Computational results of a semidefinite branch-and-bound algorithm for k-cluster

Nathan Krislock, Jérôme Malick, Frédéric Roupin

Computational results of a semidefinite branch-and-bound algorithm for k-cluster

Nathan Krislock[a,b], Jérôme Malick[c,a], Frédéric Roupin[d]

[a]*INRIA Grenoble Rhône-Alpes, Grenoble, France*
[b]*Department of Mathematical Sciences, Northern Illinois University, DeKalb, IL, USA*
[c]*CNRS, Lab. J. Kunztmann, Grenoble, France*
[d]*LIPN - CNRS UMR7030 - Université Paris-Nord*

Abstract

This computational paper presents a method to solve k-cluster problems exactly by intersecting semidefinite and polyhedral relaxations. Our algorithm uses a generic branch-and-bound method featuring an improved semidefinite bounding procedure. Extensive numerical experiments show that this algorithm outperforms the best known methods both in time and ability to solve large instances. For the first time, numerical results are reported for k-cluster problems on unstructured graphs with 160 vertices.

Keywords: combinatorial optimization, semidefinite programming, triangle inequalities, k-cluster problem, k-densest subgraph problem

1. Introduction

1.1. The k-cluster problem

Given a graph $G = (V, E)$, the *k-cluster problem* consists of determining a subset $S \subseteq V$ of k vertices such that the sum of the weights of the edges between vertices in S is maximized. This is a classical problem of combinatorial optimization, also known under the names "heaviest k-subgraph problem", "k-dispersion problem", and "k-densest subgraph problem" (when all the weights are equal to one). The problem can be seen as a generalization of

Email addresses: krislock@math.niu.edu (Nathan Krislock), jerome.malick@inria.fr (Jérôme Malick), frederic.roupin@lipn.univ-paris13.fr (Frédéric Roupin)

the max-clique problem, and also as a particular case of quadratic knapsack problem where all the costs are equal.

Letting $n = |V|$ denote the number of vertices, and w_{ij} denote the edge weight for $ij \in E$ and $w_{ij} = 0$ for $ij \notin E$, the problem can be modeled as the 0-1 quadratic optimization problem

$$\text{(KC)} \qquad \begin{array}{ll} \text{maximize} & \frac{1}{2} z^T W z \\ \text{subject to} & \sum_{i=1}^n z_i = k \\ & z \in \{0,1\}^n, \end{array} \tag{1}$$

where $W := (w_{ij})_{ij}$ is the weighted adjacency matrix of the graph G.

The k-cluster problem (1) is a fundamental graph optimization problem, and arises in many applications such as telecommunication, warehouse location, military defence, social networks, and molecular interaction networks; see more details and references in the introductions of, e.g., [Pis06, MR12, BCV$^+$12].

It is well-known that the k-cluster problem is a hard combinatorial optimization problem: it is NP-hard (even for special graphs, see,e.g., [FL01]), it does not admit a polynomial time approximation scheme [CP84, Kho05], and there is even a huge gap between the best known approximation algorithm and the known inapproximability results (see, e.g., [BCC$^+$10, BCV$^+$12]).

In practice, k-cluster problems are also difficult to solve to optimality. Even if there are many theoretical articles on k-cluster, there are only a few of them on exact resolution. Among the only works attacking this problem are the pioneering [Erk90], LP-based branch-and-bound method of [Pis06, Sec 2.4], and the convex quadratic relaxation method of [BEP09] using semidefinite programming and CPLEX. Note that, before 2006 and [Pis06], no non-trivial k-cluster problem of size $n = 100$ was able to be solved. As of 2014, the state-of-the-art method for solving problem (1) to optimality is the semidefinite-based branch-and-bound algorithm of [MR12] which is able to solve instances of size $n = 120$, and which compared positively to the previous methods on smaller problems.

1.2. Contribution and outline of this article

Our recent work [KMR12] presents an improved semidefinite bounding procedure for Max-Cut, another classical combinatorial optimization problem. Max-Cut problems can be written as maximizing a quadratic function over the vertices of a hypercube, that is, as (1) but without the equality constraint $\sum_{i=1}^n z_i = k$.

In this current paper, we build upon both [MR12] and [KMR12] by adapting and extending for the k-cluster problem the bounding procedure of [KMR12]. As we will see, extending techniques to k-cluster that have proven effective for Max-Cut brings complications due to the presence of the additional linear constraint. However, the extensive numerical experiments of this paper show that the resulting algorithm greatly outperforms the methods of [BEP09, MR12], which are the previous best existing methods to solve the k-cluster problem to optimality. Our algorithm is also able to solve unstructured k-cluster problems of sizes $n = 140$ and $n = 160$, for which no numerical results have been reported in the literature. The main contribution of this paper is thus to advance our ability to solve k-cluster problems to $n = 160$ from the previous limit of $n \leq 120$.

The outline of this paper is as follows. In Section 2 we describe our improved semidefinite bounding procedure for the k-cluster problem (1). In Section 3 we describe our branch-and-bound implementation using our improved semidefinite bounding procedure for solving k-cluster problems to optimality. In Section 4 we present our numerical results. Finally, we give concluding remarks in Section 5.

2. Improved semidefinite bounding procedure for k-cluster

In this section we describe the improved bounding procedure that is based on semidefinite programming (SDP) bounds of k-cluster. We start by briefly recalling the standard strengthened SDP relaxation of k-cluster that we will approximate with our bounds. We use the following standard notation: the inner product of two matrices X and Y is $\langle X, Y\rangle := \operatorname{trace}(X^T Y)$, and $X \succeq 0$ means that X is symmetric positive semidefinite.

2.1. Strengthened semidefinite relaxation

Semidefinite relaxations of the k-cluster problem (1) have already been considered in many papers for different purposes, such as [Rou04] for a computational study of the bounds, [BCV$^+$12] for (in)approximation results, and [Pis06] and [MR12] in the context of exact resolution.

