NEW APPROXIMATION METHODS FOR SOLVING BINARY QUADRATIC PROGRAMMING PROBLEM

BY

RUI YANG

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Systems and Entrepreneurial Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

Adviser:

Professor Jiming Peng

ABSTRACT

In this thesis, we consider a special class of binary quadratic programming problem (BQP) where the number of nonzero elements is fixed. Such problems arise frequently from various applications and have been proved to be NP-hard. After a brief review of the quadratic programming problem, several optimization algorithms are presented.

In Chapter 3, we propose a new simple second order conic relaxation of the BQP problem. We derive some additional constraints based on the information from the data matrix. The algorithm will be compared with the existing SDP relaxation algorithm in terms of their numerical performances.

In Chapter 4, we use the convex quadratic relaxation as a geometric embedding tool to reformulate the underlying BQP as a clustering problem, where the target is to find a single cluster of fixed size. This connection allows us to employ many effective clustering algorithm developed in the data mining field. A 2-approximation algorithm for the clustering problem is presented. Numerical results based on the new relaxation model and the proposed algorithm are reported.

The last Chapter mainly discusses some theoretical results we put forward on the derived clustering problem. Core-set technique is used to derive a new algorithm, which can provide a $(1+\epsilon)$ approximation ratio to the reformulated clustering problem.

To my parents, for their love and support.

ACKNOWLEDGMENTS

I would like to gratefully appreciate my supervisor Professor Jiming Peng for his valuable comments and generous support. Without his support, this work would not have been done. I also thank him for his very careful reading and insightful suggestions during the writing of this thesis, and his wise guidance during my graduate study.

I appreciate Professor Jiming Peng, Professor Liming Feng, Professor Ramavarapu Sreenivas, Professor Jong-Shi Pang and Professor Angelia Nedich, for their inspirational lectures.

I acknowledge Professor Sariel Har-Peled, Mr. Bolin Ding and Mr. Qiang Ma for their discussion and for their helpful suggestions, in particular on the core-set approach to the problem in this thesis.

I thank all members of the Industrial and Enterprise Systems Engineering (IESE) department for their friendly help during my master studies and for the pleasant and exciting working environment they built.

At last, my special thanks go to my parents for everything they gave me, their love, understanding, encouragement, and support.

TABLE OF CONTENTS

LIST OF TABLES vi

CHAPTER 1 INTRODUCTION 1
- 1.1 The quadratic problem 1
- 1.2 Graph partition problem 2
- 1.3 Our problem 2
- 1.4 Organization of this thesis 5

CHAPTER 2 LITERATURE REVIEW 6
- 2.1 Approximation algorithm for optimization problem 6
- 2.2 Linear relaxation method 7
- 2.3 SDP relaxation method 8

CHAPTER 3 A SECOND-ORDER CONIC OPTIMIZATION RELAXATION FOR BQP PROBLEM 11
- 3.1 A second-order conic optimization relaxation for BQP 11
- 3.2 Numerical result 14

CHAPTER 4 A CLUSTERING-BASED ALGORITHM FOR THE DENSEST-K-SUBGRAPH PROBLEM 18
- 4.1 Equivalence between the densest k-subgraph problem and clustering problem 18
- 4.2 A 2-approximation algorithm 21
- 4.3 Numerical results 23

CHAPTER 5 CORE-SET TECHNIQUE FOR BQP PROBLEM . . . 29
- 5.1 A Survey: clustering using core-sets 30
- 5.2 Densest k-points v.s. BQP problem 41

CHAPTER 6 CONCLUSION AND FUTURE WORKS 46

REFERENCES 48

LIST OF TABLES

3.1 **SOCO model for the BQP problem: small case** 16
3.2 **SOCO model for the BQP problem: large case** 16
3.3 **SOCO model for the BQP problem: binary case** 17

4.1 **Densest k-subgraph model for BQP problem: general weight** 26
4.2 **Densest k-subgraph model for BQP problem: binary case** 27
4.3 **Densest k-subgraph model for BQP problem: special case** 28

CHAPTER 1

INTRODUCTION

1.1 The quadratic problem

Optimizing a quadratic function over some hypercube is one of the basic discrete optimization problems. The problem has several equivalent formulations in the literature. For instance,

$$\begin{aligned} \min_x \quad & x^T Q x \\ s.t. \quad & x \in \{-1, 1\}^n \end{aligned} \tag{1.1}$$

Adding a linear term to the objective function essentially won't change the problem too much:

$$\begin{aligned} \min_x \quad & x^T Q x + q^T x \\ s.t. \quad & x \in \{-1, 1\}^n \end{aligned} \tag{1.2}$$

is equivalent to

$$\begin{aligned} \min_x \quad & x^T \tilde{Q} x \\ s.t. \quad & x \in \{-1, 1\}^{n+1}, x_{n+1} = 1 \end{aligned} \tag{1.3}$$

where

$$\tilde{Q} := \begin{pmatrix} Q & \frac{q}{2} \\ \frac{q^T}{2} & 0 \end{pmatrix}$$

Therefore, the above problem leads to a quadratic problem of the form (1.1), of problem size increased by 1 and one more constraint $x_{n+1} = 1$.

Some researchers [1] also notice that (1.1) is equivalent to quadratic (0,1) programming (1.4).

$$\begin{aligned} \min_{x} \quad & x^T Q x & (1.4) \\ s.t. \quad & x \in \{0,1\}^n \end{aligned}$$

1.2 Graph partition problem

Another important case of quadratic problem is the graph partition problem. The maximization graph partition problem(MAX-GP) calls for finding a subset $S \subset V$ of k nodes such that an objective function $w(S)$ is maximized. A special case of $Max-Cut$ Problem, $Max-Cut\ with\ size\ k$ where the total edge weights of the edges crossing between S and V/S is maximized, is a graph partition problem. A similar graph partition problem is $Max-Not-Cut\ with\ size\ k$ where the total edge weights of the edges non-crossing between S and V/S is maximized. $Max\ Vertex\ Cover\ with\ size\ k$ problem, where the goal is to maximize the total edge weights of the edges covered by S, is also a case contained by graph partition problem. $Densest-k-subgraph(DSP)$ problem, whose objective function is the total edge weights of the subgraph induced by S, is the one we are quite interested in: as elaborated later, the DSP problem is equivalent to a special quadratic problem we study. Such connection enables us to integrate results from two completely different class of problems.

1.3 Our problem

In this thesis, we consider the following specific binary quadratic programming (BQP) problem:

$$\begin{aligned} \min_{x} \quad & x^T Q x + q^T x & (1.5) \\ s.t. \quad & Ax = b & (1.6) \\ & \sum_{i=1}^{n} x_i = k \\ & x \in \{0,1\}^n \end{aligned}$$

where $Q \in \Re^{n\times n}$ is a symmetric matrix, $A \in \Re^{m\times n}$ and $q \in \Re^n, b \in \Re^m$. The above model covers many scenarios arising from various applications such as molecular conformation [2], cellular radio channel assignment[3] and capital budgeting and financial analysis [4], the feature selection problem in ranking [5]. It covers also numerous graph problems such as the sparest (or densest) k-subgraph problem [6, 7] and Maximization graph partition problem [8]. Here we take *Max Vertex Cover with size k* problem as an example to show how it can be cast as the BQP problem 1.7 ([6]):

Given a graph $G = (V, E)$ and each edge $e_{ij} \in E$ of weight w_{ij}, the *Max Vertex Cover with size k* problem asks the maximal weight of edges covered by a set U of k vertices in G. We model this problem into the following quadratic programming problem:

$$\begin{aligned} \max_{e_{ij}\in E} \quad & \sum w_{ij}\left(\frac{3 + x_i + x_j - x_i x_j}{4}\right) \\ s.t. \quad & \sum_{i=1}^{n} x_i = 2k - n \\ & x \in \{-1, 1\}^n \end{aligned} \tag{1.7}$$

Note that for each edge e_{ij} the objective function is zero if both x_i and x_j are equal to -1 and w_{ij} otherwise. Further, the first constraint ensures that exactly k variables will be of value 1. Thus by choosing the vertex $v_i \in V$ to be in the cover U if and only if $x_i = 1$, the quadratic program corresponds to the *Max Vertex Cover with size k* problem on G. This quadratic problem follows the form 1.7.

The problem has been proved to be NP-hard [7]. Several researchers have studied such a problem and numerous algorithms have been proposed. For more details, we refer to [6, 7, 5, 8, 9] and the references listed in these papers.

The relaxation model plays an important role in the development of efficient algorithms for BQPs. For example, at every iteration of exact algorithms based on branch and bound (B & B) or branch and cut (B & C) approaches, one needs to solve a relaxed optimization problem to obtain a lower bound that can be used further to decide the branching strategy. A relaxation model that can be solved effectively and provide a tight bound is crucial for the success of the B&B or B&C process. Secondly, as showed in

[10, 11], the relaxation model can also help to design efficient approximation algorithms for classes of discrete optimization problems.

Various relaxation models for BQPs have been proposed in the literature. For example, Adams and Sherali [12, 13] first proposed to the well-known lifting and relaxation technique for binary optimization. Lovász and Schrijver [14] introduced a lift and project method to approximates the convex hull of $0-1$ valued solutions for a system of linear inequalities in higher dimension, and applied their method to the so-called vertex packing polytope problems. In [15], Lasserre considered SDP relaxations for nonlinear $0-1$ Programming based on the representations of nonnegative polynomials in algebraic geometry.

In this thesis, we are interested in bounds based on conic optimization relaxations for BQPs. The idea of using conic optimization, in particular positive semi-definite programming (SDP) relaxations for discrete optimization problems can be dated back to Lovász ([16], 1979) and Shor ([17], 1987). In early 1990s, Alizadeh [18] considered the SDP relaxation of various combinatorial optimization problems and used interior-point methods to solve the relaxed problem. A remarkable achievement in the study of SDP relaxation for discrete optimization is the work by Goemans and Williamson [10] where they designed a very efficient approximation algorithm for the max-cut problem based on its SDP relaxation. Since then, many results on SDP relaxations for discrete optimization problems have been reported in the literature. The survey [11] summarized the progress made in this direction and listed most references available up to that time.

It should be mentioned that most existing SDP relaxations for BQPs in the literature are built upon the lifting technique, which usually leads to an SDP in $\Re^{n\times n}$ or a higher dimensional space. Though these relaxations have helped to obtain strong bounds for the original BQP, the scalability of these approaches depends on the computational capacity of the SDP solvers and is restricted to moderately sized problems as pointed out in [9].

One main purpose of this work is to reconsider some existing simple convex quadratic programming relaxations for problem (1.7) and explore how to improve these simple relaxations and use the convex QP to design new algorithm. In particular, we follow a similar idea as proposed in [19] that used the minimum eigenvalue of the matrix Q to derive a convex relaxation for the original BQP (See also [20, 21]). However, for the special class of

BQPs in this work, by introducing an artificial variable, we first reformulate such a convex QP as an optimization problem over the second-order conic constraints (SOCO). Such a slight modification allows us to combine the classical graph modeling techniques to further improve the relaxation model. Secondly, we use the convex quadratic programming model as a geometric embedding tool to recast a special case of the BQPs, the so-called densest k-subgraph problem, as a clustering problem where the target is to find a single cluster of fixed size that minimizes the sum of squares of distances within the cluster. A simple approximation algorithm is proposed for the new clustering problem and such an algorithm can be viewed as a heuristics for the original BQP. Numerical results based on the new relaxation model and the proposed heuristics will be reported.

1.4 Organization of this thesis

The thesis is organized as follows. In the next Chapter, we review some existing relaxation algorithms on solving BQP problem. In Chapter 4 we first represent the classical relaxation for problem (1.7) in the form of second-order conic optimization and discuss how to use graph modeling techniques to add simple constraints that can enhance the relaxation model. Numerical experiments based on the new relaxation model will be reported. In Chapter 4, we first use the convex quadratic programming as a geometric embedding tool to recast a special case of BQPs, the densest k-subgraph problem as a very specific clustering problem. We then propose a algorithm with approximation ratio of 2 to the clustering problem and report numerical results based on such an algorithm. Core-set techniques have also been employed to develop approximation algorithm in Chapter 6. We conclude the paper by remarks in Chapter 5.

CHAPTER 2

LITERATURE REVIEW

2.1 Approximation algorithm for optimization problem

Many optimization problems have been proven to be NP-hard. This means that we can not expect polynomial time algorithms to find optimal solutions for these problems. Alternatively, it's still of interest to study these NP-problems via polynomial time algorithms and see how close to the optimality we can accomplish within polynomial time. These polynomial time algorithms are called approximation algorithm. Various techniques and algorithms are proposed in this purpose: combinatorial algorithms, for a bunch of important problems using a wide variety of algorithm design techniques, and linear programming based algorithms. For a complete review, we refer to Vazirani's book [22].

An NP-optimization problem Π is a fourtuple $(I, sol, m, goal)$ such that

1. I is the set of the instances of Π and it is recognizable in polynomial time.

2. Given an instance x of I, $sol(x)$ denotes the set of feasible solution of x. These solutions are short, that is, a polynomial p exists such that, for any $y \in sol(x), |y| \leq p(|x|)$. Moreover, it is decidable in polynomial time whether, for any x and any y such that $|y| \leq p(|x|), y \in sol(x)$.

3. Given an instance x and a feasible solution y of x, $m(x, y)$ denotes the positive integer measure of y. The function m is computable in polynomial time and is also called the *objective* function.

