer-Monteiro, 2003]

$$\min_{v} q_r(v) = \sum_{i,\ j} q_{ij} \frac{v_i^T v_j}{\|v_i\| \|v_j\|}$$

Quotient function

[Burer-Monteiro, 2003]

$$\min_{v} q_r(v) = \sum_{i,\ j} q_{ij} \frac{v_i^T v_j}{\|v_i\| \|v_j\|}$$

◇ Remark: proposed originally in [Homer-Peinado, 1997] for $r = n$

Quotient function

[Burer-Monteiro, 2003]

$$\min_{v} q_r(v) = \sum_{i,\ j} q_{ij} \frac{v_i^T v_j}{\|v_i\| \|v_j\|}$$

- ◇ Remark: proposed originally in [Homer-Peinado, 1997] for $r = n$
- ◇ Very efficient in practice

Quotient function

[Burer-Monteiro, 2003]

$$\min_{v} q_r(v) = \sum_{i,\ j} q_{ij} \frac{v_i^T v_j}{\|v_i\| \|v_j\|}$$

- ◇ Remark: proposed originally in [Homer-Peinado, 1997] for $r = n$
- ◇ Very efficient in practice
- ◇ Some drawbacks in using this formulation

Quotient function

[Burer-Monteiro, 2003]

$$\min_{v} q_r(v) = \sum_{i,\ j} q_{ij} \frac{v_i^T v_j}{\|v_i\| \|v_j\|}$$

- ◇ Remark: proposed originally in [Homer-Peinado, 1997] for $r = n$
- ◇ Very efficient in practice
- ◇ Some drawbacks in using this formulation

1. No proof on the correspondence of stationary points/local minima/global minima with stationary points/local minima/global minima of (NLP)$_r$

Quotient function

[Burer-Monteiro, 2003]

$$\min_{v} q_r(v) = \sum_{i,\ j} q_{ij} \frac{v_i^T v_j}{\|v_i\| \|v_j\|}$$

- ◇ Remark: proposed originally in [Homer-Peinado, 1997] for $r = n$
- ◇ Very efficient in practice
- ◇ Some drawbacks in using this formulation
  1. No proof on the correspondence of stationary points/local minima/global minima with stationary points/local minima/global minima of (NLP)$_r$
  2. not defined if any $v_i = 0$

Quotient function

[Burer-Monteiro, 2003]

$$\min_{v} q_r(v) = \sum_{i,\ j} q_{ij} \frac{v_i^T v_j}{\|v_i\| \|v_j\|}$$

- ◇ Remark: proposed originally in [Homer-Peinado, 1997] for $r = n$
- ◇ Very efficient in practice
- ◇ Some drawbacks in using this formulation
  1. No proof on the correspondence of stationary points/local minima/global minima with stationary points/local minima/global minima of (NLP)$_r$
  2. not defined if any $v_i = 0$
  3. unbounded level sets of $q_r(v)$

Quotient function II

We solved the first two drawbacks:

Quotient function II

We solved the first two drawbacks:

1. We proved that given any stationary points/local minima/global minima $v = (v_1^T, \ldots, v_n^T)^T$ of the quotient function, the corresponding point $v = (v_1^T/\|v_1\|, \ldots, v_n^T/\|v_n\|)^T$ is a stationary point/local minimum/global minimum of (NLP)$_r$

Quotient function II

We solved the first two drawbacks:

1. We proved that given any stationary points/local minima/global minima $v = (v_1^T, \ldots, v_n^T)^T$ of the quotient function, the corresponding point $v = (v_1^T/\|v_1\|, \ldots, v_n^T/\|v_n\|)^T$ is a stationary point/local minimum/global minimum of (NLP)$_r$

2. The quotient function is not defined if any $v_i = 0$, but any gradient type method ensures v_i^k away from 0

Quotient function II

We solved the first two drawbacks:

1. We proved that given any stationary points/local minima/global minima $v = (v_1^T, \ldots, v_n^T)^T$ of the quotient function, the corresponding point $v = (v_1^T/\|v_1\|, \ldots, v_n^T/\|v_n\|)^T$ is a stationary point/local minimum/global minimum of (NLP)$_r$
2. The quotient function is not defined if any $v_i = 0$, but any gradient type method ensures v_i^k away from 0
3. However, it has unbounded level sets, so that standard convergence theory does not apply

A new unconstrained formulation of problem (NLP$_r$)

We propose a modification of function $q_r(v)$:

$$f_\epsilon(v) := \sum_{i,\ j} q_{ij} \frac{v_i^T v_j}{\|v_i\|\|v_j\|} + \frac{1}{\epsilon} \sum_{i=1}^{n} \frac{(\|v_i\|^2 - 1)^2}{d(v_i)},$$

where $\varepsilon > 0$ and

$$d(v_i) := \delta^2 - \left(1 - \|v_i\|^2\right)_+^2, \quad 0 < \delta < 1.$$

A new unconstrained formulation of problem (NLP$_r$)

We propose a modification of function $q_r(v)$:

$$f_\epsilon(v) := \sum_{i,\ j} q_{ij} \frac{v_i^T v_j}{\|v_i\|\|v_j\|} + \frac{1}{\epsilon} \sum_{i=1}^{n} \frac{(\|v_i\|^2 - 1)^2}{d(v_i)},$$

where $\varepsilon > 0$ and

$$d(v_i) := \delta^2 - \left(1 - \|v_i\|^2\right)_+^2, \quad 0 < \delta < 1.$$

For a fixed $\varepsilon > 0$ we consider the unconstrained minimization problem

$$\min_{v \in S_\delta} \ f_\epsilon(v), \tag{RQ$_r$}$$

where the open set S_δ is defined as

$$S_\delta := \{v \in \mathbb{R}^{nr} \ : \ \|v_i\|^2 > 1 - \delta, \quad i = 1, \ldots, n\}.$$

Properties of problem ($\mathrm{RQ_r}$)

$$\begin{array}{l} \min f_\epsilon(v), \\ v \in S_\delta \\ v_i \in \Re^r \; i = 1, \ldots, n \end{array} \qquad (\mathrm{RQ_r}) \qquad \begin{array}{l} \min \langle Q, VV^T \rangle \\ \langle E_{ii}, VV^T \rangle = 1, i = 1, \ldots, n \\ V \in \mathbb{R}^{n \times r} \end{array} \qquad (\mathrm{NLP_r})$$