The algorithm presented in this paper uses a strengthened semidefinite relaxation in $\{-1, 1\}$ variables as in [MR12]. The derivation of this SDP relaxation uses standard techniques (see, e.g., [PRW95, SVW00]), namely reinforcement by redundant constraints, homogenization, change of variables, and lifting to the space of matrices – see the reformulations 1-3 in [MR12, Section 1] leading to the SDP relaxation [MR12, Equation (9)]. For the sake of brevity, we do not repeat here the derivation of the relaxation, and we describe only the final SDP problem, refering to [MR12, Section 2] for more modeling information.

We consider the bound for the k-cluster problem (1) given by the following SDP problem:

$$(\mathrm{SDP}_I) \qquad \begin{array}{rc} \text{maximize} & \langle Q, X\rangle \\ \text{subject to} & \langle Q_j, X\rangle = 4k-2n, \quad j \in \{0,\dots,n\}, \\ & \operatorname{diag}(X) = e, \ \ X \succeq 0, \\ & A_I(X) \geq -e, \end{array} \tag{2}$$

where X lies in $\mathbb{S}^{n+1}$, e is the vector of all ones, and

$$Q := \frac{1}{4}\begin{bmatrix} e^TWe & e^TW \\ We & W \end{bmatrix}, \qquad Q_j := \begin{bmatrix} 0 & e^T + (n-2k)e_j^T \\ e + (n-2k)e_j & e_je^T + ee_j^T \end{bmatrix},$$

for $j \in \{0,\dots,n\}$, with $e_j \in \mathbb{R}^n$ being the j-th column of the $n \times n$ identity matrix, for $j \in \{1,\dots,n\}$, and $e_0 := 0 \in \mathbb{R}^n$. To lighten notation, we will gather all the equality constraints (including the diagonal ones) together as $B(X) = b$, where $b \in \mathbb{R}^{2n+2}$ and $B\colon \mathbb{S}^{n+1} \to \mathbb{R}^{2n+2}$ is a linear operator.

Let us explain briefly the role of the constraints; again, for more details about this formulation, we refer to [MR12, Section 1]. The $j = 0$ constraint in problem (2) comes from the constraint $\sum_{i=1}^n z_i = k$ in the original k-cluster problem (1). In addition, we further strengthen the SDP bound by adding reinforcing equality constraints and valid triangle inequality constraints:

i) **Reinforcing equality constraints.** The reinforcing equality constraints,

$$\langle Q_j, X\rangle = 4k-2n, \quad j \in \{1,\dots,n\}, \tag{3}$$

come from including the redundant product constraints $\sum_{i=1}^n z_iz_j = kz_j$, for $j \in \{1,\dots,n\}$, to the original problem before forming the SDP (Lagrangian) relaxation. It is known that the reinforcing equality constraints (3) provide the best possible SDP bound when considering the inclusion of valid *equality* constraints (see, e.g., [FR07]). More precisely, when adding any set of redundant quadratic constraints $\{z^TC_jz + b_j^Tz + a_j = 0 : j \in J\}$ before forming the SDP (Lagrangian) relaxation, the resulting bound is greater or equal to the *partial Lagrangian* *relaxation* of k-cluster,

$$\text{(DP)} \quad \min_{\mu} \max_{z \text{ s.t. } e^T z = k} \left\{ \frac{1}{2} z^T W z + \sum \mu_i (z_i^2 - z_i) \right\},$$

where only the binary constraints $z \in \{0,1\}^n$ (written equivalently as $z_i^2 - z_i = 0$, for $i = 1, \ldots, n$) are relaxed, but the equality constraint $\sum_{i=1}^n z_i = k$ (written equivalently as $e^T z = k$) is not relaxed. It is shown in [FR07] that the set of product constraints achieves the partial Lagrangian relaxation bound, and is therefore optimal when considering the inclusion of valid equality constraints.

ii) **Triangle inequalities.** The triangle inequalities are defined by

$$\begin{aligned} X_{ij} + X_{ik} + X_{jk} &\geq -1, \\ X_{ij} - X_{ik} - X_{jk} &\geq -1, \\ -X_{ij} + X_{ik} - X_{jk} &\geq -1, \\ -X_{ij} - X_{ik} + X_{jk} &\geq -1, \end{aligned}$$

for $1 \leq i < j < k \leq n+1$, and correspond to the fact that for any $x \in \{-1,1\}^{n+1}$, it is not possible to have exactly one of three products $\{x_i x_j, x_i x_k, x_j x_k\}$ equal to -1, nor is it possible to have all three of the products equal to -1. There are a large number, $4\binom{n+1}{3}$, of triangle inequalities in total. Therefore, we will iteratively add a subset of the most violated inequalities. For a subset of triangle inequalities I, we let $A_I \colon \mathbb{S}^n \to \mathbb{R}^{|I|}$ be the corresponding linear function describing the inequalities in this subset. For every set of triangle inequalities I, the SDP relaxation in problem (2) gives us an upper bound on the value of the maximum weight k-cluster:

$$\text{(KC)} \leq (\text{SDP}_I). \tag{4}$$

As was shown in [Rou04], the (SDP_I) bound with all triangle inequalities (i.e., $|I| = 4\binom{n+1}{3}$) is tight in that it achieves the optimal value of the k-cluster problem on some instances, but it is also expensive to compute. Instead of using the (SDP_I) bound directly, our approach here is based upon the semidefinite bounds of [MR13] that we present in the next section.

Before moving on to present our semidefinite bounds, we point out an easy but important result on the SDP relaxation (2): unlike the SDP relaxation of Max-Cut used in [KMR12], problem (2) is not strictly feasible.

