Computers & Operations Research 66 (2016) 153–159

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

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, France

ARTICLE INFO

Available online 29 July 2015

Keywords:
Combinatorial optimization
Semidefinite programming
Triangle inequalities
k-cluster problem
k-densest subgraph problem

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.

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 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

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

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

The k-cluster problem Eq. (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., [18,16,2].

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., [12]), it does not admit a polynomial time approximation scheme [8,14], and there is even a huge gap between the best known approximation algorithm and the known inapproximability results (see, e.g., [1,2]).

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 [11], the LP-based branch-and-bound method of [18, Section 2.4], and the convex quadratic relaxation method of [3] using semidefinite programming and CPLEX. Note that, before 2006 and [18], 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 Eq. (1) to optimality is the semidefinite-based branch-and-bound algorithm of [16] which is able to solve instances of size $n=120$. The second best method is the quadratic-programming-based [3] that shows equivalent performances for problems of size $n \leqslant 100$ (see the extensive comparison of [16]). Notice also, as reported by [3], that linear-programming-based methods are not competitive when solving large instances of k-cluster. This is due to the fact that linear relaxations for k-cluster have a poor ratio of tightness to computing time, as illustrated in [20, Section 4.3].

1.2. Contribution and outline of this article

Our recent work [15] presents an improved semidefinite bounding procedure for Max-Cut, another classical combinatorial optimization problem. Max-Cut problems can be written as maximizing a quadratic

* Corresponding author.
E-mail addresses: nkrislock@niu.edu (N. Krislock), jerome.malick@inria.fr (J. Malick), frederic.roupin@lipn.univ-paris13.fr (F. Roupin).

http://dx.doi.org/10.1016/j.cor.2015.07.008
0305-0548/

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 [16] and [15] by adapting and extending for the k-cluster problem the bounding procedure of [15]. 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 [3,16], which are the previous best existing methods for solving 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 $n\leqslant 120$.

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 $\langle X,Y\rangle := \text{trace}(X^TY)$, and $X\succcurlyeq 0$ 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 [20] for a computational study of the bounds, [2] for (in)approximation results, and [18,16] in the context of exact resolution.

The algorithm presented in this paper uses a strengthened semidefinite relaxation in $\{-1,1\}$ variables as in [16]. The derivation of this SDP relaxation uses standard techniques (see, e.g., [19,22]), namely reinforcement by redundant constraints, homogenization, change of variables, and lifting to the space of matrices – see the reformulations 1-3 in [16, Section 1] leading to the SDP relaxation [16, 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, referring to [16, Section 2] for more modeling information.

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

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

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

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

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 $B:\mathbb{S}^{n+1}\to\mathbb{R}^{2n+2}$ is a linear operator.

Let us explain briefly the role of the constraints; again, for more details about this formulation, we refer to [16, 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 adding 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., [13]). 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,

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

where only the binary constraints $z\in\{0,1\}^n$ (written equivalently as $z_i^2-z_i=0$, for $i=1,\dots,n$) are relaxed, but the equality constraint $\sum_{i=1}^{n} z_i=k$ (written equivalently as $e^Tz=k$) is not relaxed. It is shown in [13] 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

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

for $1\leqslant i<j<k\leqslant n+1$, 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 $\{x_ix_j,x_ix_k,x_jx_k\}$ 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 $A_I:\mathbb{S}^n\to\mathbb{R}^{|I|}$ 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

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

As was shown in [20], 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 [17] 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 [15], 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. Let X be feasible for problem (2). Let $v := \begin{bmatrix} 4k-2n \\ -2e\end{bmatrix}\in\mathbb{R}^{n+1}$.