Numerical Study of Semidefinite Bounds for the k-cluster Problem

Jérôme Malick, Frédéric Roupin

To cite this version:

Jérôme Malick, Frédéric Roupin. Numerical Study of Semidefinite Bounds for the k-cluster Problem. Electronic Notes in Discrete Mathematics, Elsevier, 2010, 36, pp.399-406. 10.1016/j.endm.2010.05.051 . hal-00609746

HAL Id: hal-00609746

https://hal.archives-ouvertes.fr/hal-00609746

Submitted on 19 Jul 2011

Numerical Study of Semidefinite Bounds for the k-cluster Problem

Jérôme Malick [1,2]

CNRS, Lab. Jean Kuntzmann, INRIA Rhône-Alpes
655 avenue de l'Europe, Montbonnot, 38334 St Ismier Cedex, France

Frédéric Roupin [3]

CNAM, Lab. CEDRIC
292, rue St-Martin 75141 Paris Cedex, France

Abstract

This paper deals with semidefinite bounds for the k-cluster problem, a classical NP-hard problem in combinatorial optimization. We present numerical experiments to compare the standard semidefinite bound with the new semidefinite bound of [MR09], regarding the trade-off between tightness and computing time. We show that the formulation of the semidefinite bounds has an impact on the efficiency of the numerical solvers, and that the choice of the solver depends on what we expect to get: good accuracy, cheap computational time, or a balance of both.

Keywords: combinatorial optimization, semidefinite programming, k-cluster, Lagrangian duality and relaxation

[1] This work has been supported by CNRS through "GdR Recherche Opérationnelle" and by Grenoble University through "Pôle Math-STIC". We thank Quentin Monnet and Lise-Marie Veillon for their valuable help in developing parts of the solver.
[2] Email: `jerome.malick@inria.fr`
[3] Email: `frederic.roupin@cnam.fr`

1 Introduction, motivations

Many studies have shown the power, as well as the limitations, of semidefinite programming (SDP) in combinatorial optimization. Regarding bounding and exact resolution, the current statement of fact is, roughly speaking, that semidefinite programming bounds are of very good accuracy, but demand much time to be computed. This drawback prevents in general their use in exact resolution procedures, as branch-and-bound algorithms.

The particular problem "k-cluster" (see forthcoming (1)) is well-suited for a resolution using SDP, thought. Indeed this classical combinatorial optimization problem does not admit a polynomial time approximation scheme, and it is known to be NP-hard even in very special cases (see e.g. [LMZ08] for recent results and references). The point is that linear bounds for this problem are inaccurate so that they can hardly be used directly in an exact resolution procedure. So in practice, one of the most efficient general exact resolution methods for k-cluster problems is based on convex quadratic programming combining an SDP solver with CPLEX [BEP09].

The recent work [MR09] presents a SDP-like bound for k-cluster which keeps nice features of the SDP bounds, while gaining in computation time. We sketch the approach in section 4. The driving idea is to trade computing time for a (small) deterioration of the quality of the bound. The new SDP-like bound reaches indeed a good balance between tightness and speed of computation. A basic branch-and-bound algorithm using this bound have been shown to have very good performances and to be competitive with [BEP09].

The goal of this paper is to give a numerical comparison of these two bounds (respectively the standard and the new SDP bound), in terms of tightness, computation time, and balance between both. The computing time depends obviously on the solvers, and in turn the performance of the solvers depends on the formulation of the bounds. So we consider two SDP solvers known to be very efficient, and two equivalent formulations of the SDP bound (each of both turns out to advantage one solver). The conclusion of this numerical work is that the new SDP bound has an advantageous trade-off between accuracy and speed of computation in view of embedding it inside of a branch-and-bound.

2 k-cluster problem

We briefly introduce the k-cluster problem, the combinatorial optimization problem we consider in this work. We refer to the recent [LMZ08] for complete references, and to [JS05] for discussions on the SDP approach for this problem.