Lemma 1. *The semidefinite relaxation* (2) *of the k-cluster problem is not strictly feasible (i.e. a feasible matrix cannot be positive definite).*

Proof 1. *Let X be feasible for problem* (2). *Let $v := \begin{bmatrix} 4k-2n \\ -2e \end{bmatrix} \in \mathbb{R}^{n+1}$. By partitioning X consistently with the vector v, we have*

$$Xv = \begin{bmatrix} X_{11} & X_{21}^T \\ X_{21} & X_{22} \end{bmatrix} \begin{bmatrix} 4k-2n \\ -2e \end{bmatrix} = \begin{bmatrix} (4k-2n)X_{11} - 2X_{21}^T e \\ (4k-2n)X_{21} - 2X_{22}e \end{bmatrix}.$$

Since $\operatorname{diag}(X) = e$, *we have $X_{11} = 1$. The $j = 0$ constraint can be written as*

$$\left\langle \begin{bmatrix} 0 & e^T \\ e & 0 \end{bmatrix}, \begin{bmatrix} X_{11} & X_{21}^T \\ X_{21} & X_{22} \end{bmatrix} \right\rangle = 4k - 2n.$$

Simplifying we get $2X_{21}^T e = 4k-2n$. Thus, the first component of Xv is zero:

$$(4k-2n)\,X_{11} - 2X_{21}^T e = (4k-2n) - (4k-2n) \,= 0.$$

Next we let $j \in \{1, \dots, n\}$ and consider constraint j:

$$\left\langle \begin{bmatrix} 0 & e^T + (n-2k)e_j^T \\ e + (n-2k)e_j & e_j e^T + e e_j^T \end{bmatrix}, \begin{bmatrix} X_{11} & X_{21}^T \\ X_{21} & X_{22} \end{bmatrix} \right\rangle = 4k - 2n.$$

Simplifying we get $2X_{21}^T e - (4k-2n)e_j^T X_{21} + 2e_j^T X_{22} e = 4k - 2n$. Since $2X_{21}^T e = 4k-2n$, we have $(4k-2n)e_j^T X_{21} - 2e_j^T X_{22} e = 0$. Since this holds for all $j \in \{1, \dots, n\}$, the last n components of Xv are also zero:

$$(4k-2n)X_{21} - 2X_{22}e = 0.$$

Thus we can conclude that $Xv = 0$. Since $v \neq 0$, we have that X is not positive definite. Therefore, problem (2) *is not strictly feasible.* □

In conic programming, strict feasibility is often used to guarantee no duality gap; see, e.g., [BTN01, Th.2.4.1]. In spite of the previous lemma, there is still no gap between (2) and its dual

$$\begin{array}{ll} \text{minimize} & b^T y + e^T z \\ \text{subject to} & Q - B^*(y) + A_I^*(z) \preceq 0, \\ & z \geq 0, \end{array} \tag{5}$$

where B^* and A_I^* are the adjoint operators of B and A_I respectively. Here it is the strict feasibility of the dual problem that implies zero duality gap. Since we have for $y \in \mathbb{R}^{2(n+1)}$

$$B^*(y) = \sum_{j=1}^{n+1} y_j Q_{j-1} + \operatorname{Diag}(y_{n+1}, \ldots, y_{2n+2}) \quad (\in \mathbb{S}^{n+1}),$$

we can take y arbitrarily large and make the dual problem strictly feasible. More specifically, taking $z = e$ and $y = \gamma(0, e)$, where γ is greater that the largest eigenvalue of $Q + A_I^*(e)$, gives a positive definite matrix in (5).

2.2. Our semidefinite bounds for k-cluster

We now summarize some useful results about the nonlinear SDP bounds we use to solve k-cluster to optimality. These bounds are based on the general nonlinear SDP bounds of [MR13], but are specialized here for the k-cluster problem.

For any matrix A, we denote by $\|A\|_F$ the *Frobenius norm* of A, which is defined as $\|A\|_F := \sqrt{\langle A, A\rangle}$. For a real number a, we denote its *nonnegative part* by $a_+ = \max\{a, 0\}$. We extend this definition to vectors and matrices as follows: for $x \in \mathbb{R}^n$, we define $(x_+)_i := (x_i)_+$, for $i = 1, \ldots, n$, and for a symmetric matrix A, we define $A_+ := U \operatorname{Diag}(\lambda_+) U^T$, where A has eigen-decomposition $A = U \operatorname{Diag}(\lambda) U^T$, with eigenvalues $\lambda \in \mathbb{R}^n$ and orthogonal matrix $U \in \mathbb{R}^{n \times n}$. We denote similarly a_-, x_-, and A_-.

Our first proposition is based on [MR13, Theorem 2]; we give the main ideas of the proof here to provide a convenient summary (using the notation of this paper) of the derivation of the semidefinite bounds used in this paper for the k-cluster problem.

Proposition 1. *Let I be a set of triangle inequalities. For $y \in \mathbb{R}^{2n+2}$ and $z \in \mathbb{R}^{|I|}$, let the matrix $X_I(y, z) \in \mathbb{S}^{n+1}$ be defined by*

$$X_I(y, z) := \left[Q - B^*(y) + A_I^*(z)\right]_+ .$$

Let $\alpha > 0$ and let $F_I^\alpha \colon \mathbb{R}^{2n+2} \times \mathbb{R}^{|I|} \to \mathbb{R}$ be the function defined by

$$F_I^\alpha(y, z) := \frac{1}{2\alpha} \|X_I(y, z)\|_F^2 + b^T y + e^T z + \frac{\alpha}{2}(n+1)^2. \tag{6}$$

Then $F_I^\alpha(y, z)$ is a valid upper bound for problem (1); that is,

$$(\mathrm{KC}) \leq F_I^\alpha(y, z), \qquad \text{for all } y \in \mathbb{R}^{2n+2},\ z \in \mathbb{R}_+^{|I|}. \tag{7}$$