4. $goal \in \{max,\ min\}$

The class NPO is the set of all NP optimization problems.

For example, valid instances of the vertx cover problem consist of an undirected graph $G = (V, E)$ and a cost function on vertices. A feasible solution is a set $S \subseteq V$ that is a cover for G. Its objective function value is the sum costs of all vertices in S. A feasible solution of minimum cost is an optimal solution.

An approximation algorithm, $\mathcal{A}$, for Π produces, in polynomial time, a feasible solution whose objective function value is "close" to the optimal. For example, a polynomial time approximation algorithm for a maximization problem has a performance guarantee or worst case ratio $0 < r \leq 1$, if it outputs a feasible solution whose value is at least r times the maximal value for all instances of the problem. A key step in designing a good approximation algorithm for such a maximization problem is to establish a good upper bound on the maximal objective value. As we will see in detail soon, linear programming (LP) and semidefinite programming (SDP) have been frequently used to provide such upper bounds for many NP-hard problems.

2.2 Linear relaxation method

A large fraction of the theory of approximation algorithms, is built around linear programming(LP). Many combinatorial optimization problems can be cast as integer programs. Once this is done, the linear relaxation of this program provides a natural way of lower bounding the cost of the optimal solution. This is typically a key step in designing of an approximation algorithm. However, in the case of NP-problem, we cannot expect the feasible polyhedron to have integer vertices. Thus, we need to look for a near-optimal integral solution instead of optimal solution.

Basically there are two techniques for obtaining approximation algorithm via linear programming. The first one, which is straightforward to some extent, is to solve the linear problem and convert the fractional solution obtained into an integral solution, trying to ensure that during the process the cost does not increase much. The approximation guarantee is established by comparing the cost of the integral and fractional solution. This technique is called $LP - rounding$ or simply *rounding*.

The second method, which is more sophisticated, is to employ the dual of the LP-relaxation to develop algorithm. This technique is called the $primal-$ *dual schema*.Let us call the LP-relaxation the primal problem. Under this schema, an integral solution to the primal problem and feasible solution to the dual problem are constructed iteratively. Notice that any feasible solution to the dual also provides a lower bound of the optimal solution of the primal problem. The approximation guarantee is established by comparing the two solution.

As for the quadratic problem, linear programming has been playing an important role in the design of approximation algorithm. A substantial computational study based on linear programming(LP) and cutting planes is given by Barahona et al. [1]. The numerial results imply the efficiency of the LP-based algorithm when the graph is rather sparse. Some authors approach QP with techniques developed for pseudo-Boolen functions [23]. The concept of $roof\ dual$ studied in [23] is a linear relaxation of the problem over a subset of the triangle inequalities. Pardalos and Rodgers [24] solve QP by Branch and Bound with a preprocessing phase where they try to fix some of the variables. The main idea is the observation that x_i can be fixed if the partial derivative of the objective function with respect to x_i does not change sign over the convex hull of the feasible points. Surprisedly, the computational performance of this approach is quite similar to the results reproted in [1].

Eigenvalue-based approaches have been explored when solving quadratic problem as well. Mohar and Poljak [25] observed this when tackled the max-cut problem:

$$4mc(G) \leq \max_{x^t x = n} x^t L x = n\lambda_{max}(L)$$

As is shown in [26] the eigenvalue relaxation can be cast as a semidefinite program, which will be introduced in the next subsection.

2.3 SDP relaxation method

As we mentioned early, the works by Nesterov and Nemirovski [27], Alizadeh [18] on computational semidefinite programming promoted the application of semidefinite(SDP) relaxation. Some strong theoretical results have been obtained from SDP relaxation when approximating the NP-problem. Here we take max-cut problem as an example to illustrate how the SDP relaxation works.

The max-cut problem can be written as form 1.1 as observed by Mohar and Poljak [25]. Let G be an undirected graph on vertex set $V = \{1, 2, \ldots, n\}$

with edge weights $\{c_e : e \in E(G)\}$, given by its adjacency matrix $A = (a_{ij})$ where $a_{ij} = a_{ji} = c_e$ for $e \in E(G), e = (ij)$, and $a_{ij} = 0$ otherwise.

The max-cut problem is to determine a subset $S \subset V$ such that the total weight of the edges cut by the partition is maximized.

$$mc(G) := \max_{S \subset V} \sum_{i \in S, j \notin S} a_{ij}$$

Let's present the partitions $(S, V/S)$ by vectors $x \in \{-1, 1\}^n$ with $x_i = 1$ only if $i \in S$. Denote e as the vector of ones. Using the *Laplacian* L of G, defined as

$$L := diag(Ae) - A$$

it can be checked that

$$\frac{1}{4} x_s^t L x_s = \sum_{i \in S, j \notin S} a_{ij}$$

if $x_S \in \{-1, 1\}^n$ represent the partition $(S, V/S)$. Therefore the max-cut problem is a special case of quadratic problem (1.1).

The SDP relaxation for max-cut problem is based on the following simple observation

$$x^t L x = tr L(xx^t)$$

Let $\mathcal{F} = \{-1, 1\}^n$ denote the feasible set of the max-cut problem. We consider the set $\mathcal{P}_{\mathcal{C}} := conv\{xx^t : x \in \mathcal{F}\}$, the so-called *cut polytope*. With this notation the max-cut problem can be written as

$$\max tr(LX) \ s.t. \ X \in \mathcal{P}_{\mathcal{C}}$$

Since we can not precisely describe the polyhedron $\mathcal{P}_{\mathcal{C}}$, we may instead approximate it by the following:

$$X \in \mathcal{P}_{\mathcal{C}} \Rightarrow X \succeq 0, \quad diag(X) = e$$

The set $\{X \succeq 0 : diag(X) = e\}$ is called *elliptope*. Then we have the basic SDP relaxation,

$$\varphi(G) = \max tr(LX) \ s.t.\ X \succeq 0, \quad diag(X) = e$$

The above problem is a standard SDP problem and can be solved in polynomial time using the ellipsoid method. Goemans and Williamson [10] shown the following results in term of the output of the SDP relaxation problem.

Theorem 2.3.1. *[10] If G is a graph on nonnegative edge weights, then $\varphi(G) \leq 1.138mc(G)$*

Laurent and Poljak [28] studied the geometry of the set and improved if the adjacency matrix A has the form $A = aa^t$.

Theorem 2.3.2. *[28] Let G be a graph with adjacency matrix $A = aa^t$ and $a \geq 0$, then $\varphi(G) \leq 1.138mc(G)$*

Though theoretically SDP relaxation is promising, the computational efforts needed to solve a SDP problem is too much, especially when the size of problem become large. Some improvements have been done by incorporating SDP with cutting plane approach [9].

Researchers have employed semidefinite relaxations extensively for combinatorial problems with binary variables, including values in $\{-1, 1\}$ or $\{0, 1\}$. For the $\{-1, 1\}$ model, some strong results include Goemans and Williamson [10] and Han et al. [8]. The $\{0, 1\}$ model was used for the quadratic knapsack problem by Helmberg [29]. It is known by Laurent et al. [30] that the $\{-1, 1\}$ or $\{0, 1\}$ models essentially will lead to equivalent semidefinite relaxations.

CHAPTER 3

A SECOND-ORDER CONIC OPTIMIZATION RELAXATION FOR BQP PROBLEM

This section consists of two parts. In the first subsection, we introduce a new relaxation for BQP based on the so-called second-order conic optimization. In the second subsection, we report some numerical results based on the new relaxation.

3.1 A second-order conic optimization relaxation for BQP

In this subsection, we introduce a new simple relaxation model for problem (1.7). Throughout this section, we make the following assumption

Assumption 3.1.1. *The matrix Q has only zero diagonal elements.*

The above assumption holds without loss of generality. This is because if the matrix Q has nonzero diagonal elements, we can then use the relation $x_i^2 = x_i$ to rewrite the objective function in 1.7 with another matrix with zero diagonal elements.

Let $\lambda_{min}(Q)$ denote the minimal eigenvalue of the matrix Q. Under Assumption 3.1.1, one can easily verify that $\lambda_{\min}(Q) \leq 0$. Moreover, by the choice of $\lambda_{min}(Q)$, we have

$$Q_1 = Q - \lambda_{\min}(Q) I \succeq 0.$$

Now let us consider the following binary convex quadratic programming problem:

$$\begin{aligned} \min_{x} \quad & x^T Q_1 x + q^T x + k\lambda_{\min}(Q) \\ s.t. \quad & \sum_{i=1}^{n} x_i = k \\ & Ax = b \\ & x \in \{0,1\}^n. \end{aligned} \tag{3.1}$$

It is straightforward to prove the following result.

Theorem 3.1.2. *The two binary quadratic programming problems are identical in sense that they enjoy a common set of optimal solutions and have the same objective value at the optimal solution.*

Based on the above theorem, we can relax the binary constraint in problem 3.1 to linear constraint $x \in [0,1]^n$. The solution of the resulting simple convex quadratic programming problem will provide a lower bound to problem 3.1.

To further enhance the relaxation model, we introduce an artificial variable $\alpha = x^T Q x$ and rewrite problem 3.1 as follows:

$$\begin{aligned} \min_{x} \quad & \alpha + q^T x \\ s.t. \quad & \sum_{i=1}^{n} x_i = k \\ & Ax = b \\ & x^T Q_1 x \le \alpha - k\lambda_{\min}(Q) \\ & x \in \{0,1\}^n, \end{aligned} \tag{3.2}$$

and its relaxation

$$\begin{aligned} \min_{x} \quad & \alpha + q^T x \\ s.t. \quad & \sum_{i=1}^{n} x_i = k \\ & Ax = b \\ & x^T Q_1 x \le \alpha - k\lambda_{\min}(Q) \\ & x \in [0,1]^n, \end{aligned} \tag{3.3}$$

We next discuss how to add extra constraints to model 3.3 to improve the lower bound. For this purpose, we first note that under Assumption3.1.1, we can cast the matrix Q as the weight matrix of a graph G. Suppose that x^* is the optimal solution of problem1.7. Then $\alpha^* = (x^*)^T Q x^*$ will be the total weight of the minimum weight k-subgraph of G. In other words, the total weight of the minimum weight k-subgraph of G will provide a valid lower bound for α. We next describe how to use this observation to derive some simple bounds on α.

Let $Q_{i,:}$ denote the i-row column of Q. We define the matrix $Q^{(2)}$ as follows

$$Q^{(2)} = [q_{ij}^{(2)}] \in \Re^{n\times n}, \quad q_{ij}^{(2)} = \frac{1}{2}\sum_{l=1}^{k}[sort(Q_{i,:}+Q_{j,:})]_l, \quad \forall\, i \neq j. \qquad (3.4)$$

Here $sort(v)$ is a vector generated by rearranging the elements of v in an non-decreasing order. We also then compute a constant c_2 by taking the sum of the smallest C_k^{2}[1] elements in $Q^{(2)}$. One can easily see that the constant c_2 provides a valid bound for α, i.e., $\alpha > c_2$.

Similarly we can compute a cubic matrix $Q^{(3)}$ by

$$Q^{(3)} = [q_{ijl}^{(3)}] \in \Re^{n\times n\times n}, \quad q_{ijl}^{(3)} = \frac{1}{3}\sum_{m=1}^{k}[sort(Q_{i,:}+Q_{j,:}+Q_{l,:})]_m, \qquad (3.5)$$
$$\forall\, i \neq j \neq l$$

Then we can obtain a lower bound c_3 for α by computing the sum of the smallest C_k^3 elements in $Q^{(3)}$.

In a similar vein we can construct a four-dimensional matrix $Q^{(4)}$, and compute a lower bound c_4 for α if $k \geq 4$. Since the computation cost of such a process grows exponentially in term of the dimensionality of the matrix, in this paper, we use the lower bound from c_2, c_3 and c_4 as they can provide a valid constraint at a reasonable computation cost. Consequently, we can add the constraint

$$\alpha \geq c_4. \qquad (3.6)$$

to model 3.3.

[1]Here C_k^l denotes the combinatorial function of selecting l items out of a set of k items.

3.2 Numerical result

In this subsection, we present some numerical results based on our quadratic programming relaxation model and compare it with the popular SDP relaxation based on the (-1,1) representation of problem1.7 by using the transform $y = 2x - e \in \{-1, 1\}^n$ or $x = \frac{1}{2}(y + e)$. In such a case, we can rewrite the objective in problem1.7 as $\frac{1}{4}y^T Qy + \frac{1}{2}y^T(Qe + q) + \frac{1}{4}e^T Qe$. We then use the following SDP relaxation

$$\begin{pmatrix} 1 & y^T \\ y & Y \end{pmatrix} \succeq 0, \quad \operatorname{diag}(Y) = 1, \tag{3.7}$$

with additional constraints on y derived from the constraints in problem1.7.

All numerical experiments have been carried out on Intel Core 2 Duo CPU E6850 3.00GHz processor with 2048 MBytes of RAM. The problems are solved by CVX 1.2 Build 710 solver under Matlab R2008a. Since there is no general public BQP library available, we generate most problems randomly.

In the tables below, we compare two lower bounds and the computational time to compute these bounds. In table 3.1, we use random matrices of size 50 and 80, whose elements are uniformly distributed between 0 and 1. We also mention that in our experiments, we only add the simple constraint $\alpha \geq c_2$ (as discussed at the end of Section 2.1) to the relaxation model 3.3.