Consider an undirected, weighted graph $G = (V, E)$ of n vertices $v_1, \dots, v_n$, and a non-negative weight w_{ij} on each edge (v_i, v_j). For a given integer k in $\{1, \dots, n\}$, the k-cluster problem consists in determining a subset S of k vertices such that the total edge weight of the subgraph induced by S is maximized. In unweighted graphs, the problem is also called the densest subgraph problem. Denoting $W = (w_{ij})_{ij}$ the weight-matrix of the graph G, the problem can be written as a $\{0, 1\}$-quadratic optimization problem with one linear constraint

$$\max \left\{ \frac{1}{2} x^\top W x : \quad \sum_{i=1}^{n} x_i = k, \; x \in \{0, 1\}^n \right\}. \tag{1}$$

As mentioned in the introduction, this problem is well-suited for a SDP approach. We study here the quality of different SDP relaxations of this problem.

3 Lagrangian and semidefinite relaxations

Lagrangian duality is a mechanical way to create bounds. The choice of the constraints to be dualized gives different bounds. The dualization of all the constraints for a $\{0, 1\}$-quadratic problem leads naturally to a SDP problem (see e.g. the pedagogical [LO99]). The best bound θ_{B} obtained by duality for $\{0, 1\}$-quadratic optimization is the one given by the dualizing only the $0, 1$-constraints. This bound however does not lead to a SDP problem directly. The usual technique is to add redundant constraints to the problem until the total dualization (corresponding to a SDP problem) tights the best bound θ_{B}.

General results [FR07] applied to the $\{0, 1\}$-quadratic problem (1) give that both the following constraint sets to be dualized lead to θ_{B}:

- all the "product" constraints $x_j (\sum_{i=1}^n x_i - k) = 0$ for all j in $\{1, \dots, n\}$,
- just the "squared" constraint $(\sum_{i=1}^n x_i - k)^2 = 0$.

The next theorem formalizes this result while operating at the same time several transformations: namely the change of variable $\{0, 1\} \to \{-1, 1\}$, the homogenization of linear terms in $\mathbb{R}^n$ and the lifting in $n+1$ symmetric matrix space $\mathcal{S}_{n+1}$. The notation we need is the following. Define the $n+4$ symmetric $(n+1) \times (n+1)$-matrices Q, Q_S, and Q_j (for $j \in \{0, \dots, n\}$) by

$$Q := \frac{1}{4} \begin{bmatrix} e^\top W e & e^\top W \\ W e & W \end{bmatrix}, \qquad Q_0 := \begin{bmatrix} 0 & e^\top \\ e & 0 \end{bmatrix}, \qquad Q_j := \begin{bmatrix} 0 & \tilde{e}_j^\top \\ \tilde{e}_j & C_j \end{bmatrix}$$

and $Q_S = \begin{bmatrix} 0 & 0 \\ 0 & ee^\top \end{bmatrix} + (n-2k)Q_0$ where e is the vector of all ones, $\tilde{e}_j = e + (n-2k)e_j$ is the vector of $\mathbb{R}^n$ made up from e and e_j the $j+1$-th element of the canonical basis of $\mathbb{R}^n$, and the jth line/column of C_j equals $e + e_j$ (all other elements are 0). Moreover E_i is the $(n+1)\times(n+1)$ matrix with zero entries everywhere but in position (i,i) where there is a one. Recall that the inner-product in matrix space $\mathcal{S}_{n+1}$ is $\langle X, Y\rangle = \sum_{i,j=1}^{n+1} X_{ij}Y_{ij} = \text{trace}(XY)$, then set

$$\begin{aligned}
\mathcal{A}_P(X) &:= \big(\langle Q_0, X\rangle, \dots, \langle Q_n, X\rangle, \langle E_0, X\rangle, \dots, \langle E_n, X\rangle\big) \ \in \mathbb{R}^{2n+2} \\
\mathcal{A}_S(X) &:= \big(\langle Q_S, X\rangle, \langle E_0, X\rangle, \dots, \langle E_n, X\rangle\big) \ \in \mathbb{R}^{n+2} \\
b_P &:= (4k-2n, \dots, 4k-2n, 1, \dots, 1) \ \in \mathbb{R}^{2n+2} \\
b_S &:= ((2k-n)^2, 1, \dots, 1) \ \in \mathbb{R}^{n+2}
\end{aligned}$$

Theorem 3.1 (Equivalent SDP relaxations of k-cluster) *The two SDP problems*

$$\max\{\langle Q, X\rangle : \ \mathcal{A}_P(X) = b_P, \ X \succeq 0\} \tag{2}$$
$$\max\{\langle Q, X\rangle : \ \mathcal{A}_S(X) = b_S, \ X \succeq 0\} \tag{3}$$

have the same optimal value, which corresponds to θ_B, the relaxation of the $\{0,1\}$ constraint in (1). We call it the SDP bound for k-cluster. Adding any redundant quadratic equalities to (1) cannot enhance this bound.