Proof 2. *It is clear (see, e.g., [KMR12, Lemma 1]) that if* $X \in \mathbb{S}^{n+1}$ *satisfies* $\operatorname{diag}(X) = e$ *and* $X \succeq 0$*, then* $\|X\|_F \leq n+1$*. Therefore, since* $\alpha > 0$*, the following problem*

$$(\mathrm{SDP}_I^\alpha) \qquad \begin{array}{ll} \textit{maximize} & \langle Q, X\rangle + \frac{\alpha}{2}\left((n+1)^2 - \|X\|_F^2\right) \\ \textit{subject to} & B(X) = b,\;\; A_I(X) \geq -e,\;\; X \succeq 0, \end{array}$$

satisfies

$$(\mathrm{SDP}_I) \leq (\mathrm{SDP}_I^\alpha). \tag{8}$$

The Lagrangian of the (SDP_I^α) *problem is given by*

$$\begin{aligned} L(X; y, z) &:= \langle Q, X\rangle + \frac{\alpha}{2}\left((n+1)^2 - \|X\|_F^2\right) + \langle y, b - B(X)\rangle + \langle z, e + A_I(X)\rangle \\ &= \langle Q - B^*(y) + A_I^*(z), X\rangle - \frac{\alpha}{2}\|X\|_F^2 + b^T y + e^T z + \frac{\alpha}{2}(n+1)^2, \end{aligned}$$

where $y \in \mathbb{R}^{2n+2}$ *and* $z \in \mathbb{R}_+^{|I|}$*. By [MR13, Theorem 2] we have that* F_I^α *is the corresponding dual function; that is,*

$$F_I^\alpha(y, z) = \max_{X \succeq 0} L(X; y, z).$$

Then, by weak duality we have

$$(\mathrm{SDP}_I^\alpha) \leq F_I^\alpha(y, z), \qquad \textit{for all } y \in \mathbb{R}^{2n+2},\ z \in \mathbb{R}_+^{|I|}. \tag{9}$$

Combining inequalities (4), (8), *and* (9), *we obtain inequality* (7). □

The bounds $F_I^\alpha(y, z)$ coincide, up to change of sign and notation, with the bounds $\Theta(\lambda, \mu, \alpha)$ of [MR13]. For fixed $\alpha > 0$ and set of triangle inequalities I, the best possible bound $F_I^\alpha(y, z)$ can be found by solving the problem

$$\begin{array}{ll} \text{minimize} & F_I^\alpha(y, z) \\ \text{subject to} & y \text{ free},\;\; z \geq 0. \end{array}$$

The following proposition, based on [MR13, Theorem 2], gives us an important fact that supports the practical use of these bounds: the above problem is a convex and differentiable problem.

Proposition 2. *The function* F_I^α *is convex and differentiable; its gradients are given by*

$$\nabla_y F_I^\alpha(y, z) = b - \frac{1}{\alpha} B\left(X_I(y, z)\right), \quad \textit{and} \quad \nabla_z F_I^\alpha(y, z) = e + \frac{1}{\alpha} A_I\left(X_I(y, z)\right). \tag{10}$$

Proof 3. *From the proof of Proposition 1, the function F_I^α can be interpreted as a dual function, which immediately implies its convexity. The differentiability of F_I^α follows from [MR13, Theorem 2].* □

This smoothness allows us to use a quasi-Newton method that can handle bound constraints, such as L-BFGS-B [BLNZ95], to efficiently minimize $F_I^\alpha(y, z)$ over y and $z \geq 0$. For the management of the different parameters, we closely follow the bounding procedure of [KMR12] for Max-Cut: we minimize $F_I^\alpha(y, z)$ with increasing accuracy (the stopping tolerance ε of the quasi-Newton algorithm is driven to 0) while gradually adding inequalities and reducing α. The sketch of our bounding procedure is given in Algorithm 1; for more details, we refer to [KMR12, Algorithm 1].

Algorithm 1 Sketch of bounding procedure for k-cluster

```
Input: Scalars $\alpha_1 > 0$, $\varepsilon_1 > 0$, and initial set of triangle inequalities $I_1 = \emptyset$
  for $k = 1, 2, \dots$ do
    Use L-BFGS-B to compute $(y_k, z_k)$ such that
        $$\max\left\{ \left\| b - B\left(X_k\right) \right\|_\infty, \; \left\| \left[e + A_I\left(X_k\right)\right]_- \right\|_\infty \right\} < \varepsilon_k,$$
    where $X_k \leftarrow \frac{1}{\alpha_k} X_{I_k}(y_k, z_k)$.
    Remove inequalities $I_k^-$ that are not active, and add some inequalities
    $I_k^+$ that are violated by $X_k$:
        $$I_{k+1} \leftarrow (I_k \setminus I_k^-) \cup I_k^+.$$
    if the number of inequalities added $|I_k^+|$ is small then
      Decrease $\alpha_k$ and $\varepsilon_k$
    end if
  end for
```

We can extend the theoretical convergence analysis of [KMR12] to obtain the following proposition, which corresponds, for the bounding procedure of k-cluster, to the Theorem 1 of [KMR12] regarding the bounding procedure of Max-Cut.

Proposition 3. *Let the sequence $(\alpha_k, \varepsilon_k, X_k, y_k, z_k, I_k)_k$ be generated by Algorithm 1. If $(\alpha_k)_k$ and $(\varepsilon_k)_k$ both converge to zero, and $(\bar{X}, \bar{y}, \bar{z}, \bar{I})$ is an accumulation point of the sequence $(X_k, y_k, z_k, I_k)_k$, then the sequence of bounds converges to the classic semidefinite bound:*

$$\lim_{k\to+\infty} F_{I_k}^{\alpha_k}(y_k, z_k) = (\mathrm{SDP}_{\bar{I}}). \tag{11}$$

The proof of Theorem 1 of [KMR12] (and the lemmas and results used in the proof) extends easily to the k-cluster problem – except for Lemma 2 of [KMR12]. This lemma is in fact the obstacle to a straightforward generalization, since the proof given in [KMR12] cannot be directly generalized here: it requires that the semidefinite relaxation is strictly feasible, which is not the case for the k-cluster problem, as discussed in the previous section. Let us derive here the convergence result without this argument.