As one can see from Table 3.1, the bound provided by our simple QP relaxation is always below what obtained from the SDP relaxation. But the gap between these two bounds are very small, while the CPU time to compute the QP bound is around half of the CPU time used in the SDP relaxation model.

In Table 3.2, we list our experimental results for random problems whose size are from 100 to 300. As one can see from the table, competitive bounds have been observed for these test problems, while the CPU time to compute the QP bound is only about one quarter of the CPU time for the SDP relaxation model. It should also be mentioned that for problems whose size is about $n = 300$ or above, the CVX solver we used failed to solve the SDP relaxation due to the memory limit, while it solved the QP relaxation model for much larger scale problems up to $n \approx 1000$.

Finally, we also report some numerical results for BQP problems where the matrix Q has (0,1) elements. As one can see from Table 3.3, competitive bounds can be obtained by solving the simple QP relaxation model 3.3 with less CPU time.

Problem		Quadratic Relaxation		SDP Relaxation	
no.	k	bound	Time	bound	Time
M50_1	20	142.742	2.8437	146.268	3.5877
M50_1	30	376.29	2.7928	380.051	3.7433
M50_1	40	727.89	3.037	731.792	3.8349
M50_2	15	69.9186	2.7262	74.1151	3.5493
M50_2	30	396.188	2.6996	401.978	3.5852
M50_2	45	996.301	2.988	999.413	3.9945
M80_1	20	116.942	3.117	126.025	5.5962
M80_1	40	665.914	2.963	675.859	5.6839
M80_1	60	1670.82	3.0119	1678.6	5.8015
M80_2	20	111.551	3.0574	121.901	5.4771
M80_2	40	645.984	3.2572	656.494	5.5974
M80_2	60	1625.33	3.1622	1633.71	5.7996
M80_3	10	10.1747	3.1224	8.24968	5.3421
M80_3	30	327.111	3.0837	333.843	5.4922
M80_3	50	1076.13	3.3122	1085.57	5.6676

Table 3.1: **SOCO model for the BQP problem: small case**

Problem		Quadratic Relaxation		SDP Relaxation	
no.	k	bound	Time	bound	Time
M100_1	20	105.259	3.2383	116.98	7.5186
M100_1	50	1082.4	3.091	1096.35	8.3739
M100_1	70	2294.12	3.2223	2307.94	7.9505
M100_2	30	315.107	3.3196	327.195	7.5318
M100_2	60	1599.31	3.208	1610.83	7.9901
M100_2	90	3933.87	3.2861	3942.58	8.32
M200_1	50	905.835	6.446	941.251	32.1381
M200_1	100	4444.53	6.5898	4482.06	37.8778
M200_1	150	10696	7.4001	10733.9	39.3988
M200_2	60	1422.65	6.6772	1449.6	38.7024
M200_2	90	3556.06	6.6628	3583.27	38.3663
M200_2	180	15962.1	7.4515	15987.1	40.2179
M300_1	50	810.163	20.2258	N/A	N/A
M300_1	100	4227.98	22.0701	N/A	N/A
M300_1	150	10266.1	22.7524	N/A	N/A
M300_2	100	4244.87	17.5629	N/A	N/A
M300_2	150	10313.8	17.7235	N/A	N/A
M300_2	200	19063.2	18.0601	N/A	N/A

Table 3.2: **SOCO model for the BQP problem: large case**

Problem		Quadratic relaxation		SDP Relaxation	
no.	k	bound	Time	bound	Time
MB50_1	15	38.0359	3.0425	46.4833	3.6622
MB50_1	30	347.036	2.8773	354.376	3.6933
MB50_1	45	967.06	2.8803	973.523	3.5978
MB50_2	10	8.23E-08	2.7487	-1.80747	3.5593
MB50_2	20	95.6218	2.7796	103.088	3.57
MB50_2	40	684.449	2.886	691.363	3.6251
MB50_3	10	0.254515	2.7143	0.852562	3.4224
MB50_3	20	101.656	2.9615	109.535	3.4748
MB50_3	40	684.344	2.7871	690.387	3.6215
MB100_1	30	88.1464	3.4327	116.653	7.1674
MB100_1	60	935.986	3.3804	958.452	7.1556
MB100_1	90	2665.96	3.4546	2683.03	7.4072
MB100_2	25	88.1464	3.4327	116.653	7.1674
MB100_2	50	935.986	3.3804	958.452	7.1556
MB100_2	75	2665.96	3.4546	2683.03	7.4072
MB100_3	20	88.1464	3.4327	116.653	7.1674
MB100_3	40	935.986	3.3804	958.452	7.1556
MB100_3	80	2665.96	3.4546	2683.03	7.4072

Table 3.3: **SOCO model for the BQP problem: binary case**

CHAPTER 4

A CLUSTERING-BASED ALGORITHM FOR THE DENSEST-K-SUBGRAPH PROBLEM

In this section, we consider a special variant of problem (1.7)

$$\begin{aligned} \max \quad & x^T Qx \\ s.t. \quad & \sum_{i=1}^{n} x_i = k \quad x \in \{0,1\}^n. \end{aligned} \tag{4.1}$$

Like in Section 2, we also assume that the matrix Q has zero diagonal elements. In such a case, problem(4.1) reduces to the well-known densest k-subgraph problem, which has been proved to be NP-hard [7]. Moreover, unless the matrix Q has specific structure or k is as large as $O(n)$, no approximation algorithms with a constant approximate rate has been reported in the literature [6, 7]. In this section, we first recast problem as an equivalent specific clustering problem and then propose a simple 2-approximation algorithm for the resulting clustering problem. The section has three parts. In the first subsection, we use convex quadratic programming as a geometric embedding tool to reformulate problem(4.1) as a specific clustering problem. In the second subsection, we propose a simple approximation algorithm to the clustering problem and present a local search heuristics to further refine the solution. In the last subsection, we report some numerical results based on the proposed algorithm.

4.1 Equivalence between the densest k-subgraph problem and clustering problem

We start with the following specific clustering problem. Given a data set $\mathcal{V} = \{v_i \in \Re^d, i = 1, \cdots, n\}$, we want to find a subset $\mathcal{V}_1$ of size k such that the sum of squares of distances within the subset $\mathcal{V}_1$ is minimized, i.e., we want to solve the following optimization problem

$$\min_{|\mathcal{V}_1|=k} \sum_{v\in\mathcal{V}_1} \|v - \frac{\sum_{v\in\mathcal{V}_1} v}{k}\|^2. \tag{4.2}$$

Here $|\mathcal{V}_1|$ denotes the cardinality of the subset $\mathcal{V}_1$.

We next discuss how to transfer problem(4.1) into another equivalent problem of form(4.2). To start, let us recall that Q is a matrix with zero diagonals. Therefore, there exists a constant λ satisfying $Q+\lambda I \succeq 0$. One obvious choice is $\lambda = -\lambda_{\min}(Q)$, which requires to compute the minimal eigenvalue $\lambda_{\min}(Q)$ first. In this paper we propose to select a sufficiently large constant λ such that the matrix $Q + \lambda I$ is diagonal dominant and strictly positive definite. We then perform the Cholesky decomposition on the matrix $Q + \lambda I$ such that

$$Q + \lambda I = V^T V \succ 0, \quad V = [v_1, v_2, \cdots, v_n]. \tag{4.3}$$

It follows immediately that

$$\|v_i\|^2 = \lambda, \quad i = 1, \cdots, n.$$

In other words, through the above process, we have constructed a data set whose data points are located on the surface of a sphere with radius $\sqrt{\lambda}$ in space $\Re^n$:

$$\mathcal{V} = \{v_i \in \Re^n : \|v_i\|^2 = \lambda, i = 1, \cdots, n\}, \quad Q_{ij} = v_i^T v_j, \forall i \neq j = 1, \cdots, n(4.4)$$

Now we are ready to state the main result in this section.

Theorem 4.1.1. *Suppose $\mathcal{V}$ is a data set defined by 4.4. Then the two problems4.1 and 4.2 are equivalent in the sense that from a global optimal solution of problem(4.1), one can derive a global optimal solution of problem4.2, and vice verse.*

Proof: To prove the theorem, we note that for any given subset $\mathcal{V}_1$ of size k, we can rewrite the objective function in problem4.2 as the following

$$\begin{aligned} f(\mathcal{V}_1) &= \sum_{v\in\mathcal{V}_1} \|v - \frac{\sum_{v\in\mathcal{V}_1} v}{k}\|^2 \\ &= \sum_{v\in\mathcal{V}_1} \|v\|^2 - \frac{1}{k}\|\sum_{v\in\mathcal{V}_1} v\|^2 \\ &= (k-1)\lambda - \frac{1}{k}\sum_{v_i\neq v_j\in\mathcal{V}_1} v_i^T v_j, \end{aligned} \tag{4.5}$$

where the last equality follows from4.3. Now let us define a binary vector x by

$$x \in \{0,1\}^n : x_i = 1 \text{ if and only if } v_i \in \mathcal{V}_1. \tag{4.6}$$

It follows from the above definition and 4.4 that

$$f(\mathcal{V}_1) = (k-1)\lambda - \frac{1}{k}x^T Q x,$$

which further concludes the proof of the theorem. □

Before closing this subsection, we would like to point out that although we considered only problem(4.1) in this subsection, the above reformulation scheme can be changed slightly to handle a more generic case where the objective function is $x^TQx + q^Tx$. To see this, we first introduce a new matrix Q_1 and lift x to higher dimensional space as below

$$Q_1 = \begin{pmatrix} 0 & \frac{1}{2}q^T \\ \frac{1}{2}q & Q \end{pmatrix}, \quad \tilde{x} = \begin{pmatrix} 1 \\ x \end{pmatrix}. \tag{4.7}$$

It follows

$$x^TQx + q^Tx = \tilde{x}^T Q_1 \tilde{x}.$$

Consequently, we can solve the following binary QP

$$\begin{aligned} \max \quad & \tilde{x}^T Q_1 \tilde{x} \qquad (4.8) \\ s.t. \quad & \sum_{i=1}^{n+1} = k+1 \\ & \tilde{x}_1 = 1, \quad \tilde{x}_i \in \{0,1\} \forall i = 2, \cdots, n+1. \end{aligned}$$

Recall that in such a case, we can assume without loss of generality that Q_1 has only zero diagonal elements. Following a similar vein as in our earlier discussion, we can assume

$$Q_1 + \lambda I = V_1^T V_1, \quad V_1 = [v_1^1, \cdots, v_{n+1}^1]$$

for a sufficiently large $\lambda > 0$. Similarly one can show that solving problem4.8 amounts to find a solution of the following clustering problem

$$\min_{|\mathcal{V}_1|=k+1, v_1^1 \in \mathcal{V}_1} \sum_{v^1 \in \mathcal{V}_1} \|v^1 - \frac{\sum_{v^1 \in \mathcal{V}_1} v^1}{k+1}\|^2. \tag{4.9}$$

4.2 A 2-approximation algorithm

In the last subsection, we have introduced a reformulation scheme to transfer problem3.2 into a specific clustering problem. In this subsection, we discuss how to solve problem4.2. To start, we first note that mathematically speaking, problem4.2 can be written as the following bi-level optimization problem

$$\min_{c} \min_{|\mathcal{V}_1|=k} \sum_{v \in \mathcal{V}_1} \|v - c\|^2. \tag{4.10}$$

Like the algorithm in [31], we can solve the above problem in an iterative manner by subsequently updating c and the subset $\mathcal{V}_1$ as follows.

Procedure 1

S1.0 Randomly choose an initial starting point $c \in \Re^n$;

S1.1 Sort all the distances $\|v_i - c\|$ for all the data points v_i and select the first k (or $k-1$ if $c \in \mathcal{V}_1$) shortest distance points as the subset $\mathcal{V}_1$;

S1.2 Computer the geometric center c_1 of $\mathcal{V}_1$;

S1.3 If $c_1 = c$, then stop and output $\mathcal{V}_1$ as the solution; otherwise set $c = c_1$ and go back to S1.1.

Following a similar idea as in [32], we can use any data point in $\mathcal{V}$ as the initial starting point in the above procedure, which leads to the following procedure.

Procedure 2

S2.0 For $i = 1 : n$ do

S2.1 Use v_i as the initial starting point and run Procedure 1;

S2.2 Set the final objective value from iterative Procedure 1 as $f^*(v_i)$;

S2.3 Find an index $i_0 = \arg\min_{i=1,\cdots,n} f^*(v_i)$ and select the corresponding subset as the final output.

We have

Theorem 4.2.1. *Suppose that $\mathcal{V}_1^*$ is the global optimal solution of problem4.2 with an objective value $f(\mathcal{V}_1^*)$ and $\mathcal{V}_1$ is the solution output by Procedure 2 with an objective value $f(\mathcal{V}_1)$. Then it holds*

$$f(\mathcal{V}_1) \leq 2f(\mathcal{V}_1^*).$$

Proof. Let us assume that $\mathcal{V}_1^* = \{v_1^*, \cdots, v_k^*\}$ with a geometric center c^*, and $\mathcal{V}_1 = \{v_1', \cdots, v_k'\}$ centered around $\bar{c}^*$. Let

$$v_0^* = \arg\min_{i=1,\cdots,k} \|v_i^* - c^*\|.$$

It follows immediately

$$\sum_{i=1}^k \|v_i^* - v_0^*\|^2 = \sum_{i=1}^k \|v_i^* - c^*\|^2 + k\|v_0^* - c^*\|^2 \leq 2\sum_{i=1}^k \|v_i^* - c^*\|^2.$$

Since $v_0^* \in \mathcal{V}$, it must have been used as an initial starting center in Procedure 2. Recall that $\mathcal{V}_1$ is the solution obtained from Procedure 2, it follws directly

$$f(\mathcal{V}_1) \leq \sum_{i=1}^k \|v_i^* - v_0^*\|^2 \leq 2f(\mathcal{V}_1^*). \qquad \square$$

We would like to point out that though a 2-approximation can be provided by Procedure 2, the obtained solution might be still far away from the global optimal solution of problem4.2. In order to obtain a better approximation to problem4.2, we suggest to incorporate a local search heuristics in Procedure 1. This is done by changing S1.3 in Procedure 1 as follows.