Proof. The equivalence of the SDP relaxation of (1) with the product constraints on one hand, and with the squared constraint on the other hand comes from [FR07, Prop. 5], as well as the optimality among addition of any redundant equality constraints. The fact that the transformations lead to those SDP is detailed in [MR09] for the case with product constraints. The case with the squared constraint follow the same way. $\square$

To compute the SDP bound for k-cluster (that is, to solve (2) or (3)), we use solvers that are known to be among the most efficient and reliable:

- CSDP [Bor99], which implements an interior-point method,
- SB, based on the spectral method [HR00], which is much used in the context of combinatorial optimization.

The efficiency of these solvers depends on the formulation of the SDP bound. In Section 5, we will take the best choices to have a fair comparison with the alternative bound and solver, that we present in the next section.

4 Alternative semidefinite bounds

The recent work [MR09] introduces the following SDP-like bounds for the k-cluster problem. For a given $\alpha > 0$, we define $\Theta_P(\alpha)$ by

$$\Theta_P(\alpha) = \left(\frac{\alpha}{2}n^2 + \frac{1}{2\alpha}\|Q\|^2\right) - \frac{\alpha}{2}\min\left\{\|X - Q/\alpha\|^2 : \mathcal{A}_P(X) = b_P,\ X \succeq 0\right\}. \quad (4)$$

We similarly define $\Theta_S(\alpha)$ with $\mathcal{A}_S$ and b_S; the properties we discuss below hold for both $\Theta_P(\alpha)$ and $\Theta_S(\alpha)$. Regarding our objectives (explained in the introduction), $\Theta_P(\alpha)$ has interesting features. [MR09] shows that $\Theta_P(\alpha)$ is a bound for (1) which is weaker that the SDP bound, but can still get arbitrarily close to SDP:

$$\Theta_P(\alpha) \geq \text{val}\,(2) \qquad \text{and} \qquad \Theta_P(\alpha) \longrightarrow_{\alpha\to 0} \text{val}\,(2).$$

Moreover, it turns out that solving (4) is easier to solve than (2) (for α not too small). The quadratic SDP problem inside of (4) is a particular semidefinite least-squares problem; so we call $\Theta_P(\alpha)$ and $\Theta_S(\alpha)$ the SDLS bounds. These problems have nice geometrical properties and admit an efficient resolution, even for large-scaled problems; we have implemented an solver adapted and tuned for solving (4) using the approach presented in [Mal04]. Of course, computing $\Theta_P(\alpha)$ gets harder when $\alpha \to 0$. In practice, we fix the value $\alpha = 10^{-4}$ for which our solver has no problem while the obtained bound $\Theta_P(\alpha)$ is almost as accurate as the SDP bound (2). There is still a difference which is notified in the tables of results in relative value in the column "gap(%)".

5 Numerical results

The numerical experiments aim at comparing:

- the usual SDP bound for k-cluster (2) or (3) computed by CSDP and SB,
- the new bound $\Theta_P(\alpha)$ or $\Theta_S(\alpha)$ computed by our solver, called SDLS.

We consider randomly generated instances of k-cluster (test problems already used for similar testing [BEP09]). We use a Pentium IV 2.2 GHz with 1 Go of RAM under Linux.

In Table 1, we compare the solvers on each of the two formulations: the one with the product constraints and the one with the squared constraint. For CSDP, the computing time depends on the number of constraints, so (3) is solved about ten times faster than (2) when $n = 100$. On the other hand, (2) generally provides the best results for SB (except for large values of k or in dense graphs), as for $\Theta_P(\alpha)$. Hence, the best formulation of the semidefinite bounds is not always the one with fewer constraints.