1. For any $v_0 \in S_\delta$, function $f_\epsilon(v)$ has compact level sets

Properties of problem ($\mathrm{RQ_r}$)

$$\begin{array}{l} \min f_\epsilon(v), \\ v \in S_\delta \\ v_i \in \Re^r \; i = 1, \ldots, n \end{array} \qquad (\mathrm{RQ_r}) \qquad \begin{array}{l} \min \langle Q, VV^T \rangle \\ \langle E_{ii}, VV^T \rangle = 1, i = 1, \ldots, n \\ V \in \mathbb{R}^{n \times r} \end{array} \qquad (\mathrm{NLP_r})$$

1. For any $v_0 \in S_\delta$, function $f_\epsilon(v)$ has compact level sets
2. For **any** $\epsilon > 0$:

Properties of problem ($\mathrm{RQ_r}$)

$$\begin{array}{l} \min f_\epsilon(v), \\ v \in S_\delta \\ v_i \in \Re^r \; i = 1, \ldots, n \end{array} \qquad (\mathrm{RQ_r}) \qquad \begin{array}{l} \min \langle Q, VV^T \rangle \\ \langle E_{ii}, VV^T \rangle = 1, i = 1, \ldots, n \\ V \in \mathbb{R}^{n \times r} \end{array} \qquad (\mathrm{NLP_r})$$

1. For any $v_0 \in S_\delta$, function $f_\epsilon(v)$ has compact level sets
2. For **any** $\epsilon > 0$:
   - ◇ stationary point of ($\mathrm{RQ_r}$) $\Leftrightarrow$ stationary point of ($\mathrm{NLP_r}$)
   - ◇ local minimum point of ($\mathrm{RQ_r}$) $\Leftrightarrow$ local minimum point of ($\mathrm{NLP_r}$)
   - ◇ global minimum point of ($\mathrm{RQ_r}$) $\Leftrightarrow$ global minimum point of ($\mathrm{NLP_r}$)

Properties of problem ($\mathrm{RQ_r}$)

$$\begin{array}{l} \min f_\epsilon(v), \\ v \in S_\delta \\ v_i \in \Re^r \; i = 1, \ldots, n \end{array} \qquad (\mathrm{RQ_r}) \qquad \begin{array}{l} \min \langle Q, VV^T \rangle \\ \langle E_{ii}, VV^T \rangle = 1, i = 1, \ldots, n \\ V \in \mathbb{R}^{n \times r} \end{array} \qquad (\mathrm{NLP_r})$$

1. For any $v_0 \in S_\delta$, function $f_\epsilon(v)$ has compact level sets
2. For **any** $\epsilon > 0$:
   - ◇ stationary point of ($\mathrm{RQ_r}$) $\Leftrightarrow$ stationary point of ($\mathrm{NLP_r}$)
   - ◇ local minimum point of ($\mathrm{RQ_r}$) $\Leftrightarrow$ local minimum point of ($\mathrm{NLP_r}$)
   - ◇ global minimum point of ($\mathrm{RQ_r}$) $\Leftrightarrow$ global minimum point of ($\mathrm{NLP_r}$)

Problem (NLP_r) can be solved by solving problem ($\mathrm{RQ_r}$) by any standard minimization algorithm

A specific algorithm for problem ($\mathrm{RQ_r}$) I

$$\begin{array}{ll} \min & \sum_{i,\ j} q_{ij} \frac{v_i^T v_j}{\|v_i\|\|v_j\|} + \frac{1}{\epsilon} \sum_{i=1}^{n} \frac{(\|v_i\|^2-1)^2}{d(v_i)} \\ & v \in S_\delta \\ & v_i \in \Re^r \ i = 1, \ldots, n \end{array} \qquad (\mathrm{RQ_r})$$

The barrier term $d(v_i) := \delta^2 - \left(1 - \|v_i\|^2\right)_+^2$, $0 < \delta < 1$ plays a key role to make standard optimization method be globally convergent for problem (RQ_r).

A specific algorithm for problem ($\mathrm{RQ_r}$) I

$$\begin{array}{ll} \min & \sum_{i,\ j} q_{ij} \frac{v_i^T v_j}{\|v_i\|\|v_j\|} + \frac{1}{\epsilon} \sum_{i=1}^{n} \frac{(\|v_i\|^2-1)^2}{d(v_i)} \\ & v \in S_\delta \\ & v_i \in \Re^r \ i = 1, \ldots, n \end{array} \qquad (\mathrm{RQ_r})$$

The barrier term $d(v_i) := \delta^2 - \left(1 - \|v_i\|^2\right)_+^2$, $0 < \delta < 1$ plays a key role to make standard optimization method be globally convergent for problem (RQ_r).

Nevertheless, a barrier term affects negatively the performance behavior of any optimization method, especially when the produced sequence gets closer to the boundary of S_δ.

A specific algorithm for problem ($\mathrm{RQ_r}$) II

$$\begin{array}{ll} \min & \sum_{i,\ j} q_{ij} \frac{v_i^T v_j}{\|v_i\|\|v_j\|} + \frac{1}{\epsilon} \sum_{i=1}^{n} \frac{(\|v_i\|^2-1)^2}{d(v_i)} \\ & v \in S_\delta \\ & v_i \in \Re^r \ i = 1, \ldots, n \end{array} \qquad (\mathrm{RQ_r})$$

Consider a gradient type iteration, starting from v_0 such that $\|v_i^0\| = 1$ for $i = 1, \ldots, n$:

$$v_i^{k+1} = v_i^k - \alpha^k \nabla_{v_i} f_\varepsilon(v^k) \quad i = 1, \ldots, n$$

where $\alpha^k > 0$ is obtained by a suitable linesearch procedure satisfying at least

$$f_\varepsilon(v^{k+1}) \leq f_\varepsilon(v^0).$$

A specific algorithm for problem ($\mathrm{RQ_r}$) II

$$\begin{array}{ll} \min & \sum_{i,\ j} q_{ij} \frac{v_i^T v_j}{\|v_i\|\|v_j\|} + \frac{1}{\epsilon} \sum_{i=1}^{n} \frac{(\|v_i\|^2-1)^2}{d(v_i)} \\ & v \in S_\delta \\ & v_i \in \Re^r \ i = 1, \ldots, n \end{array} \qquad (\mathrm{RQ_r})$$

Consider a gradient type iteration, starting from v_0 such that $\|v_i^0\| = 1$ for $i = 1, \ldots, n$:

$$v_i^{k+1} = v_i^k - \alpha^k \nabla_{v_i} f_\varepsilon(v^k) \quad i = 1, \ldots, n$$

where $\alpha^k > 0$ is obtained by a suitable linesearch procedure satisfying at least

$$f_\varepsilon(v^{k+1}) \leq f_\varepsilon(v^0).$$

Then, there exists $\bar{\epsilon} > 0$ such that, for any $\epsilon \geq \bar{\epsilon}$, we have for $k = 1, 2, \ldots$

$$\|v_i^k\| \geq 1, \quad i = 1, \ldots, n.$$

and hence $d(v_i) = \delta^2$

Convergence result

Let $\{v^k\}$ generated with a Gradient Method(Non-Monotone), starting from v_0 such that $\|v_i^0\| = 1$ for $i = 1, \ldots, n$. Then, for $\epsilon \geq \bar{\epsilon}$