- Modified S1.3 of Procedure 1

S1.3' If $c_1 \neq c$, go back to S1.1; If $c_1 = c$, we search the subset $\mathcal{V}_1$ and its complement $\bar{\mathcal{V}}_1 = \mathcal{V} - \mathcal{V}_1$ to find the data point pairs $v \in \mathcal{V}_1$ and $\bar{v} \in \bar{\mathcal{V}}_1$ such that if we replace v by $\bar{v}$, then the new objective value after the replacement is reduced. If such a pair is found, then we update the subset $\mathcal{V}_1$ and its geometric center correspondingly. We go back to S1.1 after the search is done. In case no such pairs is found, output the current subset $\mathcal{V}_1$ as a solution;

Remark: The local search procedure in S1.3' is rather expensive since we need to search the space of all possible replacements. To improve the practical efficiency of the algorithm, we suggest to impose an upper bound on the number of expensive searches. In our implementation, we restrict us to only 20 expensive searches for large scale problems.

We also point out that all the procedures in this subsection can be modified to deal with the specific clustering problem4.9. With a certain effort, one can also establish similar theoretical results as in Theorem4.2.1. The technical details of such a process are left to interested readers.

4.3 Numerical results

In this section we report some experimental results based on the proposed algorithm. To check whether the proposed algorithm can find the global solution or a solution close to it, we compare the proposed algorithm with the BARON solver. Baron a global optimization solver based on the branch-and-reduce method supported by GAMS [33]. Since the GAMS/BARON solver is available on the NEOS Server [34], we compute it directly on the NEOS Server Version 5.0. In our experiments, we adopted GAMS Distribution 23.0 and GAMS/BARON 7.2.5. Note that the running time of BARON solver based on the powerful Neos server, while our algorithm is run on a desktop described in Section 2.2. Therefore, the comparison of CPU time is not justified in such a case.

We also compare the proposed algorithm with a heuristics for solving densest k subgraph problem suggested by Asahiro et al in [35]. As observed in [36], such a heuristics works for general weight function and is more stable than other heuristics. The heuristics in [35] first find a vertex in the graph with the minimum weighted degree under the given weight function and remove it from the graph until only k vertices are left. Then it performs a local search as follows:

Heuristics[**35**]

1. Use the greedy algorithm to find a subgraph $G_s = (V_s, E_s)$, which is a k-vertex induced subgraph of G;

2. Compute the degree D_{min} of the least heavy vertex v_{min} in G_s and remove it from G_s. Now $G_{s1} = (V_{s1}, E_{s1})$, where $V_{s1} = V_s \backslash v_{min}$.

3. Compute the number of connections in G between each vertex in $V \backslash V_s$ and V_{s1}. Let C_{max} be the maximum of all such numbers and v_{max} the associated vertex.

4. If $C_{max} > D_{min}$ then add vmax to V_{s1}. Let $V_s = V_{s1}$, and go to 2. Otherwise, output G_s.

Since the above heuristics is very efficient, we also incorporate it into the first stage of the proposed algorithm in this paper to select the initial cluster. We first compare all the three algorithms on some random problems where the symmetric matrix is randomly generated by the so-called Prime modulus multiplicative linear congruential generators where each element is uniformly distributed between 0 and 1. The numerical results are listed in Table 4.1.

As one can see from Table4.1, the solution obtained from the proposed algorithm is very close to the solution provided by BARON. For several test problems, the solutions derived from the proposed algorithm are even better that the solutions from BARON (highlighted in the table). It should be pointed out that BARON failed to solve problems whose size equal 1000 or above.

We next report our experimental results on graphes with (0,1) weight. For such a purpose, we first generate a random matrix by the Prime modulus multiplicative linear congruential generators. Then we reset the values of the elements of random matrix as follows: if the value of an element is larger than 0.65, we reset it to be 1, otherwise 0. The numerical results are summarized in Table4.2

As illustrated by Table 4.2, the solutions obtained from the proposed algorithm are very close to the solutions provided by BARON and sometimes even better.

We also point out that as shown in Tables4.1 and 4.2, the heuristics in [35] is able to find very good solutions effectively. This is possibly due to the usage of random data and the following fact that as shown in [35], the heuristics can provide an approximate solution with a ratio around $2n/k$ for $k \leq n/3$ (See also [36]). To illustrate the difference between the proposed algorithm and the heuristics in [35], we generate another set of testing problems as follows. We first generate a random matrix Q_1 that might be a general (or 0-1) weight matrix of size n associated with a graph G_1 and compute the minimum summation of all the rows in Q_1 denoted by $minsum(Q_1)$. Then we set k to be the largest integer less than $minsum(Q_1) - 1$ and associate it with a completely connected graph G_2 whose edges have weight 1. Now let us consider the joint graph $G = G_1 \cup G_2$ and consider the problem of finding the densest k-subgraph in G, which has an obvious solution G_2. Since there is no connections between the two parts (G_1 and G_2) of G, the heuristics in [35] will remove the dense subgraph G_2 first and then find a k-subgraph in G_1. However, the proposed algorithm in this paper will find the real optimal solution G_2 easily. For numerical comparison, we reuse the test data sets in Tables 4.1 and 4.2 and associate them with graph G_1 in the above construction. We then compute k based on G_1 and construct a complete subgraph G_2. Table 4.3 lists the solutions obtained from the proposed algorithm in this paper and the heuristics in [35]. As one can see from Table 4.3, for this specific class of graphes, the solutions from the heuristics are not as good as the solutions provided by the algorithm presented in this paper.

Problem		BARON		Cluster Alg.		Heuristic Alg.	
no.	k	Obj.	Time	Obj.	Time	Obj.	Time
M100_1	25	394.375	0.003	390.32	0.284	390.323	0.215
M100_1	50	1410.403	0.001	1407.716	0.57	1407.716	0.225
M100_2	20	267.483	0.03	265.804	0.258	265.579	0.307
M100_2	40	938.357	0.036	938.357	0.447	938.151	0.277
M300_1	50	1503.247	0.008	**1527.727**	3.166	1495.985	0.535
M300_1	100	5518.131	0.002	**5587.169**	7.538	5505.820	0.524
M300_1	150	11883.081	0.038	**12053.459**	14.997	11879.158	2.379
M500_1	100	5719.759	0.231	**5734.972**	19.613	5707.361	2.069
M500_1	200	21438.225	0.01	21437.491	53.611	21421.953	1.85
M750_1	100	5783.016	0.077	**5846.048**	46.314	5846.048	4.501
M750_1	200	21833.556	0.08	**21880.064**	132.59	21867	4.067
M750_1	300	47713.9	0.109	**47761.029**	283.58	47714.503	3.870
M1000_1	100	N/A	N/A	5918.5	85.057	5914.295	8.237
M1000_1	200	N/A	N/A	22119	217.79	22103.611	8.294
M1000_1	300	N/A	N/A	48338	419.19	48326.983	8.005
M1000_1	400	N/A	N/A	84381	715.42	84352.284	8.571
M1000_1	500	N/A	N/A	130046.323	1085.6	130021.577	7.933

Table 4.1: **Densest k-subgraph model for BQP problem: general weight**

Problem		BARON		Cluster Alg.		Heuristic Alg.	
no.	k	Obj.	Time	Obj.	Time	Obj.	Time
MB100_1	25	368	0.001	356	0.266	352	0.061
MB100_1	40	807	0.03	799	0.403	799	0.078
MB100_1	50	1165	0.004	1165	0.543	1165	0.072
MB250_1	50	1325	0.008	1321	1.994	1296	0.416
MB250_1	75	2675	0.002	2648	3.256	2648	0.293
MB250_1	100	4416	0.03	4414	5.578	4383	0.270
MB250_1	125	6542	0.003	**6545**	7.902	6545	0.278
MB500_1	100	4688	0.004	**4732**	17.746	4708	0.8522
MB500_1	150	9726	0.004	**9790**	32.68	9788	0.675
MB500_1	200	16470	0.004	**16498**	54.068	16431	0.6942
MB500_1	250	24763	0.11	24750	80.401	24744	0.694
MB750_1	100	4865	0.178	**4926**	45.385	4925	2.488
MB750_1	200	17219	0.006	**17263**	131.72	17232	2.401
MB750_1	300	36264	0.053	36241	265.45	36233	2.054
MB1000_1	10	90	0.007	**91**	38.525	86	5.611
MB1000_1	20	295	0.008	**307**	40.26	296	5.604
MB1000_1	50	1425	0.120	**1462**	49.388	1457	5.652
MB1000_1	100	4994	0.008	4966	81.873	4938	4.631
MB1000_1	200	17378	0.008	**17396**	207.56	17377	4.598
MB1000_1	300	36745	0.008	**36850**	422.45	36786	5.248
MB1000_1	400	62853	0.037	**62925**	719.3	62824	4.758
MB1000_1	500	95437	0.008	95387	1091.7	95340	4.582

Table 4.2: **Densest k-subgraph model for BQP problem: binary case**

Problem		Cluster Alg.		Heuristic Alg.	
no.	k	Obj.	Time	Obj.	Time
M100_1*	11	110	0.273	68	0.165
MB100_1*	24	552	0.477	326	0.144
M100_2*	8	56	0.322	40	0.188
MB250_1*	66	4290	6.661	2118	0.280
M300_1*	40	1560	4.780	630	0.814
MB500_1*	142	20022	81.225	8856	1.301
M500_1*	69	4692	22.976	1628	2.4046
MB750_1*	224	49952	333.71	21159	3.275
M750_1*	118	13806	112.89	4246	9.0429
MB1000_1*	307	93942	827.66	38304	6.896
M1000_1*	158	24806	278.09	7182	15.419

Table 4.3: **Densest k-subgraph model for BQP problem: special case**

CHAPTER 5

CORE-SET TECHNIQUE FOR BQP PROBLEM

Clustering problems have wide applications in the areas of data compression, machine learning, data mining, VLSI design and so on. The goal is to partition a set of objects into groups, where similar objects grouped together. Different cost functions are used to guide the clustering. Formally, given a set of n points P in $\Re^d$, we want to find a set of k points K (unnecessarily in P) s.t. $\mathrm{cost}(P, K)$ is minimized:

In the k-Means problem, $\mathrm{mean}_{\mathrm{OPT}}(P, k) = \min_{K \subseteq P, |K|=k} \mathrm{cost}(P, K)$, where

$$\mathrm{cost}(P, K) = \mathrm{mean}(P, K) = \sum_{p \in P} \mathrm{dist}(p, K)^2.$$

In the k-Center problem, $\mathrm{cen}_{\mathrm{OPT}}(P, k) = \min_{K \subseteq P, |K|=k} \mathrm{cost}(P, K)$, where

$$\mathrm{cost}(P, K) = \mathrm{cen}(P, K) = \max_{p \in P} \mathrm{dist}(p, K).$$

In the k-Median problem, $\mathrm{med}_{\mathrm{OPT}}(P, k) = \min_{K \subseteq P, |K|=k} \mathrm{cost}(P, K)$, where

$$\mathrm{cost}(P, K) = \mathrm{med}(P, K) = \sum_{p \in P} \mathrm{dist}(p, K).$$

In the following, if not specified, for two points x and y in $\Re^d$, let $\mathrm{dist}(x, y) = \|x - y\|_2$ (the euclidean distance); for two point sets X and Y, let

$$\mathrm{dist}(X, Y) = \min_{x \in X, y \in Y} \mathrm{dist}(x, y).$$

If $X = \{x\}$ or $Y = \{y\}$, we also write $\mathrm{dist}(X, Y)$ as $\mathrm{dist}(x, Y)$ or $\mathrm{dist}(X, y)$. If $K = \{x\}$, we also simplify notations as $\mathrm{cost}(P, x), \mathrm{mean}(P, x), \mathrm{cen}(P, x)$, and $\mathrm{med}(P, x)$.

We are most interested in the Densest-kPoints problem for its close connection to the Binary Quadratic Programming problem, which has be demonstrated in the previous section. Formally, given a set of n points P in $\Re^d$, we want to find a set of k points $C \subseteq P$ s.t. $\text{var}(C)$ is minimized:

In the Densest-kPoints-Mean problem, $\text{var}(C)$ is the optimal 1-means of C:

$$\text{var}(C) = \text{mean}_{\text{OPT}}(C, 1) = \min_{x \in \Re^d} \text{mean}(C, x).$$

Similarly, in the Densest-kPoints-Center problem,

$$\text{var}(C) = \text{cen}_{\text{OPT}}(C, 1) = \min_{x \in \Re^d} \text{cen}(C, x).$$

In the Densest-kPoints-Median problem,

$$\text{var}(C) = \text{med}_{\text{OPT}}(C, 1) = \min_{x \in \Re^d} \text{med}(C, x).$$

Given a set of points P, let $\text{avg}(P) = \frac{\sum_{p \in P} p}{|P|}$ be the *centroid* of P. We can easily get:

$$\text{mean}(P, x) = \text{mean}(P, \text{avg}(P)) + |P| \cdot \text{dist}(\text{avg}(P), x)^2.$$

So we have the following lemma which will be used later.