Proof 4. *Consider the subsequence that converges to $(\bar{X}, \bar{y}, \bar{z}, \bar{I})$. Since the I_k are finite sets, the convergence of a subsequence to $\bar{I}$ means that there are infinitely many indexes k_i of this subsequence such that $I_{k_i} = \bar{I}$. By* (6)*, we have*

$$F_{\bar{I}}^{\alpha_{k_i}}(y_{k_i}, z_{k_i}) = e^T y_{k_i} + e^T z_{k_i} + \alpha_{k_i}(\|X_{k_i}\|^2 + n^2)/2.$$

Since $\alpha_k \to 0$, the straightforward generalization for k-cluster of Lemma 3 of [KMR12] implies that the sequence $(F_{I_k}^{\alpha_k}(y_k, z_k))_k$ converges. Passing to the limit in the bounds, we get

$$\lim_{k\to+\infty} F_{I_k}^{\alpha_k}(y_k, z_k) = \lim_{i\to+\infty} F_{\bar{I}}^{\alpha_{k_i}}(y_{k_i}, z_{k_i}) = b^T\bar{y} + e^T\bar{z}. \tag{12}$$

Let us now pass to the limit for the variables. We can repeat the arguments when passing to the limit in the proof of Lemma 2 of [KMR12] (replacing diag *and* Diag *by B and B^*): we get that $\bar{X}$ is feasible for* $(\mathrm{SDP}_{\bar{I}})$*, that $(\bar{y}, \bar{z})$ is feasible for* $(\mathrm{DSDP}_{\bar{I}})$*, and that we have complementarity*

$$\left\langle Q - B^*(\bar{y}) + A_{\bar{I}}^*(\bar{z}), \bar{X}\right\rangle = 0 \qquad \text{and} \qquad \bar{z}^T(A_{\bar{I}}(\bar{X}) + e) = 0. \tag{13}$$

Observe now that for all X feasible in $(\mathrm{SDP}_{\bar{I}})$*, we have*

$$\begin{array}{rcl} b^T\bar{y} + e^T\bar{z} & \geq & b^T\bar{y} + e^T\bar{z} + \langle Q - B^*(\bar{y}) + A_{\bar{I}}^*(\bar{z}), X\rangle \\ & = & \langle Q, X\rangle + \bar{y}^T(b - B(X)) + \bar{z}^T(e + A_{\bar{I}}(X)) \\ & \geq & \langle Q, X\rangle . \end{array}$$

Moreover (13) *gives that the above inequalities are equalities for $\bar{X}$. This shows that the limit $b^T\bar{y} + e^T\bar{z}$ coincides with the optimal value of* $(\mathrm{SDP}_{\bar{I}})$*.* □

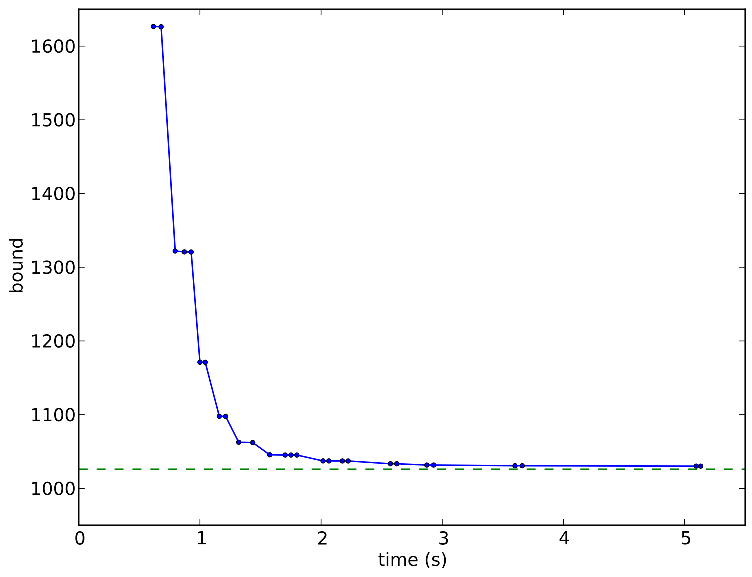

Figure 1: Time/bound plot of the Algorithm 1 bounding procedure on problem PB3_n160_d025_k80 which has optimal value 1026.

This proposition establishes the convergence and thus the consistency of the bounding procedure. More importantly, in practice the bounding procedure rapidly attains high quality bounds, which is the key property to be included in a branch-and-bound method. For illustration, we plot the convergence for our bounding procedure in Figure 1 on a k-cluster problem ($n = 160$, edge density 25%, $k = 80$). Such convergence curve is typical of what we have observed in our numerical experiments.

3. Branch-and-bound method for k-cluster

In this section, we describe our method for solving k-cluster to optimality. We list the main ingredients of our branch-and-bound implementation, and briefly compare them with the ones of [MR12].

a) **Bounds.** We have described our bounding procedure in Algorithm 1 in the previous section. We now discuss the differences between the bounding procedure of Algorithm 1 and the one used in [MR12]. With our notation, the bounds used in the branch-and-bound algorithm of [MR12] are given by (see [MR12, Lemma 2]):

$$\Theta(y,\alpha) := \frac{\alpha}{2}\left\|[Q/\alpha + B^*(y)]_+\right\|_F^2 - \alpha b^T y + \frac{\alpha}{2}(n+1)^2, \qquad \text{for } \alpha > 0. \quad (14)$$

Observe therefore that we have $\Theta(y,\alpha) = F_\emptyset^\alpha(-\alpha y)$ for all y and $\alpha > 0$. In other words, the bounds used in [MR12] essentially correspond (up to a scaling of the vector y) to the bounds that we use here, but with no triangle inequalities ($I = \emptyset$). The presence (and the efficient management) of the inequalities in Algorithm 1 is the key difference between our bounding procedure and the one of [MR12].
Though it is well-known that triangular inequalities greatly improve the SDP bounds for k-cluster (see, e.g., [Rou04]), it is a computational challenge to incorporate them in the bounding in a tractable way; this has never be done for this problem in a context of exact resolution (neither by [MR12] nor [Pis06, BEP09]). In practice, Algorithm 1 gradually reduces the two tightness parameters α and ε to zero while incorparating blocks of promising inequalities. This is another difference with [MR12] for which α and ε are kept fixed. Moreover, although both bounding procedures (ours and the one of [MR12]) use quasi-Newton algorithms, here we have to use a quasi-Newton solver that can handle bound constraints, and we also have to run this solver several times during the computation of the bounds.

b) **Heuristics.** We use the same two heuristic methods in the current branch-and-bound implementation as in the one of [Pis06] and [MR12]:
   (a) For the initial feasible point, we use the classical two-step greedy heuristic for k-cluster, since it gives very good feasible solutions.
   (b) After running the bounding procedure on a subproblem having k' nodes added to the cluster, we add the remaining $k-k'$ nodes having the largest fractional values.

c) **Branching rules.** In the current branch-and-bound implementation, we use the "difficult first" branching rule to decide which variables to branch on next: in the $\{0,1\}$ problem formulation, we select the variable having fractional value closest to $\frac{1}{2}$. This is the same branching rule that was used in [MR12]. In addition, given a list of subproblems in the branch-and-bound method, we must decide which subproblem to branch on next. We use the BOB Branch & Bound platform [CR95] that provides an easy and flexible way to implement a branch-and-bound algorithm. The BOB platform automatically handles the management of subproblems, and we select the subproblem with the weakest bound with the hope of making the most progress. In contrast, a basic depth-first traversal of the branch-and-bound tree was used for the selection of subproblems in [MR12].