1. $1 \leq \|v_i^k\|^2 \leq M$ with $i = 1, \ldots, n$

Convergence result

Let $\{v^k\}$ generated with a Gradient Method(Non-Monotone), starting from v_0 such that $\|v_i^0\| = 1$ for $i = 1, \ldots, n$. Then, for $\epsilon \geq \bar{\epsilon}$

1. $1 \leq \|v_i^k\|^2 \leq M$ with $i = 1, \ldots, n$
2. $\{v^k\} \to \hat{v}$, with $\hat{v}$ stationary point of (RQ$_\mathrm{r}$), and hence of (NLP)$_r$

Convergence result

Let $\{v^k\}$ generated with a Gradient Method(Non-Monotone), starting from v_0 such that $\|v_i^0\| = 1$ for $i = 1, \ldots, n$. Then, for $\epsilon \geq \bar{\epsilon}$

1. $1 \leq \|v_i^k\|^2 \leq M$ with $i = 1, \ldots, n$
2. $\{v^k\} \rightarrow \hat{v}$, with $\hat{v}$ stationary point of (RQ$_\mathrm{r}$), and hence of (NLP)$_r$

We use a a Fortran 90 implementation of the non monotone Barzilai-Borwein gradient method proposed in [Grippo, Sciandrone, COAP2002]

SpeeDP

Data. $Q \in \mathcal{S}^n$.

SpeeDP

Data. $Q \in \mathcal{S}^n$.

Inizialization. Compute $\widehat{r} = \left\lfloor \frac{\sqrt{1+8n}-1}{2} \right\rfloor$. Set integers $2 \leq r^1 < r^2 < \cdots < r^p$ with $r^p \in [\widehat{r}, n]$.

SpeeDP

Data. $Q \in \mathcal{S}^n$.

Inizialization. Compute $\widehat{r} = \left\lfloor \frac{\sqrt{1+8n}-1}{2} \right\rfloor$. Set integers $2 \leq r^1 < r^2 < \cdots < r^p$ with $r^p \in [\widehat{r}, n]$.

Step 1 Choose $\varepsilon \geq \bar{\varepsilon}$.

SpeeDP

Data. $Q \in \mathcal{S}^n$.

Inizialization. Compute $\widehat{r} = \left\lfloor \frac{\sqrt{1+8n}-1}{2} \right\rfloor$. Set integers $2 \leq r^1 < r^2 < \cdots < r^p$ with $r^p \in [\widehat{r}, n]$.

Step 1 Choose $\varepsilon \geq \bar{\varepsilon}$.

Step 2 Find a stationary point $\hat{v} \in \mathbb{R}^{nr^j}$ of problem (NLP$_{\mathbf{r}^j}$) by applying the Non-monotone gradient method to problem (RQ$_{r^j}$) starting from a point $v^0 \in \mathbb{R}^{nr^j}$ with $\|v_i^0\|^2 = 1$ for $i = 1, \ldots, n$.

SpeeDP

Data. $Q \in \mathcal{S}^n$.

Inizialization. Compute $\widehat{r} = \left\lfloor \frac{\sqrt{1+8n}-1}{2} \right\rfloor$. Set integers $2 \leq r^1 < r^2 < \cdots < r^p$ with $r^p \in [\widehat{r}, n]$.

Step 1 Choose $\varepsilon \geq \bar{\varepsilon}$.

Step 2 Find a stationary point $\hat{v} \in \mathbb{R}^{nr^j}$ of problem (NLP$_{r^j}$) by applying the Non-monotone gradient method to problem (RQ$_{r^j}$) starting from a point $v^0 \in \mathbb{R}^{nr^j}$ with $\|v_i^0\|^2 = 1$ for $i = 1, \ldots, n$.

Step 3 Compute the minimum eigenvalue $\mu_{\min}(\hat{v})$ of $Q + \mathrm{Diag}(\lambda(\hat{v}))$. If $\mu_{\min}(\hat{v}) = 0$ or $j = p$, then stop and return $\hat{v} \in \mathbb{R}^{nr^j}$ and $\mu_{\min}(\hat{v})$, otherwise set $j = j + 1$, and go to Step 1.

Some Remarks

- If $r = n$ problem (NLP$_r$) has no local minimum points [M. Journee, F. Bach, P.A. Absil, R. Sepulchre 2008]

Some Remarks

- If $r = n$ problem (NLP$_r$) has no local minimum points [M. Journee, F. Bach, P.A. Absil, R. Sepulchre 2008]

- In principle the algorithm may fail (also for $r = n$), since the problem we are solving is a nonconvex one, and the algorithm produces only a stationary point.

Some Remarks

- If $r = n$ problem (NLP$_r$) has no local minimum points [M. Journee, F. Bach, P.A. Absil, R. Sepulchre 2008]

- In principle the algorithm may fail (also for $r = n$), since the problem we are solving is a nonconvex one, and the algorithm produces only a stationary point.

- In practice, it never happens.

Some Remarks

- If $r = n$ problem (NLP$_r$) has no local minimum points [M. Journee, F. Bach, P.A. Absil, R. Sepulchre 2008]

- In principle the algorithm may fail (also for $r = n$), since the problem we are solving is a nonconvex one, and the algorithm produces only a stationary point.

- In practice, it never happens.

- Given any $\lambda \in \Re^n$ the point $u = \lambda + \lambda_{\min}(Q + \Lambda)e$ is feasible for problem (DUAL$_{\mathrm{MC}}$), and hence $e^T u = e^T \lambda + n\lambda_{\min}(Q + \Lambda)$ is a bound on the optimal value of (SDP$_{\mathrm{MC}}$).