Lemma 5.0.1. *For any set P of n points in $\Re^d$, we have* $\text{mean}_{\text{OPT}}(P, 1) = \text{mean}(P, \text{avg}(P))$.

In the following subsection 5.1, we will do a survey on using core-sets techniques for clustering. In the Subsection 5.2, we will demonstrate the connection between the Binary Quadratic Programming problem and the Densest-kPoints-Mean problem. We propose three classes of algorithms for the three Densest-kPoints problems in Subsection **??**. In particular, in Subsection 5.2.2, we will apply the core-sets techniques to get $(1 + \epsilon)$-approximations.

5.1 A Survey: clustering using core-sets

A class of techniques to find good clustering is: given n points P in $\Re^d$, using either construction or sampling, we find small point sets S s.t. $|S| \ll n$ (say, $O(\log n)$ or some constant parameter independent on n); the task of clustering P can be (somehow) guided by the much smaller sets S, and results in efficient $(1+\epsilon)$-approximation algorithms.

This class of techniques have been successfully applied to find $(1+\epsilon)$-approximation to the k-Center/k-Median/k-Means problems [37, 38, 39, 40] and even when the distance measure is NOT metric [41, 42]. We call this class of methods "Core-Set" techniques (named by [38]). We will first give a survey on how core-sets is used for clustering in this subsection.

First we list some important papers in this topic: [37] is the first work that uses randomized core-sets to find good k-Means clustering, but it requires the clustering to be "balanced". [38] is the first one that introduces core-set techniques for k-Center and k-Median. [43] introduces core-set for projective clustering. [39] proposes faster core-set algorithms for k-Median and k-Means in data streaming environment. [44] discovers smaller core-sets for k-Median and k-Means. [45] discovers smaller core-sets for k-Center. [40] extends to techniques in [37] to find good k-Means clustering in linear time. Some other followup works include [46, 47, 48]. Two recent works [41, 42] show that core-set techniques can be applied even with some non-metric distance measures (which satisfies some properties).

Consider a set of n points P in $\Re^n$, one can first observe that $\text{mean}_{\text{OPT}}(P, 1)$ has intrinsic statistical meanings, that is (recall lemma 5.0.1),

$$\frac{\text{mean}_{\text{OPT}}(P,1)}{n} = \frac{\sum_{p\in P} \|p - \text{avg}(P)\|^2}{n} \approx \frac{n-1}{n}\mathbf{Var}[X]$$

is a good estimate of $\mathbf{Var}[X]$ if P are n points generated from distribution X $(d=1)$.

Ideally, sampling a set of m points $S \subseteq P$ by independent draws at random from P, $\frac{n}{m-1}\text{mean}_{\text{OPT}}(S,1)$ could be a good estimate for $\text{mean}_{\text{OPT}}(P,1)$. But this is NOT true. For example, suppose P has $n-1$ points in the same position, and one point far away from them, then $\text{mean}_{\text{OPT}}(S,1)$ is almost zero with high probability.

Fortunately, $\text{avg}(S)$ is a good estimate of $\text{avg}(P)$; we can consider using $\text{mean}(P, \text{avg}(S))$ to estimate $\text{mean}(P, \text{avg}(P)) = \text{mean}_{\text{OPT}}(P,1)$. Inaba, Katoh, and Imai [37] use this observation to design a randomized core-set algorithm for the k-Means problem.

Lemma 5.1.1 ([37]). *Sampling a set of m points S from a set of n points P*

in $\Re^d$ using m independent draws at random, for any $\delta > 0$,

$$\|\text{avg}(S) - \text{avg}(P)\|^2 < \frac{1}{\delta m}\left(\frac{\sum_{p\in P}\|p - \text{avg}(P)\|^2}{n}\right)$$

with probability $1 - \delta$.

Proof Sketch. Observe that

$$\mathbf{E}\left[\text{avg}(S)\right] = \text{avg}(P), \quad \mathbf{E}\left[\|\text{avg}(S) - \text{avg}(P)\|^2\right] = \frac{1}{m}\left(\frac{\sum_{p\in P}\|p - \text{avg}(P)\|^2}{n}\right)$$

and use the Markov inequality. □

Lemma 5.1.2 ([37])**.** *Sampling a set of m points S from a set of n points P in $\Re^d$ using m independent draws at random, for any $\delta > 0$,*

$$\text{mean}(P, \text{avg}(S)) < \left(1 + \frac{1}{\delta m}\right)\text{mean}(P, \text{avg}(P)) = \left(1 + \frac{1}{\delta m}\right)\text{mean}_{\text{OPT}}(P, 1)$$

with probability $1 - \delta$.

Proof Sketch. Recall lemma 5.0.1,

$$\text{mean}(P, \text{avg}(S)) = \text{mean}(P, \text{avg}(P)) + |P| \cdot \|\text{avg}(P) - \text{avg}(S)\|^2,$$

and use Lemma 5.1.2. □

Now we present the algorithm by Kumar, Sabharwal, and Sen [40] for finding good k-Means clustering use the same core-set, the sampling set S, but of course, a different (and more complicated) clustering algorithm. We focus on 2-Means clustering first. Recall $P(c_i^*)$ are the points which are more closer to c_i^* than c_j^* $(j \neq i)$ in a point set P, for $i = 1, \ldots, k$. Algorithms in [40] are based on the following observation: (i) if P is very "unbalanced" (i.e. one of $|P(c_1^*)|$ and $|P(c_2^*)|$ is very small), then we can reduce 2-Means clustering to 1-Means clustering; (ii) at least one of $|P(c_1^*)|$ and $|P(c_2^*)|$ is larger than $|P|/2$. Observation (i) leads to the following definition. (Recall $P(c_i^*)$ is the set of points which are more closer to c_i^* than c_j^* $(j \neq i)$.)

Definition 5.1.3 (Mean-Reducible)**.** *We say the point that set P is (k, ϵ)-irreducible if* $\text{mean}_{\text{OPT}}(P, k-1) \geq (1 + 32\epsilon)\text{mean}_{\text{OPT}}(P, k)$*. Otherwise, we* *say it is* (k, ϵ)*-reducible.*

Let $\alpha = \epsilon/64$. We assume P is $(2, \alpha)$-irreducible. Otherwise, directly from the definition, we can get an $(1+\epsilon)$-approximation to 2-Means using 1-Means.

We sample a point set S of size $O(\frac{1}{\epsilon})$, say $|S| = \frac{4}{\beta\epsilon}$, from P. Consider an optimal 2-Means clustering $K = \{c_1^*, c_2^*\}$ for P. Without loss of generality, assume that $|P(c_1^*)| \geq |P|/2$. We can show that $|S(c_1^*)| \geq \frac{2}{\epsilon}$ with high probability. We can guess $S(c_1^*)$ (to be precise, its subset of size $\frac{2}{\epsilon}$—cycling through all such subsets of S), and from Lemma 5.1.2, we can assume that we have found $c_1 = \mathrm{avg}(S(c_1^*))$ s.t.

$$\mathrm{mean}(P(c_1^*), c_1) \leq (1+\alpha)\mathrm{mean}(P(c_1^*), c_1^*).$$

Lemma 5.1.4 ([40])**.** *Let* $\mathrm{dist}(c_1^*, c_2^*) = t$*. We have* c_1^* *and* c_1 *are closed:* $\mathrm{dist}(c_1^*, c_1) \leq t/4$.

Proof. Otherwise ($\mathrm{dist}(c_1^*, c_1) > t/4$), from lemma 5.1, we have

$$\begin{aligned} \alpha \cdot \mathrm{mean}(P(c_1^*), c_1^*) &\geq \mathrm{mean}(P(c_1^*), c_1) - \mathrm{mean}(P(c_1^*), c_1^*) && (5.1) \\ &= |P(c_1^*)|\mathrm{dist}(c_1^*, c_1)^2 \geq \frac{t^2|P(c_1^*)|}{16}; && (5.2) \end{aligned}$$

and thus,

$$\begin{aligned} \mathrm{mean}(P(c_1^*), c_2^*) &= \mathrm{mean}(P(c_1^*), c_1^*) + |P(c_1^*)|t^2 && (5.3) \\ &\leq (1+16\alpha)\mathrm{mean}(P(c_1^*), c_1^*). && (5.4) \end{aligned}$$

So, $\mathrm{mean}(P, c_2^*) \leq (1+16\alpha)\mathrm{mean}(P, \{c_1^*, c_2^*\})$, i.e., P is $(2, \alpha)$-reducible (contradiction). □

The previous lemma implies the ball $\mathcal{B}(c_1, t/4)$ (of radius $t/4$ centered at c_1) is contained in $\mathcal{B}(c_1^*, t/2)$. So, $\mathcal{B}(c_1, t/4) \cap P \subseteq P(c_1^*)$. We are now interested in c_2^* and $P(c_2^*)$. So we expect $P' = P - \mathcal{B}(c_1, t/4)$ has a good fraction of points from $P(c_2^*)$. Actually, it is true.

Lemma 5.1.5 ([40])**.** *Let* $P_1' = P(c_1^*) - \mathcal{B}(c_1, t/4)$*. Then* $P' = P_1' \cap P(c_2^*)$ *and* $|P(c_2^*)| \geq \alpha|P_1'|$.

Proof. The intuition is: if $P(c_2^*)$ is small in P' still, c_1^* could be a good 1-Means solution.

Suppose $|P(c_2^*)| \leq \alpha |P_1'|$. Notice

$$\text{mean}(P(c_1^*), c_1) \geq \text{mean}(P_1', c_1) \geq \left(\frac{t}{4}\right)^2 |P_1'| = t^2|P_1'|/16 \tag{5.5}$$

. Since $\text{mean}(P(c_1^*), c_1) \leq (1+\alpha)\text{mean}(P(c_1^*), c_1^*)$, we have

$$t^2|P(c_2^*)| \leq \alpha t^2 |P_1'| \leq 16\alpha(1+\alpha)\text{mean}(P(c_1^*), c_1^*) \tag{5.6}$$

and thus

$$\begin{aligned} \text{mean}(P, c_1^*) &= \text{mean}(P(c_1^*), c_1^*) + \text{mean}(P(c_2^*), c_2^*) + t^2|P(c_2^*)| \qquad (5.7) \\ &\leq (1+32\alpha)\text{mean}(P, \{c_1^*, c_2^*\}), \qquad (5.8) \end{aligned}$$

i.e., P is $(2, \alpha)$-reducible (contradiction). □

So the algorithm proceeds as follows: we sample a point set S' of size $O(\frac{1}{\alpha^2})$ from P'. Again, we can show $|S'(c_2^*)| \geq \frac{2}{\alpha}$ with high probability. So we cycle through all subsets of size $\frac{2}{\alpha}$, and can find a point c_2 s.t. $\text{mean}(P(c_2^*), c_2) \leq (1+\alpha)\text{mean}(P(c_2^*), c_2^*)$. Therefore, $K = \{c_1, c_2\}$ is an $O(1+\alpha)$-approximation to the 2-Means problem.

The only problem is that we do not know $\text{dist}(c_1^*, c_2^*) = t$, and thus cannot identify P' (as well as sample S' from P'). However, we can find c_1 without the knowledge about t, and guess the parameter i s.t. $\frac{n}{2^i} \leq |P'| \leq \frac{n}{2^{i-1}}$. Let P'' be the $\frac{n}{2^{i-1}}$ farthest points from c_1 in P, and we know $P'' \supseteq P', |P''| \leq 2|P'|$. We increase the sample size $|S'|$ (from P'') a bit, and can get the same result. This algorithm needs $O(\log n)$ iterations (guesses i). More involved analysis shows the running time is linear in n ($|P''|$ decreases by half in every iteration).

Theorem 5.1.6 ([40])**.** *Given a point set P of size n in $\Re^d$, we can find $(1+\epsilon)$-approximation to the optimal 2-Means with constant probability in time $O(2^{(1/\epsilon)^{O(1)}} nd)$. Similar idea applies to k-Means clustering with running time $O(2^{(k/\epsilon)^{O(1)}} nd)$.*

5.1.1 Core-sets for k-center clustering

Similar to the core-set for k-Means, given a set of n points P in $\Re^d$, we want to find a small set $S \subseteq P$ s.t. the 1-Center of S can be used to approximate the 1-Center of P (somehow). Because of some geometric properties, the core-set S for k-Center is stronger than the one for k-Means (recall $\text{mean}_{\text{OPT}}(S,1)$ cannot be used to estimate $\text{mean}_{\text{OPT}}(P,1)$, but, here, $\text{cen}_{\text{OPT}}(S,1)$ approximates $\text{cen}_{\text{OPT}}(P,1)$); however, S here cannot be obtained by sampling from P (while most samplings are good), but it can be obtained in a constructive way.

For a point set P, let $c_P = \arg\min_x \text{cen}(P,x)$, i.e., $\text{cen}_{\text{OPT}}(P,1) = \text{cen}(P,c_P)$ (c_P is the minimum enclosing ball of P). c_P could be found using convex programming techniques. We also write $\text{cen}_{\text{OPT}}(P,1)$ as r_P (the radius of the minimum enclosing ball of P).