4. Numerical results

As far as we are aware, the most challenging k-cluster test-problems publicly available are the ones used by [Bil05, BEP09, MR12], created by the graph generator `rudy`. The parameters of the instances of k-cluster are the size of the graph n, the value of k, and the graph density d. We take:

$$k = \frac{1}{4}n, \frac{1}{2}n, \frac{3}{4}n, \qquad d = 25\%, 50\%, 75\%.$$

The instances are randomly generated as follows: given a density $d \in [0, 1]$, a uniformly distributed random number $\rho \in [0, 1]$ is generated for any pair of indexes $i < j$; if $\rho > d$, then w_{ij} is set to 0, otherwise it is set to

- either to 1, generating an instance of pure k-densest subgraph problem,
- or to a integer in $\{0, \dots, 100\}$ (or $\{-100, \dots, 100\}$), generating an instance of (generalized) heaviest k-subgraph problem.

We implemented our algorithm in C / FORTRAN and have used the Intel Math Kernel Library (MKL) for the eigenvalue computations. We report the computational results of our algorithm on the two above sub-families of k-cluster problems. In our tests we used a Dell Intel Xeon CPU E31270 3.40 GHz (using a single core) with 8Go of memory and running the Linux operating system.

4.1. Numerical results for k-densest subgraph

For the k-densest subgraph instances, we have $n = 80, 100, 120, 140, 160$, and five instances generated by the graph generator `rudy` for each set of parameters (n, d and k). We also have a set of random instances for $n = 120, 140, 160$ generated by Amélie Lambert and available online. Thus we have a total of 360 instances of the k-densest subgraph problem.

The instances with $n = 80, 100$ have already been used in previous articles about k-cluster, such as [Bil05, BEP09, MR12]; we call them the standard instances. We call the instances with $n = 120, 140, 160$ the larger instances. As far as we are aware, this paper is the first one where numerical results are reported for instances with $n = 140$ and $n = 160$. The instances with $n = 120$ were used by [MR12], and their solver failed on some on them.

Our first experiment consists in comparing our method with the two best existing methods to solve the k-cluster problem to optimality:

1. the semidefinite method of [MR12], which is a precursor of the method presented here using the same branch-and-bound structure but basic semidefinite bounds (without dynamic control of α, ε and without triangle inequalities), as described in the previous section;
2. the quadratic convex reformulation (QCR) of [BEP09] which first convexifies the objective function of the problem (1) and then uses the state-of-the-art IBM/CPLEX mixed-integer convex quadratic solver (we use CPLEX 12.6 for the experiments). Since the best convexifying parameters of the initial problem are obtained by solving the semidefinite relaxation ($\mathrm{SDP_I}$) (we use CSDP [Bor99] for the experiments), the QCR and our method use the same (tight) bound at the root node of the branch-and-bound tree. However, the tightness of the QCR bounds deteriorates when going down this the search tree.

We have implemented the three methods, tuned their parameters to reach the best performance, and run them on the same machine. Table 1 reports their comparison in terms of CPU time and number of nodes of the search tree to reach optimality for all the problems. Note that the reported times for QCR are the ones of CPLEX 12.6 (single thread). The dual variables needed to convexify the problem are obtained by CSDP [Bor99]; but the computing times of CSDP are not consider in the reported CPU time as they are usually unsignificant (about 0.4 seconds for $n = 80$ and 0.9 or $n = 100$).

The figures of Table 1 show that our algorithm clearly outperforms the two other methods. Regarding memory issues, the two first methods use a few amount of memory (less than 4 Mo) whereas the third featuring CPLEX uses up to 6 Go. In view of this first experiment on the standard instance, we know focus on our method and try to reach larger size.

For the second experiment, we consider the larger instances and report in Table 2 the average number of nodes and time required for our solver to solve each set of five problems. We emphasize that our solver does not need to visit a lot of nodes in the branch-and-bound search tree; for example, we have found that 55% of the problems with size $n \leq 120$ are solved at the root of

Table 1: For the standard k-cluster problems: for the three best existing methods, we compare the number of nodes and CPU time (s), averaged over five instances for each triple (n, k, d).

			(b) our method		(c) [MR12]		(d) [BEP09]	
n	k	d(%)	nodes	time	nodes	time	nodes	time
80	20	25	3.4	3.2	11650	94.5	170658	39.9
		50	7.4	6.1	41857	323.1	536648	125.0
		75	13.8	9.5	102948	1002.4	1827452	407.4
	40	25	1.4	1.1	1544	13.1	26597	8.6
		50	1.0	0.8	2806	24.6	34148	9.6
		75	6.6	8.0	19789	195.7	231620	55.2
	60	25	1.0	1.1	148	1.3	946	0.6
		50	1.0	0.8	302	2.7	5128	3.1
		75	1.0	0.7	1123	11.2	5754	3.4
mean			4.1	3.5	20241	185.4	315439	72.5
100	25	25	20.6	31.3	127901	2207.1	5680415	1882.8
		50	35.0	41.6	303648	5543.0	19164583	6684.9
		75	30.6	32.3	1180710	19661.2	44336562	14275.8
	50	25	3.4	5.3	9328	164.9	415340	153.6
		50	25.4	46.1	211308	3923.5	5156390	2182.2
		75	1.4	2.1	27099	514.2	514822	203.1
	75	25	1.0	1.9	455	8.0	10261	5.1
		50	1.8	3.8	2018	39.7	108962	37.3
		75	1.0	1.6	1958	38.7	14956	6.5
mean			13.4	18.5	207158	3566.7	8378032	2825.7

the branch-and-bound tree. Our algorithm is also able to solve unstructured k-cluster problems of sizes $n = 140$ and $n = 160$, for which, as far as we are aware, no numerical results have been reported in the literature. Finally we note that 96% of the 360 test problems in are solved within three hours.