Implementation

We select the value r^1 according to an heuristic based on n

n	r^1	$\widehat{r}$
≤ 200	8	≤ 19
$\geq 250, < 800$	10	< 39
$\geq 800, < 1000$	15	< 44
$\geq 1000, \leq 5000$	18	< 99
$> 5000, \leq 20000$	25	≥ 99
≥ 20000	30	≥ 199

Implementation

We select the value r^1 according to an heuristic based on n

n	r^1	$\widehat{r}$
≤ 200	8	≤ 19
$\geq 250, < 800$	10	< 39
$\geq 800, < 1000$	15	< 44
$\geq 1000, \leq 5000$	18	< 99
$> 5000, \leq 20000$	25	≥ 99
≥ 20000	30	≥ 199

we set $r_{i+1} = \min\{1.5 * r_i, \widehat{r}\}$

Implementation

We select the value r^1 according to an heuristic based on n

n	r^1	$\widehat{r}$
≤ 200	8	≤ 19
$\geq 250, < 800$	10	< 39
$\geq 800, < 1000$	15	< 44
$\geq 1000, \leq 5000$	18	< 99
$> 5000, \leq 20000$	25	≥ 99
≥ 20000	30	≥ 199

we set $r_{i+1} = \min\{1.5 * r_i, \widehat{r}\}$

we set $\varepsilon = 10^3$

Implementation

We select the value r^1 according to an heuristic based on n

n	r^1	$\widehat{r}$
≤ 200	8	≤ 19
$\geq 250, < 800$	10	< 39
$\geq 800, < 1000$	15	< 44
$\geq 1000, \leq 5000$	18	< 99
$> 5000, \leq 20000$	25	≥ 99
≥ 20000	30	≥ 199

we set $r_{i+1} = \min\{1.5 * r_i, \widehat{r}\}$

we set $\varepsilon = 10^3$

we used the ARPACK subroutines for computing $\lambda_{\min}(Q + \bar{\Lambda})$

Problems and softwares

We compare with

1. SDPLR-MC

   [Burer, Monteiro]

Problems and softwares

We compare with

1. SDPLR-MC

   [Burer, Monteiro]

2. DSDP[Benson, Ye, Zhang]

Problems and softwares

We compare with

1. SDPLR-MC

   [Burer, Monteiro]

2. DSDP[Benson, Ye, Zhang]

3. EXPA

   [Grippo, Palagi, P.]

Problems and softwares

We compare with

1. SDPLR-MC

   [Burer, Monteiro]

2. DSDP[Benson, Ye, Zhang]

3. EXPA

   [Grippo, Palagi, P.]

4. SB: spectral bundle method [Helmberg, Rendl]

Problems and softwares

We compare with

1. SDPLR-MC

   [Burer, Monteiro]

2. DSDP[Benson, Ye, Zhang]

3. EXPA

   [Grippo, Palagi, P.]

4. SB: spectral bundle method [Helmberg, Rendl]

- HEURISTIC: same idea as SpeeDP, quotient unconstrained formulation solved by L-BFGS; suitable for large scale pb.

Problems and softwares

We compare with

1. SDPLR-MC

   [Burer, Monteiro]

   - HEURISTIC: same idea as SpeeDP, quotient unconstrained formulation solved by L-BFGS; suitable for large scale pb.

2. DSDP[Benson, Ye, Zhang]

   - EXACT: dual interior point; up to $n = 7000$ (25 pb.)

3. EXPA

   [Grippo, Palagi, P.]

4. SB: spectral bundle method [Helmberg, Rendl]

Problems and softwares

We compare with

1. SDPLR-MC

   [Burer, Monteiro]

   - HEURISTIC: same idea as SpeeDP, quotient unconstrained formulation solved by L-BFGS; suitable for large scale pb.

2. DSDP[Benson, Ye, Zhang]

   - EXACT: dual interior point; up to $n = 7000$ (25 pb.)

3. EXPA

   [Grippo, Palagi, P.]

   - same algorithmic idea of SpeeDP but using an exact penalty function (state of the art)

4. SB: spectral bundle method [Helmberg, Rendl]

Test Problems

We used three test sets for max cut problems:

Test Problems

We used three test sets for max cut problems:

1. SDPLIB (18 pb. with $100 \leq n \leq 7000$, and number of edges ranging from 150 to 17000)

Test Problems

We used three test sets for max cut problems:

1. SDPLIB (18 pb. with $100 \leq n \leq 7000$, and number of edges ranging from 150 to 17000)

2. DIMACS (4 pb. two have 512 nodes with 1,536 edges and two have 3,375 nodes with 10,125 edges)

Test Problems

We used three test sets for max cut problems:

1. SDPLIB (18 pb. with $100 \leq n \leq 7000$, and number of edges ranging from 150 to 17000)

2. DIMACS (4 pb. two have 512 nodes with 1,536 edges and two have 3,375 nodes with 10,125 edges)

3. Gset: randomly generated by a machine-independent graph generator *rudy* (19 pb. with $800 \leq n \leq 20000$ and number of edges ranging from about 5,000 to 40,000)

Performance profile[DolanMorè02]

We have a set of solvers S and a set of problems P. Let $p \in P$ denote a particular problem and $s \in S$ a particular solver.

Performance profile[DolanMorè02]

We have a set of solvers S and a set of problems P. Let $p \in P$ denote a particular problem and $s \in S$ a particular solver.

The idea is to compare the performance of solver s on problem p with the best performance by any solver on this particular problem by the performance ratio

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s'} : s' \in S\}},$$

where $t_{p,s}$ is the CPU time in seconds needed by solver s to solve problem p.

Performance profile[DolanMorè02]

We have a set of solvers S and a set of problems P. Let $p \in P$ denote a particular problem and $s \in S$ a particular solver.

The idea is to compare the performance of solver s on problem p with the best performance by any solver on this particular problem by the performance ratio

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s'} : s' \in S\}},$$

where $t_{p,s}$ is the CPU time in seconds needed by solver s to solve problem p.

A cumulative distribution function $\rho_s(\tau)$ is defined as:

$$\rho_s(\tau) = \frac{1}{n_p}\text{size}\{p \in P : r_{p,s} \leq \tau\}.$$

Performance profile[DolanMorè02]

We have a set of solvers S and a set of problems P. Let $p \in P$ denote a particular problem and $s \in S$ a particular solver.

The idea is to compare the performance of solver s on problem p with the best performance by any solver on this particular problem by the performance ratio

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s'} : s' \in S\}},$$

where $t_{p,s}$ is the CPU time in seconds needed by solver s to solve problem p.

A cumulative distribution function $\rho_s(\tau)$ is defined as:

$$\rho_s(\tau) = \frac{1}{n_p}\text{size}\{p \in P : r_{p,s} \leq \tau\}.$$

We draw $\rho_s(\tau)$ with respect to τ, that is reported on the x-axis in a logarithmic scale.

Performance profile[DolanMorè02]

We have a set of solvers S and a set of problems P. Let $p \in P$ denote a particular problem and $s \in S$ a particular solver.

The idea is to compare the performance of solver s on problem p with the best performance by any solver on this particular problem by the performance ratio

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s'} : s' \in S\}},$$

where $t_{p,s}$ is the CPU time in seconds needed by solver s to solve problem p.