Lemma 5.1.7 ([38]). *There is a subset of points $S \subseteq P$ with $|S| = O(\frac{1}{\epsilon^2})$ s.t. $\text{cen}_{\text{OPT}}(S,1) \geq \text{cen}_{\text{OPT}}(P,1)/(1+\epsilon)$ and*

$$\text{cen}(P,c_S) \leq (1+\epsilon)\text{cen}(S,c_S) \leq (1+\epsilon)\text{cen}(P,c_P) = (1+\epsilon)\text{cen}_{\text{OPT}}(P,1).$$

Proof. We start constructing S with $S_0 = \{x,y\}$ s.t. $x,y \in P$ and $\text{dist}(x,y) \geq r_P$. x is the furthest point away from y in P. Clearly, $r_P/2 \leq r_{S_0} \leq r_P$.

There are two cases:

(i) If there is no point $p \in P$ s.t. $\text{dist}(p,c_{S_i}) \geq (1+\epsilon)r_{S_i}$, it is done: $S = S_i$.

(ii) If there is a point $p \in P$ s.t. $\text{dist}(p,c_{S_i}) \geq (1+\epsilon)r_{S_i}$, set $S_{i+1} = S_i \cup \{p\}$ and $i = i+1$.

We claim $r_{S_{i+1}} \geq (1+\epsilon^2/16)r_{S_i}$ and thus the above iteration can repeat at most $O(\frac{1}{\epsilon^2})$ times. Finally, S has size $O(\frac{1}{\epsilon^2})$ and satisfies the desired property.

This claim can be proved as follows.

If $\text{dist}(c_{S_i},c_{S_{i+1}}) < \epsilon r_{S_i}/2$, then by triangle inequality,

$$r_{S_{i+1}} \geq \text{dist}(p,c_{S_{i+1}}) \geq \text{dist}(p,c_{S_i}) - \text{dist}(c_{S_i},c_{S_{i+1}}) \geq \left(1+\frac{\epsilon}{2}\right) r_{S_i}.$$

If $\text{dist}(c_{S_i},c_{S_{i+1}}) \geq \epsilon r_{S_i}/2$, then let H be the $(d-1)$-dim hyperplane that passes through c_i and is orthogonal to $c_{S_i}c_{S_{i+1}}$. Let H^- be the open half-space having p inside. Then we know there is a point $x \in S_i$ in the closed half-space $\Re^d - H^-$ s.t. $\text{dist}(c_{S_i},x) = r_i$ (note the minimum enclosing ball of

S_i centered at c_{S_i}) [49]. Therefore,

$$r_{S_{i+1}} \geq \text{dist}(c_{S_{i+1}}, x) \geq \sqrt{r_{S_i}^2 + \frac{\epsilon^2}{4} r_{S_i}^2} \geq \left(1 + \frac{\epsilon^2}{16}\right) r_{S_i}.$$

So the proof is completed. □

The algorithm of 2-Center clustering is as follows. Suppose P is partitioned into two sets X and Y in the optimal solution. We start from two empty point sets S_X and S_Y as the core-sets for X and Y, respectively. In each of the following iterations, we pick a point p furthest away from $S_X \cup S_Y$. From a guessing oracle, if $p \in X$, then put p into S_X; otherwise, put p into S_Y. From the above lemma, after $O(1/\epsilon^2)$ iterations, we can get an $O(1+\epsilon)$-approximation for 2-Center clustering ($K = \{c_{S_X}, c_{S_Y}\}$).

To remove the guessing oracle, we enumerate all the possibilities, which needs $2^{O(1/\epsilon^2)}$. This algorithm can be also extended for general k, and the running time turns to be $k^{O(k/\epsilon^2)}$.

Theorem 5.1.8 ([38]). *For any point set P with size n in $\Re^d$ and $0 < \epsilon < 1$, an $(1+\epsilon)$-approximation of k-Center clustering for P can be found in $2^{O(k \log k/\epsilon^2)} nd$ time.*

Smaller core-set of size $O(1/\epsilon)$ for the k-Center clustering is found in [45].

5.1.2 Core-sets for k-Median clustering

The construction of core-sets of k-Median clustering is more complicated than the ones of k-Means and k-Center clustering. Badoiu, Har-Peled, and Indyk proposed an randomized construction in [38]. Note: the size of core-set for k-Median might be sublinearly dependent on n (say $O(\log n)$), but is still small enough for designing efficient algorithms on it.

For a set of n points P in $\Re^d$, let $\text{AvgMed}(P, k) = \text{med}_{\text{OPT}}(P, k)/|P|$, which can be interpreted as the average "radius" of the k-Median clustering.

Specifically, let $K^* = \{c^*\}$ be the optimal solution to 1-Median clustering $\text{med}_{\text{OPT}}(P, 1)$ on P. The following lemma says, we can find an $(1+\epsilon)$-approximation c' for $\text{med}_{\text{OPT}}(P, 1)$ in a space spanned by a small number of samples from P (also, c' and c^* are closed).

Lemma 5.1.9 ([38]). *Let H be a random sample of $O(1/\epsilon^3 \log 1/\epsilon)$ points from P. With constant probability, these two events happen: (i) The flat spanned by H, $\text{span}(H)$, contains a $(1+\epsilon)$-approximation 1-Median, c', for P, and (ii) H contains a point in distance $\leq 2\text{AvgMed}(P,1)$ from the center of the optimal solution, c^*.*

Proof Sketch. We skip the detailed proof here. But the main idea is similar to the one in Lemma 5.1.7. First, from the definition, we immediately have $\mathbf{E}\left[\text{dist}(s_i, c^*)\right] = \text{AvgMed}(P,1)$, if s_i is a point sampled uniformly from P (so (ii) is true). Let F_i be the flat spanned by the first i samples $s_1, s_2, \ldots, s_i$, i.e., $\text{span}(\{s_1, \ldots, s_i\})$, and c' be the projection of c^* on F_i. It can be shown $\text{dist}(c', c^*)$ shrinks very quickly as long as there are enough points in P NOT closed to F_i. Finally, either $\text{dist}(c', c^*)$ becomes small enough or most points are closed to F_i; in both cases, c' could be a good approximation to c^*, and $H = \{s_1, s_2, \ldots\}$. □

Before presenting the construction of core-sets for k-Median clustering, we assume:

(a) The distance between any two points in P is at least one;
(b) The optimal k-Median cost $\text{med}_{\text{OPT}}(P,k)$ is at most n^b for some $b = O(1)$.

If (a) or (b) is not true, we cover space by a grid of size $L\epsilon/(5nd)$, and snap points of P to this grid, where L satisfying $L/2 \leq \text{med}_{\text{OPT}}(P,k) \leq nL$ can be find using an 2-approximation algorithm for the k-Center clustering. The cost of any k-Median clustering in the new point set differs at most a factor of $(1+\epsilon/5)$ from the same one of P.

Now we present the construction of core-sets for k-Median clustering. The idea is based on (i) and (ii) in Lemma 5.1.9. Let's first assume we have found t s.t. $t/2 \leq \text{AvgMed}(P,1) \leq t$. Clearly, t can be found by checking $t = 2^i$ for $i = 0, 1, \ldots, O(\log n)$ because of (b).

Let H be a random sample of $O(1/\epsilon^3 \log 1/\epsilon)$ points from P. As in Lemma 5.1.9 (i), c' (the projection of c^* on $\text{span}(H)$) is a good approximation to the 1-Median solution for P. So our goal is to find a small set of points $S(P,H)$ (core-set), s.t. some of them is closed to c' and thus could be used to approximate 1-Median solution for P. From Lemma 5.1.9 (ii), some point in H is in distance $\leq 2t$ from c^* (and thus $\leq 2t$ from c'). Therefore, we construct a grid near each point of H to locate c', and the vertices of the grids form the set $S(P, H)$.

A bit more formally, let $\mathcal{G}_p(t)$ be a grid of side length $O(\epsilon t/|R|)$ centered at p on H, and let $\mathcal{B}(p, 2t)$ be a ball of radius $2t$ centered at p. Let $S'(p, t) = \mathcal{G}_p(t) \cap \mathcal{B}(p, 2t)$. Clearly, if $\mathrm{dist}(p, c^*) \leq 2t$, then c' falls into $\mathcal{B}(p, 2t)$, and thus some point in $S'(p, t)$ can be used as an $(1 + \epsilon)$-approximation to the 1-Median solution for P.

Finally, $S(P, H) = \bigcup_{i=0}^{O(\log n)} \bigcup_{p \in H} S'(p, 2^i)$ and
$|S(P, H)| = O\left(2^{O(1/\epsilon^4)} \log n\right)$.

Lemma 5.1.10 ([38]). *Let H be a random sample of $O(1/\epsilon^3 \log 1/\epsilon)$ points from P. One can compute a point set $S(P, H)$ of size $O\left(2^{O(1/\epsilon^4)} \log n\right)$, s.t. with high probability (over the choice of H), there is a point $q \in S(P, H)$ s.t.* $\mathrm{med}(P, q) \leq (1 + \epsilon)\mathrm{med}_{\mathrm{OPT}}(P, 1)$.

Using the ideas similar to the algorithm for k-Means described before, one may obtain "efficient" $(1 + \epsilon)$-approximation algorithms for k-Median clustering.

Theorem 5.1.11 ([38]). *Given a point set P of size n in $\Re^n$, we can find $(1 + \epsilon)$ approximation to the optimal 2-Median with high probability in time $O(2^{(1/\epsilon)^{O(1)}} d^{O(1)} n \log^{O(1)} n)$. Similar idea applies to k-Median clustering with running time $O(2^{(k/\epsilon)^{O(1)}} d^{O(1)} n \log^{O(k)} n)$.*

5.1.3 Core-sets for non-metric distance clustering

Recently, Ackermann, Blömer, and Sohler [41] and their followup work [42] extend the core-set techniques to the non-metric distance clustering. They show that if a (maybe *non-metric*) distance measure is $[\gamma, \delta]$-*sampleable*, the algorithm in [40] can be adapted to find $(1 + \epsilon)$-approximation to the *generalized* k-Median *problem* in linear time.

Note the only different between the generalized k-Median problem and the k-Median problem is that, in the generalized k-Median problem, *we do NOT require the distance measure to be a metric.* We allow $\mathrm{dist}(x, y) \neq \mathrm{dist}(y, x)$ and $\mathrm{dist}(x, y) + \mathrm{dist}(y, z) < \mathrm{dist}(x.z)$, but only require $\mathrm{dist}(x, y) = 0 \Leftrightarrow x = y$. So the k-Means problem is a special case of the generalized

K-MEDIAN problem if the distance measure is defined to be $\text{dist}(x, y) = \|x - y\|_2^2$.

The algorithm in [41] is nearly identical to the one in [40]. But, a significant difference between Ackermann *et al.*'s work and [40] is that the analysis of Ackermann *et al.*'s algorithms does NOT depend on the symmetry or the triangle inequality of distance measure, while previous works, like [40], do. Also, they discuss which (non-metric) distance measures are $[\gamma, \delta]$-sampleable, and construct core-sets for these measures.

Below, let c_P be the optimal 1-Median of any point set P, i.e. $\text{med}(P, c_P) = \text{med}_{\text{OPT}}(P, 1)$.

We first state the main result of [41]. Note $\text{dist}(x, y)$ is unnecessarily $\|x - y\|_2$ now.

Theorem 5.1.12. *Given an integer k and any $\epsilon < 1$. Assume that for $\delta < 1$ and $\beta = \epsilon/3$, distance measure* $\text{dist}(\cdot, \cdot)$ *satisfies:*

(a) For every finite point set S, an optimal 1-Median c_S, i.e. $\text{med}(S, c_S) = \text{med}_{\text{OPT}}(S, 1)$*, can be computed in time depending only on $|S|$.*

(b) There exists a constant $m_{\gamma,\delta}$ such that for every point set P of size n and for every uniform sample multiset $S \subseteq P$ of size $m_{\gamma,\delta}$, an optimal 1-Median c_S of S satisfies

$$\mathbf{Pr}\left[\text{med}(P, c_S) \leq (1 + \gamma)\text{med}_{\text{OPT}}(P, 1)\right] \geq 1 - \delta.$$

Then there exists an algorithm that with constant probability returns an $(1 + \epsilon)$-approximation of the K-MEDIAN *problem w.r.t.* $\text{dist}(\cdot, \cdot)$ *for input point set P of size n in time $O(n2^{(\frac{k}{\epsilon})^{O(1)}})$.*

(b) in the above theorem is called *superset sampling* or *core-set sampling* (for S is the so-called "core-set"). Their analysis is even simpler than the one in [40].

Let's restate the algorithm for $k = 2$. Let $K^* = \{c_1^*, c_2^*\}$ be the optimal 2-Median, and $P_i^* = P(c_i^*)$ be the cluster containing c_i^* with $|P_1^*| \geq \alpha|P|$ $(\alpha > 1/4)$.

1. (Superset sampling) Obtain c_1 from P with

$$\text{med}(P_1^*, c_1) \leq (1 + \gamma)\text{med}_{\text{OPT}}(P_1^*, 1)$$

.

2. Let N be the smallest subset of closest points from P towards c_1 s.t. for the remaining points $R = P/N$, we have $|P_2^* \cap R| \geq \alpha|R|$. Assign N to c_1. Note: $P_2^* \cap R = P_2^*/N$.
3. (Superset sampling) Obtain c_2 from R with

$$\text{med}(P_2^* \cap R, c_2) \leq (1+\gamma)\text{med}_{\text{OPT}}(P_2^* \cap R, 1)$$

.