There are significant differences on the performance of the algorithm for problems of the same size. In particular, we notice that the number of nodes and total time tends to increase when d increases from 25%- 75%. This is due to the presence of many near-optimal k-clusters (and more generally many k-clusters can have very similar values) for larger density graphs. Consequently,

Table 2: For the larger problems: the number of nodes and CPU time, averaged over five instances for each triple (n, k, d). Entire set of results available online at http://lipn.univ-paris13.fr/BiqCrunch/results.

			(b) `rudy` instances		(c) other instances	
n	k	d(%)	nodes	time (s)	nodes	time (s)
120	30	25	119.4	315.8	16.6	36.0
		50	194.2	425.1	39.4	83.1
		75	422.2	889.8	62.2	119.5
	60	25	59.8	198.5	12.2	28.3
		50	85.8	263.2	27.4	69.4
		75	43.0	143.0	41.8	93.0
	90	25	1.8	6.8	1.0	2.9
		50	22.2	96.9	1.0	2.7
		75	1.0	3.0	1.0	2.6
140	35	25	366.2	1165.8	131.0	445.1
		50	1063.4	2888.6	383.0	964.7
		75	1558.6	4079.5	485.0	1279.8
	70	25	134.2	543.0	54.2	216.5
		50	780.6	3035.3	298.6	1133.0
		75	52.2	202.9	155.8	571.7
	105	25	2.6	14.1	1.4	7.5
		50	11.0	61.5	7.4	39.8
		75	6.6	34.8	3.0	19.1
160	40	25	744.6	2856.4	235.4	1023.6
		50	11325.4	37565.2	858.6	3280.1
		75	8050.6	26302.6	1132.2	3997.8
	80	25	395.4	1835.2	73.4	401.6
		50	993.4	4654.5	479.8	1908.6
		75	3829.0	18653.9	1425.0	6288.9
	120	25	31.4	219.8	1.4	9.5
		50	17.4	143.4	2.2	16.7
		75	9.8	82.2	4.2	31.7

even if the bound is tight, it is hard to prune nodes in the search tree since the evaluations of many nodes are almost the same. The bottomline is that

$n = 160$ is the largest size of unstructured k-cluster problems to be solved to optimality within a reasonable amount of time on a single-threaded machine.

4.2. Numerical results for heaviest k-subgraph

For the (generalized) heaviest k-subgraph instances instances, we have $n = 120, 140, 160$ and five instances generated by the graph generator `rudy` for each set of parameters (n, d, and k) and for positive weights in $0, \ldots, 100$ and for integer weights in $-100, \ldots, 100$. This makes a total of 270 instances. As above, we report the average number of nodes and time required to solve each set of five problems in Table 3.

The computing time and number of nodes reported in the two Tables 2 and 3 are comparable. The main difference between the two tables is for dense graphs with large k (look for example at the last lines of the two tables). For dense graphs with large k, unweighted instances have several optimal solutions (thus, it is easier to find an optimal solution). When k is smaller, the instances in the weighted and unweighted cases are more difficult and comparable with each other (look, for example, at the first line with $n = 160$, in the two tables).

The bottonline is that the nature of the weights (positive, negative, or 0-1) do not change the order of magnitude of computing times and the number of nodes. This confirms the folklore knowledge that the instances with $\{0, 1\}$ weights (k-densest subgraph instances) are as hard to solve as the weighted ones (heaviest k-subgraph instances). This is probably the reason why [Bil05, BEP09, MR12] only considered these instances. Such numerical observations can be related to theoretical results in complexity theory for similar optimization problems in graphs (see, e.g., [CST01]).

4.3. Online complementary material and on-going developments

We have made the following complementary material available online:

- the entire dataset of problems,
- the full numerical results of our tests,
- a web interface for the solver.

To access this material, we invite the interested reader to visit the BiqCrunch website:

http://lipn.univ-paris13.fr/BiqCrunch.

Table 3: For graphs with weights: the number of nodes and CPU time, averaged over five instances for each triple (n, k, d). Entire set of results available online at http://lipn.univ-paris13.fr/BiqCrunch/results.

			(b) 0-100 weights		(c) -100-100 weights	
n	k	$d(\%)$	nodes	time (s)	nodes	time (s)
120	30	25	37.8	178.8	21.8	112.5
		50	147.4	633.4	45.4	222.7
		75	110.6	472.6	107.4	467.9
	60	25	27.4	141.3	18.6	106.6
		50	113.0	536.4	76.2	392.4
		75	69.4	353.3	48.2	282.4
	90	25	1.0	5.4	1.8	21.2
		50	4.2	29.5	3.0	41.7
		75	4.2	32.1	2.6	26.7
140	35	25	115.8	673.9	104.6	612.4
		50	254.6	1368.1	1325.4	6581.9
		75	853.4	4282.9	899.4	4135.8
	70	25	44.6	293.3	19.8	142.3
		50	118.6	722.2	226.6	1348.0
		75	478.2	2787.6	247.8	1457.4
	105	25	3.4	32.8	1.8	35.2
		50	6.6	55.5	3.4	59.4
		75	14.6	147.7	8.2	123.8
160	40	25	372.6	2631.9	584.2	3971.7
		50	1443.0	9842.8	1767.4	11622.0
		75	2009.0	12102.4	2522.2	16667.4
	80	25	541.8	4161.5	165.0	1441.7
		50	429.4	3628.1	427.4	3330.4
		75	2473.8	17398.4	94.6	743.9
	120	25	6.2	67.7	2.2	47.5
		50	7.4	86.7	10.6	203.4
		75	20.2	242.4	12.6	227.1 I

Our current research and development are about extending the approach presented in this paper to general binary quadratic problems with quadratic and linear constraints. Since the theory has already been presented for the general case [MR13], our work essentially consists of modeling and computer implementations. For example, during the review process of this paper, we have done numerical experiments with the max-clique problem (also called "max independent set problem"), for which there exist many publicly available instances (DIMACS instances). These complementary results are and will be be posted on the web site.