A cumulative distribution function $\rho_s(\tau)$ is defined as:

$$\rho_s(\tau) = \frac{1}{n_p}\text{size}\{p \in P : r_{p,s} \leq \tau\}.$$

We draw $\rho_s(\tau)$ with respect to τ, that is reported on the x-axis in a logarithmic scale.

The higher the method the better, and the efficiency is measured by how fast the method reaches the value of 1 (since all the methods solve all the problems, all the methods reach the performance value 1 allowing a sufficiently large τ).

Comparison with other codes: cpu time

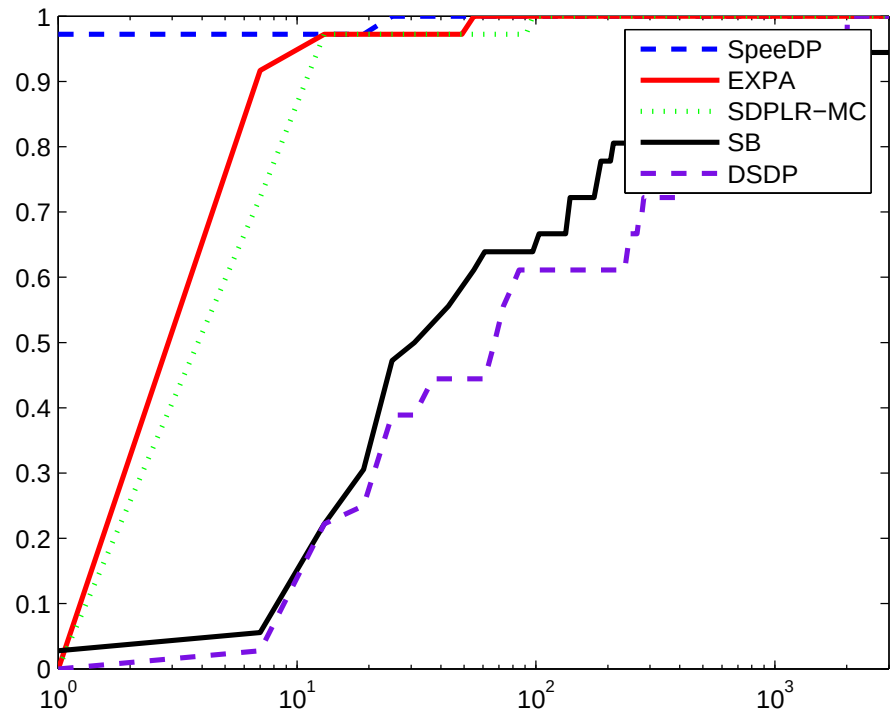


Figure 1: Comparison between NLP based methods, SB and DSDP on the problems up to 10000 variables

Comparison with LR: cpu time

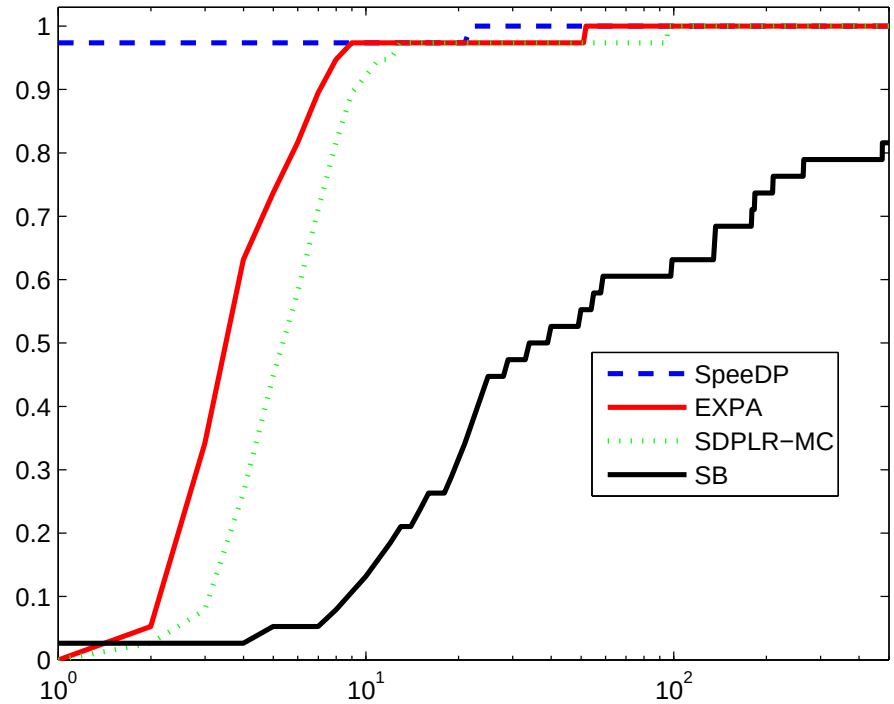


Figure 2: Comparison among the NLP based methods, and SB

Comparison with DSDP: accuracy

- SpeeDP finds perfect dual feasibility on 20 problems on a total of 38 problems

Comparison with DSDP: accuracy

- SpeeDP finds perfect dual feasibility on 20 problems on a total of 38 problems
- On the remaining, SpeeDP is more accurate than DSDP on 8 problems (comparing the dual objective function)

Comparison with DSDP: accuracy

- SpeeDP finds perfect dual feasibility on 20 problems on a total of 38 problems
- On the remaining, SpeeDP is more accurate than DSDP on 8 problems (comparing the dual objective function)
- On the 10 problems where SpeeDP is less accurate the difference is less than 10^{-4}

Some considerations

- SpeeDP is very efficient: it is much faster than interior point methods, without losing in accuracy

Some considerations

- SpeeDP is very efficient: it is much faster than interior point methods, without losing in accuracy
- SpeeDP exploit sparsity of the original problem, we never need the complete matrix Q

Some considerations

- SpeeDP is very efficient: it is much faster than interior point methods, without losing in accuracy
- SpeeDP exploit sparsity of the original problem, we never need the complete matrix Q
- In output of SpeeDP we have:

Some considerations

- SpeeDP is very efficient: it is much faster than interior point methods, without losing in accuracy
- SpeeDP exploit sparsity of the original problem, we never need the complete matrix Q
- In output of SpeeDP we have:
  1. The vector v containing already the factorization $X = VV^T$ (for free)

Some considerations

- SpeeDP is very efficient: it is much faster than interior point methods, without losing in accuracy
- SpeeDP exploit sparsity of the original problem, we never need the complete matrix Q
- In output of SpeeDP we have:
  1. The vector v containing already the factorization $X = VV^T$ (for free)
  2. The multiplier λ associated to the vector v
  3. Perfect primal feasibility, since the output of the method are $v_i / \|v_i\|$

Some considerations

- SpeeDP is very efficient: it is much faster than interior point methods, without losing in accuracy
- SpeeDP exploit sparsity of the original problem, we never need the complete matrix Q
- In output of SpeeDP we have:
  1. The vector v containing already the factorization $X = VV^T$ (for free)
  2. The multiplier λ associated to the vector v
  3. Perfect primal feasibility, since the output of the method are $v_i/\|v_i\|$
  4. The quantity $e^T\lambda + n\lambda_{\min}(Q+\Lambda)$ (and we have computed $\lambda_{\min}(Q+\Lambda)$ to check the optimality condition) is a valid bound for the original SDP relaxation

GW algorithm

- The basic SDP relaxation of Max Cut is at the basis of the randomized algorithm proposed by Goemans and Williamson [Goemans and Williamson, 95] .