In Table 2, we keep only the best formulation-software combination. Since the stopping criteria are different for the solvers, we indicate the time required by SB and CSDP to achieve the SDLS bound. The values of the stopping criteria are: 10^{-7} for SDLS, 10^{-4} for SB, and 10^{-5} for CSDP. Table 2 shows that our solver is fast. Moreover, the decrease of the bound at each iteration during the run is really strong. We illustrate this point in figure 1. In particular, our solver has a sharp decrease at the beginning. This is important for the use inside of a branch-and-bound. Note that even if the total computing time of SB is generally greater than the one of CSDP, the strong tailing-off effect of this solver can be an advantage (the two curves cross each other).

Our solver to compute the bound (4) combines several advantages of the SDP solvers SB and CSDP: it provides SDP-quality bounds, has a strong tailing-off effect, and it is significantly faster (even when the solving process is interrupted). These nice features explain why [MR09] used it successfully in a branch-and-bound to solve the k-cluster problem exactly. It is also interesting to note that the behavior of the solvers depends a lot on the formulation of the bound, and that dual solvers (like SB and SDLS) are more efficient in our case when there are more dual variables.

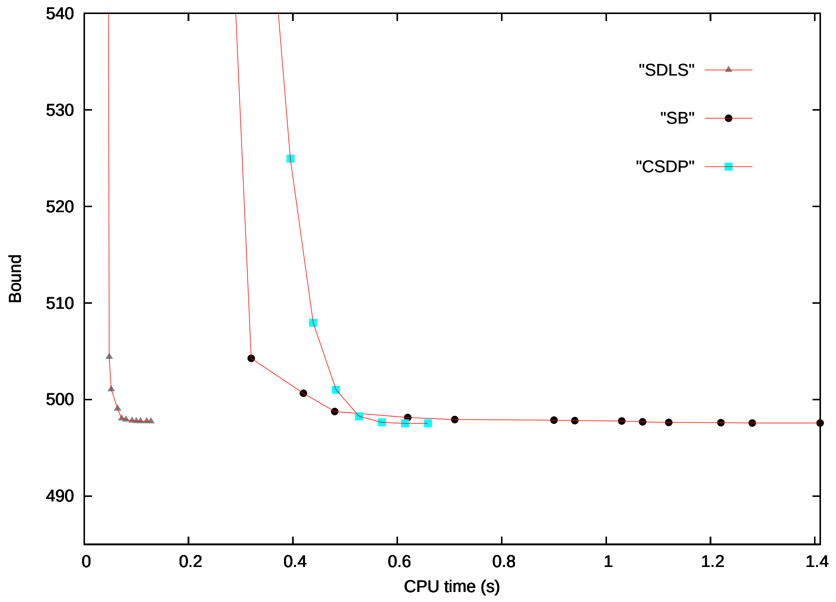

Fig. 1. Semidefinite bounds in dependence of time. $n = 80$, $d = 0.5$, and $k = 40$

Table 1
For each n, 45 problems are tested. The averaged relative difference between the computed bounds SDLS and SDP is gap(%).

Problem	SDLS $\Theta_P(\alpha)$		(2) with SB	(2) with CSDP
n	time (s)	gap(%)	time (s)	time (s)
80	0.15”	0.07%	1.09”	1.41”
100	0.19”	0.08%	2.90”	6.70”
300	2.59”	0.05%	25.84”	186.95”

	SDLS $\Theta_S(\alpha)$		(3) with SB	(3) with CSDP
n	time (s)	gap(%)	time (s)	time (s)
80	0,66”	1,46%	3.15”	0.34”
100	0,92”	1,66%	3.80”	0.58”
300	10,63”	2,11%	39.64”	10.12”

Table 2
Five problems are tested for each (n, k, d). d is the graph density in percent.