We want to point out that it is for the k-cluster problem that we have got the most spectacular computational results, as reported in this paper. This comes probably from the fact that k-cluster is harder than other problems – in particular, harder than max-clique, both in the sense of computational resolution and in the sense of complexity theory (max-clique reduces to k-cluster in bipartite graphs [CP84]). Thus we do not present other results in this paper and instead we refer the interested reader to the website for further research and developments.

5. Conclusions

In this paper, we have presented an improved branch-and-bound algorithm to solve k-cluster problems to optimality. Our method is based on the previous work of [MR12] and [KMR12] and here we have summarized the main ideas from these papers and have highlighted the differences. The two main reasons why the method presented here is much better than the results reported in [MR12] are the introduction of triangle inequalities and the strategy of reducing α and ε in the semidefinite bounding procedure presented in Algorithm 1.

The main contribution of this paper is the extensive numerical experiments on benchmark k-cluster problems, including large-scale problems that are beyond the ability of previous exact methods to solve within a reasonable amount of time. We were able to solve for the first time a set of hard k-cluster instances of size $n = 160$, advancing our ability to solve instances from the previous limit of $n = 120$.

Our other contributions are: adapting the branch-and-bound method and bounding procedure of [KMR12] to solve k-cluster problems; extending the convergence results of [KMR12] in spite of the lack of strict-feasibility of the SDP relaxation of Lemma 1; comparing practical complexity for different subfamilies the of k-cluster problem; making the benchmark dataset of k-cluster problems and entire numerical results available online together with a web interface for our solver.

Since the semidefinite relaxation (2) of the k-cluster problem is not strictly feasible, we will also consider in our future work the use of semidefinite facial reduction (see, e.g., [RTW97]) with the hope of further improving the time to solve k-cluster problems to optimality.

Acknowledgment. We thank the referee for their pertinent comments and suggestions which helped us improve our numerical comparisons, as well as the convergence analysis. We acknowledge the support of the grant "ANR GeoLMI" and of the CNRS "Mastodons" grant "Gargantua".

References

[BCC$^+$10] A. Bhaskara, M. Charikar, E. Chlamtac, U. Feige, and A. Vijayaraghavan. Detecting high log-densities: an $o(n^{1/4})$ approximation for densest k-subgraph. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC'10)*, pages 201–210, 2010.

[BCV$^+$12] A. Bhaskara, M. Charikar, A. Vijayaraghavan, V. Guruswami, and Y. Zhou. Polynomial integrality gaps for strong sdp relaxations of densest k-subgraph. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'12)*, pages 388–405, 2012.

[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.

[Bil05] A. Billionnet. Different formulations for solving the heaviest k-subgraph problem. *Information Systems and Operational Res.*, 43(3):171–186, 2005.

[BLNZ95] R. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16(5):1190–1208, September 1995.

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

[BTN01] R. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization.* MPS-SIAM Series on Optimization. Siam Publications, 2001.

[CP84] D.G. Corneil and Y. Perl. Clustering and domination in perfect graphs. *Discrete Applied Mathematics*, 9(1):7–39, 1984.

[CR95] B.L. Cun and C. Roucairol. BOB: A unified platform for implementing branch-and-bound like algorithms. Technical report, University of Versailles Saint-Quentin-en-Yvelines, 1995.

[CST01] P. Crescenzi, R. Silvestri, and L. Trevisan. On weighted vs unweighted versions of combinatorial optimization problems. *Information and Computation*, 167(1):10 – 26, 2001.

[Erk90] E. Erkut. The discrete p-dispersion problem. *European Journal of Operational Research*, 46(1):48–60, 1990.

[FL01] U. Feige and M. Langberg. Approximation algorithms for maximization problems arising in graph partitioning. *J. Algorithms*, 41(2):174–211, 2001.

[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.

[Kho05] S. Khot. Ruling out PTAS for graph min-bisection, dense k-subgraph, and bipartite clique. *SIAM J. Comput*, 36:1025–1071, 2005.

[KMR12] N. Krislock, J. Malick, and F. Roupin. Improved semidefinite bounding procedure for solving max-cut problems to optimality. *Mathematical Programming*, pages 1–26, 2012.

[MR12] J. Malick and F. Roupin. Solving k-cluster problems to optimality with semidefinite programming. *Mathematical Programming B*, 136:279–300, 2012. Special issue on Mixed-Integer Nonlinear Programming.

[MR13] J. Malick and F. Roupin. On the bridge between combinatorial optimization and nonlinear optimization: a family of semidefinite bounds for 0-1 quadratic problems leading to quasi-newton methods. *Mathematical Programming B*, 140(1):99–124, 2013.

[Pis06] D. Pisinger. Upper bounds and exact algorithms for p-dispersion problems. *Computers and Operations Research*, 33:1380–1398, 2006.

[PRW95] S. Poljak, F. Rendl, and H. Wolkowicz. A recipe for semidefinite relaxation for (0,1)-quadratic programming. *Journal of Global Optimization*, 7:51–73, 1995.

[Rou04] F. Roupin. From linear to semidefinite programming: an algorithm to obtain semidefinite relaxations for bivalent quadratic problems. *Journal of Combinatorial Optimization*, 8(4):469–493, 2004.

[RTW97] M. Ramana, L. Tunçel, and H. Wolkowicz. Strong duality for semidefinite programming. *SIAM Journal on Optimization*, 7(3):641–662, 1997.

[SVW00] R. Saigal, L. Vandenberghe, and H. Wolkowicz. *Handbook of Semidefinite Programming*. Kluwer, 2000.