- Given the solution of

$$\begin{aligned} \min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij} v_i^T v_j \\ & \|v_i\|^2 = 1, i = 1, \ldots, n \\ & v_i \in \Re^n \end{aligned}$$

  and given a random vector h uniformly distributed on the unit sphere, set $S = \{i : \ v_i^T h \geq 0\}$

GW algorithm

- The basic SDP relaxation of Max Cut is at the basis of the randomized algorithm proposed by Goemans and Williamson [Goemans and Williamson, 95] .

- Given the solution of

$$\begin{aligned} \min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij} v_i^T v_j \\ & \|v_i\|^2 = 1, i = 1, \ldots, n \\ & v_i \in \Re^n \end{aligned}$$

  and given a random vector h uniformly distributed on the unit sphere, set $S = \{i : \ v_i^T h \geq 0\}$

- If all the weights of the graph are nonnegative, the expected value of the produced cut is guaranteed to be greater than 0.87856 times the optimal value of max cut

GW algorithm

- The basic SDP relaxation of Max Cut is at the basis of the randomized algorithm proposed by Goemans and Williamson [Goemans and Williamson, 95] .
- Given the solution of

$$\begin{aligned} \min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij} v_i^T v_j \\ & \|v_i\|^2 = 1, i = 1, \ldots, n \\ & v_i \in \Re^n \end{aligned}$$

  and given a random vector h uniformly distributed on the unit sphere, set $S = \{i : \ v_i^T h \geq 0\}$
- If all the weights of the graph are nonnegative, the expected value of the produced cut is guaranteed to be greater than 0.87856 times the optimal value of max cut
- This heuristic is very used in practice, also as starting point for other heuristics (for example as in BiqMac).

A fast heuristic for large graphs I

A fast heuristic for large graphs I

- However, this approach is not practical when the graph gets too large since the SDP relaxation is too expansive to solve with interior point methods

A fast heuristic for large graphs I

- However, this approach is not practical when the graph gets too large since the SDP relaxation is too expansive to solve with interior point methods
- SpeeDP is able to solve in a short amount of time the basic SDP relaxation for larger graphs (impossible with IP methods for graphs larger than some thousand of nodes)

A fast heuristic for large graphs I

- However, this approach is not practical when the graph gets too large since the SDP relaxation is too expansive to solve with interior point methods
- SpeeDP is able to solve in a short amount of time the basic SDP relaxation for larger graphs (impossible with IP methods for graphs larger than some thousand of nodes)
- It gets for free the vectors v_i needed by the GW algorithm

A fast heuristic for large graphs I

- However, this approach is not practical when the graph gets too large since the SDP relaxation is too expansive to solve with interior point methods
- SpeeDP is able to solve in a short amount of time the basic SDP relaxation for larger graphs (impossible with IP methods for graphs larger than some thousand of nodes)
- It gets for free the vectors v_i needed by the GW algorithm
- The time can be even reduced by tuning the accuracy: we can solve the problem (NLP_r) for a small r and then get the bound $e^T\lambda + \lambda_{\min}(Q + \Lambda)$ (that again we have for free)

A fast heuristic for large graphs II

Consider again problem

$$\begin{array}{ll} \min & \langle Q, X \rangle \\ & \mathrm{diag}(X) = e \\ & X \succeq 0, X \in \mathcal{S}^n \end{array} \qquad (\mathrm{SDP_{MC}})$$

A fast heuristic for large graphs II

Consider again problem

$$\begin{array}{ll} \min & \langle Q, X \rangle \\ & \mathrm{diag}(X) = e \\ & X \succeq 0, X \in \mathcal{S}^n \end{array} \qquad (\mathrm{SDP_{MC}})$$

We solve the problem by SpeeDP, and apply the GW algorithm.

A fast heuristic for large graphs II

Consider again problem

$$\begin{array}{ll} \min & \langle Q, X \rangle \\ & \mathrm{diag}(X) = e \\ & X \succeq 0, X \in \mathcal{S}^n \end{array} \qquad (\mathrm{SDP_{MC}})$$

We solve the problem by SpeeDP, and apply the GW algorithm.

Then, a 1-opt heuristic is applied to improve the cut.

A fast heuristic for large graphs II

Consider again problem

$$\begin{array}{ll} \min & \langle Q, X \rangle \\ & \operatorname{diag}(X) = e \\ & X \succeq 0, X \in \mathcal{S}^n \end{array} \qquad (\mathrm{SDP_{MC}})$$

We solve the problem by SpeeDP, and apply the GW algorithm.

Then, a 1-opt heuristic is applied to improve the cut.

In [FischerGruberRebndlSotirov06] the GW+improvement is repeated for different $X' = \alpha X + (1-\alpha)\hat{x}\hat{x}^T$, $0 < \alpha < 1$, where $\hat{x}$ is the representative vector of the current best cut

A fast heuristic for large graphs II

Consider again problem

$$\begin{array}{ll} \min & \langle Q, X \rangle \\ & \operatorname{diag}(X) = e \\ & X \succeq 0, X \in \mathcal{S}^n \end{array} \qquad (\mathrm{SDP_{MC}})$$

We solve the problem by SpeeDP, and apply the GW algorithm.

Then, a 1-opt heuristic is applied to improve the cut.

In [FischerGruberRebndlSotirov06] the GW+improvement is repeated for different $X' = \alpha X + (1-\alpha)\hat{x}\hat{x}^T$, $0 < \alpha < 1$, where $\hat{x}$ is the representative vector of the current best cut

Idea: to bias the GW rounding with the current best cut.

A fast heuristic for large graphs II

Consider again problem

$$\begin{array}{ll} \min & \langle Q, X \rangle \\ & \operatorname{diag}(X) = e \\ & X \succeq 0, X \in \mathcal{S}^n \end{array} \qquad (\mathrm{SDP_{MC}})$$

We solve the problem by SpeeDP, and apply the GW algorithm.