4. Use $K = \{c_1, c_2\}$ as a 2-Median solution.

In 1 above, to obtain c_1, we take a sample multiset S of size $m_{\gamma,\delta}$ from P and enumerate all the $O(\frac{m_{\gamma,\delta}}{\alpha})$-subsets of S. Similar guessing oracle is used in 3. So if N is known, the running time is $O\left(n2^{(\frac{2}{\alpha}m_{\gamma,\delta})^{O(1)}}\right)$.

Of course, N is unknown. So N is approximated by partitioning P into $N^{(1)}$, $N^{(2)}$, ..., $N^{(\lceil \log n \rceil)}$. Here, $N^{(1)}$ is the $n/2$ closest points to c_1; $N^{(2)}$ is the next $n/4$ closest points to c_1; $N^{(3)}$ is the next $n/8$ closest points to c_1; .... Let $R^{(j)} = P/\bigcup_{i=1}^{j} N^{(i)}$, and let v be the minimal value s.t. $|P_2^* \cap R^{(v)}| \geq \alpha|R^{(v)}|$. We will approximate $N = N^{(1)} \cup N^{(2)} \cup \ldots \cup N^{(v)}$.

Then,

$$\text{med}(P, K) \leq \text{med}(P_1^*, c_1) + \text{med}(P_2^* \cap N, c_1) + \text{med}(P_2^*/N, c_2).$$

Using the following claims:

$$\begin{array}{rl} (i) & \text{med}(P_2^* \cap N, c_1) \leq 8\alpha \cdot \text{med}(P_1^*, c_1); \\ (ii) & \text{med}(P_1^*, c_1) \leq (1+\gamma) \cdot \text{med}(P_1^*, c_1^*); \\ (iii) & \text{med}(P_2^*/N, c_2) \leq (1+\gamma) \cdot \text{med}(P_2^*/N, c_2^*). \end{array}$$

(ii) and (iii) are because of the superset sampling technique. (i) is non-trivial but it is mainly because there are fewer points in $N^{(j)}$ from P_2^* ($\leq 2\alpha$) than from P_1^* ($> 1 - 2\alpha$), if $j \leq v$. Then we can conclude:

$$\begin{aligned} \text{med}(P, K) &\leq (1+8\alpha)(1+\gamma)\text{med}(P_1^*, c_1^*) + (1+\gamma)\text{med}(P_2^*/N, c_2^*) \quad (5.9) \\ &\leq (1+8\alpha)(1+\gamma)\text{med}_{\text{OPT}}(P, 2). \end{aligned}$$

In the case that N does not exist, or more precisely, $v = \lceil \log n \rceil$, we end up with a single point $R^{(v)} = \{q\}$. Let q itself forms a cluster with cost

0, and the above analysis is still valid. Therefore, this algorithm gives an $(1+8\alpha)(1+\gamma))$-approximation.

Constructing core-sets for non-metric distance. Given the algorithm above, the rest question is for which non-metric distance measures, the *superset sampling* technique is valid for finding good core-sets. [41, 42] introduce such sampling techniques for non-metric distance measures like the Kullback-Leibler divergence, Mahalanobis distance, Bregman divergance, etc. So the above algorithm is valid for a broad class of distance definitions.

5.2 Densest k-points v.s. BQP problem

The main reason leading us to study the DENSEST-KPOINTS-MEAN problem is its interesting connection to the Binary Quadratic Programming problem, which has been shown early the previous section.

Geometrically, the 2-approximation algorithm mentioned in previous section can be interpreted for DENSEST-KPOINTS problems ($\text{dist}(\cdot,\cdot)$ is a metric): given a set of n points P, for every point $p \in P$, let C_p be the k nearest points to p in P (including p itself); among the n choices of C_p's, we pick the one that minimize $\text{var}(C_p)$. It can be shown this is an 2-approximation algorithm for all the three versions of DENSEST-KPOINTS problems (depending on how $\text{var}(C_p)$ is defined): DENSEST-KPOINTS-MEAN, DENSEST-KPOINTS-CENTER, and DENSEST-KPOINTS-MEDIAN.

Theorem 5.2.1. *This algorithm gives 2-approximation to* DENSEST-KPOINTS *problems.*

Proof. It is straightforward to prove the performance guarantee for the Center version and Median version. Following is the proof for the Means version (a bit trickier).

Suppose the optimal solution is $C^* = \{c_1^*, c_2^*, \ldots, c_k^*\}$, $c^* = \text{avg}(C^*) = \frac{\sum_{c_i^* \in C^*} c_i^*}{k}$ is the centroid of C^*. And suppose $C' = \{c_1', c_2', \ldots, c_k'\}$ is the solution found by our algorithm, and v_0 among P minimizes minimize the distance to c^*, $\|v_0 - c^*\|_2$. Note $\sum_{c_i^* \in C^*}(c_i^* - c^*) = 0$. It is clear that we have

SOL $= \sum_{c'_i \in C'} \|c'_i - \mathrm{avg}(C')\|_2^2 \leq \sum_{c^*_i \in C^*} \|c^*_i - v_0\|_2^2$. So,

$$
\begin{aligned}
\mathrm{SOL} &\leq \sum_{c^*_i \in C^*} \|c^*_i - v_0\|_2^2 \\
&= \sum_{c^*_i \in C^*} \|(c^*_i - c^*) + (v_0 - c^*)\|_2^2 \\
&= \sum_{c^*_i \in C^*} (\|c^*_i - c^*\|_2^2 + 2(c^*_i - c^*)(v_0 - c^*) + \|v_0 - c^*\|_2^2) \\
&= \left(\sum_{c^*_i \in C^*} \|c^*_i - c^*\|_2^2 \right) + \left(2(v_0 - c^*)^T \sum_{c^*_i \in C^*} (c^*_i - c^*) \right) + k\|v_0 - c^*\|_2^2 \\
&= \left(\sum_{c^*_i \in C^*} \|c^*_i - c^*\|_2^2 \right) + k\|v_0 - c^*\|_2^2 \\
&\leq 2 \sum_{c^*_i \in C^*} \|c^*_i - c^*\|_2^2 = 2\mathrm{OPT} \quad .
\end{aligned}
$$

The last step is due to v_0 is the closed point to c^* in P. Proofs for the Center version and Median version are similar and simpler. □

5.2.1 $(1 + \epsilon)$-approximation algorithms using exponential grids

One idea to improve the approximation ratio 2 is to use the vertices in a grid to approximate the center of the optimal solution to DENSEST-KPOINTS problems. The problem is there might be too many vertices in the grid we need to check. We can extend the idea of [50] in high-dimension space to restrict the search space. Since this algorithm is extended from [50], we will only introduce the main ideas and state our main results below.

Given a set of n points P in $\Re^d$, we first use the algorithm introduced above to get an 2-approximation C'. Suppose SOL $= \mathrm{var}(C')$ and OPT $= \mathrm{var}(C^*)$, where C^* is the optimal solution, then we have SOL $\leq$ 2OPT. We know the radius of C^* (the distance from the center c^* of C^* to the farthest point in C^*) $\leq$ OPT (or $\sqrt{\mathrm{OPT}}$ for the Mean version).

So, we first use a grid of size SOL (or $\sqrt{\mathrm{SOL}}$ for the Mean version) to cover the point set P. At the first glance, there is an unbounded number of squares in this grid we need to consider (if the scale of P is unbounded).

But, we can observe that an optimal solution C^* may intersect with at most 3^d squares in this grid. Therefore, where are at most $n \cdot 3^d$ squares we need to consider.

For each of these squares we need to consider, we use a smaller grid of size $\frac{\epsilon \cdot \mathrm{SOL}}{2}$ (or $\frac{\epsilon \cdot \mathrm{SOL}}{2k}$ for the Median version, $\frac{\epsilon \cdot \sqrt{\mathrm{SOL}}}{\sqrt{2k}}$ for the Mean version) to cover it. We enumerate all the vertices in each small grid as the center of the solution, and output the best one.

Theorem 5.2.2. *Given a set of n points P, there are $(1+\epsilon)$-approximation algorithms with running time $O\left(\left(\frac{6}{\epsilon}\right)^d n^2\right)$ for* Densest-kPoints-Center*, $O\left(\left(\frac{6k}{\epsilon}\right)^d n^2\right)$ for* Densest-kPoints-Median*, and $O\left(\left(\frac{3\sqrt{2k}}{\epsilon}\right)^d n^2\right)$ for* Densest-kPoints-Mean*.*

Similar sampling techniques as in [50] might be applied to reduce the complexity from n^2 to n. However, when d is large, the grid-based algorithms discussed above does not scale well. In particular, when we transform the BQP problem into a Densest-kPoints-Mean problem, the d is equal to n. In the next subsection, we will show how the core-set techniques can be used to get faster algorithms, e.g. with running time $O(n^{1/\epsilon})$ or $O(2^{1/\epsilon}n)$.

5.2.2 Faster $(1+\epsilon)$-approximation algorithms using core-sets

The core-set techniques can be directly applied in Densest-kPoints problems to get faster algorithms. The main idea is: Consider an optimal solution C^* (of size k) for a set of n points P, if C^* has a core-set S, i.e. the center of S, c_S, is an approximation to the center of C^*, c_{C^*}, then we can first find c_S for some S, and find the nearest k points to c_S in P. It can be shown this is an $(1+\epsilon)$-approximation. The time complexity depends on how S is obtained (through enumeration or through sampling), and how fast c_S can be computed.

In the following part, we use $\mathrm{cost}(C, x)$ to denote $\mathrm{mean}(C, x)$, $\mathrm{cen}(C, x)$, or $\mathrm{med}(C, x)$. Recall the Densest-kPoints problem is: given a set of n points P in $\Re^d$, find $C \subseteq P$ of size k and $x \in \Re^d$ s.t. $\mathrm{cost}(C, x)$ is minimized.

Fixing a point set S, let c_S be the point $x \in \Re^d$ that minimizes $\mathrm{cost}(S, x)$.

Assumption 5.2.3. *For any $\epsilon < 1$, there exists a constant $m(\epsilon)$ s.t. for any point set X in $\Re^d$, there exists a subset (core-set) $S \subseteq X$ of size $m(\epsilon)$: (i)* *we can compute c_S in time $f(\epsilon)$; (ii)* $\text{cost}(X, c_S) \leq (1+\epsilon)\text{cost}(X, c_X)$.

Recall the core-set techniques surveyed in Section 5.1, the above assumption is valid the for Means/Center version of $\text{cost}(\cdot,\cdot)$. For the Median version, a slightly weaker assumption holds: we can find $g(\epsilon, |X|)$ candidates in $\Re^d$ based on S without knowing P, s.t. one of these candidates, denoted by c'_S, satisfies $\text{cost}(X, c_S) \leq (1+\epsilon)\text{cost}(X, c_X)$.

Given n points P and integer k, suppose C^* of size k is the optimal solution to the Densest-kPoints problem. From the above assumption, there exists a core-set S of size $m(\epsilon)$ for C^*. Therefore, our algorithm guess S by enumerating all $m(\epsilon)$-subsets of P, and for each possible subset S, we construct a solution by finding c_S and k nearest points to c_S in P. Finally, we pick the best solution. The running time is $O(n^{m(\epsilon)} \cdot f(\epsilon) \cdot nk)$.

For the Median version, we also need to try every candidate for a possible set S, so the running time is $O(n^{m(\epsilon)} \cdot g(\epsilon, k) \cdot nk)$.

In the Means/Center version, we have $m(\epsilon) = \frac{1}{\epsilon}$, and $f(\epsilon) = \text{poly}(\frac{1}{\epsilon})$, so the running time is $n^{O(1/\epsilon)}$. In the Median version, we have $m(\epsilon) = O(\frac{1}{\epsilon^4})$ and $g(\epsilon, k) = O(2^{O(1/\epsilon^4)} \log k)$, so the running time is $n^{O(1/\epsilon^4)}$.

Theorem 5.2.4. *There is an $(1+\epsilon)$-approximation algorithm with running time $n^{O(1/\epsilon)}$ for the* Densest-kPoints-Mean/Densest-kPoints-Center *problem, or with running time $n^{O(1/\epsilon^4)}$ for the* Densest-kPoints-Median *problem.*

Assumption 5.2.5. *For any $\epsilon < 1$, there exists a constant $m(\epsilon)$ s.t. for any point set X in $\Re^d$, from an uniformly random sample multiset (core-set) $S \subseteq X$ of size $m(\epsilon)$: (i) we can compute c_S in time $f(\epsilon)$; (ii)* $\text{cost}(X, c_S) \leq (1+\epsilon)\text{cost}(X, c_X)$ *holds with high probability.*

As is discussed in Section 5.1, this assumption is valid for the Means version. For the Median version, a slightly weaker assumption holds, but the algorithm and the analysis are similar. So we will focus on the Means version in the rest part.