Problem			SDLS		SDP with SB		SDP with CSDP	
n	k	d(%)	time (s)	gap(%)	time (s)	time (s) to achieve SDLS	time (s)	time (s) to achieve SDLS
80	20	25	0.18”	0.21%	0.55”	0.23”	0.29”	0.19”
		50	0.18”	0.14%	0.87”	0.41”	0.45”	0.35”
		75	0.16”	0.12%	1.37”	0.61”	0.31”	0.24”
	40	25	0.17”	0.06%	0.54”	0.37”	0.35”	0.28”
		50	0.10”	0.04%	0.98”	0.80”	0.33”	0.28”
		75	0.07”	0.03%	1.39”	0.81”	0.40”	0.32”
	60	25	0.15”	0.02%	1.17”	0.85”	0.35”	0.29”
		50	0.14”	0.01%	1.34”	1.30”	0.35”	0.29”
		75	0.15”	0.01%	1.64”	1.41”	0.33”	0.29”
		mean	**0.15”**	**0.07%**	**1.09”**	**0.76”**	**0.34”**	**0.28”**
100	25	25	0.19”	0.23%	1.27”	0.51”	0.64”	0.50”
		50	0.23”	0.15%	2.40”	0.86”	0.53”	0.42”
		75	0.29”	0.13%	2.54”	1.14”	0.56”	0.45”
	50	25	0.23”	0.07%	1.15”	0.67”	0.52”	0.43”
		50	0.12”	0.05%	2.77”	1.01”	0.57”	0.47”
		75	0.12”	0.05%	2.02”	1.10”	0.59”	0.48”
	75	25	0.17”	0.02%	3.12”	1.75”	0.57”	0.47”
		50	0.17”	0.01%	1.86”	1.37”	0.65”	0.55”
		75	0.16”	0.01%	8.99”	6.25”	0.54”	0.47”
		mean	**0.19”**	**0.08%**	**2.52”**	**1.63”**	**0.58”**	**0.47”**
300	75	25	3.51”	0.15%	17.27”	7.20”	15.26”	12.40”
		50	3.56”	0.09%	26.96”	8.21”	7.04”	5.38”
		75	3.79”	0.08%	20.24”	9.12”	6.89”	5.60”
	150	25	1.66”	0.05%	11,45”	8.33”	16.25”	12.64”
		50	0.92”	0.02%	13.78”	11.93”	12.04”	9.92”
		75	0.74”	0.04%	8.21”	5.95”	7.30”	6.01”
	225	25	2.68”	0.02%	10.99”	10.05”	8.43”	6.74”
		50	3.01”	0.01%	52.92”	12.85”	7.84”	6.53”
		75	3.45”	0.01%	38.14”	25.66”	10.06”	7.19”
		mean	**2.59”**	**0.05%**	**22.22”**	**9.67”**	**10.12”**	**8.05”**

References

[BEP09] A. Billionnet, S. Elloumi, and M.-C. Plateau. Improving the performance of standard solvers for quadratic 0-1 programs by a tight convex reformulation: The qcr method. *Discrete Applied Mathematics*, 157(6):1185–1197, 2009.

[Bor99] B. Borchers. Csdp, a c library for semidefinite programming. *Optimization Methods and Software*, 11(1):613–623, 1999.

[FR07] A. Faye and F. Roupin. Partial lagrangian for general quadratic programming. *4'OR, A Quarterly Journal of Operations Research*, 5(1):75–88, 2007.

[HR00] C. Helmberg and F. Rendl. A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization*, 10(3):673–696, 2000.

[JS05] G. Jäger and A. Srivastav. Improved approximation algorithms for maximum graph partitioning problems. *Journal of Combinatorial Optimization*, 10(2):133–167, 2005.

[LMZ08] M. Liazi, I. Milis, and V. Zissimopoulos. A constant approximation algorithm for the densest k-subgraph problem on chordal graphs. *Inf. Process. Lett.*, 108(1):29–32, 2008.

[LO99] C. Lemaréchal and F. Oustry. Semidefinite relaxations and Lagrangian duality with application to combinatorial optimization. Rapport de Recherche 3710, INRIA, 1999.

[Mal04] J. Malick. A dual approach to semidefinite least-squares problems. *SIAM Journal on Matrix Analysis and Applications*, 26, Number 1:272–284, 2004.

[MR09] J. Malick and F. Roupin. Solving k-cluster problems to optimality using adjustable semidefinite programming bounds. *Submitted*, 2009.