Then, a 1-opt heuristic is applied to improve the cut.

In [FischerGruberRebndlSotirov06] the GW+improvement is repeated for different $X' = \alpha X + (1-\alpha)\hat{x}\hat{x}^T$, $0 < \alpha < 1$, where $\hat{x}$ is the representative vector of the current best cut

Idea: to bias the GW rounding with the current best cut.

However, a factorization of X' is needed $\Rightarrow$ impractical for large graphs

A fast heuristic for large graphs III

Our idea: perturb the objective function of the original problem, and resolve it.

A fast heuristic for large graphs III

Our idea: perturb the objective function of the original problem, and resolve it.

Q is replaced by $Q' = Q + \beta \hat{x}\hat{x}^T$ with $\beta > 0$. Such a perturbation has again the effect of moving the solution of problem $(\mathrm{SDP_{MC}})$ and hence of the Goemans-Williamson rounding, towards a neighborhood of the current best integral solution.

A fast heuristic for large graphs III

Our idea: perturb the objective function of the original problem, and resolve it.

Q is replaced by $Q' = Q + \beta \hat{x}\hat{x}^T$ with $\beta > 0$. Such a perturbation has again the effect of moving the solution of problem $(\mathrm{SDP_{MC}})$ and hence of the Goemans-Williamson rounding, towards a neighborhood of the current best integral solution.

The new problem is solved again by SpeeDP starting by the previous solution (warm start)

A fast heuristic for large graphs III

Our idea: perturb the objective function of the original problem, and resolve it.

Q is replaced by $Q' = Q + \beta \hat{x}\hat{x}^T$ with $\beta > 0$. Such a perturbation has again the effect of moving the solution of problem $(\mathrm{SDP_{MC}})$ and hence of the Goemans-Williamson rounding, towards a neighborhood of the current best integral solution.

The new problem is solved again by SpeeDP starting by the previous solution (warm start)

Then the GW rounding and the 1-opt improvement are repeated as well. The whole procedure is repeated a few times with different values of β.

SpeeDP-MC

ALGORITHM SpeeDP-MC

Data: $Q,\ \hat{x} = e,\ \alpha > 0,\ k_{\max},\ \overline{Q} = \sum_{i,j} |Q_{ij}|/|E|.$

For $k = k_{max}, \ldots, 0$ **do** :

S.0 Set $\beta = k\alpha\overline{Q}$ and $Q' = Q + \beta(\hat{x}\hat{x}^T)$

S.1 Apply SpeeDP to problem $(\mathrm{SDP_{MC}})$ with $Q = Q'$ and let v_i, $i = 1, \ldots, n$ be the returned solution and the valid bound ϕ on the max cut problem with objective function corresponding to Q'.

S.2 Apply the Goemans-Williamson hyperplane rounding technique to the vectors v_i, $i = 1, \ldots, n$. This gives a bipartition representative vector $\bar{x}$.

S.3 Apply the 1-opt improvement to $\bar{x}$. This gives a new bipartition representative vector $\tilde{x}$. If $\langle Q, \tilde{x}\tilde{x}^T\rangle < \langle Q, \hat{x}\hat{x}^T\rangle$, set $\hat{x} = \tilde{x}$.

Return Best cut $\hat{x}$, lower bound $\langle -Q, \hat{x}\hat{x}^T\rangle$, upper bound ϕ.

Some considerations

- The amount of perturbation decreases when the iteration counter increases

Some considerations

- The amount of perturbation decreases when the iteration counter increases
- The minimization step is not expensive, thanks to the warm start technique

Some considerations

- The amount of perturbation decreases when the iteration counter increases
- The minimization step is not expensive, thanks to the warm start technique
- The algorithm provides both a lower and upper bound, giving a performance guarantee

Numerical results

We used the graph generator *rudy* to define instances with growing dimension and density and different weights.

Numerical results

We used the graph generator *rudy* to define instances with growing dimension and density and different weights.

We first considered graphs with number of nodes n equal to $500 + i \cdot 250$, for $i = 0, \ldots, 8$ and with edge $density$ equal to $10\% + i \cdot 10\%$ for $i = 0, \ldots, 9$. For each pair $(n, density)$ we generated three different graphs with positive weights ranging between 1 and 100.

Some numbers

	cut	time	gap %	0.87856*ub
n=500				
	388004.6667	6.216666667	4.595666667	356550.8369
	730297.3333	8.366666667	3.421133333	663560.8102
	1063049	13.4	2.761066667	959739.3467
	1391196.333	18.62666667	2.394766667	1251519.1163
	1714089	19.37666633	1.992666667	1535937.3951
	2032992.333	36.21333233	1.752933333	1817416.2683
	2347809	27.92000033	1.5559	2094784.6522
	2661106	30.53333433	1.295366667	2368226.2755
	2971604.333	44.373333	1.0805	2638941.8736
	3279198.333	47.47666667	0.880966667	2906354.3058
n=1500				
	3220711	60.03666533	3.394633333	2925643.823
	6195972	203.75	2.443333333	5576536.602
	9128485.333	305.3333283	1.932633333	8174914.701
	12020693.33	267.9600067	1.6445	10734573.2
	14899393	338.5299987	1.354266667	13267285.67
	17760397.33	272.16333	2.6834	16022282.32
	20603549.33	347.5099997	2.344366667	18525817.52
	23430844.67	384.0266673	2.016433333	21000489.11
	26236188	341.8233337	1.707533333	23443644.4
	29026448	521.1066593	1.414566667	25862214.06
n=2500				
	8707864.333	326.813334	2.866133333	7869652.293
	16883800	512.99999	2.059233333	15138884.38
	24971508.67	834.2133483	2.199433333	22421516.56
	32984345.33	1475.053324	1.823933333	29507260.52
	40946184	1444.76001	2.447766667	36854243.7
	48873544	1019.210001	1.725333333	43679150.81
	56771676	2992.886719	1.486266667	50618557.71
	64627920	1933.013305	1.558366667	57664353.54
	72453770.67	2257.273397	1.352333333	64515830.64
	80230896	2650.269979	0.905633333	71126099.68