Given n points P and integer k, suppose C^* of size k is the optimal solution to Densest-kPoints problem. If $k \geq \lambda n$, then, with high probability, a random sample multiset $S' \subseteq P$ of size $\frac{2}{\lambda \cdot \epsilon}$ contains at least $\frac{1}{\epsilon}$ points from C^* with high probability (from the Makov inequality). Let $S \subseteq S'$ denote these $\frac{1}{\epsilon}$ points (i.e. $S \subseteq C^*$ also). From Assumption 5.2.5 (note $m(\epsilon) = \frac{1}{\epsilon}$ for the

Means version), with high probability, $\text{cost}(C^*, c_S) \leq (1+\epsilon)\text{cost}(C^*, c_{C^*})$. So the set of the k nearest points to c_S is an $(1+\epsilon)$-approximation for DENSEST-kPOINTS-MEAN with high probability. The only problem is that we do not know the $\frac{1}{\epsilon}$-subset $S \subseteq S'$ that satisfies $S \subseteq C^*$, so we enumerate all the $\frac{1}{\epsilon}$-subsets of S' to guess S.

Our algorithm works as follows. Sample a multiset S' of size $\frac{2}{\lambda \cdot \epsilon}$ from P. Enumerate all $\frac{1}{\epsilon}$-subsets S of S'. For each S, compute c_S and find the k nearest points to c_S in P—these k points form a candidate solution. Output the best among all candidate solutions.

Theorem 5.2.6. *If $k \geq \lambda n$ for some constant λ, there is a randomized $(1+\epsilon)$-approximation algorithm with running time $O(2^{O(1/\epsilon)}nk)$ for the* DENSEST-kPOINTS-MEAN *problem, or with running time $O(2^{O(1/\epsilon^4)}nk)$ for the* DENSEST-kPOINTS-MEDIAN *problem.*

Using similar analysis and the Chernoff bounds, we can relax the requirement on k a bit. It turns out that the problem is hard when k is "small" but not constant ($k \in \omega(1) \cap o(\log n)$).

Theorem 5.2.7. *If $k = \Omega(\frac{n \log m}{m})$, there is a randomized $(1 + \frac{1}{\log m})$-approximation algorithm with running time $O(2^{O(m)}nk)$ for the* DENSEST-kPOINTS-MEAN *problem, or with running time $O(2^{O(m^4)}nk)$ for the* DENSEST-kPOINTS-MEDIAN *problem.*

CHAPTER 6

CONCLUSION AND FUTURE WORKS

For the first part of the thesis, we review the quadratic problem and its applications. Existing algorithms on quadratic problem in literature, especially linear relaxation and SDP relaxation method, have been discussed.

In the second part we derive a new second-order conic optimization problem by recasting the convex quadratic programming relaxation of BQPs. Such a modification allows us to incorporate the classical graph modeling techniques into the relaxation model to enhance the relaxation model. Numerical performance from the new model is promising in some cases.

The third part propose to use convex QP as a geometric embedding tool to reformulate the BQP problem as a specific clustering problem. Our new model not only provides a new approach to efficiently solve BQP problem, but also opens new avenue for tackling clustering problem. A 2-approximation algorithm to the new clustering problem is presented. An efficient heuristics is introduced for the original BQP based on its equivalent clustering model. Numerical experiments illustrates that the proposed heuristics can locate the global optimum or a solution very close to the global optimum quickly.

In the forth part, we propose the DENSEST-KPOINTS-MEAN problems and approximation algorithms for it. We also did a survey on core-set techniques for clustering, and then use the core-set techniques to obtain $(1+\epsilon)$-approximation algorithms for the DENSEST-KPOINTS problems (which are faster than the grid-based $(1+\epsilon)$-approximation algorithms). In particular, we propose a PTAS for the BQP problem using core-sets.

There are some ways for further improvement. For example the interrelation between certain classes of BQPs and clustering problems can not only help to better understand the theoretical properties of the underlying BQPs, but also lead to the development of efficient resolution techniques. In this paper, we focus only on two special classes of BQPs. It will be interesting to investigate whether the approaches proposed in this paper can be extended to other classes of BQPs or equivalent discrete optimization problems.

Some interesting future work also include:

Is there any faster algorithm for the Densest-kPoints problems when $k \in \omega(1) \cap o(\log n)$? Either when k is constant (then use the naive $O(n^k)$ algorithm) or when k is large (use the core-set techniques), the problem is easy. It is interesting whether there is faster $(1+\epsilon)$-approximation algorithm when k is in the middle.

It is interesting whether the core-set techniques can be used in other clustering problems, like clustering uncertain data [51, 52] or coclustering problem [53], to obtain provably good results. We have found core-sets techniques can be applied in the MinSumRadii problem (minimize the sum of radii of clusters) to obtain $(1 + \epsilon)$-approximation (the algorithm is identical to the one for k-Center clustering, introduced in Section 5.1.1).

REFERENCES

[1] M. J. F. Barahona and G. Reinelt, “Experiments in quadratic 0-1 programming.” *Mathematical Programming*, no. 44, pp. 127–137, 1989.

[2] A. Phillips and J. Rosen, “A quadratic assignment formulation of the molecular conformation problem.” *J. Global Optimization*, no. 4, pp. 229–241, 1994.

[3] P. Chardaire and A. Sutter, “A decomposition method for quadratic zero-one programming.” *Management Science*, no. 41, pp. 704–712, 1995.

[4] J. Y. R.D. McBride, “An implicit enumeration algorithm for quadratic integer programming.” *Management Science*, no. 26(3), pp. 282–296, 1980.

[5] T. Q. X. Geng, T. Liu and H. Li, “Feature selection for ranking.” *SIGIR*, 2007.

[6] U. Feige and M. Langberg, “Approximation algorithms for maximization problems arising in graph partitioning.” *J. Algorithms*, no. 41, pp. 174–211, 2001.

[7] G. K. U. Feige and D. Peleg, “The dense k-subgraph problem.” *Algorithmica*, no. 29, pp. 410–421, 2001.

[8] Y. Y. Q. Han and J. Zhang, “An improved rounded method and semidefinite programming relaxation for graph partition.” *Mathematical Programming*, no. 92(3), pp. 509–535, 2002.

[9] C. Helmberg and F.Rendl, “Solving quadratic (0,1)-problem by semidefinite programming and cutting planes.” *Mathematical Programming*, no. 82, pp. 291–315, 1998.

[10] M. Goemans and S. Williamson, “Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming.” *Journal of the ACM*, no. 42(6), pp. 1115–1145, 1995.

[11] M. X. Goemans and F. Rendl, "Semidefinite programming in combinatorial optimization." *Mathematical Programming*, no. 79, pp. 143–161, 1997.

[12] W. Adams and H. Sherali, "A tight linearization and an algorithm for zero-one quadrtic programming problems." *Management Science*, no. 32, pp. 1274 – 1290, 1986.

[13] W. Adams and H. Sherali, "A hierarchy of relaxations between continuous and convex hull representations for zero-one programming problems." *SIAM J. Discrete Applied mathematics*, no. 52, pp. 83 – 106, 1990.

[14] L. Lovász and A. Schrijver, "Cones of matrices and set functions and 0-1 optimization." *SIAM J. on Optimization*, no. 1, pp. 166–190, 1991.

[15] J. Lasserre, "An explicit exact sdp relaxation for nonlinear 0-1 programs." *SIAM J. Optimization*, no. 4, pp. 229–241, 2002.

[16] L. Lovász, "On the shannon capacity of a graph." *IEEE Transactions on Information Theory*, no. IT-25 (1), pp. 1–7, 1994.

[17] N. Shor, "Quadratic optimization problems." *Izv. USSR Technical Cybernetics (Moscow)*, no. 1, pp. 128–139, 1987.

[18] F. Alizadeh, "Interior point methods in semidefinite programming with applications to combinatorial optimization." *SIAM Journal on Optimization*, no. 5, pp. 13–51, 1995.

[19] P. Hammer and A. Rubin, "Some remarks on quadratic programming with 0-1 variables." *R.I.R.O.*, no. 3, pp. 67–69, 1970.

[20] F. Korner, "A tight bound for the boolean quadratic optimization problem and its use in a branch and bound algorithm." *Optimization*, no. 19, pp. 711–721, 1988.

[21] S. Polijk and H. Wolkowicz, "Convex relaxation of (0,1)-quadratic programming." *Mathematics of Operations Research*, no. 3, pp. 550–561, 1995.

[22] V. V. Vazirani, Ed., *Approximation Algorithms.* Berlin, Germany: Springer, 2003.

[23] P. H. P.L. Hammer and B.Simeone, "Roof duality, complementation and persistency in quadratic 0-1 optimization." *Mathematical Programming*, no. 28, pp. 121–155, 1984.

[24] P. Pardalos and G. Rodgers, “A branch and bound algorithm for the maximum clique problem.” *Computers & Operations Research*, no. 19, pp. 363–375, 1992.

[25] B. Mohar and S. Poljak, “Eigenvalues and the max-cut problem,” *Czechoslovak Mathematical Journal*, vol. 40, no. 115, pp. 343–352, 1990.

[26] S. Poljak and F. Rendl, “Nonpolyhedral relaxations of graph bisection problem,” *SIAM Journal on Optimization*, vol. 5, pp. 467–487, 1995.

[27] Y. Nesterov and A. Nemirovski, Eds., *Interior-point polynomial algorithms in convex programming.* Philadelphia: SIAM Studies in Applied Mathematics, 1995.

[28] M. Laurent and S. Poljak, “On a positive semidefinite relaxation of cut polytope,” *Linear Algebra and Applications*, vol. 223/224, pp. 439–461, 1995.

[29] F. R. C. Helmberg and R. Weismantel, “A semidefinite programming approach to the quadratic knapsack problem.” *J. of Combinatorial Optimization*, no. 4, pp. 197–215, 2000.

[30] S. P. M. Laurent and F. Rendl, “Connections between semidefinite relaxations of the max-cut and stable set problems,” *Mathematical Programming*, vol. 77, pp. 225–246, 1997.

[31] J. McQueen, “Some methods for classification and analysis of multivariate observations.” *Computer and Chemistry*, no. 4, pp. 257–272, 1967.

[32] M. I. N. K. S. Hasegawa, H. Imai and J. Nakano, “Efficient algorithms for variance-based k-clustering,” in *Proc. First Pacific Conf. Comput. Graphics Appl.*, 1993, pp. 75–89.

[33] N. Sahinidis, “Baron: Branch and reduce optimization navigator.” *http://archimedes.scs.uiuc. edu/baron/manuse.pdf*, 2000.

[34] M. M. J. Czyzyk and J. Moré, “The neos server.” *IEEE J. Comput. Sci. Eng*, no. 5, pp. 68–75, 1998.

[35] H. T. Y. Asahiro, K. Iwama and T. Tokuyama, “Greedily finding a dense subgraph.” *Journal of Algorithms*, no. 34, pp. 203–221, 2000.

[36] W. Yu, “Experimental study of approximation algorithms for the densest k-subgraph problem.” *Master Thesis, Department of Computing and Software, McMaster University,*, December, 2003.

[37] M. Inaba, N. Katoh, and H. Imai, "Applications of weighted voronoi diagrams and randomization to variance-based k-clustering (extended abstract)," in *Symposium on Computational Geometry*, 1994, pp. 332–339.

[38] M. Badoiu, S. Har-Peled, and P. Indyk, "Approximate clustering via core-sets," in *STOC*, 2002, pp. 250–257.

[39] S. Har-Peled and S. Mazumdar, "On coresets for k-means and k-median clustering," in *STOC*, 2004, pp. 291–300.

[40] A. Kumar, Y. Sabharwal, and S. Sen, "A simple linear time $(1+\epsilon)$-approximation algorithm for k-means clustering in any dimensions," in *FOCS*, 2004, pp. 454–462.

[41] M. R. Ackermann, J. Blömer, and C. Sohler, "Clustering for metric and non-metric distance measures," in *SODA*, 2008, pp. 799–808.

[42] M. R. Ackermann and J. Blömer, "Coresets and approximate clustering for bregman divergences," in *SODA*, 2009, pp. 1088–1097.

[43] S. Har-Peled and K. R. Varadarajan, "Projective clustering in high dimensions using core-sets," in *Symposium on Computational Geometry*, 2002, pp. 312–318.

[44] S. Har-Peled and A. Kushal, "Smaller coresets for k-median and k-means clustering," in *Symposium on Computational Geometry*, 2005, pp. 126–134.

[45] M. Badoiu and K. L. Clarkson, "Smaller core-sets for balls," in *SODA*, 2003, pp. 801–802.

[46] K. Chen, "On k-median clustering in high dimensions," in *SODA*, 2006, pp. 1177–1185.

[47] G. Frahling and C. Sohler, "A fast k-means implementation using coresets," in *Symposium on Computational Geometry*, 2006, pp. 135–143.

[48] D. Feldman, M. Monemizadeh, and C. Sohler, "A ptas for k-means clustering based on weak coresets," in *Symposium on Computational Geometry*, 2007, pp. 11–18.

[49] A. Goel, P. Indyk, and K. R. Varadarajan, "Reductions among high dimensional proximity problems," in *SODA*, 2001, pp. 769–778.

[50] S. Har-Peled and S. Mazumdar, "Fast algorithms for computing the smallest k-enclosing circle," *Algorithmica*, vol. 41, no. 3, pp. 147–157, 2005.

[51] G. Cormode and A. McGregor, “Approximation algorithms for clustering uncertain data,” in *PODS*, 2008, pp. 191–200.

[52] S. Guha and K. Munagala, “Exceeding expectations and clustering uncertain data,” in *PODS*, 2009, to appear.

[53] A. Anagnostopoulos, A. Dasgupta, and R. Kumar, “Approximation algorithms for co-clustering,” in *PODS*, 2008, pp. 201–210.