A picture

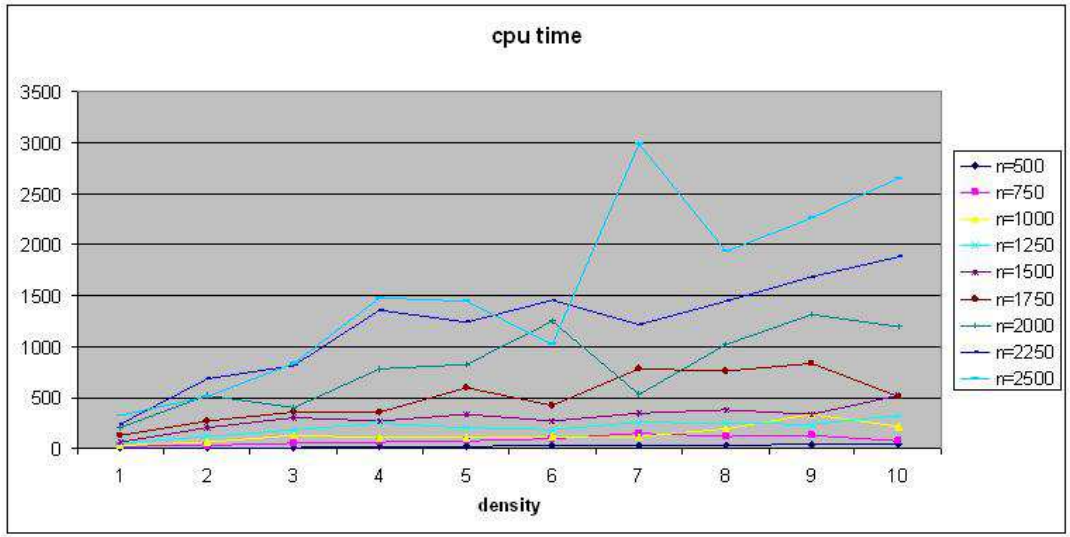


Figure 3: Average CPU time of the heuristic on the random graphs

A picture

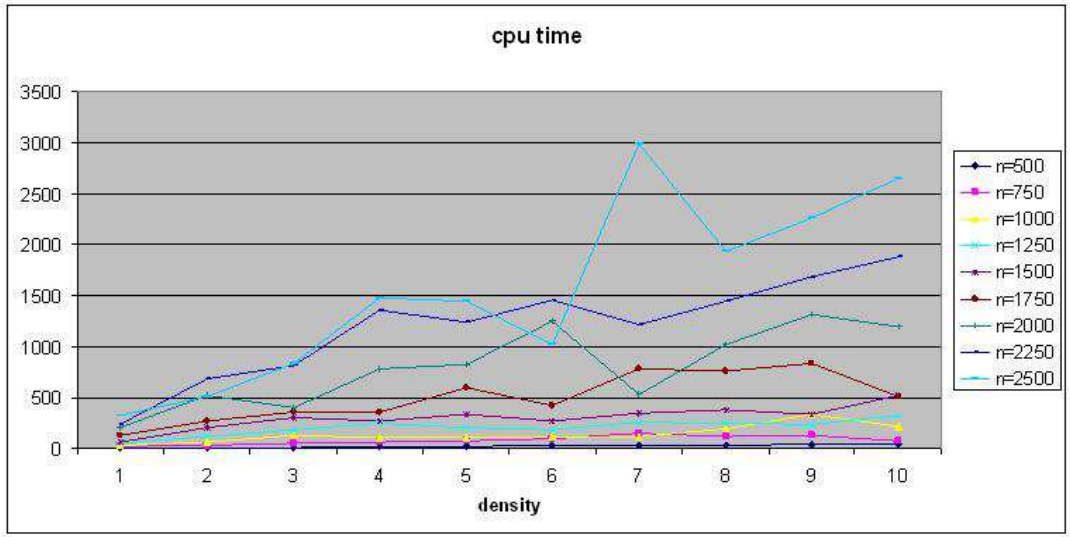


Figure 3: Average CPU time of the heuristic on the random graphs

The heuristic is able to produce a good cut in a small amount of time, and as expected the performance of the heuristic is better on sparse graphs in term of time, but the gap decreases when the density of the graph increases.

Some huge graphs

We generate three random graphs with $100\,001$ nodes, $7\,050\,827$ edges and different weights. The results are in Table 1 where we report the ranges of the weights, the total time, the value of the bound, the best cut obtained and the % gap.

Weights	Total CPU time	Upper Bound	Best Cut	gap%
1	15 043.98	4 113 227.8	3 959 852	3.87
$[1, 100]$	15 142.22	212 076 831.2	203 236 495	4.35
$[-1000, 1000]$	15 919.40	21 006 071 437.9	20 129 935 523	4.35

Table 1: Random sparse graphs with 100 001 nodes and 7 050 827 edges

Some huge graphs

We also generated 6-regular graphs (3D toroidal grid graphs) with 1 030 301 nodes and 3 090 903 edges and different weights. The results are reported in Table 5. To the best of our knowledge, no other methods can achieve this accuracy for graphs of this size.

Weights	Total CPU time	Upper Bound	Best Cut	gap%
1	4 723	3 090 133	3 060 300	0.97
$[1, 10]$	22 042	15 454 739	15 338 007	0.76
$[1, 1000]$	29 072	1 545 550 679	1 534 441 294	0.72
$[-100, 100]$	47 491	57 288 795	49 111 079	14.27

Table 2: 6-regular graphs with 1 030 301 nodes and 3 090 903 edges

Conclusions

- SpeeDP is a very fast and accurate method for solving the SDP relaxation of Max Cut

Conclusions

- SpeeDP is a very fast and accurate method for solving the SDP relaxation of Max Cut
- SpeeDP may be useful in order to increase the dimension of the problems solved with an exact algorithm

Conclusions

- SpeeDP is a very fast and accurate method for solving the SDP relaxation of Max Cut
- SpeeDP may be useful in order to increase the dimension of the problems solved with an exact algorithm
- SpeeDP allows to define efficient heuristics that produce feasible cuts with a certified upper limit on the distance from optimality

Conclusions

- SpeeDP is a very fast and accurate method for solving the SDP relaxation of Max Cut
- SpeeDP may be useful in order to increase the dimension of the problems solved with an exact algorithm
- SpeeDP allows to define efficient heuristics that produce feasible cuts with a certified upper limit on the distance from optimality
- SpeeDP-MC allows to find very good cuts for graphs with million of edges

Future work

- Extend SpeeDP to the bisection problem

Future work

- Extend SpeeDP to the bisection problem

Future work

- Extend SpeeDP to the bisection problem
- Define a parallel version of the code. The main burden in computing the function

$$f_\epsilon(v) := \sum_{i,\ j} q_{ij} \frac{v_i^T v_j}{\|v_i\|\|v_j\|} + \frac{1}{\epsilon} \sum_{i=1}^{n} \frac{(\|v_i\|^2 - 1)^2}{d(v_i)},$$

and its gradient is the vector with the i-th component equal to $\sum_j q_{ij} \frac{v_j}{\|v_j\|}$, but this can be parallelized, getting a code that is faster and able to tackle even larger graphs.